รายงานสหกิจศึกษาฉบับสมบูรณ์

พัฒนาระบบ Wongnai Delivery และ RMS

Improving Wongnai Delivery and RMS

นางสาววริษา สายเสมา

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

ชื่อโครงงานสหกิจศึกษา พัฒนาระบบ Wongnai Delivery และ RMS
ชื่อ-สกุล นักศึกษา นางสาววริษา สายเสมา
คณะ วิศวกรรมศาสตร์ **ภาควิชา** วิศวกรรมคอมพิวเตอร์
ชื่อ-สกุล อาจารย์นิเทศ อาจารย์จิระศักดิ์ สิทธิกร
ชื่อ-สกุล ผู้นิเทศงาน นายธนพล เนรัญชร
ชื่อสถานประกอบการ บริษัท วงใน มีเดีย จำกัด

บทคัดย่อ

ระบบ Wongnai Delivery และ RMS เป็นระบบสำหรับการดูและค้นหาข้อมูลของร้านอาหารที่มีบริการเดลิเวอรี่ การสั่งอาหารผ่านหน้าเว็บและระบบสำหรับการจัดการข้อมูลร้านอาหาร คำสั่งซื้อหรือระบบเดลิเวอรี่ของร้าน ในขณะที่ความต้องการเปลี่ยนแปลงอยู่ตลอดเวลา จึงทำให้ต้องมีการแก้ไข ปรับปรุงและพัฒนาระบบอยู่เสมอ ทั้งการเพิ่มฟีเจอร์ใหม่ๆ หรือการแก้ไขการทำงานของระบบให้ตอบรับกับความต้องการมากขึ้น โดยในรายงานเล่มนี้จะยกตัวอย่างงานมาอธิบายโดยละเอียดเพียงหนึ่งชิ้น

เนื่องจากปัจจุบันนี้มีร้านอาหารซื้อบริการของบริษัทมากขึ้นทำให้การออกใบแจ้งหนี้แต่ละครั้งนั้นกินเวลาและกำลังคนเป็นจำนวนมาก จึงมีการพัฒนาระบบออกใบแจ้งหนี้อัตโนมัติขึ้นมาเพื่อออกใบแจ้งหนี้ให้ร้านอาหารในทุกๆ เดือน แต่ด้วยข้อมูลที่จำเป็นต่อการออกใบแจ้งหนี้นั้นถูกเก็บอยู่ใน 2 ระบบที่ไม่มีส่วนเกี่ยวข้องกันคือระบบ RMS และ ERP จึงต้องมีการสร้างระบบใหม่ในการเชื่อมต่อ 2 ระบบข้างต้นเข้าด้วยกัน โดยระบบใหม่นี้จะทำหน้าที่เชื่อมต่อข้อมูล สร้างใบแจ้งหนี้และส่งให้ร้านอาหารโดยอัตโนมัติ เพื่อความรวดเร็วและความสะดวกสบายเพราะระบบจะช่วยลดเวลาและกำลังคนในการทำงาน

การพัฒนาและปรับปรุงระบบ Wongnai Delivery และ RMS นี้ทั้งหมดก็เพื่อให้ระบบทำงานได้อย่างถูกต้อง มีคุณภาพและประสิทธิภาพ สร้างรายได้และประโยชน์เชิงธุรกิจให้แก่บริษัท และที่สำคัญคือระบบต้องสามารถตอบสนองต่อความต้องการและความคาดหวังของผู้ใช้งานได้ด้วย

คำสำคัญ: Spring Framework, MySQL, ORM, Kubernetes, Docker, Maven, Rest API, React

Co-operative Title: Improving Wongnai Delivery and RMS

Student Intern Name: Warisa Saisema

Faculty: Engineering **Department:** Computer Engineering

Advisor Name: Jirasak Sittigorn

Mentor Name: Tanapol Nearunchorn

Company: Wongnai Media Co,. Ltd

ABSTRACT

Wongnai Delivery and RMS is a system for searching for information of restaurant which provided food delivery services or web ordering and a system for managing restaurant's information, orders or delivery services. Nowadays, human's needs are change in every second, so systems need to be improved all the time. Either adding new features or modifying the operation to respond the needs. In this report, only one example will be explained in detail.

since the restaurants buy for our service more than before, the invoice process takes a lot of time and manpower too. So, we develop the auto invoice system to automatically create a monthly invoice for restaurants. The problem is the information necessary for an invoice is kept in separated two systems, RMS and ERP. For this reason, the new system will be a connector between two systems and create an invoice automatically. In order to reduce time and manpower and increase speed and convenience in work.

Improving Wongnai Delivery and RMS will make the systems to work correctly with the best quality and performance, bring benefit and revenue to the company, and the most important thing is to respond to the needs and expectations of users.

Keywords: Spring Framework, MySQL, ORM, Kubernetes, Docker, Maven, Rest API, React

กิตติกรรมประกาศ

ในระยะเวลาตั้งแต่วันที่ 4 มิถุนายน - 22 พฤศจิกายน 2562 ข้าพเจ้าได้มีโอกาสเข้าฝึกงานและทำสหกิจศึกษาในตำแหน่ง Backend Developer ทีม Delivery และ RS ของวงใน รวมเป็นระยะเวลาเกือบ 6 เดือนที่ได้รับประสบการณ์ใหม่ๆ ในทุกๆ วัน ได้เรียนรู้ขั้นตอนในการทำงานจริงของบริษัท ได้รู้จักและลองใช้เครื่องมือและเทคโนโลยีใหม่ๆ มากมาย ได้สนุกกับทำงานและการแก้ปัญหาที่พบเจอในแต่ละวัน ได้ลองผิดลองถูกกับอะไรที่ไม่เคยทำ นับเป็นประสบการณ์ดีๆ ที่ไม่เคยได้รับจากที่ไหนมาก่อน

โดยทั้งหมดนี้จะเกิดขึ้นไม่ได้เลยหากขาดการสนับสนุนดีๆ จากทั้งบริษัทและบุคลากรเหล่านี้ ขอขอบคุณพี่บอยและพี่ชาร์ปที่ให้ผ่านสัมภาษณ์ได้โอกาสเข้ามาฝึกงานที่นี่ ขอขอบคุณพี่ป่านและพี่เอิร์ธที่คอยมอบหมายงานต่างๆ ให้ทำอยู่เสมอ ขอขอบคุณพี่อาร์ตและพี่ปิงที่คอยดูแลคอยสอนและแนะนำการทำงาน อธิบายในทุกข้อสงสัยและช่วยแก้ทุกปัญหาที่เกิดขึ้น ขอขอบคุณเพื่อนๆ ฝึกงานและพี่ๆ ในทีมทุกคนที่ให้การต้อนรับ ช่วยเหลือและสนับสนุนเป็นอย่างดี ขอขอบคุณพี่ๆ ทีม People ที่หาข้าวกลางวันอร่อยๆ ให้กินเสมอ รวมถึงพี่ๆ ที่อาจไม่ได้กล่าวถึงในที่นี้ ขอขอบคุณทุกการต้อนรับ ความช่วยเหลือและคำแนะนำต่างๆ ของทุกคนที่มีส่วนช่วยให้การฝึกงานและสหกิจศึกษาครั้งนี้เป็นไปได้ด้วยดี ขอขอบพระคุณทุกท่านมา ณ ที่นี้

นางสาววริษา สายเสมา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

บริษัท วงใน มีเดีย จำกัดได้มีการพัฒนาระบบเดลิเวอรี่บน platform ของตัวเองโดยร่วมมือกับ LINEMAN สำหรับการสั่งอาหารผ่านทาง website หรือ mobile application และยังมีระบบ RMS หรือ Restaurant Management System ที่เป็น application สำหรับร้านอาหารให้ร้านสามารถการจัดการข้อมูลของร้านและ order ต่างๆ เพื่อให้ร้านสามารถใช้ระบบเดลิเวอรี่ง่ายและสะดวกยิ่งขึ้น แต่ด้วยความต้องการของมนุษย์ที่เกิดการเปลี่ยนแปลงอยู่เสมอในปัจจุบัน ส่งผลให้ต้องมีการพัฒนาและปรับปรุงแก้ไขการทำงานของระบบให้ตอบรับการความต้องการที่เปลี่ยนแปลงไปอยู่เสมอ ทั้งการเพิ่มบริการหรือ feature ใหม่ๆ รวมไปถึงการแก้ไขให้ระบบทำงานได้ตรงกับความต้องการยิ่งขึ้น จึงเป็นที่มาของ project นี้ แต่เนื่องจากการพัฒนาระบบนั้นได้มีการแก้ไขและปรับปรุงแบบย่อยๆ กระจายไปในหลายๆ ส่วน ในรายงานเล่มนี้จึงจะยกมาตัวอย่างเพื่ออธิบายเพียงหนึ่งชิ้น คือ ระบบออกใบแจ้งหนี้อัตโนมัติ

ระบบออกใบแจ้งหนี้อัตโนมัติ คือระบบที่ถูกพัฒนาขึ้นเพื่อออกใบแจ้งหนี้ให้ร้านอาหารในทุกๆ เดือน เพื่อแจ้งข้อมูลของค่าบริการ GP ที่ร้านอาหารได้มีการทำสัญญากับทางบริษัทเอาไว้ โดยค่าบริการ GP คือส่วนแบ่งตามสัญญาจากยอดขายของร้าน ที่ระบบได้เพิ่มสิทธิพิเศษต่างๆ ไว้ให้ เช่น การจัดร้านให้ขึ้นมาอันดับต้นๆ ในการค้นหาหรือส่วนลดค่าจัดส่งต่างๆ เพื่อกระตุ้นยอดขายของร้านให้เพิ่มขึ้น โดยข้อมูลที่จำเป็นในการออกใบแจ้งหนี้ต้องมีชื่อและที่อยู่ในการออกใบแจ้ง ข้อมูลยอดขายทั้งหมดของร้านในหนึ่งเดือนและค่าบริการ GP ที่คำนวณจากยอดขายของร้าน ซึ่งข้อมูลทั้งหมดนี้ถูกเก็บอยู่ใน 2 ระบบที่ไม่มีความเชื่อมโยงกัน คือระบบ RMS และ ERP จึงต้องมีการแก้ไขและปรับเปลี่ยนโครงสร้างของระบบให้สามารถเชื่อมต่อและทำงานร่วมกันได้ เพื่อลดแรงงานคนที่ต้องมาค้นหาและบันทึกข้อมูลทีละร้านและเพิ่มประสิทธิภาพของระบบให้สามารถทำงานได้ดียิ่งขึ้น

1.2 วัตถุประสงค์ของการวิจัย

1.2.1 เพื่อพัฒนา feature ใหม่ๆ ให้ตอบรับกับความต้องการ

1.2.2 เพื่อพัฒนาระบบให้มีประสิทธิภาพมากขึ้น

1.2.3 เพื่อปรับปรุงการทำงานของระบบให้ดีขึ้น

1.2.4 เพื่อแก้ไขระบบให้สามารถทำงานได้อย่างถูกต้อง

1.3 ขอบเขตของการวิจัย

1.3.1 พัฒนาระบบเดลิเวอรี่และ RMS ให้ตอบรับกับความต้องการใหม่ๆ

1.3.2 แก้ไขการทำงานของระบบเดลิเวอรี่และ RMS ให้ทำงานได้อย่างถูกต้อง

1.3.3 ปรับปรุงการทำงานของระบบเดลิเวอรี่และ RMS ให้มีประสิทธิภาพยิ่งขึ้น

1.4 วิธีดำเนินการวิจัย

1.4.1 ศึกษาเครื่องมือ เทคโนโลยีและกระบวนการทำงานของบริษัท

1.4.2 ศึกษาและทำความเข้าใจโครงสร้างของระบบเดลิเวอรี่และ RMS

1.4.3 วิเคราะห์ปัญหาหรือความต้องการที่ได้รับมอบหมายมา

1.4.4 คิดและวางแผนวิธีแก้ปัญหาที่เหมาะสมที่สุดในช่วงเวลานั้น

1.4.5 พัฒนาหรือแก้ไขระบบตามวิธีที่เลือกไว้

1.4.6 ทดสอบการทำงานของระบบว่าทำงานได้อย่างถูกต้อง

1.4.7 ทำการ deploy ขึ้นใช้งานจริงบน production

1.4.8 ปรับปรุงแก้ไขการทำงานหากระบบเกิดปัญหา

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้เรียนรู้โครงสร้างและกระบวนการทำงานในบริษัท

1.5.2 ได้เรียนรู้เทคโนโลยีและเครื่องมือใหม่ๆ ที่มีส่วนช่วยในการพัฒนาระบบ

1.5.3 ได้ศึกษาโครงสร้างและเรียนรู้การจัดการกับระบบที่มีขนาดใหญ่

1.5.4 ได้ฝึกการคิดวิเคราะห์และวางแผนในการแก้ปัญหา

1.5.5 ได้ฝึกการบริหารจัดการเวลาในการทำงานได้อย่างเหมาะสม

1.5.6 ได้รับแนวคิดและมุมมองใหม่ๆ ที่เป็นเชิงธุรกิจมากขึ้น

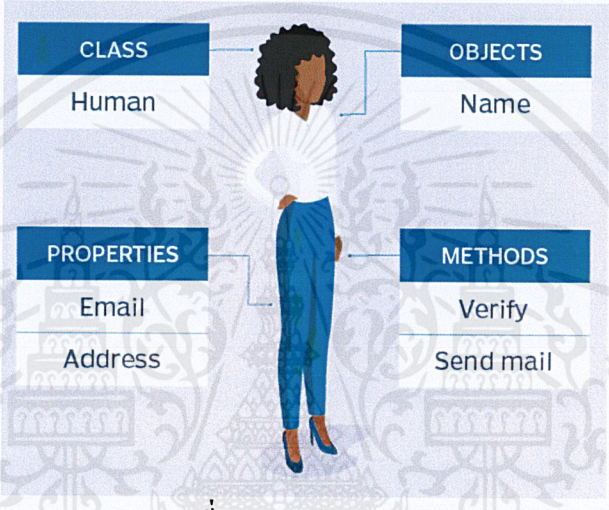
1.5.7 ได้รับประสบการณ์ทำงานจริงที่สามารถนำไปต่อยอดในอนาคตได้

บทที่ 2
แนวคิด ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ในการพัฒนาระบบ Wongnai Delivery และ RMS นั้นมีหลักการ เครื่องมือและทฤษฎีต่างๆ ที่จำเป็นต้องรู้หลักๆ ดังนี้

2.1.1 Object Oriented Programming



ภาพที่ 2.1 แนวคิดของ OOP

Object Oriented Programming หรือ OOP เป็นวิธีการเขียนโปรแกรมโดยอาศัยแนวคิดว่าวัตถุชิ้นหนึ่งมีความสามารถในการปกป้องข้อมูลและการสืบทอดคุณสมบัติของตน โดย OOP ใช้วิธีเลียนแบบวิธีคิดของวัตถุในชีวิตจริง โดยระบุสถานะและพฤติกรรมให้วัตถุในระบบ โดยวัตถุ (object) จะเก็บสถานะใน field และเปิดเผยพฤติกรรมผ่าน method โดยที่ method จะดำเนินการกับสถานะของ object นั้น และเป็นกลไกสำคัญในการสื่อสารระหว่าง object หัวใจสำคัญของ OOP นั้น มีอยู่ 4 ข้อ ดังนี้

- Abstraction เป็นหลักการนามธรรมของ object ซึ่งเป็นกระบวนการในการเอาส่วน concrete ของ class หรือส่วนที่มีการ implementation ของ object ออกมา โดยรักษาลักษณะร่วมกัน

- Encapsulation เป็นหลักการการห่อหุ้มสถานะหรือข้อมูลของ object จากภายนอกให้ไม่สามารถเข้าถึงสถานะของ object นั้นๆ ได้ การเข้าถึงจะต้องกระทำผ่านทาง method เท่านั้น ซึ่งการจำกัดการเข้าถึงข้อมูลก็แบ่งออกเป็นหลายระดับ เช่น private จำกัดการเข้าถึงอย่าง

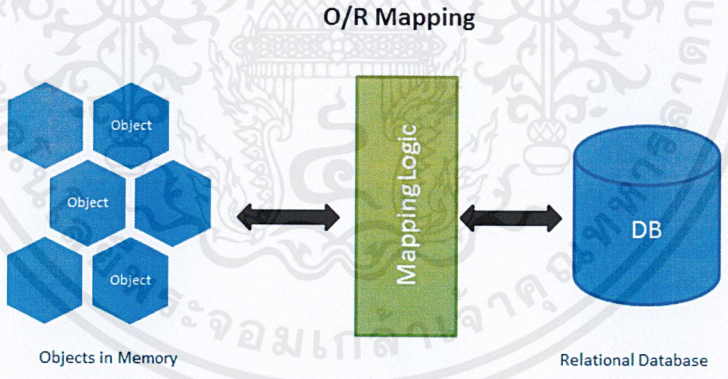
สิ้นเชิง, protected จำกัดการเข้าถึงจากภายนอกแต่ class ลูกสามารถเข้าถึงได้และ public ไม่จำกัดการเข้าถึงเลย

- Inheritance เป็นหลักการสืบทอดคุณสมบัติของ object ซึ่ง subclass ที่สืบทอดจาก superclass จะได้รับสถานะและพฤติกรรมของ superclass นั้นทั้งหมด และยังสามารถเปลี่ยนพฤติกรรมของบาง method ที่สืบทอดมาได้ อีกทั้งสามารถขยายความสามารถเพิ่มเติมได้ ทำให้เราสามารถนำ code ที่มีอยู่แล้วมาใช้ใหม่ได้

- Polymorphism เป็นหลักการพ้องรูปหรือการมีหลายรูปของ object คล้ายๆ กับหลักการของชีววิทยาที่สิ่งมีชีวิตสามารถมีหลายรูปแบบหรือหลายสายพันธุ์ โดย subclass สามารถกำหนดพฤติกรรมที่เป็นเอกลักษณ์ของตนเองได้และยังคงมีฟังก์ชันการทำงานแบบเดียวกันกับ superclass ได้

2.1.2 Object-Relational Mapping

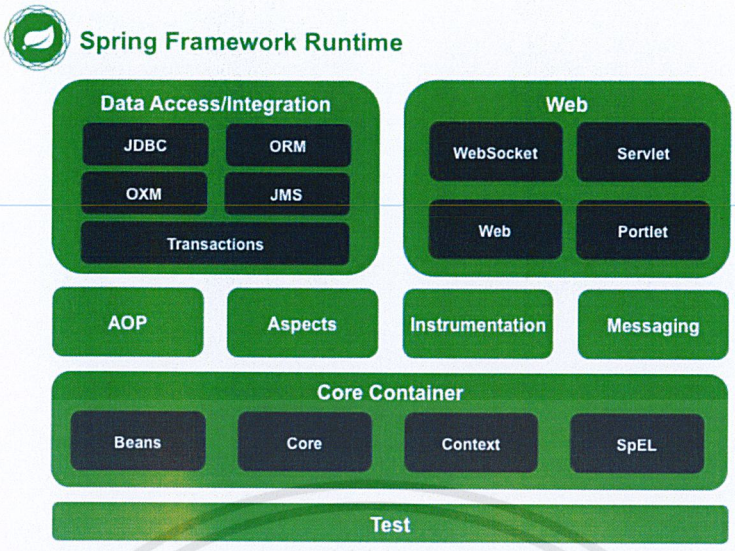
Object-Relational Mapping หรือ ORM เป็นเทคนิคการใช้งานฐานข้อมูลในรูปแบบของ object แทนการใช้ SQL โดยการจับคู่ระหว่างข้อมูลที่มีความสัมพันธ์ในฐานข้อมูล (Relational Database) ให้มาอยู่ในรูปแบบของ object ทำให้สามารถจัดการข้อมูลในฐานข้อมูลได้ด้วยการเขียนแบบ Object-Oriented Language



ภาพที่ 2.2 แนวคิดของ ORM

2.1.3 Spring Framework

Spring Framework เป็น open source framework ที่ได้รับความนิยมมากในปัจจุบัน เป็นตัวช่วยให้สามารถพัฒนาแอปพลิเคชัน Java enterprise ได้โดยง่าย มีความยืดหยุ่นในการสร้างสถาปัตยกรรมหลายรูปแบบขึ้นอยู่กับความต้องการของโปรแกรม ผู้ที่คิดค้นและพัฒนาขึ้นมาเป็นเวอร์ชั่นแรก คือ Rod Johnson โดยอยู่ภายใต้ลิขสิทธิ์ของ Apache License ใน Spring Framework มี module ต่างๆ ดังนี้



ภาพที่ 2.3 โครงสร้าง module ต่างๆ ของ Spring Framework

1) Data Access/Integration การเข้าถึงข้อมูลต่างๆ ประกอบไปด้วย

- JDBC Module ทำให้เชื่อมต่อ JDBC ได้โดยไม่ต้อง coding
- ORM Module ให้การบริการ Object-Relational Mapping APIs ซึ่งประกอบด้วย JPA, JDO, Hibernate และ iBatis
- OXM Module ให้บริการ Object/XML Mapping ซึ่งต้องใช้ใน JAXB, Castor, XMLBeans, JiBX และ XStream
- Java Messaging ให้บริการ JMS Module สำหรับรับและส่ง message
- Transaction Module สนับสนุน Transaction Management สำหรับ class ที่ implement special interfaces

2) Web layer ประกอบด้วย

- Web Module ให้บริการการทำงานพื้นฐานของเว็บ เช่น Multipart File-Upload และ Inversion of Control (IoC)
- Web-MVC Module ประกอบด้วย Model-View-Controller (MVC) ใช้สำหรับพัฒนา web application
- Web-Socket Module สนับสนุนงานที่ต้องใช้ socket ในการสื่อสารสองทางระหว่าง client และ server ใน web Application
- Web-Portlet Module สนับสนุนการใช้งาน MVC บน Portlet

3) Miscellaneous Module มีดังนี้

- AOP Module สนับสนุนแนวคิด Aspect-Oriented Programming (AOP) ที่เป็น paradigm หนึ่งของการเขียนโปรแกรมที่เพิ่มความเป็น module โดยอนุญาตให้แยก concern แบบตัดขวางได้

- Aspects Module สนับสนุน AspectJ ที่เป็นการเพิ่มเติมโครงสร้างภาษาใหม่เข้าไปในภาษา Java เพื่อให้สามารถเขียนโปรแกรมแบบ AOP ได้

- Instrumentation Module สนับสนุนการจัดการเกี่ยวกับ class

- Messaging Module สนับสนุน Text Oriented Message Protocol (STOMP) เช่น webSocket

- Test Module สนับสนุนการทดสอบ Spring Components ด้วย JUnit หรือ TestNG Frameworks

4) Core Container ประกอบด้วย

- Core Module เป็นองค์ประกอบพื้นฐานของ framework รวมไปถึง IoC และ Dependency Injection

- Bean Module ให้บริการ BeanFactory ในรูปแบบ factory pattern

- Context Module ถูกสร้างโดย Core และ Beans ใช้เป็นตัวกลางในการเข้าถึง object ต่างๆ และใช้ในการกำหนดค่าต่างๆให้ object

- SpEL Module ให้บริการ Expression Language สำหรับการ query และจัดการกับ object

จากโครงสร้างของ Spring Framework ข้างต้นนั้น ที่นิยมใช้ในการพัฒนา web application ในปัจจุบันคือ Spring MVC ซึ่งเป็นหนึ่งในหลายๆ module ของ framework โดย Spring MVC คือ framework ในการสร้างเว็บที่รองรับแนวคิดแบบ MVC (Model-View-Controller)

โดยหลักการทำงานของ Spring MVC จะออกแบบให้การทำงานทุกอย่างขึ้นอยู่กับ Servlet ที่ชื่อว่า DispatcherServlet มีการออกแบบโครงสร้างการเก็บชิ้นส่วนต่างๆ ของเว็บ เช่น ไฟล์ html, ไฟล์ jsp หรือ ไฟล์ Script ต่างๆ ไว้อย่างชัดเจน มีการจัดเก็บโครงสร้างส่วน controller ไว้ภายในโฟลเดอร์ src การจัดโครงสร้างให้เป็นระบบทำให้สะดวกต่อการค้นหา แก้ไข เพิ่มเติมหรือลบทิ้ง

นอกจากนี้ Spring MVC ก็มีการกำหนด form การเขียนเว็บในแต่ละส่วน ไม่ว่าจะเป็นส่วน request, respond หรือ controller ต่างๆ ทำให้การเข้ามาแก้ไขไฟล์ก็สามารถเข้าใจได้ง่ายและยังมีการใช้ library ต่างๆ ที่เป็นตัวช่วยในการเขียนโปรแกรม เพื่อให้ง่ายต่อการทำงานของผู้พัฒนาอีกด้วย โดยหลักการทำงานคร่าวๆของ Spring MVC มีขั้นตอนต่างๆ ดังนี้

1. client ส่ง request ไปยัง web container ในรูปแบบของ HTTP request

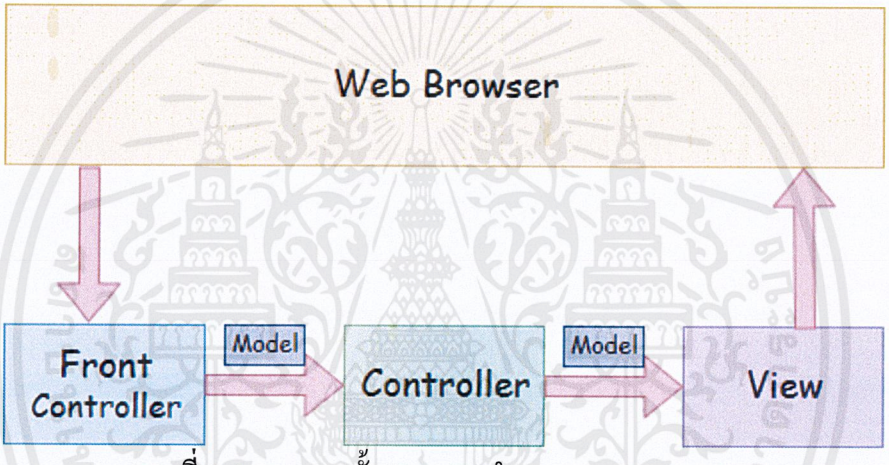
2.front controller หรือ DispatcherServlet รับ request ที่เข้ามาและค้นหา controller ที่เหมาะสมกับ request

3.DispatcherServlet ส่งข้อมูลของ request ไปให้ controller ตัวอื่นทำงานต่อโดยพิจารณาจากค่าที่ handler mappings ไว้

4.controller ทำงานตาม request ที่ส่งเข้ามาและส่งผลลัพธ์ออกมาเป็น model ไปยัง หน้า view ด้วย ModelAndView instance โดยผ่านตัว front controller

5.front controller จะทำหน้าที่จัดการปัญหาที่อาจเกิดขึ้นในการแสดงผลหน้า view โดยอาศัยตัว View Resolver

6.ทำการเลือก view ที่จะส่งกลับไปยัง client



ภาพที่ 2.4 แผนภาพขั้นตอนการทำงานของ Spring MVC

นอกจากนี้ Spring Framework ก็มีอีก project หนึ่งที่น่าสนใจและใช้กันอย่างกว้างขวาง เป็น project ที่ถูกพัฒนาต่อยอดออกมาเพื่อแก้ปัญหาความซับซ้อนและความยุ่งยากในการ configuration bean เพื่อให้ Spring container เรียกใช้งาน project นั้นคือ Spring Boot ที่สามารถทำให้พัฒนา application ได้อย่างรวดเร็ว โดยมีการทำ auto configuration ทำให้ไม่ต้องเสียเวลาไป config ทุกอย่างเองเหมือนก่อน ทำให้สามารถสร้าง stand-alone application ที่ export เป็น jar หรือจะทำเป็น war แล้วนำไป deploy ที่ application server ได้เหมือนเดิม

จะเห็นได้ว่า Spring Boot นั้นเป็นเครื่องมือที่ทำให้ผู้พัฒนาสามารถใช้งาน Spring Framework ได้ง่ายและรวดเร็วยิ่งขึ้น ด้วยการลดขั้นตอนการ configuration โดยการทำ auto configuration แต่ Spring Boot ก็มีการเพิ่ม annotation ใหม่ๆ เข้ามา ดังนั้นผู้พัฒนาจึงต้องไปศึกษาเรียนรู้วิธีการใช้งานเพิ่มเติมด้วย

2.1.4 Apache Maven

Maven เป็น Project Management Tools ที่ช่วยให้ Java developer พัฒนาระบบได้สะดวกและง่ายขึ้น สามารถใช้งานผ่าน plugin ของ IDE เช่น Eclipse NetBeans หรือ IntelliJ IDEA ได้เลย โดย Maven สามารถช่วยจัดการ library ต่างๆ ได้อย่างง่ายดาย เพียงแค่ config ค่าต่างๆ ลงในไฟล์ pom.xml ก็สามารถใช้งานได้ทันที สามารถสรุปข้อดีต่างๆ ได้ดังนี้

- มีโครงสร้างของ project ที่เป็นมาตรฐาน

```
my-app
|-- pom.xml
`-- src
    |-- main
    |   `-- java
    |       `-- com
    |           `-- mycompany
    |               `-- app
    |                   `-- App.java
    `-- test
        `-- java
            `-- com
                `-- mycompany
                    `-- app
                        `-- AppTest.java
```

ภาพที่ 2.5 โครงสร้างของ Maven project

Folder Structure	Description
my-app	ประกอบด้วยโฟลเดอร์ src และไฟล์ pom.xml
src/main/java	ประกอบด้วยไฟล์ java code package
src/main/test	ประกอบด้วยไฟล์ test java code package
src/main/resources	ประกอบด้วยไฟล์ images/properties files

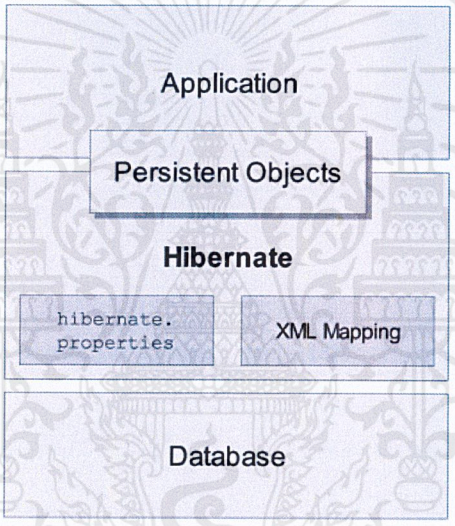
ภาพที่ 2.6 รายละเอียดโครงสร้างของ Maven project

- ทำหน้าที่เป็น Dependency Management ช่วยในการจัดการ library หรือ dependencies ต่างๆ ให้กับ Java project
- ช่วยในการ compile source code ไปเป็น .class ให้โดยอัตโนมัติ
- ช่วยในการรัน Unit Test/Integration Test ต่างๆ
- สามารถ build project ไปเป็นรูปแบบต่างๆ ตามที่ต้องการได้
- ช่วยในเรื่องการ deploy source code แบบอัตโนมัติ
- ช่วยในเรื่องการทำ Multiple modules
- สามารถทำ profile ให้สามารถ config ค่าต่างๆ ในแต่ละ environment ที่แตกต่างกันได้

2.1.5 Hibernate

Hibernate เป็น framework ที่ใช้ในการจัดการข้อมูลแบบ ORM เพื่อความสะดวกในการทำงานต่างๆ เช่น การเข้าถึงข้อมูล การแก้ไขข้อมูลหรือการเรียกค้นข้อมูล ซึ่งจะช่วยให้เราทำงานได้อย่างมีประสิทธิภาพมากขึ้น นอกจากนี้ Hibernate ยังเป็น open source จึงสามารถนำมาใช้งานได้ฟรีโดยไม่มีค่าใช้จ่ายใดๆ ทั้งสิ้น

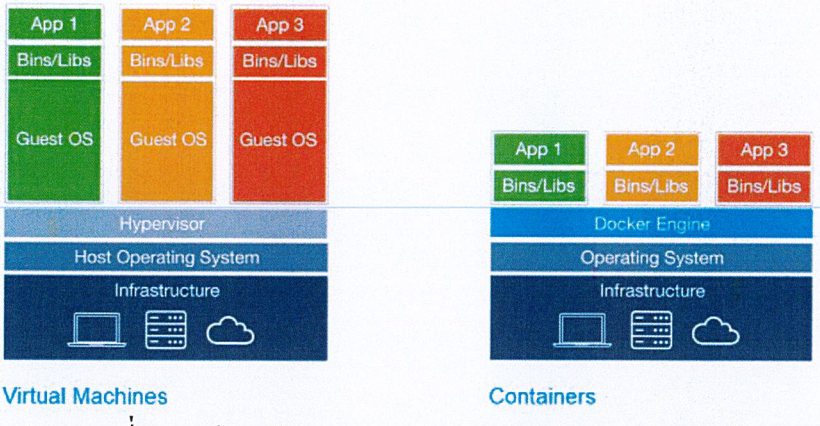
โดยการใช้งาน Hibernate นั้นจะต้องมีการจับคู่ระหว่าง Java class เข้ากับตารางในฐานข้อมูลผ่าน XML File ก่อน จากนั้น Hibernate จะสร้าง persistent object ที่ได้จากการอ่าน configuration และเมื่อเกิดการเปลี่ยนแปลงของ object นั้นๆ Hibernate ก็จะทำการเปิด transaction และเข้าไปแก้ไขข้อมูลในฐานข้อมูล



ภาพที่ 2.7 โครงสร้างของ Hibernate

2.1.6 Docker

Docker คือ engine หนึ่งที่มีการทำงานในลักษณะจำลองสภาพแวดล้อมขึ้นมาบนเครื่อง server เพื่อใช้ในการรัน service ที่ต้องการ มีการทำงานคล้ายคลึงกับ Virtual Machine แต่ต่างกันที่ Virtual Machine นั้นเป็นการจำลองทั้ง OS เพื่อใช้งานและหากต้องการใช้งาน service ใดๆ ก็ต้องทำการติดตั้งเพิ่มเติมบน OS นั้นๆ แต่สำหรับ Docker แล้วจะใช้ container ในการจำลองสภาพแวดล้อมขึ้นมาเพื่อใช้งานสำหรับ 1 service ที่ต้องการใช้งานเท่านั้น โดยไม่ต้องมีส่วนของ OS เข้าไปเกี่ยวข้องเหมือน Virtual Machines อื่นๆ



ภาพที่ 2.8 เปรียบเทียบระหว่าง Virtual Machines และ Docker

โดย Docker มีองค์ประกอบที่ควรรู้อยู่ 3 ส่วน ดังนี้

- Docker Image เป็น read only templates ที่เอาไว้สร้าง containers ถูกสร้างผ่านคำสั่ง docker build และถูกเก็บไว้ใน docker registry ซึ่งภายในจะประกอบด้วย application ต่างๆ ที่มีการติดตั้งไว้เพื่อใช้งานสำหรับ service นั้นๆ รวมทั้งมีการ config ค่าต่างๆ ไว้เรียบร้อยแล้ว จากนั้นก็นำมาสร้างเป็น docker image บน registry เพื่อนำใช้งาน ทั้งนี้ผู้ใช้งานสามารถยังสร้าง docker image สำหรับใช้งานเองได้อีกด้วย

- Docker Container ถูกสร้างมาจาก images ซึ่งภายในจะมี binaries และ dependencies ทั้งหมดที่จำเป็นของโปรแกรม สามารถมองได้เสมือนกล่อง ซึ่งนำ docker image มาติดตั้ง เพื่อให้สามารถใช้งาน service ที่ต้องการจาก image นั้นๆ ได้ โดยใน container แต่ละตัวจะมีการใช้งาน RAM, CPU และไฟล์ config ต่างๆ เป็นของแต่ละ container เอง และยังสามารถสั่ง start หรือ stop ได้ที่ container นั้นๆ อีกด้วย

- Docker Registries and Repositories เป็นที่ที่เอาไว้เก็บ images ของโปรแกรม ซึ่งเราสามารถสร้าง registry เองก็ได้หรือจะใช้ public registry ของ Docker ที่ชื่อ DockerHub ก็ได้ โดยภายใน registry จะมี repositories ที่เอาไว้จัดเก็บ images แต่ละตัว และใน repository แต่ละอันก็คือที่รวบรวม images ตัวเดียวกันแต่ต่างเวอร์ชัน โดยใช้ tags ในการแยก

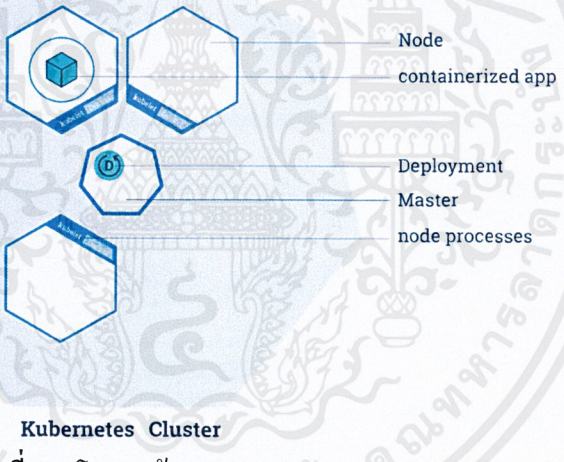
2.1.7 Kubernetes

Kubernetes หรือ k8s เป็น platform แบบ open source ที่ถูกพัฒนาและออกแบบโดยวิศวกรจาก Google ใช้สำหรับช่วยให้การปฏิบัติงานต่างๆ ที่เกี่ยวข้องกับ Linux Container สามารถทำได้โดยอัตโนมัติ ลดกระบวนการติดตั้งหรือขยายแอปพลิเคชันที่รันบน container ที่นักพัฒนาต้องลงมือทำด้วยตนเองให้เหลือน้อยที่สุด หรือกล่าวได้ว่า ช่วยให้นักพัฒนาสามารถทำ cluster กลุ่มของhost ที่รัน Linux Container ได้ และ k8s ก็เข้ามาช่วยบริหารจัดการ

cluster เหล่านั้นสำหรับใช้งานบน Public, Private และ Hybrid Cloud ได้อย่างสะดวกรวดเร็วและมีประสิทธิภาพ

สำหรับบริษัทที่นำเทคโนโลยี container มาใช้งานจะพบว่าแอปพลิเคชันบน Production มักมีการเชื่อมต่อหลายๆ container เข้าด้วยกัน ซึ่ง container เหล่านั้นมักถูกติดตั้งบนหลายๆ host k8s สามารถเข้ามาช่วยประสานการทำงาน (Orchestration) และบริหารจัดการ container ทั้งการติดตั้งและการขยายระบบในอนาคต ส่งผลให้นักพัฒนาสามารถสร้างแอปพลิเคชันที่เชื่อมต่อหลายๆ container เข้าด้วยกัน กำหนดการทำงานต่างๆ ขยายระบบ และตรวจสอบการทำงานทั้งหมดได้อย่างง่ายดาย ที่สำคัญคือสามารถดำเนินคำสั่งต่างๆ ได้โดยอัตโนมัติอีกด้วย โดยองค์ประกอบหลักๆ ของ k8s มีดังนี้

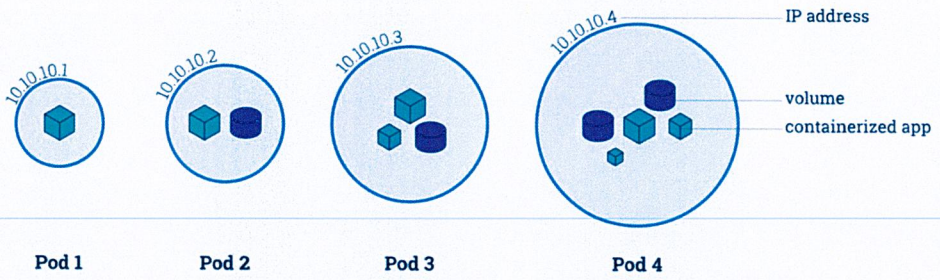
- Cluster คือกลุ่มของ server ที่ถูกรวมไว้ให้เป็น server ใหญ่ๆ เพียงเครื่องเดียว โดย node ใน cluster จะมีอยู่ 2 ประเภท คือ master node และ worker node



ภาพที่ 2.9 โครงสร้างของ Kubernetes cluster

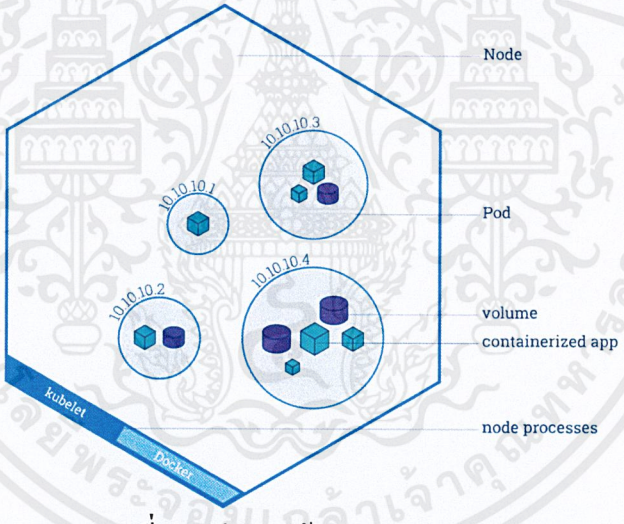
- Deployment เป็น configuration ที่ไว้สร้างหรืออัพเดท app และบอกว่า app เราจะรันยังไง กำหนด desired state ของ app โดยที่ k8s จะพยายามดูแลให้เป็นไปตาม config ในไฟล์ .yaml ตลอดเวลา (self-healing)

- Pod เป็นหน่วยที่เล็กที่สุดใน k8s โดยในแต่ละ pod จะประกอบไปด้วย containers resource ของ container นั้นๆ และค่า config ต่างๆ ของ container แต่ละอัน

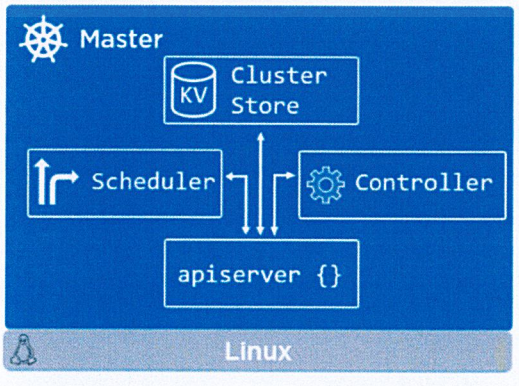


ภาพที่ 2.10 โครงสร้างของ Kubernetes pod

- Node แบ่งออกเป็น 2 ประเภท คือ master node ที่เป็นเหมือนเป็นศูนย์บัญชาการคอยสั่งการ Node ต่างๆ ว่าต้องทำอะไรบ้าง และ worker node ที่เป็นเหมือนสำนักงานย่อย และแต่ละ node จะมี process ชื่อ kubelet ไว้สื่อสารกับ master node มี docker ในตัวเพื่อรัน container และเนื่องจาก pod จะรันอยู่บน node เสมอ นั่นหมายความว่า ในแต่ละ worker node นั้นสามารถประกอบไปด้วย pod หลายๆตัวได้



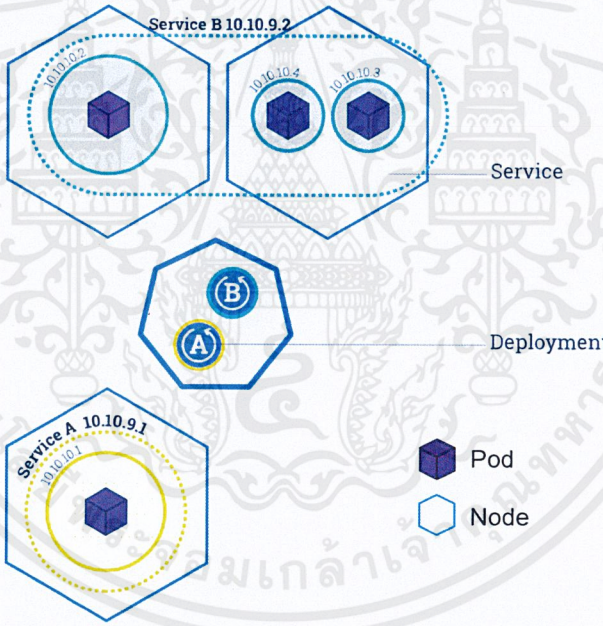
ภาพที่ 2.11 โครงสร้างของ worker node



ภาพที่ 2.12 หน้าที่การทำงานของ master node

- ReplicaSet มีหน้าที่หลักคือทำให้ pod มีจำนวนเท่ากับที่เรากำหนด ควบคุมดูแลตาม config ของ deployment ให้เป็นไปอย่างถูกต้องอยู่เสมอ คือหากมีจำนวนมากไปจากที่กำหนดก็ทำลายทิ้ง หรือหากน้อยไปก็สร้างเพิ่ม และยังสามารถใช้ในการทำ auto scale เพื่อใช้ประโยชน์ได้ เช่น กำหนดลงไปใน config ว่า หาก CPU usage over 80% ให้สร้าง pod เพิ่มขึ้นมาอีกได้

- Services ทำหน้าที่กระจาย load balance ไปยังแต่ละ pod และสามารถทำหน้าที่เป็น API Gateway ได้ในตัว นั่นทำให้เวลาเกิดการเพิ่มหรือลดของ pod ตัวที่ทำการเรียกมายัง pod ดังกล่าว จะไม่รู้สึกเลยว่าหลังบ้านของ pod มีการเปลี่ยนแปลง เพราะมี service เป็นกลไกการทำงานและเป็นตัวกลางสื่อสารให้ โดยใน service จะมีการใช้ labels ซึ่งตั้งค่าเอาไว้ที่ config file โดย set label ไว้ที่แต่ละ pod และไป set ที่ service อีกทีว่าถ้าเรียก service นี้ จะให้ส่งต่อไปยัง pod ที่มี label อะไรบ้าง



ภาพที่ 2.13 โครงสร้างของ Kubernetes services

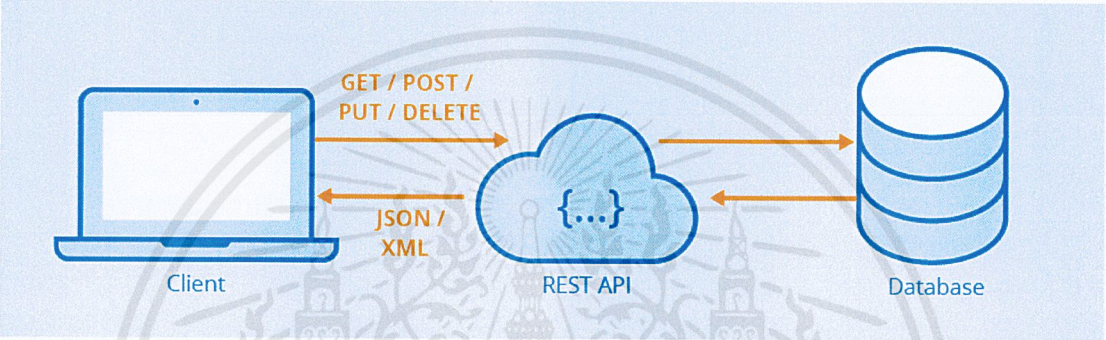
2.1.8 MySQL

MySQL คือโปรแกรมระบบจัดการฐานข้อมูลที่พัฒนาโดยบริษัท MySQL AB มี function การทำงานแบบ Relation Database Management System (RDBMS) โดยอาศัย Structured Query Language (SQL) เป็นภาษาในสื่อสาร มีหน้าที่เป็นตัวกลางในการจัดการกับข้อมูลในฐานข้อมูล เพิ่มและเก็บข้อมูลอย่างเป็นระบบ เข้าถึงและประมวลผลข้อมูลในฐานข้อมูลได้

โดยโปรแกรมถูกออกแบบให้สามารถทำงานได้บนระบบปฏิบัติการที่หลากหลาย และเป็นระบบฐานข้อมูล open source ที่ถูกนำไปใช้งานมากที่สุด

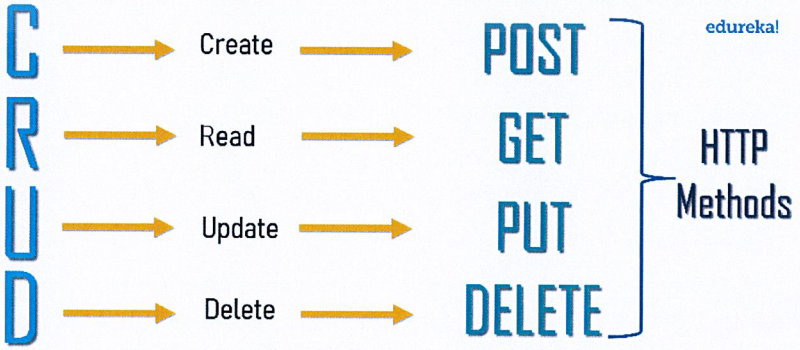
MySQL เป็นระบบจัดการฐานข้อมูลแบบ Relational ที่จะทำการเก็บข้อมูลทั้งหมดในรูปแบบของตารางแทนการเก็บข้อมูลทั้งหมดลงในไฟล์เพียงไฟล์เดียว ทำให้ทำงานได้รวดเร็วและมีความยืดหยุ่น นอกจากนั้นแต่ละตารางที่เก็บข้อมูลสามารถเชื่อมโยงเข้าหากันทำให้สามารถรวมหรือจัดกลุ่มข้อมูลได้ตามต้องการโดยการใช้ภาษา SQL

2.1.9 Rest API



ภาพที่ 2.14 หลักการทำงานของ REST API

Representational State Transfer หรือ REST หรือ RESTful ถูกตั้งขึ้นโดย Roy Fielding ในปี 2000 คือการสร้าง webservice ชนิดหนึ่งที่ใช้สื่อสารกันบน internet ที่ใช้หลักการแบบ stateless คือไม่มี session การทำงานของ RESTful จะอาศัย URI/URL ของ request เพื่อค้นหาและประมวลผลแล้วตอบกลับไปในรูป XML HTML หรือ JSON โดย response ที่ตอบกลับไปจะเป็นการยืนยันผลของคำสั่งที่ส่งมา นอกจากนี้ยังสามารถพัฒนาด้วยภาษา programming ที่หลากหลาย โดยคำสั่งจะมีตาม HTTP verbs ซึ่งก็คือ GET สำหรับการดึงข้อมูล, POST สำหรับสร้างข้อมูล, PUT สำหรับการแก้ไขข้อมูลและ DELETE สำหรับลบข้อมูล

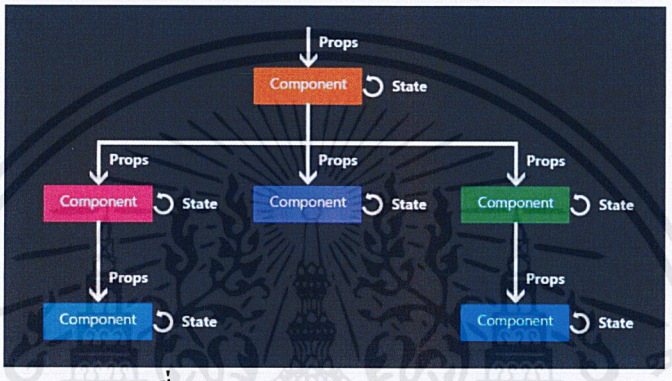


ภาพที่ 2.15 เปรียบเทียบการทำงานกับ HTTP methods

2.1.10 React

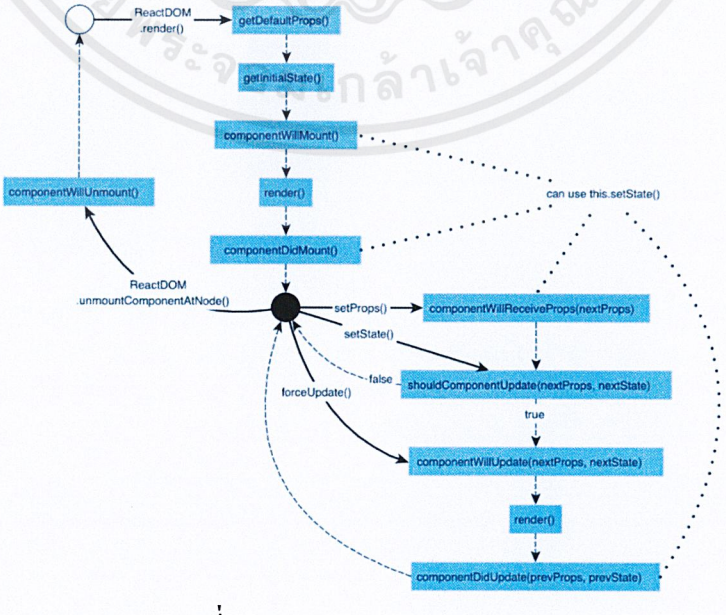
React คือ JavaScript library ที่ทีม Facebook เป็นคนพัฒนาขึ้นมา และเปิดให้คนทั่วไปสามารถนำมาใช้ได้ฟรี มีหน้าที่ในการสร้าง view ของเว็บไซต์ โดยในการเขียน React นั้นต้องเข้าใจใน 3 concept ดังนี้

- Component คือส่วนประกอบต่างๆ ในหน้าเว็บ
- State คือข้อมูลที่อยู่ภายใน component แต่ละชิ้น
- Props คือข้อมูลที่ถูกส่งต่อจาก component จากชั้นบนลงไปชั้นล่าง



ภาพที่ 2.16 concept การทำงานของ React

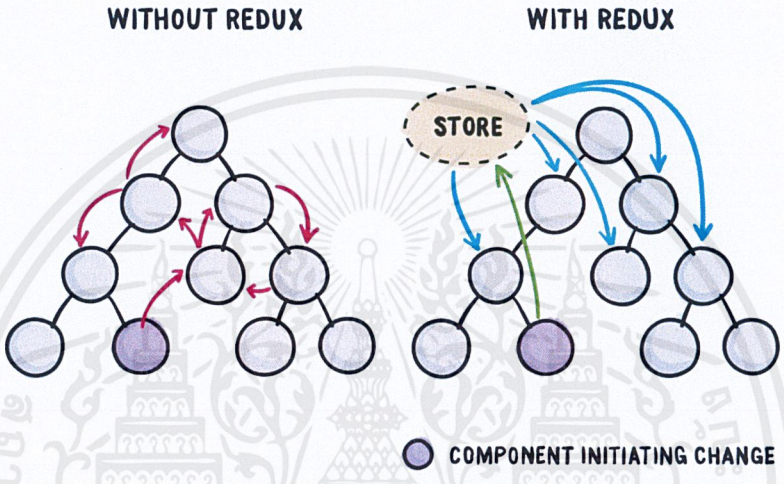
นอกจากนี้การจะเขียน React ให้ได้ดีนั้นก็ควรจะเข้าใจการเกิดและดับไปของ component ที่เรียกว่า lifecycle เพราะถ้าเราเข้าใจส่วนนี้ได้ เราก็สามารถแทรกโค้ดเพื่อให้ทำงานเฉพาะในจังหวะต่างๆ ใน cycle ได้ โดย lifecycle ของ React มีลักษณะดังภาพ



ภาพที่ 2.17 lifecycle ของ component

แต่เมื่อแอปพลิชั่นมีขนาดใหญ่ขึ้น มี component มากขึ้นก็จะทำให้เกิดปัญหาในการจัดการ state จึงทำให้ต้องใช้หลักการของ Redux เข้ามาช่วยแก้ปัญหานี้ โดยหลักการของ Redux ทั้ง 3 ข้อมีดังนี้

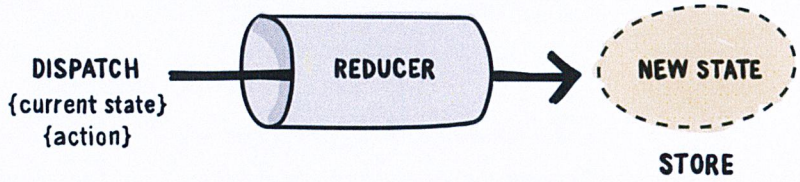
- Single Source of Truth คือ Redux จะมีการสร้างสิ่งที่เรียกว่า store ขึ้นมาเพื่อใช้เก็บ state ของทั้งแอปพลิเคชั่นไว้ที่เดียว และทำหน้าที่ส่ง state ไปให้แต่ละ component ที่ต้องการใช้



ภาพที่ 2.18 การทำงานของ store

- State is Read-Only คือ state ที่ถูกเก็บไว้ใน store จะต้องถูกแก้ก็ต่อเมื่อเกิด action ขึ้นเท่านั้น โดย action ก็เป็น object ธรรมดาที่มีการบอก type ว่าจะทำอะไรและอาจมี parameter อื่นๆ สำหรับ action นั้นๆ ด้วย

- Changes are Made with Pure Functions คือการแก้ไข state จะต้องทำผ่าน pure function ที่เรียกว่า reducer เท่านั้น reducer จะเป็น function ที่มีหน้าที่ในการดูว่า state ปัจจุบันเป็นอะไร แล้วถ้าเจอ action นี้มา state ต่อไปจะเป็นอะไร



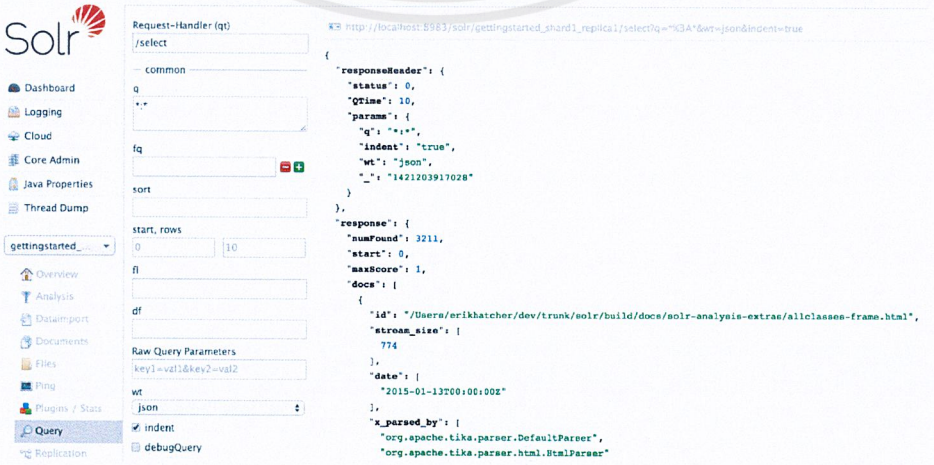
ภาพที่ 2.19 การทำงานของ reducer

2.1.11 Solr

Solr คือ open source search platform พัฒนาอยู่บน Lucene ทำงานอยู่บน Servlet container เช่น Tomcat และ Jetty ใช้ในการ query ข้อมูลและสามารถกำหนดรูปแบบหรือวิธีการดึงข้อมูลออกมาได้หลายแบบ สามารถเลือกผลการค้นหาได้ทั้ง XML, JSON, PHP, Ruby, Python, XSLT หรือ Velocity สามารถสร้างส่วน caching และ สนับสนุนระบบ plugin สนับสนุนการทำ clustering สามารถ monitoring ผ่าน logging ได้และสนับสนุนการทำ Distributed Search ด้วยการทำ sharding ทั้งนี้ยังรองรับพวก document เช่น Word, PDF และ HTML จึงทำให้ใช้งานได้หลากหลายและมีความยืดหยุ่นต่อการใช้งานมากขึ้น

หัวใจหลักของการใช้งาน Solr นั้นคือการสร้าง schema กำหนดรูปแบบโครงสร้างในการจัดเก็บข้อมูล ประกอบไปด้วยส่วนหลักๆ คือ fields/columns, data types ของ fields/columns และ text analyser ที่จะใช้ในการทำ index ของข้อมูล เมื่อมี schema จำนวนมากขึ้นเราก็สามารถจัดกลุ่มรวม schema เหล่านั้นให้เป็น collection เพื่อรวมให้เป็นระเบียบได้ การค้นหาของ Solr นั้นเป็นระบบ Full Text Search ที่จะค้นหาเป็นคำๆ ไป และไม่มีการ Relationship ความสัมพันธ์ใดๆ เนื่องจากเป็น NO-sql โดย parameter พื้นฐานในการค้นหาของข้อมูลโดยใช้ Solr ที่สามารถใช้งานผ่านหน้า Solr Admin มีรายละเอียด ดังนี้

- q คือ keyword ที่ใช้ในการค้นหา โดยใส่ในรูปแบบของ query term
- fq คือ filter query การกรองการค้นหาโดย filter จาก field ต่างๆ
- sort คือการจัดเรียงผลการค้นหาว่าให้จัดเรียงตาม field ไหน
- start, rows คือจำนวนแถวของข้อมูลที่นำมาแสดงผล
- fl คือ field ที่จะให้แสดงผลในผลการค้นหา
- df คือ default field ที่จะให้ในการค้นหากรณีที่ไม่ได้กำหนด field ไว้
- wt คือ format response รูปแบบไฟล์ของข้อมูลผลการค้นหา



ภาพที่ 2.20 interface หน้า admin ของ Solr

2.1.12 GitLab

GitLab เป็น open source ที่เป็น Git Repository Management ที่ทำหน้าที่จัดการ code collaboration และ version control นอกจากนี้ยังสามารถทำ code reviews, issue tracking, activity feeds และ wikis สามารถจัดการ Git Repository และจัดการ CI/CD (Continuous integration and Continuous delivery) ได้อีกด้วย

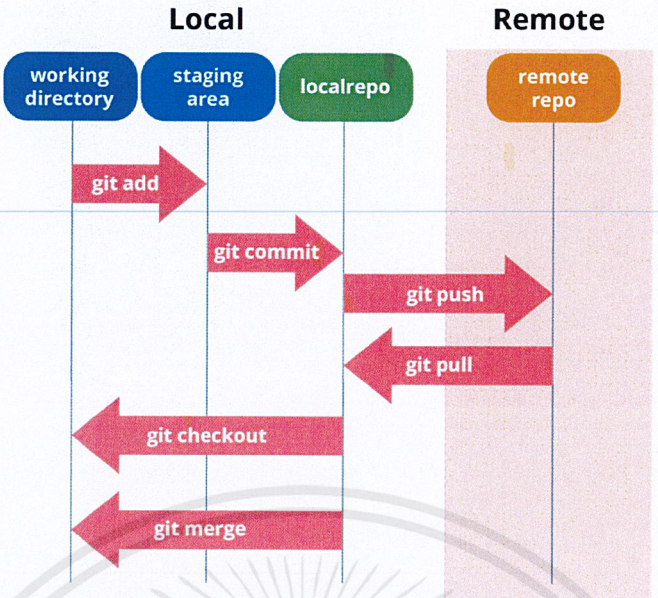
GitLab เป็น software ที่พัฒนาต่อยอดขึ้นมาจาก Git เป็น open source ที่เป็น Git Repository Management ที่เก็บ source code จัดการ Git repository เพิ่มด้วยความสามารถในจัดการ code collaboration การทำ code reviews, issue tracking, activity feeds และ wikis ที่สำคัญคือสามารถจัดการ CI/CD (Continuous integration and Continuous delivery) ได้โดยยังสามารถทำ Version Control ได้เหมือน Git โดยการใช้ Git ก็มี status และ command การใช้งานเบื้องต้นที่ควรรู้จักและใช้งานอย่างถูกต้องได้ ดังนี้

Git status ที่บ่งบอกถึงสถานะของ source code ของ Git มีดังนี้

- working directory คือสถานะที่ source code กำลังมีการเปลี่ยนแปลงหรือถูกแก้ไข
- staged คือสถานะที่ source code กำลังจะเตรียม commit เพื่อยืนยันการเปลี่ยนแปลงก่อนจะเก็บลงในสถานะถัดไป
- local repository คือสถานะที่มีการเก็บบันทึกข้อมูลการเปลี่ยนแปลงของ source code ลง Git repository ที่เป็น local
- remote repository คือสถานะที่มีการเก็บบันทึกข้อมูลการเปลี่ยนแปลงของ source code ลง Git repository ที่เป็น hosting

Git command เป็นคำสั่งในการใช้งาน Git เบื้องต้นมีดังนี้

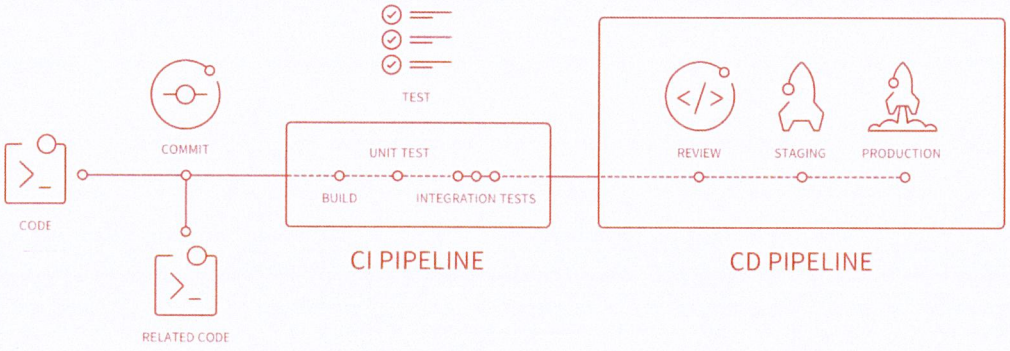
- git add เป็นคำสั่งที่ใช้เพิ่มการเปลี่ยนแปลงของ source code เข้าสู่สถานะ staged
- git commit ใช้ยืนยันการเปลี่ยนแปลงของ source code จากสถานะ staged เข้าเก็บไว้ที่ local repository
- git push ใช้ส่งการเปลี่ยนแปลงของ source code ที่เก็บไว้ที่ local repository ไปยัง remote repository
- git pull ใช้รับการเปลี่ยนแปลงของ source code ล่าสุดใน remote repository ลงมายัง local repository และทำการ auto merge
- git checkout ใช้ในการสลับ working directory ไปยัง branch หรือ commit ที่ต้อการ
- git merge ใช้ในการรวม branch หรือ commit เข้าด้วยกัน



ภาพที่ 2.21 แผนภาพแสดงความสัมพันธ์ระหว่างการทำงานของ Git command และ status

นอกจากนี้ฟีเจอร์เด่นที่สำคัญใน GitLab ก็คือความสามารถในการจัดการ CI/CD (Continuous integration and Continuous delivery) หรือเรียกสั้นๆ ได้ว่า Pipeline แบ่งได้เป็น 2 ส่วนคือ CI หรือ Continuous Integration คือกระบวนการรวบรวม software ที่มีการพัฒนาแยกส่วนกันมารวมกันอย่างอัตโนมัติ โดยจะมีการเขียน script test คอยทดสอบความเข้ากันในการทำงานของระบบเพื่อไม่ให้มี bug หลุดเข้ามาใน source code โดยการ testing จะเริ่มตั้งแต่ Unit Testing ที่ผู้พัฒนาเป็นคนสร้างจนถึง Integration Testing เลย

อีกส่วนหนึ่งคือ CD หรือ Continuous Delivery คือการทำงานต่างๆ ใน Pipeline ตั้งแต่การ compile, build ไปจนถึงการทดสอบระบบต่างๆ ทั้งหมดจะถูกทำโดยอัตโนมัติ แต่การ deploy ขึ้น production นั้นจะต้องได้รับการอนุมัติก่อน คือเป็นการทำงานแบบ manual หรืออาจเป็นแบบ One Click Deploy ก็ได้

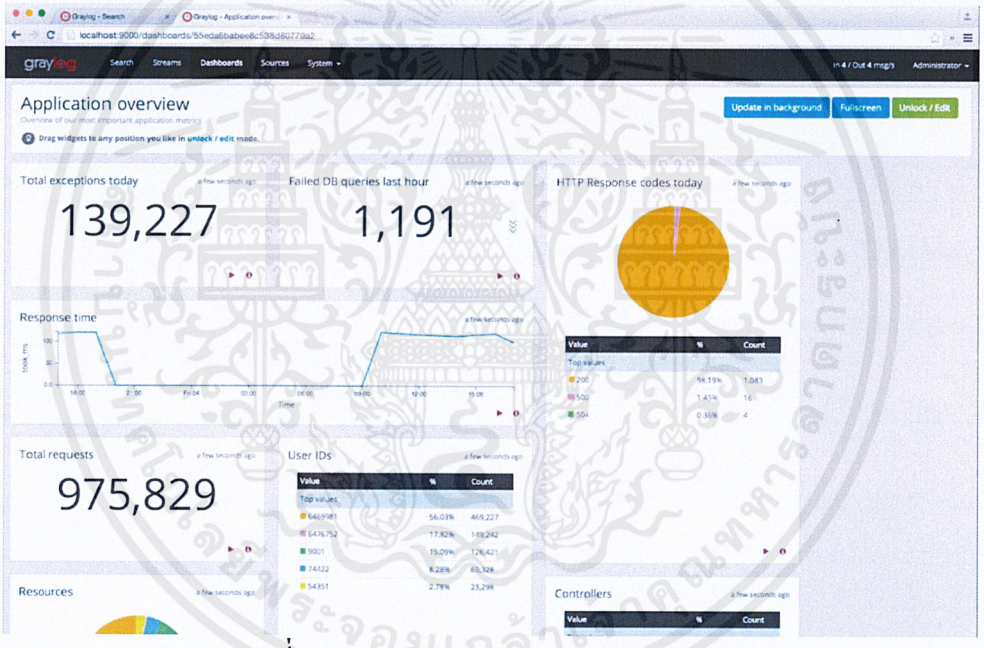


ภาพที่ 2.22 แผนภาพการทำงานแบบ CI/CD

2.1.13 Graylog

Graylog เป็น open source platform ที่มีหน้าที่ในการเก็บ วิเคราะห์และจัดการกับ log ที่มีความสามารถในการค้นหาและวิเคราะห์ผลออกจากไฟล์ log ได้ โดยถูกออกแบบมาให้รับข้อมูลจากหลายแหล่ง เพื่อจัดการข้อมูลจากศูนย์กลางซึ่งรองรับข้อมูลได้วันละหลายเทระไบต์ต่อวัน ซึ่ง log ทั้งหมดนี้จะถูกส่งมายัง log server ที่เรียกว่า Centralize Log System

โดย Graylog ก็มีความน่าสนใจหลายอย่าง เช่น การนำ ElasticSearch เข้ามาใช้งานในเก็บข้อความ ความสามารถที่จะรับข้อมูลผ่านทาง TCP/UDP รวมไปถึง AMQP (message queue) สามารถทำ dashboard เพื่อให้ผู้ใช้สามารถดูเทรนการใช้งานจาก log ได้ตลอดเวลาผ่าน web interface ที่สวยงามและใช้งานได้ง่าย รวมถึง Graylog ยังมีกระบวนการป้องกันการผิดพลาดและสามารถทำการค้นหาแบบ Multi-thread ได้อีกด้วย



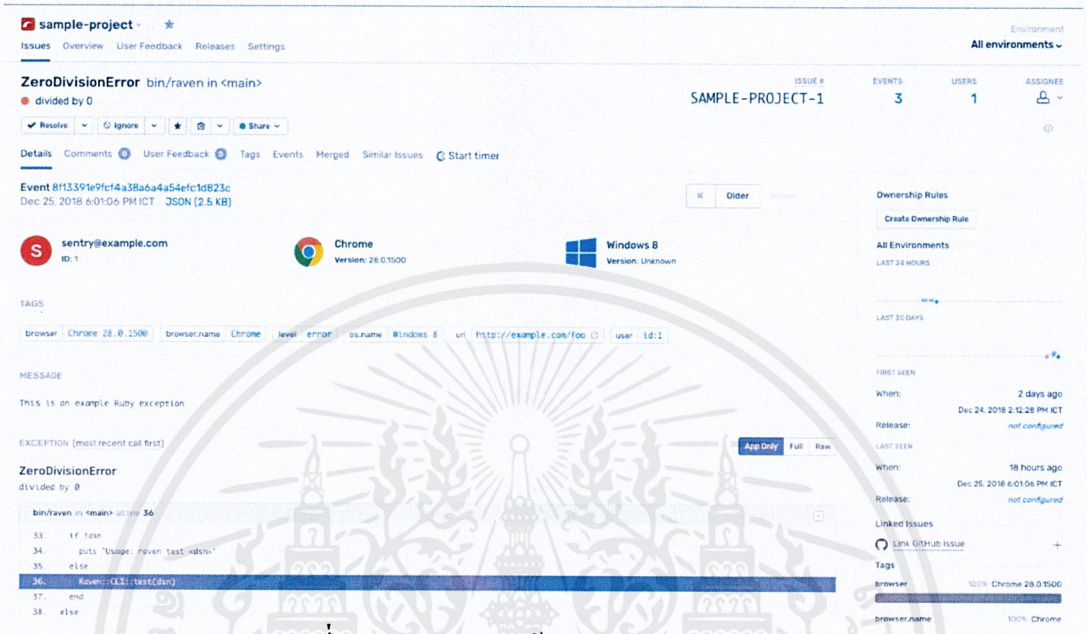
ภาพที่ 2.23 ตัวอย่าง web interface ของ Graylog

2.1.14 Sentry

Sentry เป็นเครื่องมือในการทำ Error Tracking และ Crash Reporting เป็นตัวช่วยดักจับเหตุการณ์แบบ real time ที่คอยรายงาน error จากเครื่องผู้ใช้งานไปที่เว็บ เพื่อให้เราสามารถตรวจสอบและแก้ไขปัญหาได้อย่างรวดเร็ว และยังมีการเก็บสถิติ error ที่เคยเกิดขึ้น มี traceback ในหน้า issue ที่อ่านง่าย สามารถเก็บ variable ใน state ขณะที่เกิด error ไว้ อีกทั้งยังสามารถแจ้งเตือนผ่าน email หรือ integrate เป็น chat bot ไปแจ้งเตือนใน Slack ได้อีกด้วย

โดยหลักการทำงานของ Sentry นั้นง่ายมากคือเมื่อผู้ใช้งานมี interaction หรือกระทำการใดๆ กับ website ของเราแล้วเกิด error ขึ้นมา Sentry ก็จะทำการส่ง email หรือ

ข้อความไปบอกเราว่าเกิด error ขึ้นในเว็บของเรา และเมื่อเราเข้าไปดูรายละเอียดใน dashboard ก็จะมีข้อมูลของ error ที่เจอว่าเกิดขึ้นจากไฟล์ไหน เกิดขึ้นเพราะอะไร เคยเกิด error แบบเดียวกันมาแล้วกี่ครั้ง ทำให้การ debug ทำได้ง่ายขึ้น



ภาพที่ 2.24 interface หน้า issue detail ของ Sentry

บทที่ 3
วิธีการดำเนินการวิจัย

รายงานสหกิจฉบับนี้จะยกตัวอย่างงานการพัฒนาระบบการออกใบแจ้งหนี้อัตโนมัติให้ร้านอาหารมาอธิบาย ซึ่งบทนี้จะกล่าวถึงขั้นตอนการดำเนินงาน การวิเคราะห์ การออกแบบโครงสร้างและการทำงานของระบบ

3.1 ภาพรวมของระบบ

ระบบการออกใบแจ้งหนี้อัตโนมัติให้ร้านอาหารเป็นระบบที่ถูกพัฒนาต่อยอดเพิ่มเติมในระบบ RMS ที่ถูกพัฒนาโดยใช้ Spring Boot มีการออกแบบให้เก็บข้อมูลที่อยู่สำหรับใบแจ้งหนี้เพิ่มลงไปในระบบ โดยที่อยู่นี้จะผูกติดกับผู้ใช้งานในระบบ และผู้ใช้งานในระบบที่เป็นผู้ดูแลร้านอาหาร ก็จะสามารถนำที่อยู่สำหรับใบแจ้งหนี้ที่เคยเพิ่มไว้ในบัญชีของตนเองมาใส่ให้เป็นที่อยู่ในการออกใบแจ้งหนี้ของร้านอาหารได้

ด้านการติดต่อกับผู้ดูแลระบบจะมีการพัฒนาเพิ่มเติมใน Admin RMS ที่ถูกพัฒนาด้วย React โดยมีการเพิ่มหน้าหลักของที่อยู่สำหรับใบแจ้งหนี้ เพื่อให้ผู้ดูแลระบบสร้างเพิ่ม แก้ไข และจัดการข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ที่มีอยู่ในระบบได้สะดวกยิ่งขึ้น

ในส่วนของการสร้างใบแจ้งหนี้ให้ร้านอาหารอัตโนมัตินั้นได้มีการสร้างระบบขึ้นมาใหม่เพื่อเป็นตัวเชื่อมต่อระหว่างระบบ RMS ที่มีข้อมูลยอดขายและค่าบริการต่างๆ กับระบบ ERP ที่มีข้อมูลเกี่ยวกับทางบัญชีของบริษัทเข้าด้วยกัน โดยระบบใหม่ที่สร้างขึ้นนี้มีชื่อว่า RMS Accounting มีหน้าที่หลักอยู่ 2 อย่างคือการเชื่อมต่อข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ของทั้ง 2 ระบบให้ตรงกัน และการสร้างใบแจ้งหนี้ให้ร้านอาหารเก็บเข้าสู่ระบบ ERP อัตโนมัติในทุกๆ เดือน โดยระบบนี้ก็ถูกพัฒนาโดยใช้ Spring Boot เช่นกัน เนื่องจากสามารถตั้งค่าระบบและเปิดการใช้งานได้ง่ายและรวดเร็ว

3.2 ขั้นตอนการดำเนินงาน

1. สอบถามและรวบรวมความต้องการและขอบเขตของโครงงาน
2. ศึกษาและทำความเข้าใจเกี่ยวกับความต้องการโดยรวมของระบบ
3. วิเคราะห์และวางแผนการพัฒนาโครงงานให้สอดคล้องกับความต้องการและระยะเวลาที่กำหนด
4. ศึกษาเทคโนโลยีที่ใช้สำหรับการพัฒนาโครงงาน
5. ออกแบบโครงสร้างและขั้นตอนการทำงานของระบบ
6. พัฒนาและปรับปรุงระบบให้สอดคล้องกับการเปลี่ยนแปลงของความต้องการ
7. ทดสอบและปรับปรุงการทำงานให้ถูกต้อง เหมาะสมและมีประสิทธิภาพ

8. สรุปผลและจัดทำเอกสารอธิบายกระบวนการทำงานและโครงสร้างของโครงงาน

3.3 การเก็บรวบรวมความต้องการ

จากการเก็บรวบรวมข้อมูลและความต้องการของระบบการออกใบแจ้งหนี้อัตโนมัติจึงได้ข้อสรุปคุณสมบัติการทำงานของระบบได้ดังนี้

1. สามารถกรอกข้อมูลเพื่อสร้างที่อยู่ในการออกใบแจ้งหนี้สำหรับผู้ใช้งานและบันทึกลงฐานข้อมูลได้

2. สามารถขอข้อมูลที่อยู่ในการออกใบแจ้งหนี้ที่มีอยู่ในระบบมาแสดงผลและค้นหาแบบคัดกรองตามข้อมูลที่ต้องการได้

3. สามารถเรียกดูข้อมูลที่อยู่ในการออกใบแจ้งหนี้ทั้งหมดของผู้ใช้งานได้

4. สามารถแก้ไขเปลี่ยนแปลงข้อมูลของที่อยู่ในการออกใบแจ้งหนี้ได้

5. สามารถเพิ่มที่อยู่ในการออกใบแจ้งหนี้ให้แก่ร้านอาหารที่มีอยู่ในระบบได้

6. สามารถแก้ไขข้อมูลที่อยู่ในการออกใบแจ้งหนี้ให้แก่ร้านอาหารในระบบได้

7. สามารถส่งออกข้อมูลที่จำเป็นในการออกใบแจ้งหนี้ เช่น ข้อมูลร้านอาหาร ที่อยู่ในการออกใบแจ้งหนี้และค่าบริการที่ต้องจ่ายของร้านอาหารที่มีอยู่ในระบบทั้งหมดออกมาได้

8. สามารถเชื่อมต่อที่อยู่ในการออกใบแจ้งหนี้ระหว่างระบบ RMS และ ERP ให้ตรงกันได้

9. สามารถดึงข้อมูลต่างๆ จากระบบ RMS ของร้านอาหารไปสร้างใบแจ้งหนี้ในระบบ ERP ได้

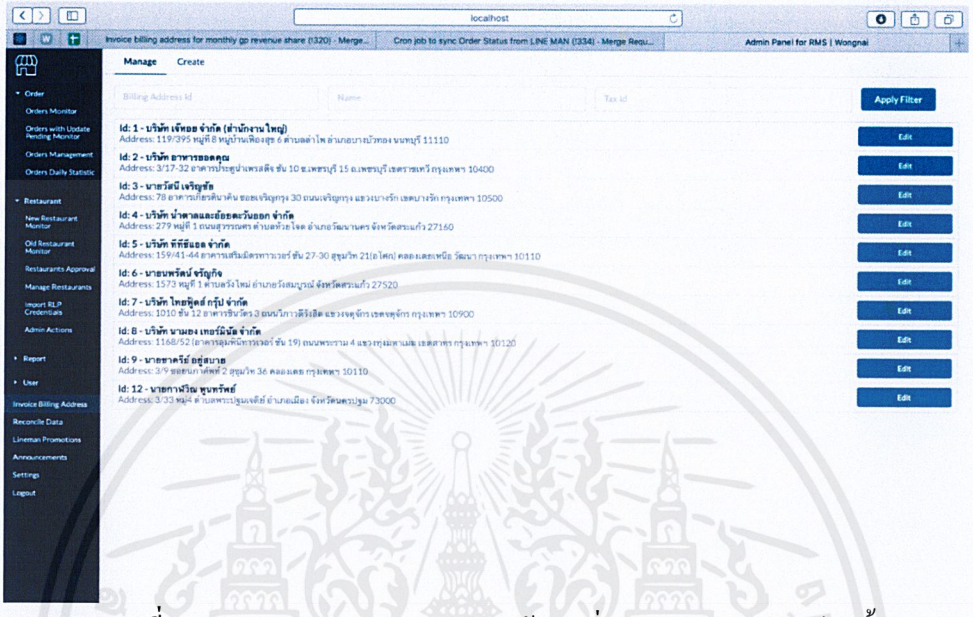
3.4 การออกแบบโครงสร้างของระบบ

เนื่องจากโครงสร้างของระบบเดิมมีการแบ่งโครงสร้างหลักออกเป็น 2 ส่วนใหญ่ คือ ส่วนที่ติดต่อกับผู้ใช้งานหรือ user interface ที่พัฒนาขึ้นด้วย React และส่วนที่ติดต่อกับฐานข้อมูลหรือ server ที่พัฒนาโดยใช้ Spring Framework ซึ่งตัว server นี้ก็จะมีการถูกเรียกใช้งานโดยอีกหลายระบบเช่นกัน จึงต้องมีการออกแบบ API สำหรับการสื่อสารกันระหว่างระบบเหล่านั้นด้วย

3.4.1 ส่วนที่ติดต่อกับผู้ใช้งาน

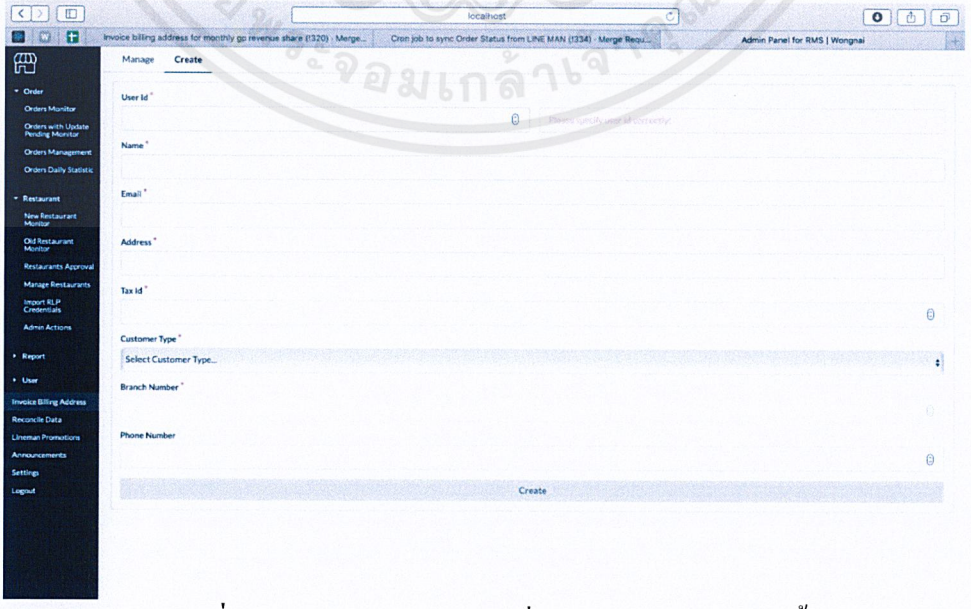
ในระบบของ RMS จะมีหน้าเว็บในการจัดการข้อมูลของผู้ใช้งานร้านอาหารหรือคำสั่งซื้อสินค้าสำหรับผู้ดูแลระบบที่เรียกว่า Admin RMS อยู่แล้ว ในการพัฒนาระบบออกใบแจ้งหนี้อัตโนมัติจึงต้องเพิ่มหน้าสำหรับการจัดการข้อมูลของที่อยู่ในการออกใบแจ้งหนี้ใน Admin RMS ด้วย เพื่อให้ผู้ดูแลระบบสามารถจัดการข้อมูลได้อย่างสะดวกและรวดเร็ว จึงมีการเพิ่มหน้าสำหรับจัดการข้อมูลต่างๆ ดังนี้

- หน้าแรกของการจัดการข้อมูลที่อยู่ในการออกใบแจ้งหนี้ จะแสดงข้อมูลที่อยู่ในการออกใบแจ้งหนี้ทั้งหมดที่มีอยู่ในระบบ โดยสามารถกรองข้อมูลหมายเลขไอดี ชื่อหรือหมายเลขประจำตัวผู้เสียภาษีของที่อยู่ใบแจ้งหนี้เพื่อค้นหาที่อยู่ที่ต้องการได้



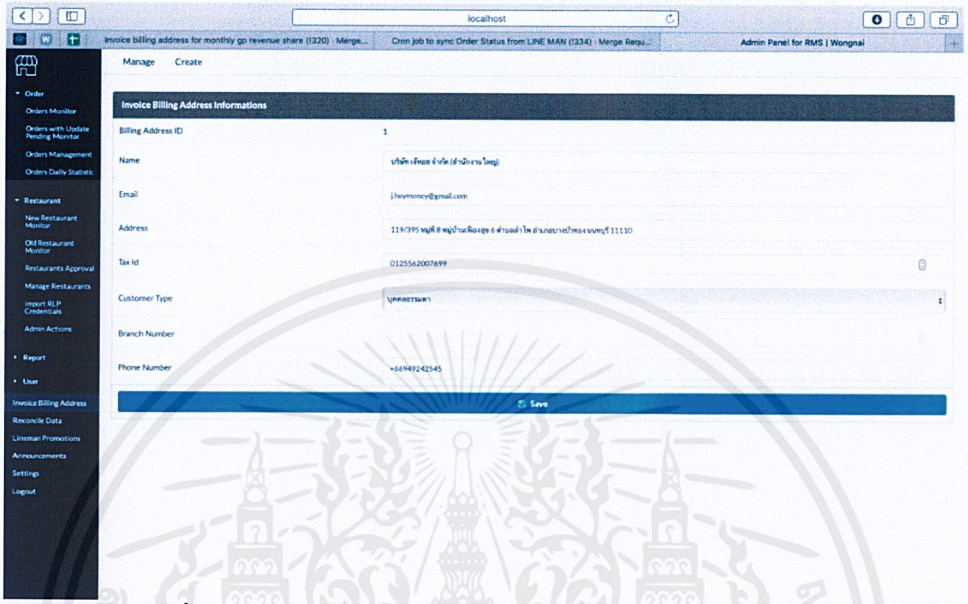
ภาพที่ 3.1 จอภาพหลักของการจัดการข้อมูลที่อยู่ในการออกใบแจ้งหนี้

- หน้าสำหรับการสร้างที่อยู่ในการออกใบแจ้งหนี้ จะมีช่องให้ผู้ดูแลระบบสามารถกรอกข้อมูลต่างๆ ของที่อยู่สำหรับใบแจ้งหนี้และสร้างเก็บเข้าไปในระบบฐานข้อมูลได้ โดยข้อมูลที่กรอกจะต้องมีความถูกต้องและครบถ้วนจึงจะสามารถสร้างและบันทึกลงระบบได้



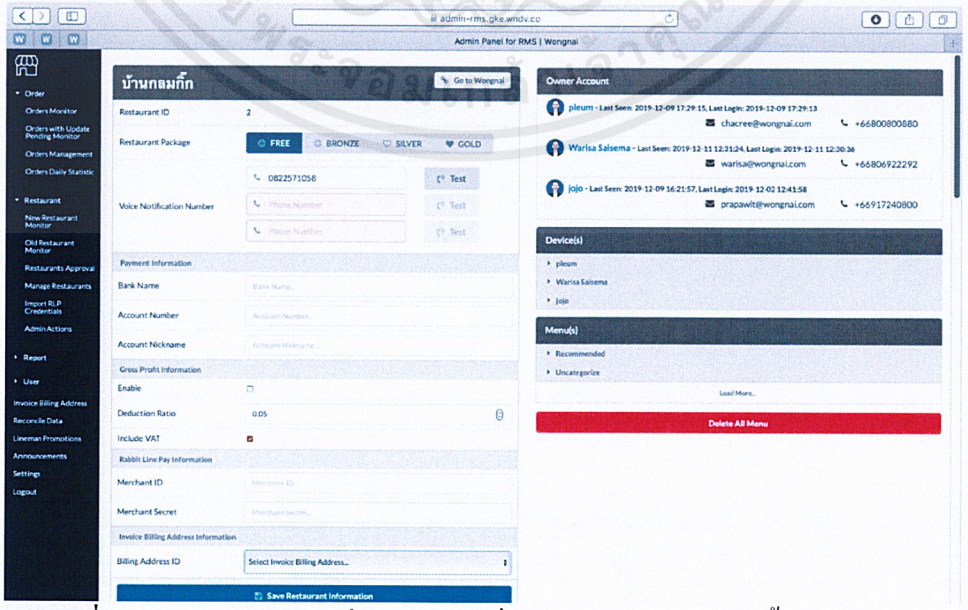
ภาพที่ 3.2 จอภาพหลักการสร้างที่อยู่ในการออกใบแจ้งหนี้

- หน้าสำหรับการแก้ไขข้อมูลที่อยู่ในการออกใบแจ้งหนี้ จะมีช่องสำหรับการแก้ไขข้อมูลต่างๆ ของที่อยู่สำหรับใบแจ้งหนี้ โดยข้อมูลที่กรอกจะต้องมีความถูกต้องและครบถ้วนจึงจะสามารถแก้ไขข้อมูลในระบบได้



ภาพที่ 3.3 จอภาพหลักการแก้ไขข้อมูลที่อยู่ในการออกใบแจ้งหนี้

- หน้าสำหรับการเพิ่มและแก้ไขที่อยู่ในการออกใบแจ้งหนี้ให้ร้านอาหาร ได้มีการเพิ่มช่อง Billing Address ID ให้ผู้ดูแลระบบสามารถเลือกที่อยู่สำหรับใบแจ้งหนี้ของ user ที่เป็นผู้ดูแลร้านเพิ่มให้ร้านอาหารและเก็บเข้าไปในระบบฐานข้อมูลได้



ภาพที่ 3.4 จอภาพหลักการเพิ่มและแก้ไขที่อยู่ในการออกใบแจ้งหนี้ให้ร้านอาหาร

3.4.2 ส่วนที่ติดต่อกับฐานข้อมูล

ระบบหลังบ้านของ RMS มีการเชื่อมต่อกับระบบฐานข้อมูล MySQL ด้วย Spring Boot ซึ่งเป็น framework ที่เขียนด้วยภาษา Java และมีการ configuration ระบบต่างๆ ของ application ให้โดยอัตโนมัติเพียงแค่เพิ่ม JAR หรือ dependency เข้าไปใน pom.xml ตัว framework ก็จะทำการ autoconfiguration ระบบต่างๆ ให้พร้อมใช้งาน

โดยการเข้าถึงข้อมูลต่างๆ ในระบบฐานข้อมูลจะใช้ JPA module ที่เราเพิ่ม dependency ของระบบในการทำ ORM (Object Relation Mapping) เพื่อแปลงข้อมูลจากระบบฐานข้อมูลในรูปแบบของตารางให้มาอยู่ในรูปของ object เพื่อให้สามารถพัฒนาระบบการจัดการข้อมูลได้แบบ OOP (Object Oriented Programming)

3.4.3 รูปแบบข้อมูลในการสื่อสารระหว่างระบบ

เนื่องจากการทำระบบออกใบแจ้งหนี้อัตโนมัติจะมีการถูกเรียกใช้งานจากอีกหลายระบบ จึงต้องมีการออกแบบโครงสร้างของข้อมูลในการสื่อสารและ API ให้ชัดเจน ซึ่งในระบบนี้สามารถแบ่ง API ออกเป็น 3 ส่วนหลักๆ ดังนี้

- การจัดการที่อยู่ในการออกใบแจ้งหนี้ เช่น การสร้าง การค้นหาหรือการแก้ไขข้อมูลต่างๆ สามารถสรุปได้ดังนี้

ตารางที่ 3.1 รูปแบบการส่งข้อมูลการจัดการที่อยู่ในการออกใบแจ้งหนี้

Base Path: /v2/invoice-billing-address			
Path	Method	Request Body	Response

	POST	{
		"name": "นายวัสนี เจริญชัย",
		"email": "test@gmail.com",
		"address": "78 อาคารเกียรตินาคิน ซอยเจริญกรุง 30 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพฯ 10500",
		"taxId": "1237890",
		"userId": 57,
		"customerType": "JURISTIC",
		"branch": "00000",
		"phoneNumber": "0812345678"
		}
			{
			"id": 3
			}

/:id	GET		{
			"id": 3,
			"name": "นายวัสนี เจริญชัย",
			"email": "test@gmail.com",
			"address": "78 อาคารเกียรตินาคิน ซอยเจริญกรุง 30 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพฯ 10500",
			"taxId": "1237890",
			"userId": 57,
			"customerType": "JURISTIC",
			"branch": "00000",
			"phoneNumber": {
			"number": "+66812345678"
			}
			}

/:id	PATCH	{
		"name": "นายวัสนี เปี่ยมสุข"
		}
			{
			"id": 3,
			"name": "นายวัสนี เปี่ยมสุข",
			"email": "test@gmail.com",
			"address": "78 อาคารเกียรตินาคิน ซอยเจริญกรุง 30 ถนนเจริญกรุง แขวงบางรัก เขตบางรัก กรุงเทพฯ 10500",
			"taxId": "1237890",
			"userId": 57,
			"customerType": "JURISTIC",
			"branch": "00000",
			"phoneNumber": {
			"number": "+66812345678"
			}
			}

- การจัดการที่อยู่ในการออกใบแจ้งหนี้ของร้านอาหาร เช่น การเพิ่มหรือการเปลี่ยนที่อยู่ในการออกใบแจ้งหนี้ของร้านอาหารสามารถสรุปได้ดังนี้

ตารางที่ 3.2 รูปแบบการส่งข้อมูลการจัดการที่อยู่ในการออกใบแจ้งหนี้ของร้านอาหาร

Base Path: /v2/restaurant-to-invoice-billing-address			
Path	Method	Request Body	Response

	POST	{
		"leftId": 216871,
		"rightId": 1
		}
			{
			"leftId": 216871,
			"rightId": 1
			}

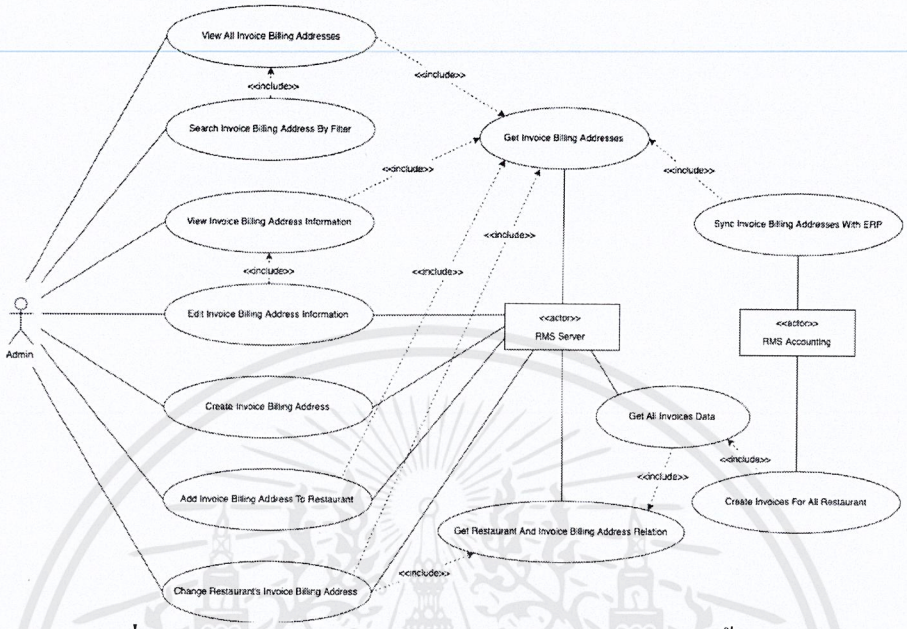
- การขอข้อมูลสำหรับการออกใบแจ้งหนี้ของร้านอาหารทั้งหมดในระบบสามารถสรุปได้ดังนี้

ตารางที่ 3.3 รูปแบบการขอข้อมูลสำหรับการออกใบแจ้งหนี้ของร้านอาหารในระบบ

Path: /v2/gp/revenue-report?year={ }&month={ }	
Method	Response
GET	[{ "restaurantId": 216871, "restaurantName": "test1", "restaurantBranchName": "Asoke", "billingAddressId": 1, "gpRevenueShare": 53.7, "totalIncome": 556.4, "excludeVatIncome": 520 }, { "restaurantId": 241573, "restaurantName": "test2", "restaurantBranchName": "Siam", "billingAddressId": null, "gpRevenueShare": 23.2, "totalIncome": 286.76, "excludeVatIncome": 268 }]

3.5 แผนภาพอธิบายโครงสร้างและการทำงานของระบบ

3.5.1 Use Case Diagrams



ภาพที่ 3.5 Use Case Diagram ของระบบการออกใบแจ้งหนี้อัตโนมัติ

จากแผนภาพ Use Case Diagram การทำงานของระบบข้างต้นสามารถเขียนอธิบายรายละเอียดต่างๆ ลงตารางได้ ดังนี้

ตารางที่ 3.4 Use Case: View All Invoice Billing Addresses

Use Case Title: View All Invoice Billing Addresses
Primary Actor: Admin
Brief Description: แสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ที่มีทั้งหมดในระบบ
Related Use Case: Get Invoice Billing Addresses
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Invoice Billing Address ที่ด้านซ้ายของหน้าเว็บ 3. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ขึ้นมาแสดงผล
Extension: 1. ไม่มีการแสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ในระบบ - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ

ตารางที่ 3.5 Use Case: Search Invoice Billing Address By Filter

Use Case Title: Search Invoice Billing Address By Filter
Primary Actor: Admin
Brief Description: ค้นหาและแสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้แบบ filter จากไอดี ชื่อหรือเลขประจำตัวผู้เสียภาษีของที่อยู่สำหรับใบแจ้งหนี้
Related Use Case: Get Invoice Billing Addresses
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Invoice Billing Address ที่ด้านซ้ายของหน้าเว็บ 3. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ขึ้นมาแสดงผล 4. ผู้ดูแลระบบกรอกข้อมูลเจาะจงที่ต้องการค้นหา 5. ผู้ดูแลระบบกดปุ่ม apply 6. ระบบแสดงผลข้อมูลที่อยู่สำหรับใบแจ้งหนี้ที่ตรงกับ keyword ที่ค้นหา
Extension: 1. ไม่มีการแสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ - ข้อมูลที่ค้นหาไม่ตรงกับข้อมูลที่อยู่สำหรับใบแจ้งหนี้ในระบบ

ตารางที่ 3.6 Use Case: View Invoice Billing Address Information

Use Case Title: View Invoice Billing Address Information
Primary Actor: Admin
Brief Description: แสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ของผู้ใช้งาน
Related Use Case: Get Invoice Billing Addresses
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Invoice Billing Address ที่ด้านซ้ายของหน้าเว็บ 3. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ขึ้นมาแสดงผล 4. ผู้ดูแลระบบกดปุ่ม edit ที่ข้อมูลที่อยู่ที่ต้องการ 5. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ที่เลือกขึ้นมาแสดง

Extension: 1. ไม่มีการแสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ที่เลือก - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ

ตารางที่ 3.7 Use Case: Edit Invoice Billing Address Information

Use Case Title: Edit Invoice Billing Address Information
Primary Actor: Admin
Brief Description: แก้ไขข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ที่เลือก
Related Use Case: View Invoice Billing Address Information
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Invoice Billing Address ที่ด้านซ้ายของหน้าเว็บ 3. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ขึ้นมาแสดงผล 4. ผู้ดูแลระบบกดปุ่ม edit ที่ข้อมูลที่อยู่ที่ต้องการ 5. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ที่เลือกขึ้นมาแสดง 6. ผู้ดูแลระบบแก้ไขข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ 7. ผู้ดูแลระบบกดปุ่ม save เพื่อบันทึกข้อมูลสู่ระบบ 8. ระบบแสดง success message เพื่อบอกผู้ใช้ว่าทำงานสำเร็จ
Extension: 1. ไม่มีการแสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ที่เลือก - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ 2. ระบบแจ้งเตือน error message ผู้ใช้งานไม่สามารถบันทึกข้อมูลได้ - ผู้ใช้งานกรอกข้อมูลไม่ถูกต้อง - ระบบไม่สามารถเชื่อมต่อกับ RMS Server ได้

ตารางที่ 3.8 Use Case: Create Invoice Billing Address

Use Case Title: Create Invoice Billing Address
Primary Actor: Admin
Brief Description: สร้างที่อยู่สำหรับใบแจ้งหนี้และบันทึกเข้าสู่ระบบ

Related Use Case: -
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Invoice Billing Address ที่ด้านซ้ายของหน้าเว็บ 3. ระบบดึงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ขึ้นมาแสดงผล 4. ผู้ดูแลระบบเลือกแถบ create ที่ด้านบนของหน้าเว็บ 5. ผู้ดูแลระบบกรอกข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ 6. ผู้ดูแลระบบกดปุ่ม create เพื่อบันทึกข้อมูลสู่ระบบ 7. ระบบแสดง success message เพื่อบอกผู้ใช้ว่าทำงานสำเร็จ
Extension: 1. ไม่มีการแสดงผลข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ที่เลือก - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ 2. ระบบแจ้งเตือน error message ผู้ใช้งานไม่สามารถบันทึกข้อมูลได้ - ผู้ใช้งานกรอกข้อมูลไม่ถูกต้อง - ระบบไม่สามารถเชื่อมต่อกับ RMS Server ได้

ตารางที่ 3.9 Use Case: Add Invoice Billing Address To Restaurant

Use Case Title: Add Invoice Billing Address To Restaurant
Primary Actor: Admin
Brief Description: เพิ่มข้อมูลที่อยู่สำหรับใบแจ้งหนี้ให้ร้านอาหาร เพื่อใช้ที่อยู่นั้นในการออกใบแจ้งหนี้ให้ร้านอาหาร
Related Use Case: Get Invoice Billing Addresses
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Manage Restaurant ใน group ของ Restaurant 3. ระบบดึงข้อมูลร้านอาหารที่มีอยู่ในระบบขึ้นมาแสดงผล 4. ผู้ดูแลระบบเลือกร้านอาหารที่ต้องการ 5. ระบบดึงข้อมูลของร้านอาหารที่เลือกขึ้นมาแสดงผล 6. ผู้ดูแลระบบเลือกที่อยู่สำหรับใบแจ้งหนี้ในช่อง Billing Address ID 7. ผู้ดูแลระบบกดปุ่ม save เพื่อบันทึกข้อมูลสู่ระบบ 8. ระบบแสดง success message เพื่อบอกผู้ใช้ว่าทำงานสำเร็จ

Extension: 1. ไม่มีการแสดงผลข้อมูลของร้านอาหารที่เลือก - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ 2. ระบบแจ้งเตือน error message ผู้ใช้งานไม่สามารถบันทึกข้อมูลได้ - ระบบไม่สามารถเชื่อมต่อกับ RMS Server ได้

ตารางที่ 3.10 Use Case: Change Restaurant's Invoice Billing Address

Use Case Title: Change Restaurant's Invoice Billing Address
Primary Actor: Admin
Brief Description: แก้ไขข้อมูลที่อยู่สำหรับใบแจ้งหนี้ให้ร้านอาหาร เพื่อใช้ที่อยู่นั้นในการออกใบแจ้งหนี้ให้ร้านอาหาร
Related Use Case: Get Invoice Billing Addresses, Get Restaurant And Invoice Billing Address Information Relation
Main Success Scenario: 1. ผู้ดูแลระบบเข้าสู่หน้าหลักของ Admin RMS 2. ผู้ดูแลระบบเลือกแถบ Manage Restaurant ใน group ของ Restaurant ที่ด้านซ้ายของหน้าเว็บ 3. ระบบดึงข้อมูลร้านอาหารที่มีอยู่ในระบบขึ้นมาแสดงผล 4. ผู้ดูแลระบบเลือกร้านอาหารที่ต้องการ 5. ระบบดึงข้อมูลของร้านอาหารที่เลือกขึ้นมาแสดงผล 6. ผู้ดูแลระบบเลือกที่อยู่สำหรับใบแจ้งหนี้ในช่อง Billing Address ID 7. ผู้ดูแลระบบกดปุ่ม save เพื่อบันทึกข้อมูลสู่ระบบ 8. ระบบแสดง success message เพื่อบอกผู้ใช้ว่าทำงานสำเร็จ
Extension: 1. ไม่มีการแสดงผลข้อมูลของร้านอาหารที่เลือก - ระบบดึงข้อมูลจาก RMS Server หรือฐานข้อมูลไม่สำเร็จ 2. ระบบแจ้งเตือน error message ผู้ใช้งานไม่สามารถบันทึกข้อมูลได้ - ระบบไม่สามารถเชื่อมต่อกับ RMS Server ได้

ตารางที่ 3.11 Use Case: Get Invoice Billing Addresses

Use Case Title: Get Invoice Billing Addresses
Primary Actor: RMS Server
Brief Description: ดึงข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ในระบบฐานข้อมูลโดยอาจมีการ query ข้อมูลที่ตรงกับความต้องการออกมาและส่งออกในรูปแบบ JSON
Related Use Case: -
Main Success Scenario: 1. ระบบส่งข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ออกมาในรูปแบบ JSON หลังจากได้รับ request
Extension: 1. ระบบส่งเป็น error package ออกมา - token ของ request ที่เข้ามาไม่ถูกต้อง - path ที่ใส่มาไม่ถูกต้อง

ตารางที่ 3.12 Use Case: Get Restaurant And Invoice Billing Address Information Relation

Use Case Title: Get Restaurant And Invoice Billing Address Information Relation
Primary Actor: RMS Server
Brief Description: ดึงข้อมูลความสัมพันธ์ระหว่างร้านอาหารและข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ในระบบฐานข้อมูลโดยอาจมีการ query ข้อมูลที่ตรงกับความต้องการออกมาและส่งออกในรูปแบบ JSON
Related Use Case: -
Main Success Scenario: 1. ระบบส่งข้อมูลความสัมพันธ์ระหว่างร้านอาหารและข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ออกมาในรูปแบบ JSON หลังจากได้รับ request
Extension: 2. ระบบส่งเป็น error package ออกมา - token ของ request ที่เข้ามาไม่ถูกต้อง - path ที่ใส่มาไม่ถูกต้อง

ตารางที่ 3.13 Use Case: Get All Invoices Data

Use Case Title: Get All Invoices Data
Primary Actor: RMS Server
Brief Description: ส่งออกข้อมูลที่จำเป็นสำหรับการสร้างใบแจ้งหนี้ คือ ไอดี ชื่อและสาขาของร้านอาหาร หมายเลขไอดีของที่อยู่สำหรับใบแจ้งหนี้ที่ตั้งไว้ ค่าบริการ GP ที่ร้านจะต้องจ่าย รายได้ทั้งหมด และรายได้รวมแบบหักภาษีของร้านในเดือนนั้นๆ ออกมาในรูปแบบ JSON
Related Use Case: Get Restaurant And Invoice Billing Address Information Relation
Main Success Scenario: 1. ระบบส่งข้อมูลสำหรับการสร้างใบแจ้งหนี้ออกมาในรูปแบบ JSON หลังจากได้รับ request
Extension: 1. ระบบส่งเป็น error package ออกมา - token ของ request ที่เข้ามาไม่ถูกต้อง - path ที่ใส่มาไม่ถูกต้อง

ตารางที่ 3.14 Use Case: Sync Invoice Billing Address With ERP

Use Case Title: Sync Invoice Billing Address With ERP
Primary Actor: RMS Accounting
Brief Description: ทำการเชื่อมต่อข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ในระบบ RMS กับ ERP เข้าด้วยกัน และปรับข้อมูลให้ตรงกันทั้ง 2 ระบบ
Related Use Case: Get Invoice Billing Addresses
Main Success Scenario: 1. ระบบส่ง request ขอข้อมูลที่อยู่สำหรับใบแจ้งหนี้ที่มีทั้งหมดในระบบของ RMS 2. ระบบนำข้อมูลที่ได้ไปสร้างหรืออัปเดตข้อมูลที่อยู่สำหรับใบแจ้งหนี้ในระบบ ERP
Extension: 1. เชื่อมต่อข้อมูลไม่สำเร็จ - ระบบไม่สามารถเชื่อมต่อกับ RMS Server ได้ - ระบบไม่สามารถเชื่อมต่อกับ ERP ได้

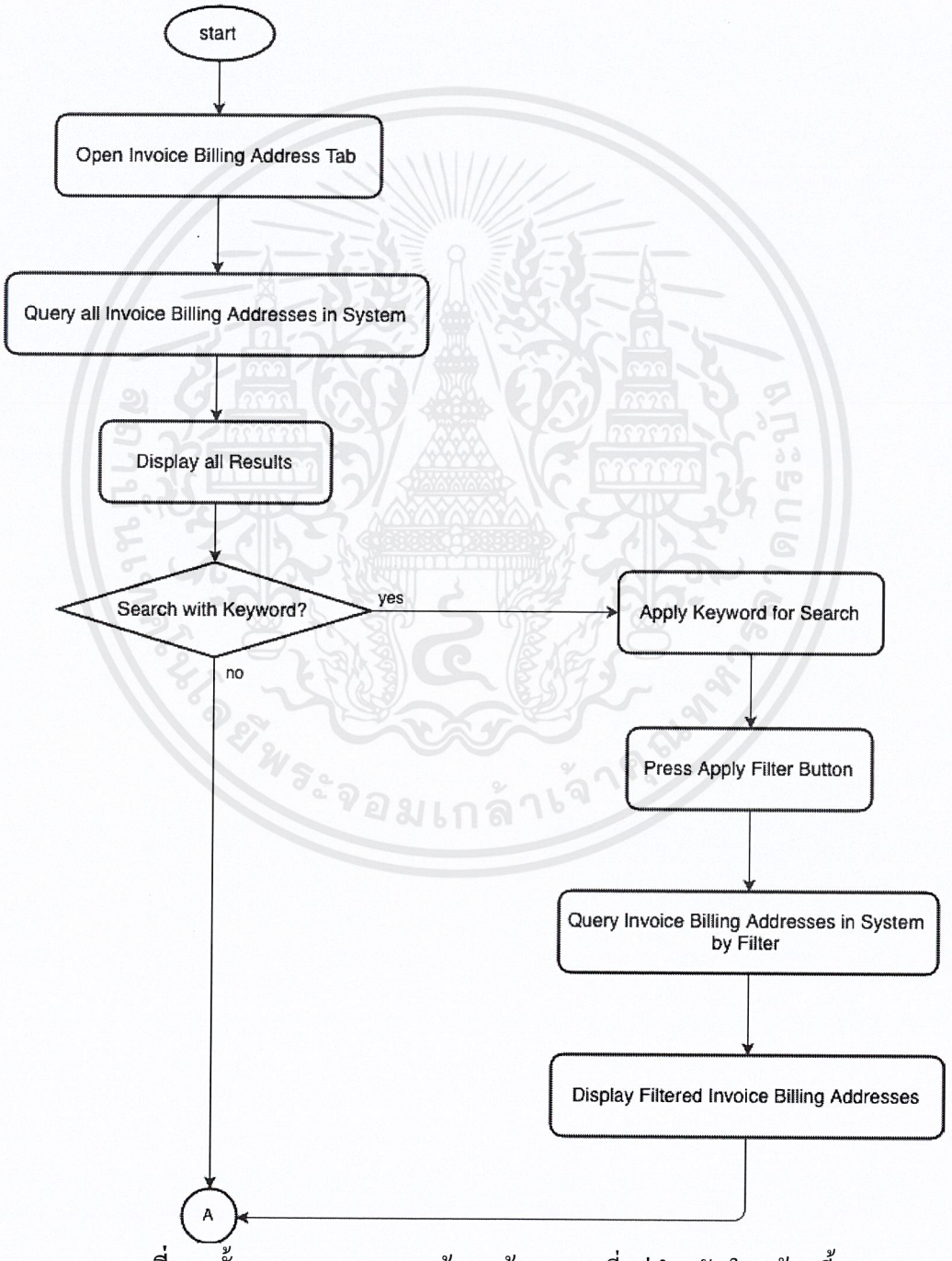
ตารางที่ 3.15 Use Case: Create Invoices For All Restaurant

Use Case Title: Create Invoices For All Restaurant
Primary Actor: RMS Accounting
Brief Description: สร้างใบแจ้งหนี้อัตโนมัติให้ร้านอาหารโดยรับข้อมูลในการสร้างใบแจ้งหนี้จากระบบ RMS แล้วส่งต่อให้ระบบ ERP สร้างใบแจ้งหนี้และบันทึกลงระบบ
Related Use Case: Get All Invoices Data
Main Success Scenario: 1. ระบบส่ง request ขอข้อมูลสำหรับการสร้างใบแจ้งหนี้จากระบบของ RMS 2. ระบบนำข้อมูลที่ได้ส่งต่อให้ไปสร้างใบแจ้งหนี้ในระบบ ERP
Extension: 1. สร้างใบแจ้งหนี้ไม่สำเร็จ - ระบบไม่สามารถเชื่อมต่อกับ RMS Server ได้ - ระบบไม่สามารถเชื่อมต่อกับ ERP ได้

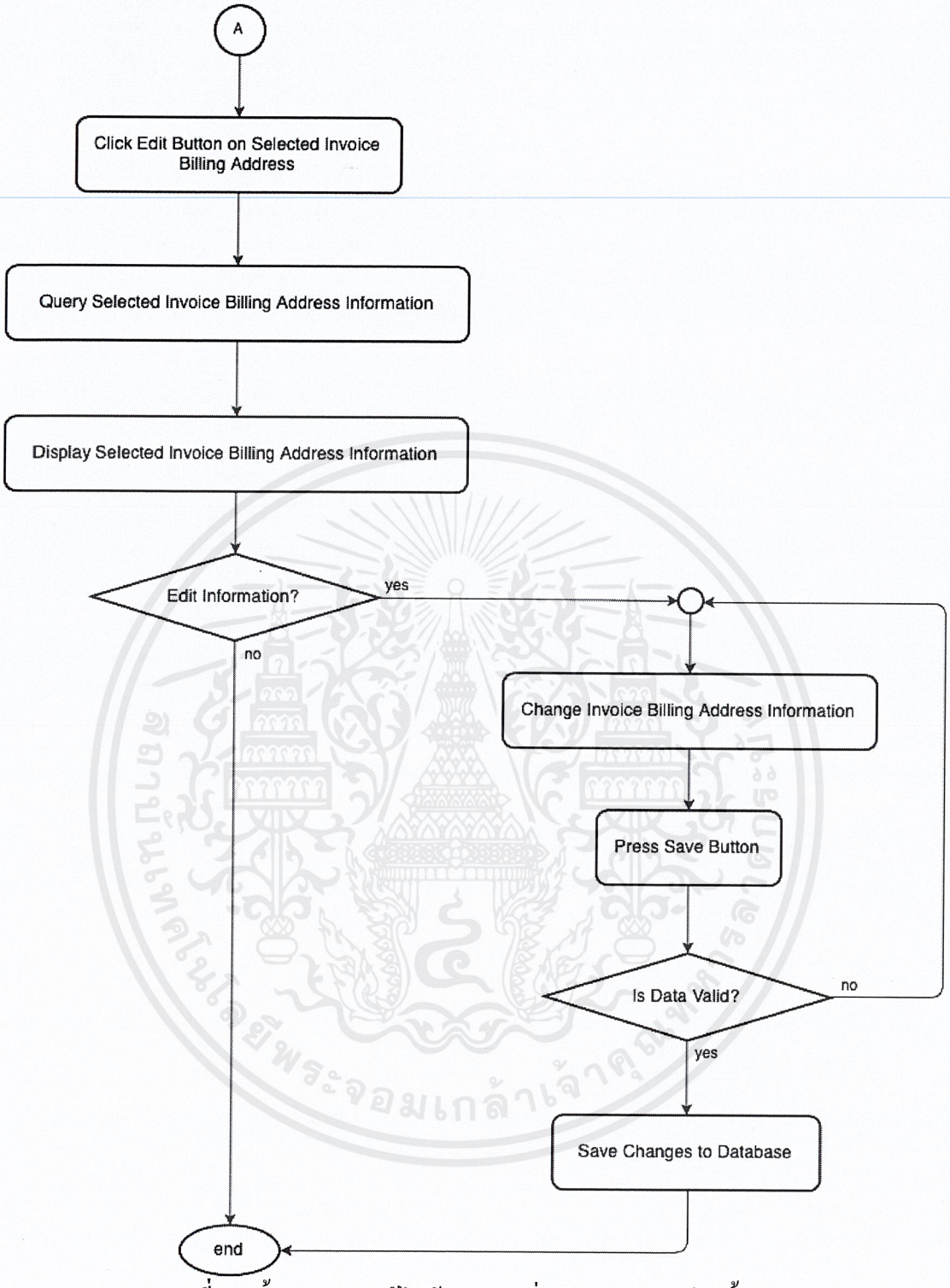
3.5.2 Flowcharts

จาก use case diagram และตารางอธิบายทั้งหมดข้างต้น ระบบมีส่วนที่เป็น user interface เพียงอย่างเดียวคือ Admin RMS และตัว Admin RMS นั้นก็ไม่สามารถเข้าถึงการทำงานทุกส่วนของระบบได้ จึงจัดทำแผนผังการทำงานได้เพียงบางส่วน ดังนี้

1) การแสดงผล ค้นหาและแก้ไขข้อมูลของ Invoice Billing Address มีแผนผังการทำงานผ่านหน้าการจัดการของ Admin RMS ดังนี้

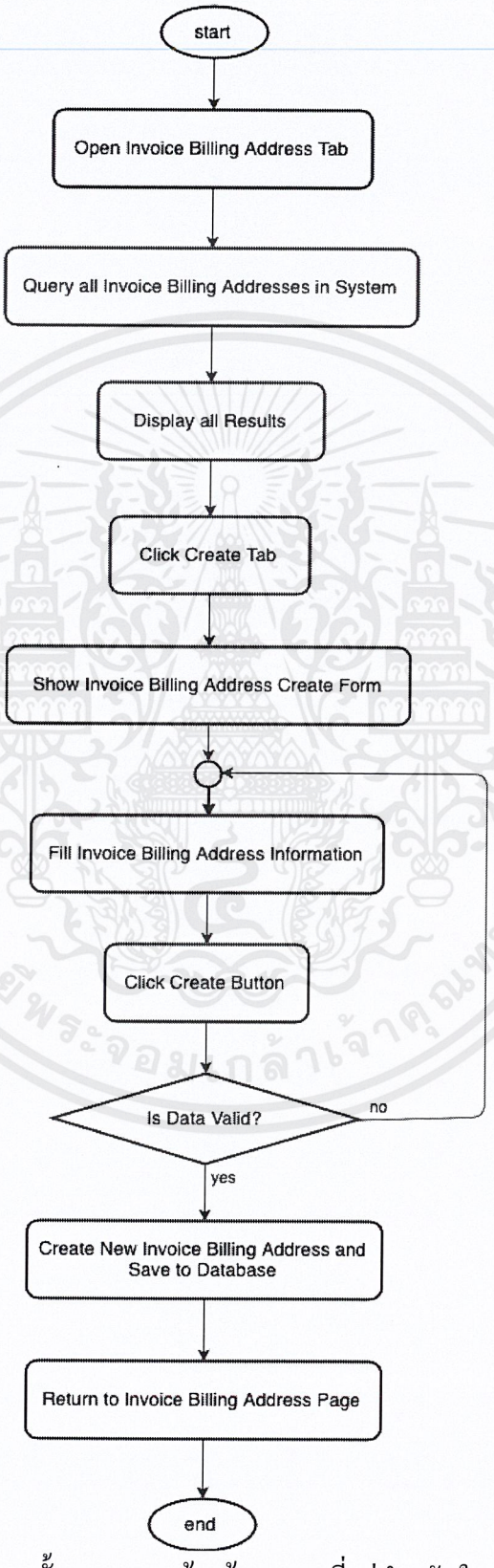


ภาพที่ 3.6 ขั้นตอนการแสดงผล ค้นหาข้อมูลของที่อยู่สำหรับใบแจ้งหนี้



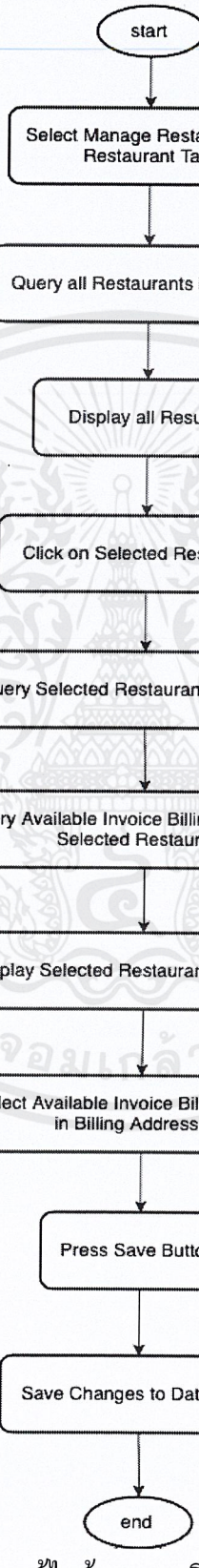
ภาพที่ 3.7 ขั้นตอนการแก้ไขข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ (ต่อ)

2) การสร้างและบันทึกข้อมูล Invoice Billing Address มีแผนผังการทำงานผ่านหน้าการจัดการของ Admin RMS ดังนี้



ภาพที่ 3.8 ขั้นตอนการสร้างข้อมูลของที่อยู่สำหรับใบแจ้งหนี้

3) การเพิ่มและแก้ไขข้อมูล Invoice Billing Address ของร้านอาหาร มีแผนผังการทำงานผ่านหน้าการจัดการของ Admin RMS ดังนี้

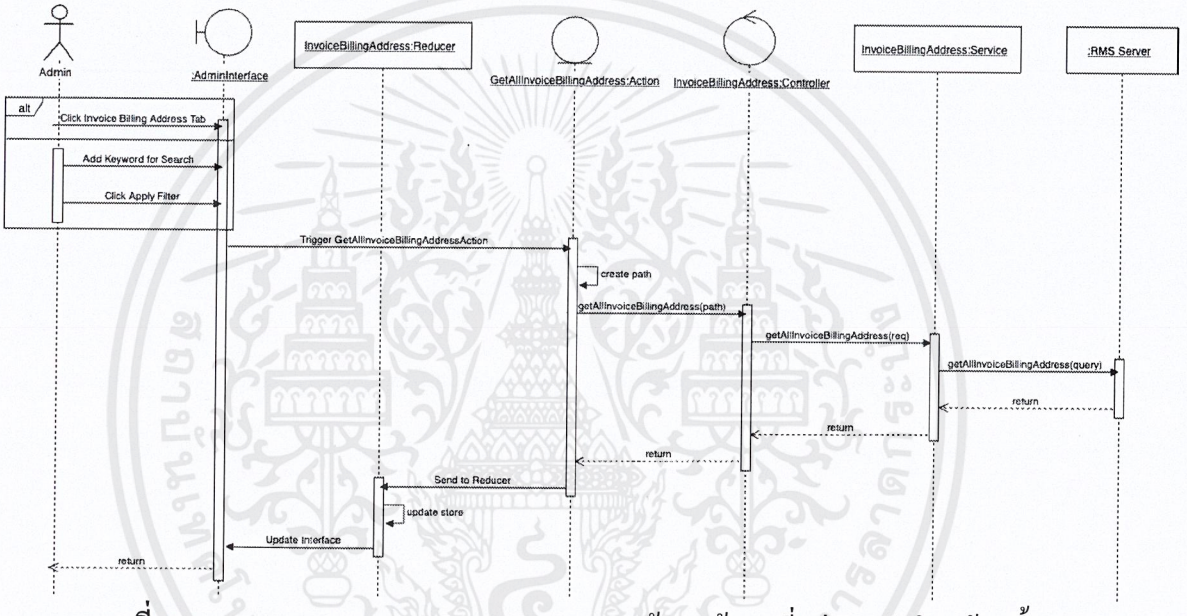


ภาพที่ 3.9 ขั้นตอนการเพิ่มและแก้ไขข้อมูลของที่อยู่สำหรับใบแจ้งหนี้ของร้านอาหาร

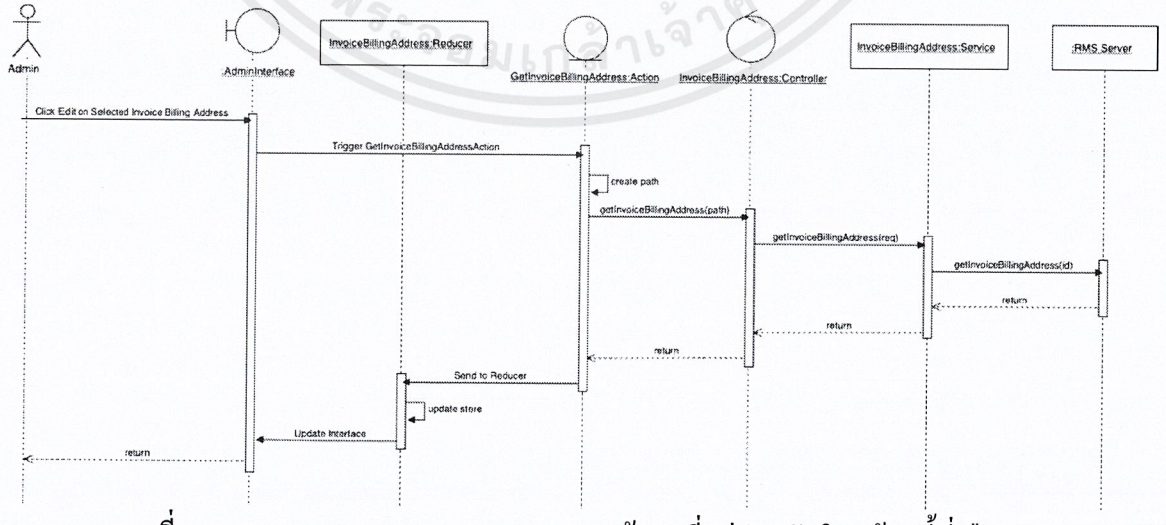
3.5.3 Sequence Diagrams

เนื่องจากระบบออกใบแจ้งหนี้อัตโนมัติมีขั้นตอนการทำงานที่หลากหลายขั้นตอน มีโครงสร้างและการสื่อสารส่งข้อมูลไปมาในหลายๆ project ดังนั้นจึงได้จัดทำ sequence diagram ที่จะอธิบายขั้นตอนการทำงานของระบบตามลำดับเวลาออกมาโดยแบ่งตามลักษณะโครงสร้างของ project ได้ ดังนี้

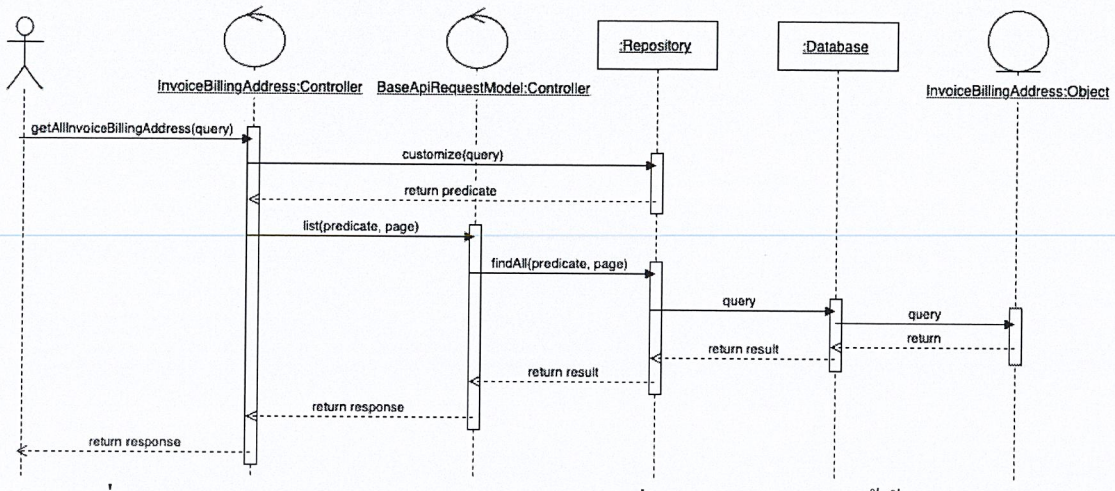
1) การแสดงและการค้นหาข้อมูลที่อยู่สำหรับใบแจ้งหนี้ มีขั้นตอนการทำงานแบ่งเป็นฝั่งของระบบหน้าการจัดการของ Admin RMS และระบบของ RMS Server ได้ตามลำดับ ดังนี้



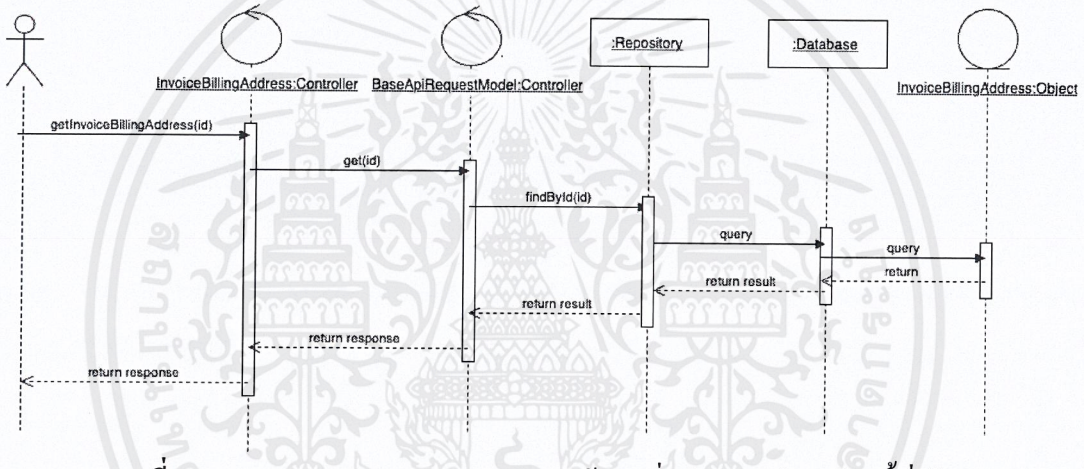
ภาพที่ 3.10 Sequence Diagram ของการแสดงและค้นหาข้อมูลที่อยู่สำหรับใบแจ้งหนี้



ภาพที่ 3.11 Sequence Diagram ของการแสดงข้อมูลที่อยู่สำหรับใบแจ้งหนี้ที่เลือก

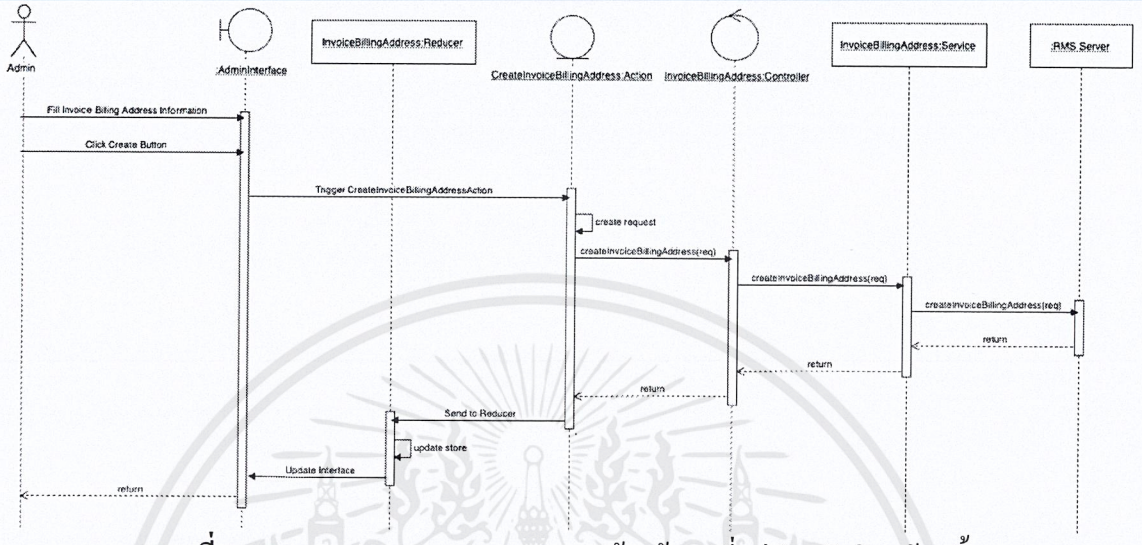


ภาพที่ 3.12 Sequence Diagram ของการค้นหาข้อมูลที่อยู่สำหรับใบแจ้งหนี้ทั้งหมดในระบบ

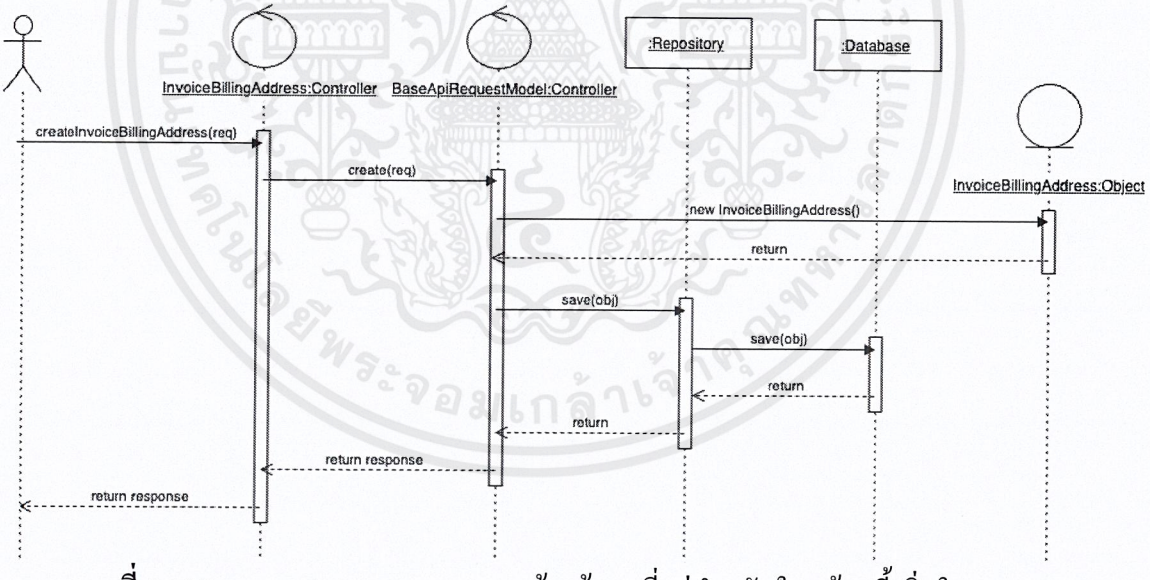


ภาพที่ 3.13 Sequence Diagram ของการขอข้อมูลที่อยู่สำหรับใบแจ้งหนี้ที่เลือก

2) การสร้างและบันทึกข้อมูลที่อยู่สำหรับใบแจ้งหนี้ มีขั้นตอนการทำงานแบ่งเป็นฝั่งของระบบหน้าการจัดการของ Admin RMS และระบบของ RMS Server ได้ตามลำดับดังนี้

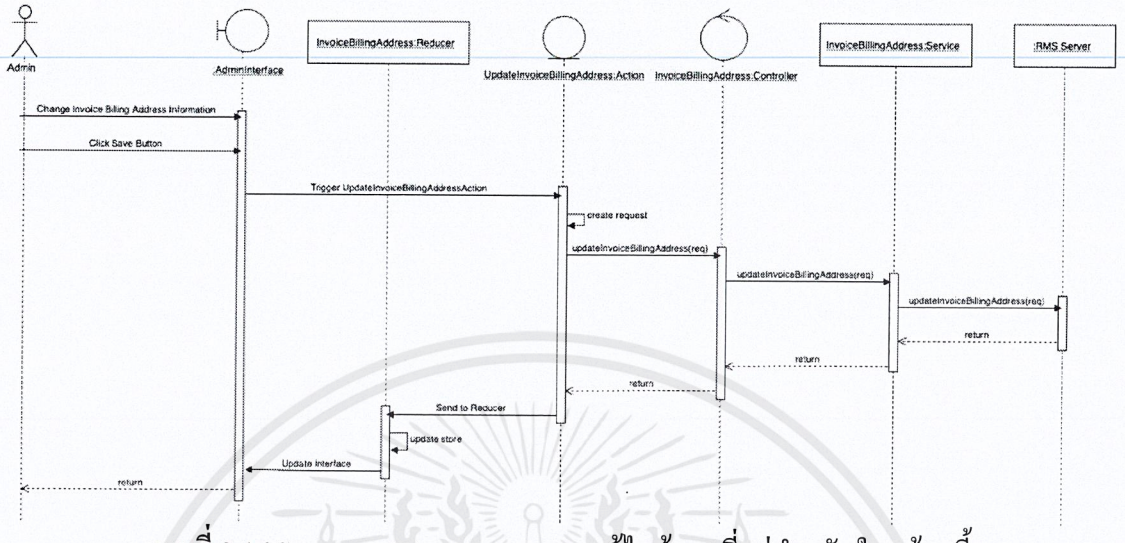


ภาพที่ 3.14 Sequence Diagram ของการสร้างข้อมูลที่อยู่สำหรับใบแจ้งหนี้

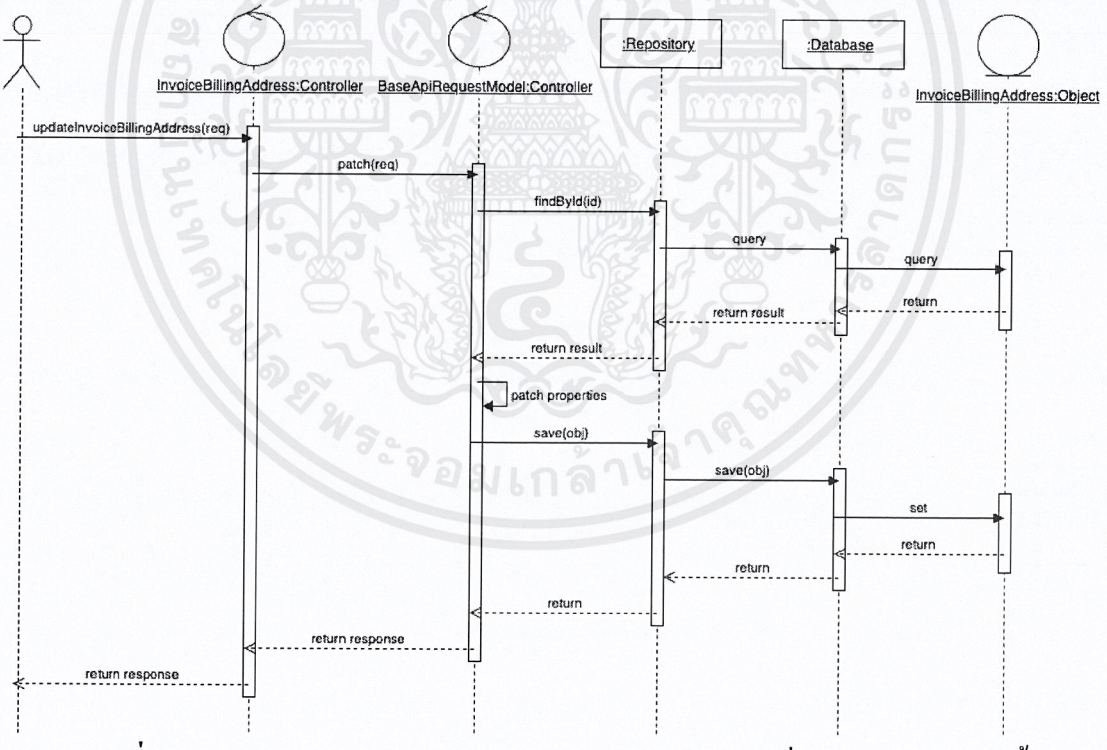


ภาพที่ 3.15 Sequence Diagram ของการสร้างข้อมูลที่อยู่สำหรับใบแจ้งหนี้เพิ่มในระบบ

3) การแก้ไขข้อมูลที่อยู่สำหรับใบแจ้งหนี้ มีขั้นตอนการทำงานแบ่งเป็นฝั่งของระบบหน้าการจัดการของ Admin RMS และระบบของ RMS Server ได้ตามลำดับ ดังนี้

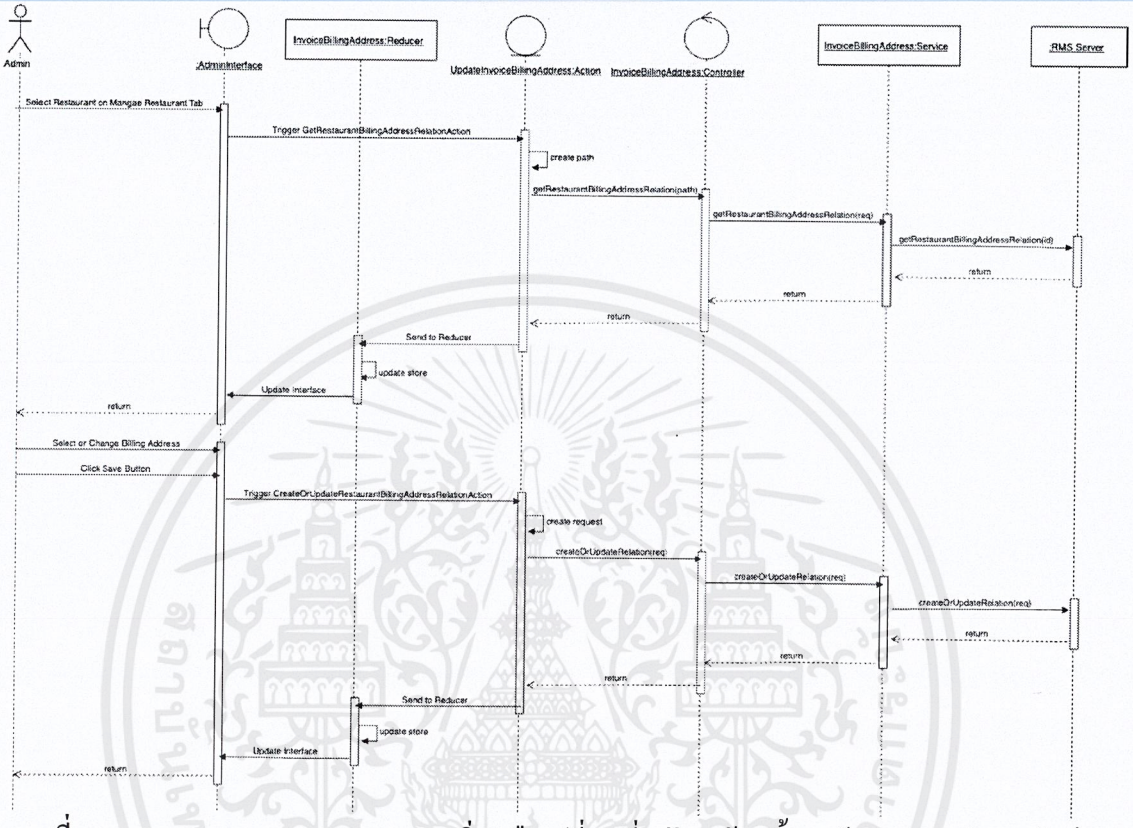


ภาพที่ 3.16 Sequence Diagram ของการแก้ไขข้อมูลที่อยู่สำหรับใบแจ้งหนี้

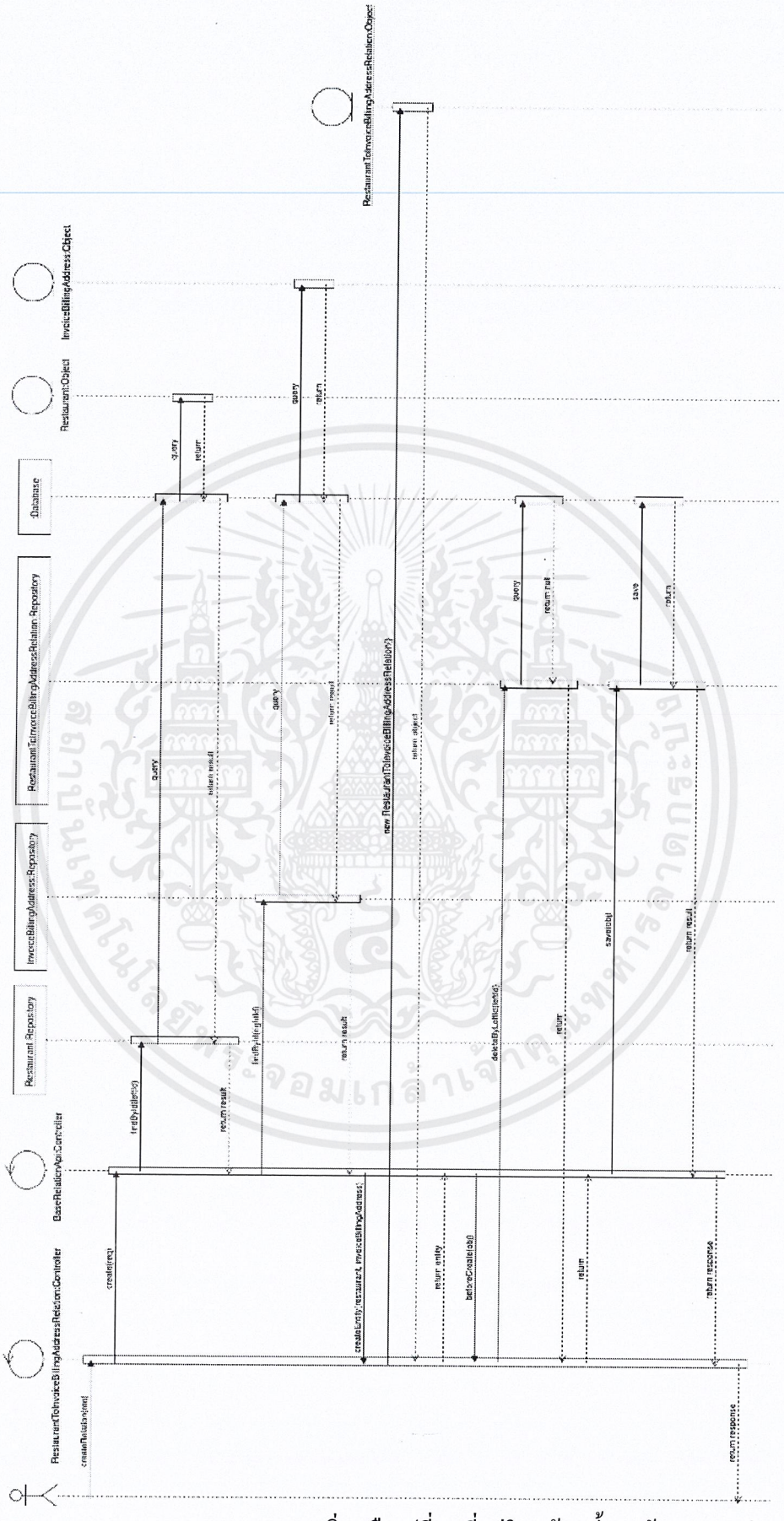


ภาพที่ 3.17 Sequence Diagram ของการแก้ไขและบันทึกข้อมูลที่อยู่สำหรับใบแจ้งหนี้

4) การเพิ่มและการเปลี่ยนที่อยู่สำหรับใบแจ้งหนี้ของร้านอาหาร มีขั้นตอนการทำงานแบ่งเป็นฝั่งของระบบหน้าการจัดการของ Admin RMS และระบบของ RMS Server ได้ตามลำดับ ดังนี้



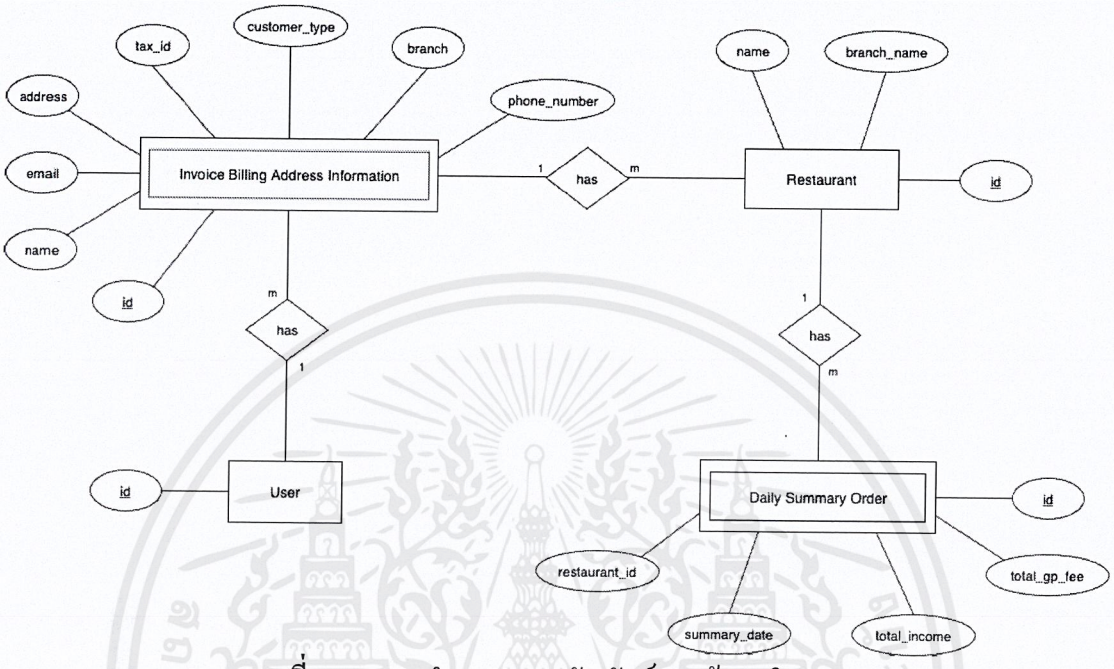
ภาพที่ 3.18 Sequence Diagram ของการเพิ่มหรือเปลี่ยนที่อยู่ใบแจ้งหนี้ของร้านอาหารผ่านหน้าเว็บ



ภาพที่ 3.19 Sequence Diagram ของการเพิ่มหรือเปลี่ยนที่อยู่ใบแจ้งหนี้ของร้านอาหารในระบบ

3.5.4 Entity Relational Diagrams

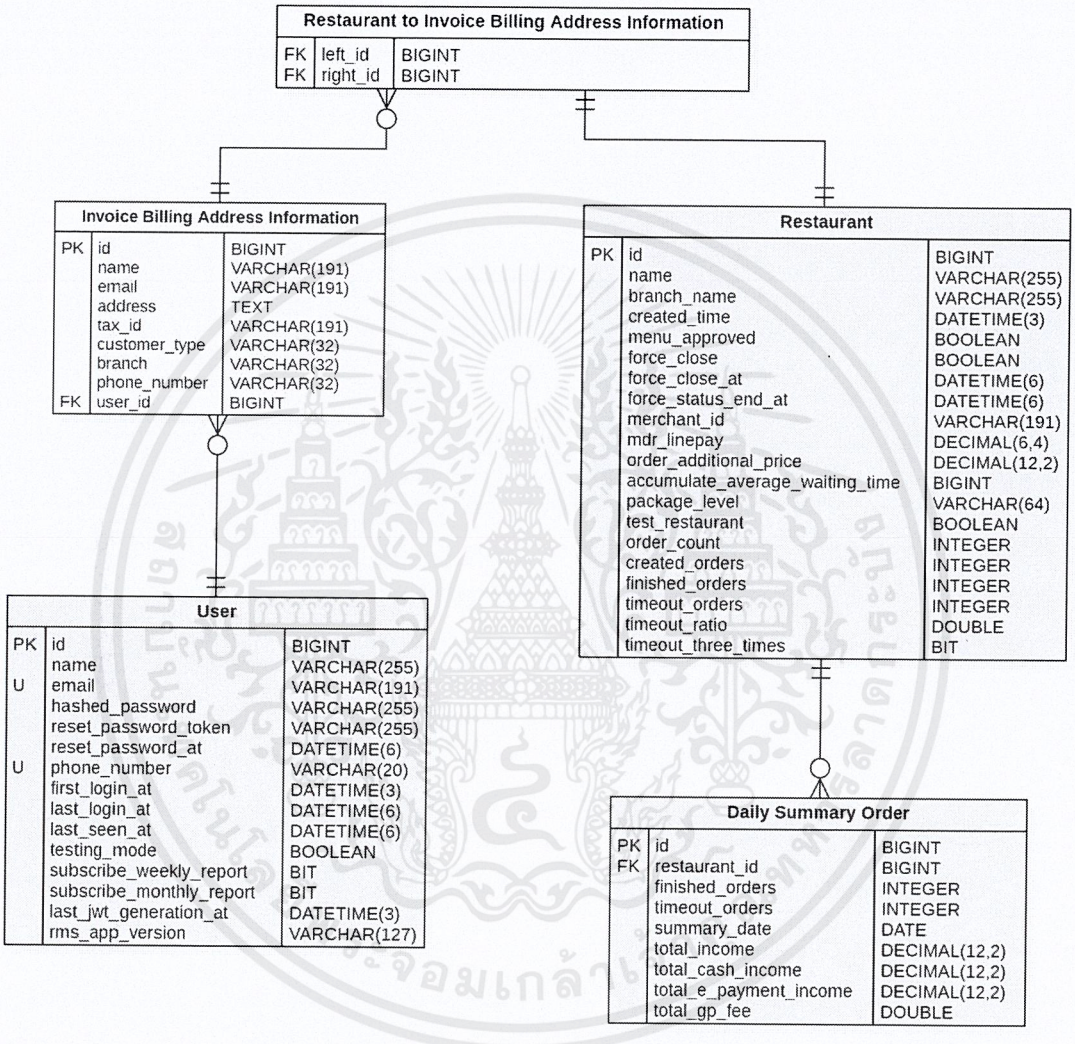
ในการออกแบบความสัมพันธ์ของข้อมูลในระบบสามารถทำ ER Diagram จำลองโครงสร้างออกมาในลักษณะของรูปภาพได้คร่าวๆ ดังนี้



ภาพที่ 3.20 แบบจำลองความสัมพันธ์ของข้อมูลในระบบ

3.5.5 Database Schema Diagrams

ในการออกแบบฐานข้อมูลสำหรับการเก็บที่อยู่ในการออกใบแจ้งหนี้นั้น ได้มีการเพิ่มตารางใหม่ขึ้นมา 2 ตาราง คือตารางเก็บที่อยู่ในการออกใบแจ้งหนี้และตารางที่เก็บความสัมพันธ์ระหว่างร้านอาหารและที่อยู่ใบแจ้งหนี้โดยทั้งหมดมีโครงสร้าง ดังนี้



ภาพที่ 3.21 โครงสร้างของฐานข้อมูลที่เกี่ยวข้องกับระบบ

จากโครงสร้างของฐานข้อมูลที่เกี่ยวข้องกับระบบในภาพที่ 3.21 สามารถแสดงรายละเอียดและโครงสร้างของตารางที่จำเป็นต้องใช้ในการออกใบแจ้งหนี้ได้ ดังนี้

- ตาราง Invoice Billing Address Information ใช้สำหรับเก็บที่อยู่ที่จำเป็นในการออกใบแจ้งหนี้ มีรายละเอียดและโครงสร้าง ดังนี้

ตารางที่ 3.16 โครงสร้างของตาราง Invoice Billing Address Information

Field	Key	Datatype	Description
id	PK	BIGINT	Unique id
name		VARCHAR	ชื่อของที่อยู่สำหรับใบแจ้งหนี้
email		VARCHAR	email สำหรับใบแจ้งหนี้
address		TEXT	ที่อยู่สำหรับใบแจ้งหนี้
tax_id		VARCHAR	เลขประจำตัวผู้เสียภาษี
customer_type		VARCHAR	บุคคลธรรมดา/นิติบุคคล
branch		VARCHAR	หมายเลขสาขา(นิติบุคคล)
phone_number		VARCHAR	หมายเลขโทรศัพท์
user_id	FK	BIGINT	หมายเลขไอดีของผู้ใช้งาน
Indexes			
Keyname	Unique	Sequence in Index	Column Name
PRIMARY	Yes	1	id
fk_invoice_billing_addres_user	No	1	user_id

- ตาราง Restaurant To Invoice Billing Address Information ใช้สำหรับเก็บความสัมพันธ์ของที่อยู่ในการออกใบแจ้งหนี้และร้านอาหาร มีรายละเอียดและโครงสร้าง ดังนี้

ตารางที่ 3.17 โครงสร้างของตาราง Restaurant To Invoice Billing Address Information

Field	Key	Datatype	Description
left_id	FK	BIGINT	หมายเลขไอดีของร้านอาหาร
right_id	FK	BIGINT	หมายเลขไอดีของที่อยู่สำหรับใบแจ้งหนี้
Indexes			
Keyname	Unique	Sequence in Index	Column Name
PRIMARY	Yes	1	left_id
		2	right_id
restaurant_fk	No	1	left_id
invoice_billing_address_fk	No	1	right_id

- ตาราง Restaurant ที่มีอยู่ในระบบอยู่แล้ว ทำหน้าที่เก็บข้อมูลรายละเอียดต่างๆ ของร้านอาหารในระบบ มีรายละเอียดที่สำคัญ ดังนี้

ตารางที่ 3.18 รายละเอียดที่สำคัญในตาราง Restaurant

Field	Key	Datatype	Description
id	PK	BIGINT	Unique id
name		VARCHAR	ชื่อของร้านอาหาร
branch_name		VARCHAR	ชื่อสาขาของร้านอาหาร

- ตาราง Daily Summary Order ที่มีอยู่ในระบบอยู่แล้ว ทำหน้าที่เก็บสรุปจำนวนยอดขายของร้านอาหารในแต่ละวัน มีรายละเอียดที่สำคัญ ดังนี้

ตารางที่ 3.19 รายละเอียดที่สำคัญในตาราง Daily Summary Order

Field	Key	Datatype	Description
id	PK	BIGINT	Unique id
restaurant_id	FK	BIGINT	หมายเลขไอดีของร้านอาหาร
summary_date		DATE	วันเดือนปีที่สรุปยอด
total_income		DECIMAL	ยอดขายทั้งหมดของวันที่สรุปยอด
total_gp_fee		DOUBLE	ค่าบริการ GP ที่คำนวณจากยอดขาย

บทที่ 4
ผลการวิจัย

ในการพัฒนาระบบ Wongnai Delivery และ RMS นั้นได้มีการเพิ่มฟีเจอร์ใหม่ๆ และการปรับปรุงการทำงานของระบบมากมาย เช่น การเปลี่ยน search algorithm ของระบบให้สามารถรองรับร้านอาหารในพื้นที่บริการใหม่ เพิ่มฟีเจอร์การจัดเรียงโปรโมชั่นของร้านอาหารตามระยะทาง แก้ไขปัญหาที่ผู้ใช้งานสร้างคำสั่งซื้ออาหารไม่ได้หรือการเพิ่ม scheduler ให้แก้ไขข้อมูลในระบบแบบอัตโนมัติ เป็นต้น โดยทั้งหมดนี้ก็ให้ผลลัพธ์เป็นที่พอใจแก่ผู้ใช้งาน การทำงานหลายๆ ส่วนที่ต้องใช้เวลาและกำลังคนมาจัดการก็ถูกเปลี่ยนให้เป็นหน้าที่ของระบบ ทำให้พนักงานสามารถไปจัดการงานส่วนอื่นๆ ได้มากยิ่งขึ้น ระบบทำงานได้อย่างถูกต้องและมีประสิทธิภาพมากขึ้น ปัญหาที่ผู้ใช้งานพบเจอก็ถูกแก้ไขให้ใช้งานได้ปกติ การปรับปรุงบางส่วนก็สามารถช่วยในเรื่องของการกระตุ้นยอดขาย สามารถเพิ่มรายได้ให้กับบริษัทได้เช่นกัน

ทั้งหมดนี้อาจเป็นผลลัพธ์เล็กๆ ที่ดูไม่สำคัญ แต่เมื่อนำสิ่งเล็กๆ เหล่านี้มารวมกันก็สามารถส่งผลต่อภาพรวมของระบบให้ดีขึ้นได้เช่นกัน โดยจากนี้จะยกตัวอย่างงานและผลลัพธ์ที่ได้พัฒนาไว้ในระบบมาแสดง ดังนี้

- การพัฒนาระบบออกใบแจ้งหนี้อัตโนมัติให้ร้านอาหารในทุกๆ เดือน

ลูกค้า

บริษัท เจ๊หอย จำกัด (สำนักงานใหญ่)
119/395 หมู่ที่ 8 หมู่บ้านเพื่องสุข 6 ตำบลลำไพ อำเภอบางบัวทอง นนทบุรี 11110
เลขประจำตัวผู้เสียภาษี: 0125562007699
อ้างอิง: -

ผู้ขาย : Orh - Marisa Danpitakkul
เลขที่ใบเสนอราคา : QT2019091255 วันที่ 30/09/2019
เลขที่ใบสั่งขาย : SO2019090976 วันที่ 30/09/2019

#	รายละเอียด	จำนวน	ราคาต่อหน่วย	ยอดเรียกเก็บ
1	[LMTHI044] GP Revenue Share - Merchant GP Revenue Share - Merchant Period: May 2019	1.00	5,877.00	5,877.00

(หกพันสองร้อยแปดสิบแปดบาทสามสิบเก้าสตางค์)

สรุปยอด

รวมเป็นเงิน	5,877.00
ส่วนลด	0.00
หลังหักส่วนลด	5,877.00
ภาษีมูลค่าเพิ่ม 7%	411.39
จำนวนเงินรวมทั้งสิ้น	6,288.39

หมายเหตุ :
- ถ้าลูกค้าต้องการแก้ไขรายละเอียดตามเอกสารฉบับนี้ ต้องทำการแจ้งต่อบริษัทภายใน 7 วันหลังจากที่ได้รับเอกสารฉบับนี้

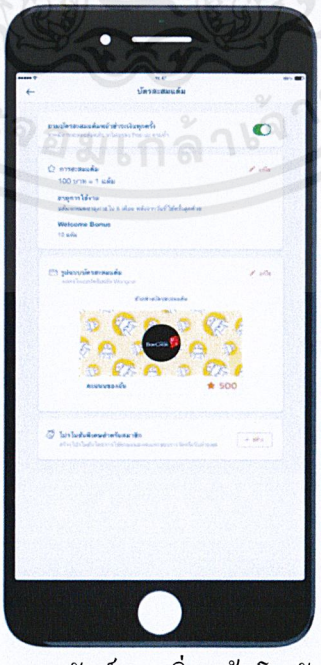
ชำระผ่าน : ธนาคารไทยพาณิชย์
ชื่อบัญชี : บจก. วงใน มีเดีย (บัญชีออมทรัพย์ 250-2-12265-1 หรือ บัญชีกระแสรายวัน 250-3-00240-6)
(กรุณาส่งหลักฐานการชำระเงินพร้อมกับเลขที่ใบเสนอราคาหรือใบแจ้งหนี้มาที่ ar@wongnai.com
เพื่อเปิดใบเสร็จได้อย่างถูกต้อง) เบอร์ติดต่อ 095-906-5551

ภาพที่ 4.1 ตัวอย่างใบแจ้งหนี้ที่ทำโดยระบบ

- การเปลี่ยน algorithm การเลือกภาพปกของร้านอาหารให้เลือกรูปอาหารที่สวยที่สุดของร้านนั้นขึ้นมาแสดง

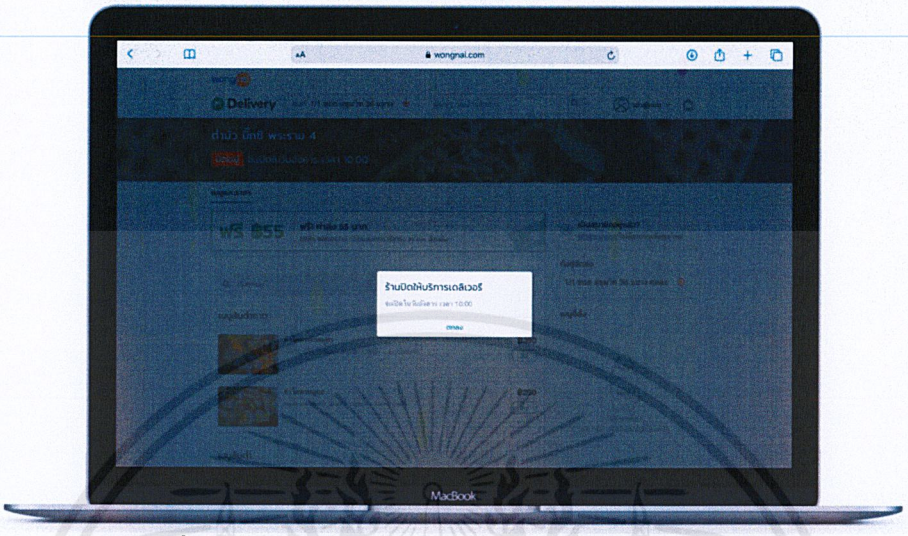
ภาพที่ 4.2 ผลลัพธ์การเปลี่ยน algorithm การเลือกภาพปกของร้านอาหาร

- การเพิ่มโบนัสแต้มแรกเข้าสำหรับลูกค้าใหม่ที่ได้รับคะแนน Reward Card จากร้านอาหารครั้งแรก



ภาพที่ 4.3 ผลลัพธ์การเพิ่มแต้มโบนัสแรกเข้า

- แก้ไขข้อความ error message ที่แจ้งเวลาเปิดร้านอาหารให้แสดงผลได้อย่างถูกต้อง

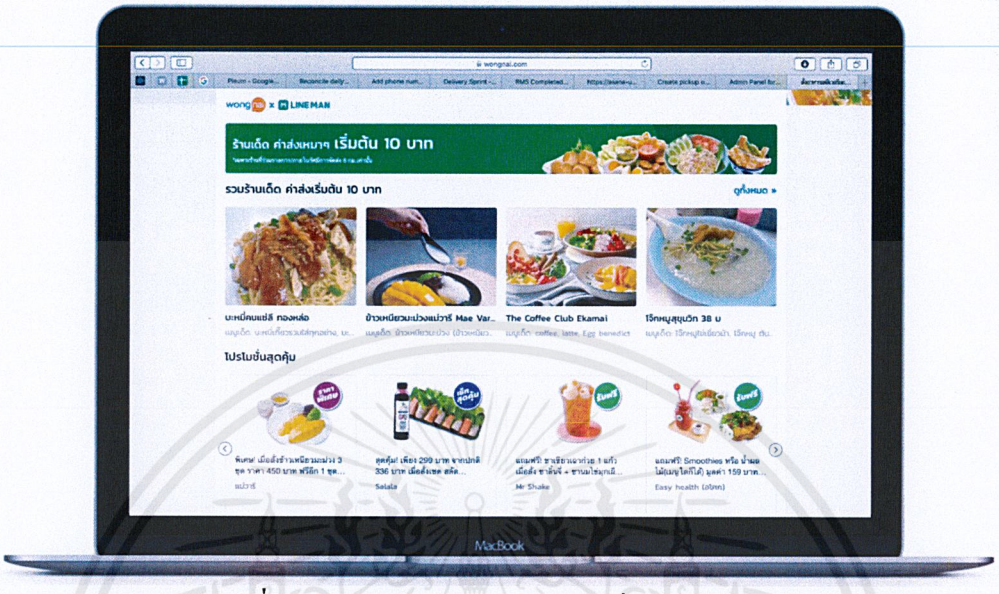


ภาพที่ 4.4 ผลลัพธ์การแก้ไขข้อความแจ้งเวลาเปิดร้านอาหาร

- เพิ่มการตรวจสอบข้อมูลเบอร์โทรศัพท์ของผู้ใช้งานก่อนการบันทึกเข้าระบบเพื่อป้องกันการบันทึกเบอร์ซ้ำลงในระบบ

ภาพที่ 4.5 ผลลัพธ์การตรวจสอบเบอร์โทรศัพท์ก่อนก็บันทึกเข้าระบบ

- แก้การแสดงผลโปรโมชั่นในหน้าเดลิเวอรี่ให้สามารถแสดงผลแบบเรียงลำดับตามระยะทางของร้านอาหารได้



ภาพที่ 4.6 ผลลัพธ์การจัดเรียงโปรโมชั่นตามระยะทาง

- แก้การแสดงผลหรือค้นหาร้านอาหารให้แสดงแค่ร้านอาหารที่อยู่ในพื้นที่บริการเดียวกัน

ภาพที่ 4.7 ผลลัพธ์การแสดงร้านอาหารที่อยู่ในพื้นที่บริการเดียวกันกับผู้ใช้งาน

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

จากการเปลี่ยนแปลงของความต้องการของมนุษย์ ทำให้ต้องมีการพัฒนาระบบ Wongnai Delivery และ RMS เพื่อทำให้ระบบสามารถทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น สามารถตอบสนองต่อความต้องการที่เปลี่ยนไปได้ ซึ่งสิ่งที่พัฒนาไปทั้งหมดนั้นใช้งานได้และถูกนำไปใช้งานจริงในปัจจุบัน

การพัฒนาระบบทำให้สามารถแก้ไขปัญหาการทำงานของระบบที่ผิดพลาดที่ผู้ใช้งานพบเจอและรายงานเข้ามา เพิ่มความสะดวกสบายและลดภาระของพนักงานในการทำงาน ทั้งยังช่วยประหยัดเวลาในการทำงานต่างๆ ให้รวดเร็วยิ่งขึ้น การเพิ่มฟีเจอร์การทำงานหรือโมเดลทางธุรกิจใหม่ๆ ก็สามารถช่วยกระตุ้นยอดขายของร้านอาหารและย้อนกลับมาเพิ่มรายได้ให้กับบริษัทได้เช่นกัน

5.2 ข้อเสนอแนะ

1) การพัฒนาหรือการแก้ปัญหาในบางส่วนอาจยังไม่ใช่วิธีที่มีดีและประสิทธิภาพมากที่สุด เนื่องจากบางครั้งจำเป็นต้องรีบแก้ไขโดยเร่งด่วน จึงต้องเลือกทำในวิธีที่ง่ายและได้ผลดีเพื่อให้สามารถแก้ปัญหาเหล่านั้นให้ได้อย่างเร็วที่สุด

2) การตั้งชื่อตัวแปรหรือข้อมูลต่างๆ ในระบบที่มีขนาดใหญ่ควรคำนึงถึงการใช้งานต่อๆ ไปในอนาคตด้วย เพราะเมื่อระบบใหญ่มากขึ้น ข้อมูลต่างๆ ก็เพิ่มขึ้นเช่นกัน การใช้ชื่อตัวแปรหรือข้อมูลที่ใกล้เคียงกันจะทำให้เกิดการสับสนและเข้าใจผิดได้

เอกสารอ้างอิง

[1] Spring Framework Documentation. Retrieved Oct 23, 2019, from https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html

[2] [How 2 solr search simply] สร้างระบบค้นหาที่ไวกว่า mysql. Retrieved Nov 12, 2019, from https://macxoomscie.blogspot.com/2014/09/how-2-solr-search-simply-mysql.html

[3] Kubernetes Documentation. Retrieved Nov 13, 2019, from https://kubernetes.io/docs/home/

[4] ทำความรู้จัก Docker และการใช้งานบน CentOS 7. Retrieved Nov 13, 2019, from https://www.hostpacific.com/using-docker-on-centos7/

[5] Nuttawat, R. 2017. อธิบาย React Lifecycle แต่ละอันมีหน้าที่อย่างไร. Retrieved Nov 18, 2019, from https://igokuz.com/48e65c922af1

[6] ทำความรู้จักกับ Redux แบบฉบับย่อยแล้วย่อยอีก. Retrieved Nov 18, 2019, from https://microbenz.in.th/b464808aca12

[7] Jedsada, S. 2018. [GitLab] คืออะไร เริ่มใช้งานเบื้องต้น. Retrieved Nov 24, 2019, from https://medium.com/jed-ng/5ccffc4304