การพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล

น.ส.วิภาดา กลึงเทศ

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

A Development of an Automatic Tool for Modeling Relationships of RESTful API Web Service

Miss Wipada Klungtes

An Independent Study Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Software Engineering
Department of Computer Engineering
FACULTY OF ENGINEERING
Chulalongkorn University
Academic Year 2022

หัวข้อสารนิพนธ์	การพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล
โดย	น.ส.วิภาดา กลึงเทศ
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาหลัก	ผู้ช่วยศาสตราจารย์ ดร.เนื่องวงศ์ ทวยเจริญ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับสารนิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

คณะกรรมการสอบสารนิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... อาจารย์ที่ปรึกษาหลัก
(ผู้ช่วยศาสตราจารย์ ดร.เนื่องวงศ์ ทวยเจริญ)

..... กรรมการ
(รองศาสตราจารย์ ดร.ธาราทิพย์ สุวรรณศาสตร์)

วิภาดา กลึงเทศ : การพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล. (A Development of an Automatic Tool for Modeling Relationships of RESTful API Web Service) อ.ที่ปรึกษาหลัก : ผศ. ดร.เนื่องวงศ์ ทวยเจริญ

ในอุตสาหกรรมซอฟต์แวร์ นิยมนำซอฟต์แวร์กลับมาใช้ใหม่เป็นจำนวนมาก เนื่องจากเป็นการลดต้นทุนในการพัฒนาซอฟต์แวร์ เอกสารต่าง ๆ ในการพัฒนาระบบจึงมีความสำคัญในการอ้างอิง งานวิจัยนี้ ให้ความสนใจกับแผนภาพยูเอ็มแอลที่เป็นส่วนหนึ่งของเอกสารส่วนต่อประสานโปรแกรมประยุกต์ จึงได้นำเสนอการออกแบบ และพัฒนาเครื่องมืออัตโนมัติสำหรับการสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลโดยใช้เครื่องมือเสริม PlantUML ที่เป็นเครื่องมือเสริมสำหรับการสร้างแผนภาพ ซึ่งงานวิจัยนี้ได้นำเครื่องมือเสริม PlantUML มาสร้างแผนภาพแบบจำลองความสัมพันธ์ระหว่างคอนโทรลเลอร์ เมธอด และคุณลักษณะภายในของพารามิเตอร์ ที่นำเสนอในรูปแบบของแผนภาพยูเอ็มแอล ดังนั้นเมื่อนำเครื่องมือที่พัฒนาขึ้นไปประยุกต์ใช้ จะช่วยให้ปรับปรุงเอกสารส่วนต่อประสานโปรแกรมประยุกต์ได้ง่ายยิ่งขึ้น และยังช่วยให้เอกสารตรงกับรหัสต้นฉบับ จากการประยุกต์ใช้เครื่องมือกับโครงการ ทำให้ได้ผลลัพธ์ของความถูกต้องเป็น 100%

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อนิสิต .....

ปีการศึกษา 2565 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6470275321 : MAJOR SOFTWARE ENGINEERING

KEYWORD: UML Diagram, RESTful, Class Diagram, Structural and Behavioral, Automatic Tools

Wipada Klungtes : A Development of an Automatic Tool for Modeling Relationships of RESTful API Web Service. Advisor: Asst. Prof. Dr. NUENGWONG TUAYCHAROEN

In the software industry, the software has often been made reusable because it can reduce the cost of the new project. So, documentation in developing software is important for later references. This paper focuses on the UML diagram that is a part of API documentation. This paper proposes to design and develop an automatic tool for creating a diagram representing RESTful API using PlantUML, an open-source tool for creating a diagram. This paper utilizes PlantUML to create a diagram that represents the relationship between controllers, methods, and attributes in the form of a UML diagram. When we apply this developed tool, we can update the API documentation more easily. Therefore, the API documentation will align with the API's source code. After we validate the diagram from our tool, the correctness of the tool is 100%.

Field of Study: Software Engineering Student's Signature .....

Academic Year: 2022 Advisor's Signature .....

กิตติกรรมประกาศ

ขอกราบขอบพระคุณท่าน ผู้ช่วยศาสตราจารย์ ดร. เนื่องวงศ์ ทวยเจริญ อาจารย์ที่ปรึกษาโครงงานมหาบัณฑิตของข้าพเจ้า เป็นอย่างยิ่งที่ท่านได้เสียสละเวลาอันมีค่าให้คำปรึกษาแนะนำทางด้านการศึกษาหาความรู้คุณธรรม จริยธรรมและแนวทางสำหรับการทำโครงงานมหาบัณฑิตนี้ตลอดจนคอยดูแล และประสานงานให้ความช่วยเหลือแก่นิสิตที่ทำโครงงานมหาบัณฑิตทุกคน

ขอขอบคุณ เพื่อน ๆ หลักสูตรวิศวกรรมซอฟต์แวร์สำหรับกำลังใจและคำปรึกษาแนะนำในการจัดทำโครงงานมหาบัณฑิต

สุดท้ายนี้ขอกราบขอบพระคุณ บิดา มารดา และสมาชิกในครอบครัวทุกท่านที่คอยให้กำลังใจและให้การช่วยเหลือสนับสนุนมาโดยตลอด

วิภาดา กลึงเทศ

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบันการนำซอฟต์แวร์กลับมาใช้ใหม่เป็นที่นิยมอย่างแพร่หลาย เนื่องจากการนำซอฟต์แวร์กลับมาใช้ใหม่นั้น ถือเป็นการนำซอฟต์แวร์ที่มีคุณภาพมาพัฒนาให้มีคุณภาพมากขึ้นกว่าเดิม และเป็นการลดต้นทุนในการพัฒนาซอฟต์แวร์ เช่น ลดระยะเวลาในการพัฒนาซอฟต์แวร์เป็นต้น ซึ่งการนำซอฟต์แวร์กลับมาใช้ใหม่เป็นการสร้างหรือปรับปรุงซอฟต์แวร์ที่มีอยู่โดยใช้โครงสร้างของระบบเดิม ดังนั้นเอกสารของระบบเดิมจึงมีความสำคัญ เช่น เอกสารความต้องการของระบบ เอกสารประกอบการออกแบบ เอกสารข้อกำหนดทางเทคนิค ประกอบด้วย เอกสารที่บอกรายละเอียดเกี่ยวกับกระบวนการทำงานในแต่ละฟังก์ชันงาน (Technical Specification) เอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์ (API: Application Program Interface) เอกสารรายละเอียดโครงสร้างของระบบ (Infrastructure Specification) เอกสารรายละเอียดของฐานข้อมูลของระบบ (Data Dictionary)

ผู้วิจัยได้มองเห็นถึงความสำคัญของการมีเอกสารประกอบของระบบ โดยให้ความสำคัญที่เอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์ เป็นเอกสารอธิบายรายละเอียดของส่วนต่อประสานโปรแกรมประยุกต์ที่อยู่ภายในระบบ ที่อธิบายถึงการรับส่งข้อมูลในรูปแบบ เก็ท (Get), โพสต์ (Post), พุต (Put), ดีลีต (Delete) และระบุถึงพารามิเตอร์ที่เกี่ยวข้อง [8][10][11] กับส่วนต่อประสานโปรแกรมประยุกต์ โดยส่วนใหญ่เอกสารนี้ถูกทำขึ้นจากการออกแบบโดยผู้พัฒนาตั้งแต่ขั้นตอนเริ่มต้นและระหว่างการพัฒนาอาจนำเครื่องมือเสริมโอเพ่นเอพีไอ (OpenAPI) เข้ามาช่วยในการสื่อสารกันระหว่างทีมผู้พัฒนา เมื่อระบบอยู่ในขั้นตอนการพัฒนาหรือพัฒนาเสร็จสิ้นแล้ว บางครั้งเอกสารนี้ไม่ได้ถูกกลับไปปรับปรุง หรือการแก้ไขภายหลังการพัฒนาเสร็จสิ้น หากส่วนต่อประสานโปรแกรมประยุกต์มีการแก้ไข ผู้พัฒนาระบบส่วนใหญ่จะละเลยเอกสารนี้ นอกจากนี้ ผู้ที่จำเป็นต้องใช้เอกสารนี้ อาจไม่เพียงแต่ผู้พัฒนาระบบเท่านั้น แต่อาจเป็นผู้วิเคราะห์ระบบ หรือผู้ที่ต้องการทำความเข้าใจในการรับส่งข้อมูลของระบบ หากเอกสารนี้ถูกจัดทำขึ้นในรูปแบบของแผนภาพ จะช่วยให้ผู้อ่านเอกสารเข้าใจได้ง่าย อีกทั้งยังช่วยให้ผู้พัฒนาระบบสามารถเข้าใจระบบได้จากแผนภาพ ซึ่งจะช่วยลดเวลาในการใช้เครื่องมือเสริมเพื่อให้เข้าใจในการรับส่งข้อมูลของทั้งระบบ

จากปัญหาดังกล่าวข้างต้น ทางผู้วิจัยมีแนวคิดในการออกแบบและพัฒนาเครื่องมืออัตโนมัติสำหรับการสร้างและปรับปรุงแผนภาพที่ใช้ในการรับส่งข้อมูล ซึ่งใช้เป็นแผนภาพยูเอ็มแอล (UML:

Unified Modeling Language) [12] โดยประกอบด้วย เส้นทางการรับส่งข้อมูล พารามิเตอร์สำหรับเส้นทางนั้น ๆ เมื่อมีการนำเครื่องมือนี้ รวมเข้ากับโค้ดของระบบ หากมีการรันระบบ แผนภาพนี้จะถูกสร้างขึ้นโดยอัตโนมัติ เพราะฉะนั้น เมื่อมีการปรับปรุงโค้ดที่มีผลกระทบต่อการรับส่งข้อมูล แผนนี้จะถูกปรับปรุงโดยอัตโนมัติ และสามารถนำไปปรับปรุงเอกสารได้ทันที เพื่อป้องกันการละเลยในการปรับปรุงเอกสาร

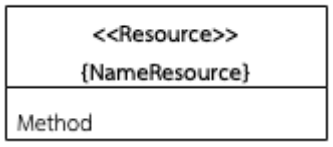
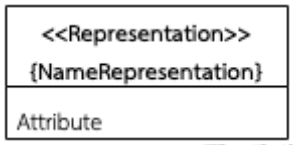
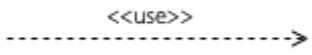
1.2 วัตถุประสงค์ของโครงงาน

เพื่อออกแบบและพัฒนาเครื่องมือที่แปลงส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลในภาษาซีชาร์ปด็อทเน็ตเป็นเอกสารระบบในรูปแบบยูเอ็มแอล

1.3 ขอบเขตของโครงงาน

1. เครื่องมือที่พัฒนาขึ้นเป็นภาษาซีชาร์ปเท่านั้น
2. เครื่องมือที่พัฒนาขึ้นสามารถนำไปใช้ได้กับโครงการที่พัฒนาขึ้นด้วยภาษาซีชาร์ปเท่านั้น
3. เมื่อนำเครื่องมือที่พัฒนาขึ้นนี้ไปใช้ ต้องลงเครื่องมือเสริม PlantUML และสร้างแผนภาพผ่านคอมมานไลน์
4. ในการประเมินและทดสอบเครื่องมือ ทำโดยการนำเครื่องมือไปประยุกต์ใช้กับโครงการทั้งหมดจำนวน 3 โครงการ โดยกำหนดให้แต่ละโครงการมีอย่างน้อย 3 Controllers
5. ในการประเมินและทดสอบเครื่องมือ ต้องทดสอบจากการนำเครื่องมือไปประยุกต์ใช้จริง โดยแผนภาพผลลัพธ์ที่ได้จากเครื่องมือที่พัฒนาขึ้นต้องตรงกับแผนภาพที่ได้จากการสร้างโดยผู้ทดสอบเป็นผู้วาดผ่านเครื่องมือ draw.io โดยกำหนดจำนวนแผนภาพที่ต้องนำมาทดสอบเปรียบเทียบผลลัพธ์โครงการละ 3 Controllers โดยตรวจสอบความถูกต้องของข้อมูลตามที่ได้ระบุไว้ในบทที่ 5 เท่านั้น
6. กำหนดสัญลักษณ์สำหรับการทำแผนภาพ ดังตารางที่ 1.1

ตารางที่ 1.1 สัญลักษณ์สำหรับการทำแผนภาพ

สัญลักษณ์	คำอธิบาย
	ใช้สำหรับชื่อของ Controller ที่กำลังสนใจ
	ใช้สำหรับบ่งบอกถึงเส้นทางการเดินข้อมูล โดยมีการระบุวิธีการดำเนินงานภายใน
	ใช้สำหรับบ่งบอกถึงคุณลักษณะของโมเดลต่าง ๆ
	ใช้สำหรับบ่งบอกถึงความสัมพันธ์ของ Resource กับ Representation
	ใช้สำหรับบ่งบอกถึงเส้นทางการสื่อสารข้อมูลจาก Application ไปยัง Resource

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาหลักการและหาแนวทางในการออกแบบและพัฒนาเครื่องมือ
2. นำแนวทางที่ได้จากงานวิจัยอื่น ๆ มาแก้ไข ปรับปรุงเพิ่มเติม
3. ค้นหาโค้ดตัวอย่างเพื่อนำมาเป็นแนวทางในการพัฒนาเครื่องมือ
4. กำหนดโครงร่างของแผนภาพที่จะพัฒนาขึ้น
5. พัฒนาโค้ดด้วยเครื่องมือเสริม PlantUML ให้ได้ตามโครงร่างแผนภาพที่ได้ร่างไว้
6. พัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล
7. ทดสอบประสิทธิภาพและประเมินผล
8. สรุปผลการทดสอบ
9. จัดทำเอกสารโครงงานมหาบัณฑิต
10. จัดทำบทความทางวิชาการ

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เครื่องมือที่สามารถช่วยปรับปรุงเอกสารส่วนต่อประสานโปรแกรมประยุกต์
2. ช่วยลดระยะเวลาในการปรับปรุงเอกสารส่วนต่อประสานโปรแกรมประยุกต์
3. ช่วยให้ทีมผู้พัฒนาสามารถสื่อสารกันได้สะดวกมากยิ่งขึ้น โดยมีแผนภาพจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลเป็นตัวช่วยในการสื่อสาร

1.6 ผลงานวิจัยที่ได้รับการตีพิมพ์

“การพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟุล” โดย วิภาดา กลึงเทศ และ เนื่องวงศ์ ทวยเจริญ ได้ตีพิมพ์ และนำเสนอในงานประชุมวิชาการ “The 19th National Conference on Computing and Information Technology (NCCIT 2023)” จัดขึ้น ณ ประเทศไทย ระหว่างวันที่ 18 ถึง 19 พฤษภาคม 2566

1.7 โครงสร้างของเนื้อหาโครงงาน

โครงสร้างของเนื้อาหารายงานโครางงานมหาบัณฑิตประกอบด้วยรายละเอียด 6 บท และภาคผนวก ดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน ขั้นตอนและวิธีดำเนินโครงงาน และประโยชน์ที่คาดว่าจะได้รับจากโครงงานมหาบัณฑิต

บทที่ 2 กล่าวถึงทฤษฎี งานวิจัย และเครื่องมือที่เกี่ยวข้อง

บทที่ 3 กล่าวถึงกระบวนการทำงานของเครื่องมืออัตโนมัติสำหรับการสร้างแบบจำลองความสัมพันธ์

บทที่ 4 กล่าวถึงการพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลอง

บทที่ 5 กล่าวถึงการทดสอบและประเมินผลความสามารถของเครื่องมือ

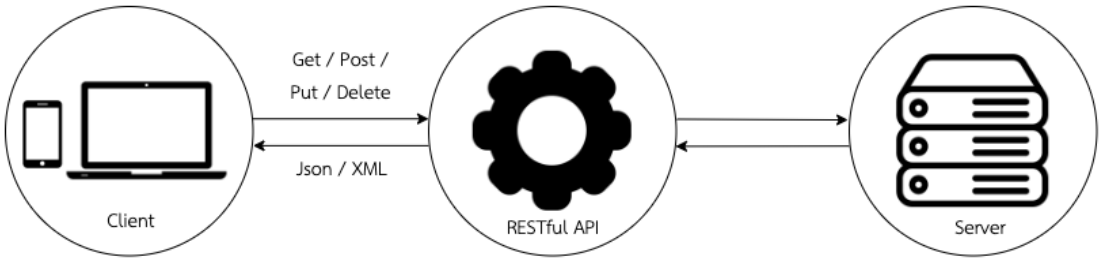
บทที่ 6 กล่าวถึงบทสรุปของโครงงานมหาบัณฑิต และข้อเสนอแนะ

บทที่ 2
เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เรสต์ฟูลเว็บเซอร์วิส

เรสต์ฟูลเว็บเซอร์วิส หรือ RESTful Web Services (RWS) เป็นเว็บเซอร์วิชชนิดหนึ่ง ที่ใช้สถาปัตยกรรมแบบเรสต์ โดยอนุญาตให้มีผู้ร้องขอการเข้าถึงทรัพยากร เรียกว่า เครื่องลูกข่าย และมีผู้ให้บริการทรัพยากร เรียกว่า เครื่องแม่ข่าย



ภาพที่ 2.1 กระบวนการทำงานของเรสต์ฟูลเว็บเซอร์วิส

จากภาพที่ 2.1 การรับส่งข้อมูลผ่านส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล โดยมีกระบวนการทำงาน ดังนี้

1. เครื่องลูกข่ายเป็นผู้ร้องขอข้อมูลจากเครื่องแม่ข่ายผ่าน Http หรือ Https ในรูปแบบของเมธอด Get, Post, Put, Delete ผ่านเรสต์ฟูลเว็บเซอร์วิส
2. เรสต์ฟูลเว็บเซอร์วิส ทำหน้าที่เป็นตัวกลางระหว่างเครื่องลูกข่ายและเครื่องแม่ข่าย ซึ่งจะรับคำร้องขอจากเครื่องลูกข่าย และส่งคำร้องขอนั้นไปที่เครื่องแม่ข่าย
3. เครื่องแม่ข่ายเป็นผู้ให้บริการข้อมูล เมื่อรับคำร้องขอจากเครื่องลูกข่ายผ่านเรสต์ฟูลเว็บเซอร์วิช จะนำคำร้องขอนั้น ติดต่อสื่อสารกับฐานข้อมูล และส่งผลการตอบสนองกลับผ่านเรสต์ฟูลเว็บเซอร์วิส
4. เรสต์ฟูลเว็บเซอร์วิส รับผลการตอบสนองจากเครื่องแม่ข่าย และส่งกลับให้เครื่องลูกข่าย ซึ่งผลการตอบสนองนั้น อาจอยู่ในรูปแบบของ JSON หรือ XML

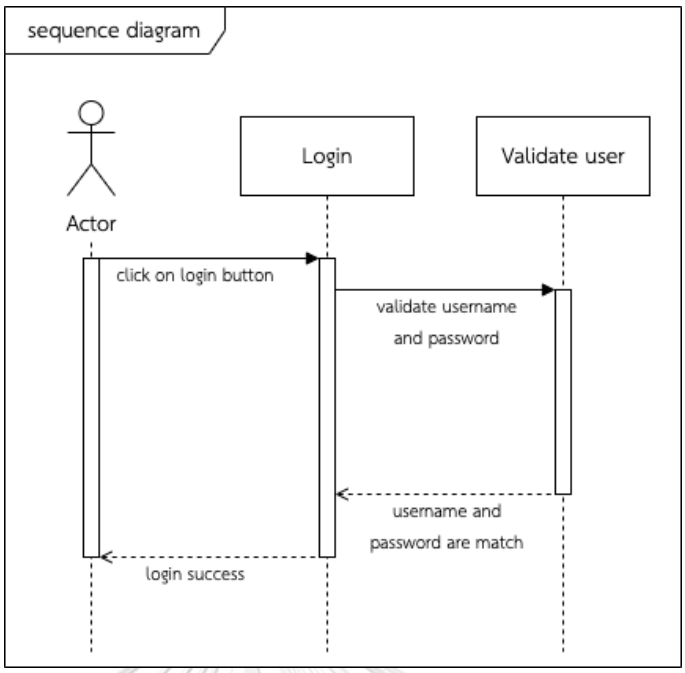
2.1.2 แผนภาพยูเอ็มแอล

แผนภาพยูเอ็มแอล (Unified Modeling Language: UML) เป็นแผนภาพที่แสดงพฤติกรรมและโครงสร้างของระบบ ที่เป็นสัญลักษณ์มาตรฐานจึงทำให้ทุกคนสามารถอ่านและเข้าใจได้ หากนำมาใช้ก่อนเริ่มพัฒนาระบบจะทำให้เห็นภาพรวมของระบบ และหาจุดบกพร่องเพื่อทำการแก้ไขได้เร็วซึ่งสามารถช่วยลดต้นทุนและระยะเวลาในการพัฒนาระบบ ประเภทของแผนภาพยูเอ็มแอล มี 2 ประเภท ดังนี้

1. แผนภาพโครงสร้าง เป็นแผนภาพที่อธิบายการออกแบบเชิงตรรกะและทางกายภาพของระบบ ซึ่งมีโครงสร้างที่ชัดเจน โดยมีส่วนประกอบ 3 ส่วน คือ ส่วนหัวเป็นส่วนที่ระบุชื่อคลาส ส่วนถัดไปเป็นส่วนที่ที่ระบุคุณสมบัติของคลาส และส่วนสุดท้ายเป็นวิธีดำเนินงานของคลาสนั้น ๆ ซึ่งแต่ละคลาสที่สัมพันธ์กันจะมีเส้นเชื่อมระหว่างกัน ตัวอย่างของแผนภาพโครงสร้างยูเอ็มแอล เช่น แผนภาพคลาส ดังภาพที่ 2.2 เป็นต้น

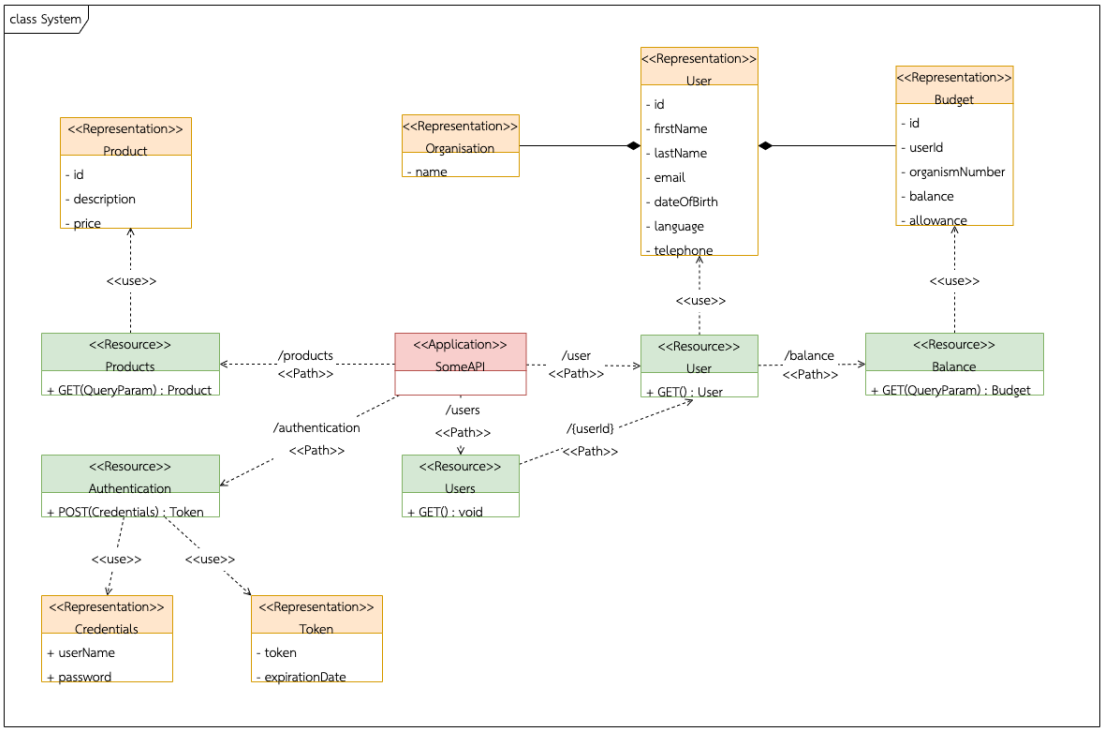
ภาพที่ 2.2 ตัวอย่างแผนภาพคลาส

2. แผนภาพพฤติกรรมหรือแผนภาพกิจกรรม เป็นแผนภาพที่แสดงถึงกระบวนการทำงานของระบบ หรือส่วนหนึ่งของระบบ เป็นชุดของกิจกรรมที่เกิดขึ้นโดยมีจุดเริ่มต้นและสิ้นสุดอย่างชัดเจนซึ่งแสดงถึงการทำงานเป็นขั้นตอนเพื่อให้บรรลุเป้าหมายของผู้ใช้งานระบบ เช่น State Machine Diagram, Sequence Diagram ดังภาพที่ 2.3 เป็นต้น



ภาพที่ 2.3 ตัวอย่างแผนภาพพฤติกรรมแบบ Sequence Diagram

2.1.3 แผนภาพส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล



ภาพที่ 2.4 ตัวอย่างแผนภาพส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล

จากภาพที่ 2.4 แผนภาพส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล [9][13] แสดงความสัมพันธ์ และพฤติกรรมที่เป็นส่วนหนึ่งของระบบ โดยมีส่วนประกอบ 3 ส่วน ดังนี้

- <<Application>> เป็นคลาสของตัวกลางในการรับส่งข้อมูล
- <<Resource>> เป็นคลาสของแอ็กชันต่าง ๆ โดยมีการดำเนินงานอยู่ภายใน ซึ่งจะมีเส้นทางการขอข้อมูลจาก <<Application>> ไปยัง <<Resource>> ซึ่งที่เส้นเชื่อมระหว่างคลาสจะมี <<Path>> กำกับที่ระบุชื่อของเส้นทางการขอข้อมูล
- <<Representation>> เป็นคลาสของทรัพยากรของข้อมูลโดยมีคุณลักษณะของคลาสอยู่ภายใน ซึ่งที่เส้นเชื่อมระหว่างคลาสจะมี <<use>> กำกับเพื่อบอกถึงการเรียกใช้จาก <<Resource>>

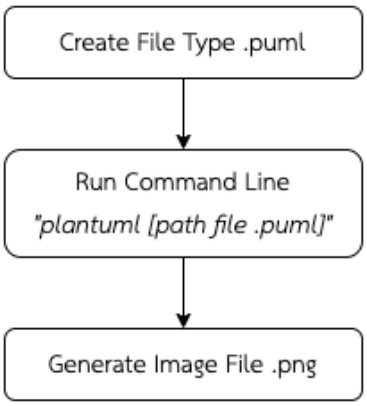
ยกตัวอย่างเช่น <<Application>> SomeAPI มีการร้องขอข้อมูลไปยัง <<Resource>> Products โดยผ่านเส้นทาง /products ซึ่งเป็นเมธอด Get ที่รับพารามิเตอร์ 1 ตัว คือ QueryParam และมีค่าในการส่งกลับ คือ Product ที่เกิดเป็นความสัมพันธ์กับ <<Representation>> Product ที่มีคุณลักษณะ ที่ประกอบด้วย Id, Description, Price เป็นต้น

หากเส้นทางในการขอข้อมูลที่มีการขอข้อมูลในรูปแบบของ <<Representation>> และมีการส่งกลับในรูปแบบของ <<Representation>> เช่นกัน โดยอยู่ภายใต้เงื่อนไข คือ ทั้งการขอข้อมูลและการส่งกลับข้อมูลมี <<Representation>> ต่างกัน จะทำให้เกิดความสัมพันธ์สำหรับ <<Representation>> ของ 2 <<Representation>> ที่ต่างกัน

ยกตัวอย่างเช่น <<Application>> SomeAPI มีการร้องขอข้อมูลไปยัง <<Resource>> Authentication โดยผ่านเส้นทาง /authentication ซึ่งเป็นเมธอด Post ที่รับพารามิเตอร์ 1 ตัว คือ Credential เป็น <<Representation>> ที่ส่งข้อมูลกลับ คือ Token เป็น <<Representation>> เช่นกัน จึงเกิดเป็นความสัมพันธ์ระหว่าง <<Resource>> และ <<Representation>> 2 ตัว เป็นต้น

2.1.4 PlantUML

PlantUML [6] เป็นเครื่องมือเสริม ที่ช่วยในการสร้างแผนภาพต่าง ๆ เช่น แผนภาพยูสเคส แผนภาพคลาส แผนภาพกิจกรรม เป็นต้น โดยนำมาพัฒนาร่วมกันกับภาษาซีชาร์ปดอทเน็ตเวอร์ชัน 3.1 [7] มีกระบวนการทำงานดังภาพที่ 2.5



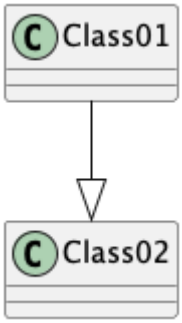
ภาพที่ 2.5 กระบวนการทำงานของเครื่องมือ PlantUML

จากภาพที่ 2.5 กระบวนการทำงานของเครื่องมือ PlantUML เริ่มจากการสร้างไฟล์ที่มีนามสกุลของไฟล์เป็น .puml ซึ่งรูปแบบของไฟล์ จะอยู่ในลักษณะของข้อความโดยมี Syntax เฉพาะของเครื่องมือ PlantUML หลังจากนั้นให้ทำการรันคำสั่งผ่านคอมมานไลน์ด้วยคำสั่ง “plantuml ตามด้วยพาทของไฟล์ .puml ที่ได้สร้างขึ้น” เครื่องมือ PlantUML จะทำการสร้างแผนภาพขึ้นมาให้โดยอัตโนมัติ ซึ่งรูปแบบของแผนภาพที่ได้ ขึ้นอยู่กับการเขียนข้อความในไฟล์ .puml ที่สร้างขึ้น

การสร้างไฟล์ .puml จำเป็นต้องใช้ Syntax ตามที่ผู้พัฒนาเครื่องมือ PlantUML ได้กำหนดไว้ โดยมีส่วนประกอบหลัก 3 ส่วน คือ เมื่อเริ่มไฟล์ต้องกำหนด “@startuml” ซึ่งเป็นส่วนเปิดข้อความ ส่วนถัดมาเป็นข้อความของแผนภาพที่ต้องการให้เครื่องมือ PlantUML สร้างให้อัตโนมัติ เช่น “Class01 --|> Class02” เป็นต้น และส่วนสุดท้ายต้องมีการปิดข้อความด้วย “@enduml” จะได้ข้อความดังภาพที่ 2.6 ซึ่งเมื่อใช้คำสั่ง “plantuml ตามด้วยพาทของไฟล์ .puml ที่ได้สร้างขึ้น” จะทำให้เกิดเป็นแผนภาพดังภาพที่ 2.7


```
@startuml
Class01 --|> Class02
@enduml
```

ภาพที่ 2.6 ตัวอย่างข้อความสำหรับสร้างแผนภาพ

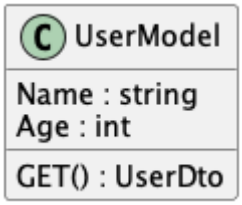


ภาพที่ 2.7 ตัวอย่างแผนภาพที่ถูกสร้างขึ้นด้วยเครื่องมือ PlantUML

การกำหนดคุณลักษณะของคลาสสามารถทำได้โดยกำหนดชื่อของตัวแปร และชนิดของตัวแปร หากต้องการกำหนดวิธีดำเนินการสามารถทำได้โดยใช้ข้อความ “void method()”, “GET() : User” เป็นต้น เครื่องมือ PlantUML จะสามารถทราบได้ถึงหน้าที่ในข้อความแต่ละบรรทัด ดังภาพที่ 2.8 มีการกำหนดคุณลักษณะและวิธีดำเนินการ เครื่องมือ PlantUML จะสามารถแยกคุณลักษณะและวิธีดำเนินการให้ได้ดังภาพที่ 2.9

```
@startuml
class UserModel {
    Name : string
    Age : int
    GET() : UserDto
}
@enduml
```


ภาพที่ 2.8 ตัวอย่างข้อความกำหนดคุณลักษณะและวิธีดำเนินงาน



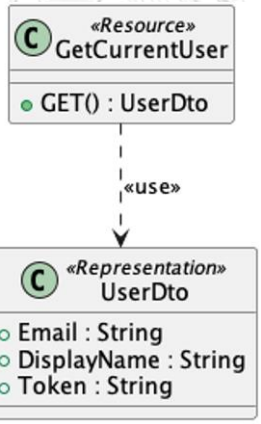
ภาพที่ 2.9 ตัวอย่างแผนภาพจากการกำหนดคุณลักษณะและวิธีดำเนินงาน

การกำหนดข้อความกำกับบนเส้น สามารถกำหนดได้ด้วยการกำหนดข้อความโดยมี Syntax คือ “Class1 --|> Class2 : ข้อความที่ต้องการกำกับ” ดังภาพที่ 2.10 รหัสต้นฉบับตัวอย่างของการเขียน

กำกับบนเส้น คือ GetCurrentUser ..> UserDto : <<use>> เกิดเป็นการกำกับบนเส้นระหว่าง 2 คลาส คือ คลาส GetCurrentUser กับคลาส UserDto ว่า <<user>> ดังภาพที่ 2.11

```
@startuml
class GetCurrentUser <<Resource>>
{
    + GET() : UserDto
}
class UserDto <<Representation>>
{
    + Email : String
    + DisplayName : String
    + Token : String
}
GetCurrentUser ..> UserDto : <<use>>
@enduml
```

ภาพที่ 2.10 ตัวอย่างข้อความกำกับบนเส้นเชื่อม



ภาพที่ 2.11 ตัวอย่างแผนภาพจากการกำกับข้อความบนเส้นเชื่อม

ในการนำมาเครื่องมือ PlantUML มาใช้ จำเป็นต้องคำนึงถึง Syntax ซึ่งในหัวข้อนี้ นำตัวอย่าง Syntax มาเพียงส่วนหนึ่งที่จำเป็นต่องานวิจัยนี้ นอกจากเครื่องมือ PlantUML จะสามารถสร้างแผนภาพได้ ในการสร้างแผนภาพนั้นยังสามารถปรับแต่ง แก้ไขความสวยงามเพิ่มเติมได้อีกมาก ตามเอกสารแนวทางของเครื่องมือ PlantUML [6]

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

งานวิจัยนี้ได้ทำการศึกษาเกี่ยวกับการสร้างและปรับปรุงเอกสารที่มีความสำคัญต่อการพัฒนาระบบ ทำให้พบช่องว่างต่าง ๆ จนเกิดเป็นแนวคิดในการออกแบบและพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล โดยมีงานวิจัยที่เกี่ยวข้องดังนี้

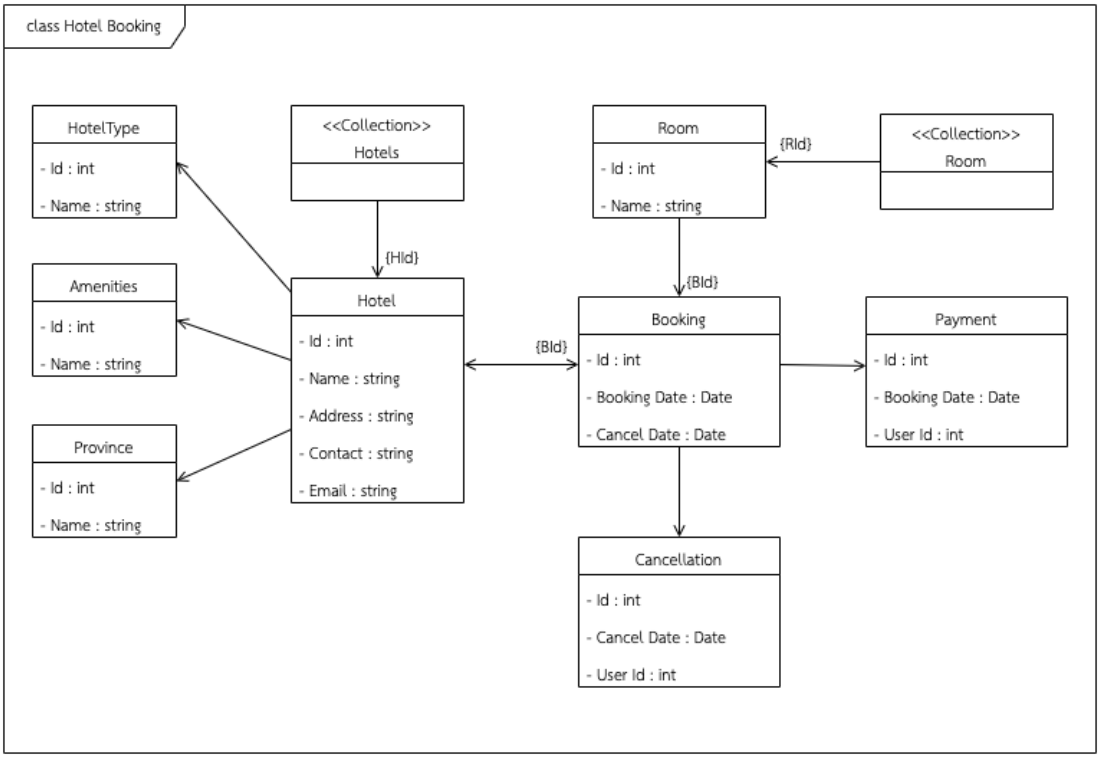
2.2.1 A Systematic Mapping Study on API Documentation Generation Approaches

งานวิจัยของ Kristian Nybom และคณะ [1] ได้ทำการวิเคราะห์ถึงความสำคัญของเอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์ เนื่องจากในปัจจุบันมีการนำซอฟต์แวร์กลับมาใช้ใหม่เป็นจำนวนมาก ทำให้การมีเอกสารแสดงส่วนต่อประสานโปรแกรมที่ดี ช่วยในการทำงานของนักพัฒนาโปรแกรมส่วนด้านหน้าและนักพัฒนาโปรแกรมส่วนเบื้องหลังเป็นไปได้อย่างราบรื่น เพราะเอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์ เป็นเอกสารสำหรับการนำเสนอภาพรวมของระบบ โดยอธิบายความซับซ้อนและโครงสร้างของระบบในรูปแบบภาษาที่เข้าใจง่าย รวมถึงการวิเคราะห์เกี่ยวกับการทำนุบำรุงรักษาเอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์ด้วย

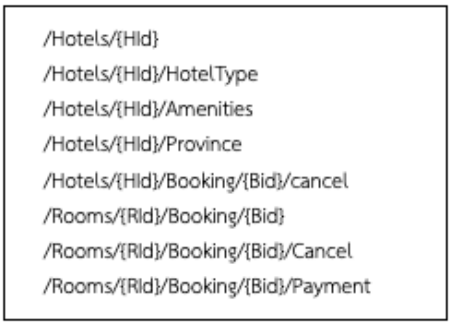
ผู้วิจัยจึงได้เห็นถึงความสำคัญของการมีเอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์ เนื่องจากงานวิจัยนี้ เป็นเพียงการวิเคราะห์และเสนอแนวทางในการสร้างหรือปรับปรุงเอกสารเท่านั้น ซึ่งการปรับปรุงเอกสารมีขั้นตอนที่ซับซ้อนและใช้เวลามาก ทำให้เอกสารส่วนมากไม่ได้รับการปรับปรุงอย่างสม่ำเสมอ ผู้วิจัยจึงมีแนวคิดในการทำเครื่องมือสำหรับการปรับปรุงเอกสารเอกสารแสดงส่วนต่อประสานโปรแกรมประยุกต์แบบอัตโนมัติ เพื่ออำนวยความสะดวกต่อผู้พัฒนาระบบในการนำระบบกลับมาใช้ใหม่อีกครั้ง

2.2.2 Structural and Behavioral modeling of RESTful web service interface using UML

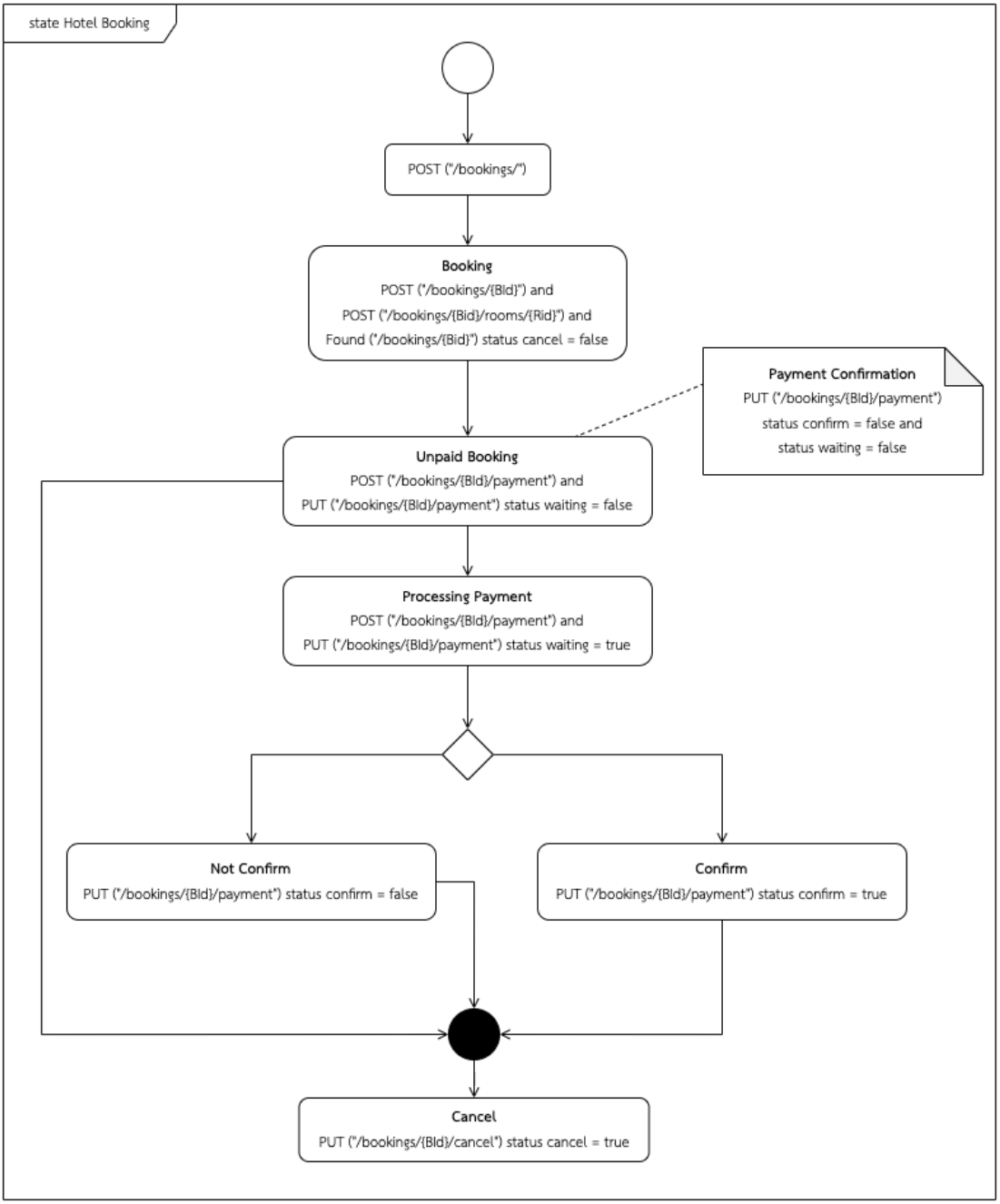
งานวิจัยของ Digvijaysinh M.Rathod และคณะ [2] ได้ทำการวิเคราะห์โครงสร้างและพฤติกรรมของแหล่งข้อมูลและความสัมพันธ์ของทรัพยากร รวมถึงคุณลักษณะต่าง ๆ ของทรัพยากร โดยใช้แบบจำลองแผนภาพยูเอ็มแอล และแผนภาพแสดงสถานะของวัตถุ (State Machine Diagram) เพื่อออกแบบแผนภาพจำลองที่แสดงการสื่อสารแลกเปลี่ยนข้อมูลเว็บเซอร์วิสของแอปพลิเคชันการจองห้องในโรงแรม ซึ่งในแบบจำลอง สนใจที่การรับส่งข้อมูลแบบเก็ท โพสต์ พุต ดีลีต



ภาพที่ 2.12 ภาพความสัมพันธ์ของทรัพยากรในระบบ



ภาพที่ 2.13 ภาพตัวอย่างเส้นทางการรับส่งข้อมูลในระบบ



ภาพที่ 2.14 ภาพสถานะการทำงานในการจองห้องพักของระบบ

จากภาพที่ 2.12 อธิบายความสัมพันธ์ของทรัพยากรในระบบโดยใช้การจำลองแผนภาพยูเอ็มแอล ซึ่งมีการเปลี่ยนสถานะการจองห้องพักเป็นไปดังภาพที่ 2.14 เมื่อผู้ใช้งานระบบต้องการจองห้องพัก ระบบส่งข้อมูลผ่านเมธอด POST: http://example/Hotels/{Hid}/Bookings/{Bid} ซึ่งมีการระบุเส้นทางการเดินข้อมูลดังภาพที่ 2.13 โดยมีความสัมพันธ์ระหว่างทรัพยากร Hotels กับ Bookings ทำให้เกิดการสร้างข้อมูลของการจองห้องพักที่ยังไม่ถูกชำระเงิน ซึ่งผู้ใช้งานระบบสามารถ

ยกเลิกการจองห้องได้ในสถานะนี้ เมื่อผู้ใช้งานระบบต้องการชำระเงิน ระบบจะมีการสื่อสารโดยการส่งข้อมูลผ่านเมธอด PUT : http://example/Rooms/{RId}/Bookings/{BId}/Payment/ และเข้าสู่สถานะกำลังชำระเงิน (Processing Payment) เพื่อรอผู้ใช้งานระบบยืนยันอีกครั้ง โดยมีเงื่อนไขจากพฤติกรรมของระบบ ดังนี้

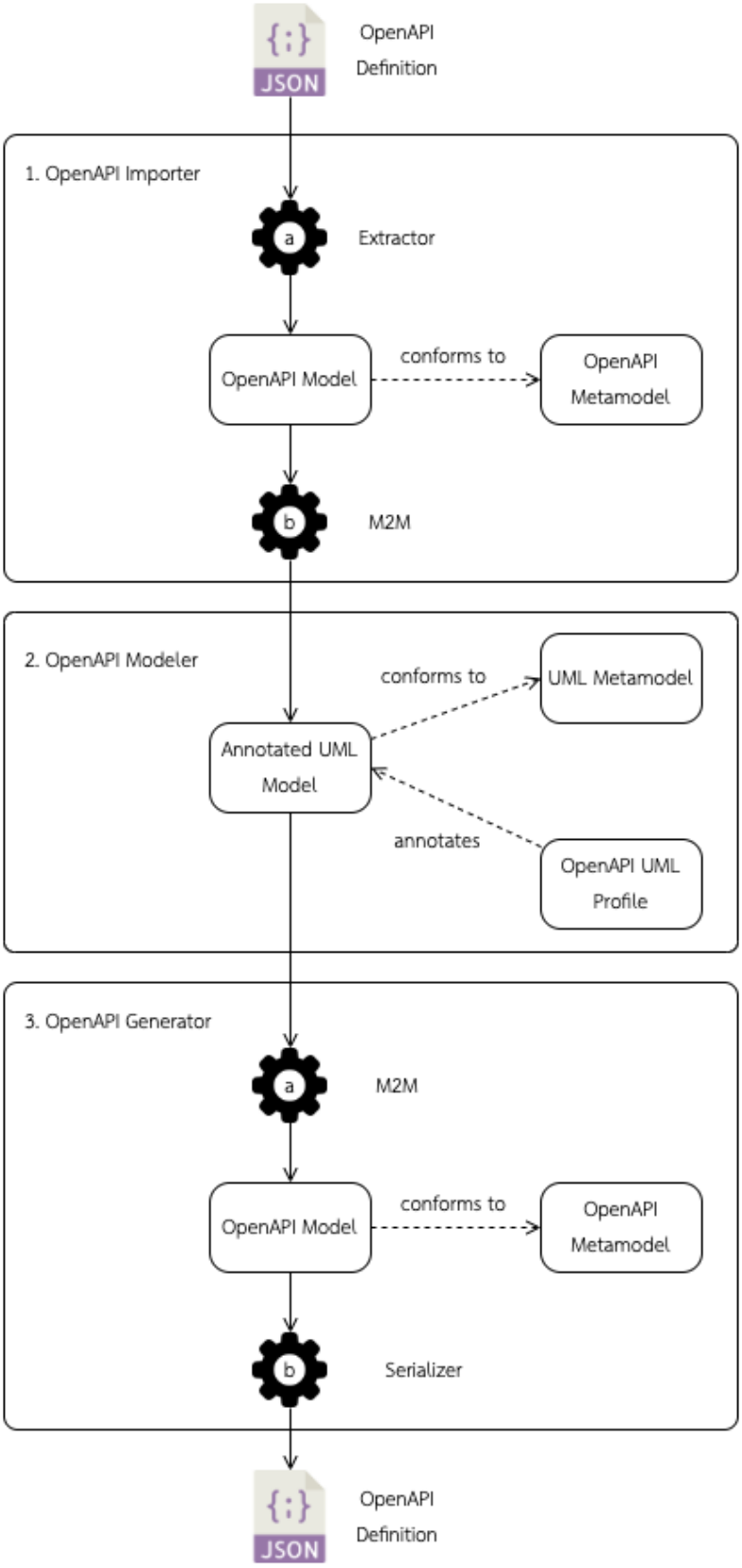
1. การชำระเงินสามารถทำได้ ก็ต่อเมื่อมีการจองห้องพักเกิดขึ้นแล้ว
2. เมื่ออยู่ในสถานะรอการชำระเงิน จะทำให้ไม่สามารถยกเลิกการจองพักได้
3. การจองห้องพักสามารถยกเลิกได้ ก็ต่อเมื่อผ่านการยกเลิกการจองห้องพักแล้ว โดยผ่านการส่งข้อมูลที่ DELETE: http://example/Rooms/{RId}/Booking/{Bid}/cancel

ผู้วิจัยได้แนวคิดในการทำแบบจำลองแผนภาพยูเอ็มแอล จากการแสดงแผนภาพความสัมพันธ์ของทรัพยากรของระบบดังภาพที่ 2.12 ซึ่งผู้วิจัยได้มองเห็นถึงรายละเอียดของแผนภาพที่ยังขาดในส่วนของการดำเนินการ เมื่ออยู่ในเอกสารแสดงส่วนต่อประสานประยุกต์อาจไม่สมบูรณ์และครบถ้วน จึงเกิดแนวคิดในการนำทฤษฎีในข้อ 2.3 มาประยุกต์ใช้

2.2.3 WAPIml: Towards a Modeling Infrastructure for Web APIs

งานวิจัยของ Hamza Ed-douibi และคณะ [3] ได้นำเสนอเครื่องมือสำหรับการทำ OpenAPI โดยใช้เครื่องมือชื่อ WAPIml เป็นเครื่องมือที่ นำเครื่องมือเสริมแบบโอเพ่นซอร์สที่ช่วยในการเขียนโปรแกรมมาแก้ไขไฟล์ JSON เมื่อมีการแก้ไขสคีมา เครื่องมือ WAPIml จะทำการแก้ไข ที่ประกอบด้วย 2 องค์ประกอบหลัก ดังนี้

1. OpenAPI Metamodel ที่มีการอ้างอิงถึงสคีมาของทรัพยากรอยู่ภายใน เส้นทางการสื่อสารของข้อมูล และพารามิเตอร์สำหรับเส้นทางการสื่อสารของข้อมูล
2. OpenAPI UML profile สำหรับการขยายแผนภาพยูเอ็มแอล แบบพื้นฐานเพื่อรองรับ OpenAPI definitions

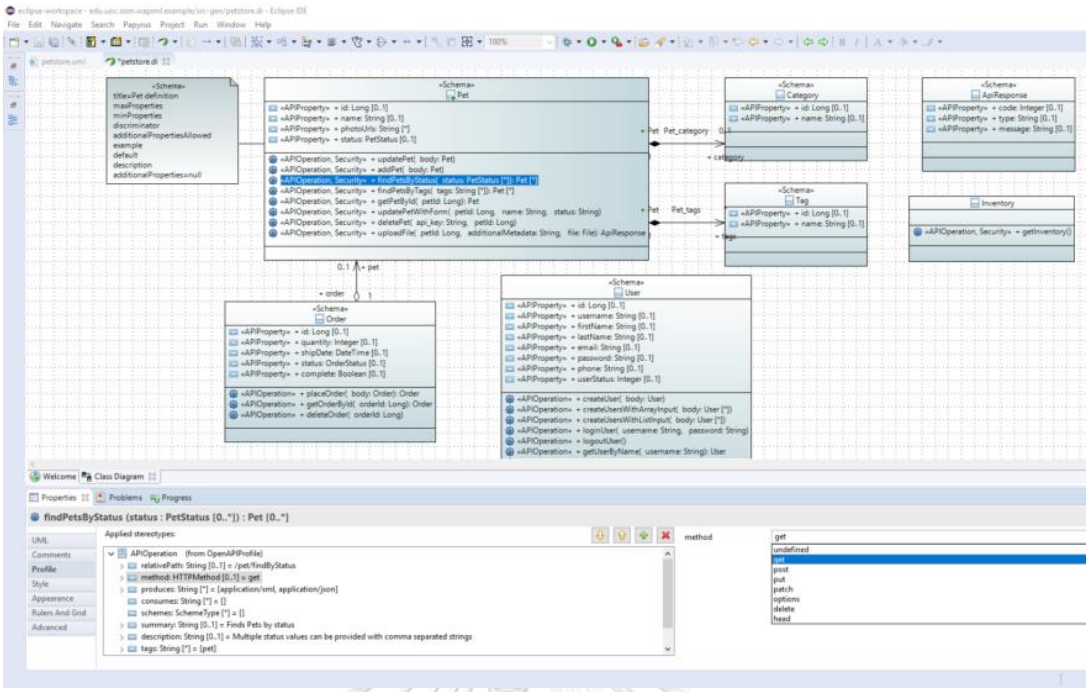


ภาพที่ 2.15 กระบวนการทำงานของเครื่องมือ WAPIml

จากภาพที่ 2.15 กระบวนการทำงานของเครื่องมือ WAPIml ซึ่งมีกระบวนการทำงาน ดังนี้

1. ผู้ใช้งานเครื่องมือ WAPIml ส่งอินพุตเป็นไฟล์ JSON
2. ไฟล์ JSON ถูกนำส่งเข้า OpenAPI Importer ที่มีหน้าที่ในการสร้างแผนภาพยูเอ็มแอล โดยมีคำอธิบายของ OpenAPI Stereotype ที่ประกอบด้วย Classes, Properties, Operations, Data Types, Enumeration, Parameters, Accordingly และแปลง OpenAPI Model ที่เป็นไปตาม OpenAPI Metamodel
3. OpenAPI Model ที่ได้จาก OpenAPI Importer ถูกนำส่งเข้า OpenAPI Modeler โดยนำ OpenAPI Model เดิม และ OpenAPI Model ที่มีการแก้ไข มาสร้างเป็น OpenAPI Model ใหม่
4. OpenAPI Model ที่ถูกแก้ไขแล้ว ถูกนำส่งเข้า OpenAPI Generator เพื่อทำการสร้างไฟล์ JSON OpenAPI Definition ใหม่

จากกระบวนการดังกล่าว เครื่องมือ WAPIml ถือเป็นเครื่องมือที่ช่วยสร้างความสัมพันธ์ของทรัพยากรที่ต้องการออกแบบก่อนนำไปใช้ ให้อยู่ในรูปแบบของแผนภาพยูเอ็ม เพื่อให้เห็นภาพรวมของระบบก่อนเริ่มพัฒนาระบบ จึงช่วยให้ทราบถึงปัญหาที่จะเกิดได้เร็ว และแก้ไขได้เร็วกว่าการลงมือพัฒนาแล้วเกิดปัญหาทำให้ต้องแก้ไขระบบภายหลัง



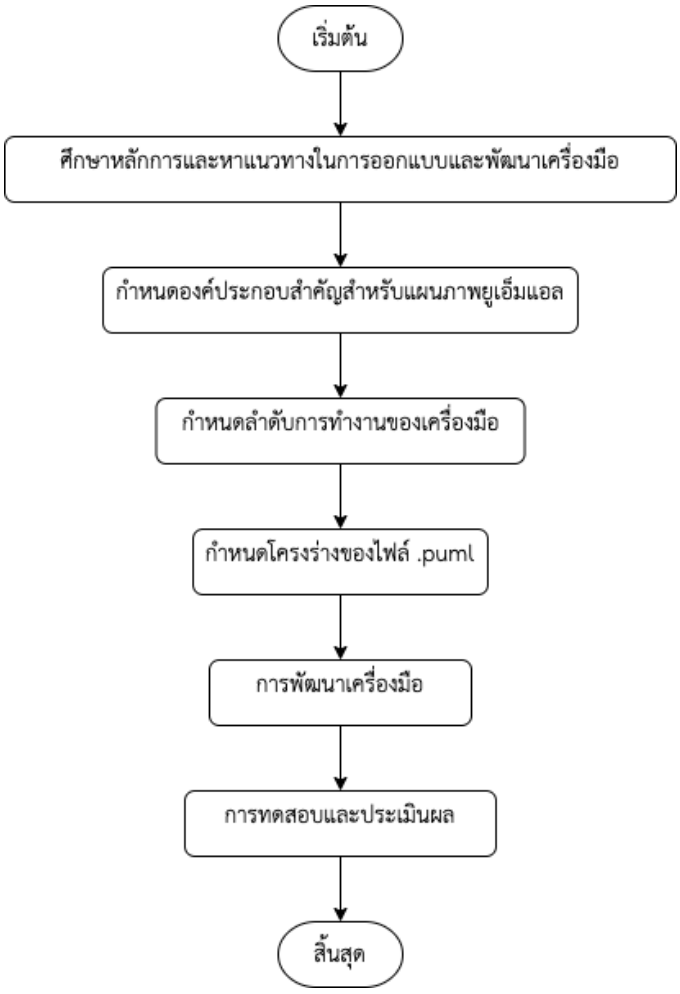
ภาพที่ 2.16 ภาพตัวอย่างเครื่องมือเสริม

จากภาพที่ 2.16 เป็นภาพตัวอย่างเครื่องมือเสริมจากงานวิจัยนี้ ที่เมื่อมีการแก้ไขโมเดลผ่านเครื่องมือเสริม WAPIml จะอ่านข้อมูลทั้งหมดใหม่อีกครั้ง เพื่อแก้ไข OpenAPI Definition

ผู้วิจัยได้แนวคิดในการพัฒนาเครื่องมือที่ช่วยสร้างแผนภาพ ที่มีการช่วยวางแผนก่อนพัฒนาระบบโดยอธิบายถึงความสัมพันธ์ของทรัพยากรของระบบ และการสื่อสารแลกเปลี่ยนข้อมูลเว็บเซอร์วิส หากมีการพัฒนาแผนภาพที่แสดงถึงการสื่อสารแลกเปลี่ยนข้อมูลเว็บเซอร์วิส เมื่อระบบถูกนำกลับมาใช้ใหม่ แผนภาพจะช่วยให้ผู้พัฒนาระบบมองเห็นถึงภาพรวมของระบบก่อนนำโค้ดไปพัฒนาต่อ และได้นำแนวคิดของกระบวนการทำงานของเครื่องมือ WAPIml เป็นแนวทางในการพัฒนาเครื่องมือสำหรับการสร้างแผนภาพที่แสดงถึงการแลกเปลี่ยนข้อมูลเว็บเซอร์วิสแบบอัตโนมัติ ซึ่งผู้วิจัยได้ปรับเปลี่ยนอินพุตจาก OpenAPI Definition ที่เป็นไฟล์ JSON มาเป็นโค้ดที่เขียนด้วยภาษาซีชาร์ป และทำการสร้างไฟล์แผนภาพด้วยเครื่อง PlantUML

บทที่ 3
กระบวนการทำงานของเครื่องมืออัตโนมัติสำหรับการสร้างแบบจำลองความสัมพันธ์

โครงงานมหาบัณฑิตนี้ได้ออกแบบและพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล



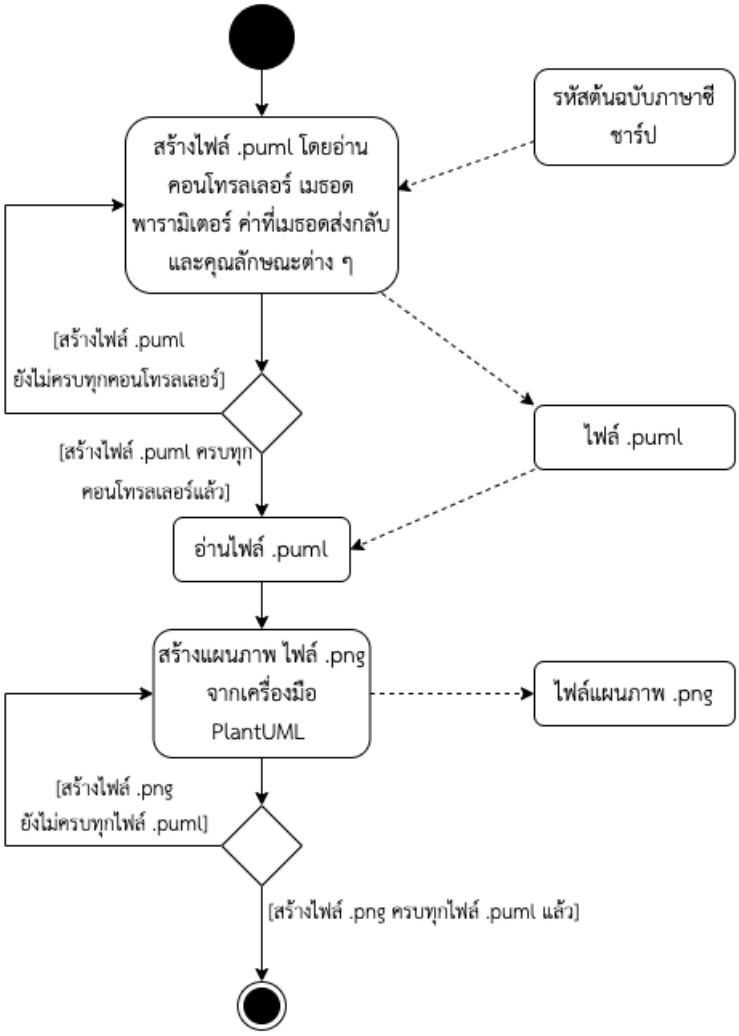
ภาพที่ 3.1 ภาพรวมแนวทางดำเนินงานทั้งหมด

โดยมีแนวทางการดำเนินงานทั้งหมด 6 ขั้นตอน ดังภาพที่ 3.1 ซึ่งในแต่ละขั้นตอนมีรายละเอียดดังนี้

3.1 ศึกษาหลักการและหาแนวทางในการออกแบบและพัฒนาเครื่องมือ

ผู้วิจัยได้ศึกษางานวิจัย [2] ซึ่งได้วิเคราะห์เกี่ยวกับพฤติกรรมของแหล่งข้อมูลและโครงสร้างของระบบ ที่ระบุถึงการเดินทางของข้อมูลโดยแสดงอยู่ในรูปแบบของแผนภาพ และได้นำกระบวนการ

ทำงานของเครื่องมือที่งานวิจัย [3] ใช้ มาปรับปรุงเพิ่มเติม ซึ่งมีการพัฒนาเครื่องมือโดยใช้ภาษาซีชาร์ป และเครื่องมือเสริม PlantUML ในการสร้างแผนภาพ



ภาพที่ 3.2 กระบวนการทำงานของเครื่องมือที่ได้ออกแบบไว้

จากภาพที่ 3.2 จากการออกแบบกระบวนการทำงานของเครื่องมือ มีกระบวนการดังนี้

3.1.1 นำเครื่องมือที่ได้พัฒนาขึ้นไปประยุกต์ใช้กับโครงการที่ต้องการสร้างแผนภาพ และเริ่มการทำงานของโครงการนั้น

3.1.2 เครื่องมือที่ได้พัฒนาขึ้น จะทำการสร้างไฟล์ .puml โดยการอ่านคอนโทรลเลอร์ ที่ภายในประกอบด้วยเมธอดต่าง ๆ ซึ่งมีพารามิเตอร์และค่าที่ส่งกลับ และอ่านคุณลักษณะต่าง ๆ ภายในนั้น ซึ่งเครื่องมือจะทำการสร้างไฟล์ .puml จนครบทุกคอนโทรลเลอร์ในโครงการ

3.1.3 เครื่องมือที่ได้พัฒนาขึ้น จะทำการสร้างไฟล์ .png โดยรันคำสั่ง “plantuml {path file}” [14] เพื่อสร้างไฟล์แผนภาพตามไฟล์ .puml จากข้อ 4.1.2 ให้อัตโนมัติ โดยใช้เครื่องมือเสริม PlantUML

3.2 กำหนดองค์ประกอบสำคัญสำหรับแผนภาพยูเอ็มแอล

การกำหนดองค์ประกอบสำคัญสำหรับแผนภาพยูเอ็มแอลของโครงงานนี้ จะอธิบายจากตัวอย่างรหัสต้นฉบับจากโค้ดตั้งต้นเมื่อสร้างโครงการใหม่ ดังนี้

```
10      [ApiController]
11      [Route("[controller]")]
        0 references
12      public class ControllerName : Controller
13      {
14          [HttpGet] // Type Method
15          [Route("[action]")] // Path
            0 references
16          public ActionResult<ReturnModel> MethodName(string inputParameter)
17          {
18              return new ReturnModel { Id = 0, Data = "" };
19          }
20      }
```

ภาพที่ 3.3 ตัวอย่างรหัสต้นฉบับ

จากภาพที่ 3.3 ตัวอย่างรหัสต้นฉบับ ซึ่งให้ความสำคัญกับองค์ประกอบต่าง ๆ ดังนี้

- ชื่อคอนโทรลเลอร์ คือ ControllerName
- ชื่อเมธอด คือ MethodName
- ชื่อของเส้นทางการสื่อสารข้อมูล [Route(“[action]”)] คือ /MethodName
- ชนิดของเมธอด คือ HttpGet
- พารามิเตอร์ที่เมธอดรับ string inputParameter
- ค่าที่เมธอดส่งออก คือ ReturnModel

3.3 กำหนดลำดับการทำงานของเครื่องมือ

กำหนดให้เครื่องมือค้นหาองค์ประกอบต่าง ๆ จากรหัสต้นฉบับ และนำมาเขียนให้อยู่ในรูปแบบของเครื่องมือ PlantUML โดยให้อยู่ในไฟล์ .puml ได้ดังตัวอย่างต่อไปนี้

1. เครื่องมืออ่านรหัสต้นฉบับและค้นหาคอนโทรลเลอร์จากตัวอย่างรหัสต้นฉบับ เครื่องมืออ่าน ControllerName ดังภาพที่ 3.4 ในบรรทัดที่ 12 จะถูกนำไปแปลงเป็น class ControllerName <<Application>> ในไฟล์ .puml

```
10      [ApiController]
11      [Route("[controller]")]
        0 references
12      public class ControllerName : Controller
13      {
14          [HttpGet] // Type Method
15          [Route("[action]")] // Path
            0 references
16          public ActionResult<ReturnModel> MethodName(string inputParameter)
17          {
18              return new ReturnModel { Id = 0, Data = "" };
19          }
20      }
```

ภาพที่ 3.4 ตัวอย่างรหัสต้นฉบับที่สนใจชื่อคอนโทรลเลอร์

2. เครื่องมืออ่านรหัสต้นฉบับเพื่อค้นหาเมธอดภายในคอนโทรลเลอร์ จากตัวอย่างรหัสต้นฉบับเครื่องมืออ่านเมธอดภายใน ControllerName ในบรรทัดที่ 16 ชื่อ MethodName() ดังภาพที่ 3.5 จะถูกนำไปแปลงเป็น class MethodName <<Resource>> ประเภท Get จากการประกาศ [HttpGet] ในบรรทัดที่ 14 และทำให้เกิดเส้นความสัมพันธ์ โดยกำหนดให้เป็นลูกศรเส้นประ ซึ่งระบุ Path ของการสื่อสารของข้อมูล จากการประกาศ [Route(“[action]”)] ในบรรทัดที่ 15 จึงถูกนำไปแปลงเป็น ControllerName ..> MethodName : <<Path>> /MethodName ในไฟล์ .puml

```
10      [ApiController]
11      [Route("[controller]")]
        0 references
12      public class ControllerName : Controller
13      {
14          [HttpGet] // Type Method
15          [Route("[action]")] // Path
            0 references
16          public ActionResult<ReturnModel> MethodName(string inputParameter)
17          {
18              return new ReturnModel { Id = 0, Data = "" };
19          }
20      }
```

ภาพที่ 3.5 ตัวอย่างรหัสต้นฉบับที่สนใจชื่อเมธอด

3. เครื่องมือค้นหาพารามิเตอร์ที่รับเข้าและส่งออกของเมธอด ในตัวอย่างต้นฉบับพบว่า มีพารามิเตอร์ที่รับเข้า คือ inputParemeter ชนิด string ดังภาพที่ 3.6 บรรทัดที่ 16 และเจอค่าพารามิเตอร์ที่เมธอดส่งออก เป็น ReturnModel ซึ่งเป็นพารามิเตอร์ที่มีคุณลักษณะภายใน จะถูกนำไปแปลงเป็น class ReturnModel <<Representation>> จึงทำให้เกิด

เส้นความสัมพันธ์โดยกำหนดให้เป็นลูกศรเส้นประ ซึ่งถูกนำไปแปลงเป็น MethodName ..> ReturnModel: <<use>> ในไฟล์ .puml เพื่อบ่งบอกถึงการกล่าวถึงจาก <<Resource>> หากพารามิเตอร์เป็นชนิดไม่มีคุณลักษณะภายใน จะไม่เกิดเส้นประที่บ่งบอกถึงการกล่าวถึงจาก <<Resource>>

```
10      [ApiController]
11      [Route("[controller]")]
        0 references
12      public class ControllerName : Controller
13      {
14          [HttpGet] // Type Method
15          [Route("[action]")] // Path
            0 references
16          public ActionResult<ReturnModel> MethodName(string inputParameter)
17          {
18              return new ReturnModel { Id = 0, Data = "" };
19          }
20      }
```

ภาพที่ 3.6 ตัวอย่างรหัสต้นฉบับที่สนใจพารามิเตอร์ที่เมธอดส่งออก

4. จากข้อ 3 เมื่อพารามิเตอร์ที่รับเข้าหรือส่งออก เป็นพารามิเตอร์แบบมีคุณลักษณะภายใน เครื่องมือจะทำการอ่านคุณลักษณะภายในต่อไป เช่น ค่าพารามิเตอร์ที่เมธอดส่งออกเป็น ReturnModel เครื่องมือจะทำการอ่านคุณลักษณะภายในของ ReturnModel จากรหัสต้นฉบับดังภาพที่ 3.7 จึงถูกนำไปเขียนอยู่ภายใต้คลาส ReturnModel โดยจะเขียนให้อยู่ในรูปแบบของไฟล์ .puml ได้ดังภาพที่ 3.8 ซึ่งเครื่องหมาย + ในภาพจะหมายถึง การประกาศตัวแปรที่เป็น public เป็นต้น

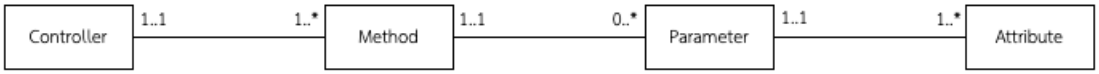
```
        2 references
5       public class ReturnModel
6       {
            1 reference
7           public int Id { get; set; }
            1 reference
8           public string Data { get; set; }
9       }
```

ภาพที่ 3.7 คุณลักษณะต่าง ๆ ของ ReturnModel

```
class ReturnModel <<Representation>>
{
    + Id : Int32
    + Data : String
}
```

ภาพที่ 3.8 คุณลักษณะต่าง ๆ ของ ReturnModel ที่อยู่ในรูปแบบของไฟล์ .puml

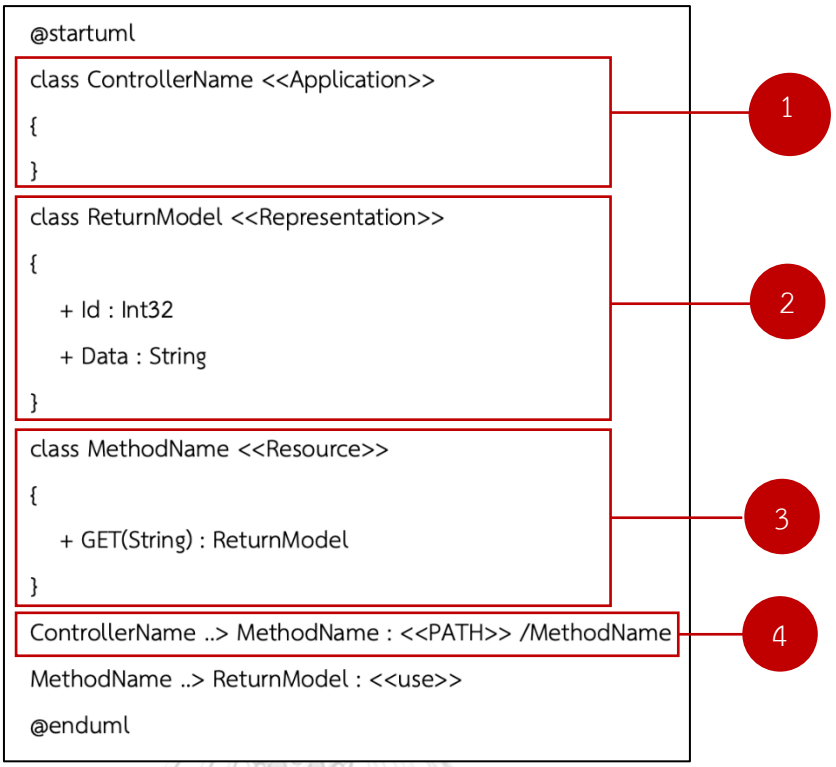
จากลำดับการอ่านโค้ด ทำให้ได้รูปแบบของโครงสร้างและความสัมพันธ์ของทรัพยากร ดังภาพที่ 3.9 ที่เกิดเป็นความสัมพันธ์ระหว่างคอนโทรลเลอร์กับเมธอด ซึ่งเมธอดจะมีความสัมพันธ์ไปยังพารามิเตอร์ที่สนใจ ทั้งพารามิเตอร์ที่เมธอดรับและพารามิเตอร์ที่เมธอดส่งออก เมื่อพารามิเตอร์มีคุณสมบัติภายใน ก็จะทำการอ่านไปถึงคุณลักษณะต่าง ๆ ภายในนั้น



ภาพที่ 3.9 รูปแบบของโครงสร้างและความสัมพันธ์ของทรัพยากร

3.4 กำหนดโครงร่างของไฟล์ .puml

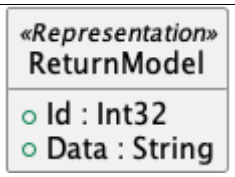
จากหัวข้อที่ 3.2 กำหนดองค์ประกอบสำคัญสำหรับแผลภาพยูเอ็มแอล และ หัวข้อที่ 3.3 กำหนดลำดับการทำงานของเครื่องมือ สามารถตัวอย่างรหัสต้นฉบับมาเขียนให้อยู่ในรูปแบบของไฟล์ .puml ได้ดังภาพ 3.10



ภาพที่ 3.10 ตัวอย่างโค้ดในไฟล์ .puml

ตารางที่ 3.1 ตารางเปรียบเทียบองค์ประกอบกับตัวอย่างรหัสต้นฉบับ

องค์ประกอบ	จากตัวอย่างรหัสต้นฉบับ	ตัวอย่างสัญลักษณ์บนแผนภาพ	ตัวอย่างโค้ดที่อยู่ในไฟล์ .puml
ชื่อ Controller	ControllerName	«Application» ControllerName	หมายเลข 1 จากภาพที่ 3.10
ชื่อเมธอด	MethodName	«Resource» MethodName GET(String) : ReturnModel	หมายเลข 3 จากภาพที่ 3.10
ชื่อของเส้นทางการสื่อสารข้อมูล	/MethodName	«PATH» /MethodName	หมายเลข 4 จากภาพที่ 3.10
ชนิดของเมธอด	GET	«Resource» MethodName GET(String) : ReturnModel	หมายเลข 3 จากภาพที่ 3.10
พารามิเตอร์ที่เมธอดรับ	string	«Resource» MethodName GET(String) : ReturnModel	หมายเลข 3 จากภาพที่ 3.10

องค์ประกอบ	จากตัวอย่างรหัสต้นฉบับ	ตัวอย่างสัญลักษณ์บนแผนภาพ	ตัวอย่างโค้ดที่อยู่ในไฟล์ .puml
ค่าที่เมธอดส่งออก	ReturnModel		หมายเลข 2 จากภาพที่ 3.10

จากตารางที่ 3.1 ได้เปรียบเทียบองค์ประกอบที่สำคัญกับตัวอย่างรหัสต้นฉบับในภาพที่ 3.3 และทำการแปลงรหัสต้นฉบับให้อยู่ในรูปแบบไฟล์ .puml เมื่อนำเครื่องมือที่พัฒนาขึ้นไปประยุกต์ใช้ โดยมีการอ่านรหัสต้นฉบับตามลำดับที่กำหนด จึงเกิดเป็นโค้ดของ PlantUML ตามรูปแบบของไฟล์ .puml ดังภาพที่ 3.10 หลังจากนั้นจึงทำการสร้างแผนภาพผ่านเครื่องมือ PlantUML ให้โดยอัตโนมัติ

3.5 การพัฒนาเครื่องมือ

ในขั้นตอนการพัฒนาเครื่องมือนี้ มีจุดประสงค์เพื่ออธิบายขั้นตอนในการพัฒนาเครื่องมือ และ นำเสนอวิธีการนำเครื่องมือที่พัฒนาขึ้นไปประยุกต์ใช้กับโครงการ โดยสามารถนำแผนภาพยูเอ็มแอลที่ได้จากเครื่องมือ ไปเป็นส่วนหนึ่งของเอกสารส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสได้ รายละเอียดการพัฒนาเครื่องมืออยู่ในบทที่ 4

3.6 การทดสอบและประเมินผล

ในขั้นตอนนี้ มีจุดประสงค์ในการทดสอบประสิทธิภาพของเครื่องมือที่ได้พัฒนาขึ้น โดยจะทำการนำเครื่องมือนี้ ประยุกต์ใช้เข้ากับโครงการอื่น ๆ ซึ่งจะต้องสามารถสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลได้

ในขั้นตอนของการทดสอบและประเมินผลนี้ กำหนดให้ทดสอบกับโครงการจำนวน 3 โครงการ โดยในแต่ละโครงการจะทำการนำ 3 คอนโทรลเลอร์ เพื่อทำการทดสอบความถูกต้องและครบถ้วนของเครื่องมือ รายละเอียดการทดสอบและประเมินผลอยู่ในบทที่ 5

บทที่ 4
การพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลอง

จากการออกแบบเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลอง ในบทที่ 3 สามารถนำมาพัฒนาต่อ โดยมีขั้นตอนและรายละเอียดของการพัฒนาเครื่องมือ ดังนี้

4.1 กำหนดสภาพแวดล้อมที่ใช้ในการออกแบบและพัฒนาเครื่องมือ

ในขั้นตอนนี้ เป็นการจำกัดสภาพแวดล้อมสำหรับการออกแบบและพัฒนาเครื่องมือ โดยมีรายละเอียด ดังนี้

4.1.1 กำหนดภาษาที่ใช้ในการพัฒนาแผนภาพจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล โดยใช้ภาษาซีชาร์ปเท่านั้น

4.1.2 กำหนดเฟรมเวิร์กที่ใช้ในการพัฒนาแผนภาพจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล โดยใช้ดอทเน็ตเฟรมเวิร์คเท่านั้น

4.1.3 กำหนดเครื่องมือเสริมที่ใช้ในการพัฒนาแผนภาพจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล โดยใช้เครื่องมือ PlantUML เท่านั้น

4.2 สร้างโครงการตัวอย่างสำหรับการออกแบบและพัฒนาเครื่องมือและทดสอบเครื่องมือเบื้องต้น

ผู้วิจัยได้ทำการสร้างโครงการขึ้นมาใหม่ผ่านการสร้างจากคอมมานไลน์ โดยใช้คำสั่ง “*dotnet new webapi --name WebApiProject --framework netcoreapp3.1*” ทำให้ได้โค้ดตั้งต้นโดยมีคอนโทรลเลอร์ 1 คอนโทรลเลอร์ คือ WeatherForecastController ดังภาพที่ 4.1 ผู้วิจัยจึงนำรหัสต้นฉบับนี้ มาเป็นตัวอย่างในการพัฒนาเครื่องมืออัตโนมัติสำหรับสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล

```
14      [HttpGet]
15      [Route("[action]")]
        0 references
16      public IEnumerable<WeatherForecast> GetTemperatureC()
17      {
18          var rng = new Random();
19          return Enumerable.Range(1, 5).Select(index => new WeatherForecast
20          {
21              Date = DateTime.Now.AddDays(index),
22              TemperatureC = rng.Next(-20, 55),
23              Summary = Summaries[rng.Next(Summaries.Length)]
24          })
25          .ToArray();
26      }
```

ภาพที่ 4.1 รหัสต้นฉบับตั้งต้นจากการสร้างโครงการขึ้นใหม่

4.3 กระบวนการทำงานของเครื่องมือที่พัฒนาขึ้น

ในขั้นตอนนี้ เป็นการนำลำดับการทำงานของเครื่องมือที่ได้กำหนดไว้ในบทที่ 3 ข้อ 3.3 มาพัฒนาเครื่องมือ โดยการสร้างสคริปต์สำหรับไฟล์ .puml

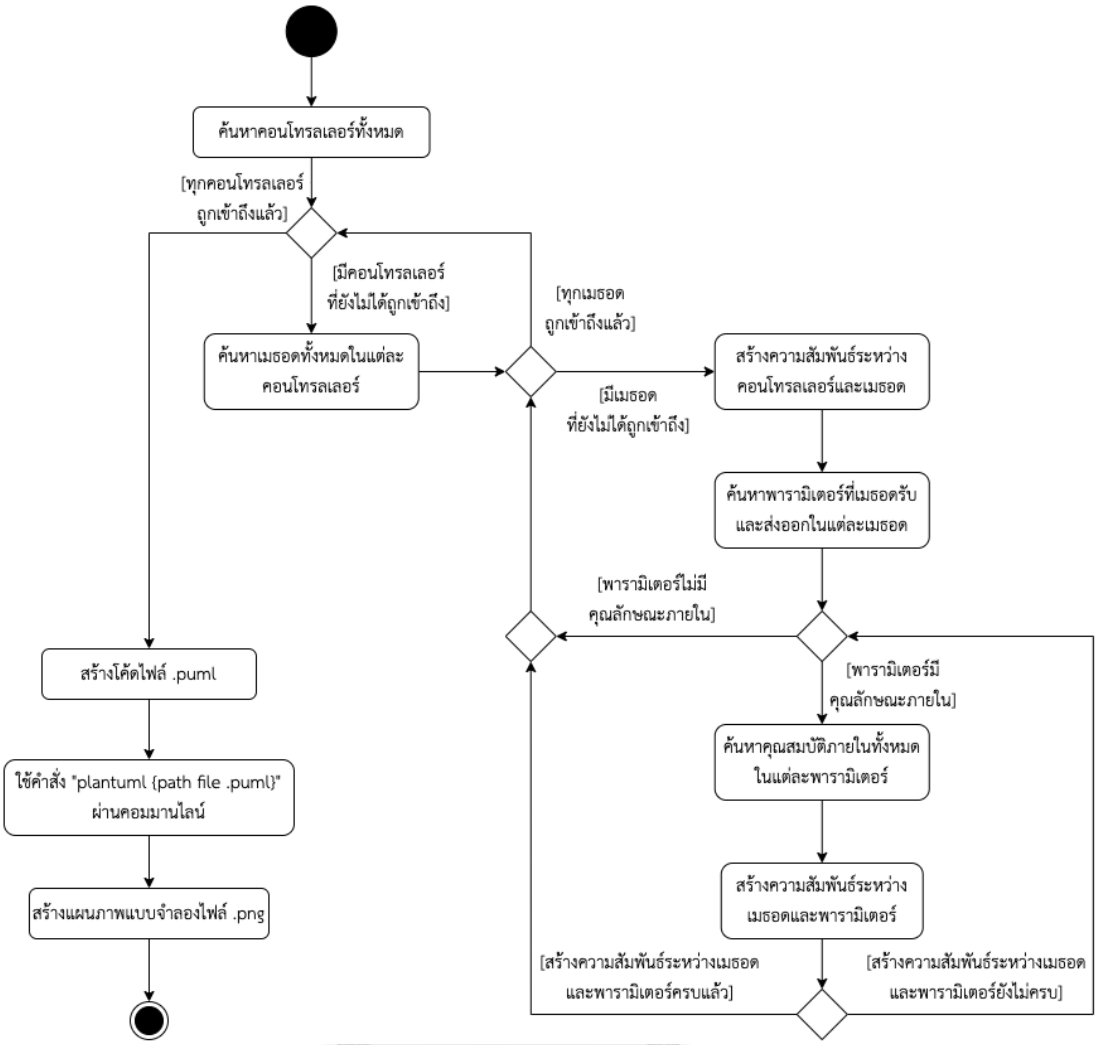
เริ่มจากการค้นหาคอนโทรลเลอร์ เมธอดภายใน พารามิเตอร์ที่รับเข้า และ พารามิเตอร์ที่ส่งออกทั้งหมดในโครงการ จากการใช้คำสั่งของดอทเน็ต ดังภาพ 4.2

```
73              Assembly asm = Assembly.GetExecutingAssembly();
74              var actionList = asm.GetTypes()
75              .Where(type => typeof(Controller).IsAssignableFrom(type))
76              .SelectMany(type => type.GetMethods(BindingFlags.Public))
77              .Select(x => new
78              {
79                  Controller = x.DeclaringType.Name,
80                  Method = x.Name,
81                  InputParam = x.GetParameters(),
82                  ReturnParam = x.ReturnType.GetGenericArguments(),
83
84                  TypeAPI = x.GetCustomAttributesData().Count() > 0 ? x.
                    GetCustomAttributesData().FirstOrDefault(c => c.AttributeType.
                    FullName.StartsWith("Microsoft.AspNetCore.Mvc.Http")).
                    AttributeType.FullName : ""
85              }).ToList();
86
```

ภาพที่ 4.2 คำสั่งที่ใช้สำหรับการค้นหาองค์ประกอบที่สำคัญ

จากภาพที่ 4.2 คำสั่งที่ใช้สำหรับการค้นหาองค์ประกอบที่สำคัญสำหรับการสร้างสคริปต์ไฟล์ .puml เพื่อนำไปสร้างแผนภาพ ดังนี้

- บรรทัดที่ 73 หมายถึง การค้นหาโค้ดที่มีการดำเนินอยู่บนโครงการนั้น
- บรรทัดที่ 74 หมายถึง นำโค้ดที่มีการดำเนินการอยู่บนโครงการนั้น มาหาองค์ประกอบภายใน
- บรรทัดที่ 75 หมายถึง โค้ดที่ดำเนินการอยู่บนโครงการนั้น จะสนใจเพียงแค่ประเภทคอนโทรลเลอร์เท่านั้น
- บรรทัดที่ 76 หมายถึง เมธอดที่ดำเนินการอยู่บนโครงการนั้น จะสนใจเพียงแค่ประเภท Public เท่านั้น
- บรรทัดที่ 77 ถึง 84 หมายถึง การนำองค์ประกอบที่สำคัญต่าง ๆ จากบรรทัดที่ 76 มาสร้างวัตถุใหม่
- บรรทัดที่ 79 หมายถึง การเลือกใช้ข้อมูลคอนโทรลเลอร์ โดยสนใจที่ชื่อของคอนโทรลเลอร์นั้น ๆ
- บรรทัดที่ 80 หมายถึง การเลือกใช้ข้อมูลเมธอด โดยสนใจที่ชื่อของเมธอดนั้น ๆ
- บรรทัดที่ 81 หมายถึง การเลือกใช้ข้อมูลพารามิเตอร์ที่รับเข้าเมธอด โดยสนใจพารามิเตอร์ทั้งหมดของเมธอดนั้น ๆ ซึ่งภายในมีข้อมูลชื่อประเภทของพารามิเตอร์และอาจมีมากกว่าหนึ่งตัว
- บรรทัดที่ 82 หมายถึง การเลือกใช้ข้อมูลพารามิเตอร์ที่ส่งออก ซึ่งภายในมีข้อมูลชื่อ และประเภทของพารามิเตอร์
- บรรทัดที่ 84 หมายถึง การเลือกใช้ข้อมูลที่เป็นประเภทของเมธอดนั้น ๆ ยกตัวอย่างข้อมูล Get, Put, Post, Delete เป็นต้น
- บรรทัดที่ 85 หมายถึง การนำวัตถุที่สร้างขึ้นใหม่ จัดเก็บให้อยู่ในตัวแปรประเภทรายการ (List) เพื่อนำไปใช้งานต่อสำหรับการสร้างแผนภาพ จากหัวข้อ 3.3 สามารถนำมาเขียนแผนภาพที่แสดงถึงตรรกะการสร้างสคริปต์ไฟล์ .puml ดังภาพที่ 4.3 เพื่อนำไปสร้างแผนภาพแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล



ภาพที่ 4.3 ตรรกะการทำงานของเครื่องมือที่พัฒนาขึ้น

จากภาพที่ 4.3 ตรรกะการทำงานของเครื่องมือที่พัฒนาขึ้น มีลำดับขั้นตอน ดังนี้

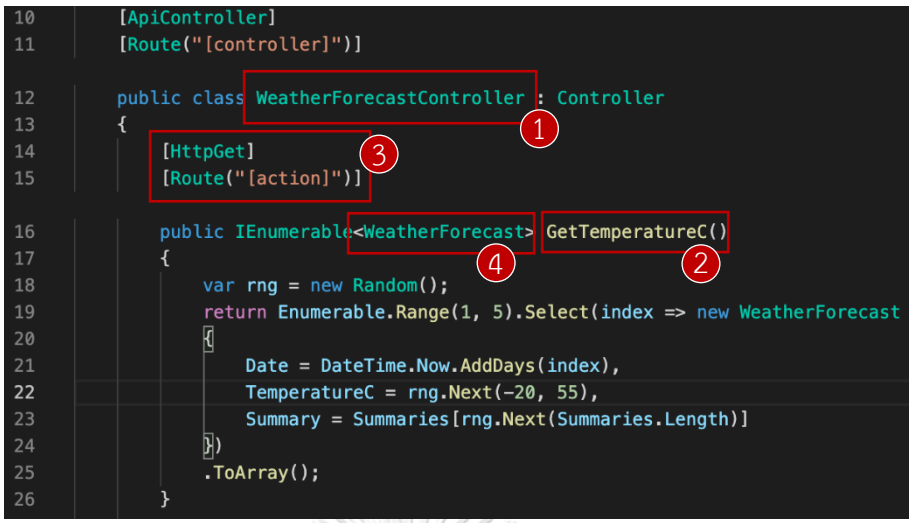
1. ค้นหาคอนโทรลเลอร์ทั้งหมดภายในโครงการ จะได้รายการคอนโทรลเลอร์ทั้งหมด เพื่อนำไปค้นหาเมธอดภายใน และสร้างตัวแปรหนึ่ง เพื่อนำมาเก็บข้อความสำหรับสคริปต์ไฟล์ .puml ในหัวข้อนี้กำหนดให้เป็นตัวแปรที่ชื่อ script ดังนั้นเครื่องมือจึงสร้างข้อความสำหรับคลาสคอนโทรลเลอร์ คือ “class {ControllerName} <<Application>>”
2. ค้นหาเมธอดในแต่ละคอนโทรลเลอร์ จะได้รายการของเมธอดทั้งหมดในคอนโทรลเลอร์นั้น ๆ และเพิ่มข้อความสำหรับเมธอดในตัวแปร script คือ “class {MethodName} <<Resource>>” ซึ่งภายในคลาสเมธอดจะระบุประเภทของเมธอด พารามิเตอร์ที่รับเข้าและพารามิเตอร์ที่ส่งออก หากเมธอดทั้งหมดถูกเข้าถึงแล้ว เครื่องมือจะกลับไปเช็คเงื่อนไข

การเข้าถึงของคอนโทรลเลอร์อีกครั้ง แต่ถ้าหากมีเมธอดที่ยังไม่ได้ถูกเข้าถึง เครื่องมือจะทำการอ่านชื่อของเมธอด เพื่อนำไปสร้างเป็นความสัมพันธ์ระหว่างคอนโทรลเลอร์และเมธอดนั้น ๆ โดยกำกับข้อความบนเส้นประเป็นเส้นทางการสื่อสารของข้อมูล และเพิ่มข้อความสำหรับการระบุถึงความสัมพันธ์ระหว่างคอนโทรลเลอร์และเมธอดนั้น ๆ คือ “{ControllerName} ..> {MethodName} : <<PATH>> /{MethodName}”

3. ค้นหาพารามิเตอร์ทั้งหมดของเมธอด ประกอบด้วยพารามิเตอร์ที่เมธอดรับ พารามิเตอร์ที่เมธอดส่งออก และเพิ่มข้อความสำหรับพารามิเตอร์ในตัวแปร script คือ “class {Parameter} <<Representation>>” หากพารามิเตอร์ไม่มีคุณลักษณะภายใน เครื่องมือจะไปเช็คเงื่อนไขการเข้าถึงของเมธอดอีกครั้ง แต่ถ้าหากพารามิเตอร์มีคุณลักษณะภายในเครื่องมือจะทำการค้นหาคุณลักษณะภายในทั้งหมด และนำชื่อของพารามิเตอร์นั้นไปสร้างเป็นความสัมพันธ์ระหว่างเมธอดและพารามิเตอร์ ซึ่งจะทำการเพิ่มข้อความในตัวแปร script โดยนำคุณลักษณะระบุภายใต้คลาสของพารามิเตอร์ และเพิ่มข้อความสำหรับการระบุถึงความสัมพันธ์ระหว่างเมธอดและพารามิเตอร์ คือ “{MethodName} ..> {Parameter} : <<user>>” หากยังมีพารามิเตอร์ที่มีคุณลักษณะภายในที่ยังไม่ได้สร้างความสัมพันธ์ เครื่องมือจะทำการค้นหาคุณลักษณะภายในของพารามิเตอร์ถัดไป
4. เครื่องมืออ่านคอนโทรลเลอร์ครบทุกคอนโทรลเลอร์แล้ว เครื่องมือจะทำการนำตัวแปร script ที่เก็บข้อความสำหรับการสร้างโค้ดไฟล์ .puml มาสร้างเป็นไฟล์ .puml เพื่อใช้สำหรับการสร้างแผนภาพผ่านเครื่องมือเสริม PlantUML
5. จากข้อ 4 เครื่องมือจะนำไฟล์ .puml ที่ได้ ดำเนินการผ่านคอมมานไลน์โดยใช้คำสั่ง “plant uml {path file .puml} ให้อัตโนมัติ
6. จากข้อ 5 เมื่อดำเนินการโดยใช้คำสั่งผ่านคอมมานไลน์เรียบร้อยแล้ว จะทำให้ได้แผนภาพแบบจำลองความสัมพันธ์นามสกุล .png โดยอัตโนมัติ

4.4 โครงสร้างของโค้ดในไฟล์ .puml ที่ได้จากการอ่านของเครื่องมือ

จากหัวข้อที่ 4.3 เมื่อนำตรรกะภายในของเครื่องมือ ไปเขียนโค้ดและทำการเรียกใช้เครื่องมือที่ได้พัฒนาขึ้น เครื่องมือจะทำการสร้างไฟล์ .puml โดยอัตโนมัติ ซึ่งสามารถอธิบายลำดับการอ่านโค้ด โดยสนใจที่ WeatherForcastController ที่มาจากการสร้างโครงการตัวอย่างเพื่อใช้ทดสอบเบื้องต้น ดังภาพที่ 4.4



ภาพที่ 4.4 ตัวอย่างรหัสต้นฉบับ WeatherForcastController

จากภาพที่ 4.4 เครื่องมือจะทำการอ่านรหัสต้นฉบับโดยเรียงลำดับดังนี้

1. ชื่อของคอนโทรลเลอร์ ในหมายเลขที่ 1 เป็น WeatherForcastController ทำให้ได้ออกมาเป็นคลาส <<Application>> WeatherForcastController
2. ชื่อของเมธอด ในหมายเลขที่ 2 เป็น GetTemperatureC ทำให้ได้ออกมาเป็นคลาส <<Resource>> GetTemperatureC
3. ประเภทของเมธอด และเส้นทางการสื่อสารข้อมูล ในหมายเลขที่ 3 เกิดเป็นข้อมูลที่ระบุอยู่ภายในคลาส <<Resource>> GetTemperatureC จะได้ Get() : WeatherForcast หมายความว่า เป็นเมธอดประเภท Get ที่ไม่มีการรับพารามิเตอร์ แต่มีค่าส่งกลับเป็น WeatherForcast และมีเส้นเชื่อมจากคลาส <<Application>> WeatherForcastController ที่ระบุเส้นทางการสื่อสารข้อมูลเป็น <<PATH>> /GetTemperatureC
4. ค่าที่ส่งกลับ ในหมายเลขที่ 4 เป็น WeatherForcast เนื่องจาก WeatherForcast เป็นพารามิเตอร์ที่มีคุณลักษณะภายใน เครื่องมือจะทำการอ่านคุณลักษณะภายในทั้งหมด จึงทำให้เกิดเป็นคลาส <<Representation>> WeatherForcast

เมื่อเครื่องมืออ่านโค้ดตามลำดับที่ได้กำหนดไว้ เครื่องมือจะทำการสร้างไฟล์ .puml โดยอัตโนมัติ ดังภาพที่ 4.5

```
@startuml
class WeatherForecastController <<Application>>
{
}
class WeatherForecast <<Representation>>
{
    + Date : DateTime
    + TemperatureC : Int32
    + TemperatureF : Int32
    + Summary : String
}
class GetTemperatureC <<Resource>>
{
    + GET() : WeatherForecast
}
WeatherForecastController ..> GetTemperatureC : <<PATH>> /GetTemperatureC
GetTemperatureC ..> WeatherForecast : <<use>>
@enduml
```

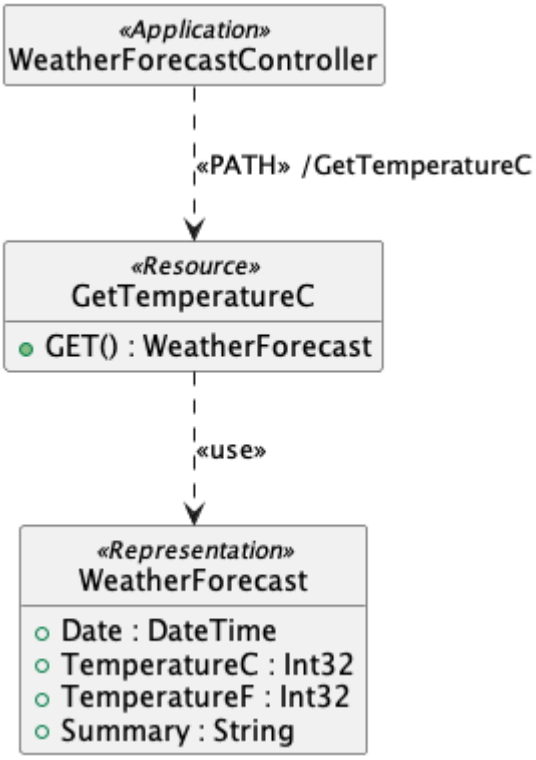
ภาพที่ 4.5 ตัวอย่างโค้ดสำหรับการสร้างแผนภาพตามรูปแบบของ PlantUML

จากโค้ดในภาพที่ 4.5 เครื่องมือจะใช้คำสั่ง plantuml ตามด้วยที่อยู่ของไฟล์ .puml นั้น ตัวอย่างดังภาพที่ 4.6 เครื่องมือจะใช้คำสั่งผ่านคอมมานไลน์ให้อัตโนมัติ

```
// Run command
Process p = new Process();
p.StartInfo.FileName = "/bin/bash";
p.StartInfo.UseShellExecute = true;
p.StartInfo.Arguments = "plantuml PlantUml/WeatherForecastController.puml";
p.Start();
```

ภาพที่ 4.6 ตัวอย่างคำสั่งที่ใช้สำหรับสร้างแผนภาพ

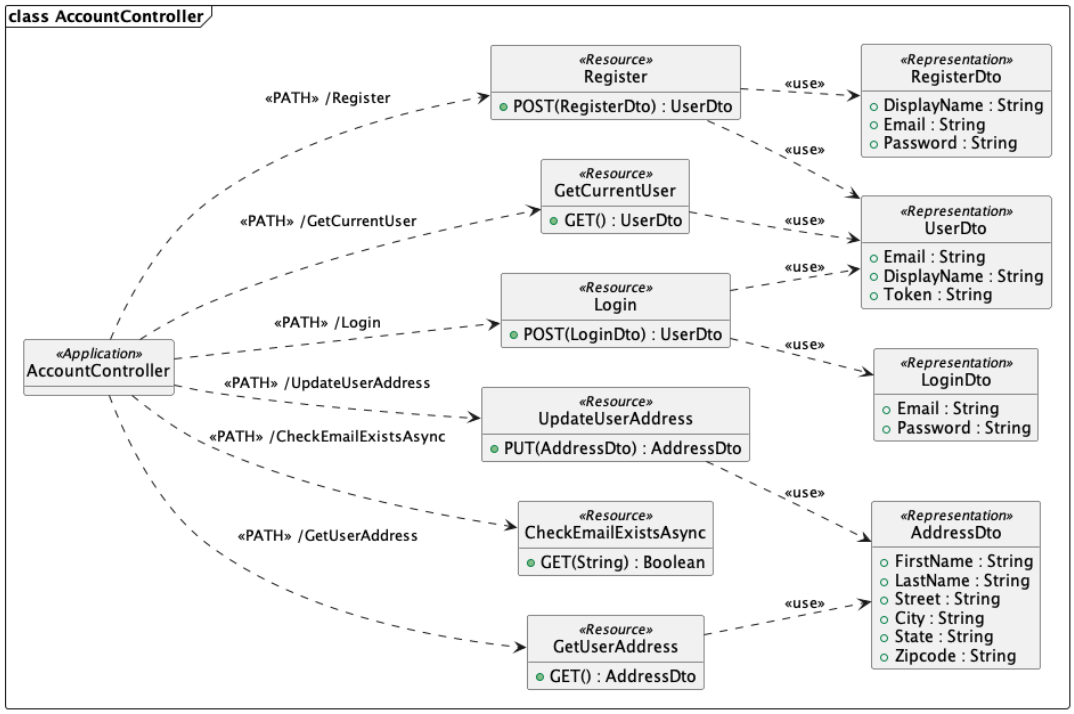
เมื่อใช้คำสั่งดังภาพที่ 4.6 จะทำให้ได้แผนภาพแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลของ WeatherForecastController ดังภาพที่ 4.7



ภาพที่ 4.7 ตัวอย่างแผนภาพที่ได้จากเครื่องมือ

4.5 ทดลองประยุกต์ใช้เครื่องมือที่พัฒนาขึ้น

นำตัวอย่างรหัสต้นฉบับโดยสนใจที่ AccountController เป็นแนวทาง จากโครงการ E–CommerceApp [5] เมื่อเครื่องมือทำการอ่านคอนโทรลเลอร์ ทำให้ได้ไฟล์ .puml และได้ดำเนินการใช้คำสั่งสร้างแผนภาพ “plantuml {path file .puml}” โดยอัตโนมัติ ทำให้เกิดเป็นแผนภาพที่แสดงพฤติกรรมและโครงสร้างของ AccountController



ภาพที่ 4.8 แผนภาพที่ได้จากการพัฒนาเครื่องมือ

จากภาพที่ 4.8 มีองค์ประกอบหลักด้วยกัน 3 ส่วน คือ <<Application>>, <<Resource>>, <<Representation>> ซึ่งมีรายละเอียด ดังนี้

- <<Application>> เป็นส่วนที่นำมาจาก Controller ที่เป็นส่วนควบคุม และรับคำร้องขอจากผู้ใช้งาน ส่วนที่เป็น Controller คือ <<Application>> Account Controller
- <<Resource>> เป็นส่วนที่นำมาจากแอ็กชัน จาก Account Controller มีแอ็ก ชันอยู่ภายในทั้งหมด 6 แอ็กชัน ได้แก่ Login, GetCurrentUser, Register, CheckEmailExistsAsync, GetUserAddress, UpdateUserAddress ซึ่งมีวิธีดำเนินงานระบุอยู่ภายใน และมีเส้นทางการเดินทางของข้อมูลกำกับที่เส้นระหว่าง <<Application>> และ <<Resource>> ซึ่งจะใช้เป็นสัญลักษณ์ <<PATH>> ยกตัวอย่างส่วนที่เป็นแอ็กชัน เช่น <<Resource>> Login ซึ่งมีวิธีดำเนินงาน POST(LoginDto): UserDto หมายความว่ามีการรับพารามิเตอร์ LoginDto และมีการส่งข้อมูลกลับเป็น UserDto โดยเส้นกำกับระหว่าง <<Application>> และ <<Resource>> คือ <<PATH>> /Login เป็นต้น
- <<Representation>> เป็นส่วนที่นำมาจากการประกาศทรัพยากรภายในระบบหรือเรียกอีกอย่างว่า โมเดล จาก Account Controller ที่มีวิธีดำเนินงานต่าง ๆ ทำให้เกิด

ความสัมพันธ์กับทรัพยากรอีก 4 ทรัพยากร ได้แก่ LoginDto, UserDto, RegisterDto, AddressDto ซึ่งมีการระบุคุณลักษณะของทรัพยากรอยู่ภายใน และมีการกำกับบนเส้นระหว่าง <<Resource>> และ <<Representation>> ซึ่งจะใช้เป็นสัญลักษณ์ <<use>> ยกตัวอย่าง เช่น <<Resource>> Login ที่มีวิธีดำเนินงาน POST(LoginDto): UserDto หมายความว่า การรับพารามิเตอร์เป็น LoginDto มีการเรียกใช้ทรัพยากร LoginDto ทำให้เกิดเป็นความสัมพันธ์ระหว่าง <<Resource>> Login กับ <<Representation>> LoginDto รวมทั้งมีการส่งข้อมูลกลับเป็น UserDto ทำให้เกิดความสัมพันธ์ระหว่าง <<Resource>> Login กับ <<Representation>> UserDto เป็นต้น

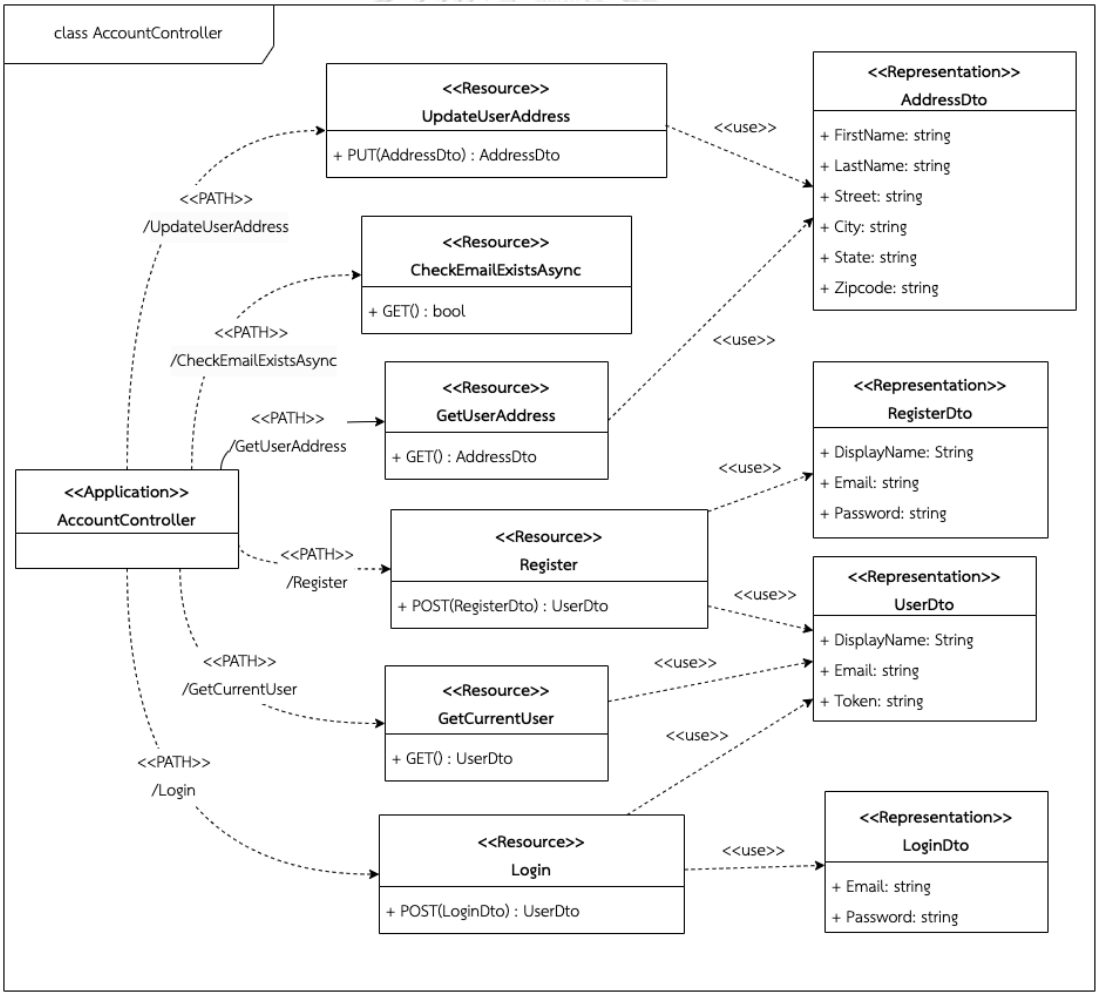
บทที่ 5

การทดสอบและประเมินผลความสามารถของเครื่องมือ

5.1 ทดสอบและประเมินเครื่องมือที่พัฒนาขึ้น

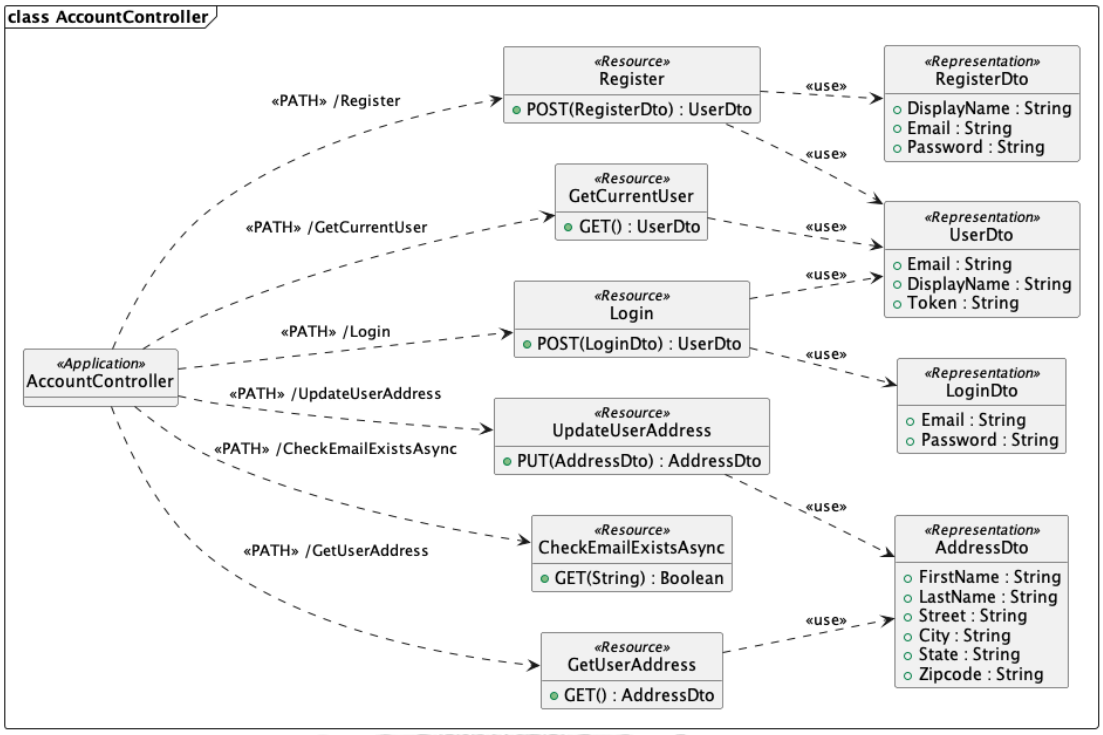
ในขั้นตอนนี้ มีจุดประสงค์ในการทดสอบประสิทธิภาพของเครื่องมือที่ได้พัฒนาขึ้น โดยจะทำการนำเครื่องมือนี้ ประยุกต์ใช้เข้ากับโครงการอื่น ๆ ซึ่งจะต้องสามารถสร้างแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลได้ ที่ประกอบด้วยคอนโทรลเลอร์ เส้นทางการสื่อสารข้อมูล และทรัพยากรที่เกี่ยวข้องกับเส้นทางการสื่อสารข้อมูลนั้น ๆ โดยกำหนดขั้นตอนในการนำรหัสต้นฉบับของตัวอย่างโครงการต่าง ๆ มาทดสอบ ดังนี้

1. นำรหัสต้นฉบับมาสร้างแผนภาพโดยผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.1 ที่แสดงตัวอย่างแผนภาพโดยผู้ทดสอบเป็นผู้วาดจากคอนโทรลเลอร์ในโครงการ E-CommerceApp



ภาพที่ 5.1 ตัวอย่างแผนภาพที่วาดโดยผู้ทดสอบ

2. นำแผนภาพที่ผู้ทดสอบเป็นผู้วาดดังภาพที่ 5.1 มาเปรียบเทียบกับแผนภาพที่ได้จากการสร้างโดยเครื่องมือดังภาพที่ 5.2 ที่แสดงตัวอย่างแผนภาพที่สร้างโดยเครื่องมือที่พัฒนาขึ้น และประยุกต์ใช้เครื่องมือกับโครงการ E–CommerceApp



ภาพที่ 5.2 ตัวอย่างแผนภาพที่ได้จากการสร้างของเครื่องมือ

ซึ่งการทดสอบและประเมินนี้ จะถูกทดสอบโดยการนำรหัสต้นฉบับมาสร้างแผนภาพโดยผู้ทดสอบเป็นผู้วาดจำนวน 3 คอนโทรลเลอร์จากแต่ละโครงการใน 3 โครงการ ผลการทดสอบที่ได้รับคือ แผนภาพที่เกิดจากการวาดโดยผู้ทดสอบและแผนภาพที่เกิดจากเครื่องมือจะต้องได้แผนภาพที่มีข้อมูลตรงกันทั้งหมด ซึ่งการเปรียบเทียบแผนภาพเพื่อตรวจสอบนี้ จะให้ความสำคัญกับความถูกต้องและความครบถ้วนของข้อมูลเท่านั้น และจะมองข้ามรูปแบบการจัดวางของแผนภาพ เช่น ตำแหน่ง ช่องว่าง สี และลักษณะอักษร เป็นต้น

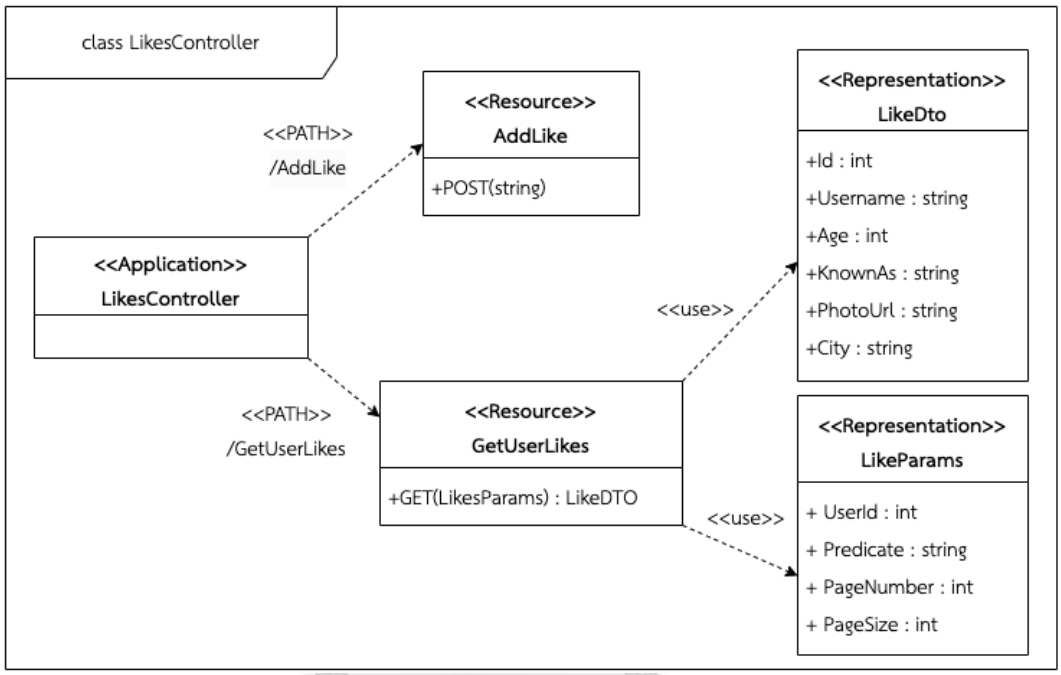
5.2 ผลการนำเครื่องมือที่พัฒนาขึ้นไปประยุกต์ใช้

เมื่อนำเครื่องมือที่พัฒนาขึ้นไปประยุกต์ใช้ เครื่องมือที่พัฒนาขึ้นจะทำการอ่านคอนโทรลเลอร์ทั้งหมดของโครงการนั้น ๆ และทำการสร้างแผนภาพโดยอัตโนมัติ จึงได้แผนภาพแบบจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูลจากการสร้างของเครื่องมือและนำไปเปรียบเทียบกับแผนภาพที่ได้จากผู้ทดสอบเป็นผู้วาด ดังนี้

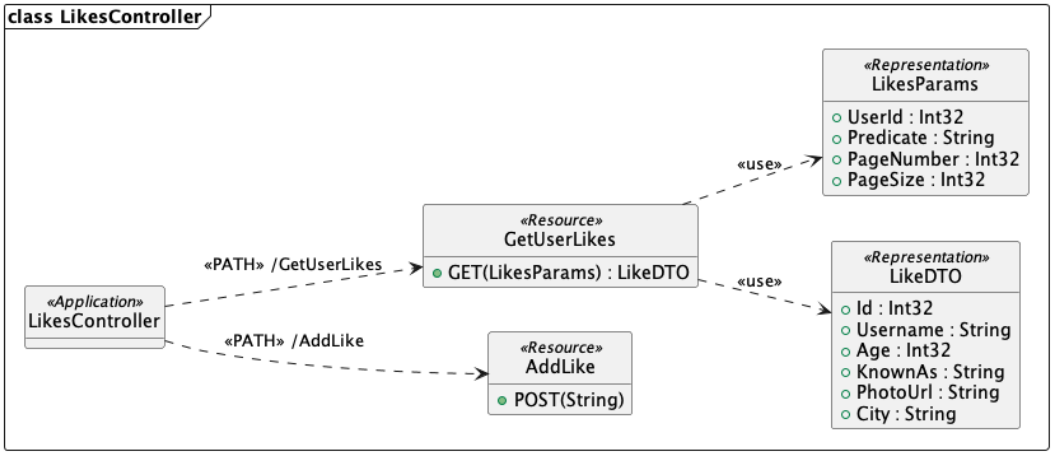
1. โครงการ Dating [15]

นำคอนโทรลเลอร์ 3 คอนโทรลเลอร์จากโครงการ Dating มาทำการทดสอบ โดยสนใจที่คอนโทรลเลอร์ต่าง ๆ ดังนี้

- LikesController โดยมี LikesController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.3 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.4

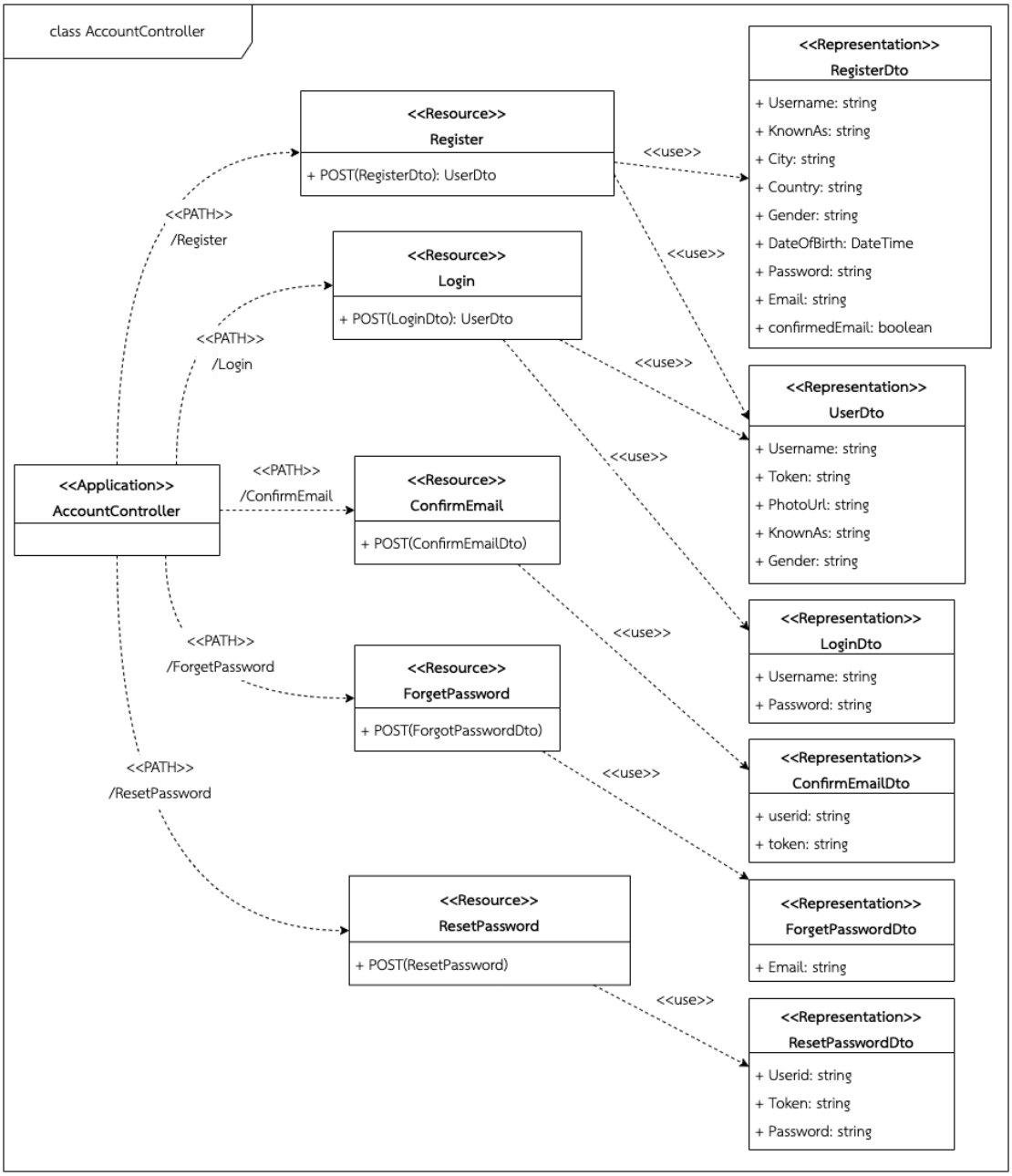


ภาพที่ 5.3 แผนภาพของ LikesController จากผู้ทดสอบ

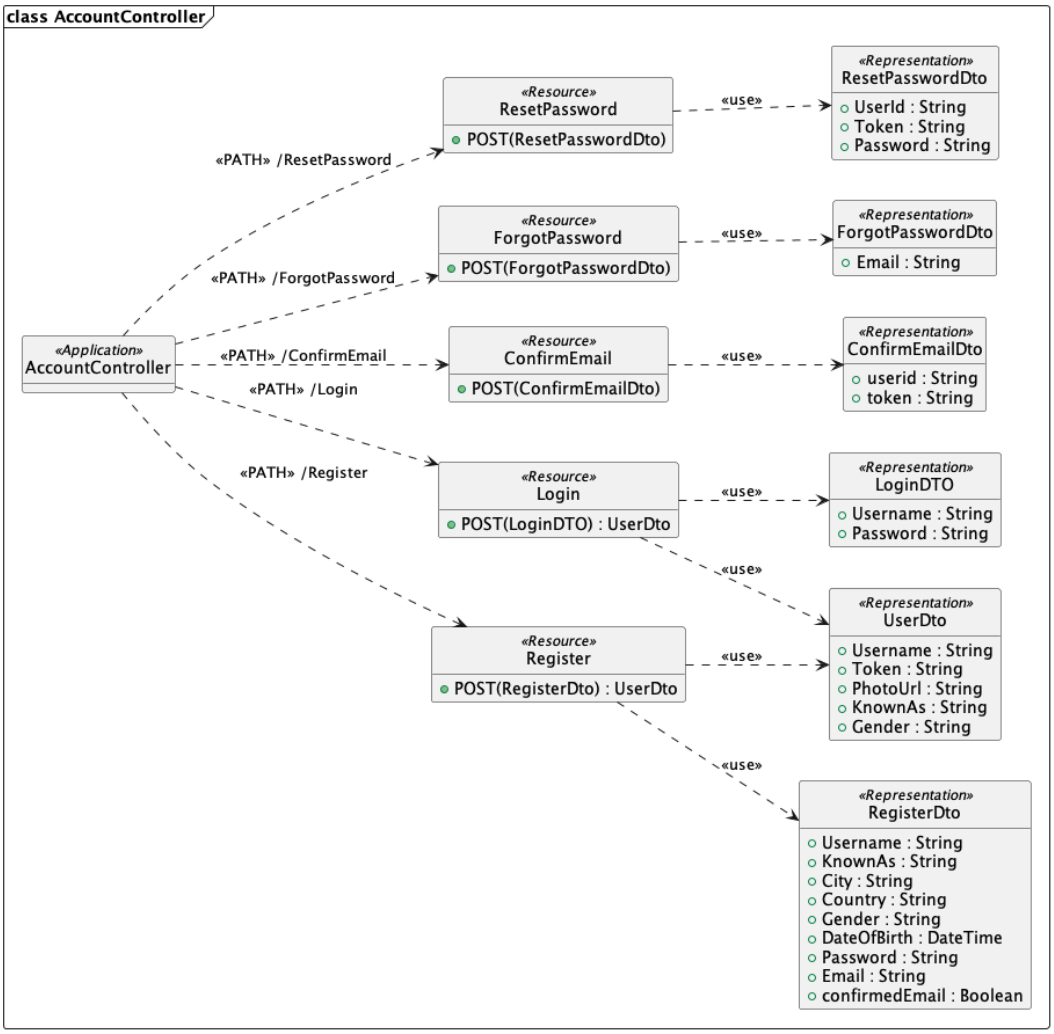


ภาพที่ 5.4 แผนภาพของ LikesController จากเครื่องมือ

- AccountController โดยมี AccountController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.5 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.6

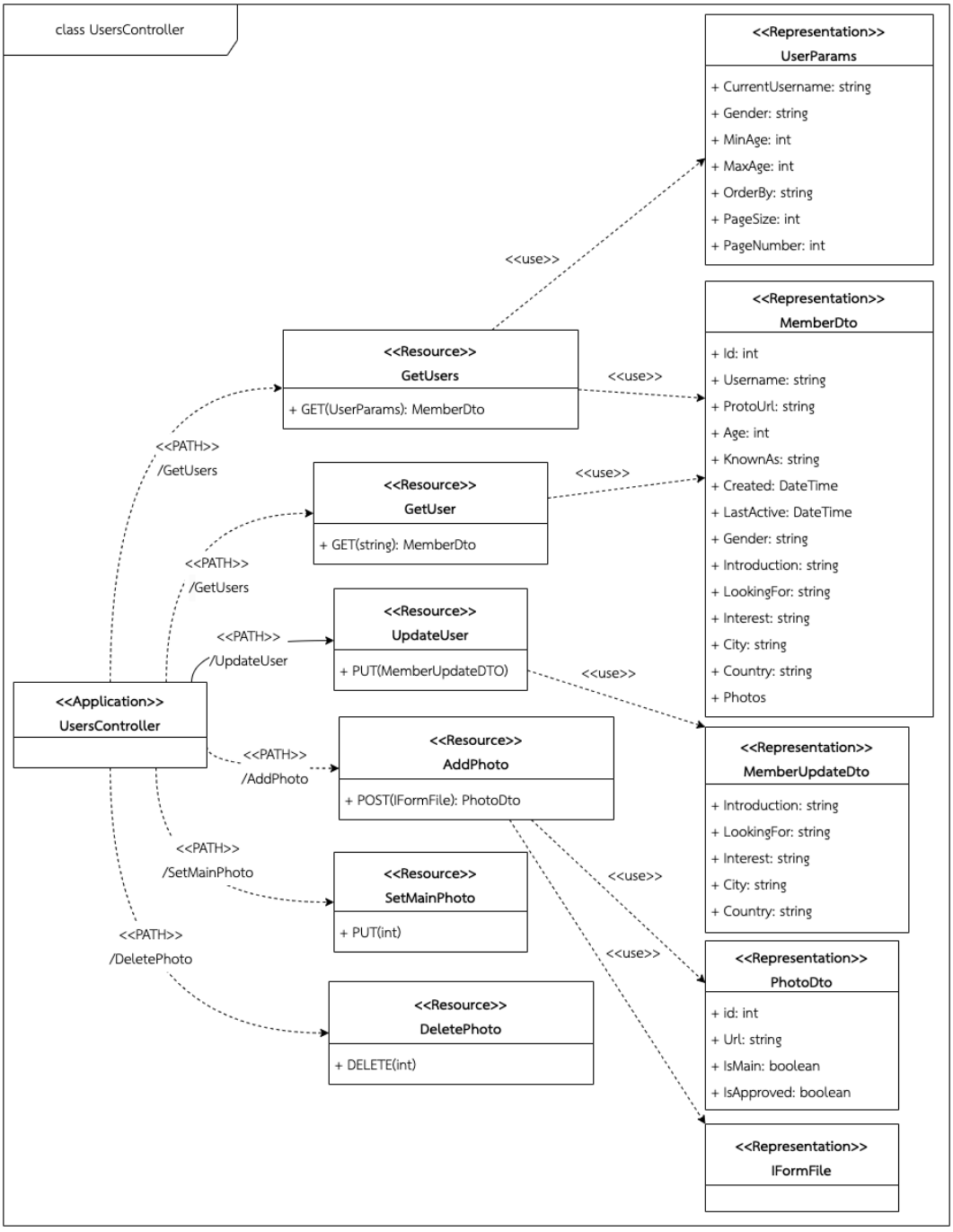


ภาพที่ 5.5 แผนภาพของ AccountController จากผู้ทดสอบ

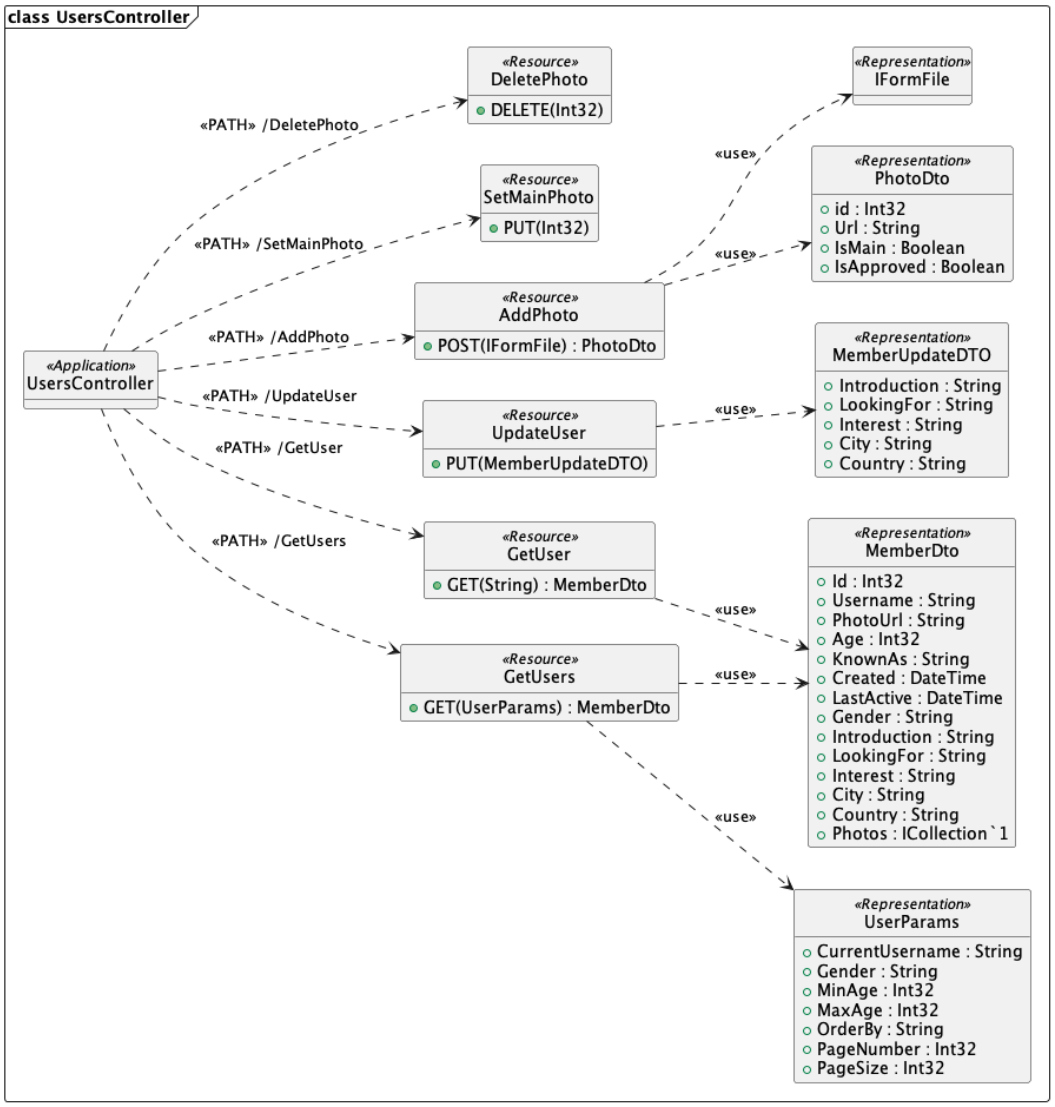


ภาพที่ 5.6 แผนภาพของ AccountController จากเครื่องมือ

- UsersController โดยมี UsersController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.7 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.8



ภาพที่ 5.7 แผนภาพของ UsersController จากผู้ทดสอบ

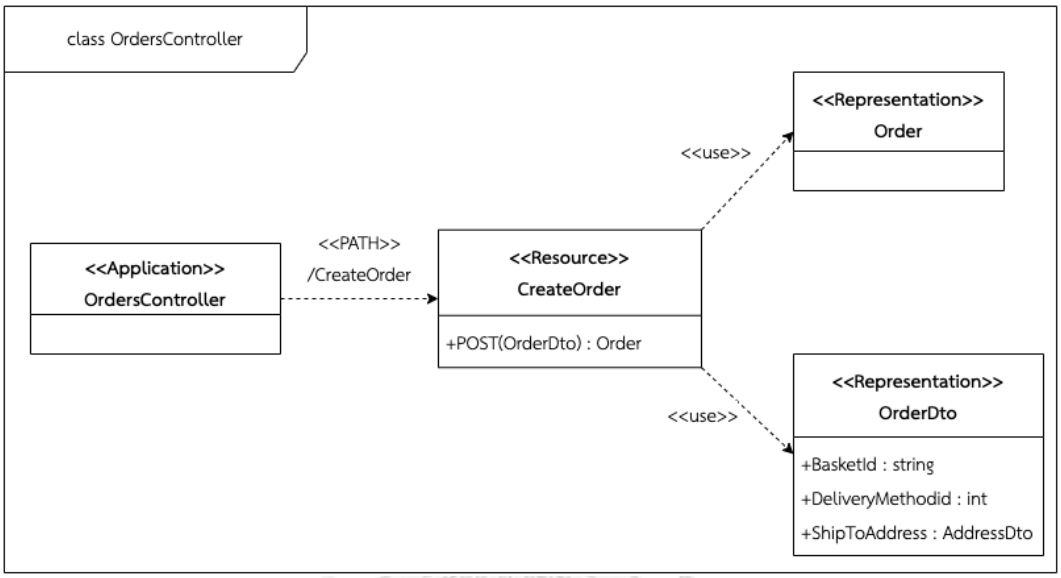


ภาพที่ 5.8 แผนภาพของ UsersController จากเครื่องมือ

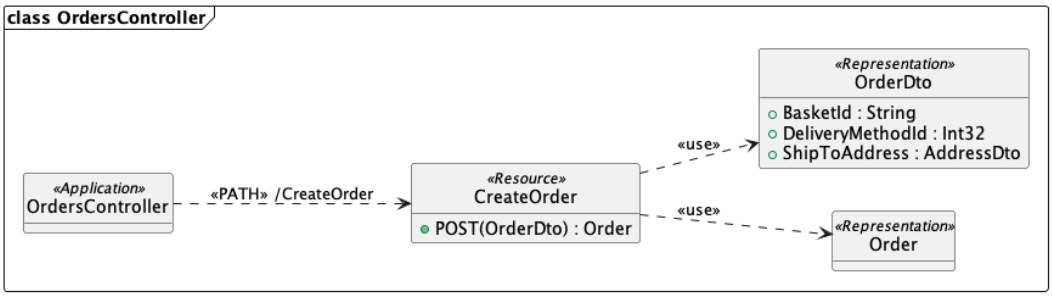
2. โครงการ eCommerce [5]

นำคอนโทรลเลอร์ 3 คอนโทรลเลอร์จากโครงการ eCommerce มาทำการทดสอบ โดยสนใจที่คอนโทรลเลอร์ต่าง ๆ ดังนี้

- OrdersController โดยมี OrdersController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.9 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.10

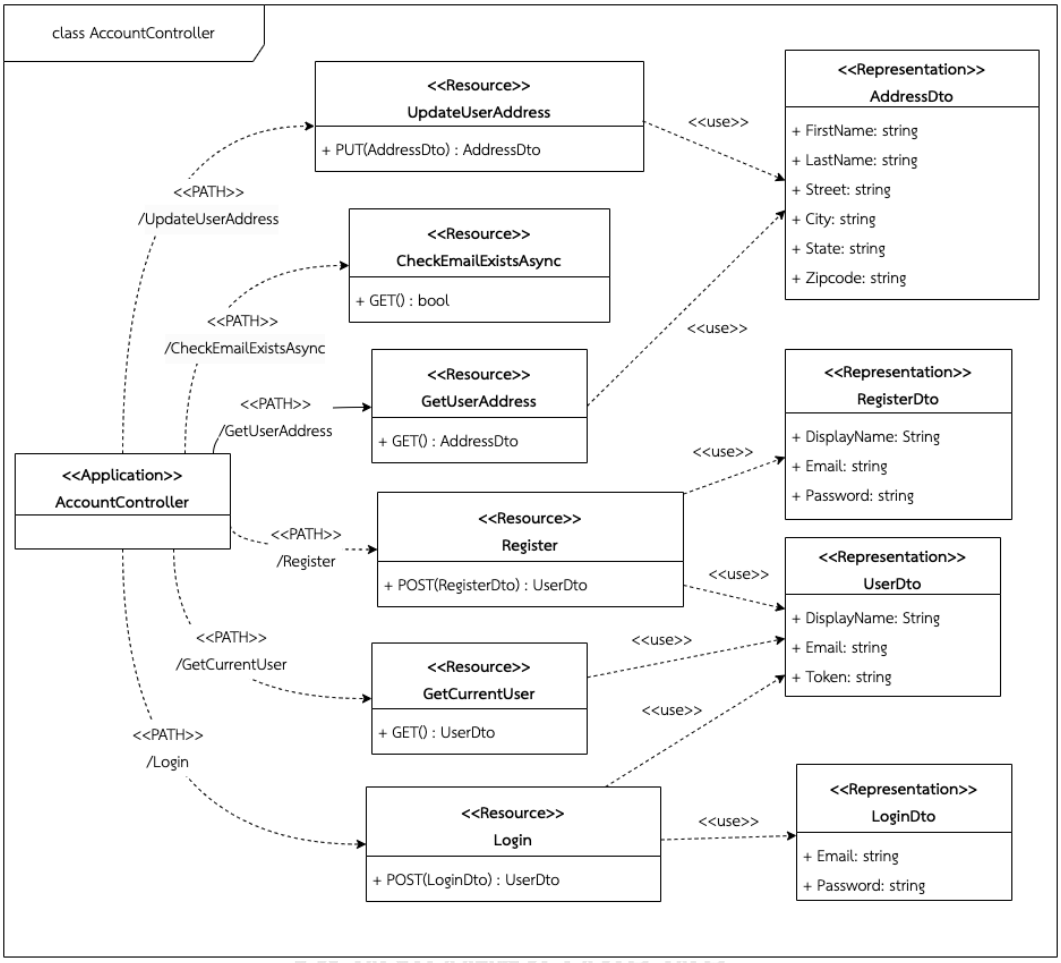


ภาพที่ 5.9 แผนภาพของ OrdersController จากผู้ทดสอบ

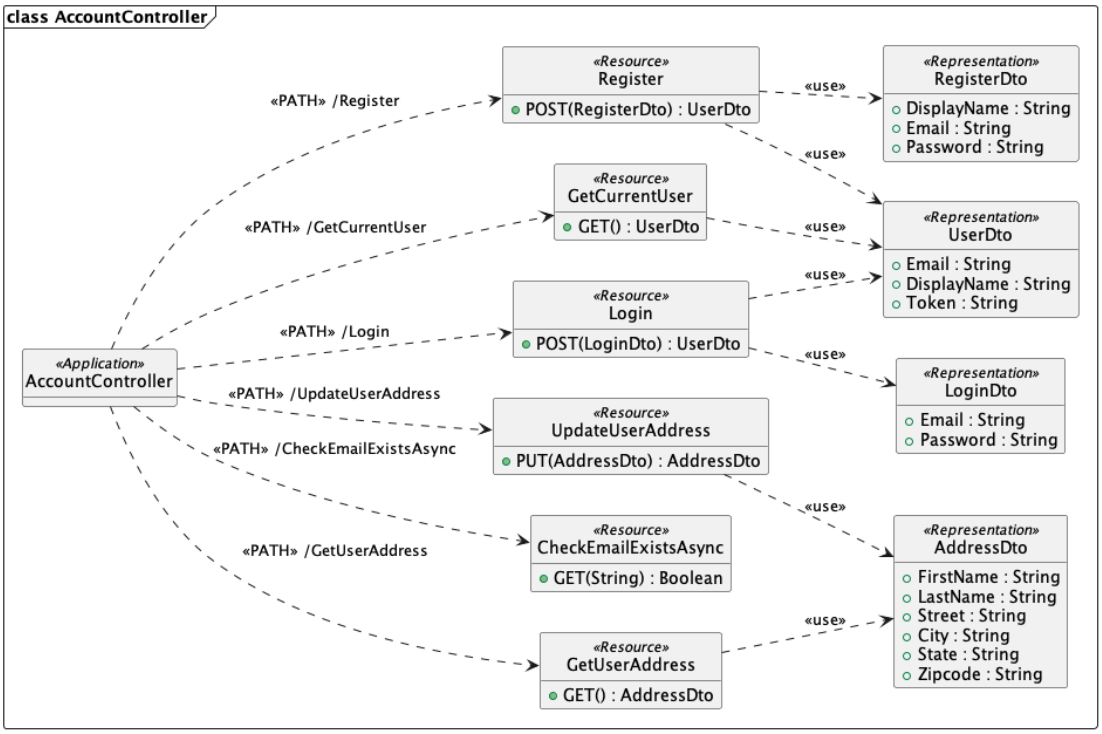


ภาพที่ 5.10 แผนภาพของ OrdersController จากเครื่องมือ

- AccountController โดยมี AccountController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.11 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.12

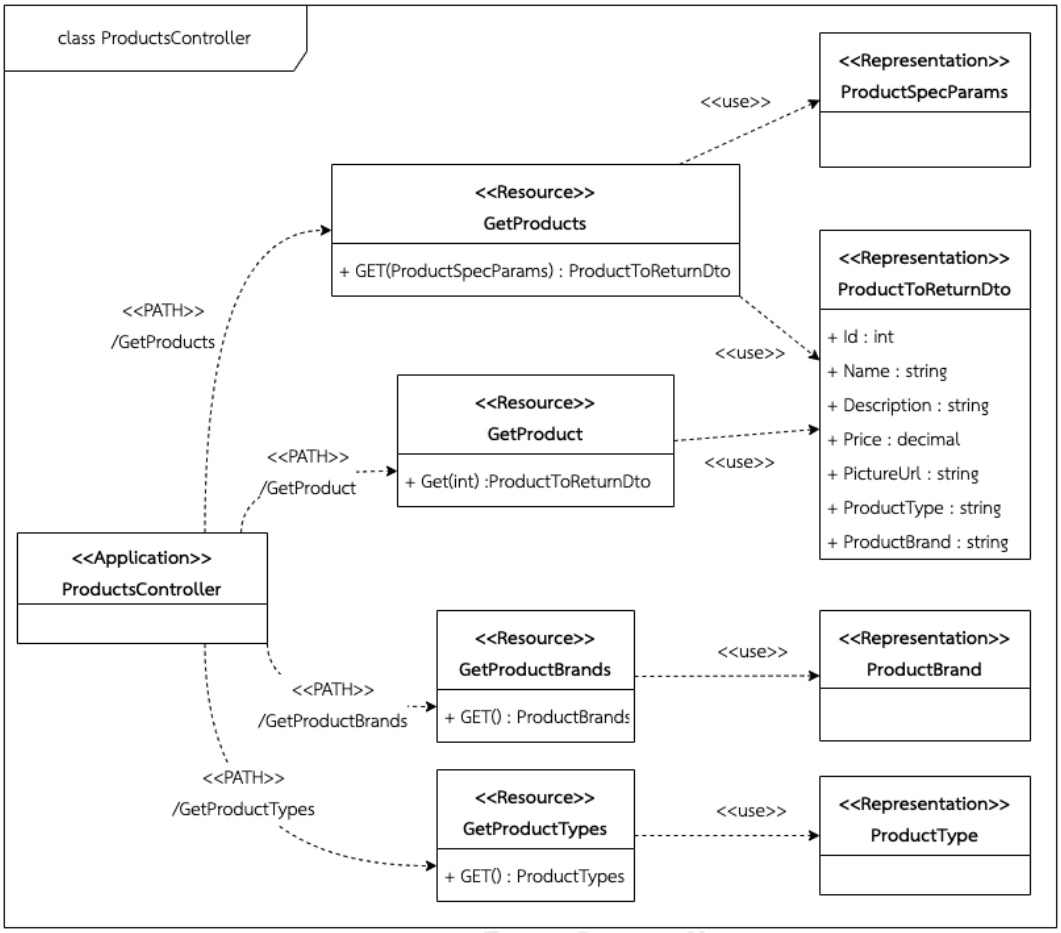


ภาพที่ 5.11 แผนภาพของ AccountController จากผู้ทดสอบ

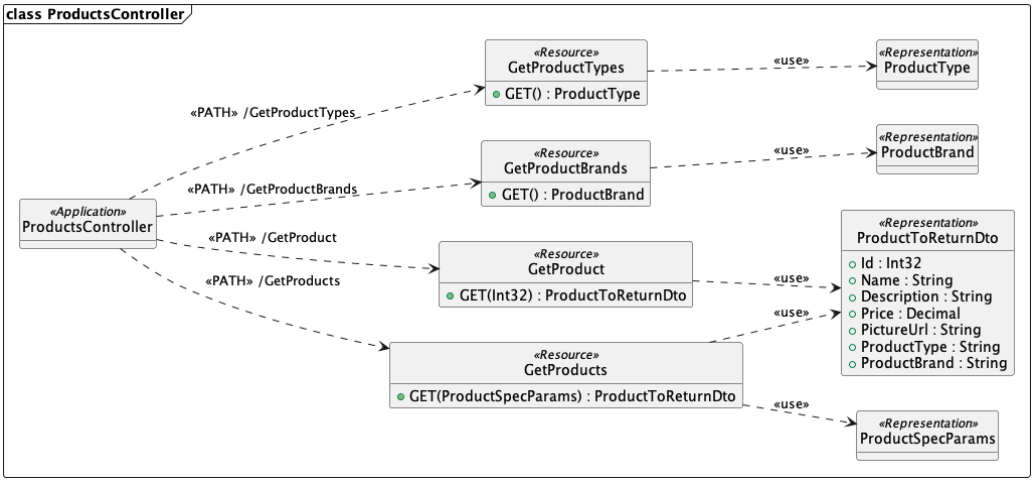


ภาพที่ 5.12 แผนภาพของ AccountController จากเครื่องมือ

- ProductsController โดยมี ProductsController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.13 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.14



ภาพที่ 5.13 แผนภาพของ ProductsController จากผู้ทดสอบ

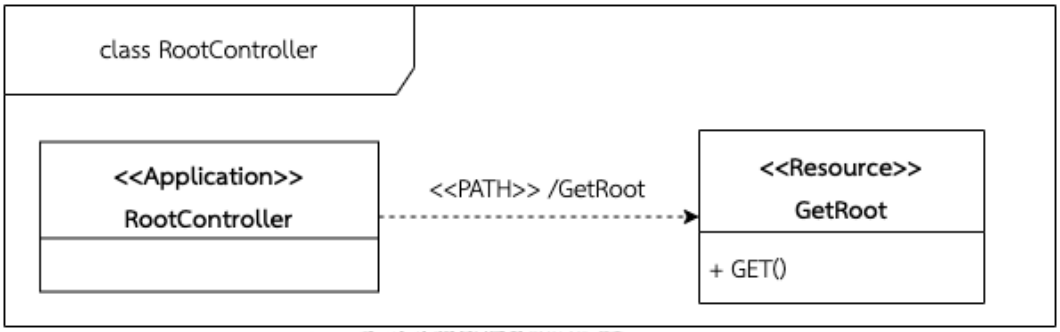


ภาพที่ 5.14 แผนภาพของ ProductsController จากเครื่องมือ

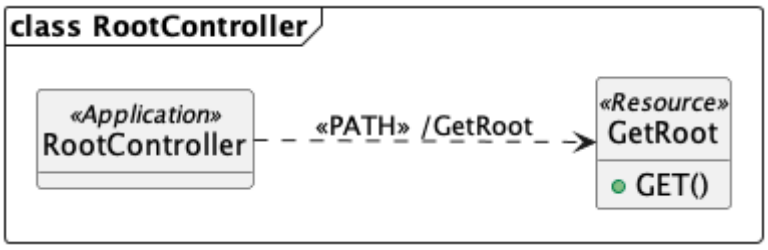
3. โครงการ Routine [16]

นำคอนโทรลเลอร์ 3 คอนโทรลเลอร์จากโครงการ Routine มาทำการทดสอบ โดยสนใจที่คอนโทรลเลอร์ต่าง ๆ ดังนี้

- RootController โดยมี RootController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.15 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.16

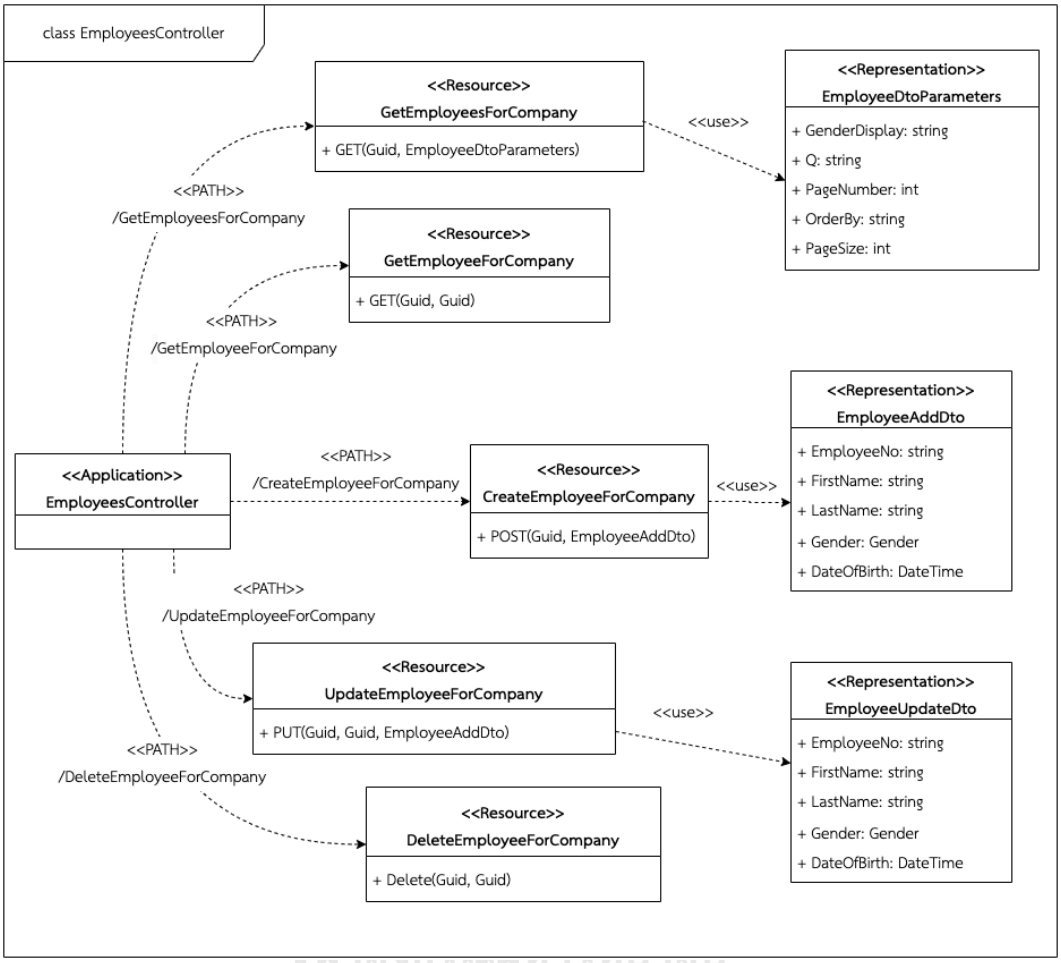


ภาพที่ 5.15 แผนภาพของ RootController จากผู้ทดสอบ

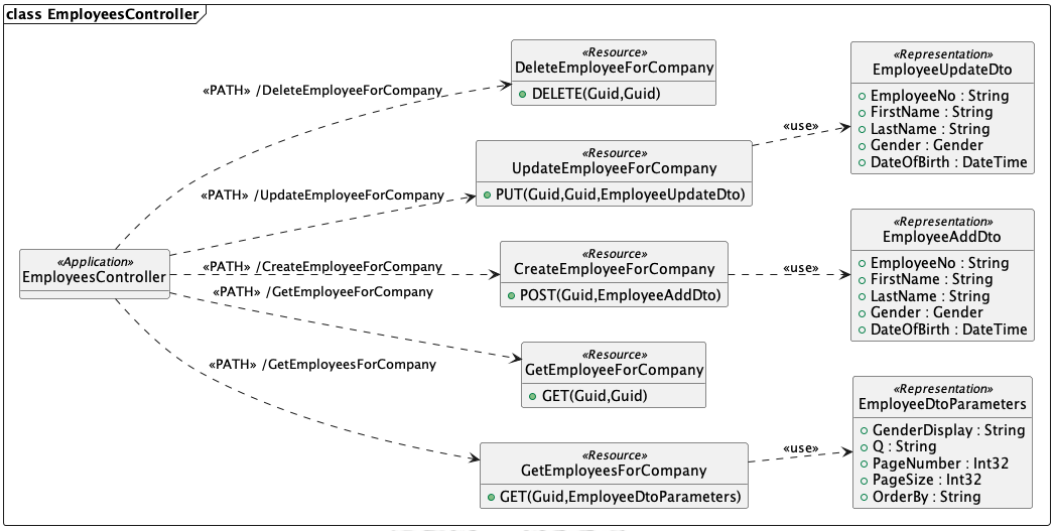


ภาพที่ 5.16 แผนภาพของ RootController จากเครื่องมือ

- EmployeesController โดยมี EmployeesController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.17 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.18

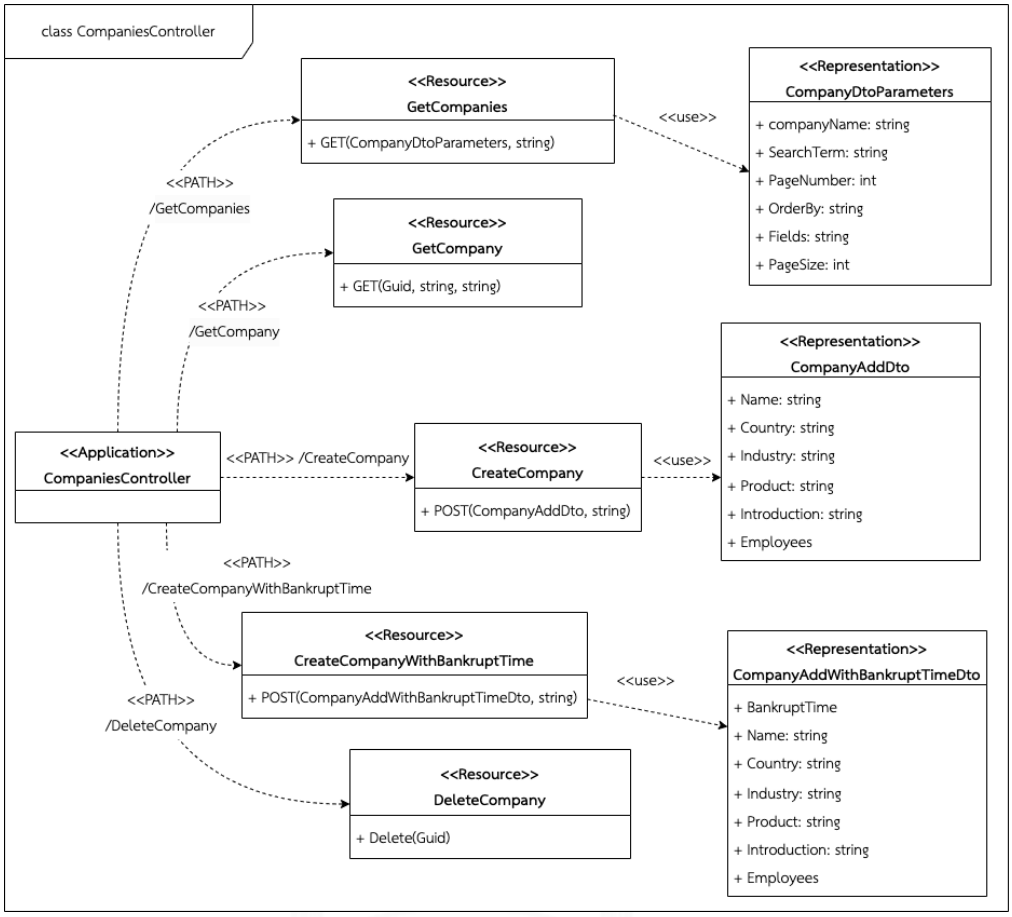


ภาพที่ 5.17 แผนภาพของ EmployeesController จากผู้ทดสอบ

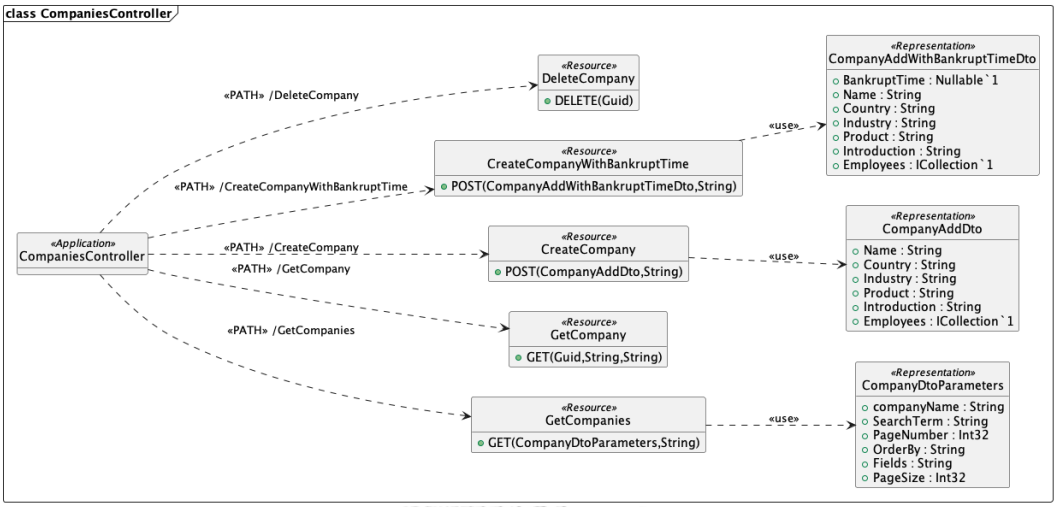


ภาพที่ 5.18 แผนภาพของ EmployeesController จากเครื่องมือ

- CompaniesController โดยมี CompaniesController ที่มาจากผู้ทดสอบเป็นผู้วาด ดังภาพที่ 5.19 และมาจากการสร้างของเครื่องมือ ดังภาพที่ 5.20



ภาพที่ 5.19 แผนภาพของ CompaniesController จากผู้ทดสอบ



ภาพที่ 5.20 แผนภาพของ CompaniesController จากเครื่องมือ

จากการตรวจสอบความถูกต้องและความครบถ้วนของการสร้างแผนภาพ ได้ทำการทดสอบโดยเปรียบเทียบแผนภาพระหว่างแผนภาพที่ผู้ทดสอบเป็นผู้วาดกับแผนภาพที่ได้จากเครื่องมือ ซึ่งได้ผลของการเปรียบเทียบที่มีความถูกต้องและครบถ้วนเป็น 100% ทั้งนี้ในการทดสอบจะไม่พิจารณารูปแบบการจัดวาง สี และลักษณะของอักษร

บทที่ 6
บทสรุปโครงงานมหาบัณฑิตและข้อเสนอแนะ

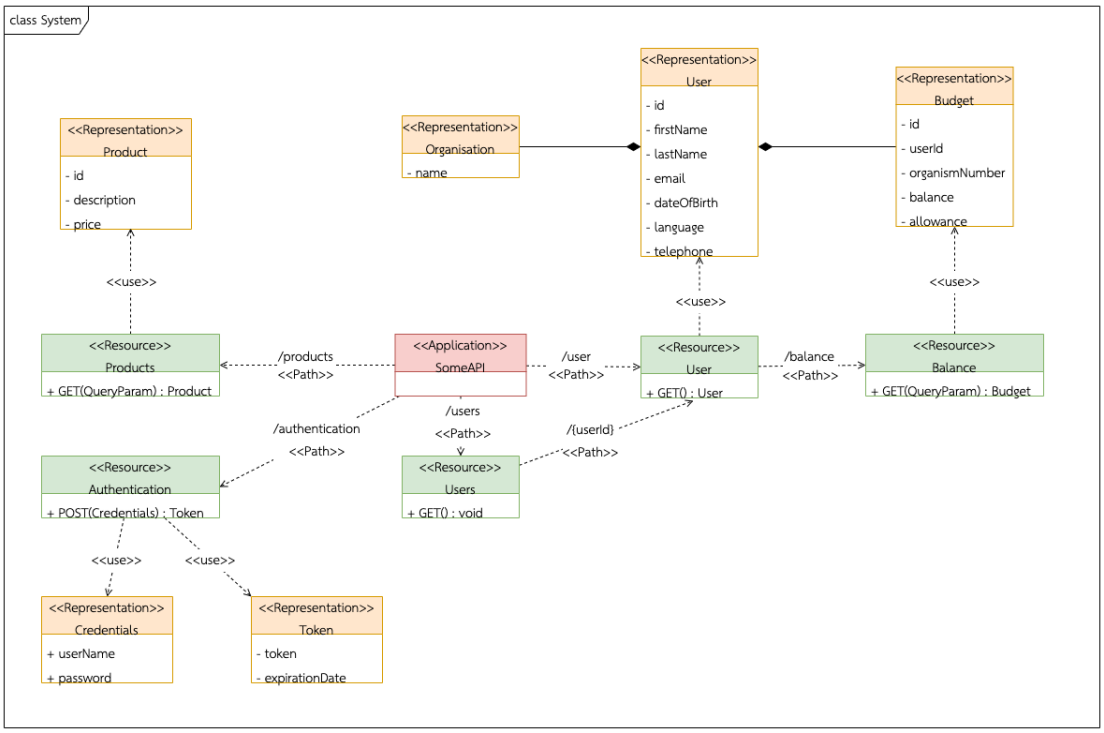
6.1 สรุปผลโครงงานมหาบัณฑิต

โครงงานมหาบัณฑิตนี้ ได้ทำการออกแบบและพัฒนาเครื่องมือสำหรับการสร้างแผนภาพจำลองความสัมพันธ์ของส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล เพื่อช่วยในการนำแผนภาพนี้ไปเป็นส่วนหนึ่งของเอกสารส่วนต่อประสานโปรแกรมประยุกต์ได้ เมื่อมีการอธิบายที่อยู่ในรูปแบบของแผนภาพยูเอ็มแอล จะช่วยให้ผู้อ่านสามารถเข้าใจเอกสารส่วนต่อประสานโปรแกรมประยุกต์ได้ง่ายยิ่งขึ้น เมื่อนำเครื่องมือที่พัฒนาขึ้น ไปประยุกต์ใช้กับโครงการต่าง ๆ และทำการทดสอบโดยการนำคอนโทรลเลอร์ในแต่ละโครงการ จำนวน 3 โครงการ โครงการละ 3 คอนโทรลเลอร์ ทำให้ได้ผลลัพธ์ของความถูกต้องและครบถ้วน เป็น 100%

6.2 ปัญหาและข้อจำกัดในการทำโครงงานมหาบัณฑิตนี้

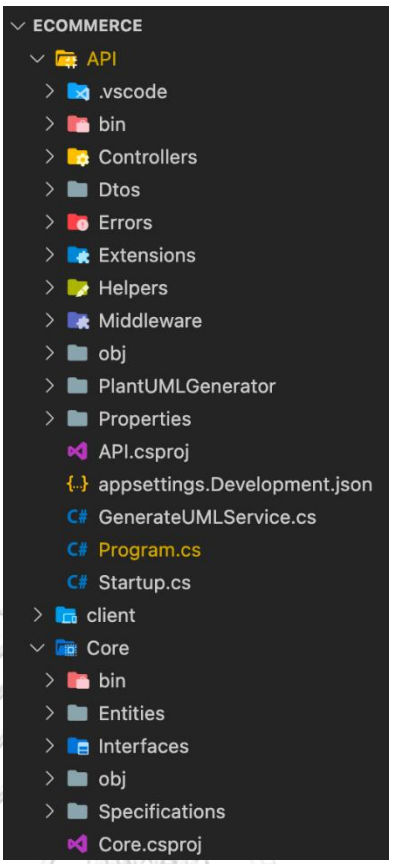
จากการออกแบบ และพัฒนาเครื่องมือในโครงงานมหาบัณฑิตนี้ ได้พบปัญหา และข้อจำกัดจำนวน 3 ข้อ ดังนี้

1. เครื่องมือที่พัฒนาขึ้นนี้ ไม่สามารถรองรับการสื่อสารข้อมูลระหว่างคลาส <<Resource>> กับ <<Resource>> ได้ ตัวอย่างเช่น ความสัมพันธ์ของคลาส <<Resource>> User กับ <<Resource>> Balance เครื่องมือจะสามารถสร้างแผนภาพโดยเกิดเป็น <<Application>> SomeAPI ไปยัง <<Resource>> User ได้เท่านั้น ดังภาพที่ 6.1



ภาพที่ 6.1 ตัวอย่างแผนภาพส่วนต่อประสานโปรแกรมประยุกต์เว็บเซอร์วิสแบบเรสต์ฟูล

2. เมื่อนำเครื่องมือที่พัฒนาขึ้นไปประยุกต์ใช้กับโครงการต่าง ๆ พบว่า มี 1 โครงการที่มีโครงสร้างที่อ้างอิงคุณลักษณะภายในจากแฟ้มข้อมูลภายนอก ดังภาพที่ 6.2 จากการนำเครื่องมือไปประยุกต์ใช้ในแฟ้มข้อมูล API เมื่อมีเมธอดที่รับค่าพารามิเตอร์ที่มีคุณลักษณะภายใน หรือเมธอดที่ส่งค่าพารามิเตอร์ออกมีคุณลักษณะภายใน โดยเป็นพารามิเตอร์ที่ประกาศไว้ในแฟ้มข้อมูล Core ที่อยู่ภายนอกแฟ้มข้อมูล API จะทำให้ไม่สามารถอ่านคุณลักษณะภายในได้ ยกตัวอย่างเมธอดที่ส่งค่าพารามิเตอร์ออกมีคุณลักษณะภายใน ดังภาพที่ 6.3 เมธอด GetProductType มีค่าพารามิเตอร์ที่ส่งออกเป็น ProductType เมื่อค้นหาคุณลักษณะภายในของ ProductType ดังภาพที่ 6.4 จะเห็นได้ว่าการประกาศ namespace อยู่ภายใต้แฟ้มข้อมูลของ Core ไม่ใช่แฟ้มข้อมูล API ที่นำเครื่องมือไปประยุกต์ใช้ ทำให้เครื่องมือหาคุณลักษณะภายในของ ProductType ไม่เจอ เป็นต้น



ภาพที่ 6.2 โครงสร้างของแฟ้มข้อมูล

```
[HttpGet("types")]
0 references
public async Task<ActionResult<IReadOnlyList<ProductType>>> GetProductTypes()
{
    return Ok(await _productTypeRepo.ListAllAsync());
}
```

ภาพที่ 6.3 ตัวอย่างเมธอดที่มีค่าที่ส่งออกเป็นพารามิเตอร์ที่มีคุณลักษณะภายใน

```
namespace Core.Entities
{
    8 references | You, 3 months ago | 1 author (You)
    public class ProductType : BaseEntity
    {
        1 reference
        public string Name { get; set; }
    }
}
```

ภาพที่ 6.4 ตัวอย่างคุณลักษะในอยู่ที่แฟ้มข้อมูล Core

3. หากพารามิเตอร์ที่นำเข้า หรือส่งออกมีคุณลักษณะภายในที่มีคุณลักษณะภายในอีกชั้นหนึ่ง จากการออกแบบเครื่องมือนี้ ยังไม่สามารถรองรับให้เกิดความสัมพันธ์ระหว่าง <<Representation>> กับ <<Representation>> ได้

6.3 ข้อเสนอแนะ

- จากข้อจำกัดในหัวข้อที่ 6.2 หากสามารถแก้ข้อจำกัดเหล่านี้ได้ เครื่องมือที่พัฒนาขึ้นจะสมบูรณ์แบบมากยิ่งขึ้น
- โครงงานมหาบัณฑิตนี้ ได้ทำการพัฒนาเครื่องมือโดยใช้ภาษาซีชาร์ปเท่านั้น หากต้องการนำเครื่องมือไปประยุกต์ใช้กับโครงการที่พัฒนาขึ้นโดยใช้ภาษาอื่น ๆ อาจพัฒนาเครื่องมือสำหรับภาษานั้น ๆ ได้ โดยใช้ตรรกะความคิดดังภาพที่ 4.3 ในบทที่ 4 เป็นแนวทางได้

บรรณานุกรม

1. Nybom, K., A. Ashraf, and I. Porres. *A systematic mapping study on API documentation generation approaches*. in *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 2018. IEEE.
2. Rathod, D.M., S.M. Parikh, and B. Buddhadev. *Structural and behavioral modeling of RESTful web service interface using UML*. in *2013 International conference on intelligent systems and signal processing (ISSP)*. 2013. IEEE.
3. Ed-Douibi, H., et al. *WAPIml: towards a modeling infrastructure for Web APIs*. in *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. 2019. IEEE.
4. Porres, I. and I. Rauf. *Modeling behavioral RESTful web service interfaces in UML*. in *Proceedings of the 2011 ACM Symposium on Applied Computing*. 2011.
5. Manigos., C.P. *E-CommerceApp*. 2021; Available from: https://github.com/xchanmolx/E-CommerceApp.
6. *PlantUML Language Reference Guide*. 2021; Available from: https://plantuml.com/guide.
7. Ed-Douibi, H., J.L. Cánovas Izquierdo, and J. Cabot. *OpenAPItoUML: a tool to generate UML models from OpenAPI definitions*. in *Web Engineering: 18th International Conference, ICWE 2018, Cáceres, Spain, June 5-8, 2018, Proceedings*. 2018. Springer.
8. Haldar, M., *RESTful API Designing guidelines—The best practices.* Hacker Noon, 2017.
9. *Create Spec for REST API in Enterprise Architect*. 2019; Available from: https://newbedev.com/create-spec-for-rest-api-in-enterprise-architect.
10. *REST API Architectural Constraints*. 2022; Available from: https://www.geeksforgeeks.org/rest-api-architectural-constraints/.
11. Patel, A. *Best Practices to Follow for REST API Development*. 2020; Available from: https://www.mindinventory.com/blog/best-practices-rest-api-development/.

12. Chansuwath, W. and T. Senivongse. *A model-driven development of web applications using AngularJS framework. 2016 IEEE*. in *ACIS 15th International Conference on Computer and Information Science (ICIS)*. 2016.
13. *Guide to UML Diagramming and Database Modeling*. 2022; Available from: https://www.microsoft.com/th-th/microsoft-365/business-insights-ideas/resources/guide-to-uml-diagramming-and-database-modeling.
14. *Plantuml - Generate UML Diagrams from a Text Description*. 2019; Available from: https://manpages.ubuntu.com/manpages/impish/man1/plantuml.1.html#name.
15. Matute, A. *DatingApp*. 2020; Available from: https://github.com/tonymatute/DatingApp.
16. Surbowl. *Routine*. 2020; Available from: https://github.com/Surbowl/ASP.NET-Core-RESTful-Note.

ประวัติผู้เขียน

ชื่อ-สกุล	วิภาดา กลึงเทศ
วัน เดือน ปี เกิด	22 กรกฎาคม 2541
สถานที่เกิด	โรงพยาบาลนพรัตนราชธานี
วุฒิการศึกษา	วิศวกรรมศาสตรบัณฑิต สาขาวิชา วิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ที่อยู่ปัจจุบัน	261/215 โมดิซสเตชั่นคอนโด ถนน พหลโยธิน แขวง อนุสาวรีย์ เขตบางเขน กรุงเทพมหานคร 10220