กระบวนการพัฒนาโปรแกรมตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ด้วยเทคนิคโลวโค้ดบนโอดู

นายโสภณวิชญ์ พิชิตเธียรธรรม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์เพื่ออุตสาหกรรม ไม่สังกัดภาควิชา/เทียบเท่า
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2563

Object Relational Mapping API Development Process Using Low Code Technique On Odoo

Mr. Sopanawit Pichidtienthum

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Science for Industry
Common Course
FACULTY OF SCIENCE
Chulalongkorn University
Academic Year 2020

หัวข้อวิทยานิพนธ์	กระบวนการพัฒนาโปรแกรมตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ด้วยเทคนิคโลวโค้ดบนโอดู
โดย	นายโสภณวิชญ์ พิชิตเธียรธรรม
สาขาวิชา	วิทยาศาสตร์เพื่ออุตสาหกรรม
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.ภควรรณ ปักษี
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะวิทยาศาสตร์
(ศาสตราจารย์ ดร.พลกฤษณ์ แสงวณิช)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร.ธิติ บวรรัตนารักษ์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร.ภควรรณ ปักษี)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(รองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.ฑิตยา หวานวารี)

..... กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร.กิตติพันธุ์ เตชะกิตติโรจน์)

โสภณวิชญ์ พิชิตเธียรธรรม : กระบวนการพัฒนาโปรแกรมตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ด้วยเทคนิคโลวโค้ดบนโอดู. (Object Relational Mapping API Development Process Using Low Code Technique On Odoo) อ.ที่ปรึกษาหลัก : ผศ. ดร.ภควรรณ ปักษี, อ.ที่ปรึกษาร่วม : รศ. ดร.นกุล คูหะโรจนานนท์

งานวิจัยนี้นำเสนอมอดูลเจนเนอเรเตอร์ ซึ่งเป็นเครื่องมือสำหรับการเขียนโปรแกรมตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์บนโอดู โอดูเป็นซอฟต์แวร์อีอาร์พีแบบเปิดเผยรหัสต้นฉบับที่รวบรวมมอดูลที่จำเป็นสำหรับการจัดการธุรกิจต่าง ๆ และผู้พัฒนาสามารถพัฒนามอดูลเพื่อขยายขีดความสามารถของโอดูได้ โดยในช่วงไม่กี่ปีที่ผ่านมาการพัฒนาซอฟต์แวร์บนโอดูมักต้องใช้เวลาในการเรียนรู้เพราะมีความซับซ้อนของเฟรมเวิร์ก จึงมีการนำแนวทางการพัฒนาซอฟต์แวร์แบบโลวโค้ด (การเขียนโค้ดที่น้อยกว่าปกติ) มาใช้ในการพัฒนามอดูลเจนเนอเรเตอร์ขึ้นเพื่อให้เป็นเครื่องมือสำหรับออกแบบและสร้างรหัสต้นฉบับสำหรับมอดูล ทำให้ผู้พัฒนาซอฟต์แวร์บนโอดูไม่ต้องกังวลเรื่องข้อผิดพลาดอันเนื่องมาจากความซับซ้อนของเฟรมเวิร์ก และให้ความสำคัญกับการเขียนโปรแกรมในด้านอื่น ๆ ได้มากขึ้น โดยเครื่องมือนี้ถูกพัฒนาด้วยภาษาไพธอนให้เป็นเว็บแอปพลิเคชันทำงานบนเว็บเบราว์เซอร์ และได้มีการทดสอบการใช้งานกับทั้งผู้พัฒนามอดูลบนโอดู ผู้ใช้งานโอดูที่มีทักษะการเขียนโปรแกรม ผู้ที่ไม่เคยมีประสบการณ์กับโอดูและอาสาสมัครภายนอกบริษัท ซึ่งได้ผลลัพธ์ว่า ผู้ทดสอบทั้งหมดสามารถพัฒนามอดูลโดยใช้มอดูลเจนเนอเรเตอร์ได้สำเร็จ เครื่องมือนี้จึงสามารถช่วยให้ผู้ที่ไม่มีประสบการณ์ในการพัฒนามอดูลบนโอดูสามารถพัฒนามอดูลขึ้นมาได้โดยใช้เวลาไม่นาน และเมื่อเปรียบเทียบเวลาที่ใช้ในการพัฒนามอดูลของกลุ่มผู้พัฒนามอดูลบนโอดู ระหว่างแบบปกติที่เขียนโค้ดด้วยตนเองกับการใช้มอดูลเจนเนอเรเตอร์ พบว่าการใช้เครื่องมือนี้สามารถลดเวลาการพัฒนามอดูลโดยเฉลี่ยได้ถึง 20% อีกทั้งมอดูลเจนเนอเรเตอร์ยังนำไปใช้ได้จริงในโครงการการพัฒนาซอฟต์แวร์ของบริษัท

สาขาวิชา	วิทยาศาสตร์เพื่ออุตสาหกรรม	ลายมือชื่อนิสิต .....
ปีการศึกษา	2563	ลายมือชื่อ อ.ที่ปรึกษาหลัก .....
		ลายมือชื่อ อ.ที่ปรึกษาร่วม .....

6270223023 : MAJOR SCIENCE FOR INDUSTRY

KEYWORD: Low-Code Development Platform, Code Generation, Odoo, Automation

Sopanawit Pichidtienthum : Object Relational Mapping API Development Process Using Low Code Technique On Odoo. Advisor: Asst. Prof. PAKAWAN PUGSEE, Ph.D. Co-advisor: Assoc. Prof. NAGUL COOHAROJANANONE, Ph.D.

In this research, a module generator which is an object-relational mapping (ORM) tool on Odoo was proposed. Odoo is an open-source ERP software that provides basic modules for various aspects of business management which also allows developers to create new modules to extend the capability of the Odoo system. However, due to the complexity of the framework, software development on Odoo usually requires a considerable amount of time to learn and specialize in. Therefore, low-code software development concept (the software implementation with less code writing) was used to create this module generator, a tool for designing modules and generating source codes of them. This lead to that software developers can develop modules with less concern about errors and more focus on the logical side of programming. This tool was developed using Python as the web application running in the web browser and was tested by module developers on Odoo, Odoo users with programming skills, non-Odoo users, and external volunteers. The results revealed that all testers are successfully able to develop a module using this tool, so it can help non-Odoo users to develop a module in a short period of time. The development time was also found reduced by 20% on average for module developers on Odoo, when comparing between coding manually and using this module generator. In addition, this tool has been applied to the software development project.

Field of Study: Science for Industry Student's Signature .....

Academic Year: 2020 Advisor's Signature .....

Co-advisor's Signature .....

กิตติกรรมประกาศ

งานวิจัยฉบับนี้ได้รับการสนับสนุนจากบริษัท แฟคเกอร์ จำกัด และโปรแกรมสนับสนุนการพัฒนาเทคโนโลยีและนวัตกรรมภายใต้สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ

ขอขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.ภควรรณ ปักษี อาจารย์ที่ปรึกษาวิทยานิพนธ์ และรองศาสตราจารย์ ดร.นกุล คูหะโรจนานนท์ อาจารย์ที่ปรึกษาร่วม ซึ่งสละเวลาให้คำปรึกษา ช่วยตรวจสอบแก้ไขข้อบกพร่องต่าง ๆ จนทำให้การวิจัยครั้งนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณกรรมการสอบวิทยานิพนธ์ รองศาสตราจารย์ ดร.ธิติ บวรรัตนารักษ์ ผู้ช่วยศาสตราจารย์ ดร.ฑิตยา หวานวารี และผู้ช่วยศาสตราจารย์ ดร.กิตติพันธุ์ เตชะกิตติโรจน์ ที่กรุณาสละเวลาให้คำแนะนำ ตรวจสอบและแก้ไขวิทยานิพนธ์ ซึ่งเป็นประโยชน์ในการทำวิทยานิพนธ์ฉบับนี้อย่างยิ่ง

สุดท้ายนี้ ข้าพเจ้าหวังเป็นอย่างยิ่งว่า เนื้อหาในวิทยานิพนธ์ฉบับนี้จะเป็นประโยชน์แก่ผู้อื่นไม่มากก็น้อย

โสภณวิชญ์ พิชิตเธียรธรรม

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ค

บทคัดย่อภาษาอังกฤษ ..... ง

กิตติกรรมประกาศ ..... จ

สารบัญ ..... ฉ

สารบัญตาราง ..... ฌ

สารบัญภาพ ..... ญ

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 วัตถุประสงค์ของงานวิจัย 2

1.3 ขอบเขตงานวิจัย 2

1.4 ขั้นตอนและวิธีดำเนินการวิจัย 3

1.5 ประโยชน์ที่ได้รับ 3

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 4

2.1 ตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ (Object Relational Mapping: ORM) บนโอดู 4

2.2 การพัฒนาโปรแกรมโดยใช้โลวโค้ดในอุตสาหกรรมปัจจุบัน 4

2.3 หลักการอธิบายความสัมพันธ์ระหว่างข้อมูลต้นทางและปลายทางด้วยซีดีเอ็ม 5

2.4 การพัฒนามอดูลสำหรับโอดู 6

บทที่ 3 วิธีการดำเนินงานวิจัย 9

3.1 การศึกษาและวิเคราะห์ปัญหาที่เกิดขึ้นจากการพัฒนามอดูลภายในบริษัท 9

3.2 สถาปัตยกรรมของมอดูลเจนเนอเรเตอร์ 10

3.3 ฐานข้อมูลของมอดูลเจนเนอเรเตอร์ 11

3.3.1 ตาราง generator_module 12
3.3.2 ตาราง generator_model 12
3.3.3 ตาราง generator_field 13
3.3.4 ตาราง generator_model_access 13
3.3.5 ตาราง generator_model_menu 13
3.4 การออกแบบการทำงานของมอดูลเจนเนอเรเตอร์ตามมุมมองการใช้งาน 13
3.4.1 การออกแบบมอดูลโดยใช้มอดูลเจนเนอเรเตอร์ 15
3.4.2 การสร้างรหัสต้นฉบับโดยใช้มอดูลเจนเนอเรเตอร์ 16
3.4.2.1 การสร้างรหัสต้นฉบับสำหรับไฟล์รูปแบบไพธอนและซีเอสวี 17
3.4.2.2 การสร้างรหัสต้นฉบับสำหรับไฟล์รูปแบบเอ็กซ์เอ็มแอล 19
3.5 เครื่องมือที่ใช้ในการออกแบบและพัฒนามอดูลเจนเนอเรเตอร์ 21
บทที่ 4 ผลการวิจัยและอภิปรายผล 22
4.1 ภาพรวมของมอดูลเจนเนอเรเตอร์ 22
4.2 การพัฒนามอดูลโดยใช้มอดูลเจนเนอเรเตอร์ 22
4.2.1 เริ่มต้นสร้างมอดูล 22
4.2.2 การสร้างโมเดล 24
4.2.3 การสร้างฟิลด์ 24
4.2.4 การสร้างสิทธิ์การเข้าถึงข้อมูลภายในโมเดล 26
4.2.5 การสร้างเมนู 26
4.2.6 การสร้างรหัสต้นฉบับ (Generate) 27
4.3 การทดสอบการใช้งานมอดูลเจนเนอเรเตอร์ 28
4.3.1 การทดสอบการใช้งานเบื้องต้น 29
4.3.2 การทดสอบการใช้งานโดยอาสาสมัครภายนอก 32
4.4 การนำมอดูลเจนเนอเรเตอร์มาใช้เพื่อการพัฒนาซอฟต์แวร์ที่ใช้งานจริง 34

บทที่ 5 สรุปผลการวิจัย 35
5.1 สรุปผลการวิจัย 35
5.2 แนวทางวิจัยในอนาคต 36
ภาคผนวก 37
ภาคผนวก ก พจนานุกรมข้อมูล (data dictionary) ของมอดูลเจนเนอเรเตอร์ 38
ภาคผนวก ข พจนานุกรมข้อมูล (data dictionary) ของระบบบันทึกข้อมูลผู้ได้รับความเดือดร้อนจากผลกระทบของโรคโควิด 19 42
ภาคผนวก ค แบบประเมินตนเองของอาสาสมัครผู้ทดสอบใช้งานมอดูลเจนเนอเรเตอร์ 45
ภาคผนวก ง ความคิดเห็นเกี่ยวกับการทดสอบของผู้ทดสอบใช้งานมอดูลเจนเนอเรเตอร์ 46
ภาคผนวก จ ผลงานการตีพิมพ์ในงานประชุมวิชาการ 48
บรรณานุกรม 53
ประวัติผู้เขียน 54

สารบัญตาราง

หน้า

ตารางที่ 1 ปัญหาที่เกิดขึ้นระหว่างการพัฒนามอดูลด้วยวิธีปกติ 9

ตารางที่ 2 กลุ่มผู้ทดสอบและจำนวนผู้ทดสอบในการทดสอบใช้งานมอดูลเจนเนอเรเตอร์ 28

ตารางที่ 3 จำนวนข้อมูลของแต่ละโมเดลที่ใช้ในการทดสอบ 29

สารบัญภาพ

หน้า

ภาพที่ 1 ตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ (ORM) 4

ภาพที่ 2 ตัวอย่างการทำงานของโลวโค้ด 5

ภาพที่ 3 การใช้ซีดีเอ็มเพื่ออธิบายความสัมพันธ์ระหว่างข้อมูลต้นทางและปลายทาง 6

ภาพที่ 4 การระบุคุณสมบัติของข้อมูลในไฟล์โมเดล 7

ภาพที่ 5 การกำหนดมุมมองของไฟล์วิวตามที่โอดูกำหนด 7

ภาพที่ 6 การกำหนดสิทธิ์การเข้าถึงข้อมูลสำหรับโมเดล 7

ภาพที่ 7 การกำหนดข้อมูลและระบุคุณสมบัติของมอดูลในไฟล์แมนนิเฟส 8

ภาพที่ 8 มอดูลที่ถูกติดตั้งเข้าสู่โอดู 8

ภาพที่ 9 ภาพรวมสถาปัตยกรรมระบบของโอดู 10

ภาพที่ 10 แผนภาพแสดงสถาปัตยกรรมซอฟต์แวร์ของมอดูลเจนเนอเรเตอร์ 11

ภาพที่ 11 แผนภาพแสดงความสัมพันธ์ระหว่างเอนทิตี้ 12

ภาพที่ 12 ส่วนประกอบของมอดูล 14

ภาพที่ 13 ผังงานวิธีการออกแบบมอดูลด้วยมอดูลเจนเนอเรเตอร์ 16

ภาพที่ 14 โครงสร้างไดเร็กทอรีและไฟล์ต่าง ๆ ของมอดูล 17

ภาพที่ 15 การแบ่งกลุ่มส่วนประกอบของไฟล์ตามลักษณะของเนื้อหา 18

ภาพที่ 16 ผังงานวิธีการสร้างไฟล์โดยใช้โค้ดเทมเพลต 18

ภาพที่ 17 กระบวนการแปลงข้อมูลเอ็กซ์เอ็มแอลโดยใช้เอ็กซ์เอสแอลที 19

ภาพที่ 18 ไฟล์รูปแบบเอ็กซ์เอสแอลสำหรับใช้แปลงข้อมูลเอ็กซ์เอ็มแอล 20

ภาพที่ 19 ข้อมูลตั้งต้นที่ถูกจัดทำในรูปแบบเอ็กซ์เอ็มแอล 21

ภาพที่ 20 หน้าจอแบบฟอร์มสำหรับสร้างมอดูล 23

ภาพที่ 21 หน้าจอแบบฟอร์มสำหรับสร้างโมเดล 24

ภาพที่ 22 หน้าจอแบบฟอร์มสำหรับสร้างฟิลด์ 25
ภาพที่ 23 หน้าจอแบบฟอร์มสำหรับสร้างสิทธิ์การเข้าถึงข้อมูล 26
ภาพที่ 24 หน้าจอแบบฟอร์มสำหรับสร้างเมนู 27
ภาพที่ 25 หน้าจอเพื่อยืนยันการสร้างโค้ดมอดูล 27
ภาพที่ 26 ไดเร็กทอรีและไฟล์ที่ถูกสร้างขึ้นโดยมอดูลเจนเนอเรเตอร์ 28
ภาพที่ 27 กราฟแสดงเวลารวมเฉลี่ยในการพัฒนามอดูลของผู้ทดสอบแต่ละกลุ่ม 30
ภาพที่ 28 เวลาเฉลี่ยที่ใช้สร้างแยกตามโมเดลและกลุ่มผู้ทดสอบ 30
ภาพที่ 29 เวลารวมเฉลี่ยของการพัฒนามอดูลด้วยวิธีปกติและการใช้มอดูลเจนเนอเรเตอร์ 31
ภาพที่ 30 กราฟแสดงเวลาในพัฒนามอดูลรายบุคคล 33
ภาพที่ 31 เวลาเฉลี่ยในการสร้างรายโมเดลของอาสาสมัครภายนอก 34

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการใช้ซอฟต์แวร์เพื่อนำมาแก้ปัญหาและเสริมประสิทธิภาพด้านการบริหารงานในองค์กรเป็นที่นิยมในประเทศไทย โดยเฉพาะอุตสาหกรรม บริษัท ห้างร้านในระดับวิสาหกิจขนาดกลางและขนาดย่อม ซึ่งทยอยปรับเปลี่ยนเพื่อใช้เทคโนโลยีและซอฟต์แวร์เพื่อบริหารจัดการมากขึ้น สังเกตได้จากซอฟต์แวร์จากกลุ่มธุรกิจที่เกิดขึ้นใหม่ ที่สามารถเลือกสรรนำเข้ามาใช้งาน หรือซอฟต์แวร์ชนิดบริการบนระบบคลาวด์ (Software as a Service) จากต่างประเทศที่มีการขยายพื้นที่บริการเข้ามาในประเทศและภูมิภาคเอเชียเช่นกลุ่มซีแอลเอ็มวี (CLMV) ได้แก่ กัมพูชา ลาว พม่า เวียดนาม มากขึ้น

ซอฟต์แวร์จัดการและวางแผนการใช้ทรัพยากรขององค์กร (Enterprise Resource Planning: ERP) สำหรับการบวนการทางธุรกิจ คือซอฟต์แวร์ที่องค์กรต่าง ๆ พิจารณาที่จะมีเป็นอันดับต้น ๆ ซึ่งความสามารถของซอฟต์แวร์สำหรับจัดการทางธุรกิจนั้นประกอบไปด้วยมอดูลที่ช่วยจัดระเบียบและควบคุมคุณภาพในองค์กรเช่น การขาย การจัดการความสัมพันธ์กับลูกค้า การบัญชี ทรัพยากรบุคคล การจัดการคลังสินค้า อีกทั้งยังสามารถช่วยในการบริหารความเสี่ยงภายในองค์กร [1] และยังสามารถพัฒนามอดูลใหม่ ๆ ที่สามารถตอบโจทย์การทำงานขององค์กรได้

โอดู (Odoo) เป็นหนึ่งในซอฟต์แวร์ที่ได้รับความนิยมอย่างมากในองค์กรต่าง ๆ เนื่องจากโอดูคือซอฟต์แวร์อีอาร์พีแบบเปิดเผยรหัสต้นฉบับ (Open Source ERP) ที่เมื่อมีการนำโครงสร้างและรายละเอียดการทำงาน รวมถึงมอดูลต่าง ๆ มาเปรียบเทียบกับซอฟต์แวร์อีอาร์พีที่มีชื่อเสียงอื่น ๆ โอดูก็สามารถเป็นที่ยอมรับเมื่อเทียบกับคู่แข่ง โดยเหตุผลมาจากโครงสร้างของระบบที่มีมาตรฐาน รวมถึงมีมอดูลสำหรับการจัดการทางธุรกิจมาให้อย่างครบถ้วน [1] อีกทั้งยังเปิดให้สามารถพัฒนามอดูลอื่น ๆ ที่องค์กรต้องการนอกเหนือไปจากที่ระบบมีมาให้ หรือแก้ไขมอดูลที่มีมาให้ในตัวซอฟต์แวร์ได้อย่างอิสระ แต่อย่างไรก็ตามทางองค์กรจำเป็นที่จะต้องพัฒนาส่วนประกอบต่าง ๆ ภายในระบบให้เข้ากับองค์กรด้วยตนเอง จึงเป็นจุดอ่อนหนึ่งของโอดู

กรณีศึกษาซึ่งเป็นบริษัทที่เขียนโปรแกรมพัฒนาระบบด้วยโอดู เพื่อตอบโจทย์ในเชิงการแก้ปัญหาและจัดระเบียบงานในกลุ่มลูกค้าองค์กร จำเป็นต้องเขียนโปรแกรมในการสร้างโครงสร้างข้อมูลของระบบ เรียกว่าตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ (Object-Relational Mapping: ORM) ซึ่งทำหน้าที่กำหนดโครงสร้างข้อมูลในฐานข้อมูลของระบบ และเป็นส่วนต่อประสานโปรแกรม

(Application Programming Interface: API) ภายในโอดู ซึ่งการเขียนโปรแกรมนี้จำเป็นต้องใช้ผู้พัฒนาที่มีความเข้าใจในรูปแบบของการเขียนโปรแกรมดังกล่าว ทำให้ยากที่จะสร้างขึ้นมาให้สามารถใช้งานได้ โดยผู้ที่ยังไม่มีความเข้าใจหรือความชำนาญมากพอ และทำให้สูญเสียเวลาเพิ่มมากขึ้นในกระบวนการสำหรับการพัฒนาซอฟต์แวร์

หากนักพัฒนาสามารถลดเวลาในการเขียนโปรแกรมได้ หมายความว่าสามารถนำเวลาไปพัฒนาส่วนที่สำคัญอื่น ๆ หรือแม้กระทั่งสามารถส่งมอบงานได้รวดเร็วมากยิ่งขึ้น การสร้างรหัสต้นฉบับแบบอัตโนมัติเป็นทางเลือกหนึ่งที่นิยมในอุตสาหกรรมปัจจุบันเนื่องจากสามารถเพิ่มประสิทธิภาพในการทำงานและการผลิตชิ้นงานให้กับลูกค้าได้เป็นอย่างดี [2] ทั้งนี้โปรแกรมสำหรับสร้างรหัสต้นฉบับแบบอัตโนมัติในปัจจุบันถูกสร้างขึ้นมาจำนวนมาก ซึ่งส่วนใหญ่นั้นเป็นการสร้างรหัสต้นฉบับที่รองรับเฉพาะโครงสร้างที่พัฒนาโดยผู้ผลิตนั้น ๆ เท่านั้น อีกทั้งยังมีข้อจำกัดที่ผู้พัฒนาอื่น ๆ ไม่สามารถเขียนโปรแกรมเพิ่มได้

ในการวิจัยนี้จะทำการสร้างเครื่องมือ ซึ่งทำหน้าที่สร้างมอดูลใหม่พร้อมกับรหัสต้นฉบับสำหรับตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ รวมถึงรหัสต้นฉบับอื่น ๆ ที่จำเป็นสำหรับมอดูล เช่นการแสดงผลและสิทธิการเข้าถึงข้อมูล ทำให้ผู้พัฒนาสามารถพัฒนามอดูลได้ด้วยความรวดเร็วและไม่ต้องกังวลถึงความซับซ้อนของการสร้างมอดูลด้วยวิธีปกติ รวมถึงสามารถช่วยให้ผู้ที่ไม่มีความชำนาญในการพัฒนามอดูลด้วยวิธีปกติสามารถสร้างมอดูลที่ไม่ซับซ้อนออกมาได้ด้วยตนเอง

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อพัฒนาเครื่องมือสำหรับการเขียนโปรแกรมตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์บนระบบโอดู

1.3 ขอบเขตงานวิจัย

1. เครื่องมือสามารถทำการติดตั้งและใช้งานได้กับโอดู (Odoo) เวอร์ชัน 12 ถึง 14
2. รองรับการออกแบบและสร้างรหัสต้นฉบับ (source code) สำหรับมอดูลโดยอัตโนมัติในส่วนของตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์
3. รองรับการออกแบบมอดูลได้แก่ โมเดล (model) เมนู (menu) และสิทธิ์การเข้าถึงโมเดล (security) ผ่านส่วนต่อประสานหน้าจอหรือส่วนต่อประสานผู้ใช้กราฟิก (Graphic User Interface: GUI) ได้

1.4 ขั้นตอนและวิธีดำเนินการวิจัย

1. ศึกษาทฤษฎีและกระบวนการทำงานของตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์บนโอดู
2. เก็บข้อมูลความต้องการและปัญหาที่เกิดขึ้นในกระบวนการพัฒนาตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์
3. ออกแบบและพัฒนาเครื่องมือสำหรับการเขียนโปรแกรมฯ
4. ทดสอบและปรับปรุงการทำงานของเครื่องมือสำหรับการเขียนโปรแกรมฯ
5. สรุปและเขียนรายงาน

1.5 ประโยชน์ที่ได้รับ

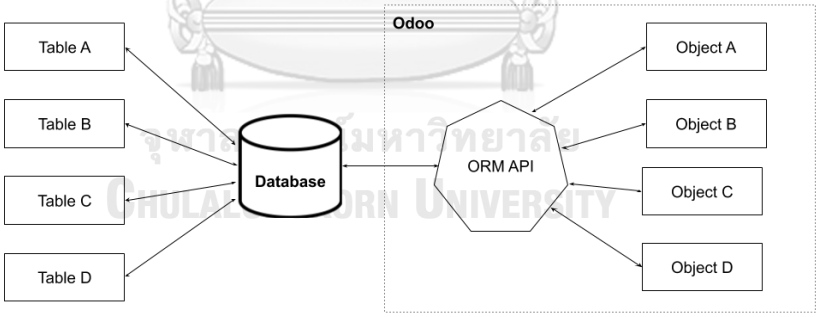
1. ปรับปรุงประสิทธิภาพกระบวนการพัฒนาซอฟต์แวร์
2. ลดเวลาและลดการเขียนโปรแกรมในการพัฒนาซอฟต์แวร์
3. เพื่อสามารถให้ผู้ใช้งานที่ไม่ใช่นักพัฒนาซอฟท์แวร์สามารถมีส่วนร่วมในการพัฒนาซอฟต์แวร์ได้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

เครื่องมือที่งานวิจัยนี้นำเสนอ มีทฤษฎีที่เกี่ยวข้องกับการออกแบบและพัฒนาได้แก่ ตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ การพัฒนาโปรแกรมโดยใช้โลวโค้ดในอุตสาหกรรมปัจจุบัน หลักการอธิบายความสัมพันธ์ระหว่างข้อมูลต้นทางและปลายทางด้วยซีดีเอ็ม โมเดล-วิว-คอนโทรลเลอร์ ภาษาสำหรับการแปลงเอกสารเอ็กซ์เอ็มแอล และการสร้างรหัสต้นฉบับด้วยการใช้โค้ดเทมเพลต

2.1 ตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ (Object Relational Mapping: ORM) บนโอดู

การเขียนโปรแกรมบนระบบที่มีโครงสร้างขนาดใหญ่จะมีการเชื่อมต่อกับฐานข้อมูลจำนวนมาก จึงต้องมีตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ซึ่งคือ เทคนิคการเขียนโปรแกรมเพื่อจัดการเกี่ยวกับการแลกเปลี่ยนข้อมูลระหว่างฐานข้อมูลกับโปรแกรมเชิงวัตถุ และแสดงลักษณะความสัมพันธ์ รวมถึงเอกลักษณ์ของข้อมูลบนโปรแกรมเชิงวัตถุให้เหมือนกับฐานข้อมูล [3] สามารถอธิบายความสัมพันธ์ได้ตามภาพที่ 1

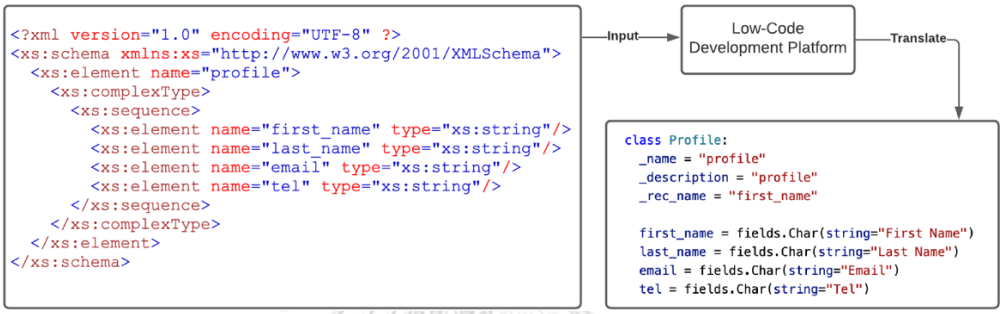


ภาพที่ 1 ตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ (ORM)

2.2 การพัฒนาโปรแกรมโดยใช้โลวโค้ดในอุตสาหกรรมปัจจุบัน

ในการพัฒนาโปรแกรม ผู้พัฒนาจำเป็นที่จะต้องเขียนรหัสต้นฉบับด้วยภาษาโปรแกรมทางคอมพิวเตอร์ ซึ่งมีอยู่หลากหลายภาษา ผู้พัฒนาจึงจะต้องเรียนรู้หลักโครงสร้างภาษา และวิธีการใช้คำสั่งฟังก์ชันต่าง ๆ ของภาษานั้น ทำให้ผู้ที่จะพัฒนาต้องใช้เวลาในการเรียนรู้ก่อนที่จะพัฒนาโปรแกรมได้ โดยใช้เวลามากน้อยขึ้นอยู่กับความซับซ้อนของภาษา จากจุดนี้ทำให้มีการพัฒนา

เครื่องมือสำหรับเขียนโปรแกรมที่ใช้เทคนิคโลวโค้ดขึ้นมาเพื่อช่วยให้นักพัฒนาหรือบุคคลอื่น ๆ ที่ไม่ใช่นักพัฒนาสามารถเขียนโปรแกรมได้ [4] โดยอาจใช้เวลาในการเรียนรู้การใช้งานเครื่องมือเพียงไม่กี่วันก็สามารถใช้งานได้ พื้นฐานของเครื่องมือนั้น จะใช้หน้าจอ ประกอบกับรูปภาพ หรือภาษาบางชนิดที่สามารถเข้าใจได้ง่าย [5] หรือสามารถสร้างโดยใช้โปรแกรมอื่น ๆ ที่มีในท้องตลาด นำมาประกอบกันเพื่อแทนชุดของคำสั่งของภาษาโปรแกรมนั้น ๆ ทำให้ผู้ใช้งานจัดเรียงคำสั่งต่าง ๆ โดยที่ไม่ต้องกังวลกับโครงสร้างหรือการใช้คำสั่งฟังก์ชันของภาษาที่ต้องการตามภาพที่ 2

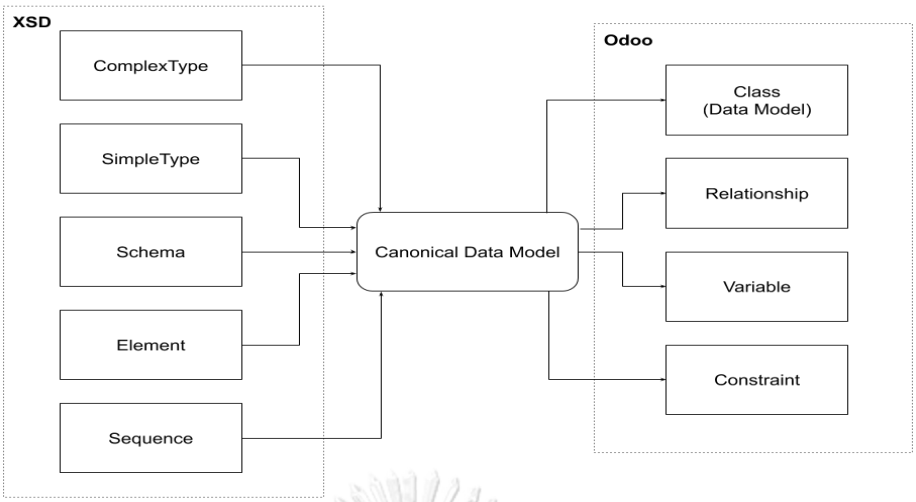


ภาพที่ 2 ตัวอย่างการทำงานของโลวโค้ด

จากภาพที่ 2 จะเห็นว่าข้อมูลต้นทางคือข้อมูลชนิดเอ็กซ์เอ็มแอล เป็นภาษาที่ใช้สำหรับการอธิบายโครงสร้างข้อมูลที่สามารถสร้างขึ้นมาได้ง่ายจากเครื่องมือในท้องตลาดทั่วไป เมื่อนำข้อมูลดังกล่าวมาใช้กับซอฟต์แวร์สำหรับการพัฒนาแบบโลวโค้ด จะสามารถนำข้อมูลดังกล่าวมาสร้างเป็นซอฟต์แวร์ที่ต้องการได้ทันที อย่างไรก็ตามการพัฒนาซอฟต์แวร์แบบโลวโค้ดอาจไม่ใช่ทางเลือกที่เหมาะสมในทุกสถานการณ์เนื่องจากข้อจำกัดที่เกิดจากการใช้คำสั่งให้รองรับการตามที่เครื่องมือกำหนดเท่านั้นยกตัวอย่างเช่นโอดูสตูดิโอ (Odoo Studio) [6] ที่รองรับการใช้งานแบบโลวโค้ดแต่ไม่สามารถนำผลลัพธ์ของโปรแกรมมาพัฒนาต่อได้

2.3 หลักการอธิบายความสัมพันธ์ระหว่างข้อมูลต้นทางและปลายทางด้วยซีดีเอ็ม

ในการวิจัยนี้ ผู้วิจัยนำเสนอการพัฒนาการเขียนโปรแกรมด้วยเทคนิคโลวโค้ด ซึ่งจำเป็นต้องมีส่วนที่ใช้จัดการหรืออธิบายความสัมพันธ์ของข้อมูลต้นทางและปลายทางเข้าด้วยกัน ซึ่งอาจไม่ได้ทำหน้าที่เฉพาะการอธิบายความสัมพันธ์ แต่ยังเป็นส่วนที่ทำหน้าที่เปลี่ยนข้อมูลต้นทางเพื่อให้ได้ข้อมูลปลายทาง [7] สำหรับซีดีเอ็ม (Canonical Data Model: CDM) นั้นคือรูปแบบการอธิบายชนิดหนึ่งเพื่อใช้อธิบายความสัมพันธ์และส่วนประกอบต่าง ๆ ระหว่างข้อมูลต้นทางและปลายทางแสดงดังภาพที่ 3 ทั้งนี้รูปแบบของการใช้ซีดีเอ็มสามารถใช้ได้ในหลากหลายรูปแบบเช่น เอ็กซ์เอสดี (XML Schema Definition: XSD)



ภาพที่ 3 การใช้ซีดีเอ็มเพื่ออธิบายความสัมพันธ์ระหว่างข้อมูลต้นทางและปลายทาง

2.4 การพัฒนามอดูลสำหรับโอดู

มอดูลคือชุดของข้อมูลและ/หรือโปรแกรมสำหรับประมวลผลที่เขียนในรูปแบบโมเดล-วิว-คอนโทรลเลอร์ (Model-View-Controller: MVC) [8] โดยขั้นตอนปกติของการพัฒนามอดูลประกอบด้วย 1) สร้างไดเร็กทอรี (directory) 2) สร้างตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ 3) สร้างมุมมองข้อมูล 4) สร้างสิทธิ์การเข้าถึงข้อมูล 5) สร้างไฟล์สำหรับข้อมูลพื้นฐานของมอดูล (manifest) 6) ติดตั้งมอดูลเข้าสู่โอดู ซึ่งขั้นตอนทั้งหมดจะต้องเขียนรหัสต้นฉบับขึ้นมาด้วยตนเอง โดยสำหรับขั้นตอนมีรายละเอียดวิธีการ ดังนี้

1) ดำเนินการเรียกคำสั่ง scaffold โดยใช้คอมมานด์ไลน์ (command line) python odoo-bin scaffold book_library เพื่อให้โอดูทำการสร้างไดเร็กทอรีสำหรับมอดูลที่จะพัฒนา รวมถึงสร้างตัวอย่างไฟล์ต่าง ๆ ในไดเร็กทอรีดังกล่าว
2) สร้างโมเดลของข้อมูลโดยการประกาศชื่อโมเดลที่จะใช้งานและประกาศตัวแปรสำหรับการเก็บข้อมูลพร้อมกับระบุคุณสมบัติของข้อมูลที่ต้องการใช้งานดังภาพที่ 4 ซึ่งโดยปกติแล้วตรรกะสำหรับการทำงานของโปรแกรมจะถูกเขียนอยู่ในไฟล์นี้เช่นกัน

```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api


class book_library(models.Model):
    _name = 'book_library'
    _description = 'book_library'

    name = fields.Char(string="Book name")
```

ภาพที่ 4 การระบุคุณสมบัติของข้อมูลในไฟล์โมเดล

3) สร้างวิวหรือการแสดงผลของโมเดลโดยการสร้างไฟล์เอ็กซ์เอ็มแอล (XML) โดยอ้างอิงโมเดลตามชื่อที่ประกาศไว้ในข้อที่ 2) และกำหนดรูปแบบการแสดงผลของการแสดงรายการข้อมูล การแสดงผลแบบฟอร์ม รูปแบบการแสดงผลของหน้าต่าง และเมนูสำหรับการเข้าถึงแบบฟอร์ม โดยลักษณะการประกาศรูปแบบการแสดงผลนั้นจะต้องอ้างอิงรูปแบบตามที่โอดูกำหนดดังภาพที่ 5 และใส่ชื่อข้อมูลที่ต้องการให้ปรากฏบนหน้าจอแสดงผลลงในรูปแบบที่กำหนด

```xml
<odoo>
  <data>
    <!-- explicit list view definition -->
    <record model="ir.ui.view" id="book_library.list">
      <field name="name">book_library list</field>
      <field name="model">book_library</field>
      <field name="arch" type="xml">
        <tree>
          <field name="name" />
        </tree>
      </field>
    </record>

    <!-- actions opening views on models -->
    <record model="ir.actions.act_window" id="book_library_action_window">
      <field name="name">book_library window</field>
      <field name="res_model">book_library</field>
      <field name="view_mode">tree,form</field>
    </record>

    <menuitem name="Library" id="book_library_menu_root" />
    <menuitem name="Book" id="book_library_menu_book" parent="book_library_menu_root"
              action="book_library_action_window" />
  </data>
</odoo>
```

ภาพที่ 5 การกำหนดมุมมองของไฟล์วิวตามที่โอดูกำหนด

4) กำหนดสิทธิ์และความปลอดภัยสำหรับการเข้าถึงข้อมูล โดยการสร้างไฟล์ซีเอสวี (Comma-Separated Values: CSV) และระบุสิทธิ์การใช้งานข้อมูลดังแสดงตามภาพที่ 6

```csv
id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
access_book_library,book_library,model_book_library,base.group_user,1,1,1,1
```

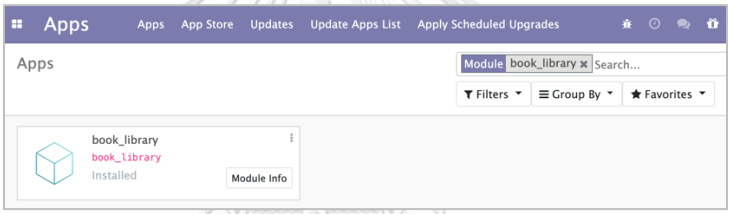
ภาพที่ 6 การกำหนดสิทธิ์การเข้าถึงข้อมูลสำหรับโมเดล

5) เพิ่มข้อมูลและระบุคุณลักษณะของมอดูลเพื่อใช้สำหรับการติดตั้งมอดูลในไฟล์แมนนิเฟส (manifest) ดังภาพที่ 7

```
# -*- coding: utf-8 -*-
{
    'name': "book_library",

    # Categories can be used to filter modules in modules listing
    # Check https://github.com/odoo/odoo/blob/13.0/odoo/addons/base/data/ir_module_category_data.xml
    # for the full list
    'category': 'Uncategorized',
    'version': '0.1',

    # any module necessary for this one to work correctly
    'depends': ['base'],

    # always loaded
    'data': [
        'security/ir.model.access.csv',
        'views/views.xml',
    ],
```

ภาพที่ 7 การกำหนดข้อมูลและระบุคุณสมบัติของมอดูลในไฟล์แมนนิเฟส

6) ติดตั้งมอดูลเข้าสู่โอดูดังแสดงในภาพที่ 8



ภาพที่ 8 มอดูลที่ถูกติดตั้งเข้าสู่โอดู

บทที่ 3
วิธีการดำเนินงานวิจัย

งานวิจัยนี้เป็นการพัฒนาเครื่องมือสำหรับการเขียนโปรแกรมตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์บนระบบโอดู มีชื่อเรียกว่า “มอดูลเจนเนอเรเตอร์ (Module Generator)” มีหน้าที่การทำงานหลักคือการสร้างรหัสต้นฉบับของตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ ซึ่งเป็นส่วนประกอบส่วนหนึ่งของมอดูล มีวิธีการศึกษาและดำเนินการวิจัย ดังนี้

3.1 การศึกษาและวิเคราะห์ปัญหาที่เกิดขึ้นจากการพัฒนามอดูลภายในบริษัท

กระบวนการพัฒนามอดูลเจนเนอเรเตอร์นั้นมีขึ้นเพื่อแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นจากการพัฒนามอดูล ในงานวิจัยนี้ได้มีการสังเกตการณ์พนักงานภายในบริษัท แฟคเกอร์ จำกัด ที่ใช้โอดูในการพัฒนาซอฟต์แวร์ที่หลากหลายเช่น เป็นระบบหลังบ้าน (back office) ของโมบายแอปพลิเคชัน (mobile application) หรือใช้เป็นระบบการจัดการข้อมูลแบบบูรณาการ (Data Integration Management System: DIMS) ประกอบกับประสบการณ์ในการเป็นผู้พัฒนาโปรแกรมที่ใช้งานโอดูในการพัฒนาซอฟต์แวร์ในบริษัทเป็นระยะเวลา 1 ปี พบว่าการพัฒนามอดูลด้วยการเขียนรหัสต้นฉบับด้วยตนเองตามขั้นตอนที่อธิบายในบทที่ 2 นั้น มักจะเกิดปัญหาที่มีสาเหตุจากโครงสร้างส่วนประกอบต่าง ๆ สามารถจำแนกปัญหาที่สำคัญได้ดังตารางที่ 1

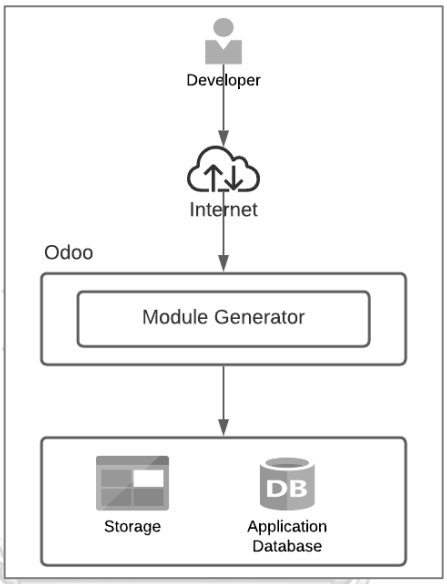
ตารางที่ 1 ปัญหาที่เกิดขึ้นระหว่างการพัฒนามอดูลด้วยวิธีปกติ

ลำดับที่	ปัญหาที่เกิดขึ้น	ความถี่ของปัญหา
1	พบข้อผิดพลาดในไฟล์โมเดลเนื่องจากตั้งค่าความสัมพันธ์ของข้อมูลระหว่างโมเดลไม่ถูกต้อง	น้อย
2	พบข้อผิดพลาดในไฟล์สิทธิ์ความปลอดภัยเนื่องจากการใช้ชื่อสำหรับการอ้างอิงโมเดลไม่ถูกต้อง	น้อย
3	พบข้อผิดพลาดในไฟล์วิวเนื่องจากการใช้ชื่อสำหรับการอ้างอิงโมเดลไม่ถูกต้อง	บ่อย

หมายเหตุ จุดที่พบความผิดพลาดและความถี่ที่เกิดขึ้น มีความแปรผันตรงกับประสบการณ์ในการพัฒนามอดูลของพนักงานในบริษัท อย่างไรก็ตามระหว่างการสังเกตการณ์พบว่า หากมีพนักงานใหม่มาเข้าร่วมการพัฒนา ก็ยังคงพบปัญหาตามตารางที่ 1 แม้จะมีการสอนวิธีการพัฒนามอดูลแล้วก็ตาม

3.2 สถาปัตยกรรมของมอดูลเจนเนอเรเตอร์

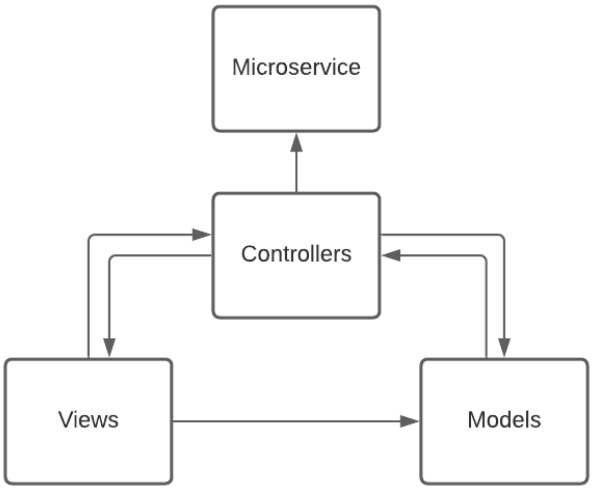
งานวิจัยชิ้นนี้นำเสนอมอดูลเจนเนอเรเตอร์ที่พัฒนาในรูปแบบของมอดูลที่จำเป็นต้องติดตั้งและทำงานภายใต้โอดูซึ่งเป็นเว็บแอปพลิเคชัน (web application) ในขณะที่ผู้ใช้งานออกแบบมอดูลข้อมูลต่าง ๆ จะถูกบันทึกเข้าสู่ฐานข้อมูลของโอดู และรหัสต้นฉบับที่สร้างขึ้นจากการออกแบบของผู้ใช้งานจะถูกสร้างเข้าสู่พื้นที่เก็บข้อมูลตามเส้นทาง (path) ที่ผู้ใช้งานกำหนดและโอดูสามารถเข้าถึงได้ ซึ่งสามารถเขียนภาพรวมสถาปัตยกรรมระบบ (system architecture) ของโอดูได้ดังภาพที่ 9



ภาพที่ 9 ภาพรวมสถาปัตยกรรมระบบของโอดู

ขณะที่ผู้ใช้งานออกแบบมอดูลซึ่งประกอบไปด้วยส่วนประกอบต่าง ๆ มอดูลเจนเนอเรเตอร์จะใช้สภาพแวดล้อมการทำงานที่จัดเตรียมจากโอดูเฟรมเวิร์ก (Odoo framework) เช่น รูปแบบการเขียนโปรแกรมแบบโมเดล-วิว-คอนโทรลเลอร์ และการใช้งานตัวส่งของโมเดลเชิงวัตถุและเชิงสัมพันธ์ เข้ามาช่วยในการบันทึกข้อมูลและตรวจสอบความถูกต้องของข้อมูล

อย่างไรก็ตามการทำงานสำหรับการสร้างรหัสต้นฉบับนั้นจะอยู่ในรูปแบบของชุดโปรแกรมการให้บริการขนาดเล็ก (microservice) ที่ถูกเขียนแยกออกมาจากโอดูเฟรมเวิร์ก และจะถูกเรียกใช้งานผ่านคอนโทรลเลอร์ดังแสดงในภาพที่ 10 ทำให้การพัฒนามอดูลเจนเนอเรเตอร์ในส่วนของการสร้างรหัสต้นฉบับมีส่วนที่ขึ้นกับ (dependency) โอดูเฟรมเวิร์กน้อยลง ส่งผลให้สามารถแก้ไขได้ง่ายในภายหลัง



ภาพที่ 10 แผนภาพแสดงสถาปัตยกรรมซอฟต์แวร์ของมอดูลเจนเนอเรเตอร์

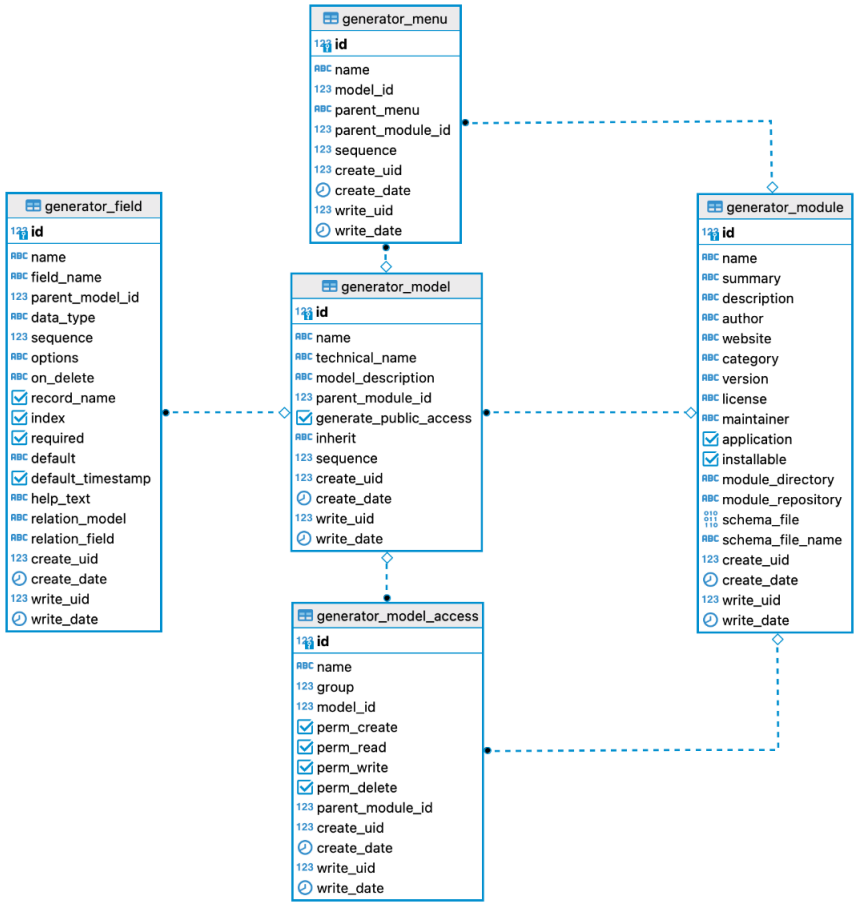
3.3 ฐานข้อมูลของมอดูลเจนเนอเรเตอร์

มอดูลเจนเนอเรเตอร์ถูกออกแบบให้ใช้งานผ่านโอดูในรูปแบบของมอดูล เพื่อให้สามารถใช้เครื่องมือที่ติดตั้งมากับโอดูเช่น การพัฒนาโปรแกรมในรูปแบบของโมเดล-วิว-คอนโทรลเลอร์ ดังนั้น การกรอกข้อมูลเพื่อพัฒนามอดูลผ่านแบบฟอร์มสำหรับส่วนประกอบต่าง ๆ ในมอดูลเจนเนอเรเตอร์จะถูกบันทึกเข้าสู่ฐานข้อมูลเพื่อใช้เป็นข้อมูลตั้งต้นสำหรับการสร้างรหัสต้นฉบับ

ฐานข้อมูลจะถูกสร้างขึ้นเมื่อผู้ใช้งานทำการติดตั้งมอดูลเจนเนอเรเตอร์เข้ากับระบบโอดู โดยตารางข้อมูลต่าง ๆ ที่ใช้งานจะถูกผนวกรวมเข้ากับตารางอื่น ๆ ที่มีในระบบโอดูด้วย ซึ่งโครงร่างฐานข้อมูล (database schema) ของมอดูลเจนเนอเรเตอร์ ประกอบด้วย 5 ตาราง และมีแผนภาพแสดงความสัมพันธ์ระหว่างเอนทิตี้ (ER-diagram) ดังภาพที่ 11 มีรายละเอียดดังนี้

1. ตาราง generator_module สำหรับเก็บข้อมูลมอดูล
2. ตาราง generator_model สำหรับเก็บข้อมูลโมเดล
3. ตาราง generator_field สำหรับเก็บข้อมูลสำหรับเก็บข้อมูลฟิลด์ของโมเดล
4. ตาราง generator_model_access สำหรับเก็บข้อมูลสิทธิ์และความปลอดภัย
5. ตาราง generator_menu สำหรับเก็บข้อมูลเมนูเพื่อแสดงผลหรือวิวของโมเดล

ทั้งนี้ชื่อของตารางทั้ง 5 นั้นเกิดจากการนำชื่อทางเทคนิคของโมเดล (technical name) ที่อยู่ภายใต้มอดูลมาสร้างเป็นชื่อตาราง ซึ่งการสร้างตารางนั้นเป็นกระบวนการแบบอัตโนมัติของตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ของโอดู ที่เกิดขึ้นเมื่อทำการติดตั้งมอดูลเข้าสู่ระบบ



ภาพที่ 11 แผนภาพแสดงความสัมพันธ์ระหว่างเอนทิตี้

3.3.1 ตาราง generator_module

เมื่อผู้ใช้งานต้องการพัฒนามอดูลผ่านมอดูลเจนเนอเรเตอร์ ต้องกำหนดข้อมูลพื้นฐานซึ่งเป็นจุดเริ่มต้นของการสร้างมอดูล ข้อมูลภายในตารางนี้จะใช้เป็นข้อมูลตั้งต้นสำหรับการสร้างไฟล์แมนนิเฟสของมอดูล และไอดี (Identifier: ID) ซึ่งจะใช้เป็นการอ้างอิงส่วนประกอบของมอดูลในตารางอื่น ๆ ต่อไป

3.3.2 ตาราง generator_model

หลังจากที่ผู้ใช้งานกำหนดข้อมูลพื้นฐานเรียบร้อยแล้ว จะต้องออกแบบและสร้างโมเดล โดยตารางนี้จะเก็บข้อมูลชื่อโมเดล ชื่อทางเทคนิคของโมเดลและคำอธิบายของโมเดล โดยจะมีฟิลด์ชื่อ parent_module_id ใช้สำหรับการบ่งบอกว่าโมเดลดังกล่าวอยู่ภายใต้มอดูลใด ซึ่งข้อมูลภายในตารางนี้จะใช้เป็นข้อมูลตั้งต้นสำหรับการสร้างไฟล์โมเดล

3.3.3 ตาราง generator_field

ระหว่างที่ผู้ใช้งานสร้างโมเดล จำเป็นที่จะต้องกำหนดว่าโมเดลดังกล่าวจะมีการใช้ฟิลด์ใดเพื่อใช้ในการเก็บข้อมูล โดยการสร้างฟิลด์นั้นจะรองรับการกำหนดข้อมูลได้แก่ 1) ชนิดของข้อมูล (data type) 2) ชื่อตัวแปร (variable name) ในตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ 3) ค่าตั้งต้นของข้อมูล 4) คำอธิบายข้อมูล 5) ความสัมพันธ์ของข้อมูล (data relationship) และ 6) ข้อมูลที่จำเป็นต้องใส่ (required) ซึ่งข้อมูลดังกล่าวจะเป็นข้อมูลตั้งต้นสำหรับการสร้างไฟล์โมเดลในส่วนของตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ รวมถึงการสร้างวิว โดยมีฟิลด์ชื่อ parent_model_id ใช้สำหรับการบ่งบอกว่าฟิลด์ดังกล่าวเป็นสมาชิกของโมเดลใด

3.3.4 ตาราง generator_model_access

เมื่อผู้ใช้งานสร้างโมเดลแล้ว จะสามารถกำหนดสิทธิ์การเข้าถึงข้อมูลภายในโมเดล ซึ่งผู้ใช้งานสามารถกำหนดสิทธิ์การเข้าถึงข้อมูลได้แก่ สิทธิ์การอ่านข้อมูล สิทธิ์การเขียนข้อมูล สิทธิ์การแก้ไขข้อมูล และสิทธิ์การลบข้อมูล ซึ่งข้อมูลดังกล่าวจะเป็นข้อมูลตั้งต้นสำหรับการสร้างไฟล์สิทธิ์และความปลอดภัยของมอดูล โดยมีฟิลด์ชื่อ model_id เป็นตัวบ่งบอกว่าเป็นการกำหนดสิทธิ์การเข้าถึงข้อมูลให้กับโมเดลใด

3.3.5 ตาราง generator_model_menu

หลังจากผู้ใช้งานสร้างส่วนประกอบของโมเดลเสร็จสิ้นแล้ว สามารถกำหนดเมนูเพื่อทำหน้าที่เป็นปุ่มเพื่อให้ผู้ใช้งานกดเพื่อแสดงหน้าจอของรายการข้อมูลหรือแบบฟอร์มข้อมูล ได้แก่ ชื่อของเมนู ชื่อของโมเดล และลำดับของเมนู ซึ่งข้อมูลดังกล่าวจะเป็นข้อมูลตั้งต้นสำหรับการสร้างเมนูซึ่งเป็นส่วนประกอบส่วนหนึ่งของวิว โดยมีฟิลด์ชื่อ model_id สำหรับบ่งบอกว่าเป็นเมนูสำหรับการแสดงผลของโมเดลใด

3.4 การออกแบบการทำงานของมอดูลเจนเนอเรเตอร์ตามมุมมองการใช้งาน

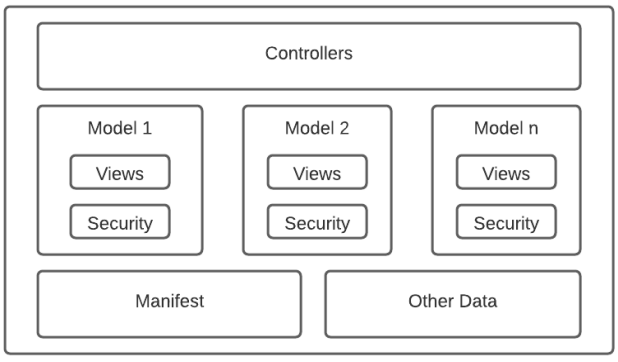
จากการศึกษาในข้อ 3.1 พบว่าการพัฒนามอดูลโดยใช้วิธีการเขียนรหัสต้นฉบับด้วยตนเองนั้นมีความยุ่งยาก เนื่องจากลักษณะโครงสร้างของโปรแกรมที่มีความซับซ้อน ส่งผลให้ผู้เริ่มต้นพัฒนาต้องใช้เวลาในการศึกษาเรียนรู้เป็นเวลานาน ซึ่งโดยปกติแล้วมอดูลที่สร้างขึ้นมาในครั้งแรกนั้นจะมีลักษณะของรหัสต้นฉบับที่เป็นแบบแผนชัดเจน ส่งผลให้ถึงแม้ว่าการพัฒนามอดูลจะทำโดยนักพัฒนาต่างคนกัน จะมีผลลัพธ์ออกมาในลักษณะคล้ายคลึงกัน โดยการตั้งชื่อโมเดลจะใช้เครื่องหมาย จุด (.)

เป็นตัวเชื่อมระหว่างชื่อมอดูลและชื่อโมเดล และมีการกำหนดตัวแปรภายในโมเดลโดยใช้ฟังก์ชันพื้นฐาน (built-in function) ของโอดู

การพัฒนาต้นแบบของมอดูลเจนเนอเรเตอร์นี้จะอยู่ในรูปแบบของมอดูลและทำงานภายใต้โอดู เพื่อให้มอดูลเจนเนอเรเตอร์สามารถใช้ประโยชน์จากฟังก์ชันพื้นฐานของโอดู โดยผู้ใช้งานสามารถติดตั้งมอดูลเจนเนอเรเตอร์เข้ากับโอดูของตนเองก่อนเริ่มการใช้งาน จากนั้นจึงออกแบบและกำหนดข้อมูลต่าง ๆ ผ่าน GUI ซึ่งการพัฒนามอดูลเจนเนอเรเตอร์จะมุ่งเน้นไปที่การสนับสนุนผู้พัฒนาในการพัฒนามอดูลได้โดยง่ายและสามารถนำไปใช้งานหรือพัฒนาต่อได้ทันที โดยผู้ใช้งานสามารถพัฒนามอดูลผ่าน GUI และบันทึกข้อมูลเข้าสู่ฐานข้อมูลของมอดูลเจนเนอเรเตอร์ ซึ่งจะใช้เป็นข้อมูลตั้งต้นสำหรับการสร้างรหัสต้นฉบับของมอดูล

การใช้ GUI สามารถกำหนดข้อมูลตั้งต้นที่จำเป็นสำหรับการสร้างส่วนประกอบต่าง ๆ ของมอดูล ซึ่งส่วนประกอบที่ออกแบบและสร้างได้จากมอดูลเจนเนอเรเตอร์ (ตามขั้นตอนการพัฒนามอดูลในบทที่ 2 หัวข้อ 2.4) ประกอบด้วย

1. มอดูลไดเร็กทอรี (Module directory)
2. ข้อมูลของมอดูล (__manifest__.py)
3. ไฟล์สำหรับตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ (Models)
4. ไฟล์สำหรับตัวกำหนดการแสดงผล (Views)
5. ไฟลสำหรับสิทธิ์และความปลอดภัย (Security)



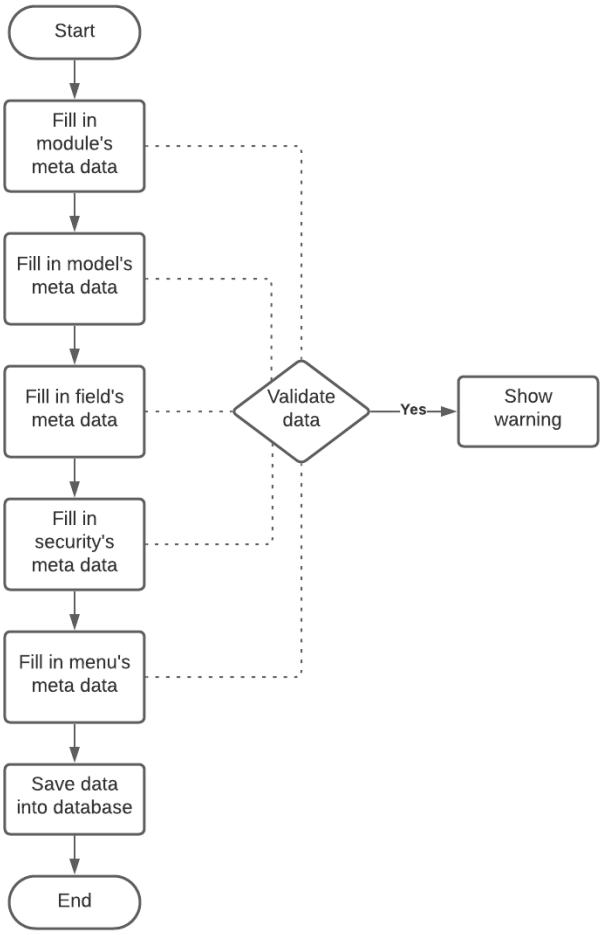
ภาพที่ 12 ส่วนประกอบของมอดูล

จากภาพที่ 12 มอดูลอาจประกอบด้วยส่วนประกอบอื่น ๆ มากกว่าที่มอดูลเจนเนอเรเตอร์รองรับเช่น คอนโทรลเลอร์ (Controllers) ซึ่งทำหน้าที่จัดการคำขอข้อมูลผ่านช่องทางเอชทีทีพี (HTTP Protocol) ทั้งนี้ส่วนประกอบอื่น ๆ ที่ไม่ปรากฏในมอดูลเจนเนอเรเตอร์ ไม่ได้อยู่ในขอบเขต

ของการวิจัยในครั้งนี้ อย่างไรก็ตามผู้พัฒนาสามารถเขียนรหัสต้นฉบับเพิ่มเติมในผลลัพธ์ของมอดูลเจนเนอเรเตอร์เพิ่มเติมได้เองในภายหลัง

3.4.1 การออกแบบมอดูลโดยใช้มอดูลเจนเนอเรเตอร์

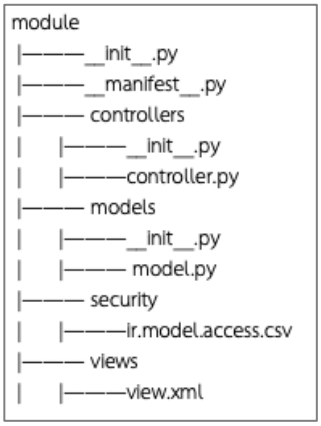
การออกแบบและกำหนดข้อมูลต่าง ๆ ผ่าน GUI ของมอดูลเจนเนอเรเตอร์ ช่วยลดขั้นตอนในการทำงานให้กับผู้พัฒนา โดยผู้พัฒนาสามารถเขียนโปรแกรมโดยการใส่ข้อมูลลงไปเพียงครั้งเดียว หลังจากนั้นมอดูลเจนเนอเรเตอร์จะดึงข้อมูลไปดำเนินการสร้างรหัสต้นฉบับให้โดยอัตโนมัติ ซึ่งสามารถอธิบายขั้นตอนการออกแบบมอดูลได้ดังนี้ เริ่มจากใส่ข้อมูลต่าง ๆ ของมอดูล ระหว่างนั้นมอดูลเจนเนอเรเตอร์จะตรวจสอบความถูกต้องของข้อมูลได้แก่ ตรวจสอบการตั้งชื่อมอดูลและโมเดลไม่ให้มีอักขระพิเศษ การเตือนเมื่อใส่ข้อมูลไม่ครบถ้วน และการใส่ข้อมูลให้อัตโนมัติในส่วนที่เป็นข้อมูลทางเทคนิคประกอบด้วย ชื่อทางเทคนิคของโมเดล ชื่อฟิลด์ข้อมูลสำหรับการแสดงผล โดยโปรแกรมจะตัดส่วนที่เป็นอักขระพิเศษรวมถึงช่องว่างก่อนนำไปเติมในช่องข้อมูลดังกล่าว หลังจากนั้นจะบันทึกข้อมูลทั้งหมดเข้าสู่ฐานข้อมูลของมอดูลเจนเนอเรเตอร์เพื่อใช้เป็นข้อมูลตั้งต้นในการสร้างไดเร็กทอรีและรหัสต้นฉบับสำหรับส่วนประกอบต่าง ๆ ดังแสดงตามภาพที่ 13



ภาพที่ 13 ผังงานวิธีการออกแบบมอดูลด้วยมอดูลเจนเนอเรเตอร์

3.4.2 การสร้างรหัสต้นฉบับโดยใช้มอดูลเจนเนอเรเตอร์

มอดูลเจนเนอเรเตอร์รองรับการสร้างรหัสต้นฉบับสำหรับมอดูล โดยใช้ข้อมูลตั้งต้นจากฐานข้อมูลของมอดูลเจนเนอเรเตอร์ที่ได้จากการออกแบบในข้อ 3.4.1 สามารถอธิบายกระบวนการสำหรับการสร้างรหัสต้นฉบับได้ดังนี้ เริ่มจากมอดูลเจนเนอเรเตอร์ดึงข้อมูลตั้งต้นเพื่อตรวจสอบตำแหน่งสำหรับการสร้างไฟล์ผลลัพธ์ โดยจะมีการตรวจสอบสิทธิ์ของการเขียนไฟล์ก่อนเริ่มสร้างไดเร็กทอรีสำหรับมอดูล ที่มีโครงสร้างดังแสดงในภาพที่ 14 ประกอบด้วย 1) ไดเร็กทอรีสำหรับมอดูล จะใช้เป็นชื่อมอดูลด้วยตัวอักษรภาษาอังกฤษพิมพ์เล็ก 2) ไดเร็กทอรีสำหรับคอนโทรลเลอร์ โมเดล วิว และสิทธิ์และความปลอดภัย ลำดับถัดมาโปรแกรมจะตรวจสอบข้อมูลตั้งต้นของโมเดล สิทธิ์และความปลอดภัย และข้อมูลเมนูตามลำดับ โดยหากมีข้อมูลตั้งต้นรายการใดปรากฏ โปรแกรมจะทำการสร้างไฟล์ของข้อมูลให้อยู่ในไดเร็กทอรีตามประเภทของไฟล์ จากนั้นจึงสร้างไฟล์แมนนิเฟสเป็นลำดับสุดท้ายก่อนจบการทำงาน

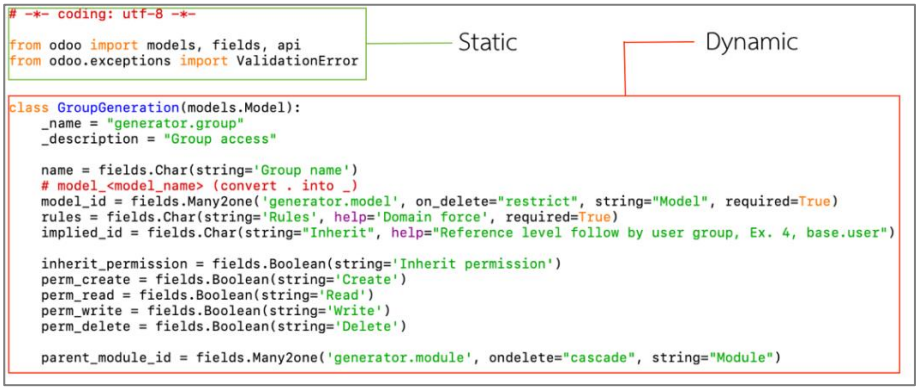


ภาพที่ 14 โครงสร้างไดเร็กทอรีและไฟล์ต่าง ๆ ของมอดูล

จากภาพที่ 14 แสดงให้เห็นไดเร็กทอรีและรหัสต้นฉบับที่ถูกสร้างโดยมอดูลเจนเนอเรเตอร์ โดยรูปแบบของไฟล์รหัสต้นฉบับนั้นประกอบด้วย ไพธอน ซีเอสวี และเอ็กซ์เอ็มแอล ซึ่งมีวิธีและกระบวนการสร้างที่แตกต่างกัน ดังนี้

3.4.2.1 การสร้างรหัสต้นฉบับสำหรับไฟล์รูปแบบไพธอนและซีเอสวี

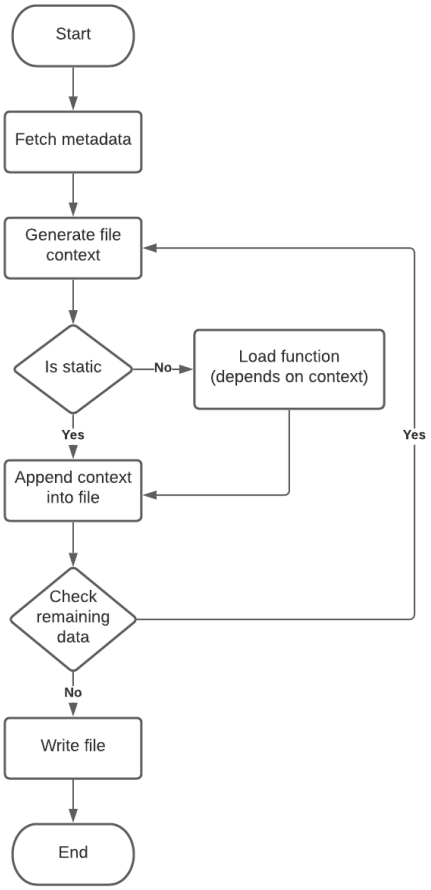
การสร้างรหัสต้นฉบับของไฟล์ภาษาไพธอนและซีเอสวี ใช้การเขียนโปรแกรมโดยประยุกต์หลักการของโค้ดเทมเพลต (code template) มาปรับใช้สำหรับการสร้างไฟล์ที่มีลักษณะไม่ตายตัวและไม่มีมาตรฐานรองรับ ซึ่งแตกต่างต่างกับไฟล์เอ็กซ์เอ็มแอลที่มีมาตรฐานรองรับโดย World Wide Web Consortium (W3C) ดังนั้นการประยุกต์ใช้หลักการของโค้ดเทมเพลตจะมีส่วนช่วยในการสร้างไฟล์ดังกล่าว โดยมีลักษณะการทำงาน คือ เริ่มจากศึกษารูปแบบไฟล์ที่ต้องการสร้าง เพื่อแบ่งกลุ่มของส่วนประกอบของเนื้อหาภายในไฟล์ออกเป็น 2 รูปแบบได้แก่เนื้อหาแบบคงที่ (static) คือเนื้อหาที่ไม่มีการเปลี่ยนแปลงไปตามข้อมูลเช่น ส่วนหัวของไฟล์ไพธอน หรือส่วนหัวของไฟล์ซีเอสวี และรูปแบบที่ 2 คือเนื้อหาที่มีความยืดหยุ่น (dynamic) คือเนื้อหาที่เปลี่ยนแปลงไปตามลักษณะของข้อมูลตั้งต้นเช่น ข้อมูลสิทธิ์การเข้าถึงข้อมูลในโมเดล หรือชื่อตัวแปรของตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์ ดังแสดงในภาพที่ 15




```python
# -*- coding: utf-8 -*-

from odoo import models, fields, api
from odoo.exceptions import ValidationError

class GroupGeneration(models.Model):
    _name = "generator.group"
    _description = "Group access"

    name = fields.Char(string='Group name')
    # model_<model_name> (convert . into _)
    model_id = fields.Many2one('generator.model', on_delete="restrict", string="Model", required=True)
    rules = fields.Char(string='Rules', help='Domain force', required=True)
    implied_id = fields.Char(string="Inherit", help="Reference level follow by user group, Ex. 4, base.user")

    inherit_permission = fields.Boolean(string='Inherit permission')
    perm_create = fields.Boolean(string='Create')
    perm_read = fields.Boolean(string='Read')
    perm_write = fields.Boolean(string='Write')
    perm_delete = fields.Boolean(string='Delete')

    parent_module_id = fields.Many2one('generator.module', ondelete="cascade", string="Module")
```

ภาพที่ 15 การแบ่งกลุ่มส่วนประกอบของไฟล์ตามลักษณะของเนื้อหา

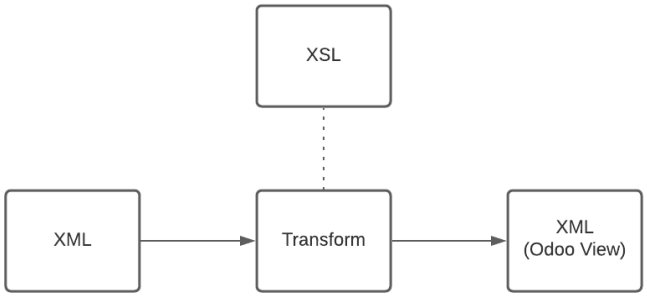
จากภาพที่ 15 เนื้อหาที่มีความยืดหยุ่นนั้นจะมีฟังก์ชันสำหรับการสร้างรหัสต้นฉบับที่เปลี่ยนไปตามข้อมูลตั้งต้น จากนั้นจึงนำเนื้อหาทั้ง 2 รูปแบบมาต่อกันเพื่อให้ได้ข้อมูลทั้งหมดก่อนจะเขียนข้อมูลดังกล่าวเข้าสู่ไดเร็กทอรีตามประเภทของไฟล์ ซึ่งสามารถเขียนลำดับการทำงานออกมาได้ตามภาพที่ 16



ภาพที่ 16 ผังงานวิธีการสร้างไฟล์โดยใช้โค้ดเทมเพลต

3.4.2.2 การสร้างรหัสต้นฉบับสำหรับไฟล์รูปแบบเอ็กซ์เอ็มแอล

ไฟล์รูปแบบเอ็กซ์เอ็มแอลที่ใช้ในโอดูนั้นมีไว้สำหรับกำหนดการแสดงผลของโมเดล เช่น มุมมองของรายการแบบต้นไม้ (tree view) หรือมุมมองของแบบฟอร์มสำหรับใส่ข้อมูล (form view) ซึ่งมีลักษณะการเขียนที่เหมือนกันสำหรับทุกโมเดล โดยข้อมูลที่ใช้สำหรับสร้างวิวนั้นได้แก่ข้อมูลพื้นฐานของโมเดลและข้อมูลฟิลด์ของโมเดลนั้น ๆ โดยเครื่องมือสำหรับใช้ในกระบวนการสร้างไฟล์รูปแบบนี้คือเอ็กซ์เอสแอลที (Extensible Stylesheet Language Transformations: XSLT) หรือภาษาสำหรับการแปลงเอกสารประเภทเอ็กซ์เอ็มแอลให้อยู่ในรูปแบบที่ต้องการ ดังแสดงตามภาพที่ 17 ซึ่งจะใช้ไฟล์เอ็กซ์เอสแอล (XSL) สำหรับการกำกับรูปแบบของผลลัพธ์หรือไฟล์รูปแบบเอ็กซ์เอ็มแอลดังแสดงตามภาพที่ 18



ภาพที่ 17 กระบวนการแปลงข้อมูลเอ็กซ์เอ็มแอลโดยใช้เอ็กซ์เอสแอลที

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <odoo>
            <data>
                <!-- explicit list view definition -->
                <record model="ir.ui.view">
                    <xsl:attribute name="id">
                        <xsl:value-of select="concat(model/model_name_technical, '_list')"/>
                    </xsl:attribute>
                    <field name="name"><xsl:value-of select="model/model_name"/> List</field>
                    <field name="model"><xsl:value-of select="model/module_name_technical"/>.<xsl:value-of select="model/model_name_technical"/></field>
                    <field name="arch" type="xml">
                        <tree>
                            <xsl:for-each select="models/form_view/item">
                                <xsl:element name="field">
                                    <xsl:attribute name="name">
                                        <xsl:value-of select="field"/>
                                    </xsl:attribute>
                                </xsl:element>
                            </xsl:for-each>
                        </tree>
                    </field>
                </record>

                <record model="ir.ui.view">
                    <xsl:attribute name="id">
                        <xsl:value-of select="concat(model/model_name_technical, '_form')"/>
                    </xsl:attribute>
                    <field name="name"><xsl:value-of select="model/model_name"/> Form</field>
                    <field name="model"><xsl:value-of select="model/module_name_technical"/>.<xsl:value-of select="model/model_name_technical"/></field>
                    <field name="arch" type="xml">
                        <form>
                            <sheet>
                                <group>
                                    <xsl:for-each select="model/form_view/item">
                                        <xsl:element name="field">
                                            <xsl:attribute name="name">
                                                <xsl:value-of select="field"/>
                                            </xsl:attribute>
                                        </xsl:element>
                                    </xsl:for-each>
                                </group>
                            </sheet>
                        </form>
                    </field>
                </record>

                <record model="ir.actions.act_window">
                    <xsl:attribute name="id">
                        <xsl:value-of select="concat(model/model_name_technical, '_action')"/>
                    </xsl:attribute>
                    <field name="name"><xsl:value-of select="model/model_name"/> List</field>
                    <field name="type">ir.actions.act_window</field>
                    <field name="res_model"><xsl:value-of select="model/module_name_technical"/>.<xsl:value-of select="model/model_name_technical"/></field>
                    <field name="view_mode">tree,form</field>
                    <xsl:element name="field">
                        <xsl:attribute name="name">view_id</xsl:attribute>
                        <xsl:attribute name="ref">
                            <xsl:value-of select="concat(model/model_name_technical, '_list')"/>
                        </xsl:attribute>
                    </xsl:element>
                    <field name="help" type="html">
                        <p class="oe_view_nocontent_create">There is no record here.</p>
                    </field>
                </record>
            </data>
        </odoo>
    </xsl:template>
</xsl:stylesheet>
```

ภาพที่ 18 ไฟล์รูปแบบเอ็กซ์เอสแอลสำหรับใช้แปลงข้อมูลเอ็กซ์เอ็มแอล

สำหรับไฟล์ข้อมูลเอ็กซ์เอ็มแอลที่ใช้เป็นข้อมูลตั้งต้น จะถูกจัดทำจากข้อมูลของโมเดล ได้แก่ ชื่อโมเดล ชื่อทางเทคนิคของโมเดล คำอธิบายและฟิลด์ของโมเดล (ภาพที่ 19) ก่อนนำมาเข้าสู่กระบวนการแปลง (transform) โดยใช้ไฟล์เอ็กซ์เอสแอลเพื่อให้มีข้อมูลอยู่ในลักษณะที่โอดูกำหนด

```
<?xml version="1.0" encoding="UTF-8"?>
<model>
    <model_name>Module Generation</model_name>
    <model_name_technical>module_generation</model_name_technical>
    <list_view>
        <item><field>name</field></item>
        <item><field>summary</field></item>
        <item><field>description</field></item>
    </list_view>
    <form_view>
        <item>
            <field>name</field>
            <type>char</type>
        </item>
        <item>
            <field>summary</field>
            <type>char</type>
        </item>
        <item>
            <field>description</field>
            <type>text</type>
        </item>
    </form_view>
</model>
```

ภาพที่ 19 ข้อมูลตั้งต้นที่ถูกจัดทำในรูปแบบเอ็กซ์เอ็มแอล

3.5 เครื่องมือที่ใช้ในการออกแบบและพัฒนามอดูลเจนเนอเรเตอร์

มอดูลเจนเนอเรเตอร์ในงานวิจัยชิ้นนี้ ถูกพัฒนาขึ้นโดยใช้ภาษาไพธอนเวอร์ชัน 3.6.0 และฐานข้อมูลโพสเกรสเอสคิวแอล (PostgreSQL) เวอร์ชัน 10.0 และรองรับการใช้งานกับโอดูผ่านเว็บเบราว์เซอร์ ได้แก่ Google Chrome, Microsoft Edge และ Safari โดยผู้วิจัยได้พัฒนามอดูลเจนเนอเรเตอร์ให้สอดคล้องกับวัตถุประสงค์ในการทำงานวิจัยในครั้งนี้ 1) เพื่อให้ผู้ใช้สามารถใช้งานกับโอดูได้ทันทีโดยไม่จำเป็นต้องติดตั้งซอฟต์แวร์อื่น ๆ เพิ่มเติม 2) การพัฒนามอดูลเจนเนอเรเตอร์เป็นมอดูลที่ทำงานภายใต้โอดู จะช่วยให้ผู้วิจัยสามารถเรียนรู้เกี่ยวกับรหัสต้นฉบับที่มีความสำคัญในการพัฒนามอดูล และสามารถนำมาปรับปรุงการทำงานของมอดูลเจนเนอเรเตอร์ในการสร้างรหัสต้นฉบับที่สามารถสนับสนุนผู้ใช้งานได้มากขึ้นระหว่างการพัฒนา 3) ผู้วิจัยสามารถใช้เครื่องมือต่าง ๆ ที่โอดูมีอยู่ในการพัฒนามอดูลเจนเนอเรเตอร์ได้แก่ การพัฒนาโปรแกรมในรูปแบบโมเดล-วิว-คอนโทรลเลอร์ ที่คุ้นชินกับผู้พัฒนามอดูลอยู่แล้ว รวมถึงไลบรารี (library) พื้นฐานที่โอดูมี เช่น การอ่านและประมวลผลข้อมูลเอ็กซ์เอ็มแอล

บทที่ 4
ผลการวิจัยและอภิปรายผล

ในบทนี้ผู้วิจัยจะนำสิ่งที่ได้ศึกษาและขั้นตอนการออกแบบไว้ในบทที่ 3 มาพัฒนาโปรแกรมชื่อ “มอดูลเจนเนอเรเตอร์” เพื่อสนับสนุนการพัฒนามอดูลโดยใช้เทคนิคโลว์โค้ดซึ่งมีรายละเอียดดังนี้

4.1 ภาพรวมของมอดูลเจนเนอเรเตอร์

มอดูลเจนเนอเรเตอร์ เป็นเครื่องมือที่ช่วยให้ผู้ใช้งานสามารถกำหนดส่วนประกอบต่าง ๆ ที่จำเป็นสำหรับการสร้างมอดูลใหม่ผ่านหน้าจอแสดงผล และสามารถสร้างผลลัพธ์เป็นไดเร็กทอรีของมอดูลและรหัสต้นฉบับที่คล้ายคลึงกับการสร้างมอดูลด้วยตนเอง

ในการพัฒนาเวอร์ชันแรกนี้ การออกแบบตัวสร้างมอดูลมุ่งเน้นไปที่การช่วยให้ผู้ใช้งานสามารถพัฒนามอดูลที่พร้อมใช้งานได้ทันที นอกจากนี้ยังสามารถพัฒนาส่วนประกอบที่จำเป็นในการสร้างมอดูลและลดรายละเอียดที่ซับซ้อนจากการเขียนโปรแกรมด้วยตนเอง ส่วนประกอบของมอดูลที่สามารถสร้างได้ด้วยมอดูลเจนเนอเรเตอร์มีดังต่อไปนี้

1) ไดเร็กทอรีพร้อมโครงสร้างสำหรับวางส่วนประกอบของมอดูลตามมาตรฐานของโอดู
2) ไฟล์เมนิเฟส (__manifest__.py) สำหรับข้อมูลพื้นฐานของมอดูล
3) ไฟล์อินนิท (__init__.py) สำหรับการกำหนดค่าเริ่มต้นโปรแกรมของมอดูล
4) ไฟล์โมเดล (model.py) สำหรับตัวส่งระหว่างโมเดลเชิงวัตถุและเชิงสัมพันธ์
5) ไฟล์วิว (view.xml) สำหรับกำหนดการแสดงผลสำหรับโมเดล
6) ไฟล์สิทธิ์และความปลอดภัย (ir.access.model.csv) สำหรับการกำหนดสิทธิ์การเข้าถึงสำหรับโมเดล

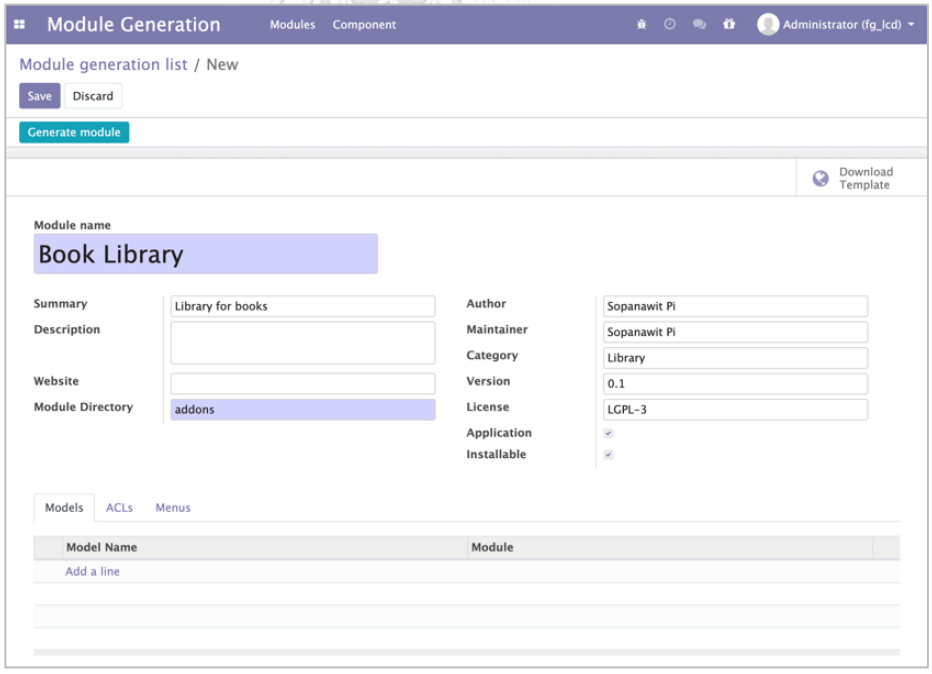
4.2 การพัฒนามอดูลโดยใช้มอดูลเจนเนอเรเตอร์

การสร้างมอดูลใหม่ขึ้นมาสามารถกำหนดรูปแบบและคุณลักษณะต่าง ๆ ของมอดูล โดยมีรายละเอียดหน้าจอแสดงผลการใช้งานของมอดูลเจนเนอเรเตอร์ ดังนี้

4.2.1 เริ่มต้นสร้างมอดูล

การเริ่มต้นสร้างมอดูลสามารถใช้แบบฟอร์มดังแสดงตามภาพที่ 20 เพื่อกำหนดคุณลักษณะและข้อมูลพื้นฐานของมอดูล มีรายละเอียดดังนี้

- ชื่อของมอดูล (Module name)
- คำอธิบายมอดูลแบบย่อ (Summary)
- คำอธิบายมอดูลแบบเต็ม (Description)
- เว็บไซต์สำหรับข้อมูลมอดูล (Website)
- ผู้เป็นเจ้าของ (Author)
- ผู้พัฒนา (Maintainer)
- ประเภทของมอดูล (Category)
- เวอร์ชัน (Version)
- ประเภทของการอนุญาตให้ใช้งาน (License)
- การปรากฏเป็นแอปพลิเคชันบนโอดู (Application)
- การอนุญาตให้ผู้ใช้ติดตั้งด้วยตนเอง (Installable)
- สถานที่ในการสร้างมอดูล (Module Directory)

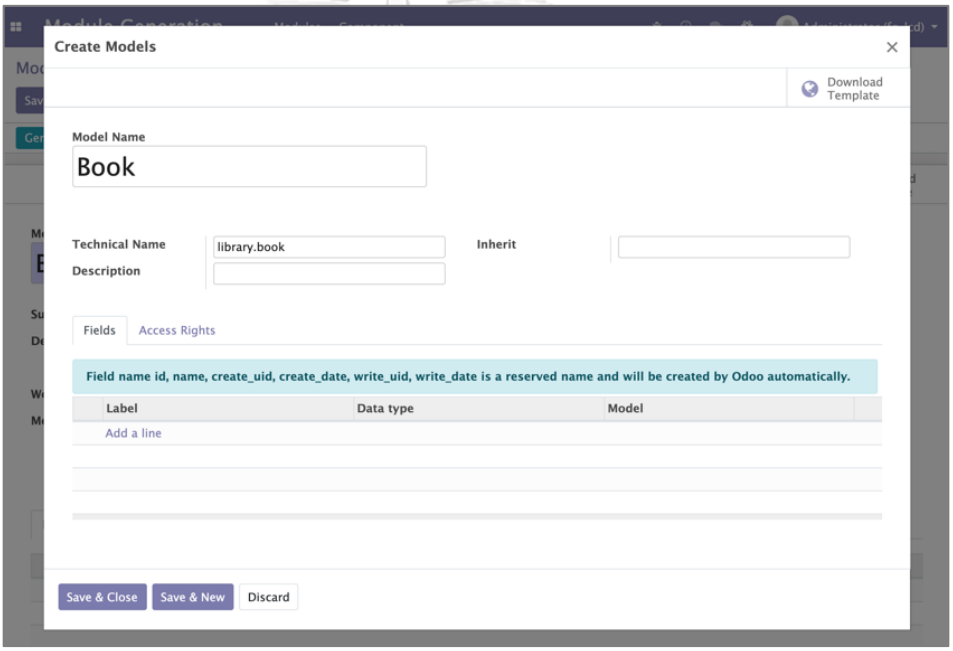


ภาพที่ 20 หน้าจอแบบฟอร์มสำหรับสร้างมอดูล

4.2.2 การสร้างโมเดล

แบบฟอร์มของสำหรับการออกแบบโมเดลดังแสดงตามภาพที่ 21 สามารถกดสร้างโมเดลสำหรับเก็บข้อมูลและกำหนดรายละเอียดต่าง ๆ ที่จำเป็น โดยข้อมูลในแบบฟอร์มนี้จะถูกนำไปสร้างเป็นไฟล์โมเดล โดยมีรายละเอียดดังนี้

- ชื่อโมเดล (Model Name)
- ชื่อโมเดลสำหรับใช้อ้างอิงในโปรแกรม (Technical name)
- คำอธิบายโมเดล (Description)
- ชื่อโมเดลที่ต้องการสืบทอด (Inherit)



ภาพที่ 21 หน้าจอแบบฟอร์มสำหรับสร้างโมเดล

4.2.3 การสร้างฟิลด์

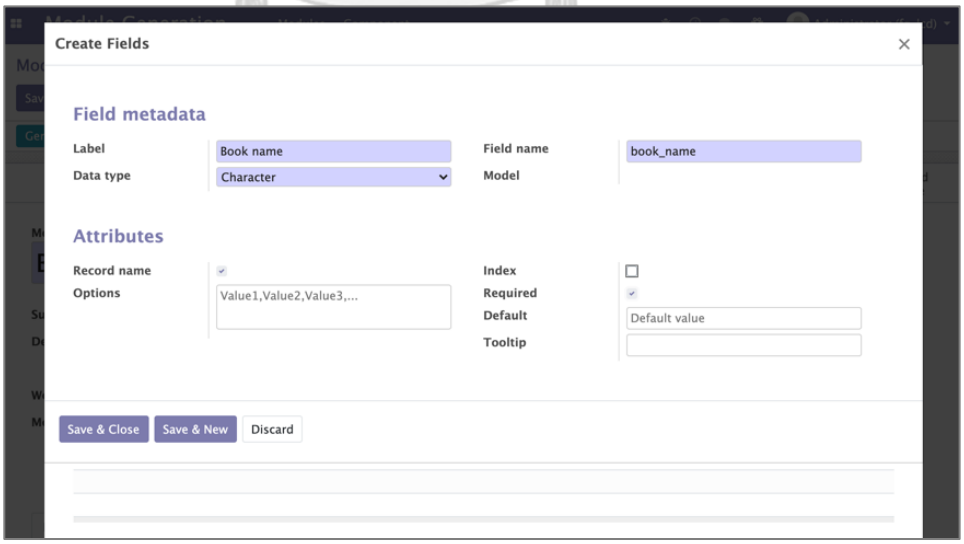
แบบฟอร์มของสำหรับการออกแบบฟิลด์ดังแสดงตามภาพที่ 22 สามารถกดสร้างฟิลด์สำหรับใช้เก็บข้อมูล โดยสามารถเลือกชนิดของข้อมูลที่จะใช้เก็บ รวมไปถึงการตั้งค่าเริ่มต้น โดยชนิดของข้อมูลที่รองรับประกอบด้วย

- ค่าความเป็นจริง (Boolean)
- ตัวเลขจำนวนเต็ม (Integer)

- ตัวเลขทศนิยม (Floating point number)
- ตัวอักษรแบบกำหนดความยาวได้ (Character)
- ตัวอักษรแบบหลายบรรทัด (Text)
- วันที่ (Date)
- วันที่และเวลา (Date Time)
- ข้อมูลไบนารี (Binary)
- ข้อมูลแบบกลุ่มที่ยอมให้เลือกได้ 1 ค่า (Selection)
- ข้อมูลสำหรับเก็บตัวเลขของเงิน (Monetary)
- การเชื่อมโยงข้อมูลแบบ 1 ต่อจำนวนมาก (One to Many)
- การเชื่อมโยงข้อมูลแบบ จำนวนมากต่อ 1 (Many to One)
- การเชื่อมโยงข้อมูลแบบ จำนวนมากต่อจำนวนมาก (Many to Many)

สำหรับการสร้างฟิลด์จะสามารถกำหนดคุณสมบัติของข้อมูลได้ดังนี้

- การแสดงผลชื่อบนระบบโอดู (String)
- ห้ามเป็นข้อมูลที่ว่าง (Required)
- การเก็บตำแหน่งของข้อมูลสำหรับการค้นหาที่รวดเร็ว (Index)
- ค่าเริ่มต้น (Default)

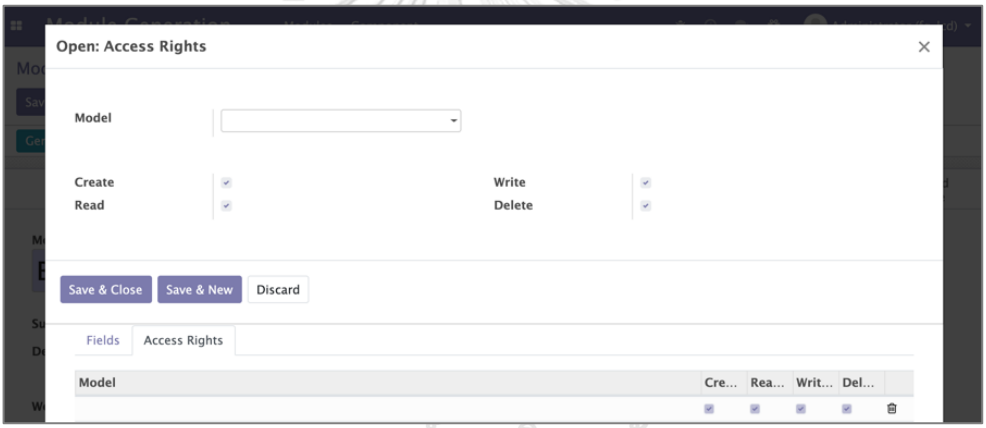


ภาพที่ 22 หน้าจอแบบฟอร์มสำหรับสร้างฟิลด์

4.2.4 การสร้างสิทธิ์การเข้าถึงข้อมูลภายในโมเดล

ภายใต้แบบฟอร์มของโมเดล สามารถสร้างสิทธิ์การเข้าถึงข้อมูลสำหรับโมเดลนั้น ๆ ได้ สามารถกำหนดสิทธิ์จะครอบคลุมการอ่านข้อมูล การเขียนข้อมูล การแก้ไขข้อมูล และการลบข้อมูลตามแบบฟอร์มที่ปรากฏดังภาพที่ 23 โดยสิทธิ์ที่กำหนดนี้จะถูกกำหนดให้กับผู้ใช้งานทั้งหมดบนโอดู โดยมีรายละเอียดดังนี้

- ชื่อโมเดล (Model)
- สิทธิ์ในการสร้างข้อมูล (Create)
- สิทธิ์ในการอ่านข้อมูล (Read)
- สิทธิ์ในการเขียนข้อมูล (Write)
- สิทธิ์ในการลบข้อมูล (Delete)

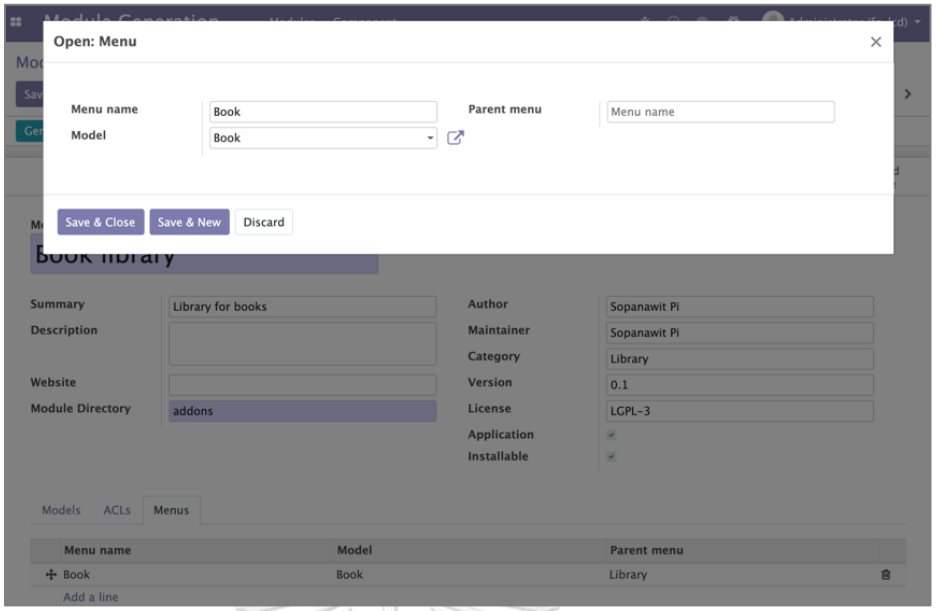


ภาพที่ 23 หน้าจอแบบฟอร์มสำหรับสร้างสิทธิ์การเข้าถึงข้อมูล

4.2.5 การสร้างเมนู

แบบฟอร์มสำหรับการสร้างเมนูดังแสดงตามภาพที่ 24 สามารถสร้างเมนูเพื่อใช้เป็นช่องทางในการเปิดหน้ารายการข้อมูลหรือแบบฟอร์มของโมเดล โดยมีรายละเอียดดังนี้

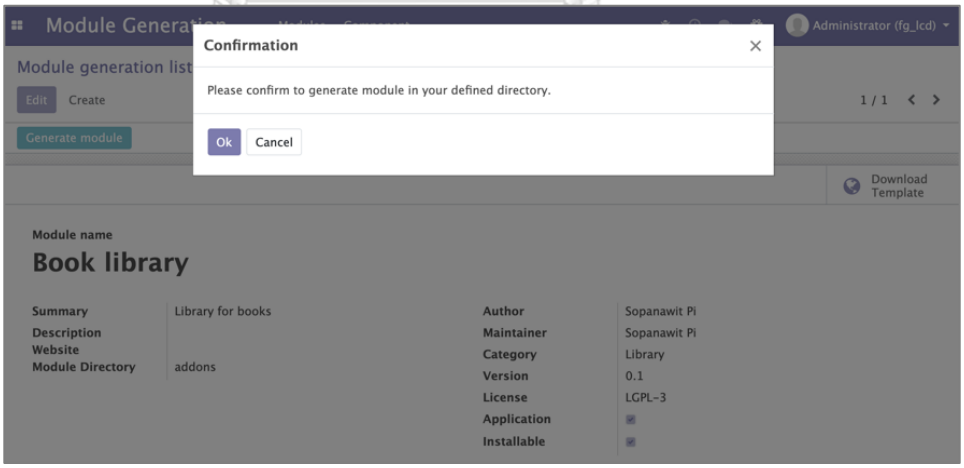
- ชื่อเมนู (Menu name)
- ชื่อโมเดล (Model)
- ชื่อเมนูที่ต้องการแสดงผลแบบตัวเลือก (Parent menu)



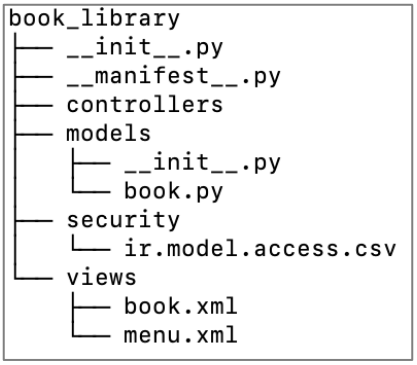
ภาพที่ 24 หน้าจอแบบฟอร์มสำหรับสร้างเมนู

4.2.6 การสร้างรหัสต้นฉบับ (Generate)

เมื่อผู้พัฒนากำหนดข้อมูลครบถ้วนแล้ว สามารถกดปุ่ม “Generate module” และยืนยันการสร้างรหัสต้นฉบับดังแสดงตามภาพที่ 25 เพื่อให้โปรแกรมสร้างมอดูลพร้อมรหัสต้นฉบับในพื้นที่ที่กำหนด โดยมีผลลัพธ์เป็นไดเร็กทอรีและไฟล์ดังแสดงตามภาพที่ 26



ภาพที่ 25 หน้าจอเพื่อยืนยันการสร้างโค้ดมอดูล



ภาพที่ 26 ไดเร็กทอรีและไฟล์ที่ถูกสร้างขึ้นโดยมอดูลเจนเนอเรเตอร์

4.3 การทดสอบการใช้งานมอดูลเจนเนอเรเตอร์

การทดสอบการใช้งานมอดูลเจนเนอเรเตอร์แบ่งเป็น 2 การทดสอบได้แก่ 1) การทดสอบการใช้งานเบื้องต้น 2) การทดสอบการใช้งานโดยอาสาสมัครภายนอก โดยมีกลุ่มผู้ทดสอบและจำนวนผู้ทดสอบดังแสดงในตารางต่อไปนี้

ตารางที่ 2 กลุ่มผู้ทดสอบและจำนวนผู้ทดสอบในการทดสอบใช้งานมอดูลเจนเนอเรเตอร์

การทดสอบใช้งาน / กลุ่มผู้ทดสอบ	**จำนวนผู้ทดสอบ**	
	การทดสอบการใช้งานเบื้องต้น	**การทดสอบการใช้งานโดยอาสาสมัครภายนอก**
กลุ่มที่ 1 ผู้พัฒนามอดูลบนโอดูที่มีประสบการณ์ใกล้เคียงกัน (Odoo developer)	3	-
กลุ่มที่ 2 ผู้ใช้งานโอดูที่มีทักษะการเขียนโปรแกรม (Odoo user)	1	-
กลุ่มที่ 3 ผู้ที่ไม่เคยมีประสบการณ์กับโอดู (Non-Odoo user)	1	41

การทดสอบจะให้ผู้ทดสอบพัฒนามอดูลสำหรับระบบบันทึกข้อมูลผู้ได้รับความเดือดร้อนจากผลกระทบของโรคโควิด 19 ในประเทศไทย ซึ่งมีโครงสร้างและองค์ประกอบของมอดูลที่ใช้งานจริงในกระทรวงการวิทยาศาสตร์ วิจัย และนวัตกรรม (อว.) เพื่อเปรียบเทียบระยะเวลาที่ผู้ทดสอบทั้ง 3 กลุ่มใช้ในการพัฒนามอดูล ดังนี้ 1) ระยะเวลารวมที่ใช้ในการพัฒนามอดูล 2) ระยะเวลาที่ใช้ในการสร้างแต่ละโมเดล รวมถึงเปรียบเทียบเวลาเฉลี่ยที่ผู้ทดสอบในกลุ่มที่ 1 ใช้ในการพัฒนามอดูล

ระหว่างวิธีการเขียนรหัสต้นฉบับเองทั้งหมดและใช้มอดูลเจนเนอเรเตอร์ซึ่งมีการสร้างรหัสต้นฉบับให้โดยอัตโนมัติ

ข้อมูลสำหรับใช้ในการทดสอบทั้งหมด ประกอบด้วยโมเดลจำนวน 7 โมเดล และมีฟิลด์ข้อมูลรวม 51 ฟิลด์ โดยแต่ละโมเดลจะมีจำนวนฟิลด์ที่แตกต่างกันดังแสดงในตารางที่ 3

ตารางที่ 3 จำนวนข้อมูลของแต่ละโมเดลที่ใช้ในการทดสอบ

โมเดลที่	จำนวนฟิลด์	จำนวนชนิดข้อมูลที่แตกต่างกัน
1	26	6
2	10	4
3	2	2
4	7	4
5	3	2
6	2	2
7	1	1
รวม	**51**	

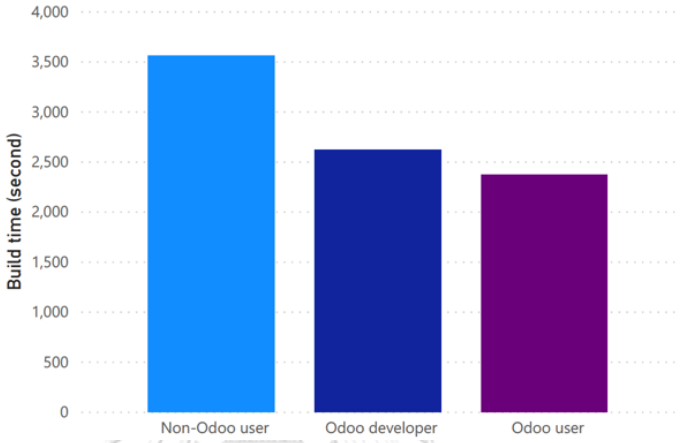
จากตารางที่ 3 แสดงถึงจำนวนฟิลด์และจำนวนชนิดของข้อมูล (data type) ที่แตกต่างกันในแต่ละโมเดล แสดงให้เห็นถึงความหลากหลายของข้อมูลภายในโครงร่างฐานข้อมูลของระบบบันทึกข้อมูลฯ ที่ผู้ทดสอบจะต้องพัฒนามอดูลขึ้นมาในการทดสอบ และสามารถเทียบเคียงได้กับการพัฒนามอดูลเพื่อใช้ในโครงการพัฒนาซอฟต์แวร์ในสถานการณ์จริง

4.3.1 การทดสอบการใช้งานเบื้องต้น

การทดสอบการใช้งานเบื้องต้นเกิดขึ้นภายในบริษัท แฟคเกอร์ จำกัด ที่ผู้วิจัยเข้าสังเกตการณ์การทำงานเพื่อศึกษาและวิเคราะห์ปัญหาที่เกิดขึ้นจากการพัฒนามอดูล โดยมีผู้ทดสอบเป็นพนักงานภายในบริษัทจำนวน 5 คน สามารถจำแนกคุณสมบัติของผู้เข้าทดสอบได้เป็น 3 กลุ่มดังแสดงในตารางที่ 2 การทดสอบเริ่มจากผู้วิจัยอธิบายการใช้งานมอดูลเจนเนอเรเตอร์พร้อมมอบเอกสารประกอบการอธิบาย รวมถึงเอกสารโครงร่างฐานข้อมูลของระบบบันทึกข้อมูลฯ เพื่อใช้สำหรับอ้างอิงในการทดสอบ จากนั้นผู้ทดสอบจะพัฒนามอดูลโดยใช้ข้อมูลตามเอกสารที่ได้รับ โดยมีผู้วิจัยเป็นผู้จับเวลาในการทดสอบ ดังนี้ 1) เวลารวมที่ใช้ในการสร้างมอดูล 2) เวลาที่ใช้ในการสร้างแต่ละโมเดล ทั้งนี้เวลาในการสร้างมอดูลทั้งหมดจะมากกว่าเวลาการ

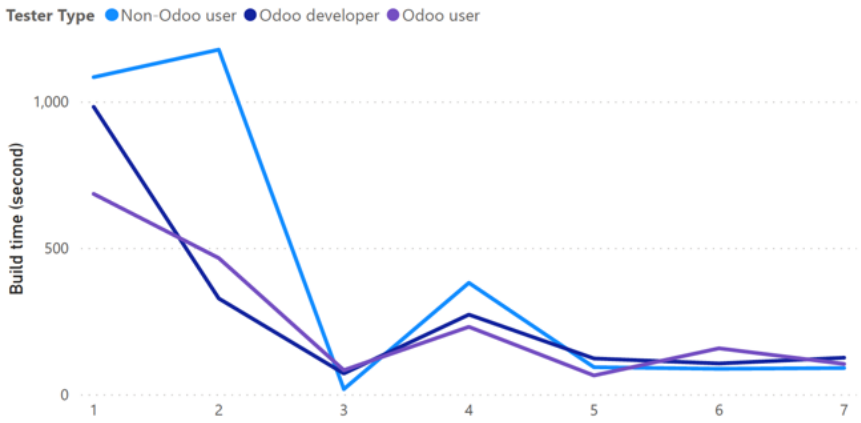
สร้างแต่ละโมเดลรวมกัน เนื่องจากผู้ทดสอบจะต้องสร้างส่วนประกอบอื่น ๆ ของมอดูลด้วยได้แก่ สิทธิ์การเข้าถึงข้อมูลและวิว รวมถึงการแก้ไขข้อผิดพลาดระหว่างการทดสอบด้วย

ผลการทดสอบพบว่าผู้ทดสอบทั้ง 3 กลุ่มสามารถพัฒนามอดูลโดยใช้มอดูลเจนเนอเรเตอร์ได้สำเร็จทั้งหมด ทั้งนี้ผู้ทดสอบกลุ่มที่ 3 ที่ไม่เคยมีประสบการณ์กับโอดูนั้นใช้เวลาในการสร้างมอดูลนานที่สุด 3,558 วินาที (59 นาที 18 วินาที) ดังภาพที่ 27



ภาพที่ 27 กราฟแสดงเวลารวมเฉลี่ยในการพัฒนามอดูลของผู้ทดสอบแต่ละกลุ่ม

หลังจากการทดสอบเสร็จสิ้นพบว่าผู้ทดสอบทั้ง 3 กลุ่มนั้นสามารถพัฒนามอดูลได้ในเวลาระหว่าง 2,364 วินาที (39 นาที 24 วินาที) ถึง 3,558 วินาที (59 นาที 18 วินาที) อย่างไรก็ตามผู้ทดสอบกลุ่มที่ 3 ใช้เวลาในการสร้างโมเดลในช่วงแรกนานกว่ากลุ่มผู้ทดสอบอื่นเนื่องจากไม่เข้าใจความสัมพันธ์ของข้อมูล (data relationship) ซึ่งเวลาเฉลี่ยที่ผู้ทดสอบแต่ละกลุ่มใช้ในการสร้างโมเดลแต่ละโมเดลนั้นสามารถดูได้จากภาพที่ 28

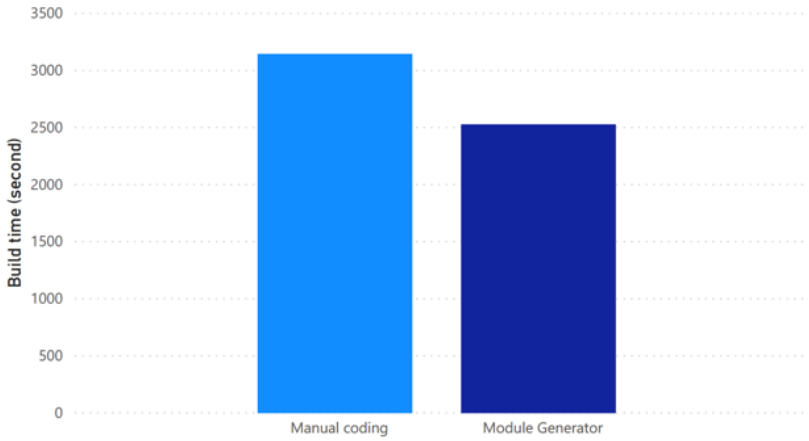


ภาพที่ 28 เวลาเฉลี่ยที่ใช้สร้างแยกตามโมเดลและกลุ่มผู้ทดสอบ

จากภาพที่ 28 แสดงให้เห็นว่าผู้ทดสอบกลุ่มที่ 3 นั้นใช้เวลาในการสร้างโมเดลที่ 1 และโมเดลที่ 2 เป็นเวลานาน ซึ่งเกิดจากความไม่เข้าใจถึงการสร้างโมเดลในส่วนของฟิลด์ที่มีความสัมพันธ์ระหว่างข้อมูล อย่างไรก็ตามในช่วงแรกของการทดสอบผู้ทดสอบในกลุ่มนี้ต้องใช้เวลาในการแก้ไขการสร้างโมเดลจนไม่มีข้อความแจ้งเตือนข้อผิดพลาดจากมอดูลเจนเนอเรเตอร์ จึงสามารถสร้างโมเดลถัดไปได้โดยไม่พบกับปัญหา และใช้เวลาในการสร้างได้ใกล้เคียงกับผู้ทดสอบในกลุ่มอื่น ๆ โดยผู้ทดสอบทั้ง 3 กลุ่มต่างใช้เวลาในการสร้างโมเดลแรกเป็นเวลานานกว่าโมเดลอื่น ๆ เนื่องจากจำนวนฟิลด์ข้อมูลที่มีจำนวนมาก อย่างไรก็ตามผู้ทดสอบทั้ง 3 กลุ่มสามารถพัฒนามอดูลและสามารถนำมาติดตั้งและใช้งานได้จริง

จากการทดสอบดังกล่าวสามารถบอกได้ว่า มอดูลเจนเนอเรเตอร์นั้นสามารถใช้งานได้แม้แต่ผู้ที่ไม่มีประสบการณ์กับโอดู ดังนั้นมอดูลเจนเนอเรเตอร์นี้อาจช่วยให้ผู้พัฒนาสามารถพัฒนามอดูลที่มีโมเดลจำนวนมากได้โดยไม่ต้องกังวลกับความซับซ้อนของการเขียนรหัสต้นฉบับ

การเปรียบเทียบเวลาการพัฒนามอดูลระหว่างการเขียนรหัสต้นฉบับแบบปกติ (manual coding) และการใช้มอดูลเจนเนอเรเตอร์ โดยพิจารณาจากเวลารวมเฉลี่ยที่ใช้ในการพัฒนามอดูลของผู้ทดสอบกลุ่มที่ 1 ผู้มีประสบการณ์พัฒนามอดูลบนโอดูมีผลลัพธ์ดังแสดงในภาพที่ 29



ภาพที่ 29 เวลารวมเฉลี่ยของการพัฒนามอดูลด้วยวิธีปกติและการใช้มอดูลเจนเนอเรเตอร์

จากภาพที่ 29 ผู้ทดสอบกลุ่มที่ 1 สามารถพัฒนามอดูลด้วยวิธีปกติใช้เวลาเฉลี่ย 3,120 วินาที (52 นาที) มากกว่าการพัฒนาโดยใช้มอดูลเจนเนอเรเตอร์ที่ใช้เวลา 2,520 วินาที (42 นาที) ซึ่งเร็วกว่าแบบปกติถึง 20% สาเหตุจากส่วนประกอบของมอดูลที่ใช้กำหนดมุมมองต่าง ๆ (วิว) ที่ต้องเขียนรหัสต้นฉบับจำนวนมากถึงแม้จะมีจำนวนของฟิลด์ภายในโมเดลน้อยก็ตาม อีกทั้งยังต้องให้ความสำคัญกับการเขียนรหัสต้นฉบับให้ถูกต้อง ในทางตรงกันข้าม

มอดูลเจนเนอเรเตอร์สามารถดึงข้อมูลตั้งต้นในส่วนของโมเดลมาสร้างเป็นรหัสต้นฉบับในส่วนของวิวได้ทันที

อย่างไรก็ตามการเปรียบเทียบเวลาในลักษณะนี้พิจารณาเฉพาะผู้ทดสอบกลุ่มที่ 1 เท่านั้น เนื่องจากผู้ทดสอบในกลุ่มอื่น ๆ นั้นไม่สามารถพัฒนามอดูลด้วยการเขียนรหัสต้นฉบับแบบปกติได้ นอกเหนือจากนี้เวลาในการพัฒนามอดูลด้วยมอดูลเจนเนอเรเตอร์อาจลดลงได้มากขึ้น หากผู้ทดสอบมีความคุ้นชินกับการใช้งานมอดูลเจนเนอเรเตอร์ อีกทั้งความคิดเห็นของผู้ทดสอบกลุ่มในนี้ต่างเห็นตรงกันว่า มอดูลเจนเนอเรเตอร์นี้สามารถช่วยให้การพัฒนามอดูลมีความสะดวกสบายมากยิ่งขึ้น เนื่องจากสามารถใส่ข้อมูลของโมเดลเพียงครั้งเดียวและเครื่องมือนี้สามารถสร้างรหัสต้นฉบับของมอดูล ที่ประกอบด้วย โมเดล สิทธิ์การเข้าถึง และวิว ที่สามารถนำไปใช้งานจริงได้ในทันที

4.3.2 การทดสอบการใช้งานโดยอาสาสมัครภายนอก

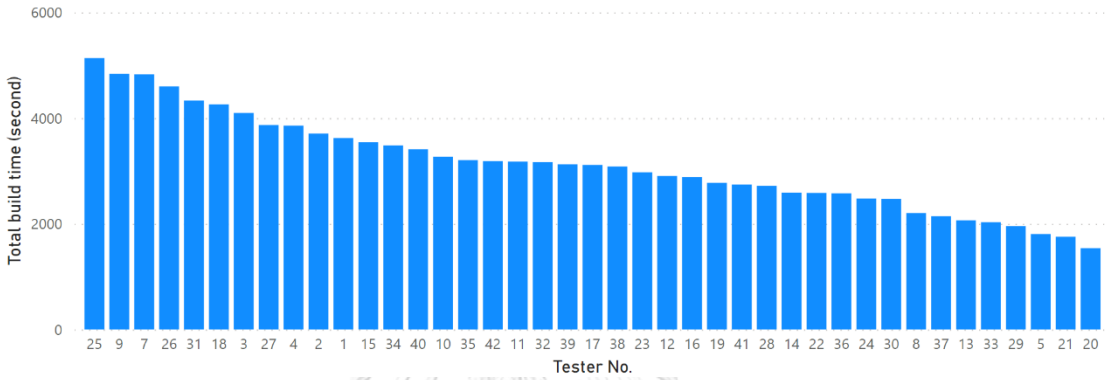
การทดสอบการใช้งานโดยอาสาสมัครภายนอกจำนวน 41 คน เป็นนิสิตระดับปริญญาตรีชั้นปีที่ 3 และ 4 สาขาวิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย โดยมอดูลเจนเนอเรเตอร์จะถูกติดตั้งไว้ยังเครื่องคอมพิวเตอร์แม่ข่ายและให้อาสาสมัครทำการทดสอบที่เครื่องคอมพิวเตอร์ของตนเองผ่านเว็บเบราว์เซอร์ ทั้งนี้อาสาสมัครจะมีการติดตั้งโปรแกรมสำหรับใช้บันทึกภาพหน้าจอตลอดระยะเวลาที่ทำการทดสอบ ในขณะเดียวกันผู้วิจัยจะควบคุมการทดสอบผ่านช่องทางออนไลน์โดยใช้โปรแกรมซูม (Zoom)

ในการทดสอบจะให้อาสาสมัครดำเนินการดังนี้ 1) ทำการทดสอบตามที่เอกสารกำหนด 2) ทำแบบสอบถามเพื่อประเมินตนเองและความคิดเห็นจากการทดสอบ ซึ่งจากการทำแบบสอบถามและประเมินตนเองสามารถสรุปคุณสมบัติของผู้ทดสอบทั้งหมดได้ดังนี้

1. เป็นผู้ที่ไม่เคยมีประสบการณ์กับโอดู (Non-Odoo user) และแยกเป็นผู้ที่เคยรู้จักโอดูมาก่อนการทดสอบจำนวน 1 คน และไม่รู้จักโอดูจำนวน 40 คน
2. เป็นผู้ที่มีทักษะในการเขียนโปรแกรมทั่วไปอยู่ในระดับแย่จำนวน 2 คน ระดับพอใช้จำนวน 27 คน และระดับดีจำนวน 12 คน
3. เป็นผู้ที่มีพื้นฐานเรื่องการออกแบบฐานข้อมูลจำนวน 31 คน และไม่มีจำนวน 10 คน

การทดสอบในครั้งนี้จะใช้โครงร่างฐานข้อมูลสำหรับการทดสอบแบบเดิม ซึ่งในส่วนของเอกสารสำหรับการทดสอบ การอธิบาย และการสอนวิธีการใช้งานมอดูลเจนเนอเรเตอร์ก่อนเริ่มการทดสอบ ถูกปรับปรุงและแก้ไขให้มีความละเอียดมากขึ้นกว่าการทดสอบการใช้งานเบื้องต้น

ผลจากการทดสอบการใช้งานมอดูลเจนเนอเรเตอร์ผู้ทดสอบสามารถพัฒนามอดูลได้โดยใช้เวลาน้อยที่สุด 1,538 วินาที (25 นาที) และมากที่สุด 5,138 วินาที (1 ชั่วโมง 25 นาที) ดังแสดงตามภาพที่ 30 โดยมอดูลที่ถูกพัฒนาขึ้นสามารถนำไปติดตั้งและใช้งานจริงได้ทั้งหมด

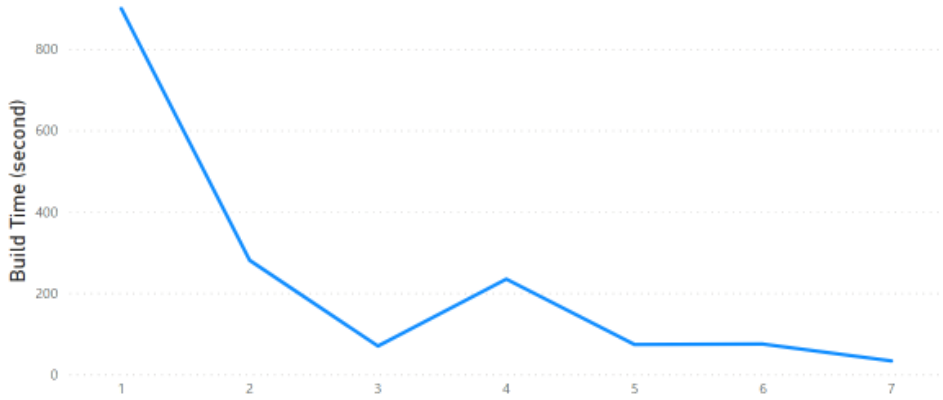


ภาพที่ 30 กราฟแสดงเวลาในพัฒนามอดูลรายบุคคล

จากภาพที่ 30 แสดงให้เห็นว่าเวลารวมที่ใช้ทดสอบมีความแตกต่างกันมากถึง 1 ชั่วโมงซึ่งเกิดจากปัญหาต่าง ๆ ในระหว่างการทดสอบ โดยสามารถสรุปสาเหตุหลักได้ดังต่อไปนี้

1. ผู้ทดสอบกรอกข้อมูลไม่ถูกต้อง ทำให้ไม่สามารถสร้างรหัสต้นฉบับได้
2. ผู้ทดสอบไม่ทราบวิธีการและลำดับขั้นตอนของการทดสอบ เนื่องจากขาดสมาธิในระหว่างการอธิบายและสอนวิธีการใช้งานมอดูลเจนเนอเรเตอร์ก่อนเริ่มการทดสอบ
3. ข้อผิดพลาดเล็กน้อย (bug) ของมอดูลเจนเนอเรเตอร์

จากปัญหาที่กล่าวมาข้างต้นทำให้ผู้ทดสอบทั้ง 41 คนต้องใช้เวลาในการแก้ไขข้อมูล ซึ่งเวลาการแก้ไขข้อมูลเฉลี่ยจากผู้ทดสอบทุกคนคือ 1,060 วินาที (17 นาที) อย่างไรก็ตามหากดูจากเวลาเฉลี่ยที่ผู้ทดสอบสามารถสร้างโมเดลซึ่งเป็นส่วนประกอบสำคัญของการพัฒนามอดูลสามารถทำเวลาเฉลี่ยที่ 3,130 วินาที (52 นาที) และมีเวลาเฉลี่ยในการสร้างโมเดลแต่ละโมเดลแสดงดังภาพที่ 31



ภาพที่ 31 เวลาเฉลี่ยในการสร้างรายโมเดลของอาสาสมัครภายนอก

จากภาพที่ 31 แสดงให้เห็นว่าเวลาเฉลี่ยจากการสร้างโมเดลนั้นมีความใกล้เคียงกับผู้ทดสอบกลุ่มที่ 1 ในการทดสอบเบื้องต้น ซึ่งกล่าวได้ว่าการใช้งานมอดูลเจนเนอเรเตอร์สามารถทำให้ผู้ทดสอบที่เป็นอาสาสมัครภายนอกทั้ง 41 คนสามารถสร้างโมเดลที่เป็นส่วนประกอบหลักของมอดูลโดยใช้เวลาใกล้เคียงกับผู้ทดสอบกลุ่มที่ 1 ให้ออกมาใช้งานจริงได้ อย่างไรก็ตามการใช้งานมอดูลเจนเนอเรเตอร์มีความจำเป็นต้องใช้ความเข้าใจในส่วนประกอบและวิธีการพัฒนามอดูล เพื่อให้การใส่ข้อมูลในมอดูลเจนเนอเรเตอร์นั้นไม่เกิดความผิดพลาดระหว่างการใช้งาน

4.4 การนำมอดูลเจนเนอเรเตอร์มาใช้เพื่อการพัฒนาซอฟต์แวร์ที่ใช้งานจริง

หลังจากการทดสอบและปรับปรุงการทำงานของมอดูลเจนเนอเรเตอร์ ทางบริษัทที่เป็นผู้สนับสนุนการวิจัย ได้นำเครื่องมือนี้มาช่วยในการพัฒนามอดูลที่ใช้สำหรับโครงการการพัฒนาซอฟต์แวร์ต้นแบบของบริษัท ประกอบด้วยโมเดลประมาณ 22 โมเดล และมีฟิลด์ข้อมูล ณ ปัจจุบันมากกว่า 177 ฟิลด์ ซึ่งใช้งานโดยผู้พัฒนามอดูลบนโอดูจำนวน 4 คน ที่มีความเชี่ยวชาญในการพัฒนามอดูลแตกต่างกัน (ใกล้เคียงกับผู้ทดสอบกลุ่มที่ 1) โดยได้นำมอดูลเจนเนอเรเตอร์มาใช้ในการพัฒนามอดูล และนำผลลัพธ์ที่ได้จากมอดูลเจนเนอเรเตอร์ที่เป็นรหัสต้นฉบับไปพัฒนาต่อได้ทันทีโดยที่ไม่จำเป็นต้องเขียนรหัสต้นฉบับจำนวนมาก ทำให้ผู้ใช้งานสามารถลดเวลาในการพัฒนามอดูลได้เป็นอย่างมาก และสามารถให้ความสนใจไปกับการพัฒนามอดูลในส่วนที่เป็นตรรกะหรือการประมวลผลได้มากขึ้น

บทที่ 5
สรุปผลการวิจัย

5.1 สรุปผลการวิจัย

งานวิจัยนี้นำเสนอมอดูลเจนเนอเรเตอร์ สำหรับการออกแบบมอดูลและสร้างรหัสต้นฉบับของมอดูลได้โดยอัตโนมัติ มอดูลเจนเนอเรเตอร์นี้ถูกพัฒนาด้วยภาษาไพธอน ภายใต้สภาพแวดล้อมการทำงานของโอดูเฟรมเวิร์ก สามารถทำงานบนเว็บเบราว์เซอร์ มีส่วนต่อประสานผู้ใช้ที่ช่วยอำนวยความสะดวกในการพัฒนามอดูล ผลการทดสอบการใช้งานแสดงให้เห็นว่า ผู้ใช้งานที่ไม่มีประสบการณ์กับโอดูจนถึงผู้พัฒนามอดูลบนโอดูนั้น สามารถใช้งานเครื่องมือนี้ได้โดยใช้เวลาในการสร้างส่วนประกอบต่าง ๆ ใกล้เคียงกัน ซึ่งหากผู้ใช้งานเป็นผู้ที่มีทักษะการเขียนโปรแกรมและประสบการณ์การพัฒนามอดูล จะสามารถสร้างมอดูลด้วยมอดูลเจนเนอเรเตอร์ได้อย่างรวดเร็วยิ่งขึ้น ตามผลการทดลองที่นำเสนอ และจากการเปรียบเทียบเวลาที่ใช้ในการสร้างมอดูลและส่วนประกอบต่าง ๆ จะเห็นได้ว่าเวลาที่ใช้ในการพัฒนามอดูลลดลง 20% เมื่อเทียบกับวิธีการเขียนโปรแกรมแบบปกติ โดยเมื่อนำมาใช้จริงกับโครงการการพัฒนาซอฟต์แวร์ต้นแบบพบว่า ผู้ใช้งานมอดูลเจนเนอเรเตอร์สามารถพัฒนามอดูลที่มีส่วนประกอบจำนวนมากได้อย่างรวดเร็ว ทั้งนี้ผู้ที่สามารถใช้งานเครื่องมือนี้ไม่ได้จำกัดอยู่เฉพาะผู้พัฒนามอดูลเท่านั้น แต่รวมถึงผู้ใช้งานที่ไม่มีประสบการณ์การพัฒนามอดูลสำหรับโอดูก็สามารถใช้งานได้ หากมีเอกสารกำกับเพื่อบ่งบอกวิธีการใช้งาน อย่างไรก็ตามมอดูลเจนเนอเรเตอร์สามารถช่วยเหลือในการพัฒนามอดูลในระยะเริ่มต้นได้อย่างรวดเร็ว และสามารถช่วยให้ผู้ใช้งานทั่วไปให้สามารถพัฒนามอดูลเพื่อใช้งานเองได้โดยไม่ต้องพึ่งผู้พัฒนามอดูลบนโอดู

5.2 แนวทางวิจัยในอนาคต

ข้อจำกัดของมอดูลเจนเนอเรเตอร์นี้คือรหัสต้นฉบับที่สร้างขึ้นนั้นมีเพียงโมเดล วิว และสิทธิ์การเข้าถึงข้อมูลภายในโมเดลเท่านั้น ในอนาคตจะเพิ่มความสามารถในการสร้างรหัสต้นฉบับให้รองรับส่วนประกอบของมอดูลที่มากขึ้น เช่น คุณสมบัติเฉพาะตัวของฟิลด์ข้อมูล การจัดรูปแบบการแสดงในวิว การสร้างกลุ่มผู้ใช้งาน (user group) แบบกำหนดเอง นอกเหนือจากนั้นการแสดงผลบางส่วนยังไม่เป็นมิตรกับผู้ใช้ที่ไม่ใช่ผู้พัฒนามอดูลบนโอดู เช่น ชื่อแบบฟอร์มที่เป็นศัพท์ทางเทคนิคของการพัฒนามอดูล หรือความสัมพันธ์ภายในโมเดลที่ยังไม่มีการตรวจสอบความถูกต้องอย่างละเอียด การพัฒนาเครื่องมือนี้เพิ่มเติมจะมุ่งเน้นไปในการปรับปรุงเครื่องมือให้สามารถช่วยเหลือผู้ใช้งานในการพัฒนามอดูล และขยายความสามารถการทำงานของมอดูลเจนเนอเรเตอร์ เพื่อรองรับการใช้งานให้เหมาะกับผู้ใช้ทั่วไปที่ไม่ใช่ผู้พัฒนามอดูลบนโอดูมากยิ่งขึ้น

ภาคผนวก

ภาคผนวก ก

พจนานุกรมข้อมูล (data dictionary) ของมอดูลเจนเนอเรเตอร์

1. generator_module

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	id	Int	หมายเลขอ้างอิงรายการมอดูล
2	name	Char	ชื่อมดูล
3	summary	Char	คำอธิบายมอดูล
4	description	Char	รายละเอียดของมอดูล
5	author	Char	ผู้เขียน
6	website	Char	เว็บไซต์
7	category	Char	หมวดหมูของมอดูล
8	version	Char	เวอร์ชัน
9	license	Char	ลิขสิทธิ์การใช้งาน
10	maintainer	Char	ผู้ดูแล/ปรับปรุง
11	application	Bool	เป็นรายการแอปพลิเคชัน
12	installable	Bool	สามารถติดตั้งได้ผ่าน GUI
13	module_directory	Char	เส้นทางการในการสร้างมอดูล
14	module_repository	Char	URL ของสถานที่เก็บรหัสต้นฉบับ
15	schema_file	Binary	ไฟล์สำหรับการออกแบบมอดูล
16	schema_file_name	Char	ชื่อไฟล์สำหรับการออกแบบมอดูล
17	create_uid	Int	รหัสอ้างอิงผู้สร้างข้อมูล
18	create_date	Datetime	วันที่สร้างรายการ
19	write_uid	Int	รหัสอ้างอิงผู้แก้ไขข้อมูลล่าสุด
20	write_date	Datetime	วันที่แก้ไขข้อมูลล่าสุด

2. generator_model

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	id	Int	รหัสอ้างอิงโมเดล
2	name	Char	ชื่อโมเดล
3	technical_name	Char	ชื่อทางเทคนิคของโมเดล
4	model_description	Char	คำอธิบายโมเดล
5	parent_module_id	Int	รหัสอ้างอิงมอดูล
6	generate_public_access	Bool	การสร้างสิทธิ์การเข้าถึงแบบสารธารณะ
7	inherit	Char	ชื่อโมเดลสำหรับสืบทอดโครงสร้างโมเดล
8	sequence	Int	ลำดับรายการข้อมูล
9	create_uid	Int	รหัสอ้างอิงผู้สร้างข้อมูล
10	create_date	Datetime	วันที่สร้างรายการ
11	write_uid	Int	รหัสอ้างอิงผู้แก้ไขข้อมูลล่าสุด
12	write_date	Datetime	วันที่แก้ไขข้อมูลล่าสุด

3. generator_field

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	id	Int	รหัสอ้างอิงฟิลด์
2	name	Char	ชื่อฟิลด์ที่แสดงบนมุมมองผู้ใช้
3	field_name	Char	ชื่อฟิลด์
4	parent_model_id	Int	รหัสอ้างอิงโมเดล
5	data_type	Char	รูปแบบข้อมูล
6	sequence	Int	ลำดับข้อมูล
7	options	Char	ตัวเลือกข้อมูลสำหรับฟิลด์แบบซีเล็คชัน
8	on_delete	Char	ตัวเลือกเมื่อทำการลบข้อมูลความสัมพันธ์
9	record_name	Bool	ใช้เป็นชื่อข้อมูล
10	index	Bool	ใช้เป็นดัชนีข้อมูล
11	required	Bool	ข้อมูลที่ต้องการ (บังคับ)
12	default	Char	ค่ามาตรฐาน

13	default_timestamp	Bool	ค่ามาตรฐานแบบวันที่และเวลา
14	help_text	Char	คำอธิบายฟิลด์
15	relation_model	Char	ชื่อโมเดลที่มีความสัมพันธ์
16	relation_field	Char	ชื่อฟิลด์ที่มีความสัมพันธ์
17	create_uid	Char	รหัสอ้างอิงผู้สร้างข้อมูล
18	create_date	Selection	วันที่สร้างรายการ
19	write_uid	Char	รหัสอ้างอิงผู้แก้ไขข้อมูลล่าสุด
20	write_date	Char	วันที่แก้ไขข้อมูลล่าสุด

4. generator_model_access

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	id	Char	รหัสอ้างอิงสิทธิ์การเข้าถึงโมเดล
2	name	Char	ชื่อสิทธิ์การเข้าถึง
3	group	Int	รหัสอ้างอิงกลุ่มผู้ใช้งาน
4	model_id	Int	รหัสอ้างอิงโมเดล
5	perm_create	Bool	สิทธิ์ในการสร้างข้อมูล
6	perm_read	Bool	สิทธิ์ในการอ่านข้อมูล
7	perm_write	Bool	สิทธิ์ในการแก้ไขข้อมูล
8	perm_delete	Bool	สิทธิ์ในการลบข้อมูล
9	parent_module_id	Int	รหัสอ้างอิงมอดูล
10	create_uid	Char	รหัสอ้างอิงผู้สร้างข้อมูล
11	create_date	Selection	วันที่สร้างรายการ
12	write_uid	Char	รหัสอ้างอิงผู้แก้ไขข้อมูลล่าสุด
13	write_date	Char	วันที่แก้ไขข้อมูลล่าสุด

5. generator_menu

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	id	Char	รหัสอ้างอิงเมนู
2	name	Char	ชื่อเมนู
3	model_id	Int	รหัสอ้างอิงโมเดล
4	parent_menu	Char	ชื่อเมนูหลัก
5	parent_module_id	Int	รหัสอ้างอิงเมนูหลัก
6	sequence	Int	ลำดับการแสดงเมนู
7	create_uid	Char	รหัสอ้างอิงผู้สร้างข้อมูล
8	create_date	Selection	วันที่สร้างรายการ
9	write_uid	Char	รหัสอ้างอิงผู้แก้ไขข้อมูลล่าสุด
10	write_date	Char	วันที่แก้ไขข้อมูลล่าสุด

ภาคผนวก ข

พจนานุกรมข้อมูล (data dictionary) ของระบบบันทึกข้อมูลผู้ได้รับความเดือดร้อน จากผลกระทบของโรคโควิด 19

1. mhesi_applicants

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	citizen_id	Char	หมายเลขบัตรประชาชน
2	first_name	Char	ชื่อจริง
3	last_name	Char	นามสกุล
4	full_name	Char	ชื่อเต็ม
5	institution	Many2one	สังกัดหน่วยจ้างงาน
6	state	Selection	สถานการณ์จ้างงาน (ลงทะเบียน/จ้างงาน/สิ้นสุด)
7	period	Integer	ระยะของโครงการที่เข้าสมัคร
8	address	Text	ที่อยู่
9	tel	Char	เบอร์โทรศัพท์
10	tambon	Many2one	ตำบล
11	mobile	Char	เบอร์มือถือ
12	email	Char	อีเมล
13	birth_date	Date	วันที่เกิด
14	race	Char	เชื้อชาติ
15	nationality	Char	สัญชาติ
16	religion	Char	ศาสนา
17	domicile	Char	ถิ่นที่อยู่
18	is_help	Selection	ได้รับเงินเยียวยาจากรัฐบาล (ได้/ไม่ได้)
19	covid_effect	Char	ผลกระทบจากสถานการโควิด
20	expect_job	Char	อาชีพที่คาดหวังในอนาคต
21	exp	Char	ประสบการณ์ทำงาน
22	special	Char	ทักษะพิเศษ
23	expect_speciality	Char	ทักษะพิเศษที่ต้องการเพิ่มเติม

24	expect_entrepreneur	Char	สิ่งที่ต้องการทำหากเป็นผู้ประกอบการ
25	responsibilities	Many2one	ภาระหน้าที่
26	is_duplicate	Boolean	ตรวจพบการลงทะเบียนซ้ำซ้อน

2. mhesi_institutions

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	name	Char	ชื่อหน่วยจ้างงาน
2	address	Char	ที่อยู่
3	tambon	Many2One	ตำบล
4	workloads	One2many	รายชื่อภาระหน้าที่
5	proceed_area	Char	พื้นที่กำเนินการ
6	academic_requirement	Char	วุฒิการศึกษาที่เปิดรับ
7	registration_channel	Char	ช่องทางการรับสมัคร
8	social_security	Boolean	สถานะการจ่ายเงินประกันสังคม
9	project_phase_1	Boolean	เข้าร่วมโครงการระยะที่ 1
10	project_phase_2	Boolean	เข้าร่วมโครงการระยะที่ 2

3. mhesi_workloads

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	name	Char	ชื่อภาระหน้าที่
2	institution	Many2One	หน่วยจ้างงาน

4. mhesi_responsibilities

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	applicant_id	Many2One	ผู้ถูกจ้าง
2	result	Selection	ผลลัพธ์การทำงาน (ต่ำกว่าแผน/ตามแผน/สูงกว่าแผน)
3	remarks	Char	หมายเหตุ

4	workload	Many2one	ภาระหน้าที่
5	work_area	Char	บริเวณที่รับผิดชอบ
6	work_area_tambon	Many2one	บริเวณที่รับผิดชอบ (ตำบล)
7	file_name	Binary	ไฟล์รายงานผลการปฏิบัติงาน

5. mhesi_cds_tambon

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	name	Char	ชื่อตำบล
2	amphur	Many2One	อำเภอ
3	zip_code	Char	รหัสไปรษณีย์

6. mhesi_cds_amphur

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	name	Char	ชื่ออำเภอ
2	province	Many2one	จังหวัด

7. mhesi_cds_province

ลำดับที่	ชื่อข้อมูล	ประเภทข้อมูล	คำอธิบาย
1	name	Char	ชื่อจังหวัด

ภาคผนวก ค

แบบประเมินตนเองของอาสาสมัครผู้ทดสอบใช้งานมอดูลเจนเนอเรเตอร์

1. แบบประเมินตนเองของอาสาสมัครผู้ทดสอบใช้งานมอดูลเจนเนอเรเตอร์

ลำดับที่	คำถาม	ตัวเลือกคำตอบ
1	คุณรู้จักระบบโอดูมาก่อนหรือไม่	ใช่/ไม่ใช่
2	ทักษะในการเขียนโปรแกรมของคุณอยู่ในระดับใด	แย่/พอใช้/ดี/ดีมาก
3	คุณมีทักษะเรื่องการออกแบบฐานข้อมูลหรือไม่	มี/ไม่มี
4	คุณเคยใช้งานฐานข้อมูลประเภทใดมาบ้าง	RDBMS/NoSQL/ไม่ทราบ
5	คุณรู้จักโมเดลสำหรับการเขียนโปรแกรมแบบ MVC หรือไม่	รู้จัก/ไม่รู้จัก
6	คุณได้ศึกษาวิธีการพัฒนามอดูลด้วยวิธีการเขียนโค้ดแบบปกติมาบ้างหรือไม่	ใช่/ไม่ใช่

ภาคผนวก ง

ความคิดเห็นเกี่ยวกับการทดสอบของผู้ทดสอบใช้งานมอดูลเจนเนอเรเตอร์

1. ความคิดเห็นของผู้ทดสอบใช้งานมอดูลเจนเนอเรเตอร์

ลำดับที่	ความคิดเห็น
1	อยากให้เพิ่มการย้ายช่องโดยการกด tab
2	อยากให้ผู้พัฒนา ปรับเรื่องหน้าต่างการกรอกข้อมูล เนื่องจาก ทำให้สับสนว่ากรอกข้อมูลแล้วหรือยัง ในส่วนของ save&close ในตอนที่สร้าง field ที่เป็นหน้าต่าง pop-up ขึ้นมาไปตรงกับ save&close ของmodel ทำให้อาจเกิดความสับสนได้ และเนื่องการการที่ต้อง save หลายรอบ บางครั้งผู้ใช้งานอาจเกิดความเคยชินที่กด save หน้าต่างนึงแล้วกดปิดเลย เป็นต้น
3	อยากให้มีหน้าต่างแสดงข้อมูลที่ใส่ไปแล้วขณะที่กรอกข้อมูล
4	อยากให้ปรับเรื่องการบันทึก record name เนื่องจากถ้ากดผิดแล้วจะแก้ไขยากใช้เวลานาน
5	UI เข้าใจง่าย สามารถใช้งานได้โดยอ่านคู่มือเพียงรอบเดียว
6	รู้สึกสับสนในการใช้งานช่วงแรก แต่พอลองทำไปสักพักก็เข้าใจได้ และใช้งานได้ดี
7	จากการใช้งาน พบว่าใช้งานค่อนข้างง่าย เนื่องจาก UI ค่อนข้างดูสะอาดตา ส่วนที่ผมชอบมากที่สุดคือ ปุ่ม Save & New เพราะทำให้เราไม่ต้องเสียเวลาในการกด create ซ้ำ ๆ ซึ่งลดขั้นตอนในการ create ไปได้มาก ผมมีข้อเสนอแนะตรงที่ ACIs หรือการกำหนด permission ในส่วนของ checkbox มีระยะห่างออกจากกันค่อนข้างมาก ทำให้ต้องใช้เวลาในการเลื่อนเมาส์ไปยังช่อง checkbox จึงอยากให้ระยะห่างระหว่าง checkbox ลดลงครับ ขอบคุณครับ
8	อยากให้เพิ่มคำอธิบายแต่ละช่อง จะทำให้ใช้งานง่ายขึ้น
9	อยากให้เพิ่มระบบ save อัตโนมัติ
10	อยากให้ตัวอักษรที่เตือนยังไม่ค่อยเด่นชัดหากมองผ่านๆอาจจะไม่เป็นจุดโฟกัสมากนัก อยากให้ตรงเมนู ACLs ในส่วนของช่อง model กับ เมนู Menu ในส่วนของช่อง model เมื่อเลือกอันไหนไปแล้วอยากให้อันที่เลือกไปแล้วไม่ปรากฏขึ้นมาเลยเพราะอาจทำให้เกิดการกดผิดพลาดได้
11	อยากให้มีการถามว่าจะ record table หรือไม่ ในการทำ field แรก และ ปุ่ม select all ในACLs ก็จะทำให้ใช้งานสะดวกขึ้น

12	คิดว่าควรมีความรู้เบื้องต้นในการใช้โปรแกรมนี้ จะทำให้ใช้งานได้ง่ายมากขึ้น
13	โดยภาพรวมถือว่าใช้งานได้ง่ายมีคำอธิบายของปุ่มกดเข้าใจง่ายมีการแจ้งเตือนที่ค่อนข้างดี เมื่อลิมติ้ก primary key
14	อยากให้เพิ่มการ confirm discard
15	อยากให้การแจ้งเตือน ใช้ภาษาที่เข้าใจง่ายมากขึ้น
16	อยากให้ทำ Dropbox ในขั้นตอนการเชื่อมความสัมพันธ์แบบ Many2One เพื่อป้องกัน user พิมพ์ผิด
17	สับสนเรื่องการกรอกข้อมูลเกี่ยวกับ Many2One กับ One2many
18	พบปัญหาในการแก้ไขข้อมูลใน model แล้วข้อมูล ACLs หาย ทำให้ต้องบันทึกใหม่

ภาคผนวก จ

ผลงานการตีพิมพ์ในงานประชุมวิชาการ

2021 IEEE 8th International Conference on Industrial Engineering and Applications

Developing Module Generation for Odoo Using Concept of Low-Code Development Platform and Automation Systems

Sopanawit Pichidtienthum
Program of Science for Industry, Faculty of Science
Chulalongkorn University,
Bangkok, Thailand
e-mail: 6270223023@student.chula.ac.th

Pakawan Pugsee, Nagul Cooharojananone
Department of Mathematics and Computer Science,
Faculty of Science, Chulalongkorn University,
Bangkok, Thailand
e-mail: pakawan.p@chula.ac.th, nagul.c@chula.ac.th

2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA) | 978-1-6654-2895-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIEA52957.2021.9436754

***Abstract*— In this study, a module generator for Odoo using a low-code development concept was proposed. Odoo is an open-source ERP software that brings together necessary modules for various business management and a Python web framework which allows developers to develop a module to extend the capability of Odoo. In recent years, software development on Odoo usually requires time to learn due to the complexity of the framework. Therefore, a platform developed using low-code development - a concept of software creation with less code writing - was proposed. This platform is a web-based application tool that generates codes for developers to assist them in creating modules with less concern about errors and more focus on the logical side of programming. Developers can also use this application as a tool to validate and generate a source code for a ready-to-use module. This application was tested with non-Odoo users, Odoo users, and Odoo developers with various models containing various fields. Time efficiency between three types of users was compared, as well as time efficiency between generating a module using the conventional method and using the low-code concept module generator. The result revealed that using this application could reduce time by 20% on average.**

Keywords-automation; low-code development platform; code generation; automation; odoo

I. INTRODUCTION

Odoo is an open source business management software that brings together both structure and work details. In comparison, Odoo is well established over the competition due to its standardized system structure including a full range of business management modules [1]. It is also open to the development of other modules that an organization might need using the editable software modules in addition to the default provided modules. As an open software, it features customization of operation or freedom of use, thus making it possible to use an in-house developer to develop or customize software. With its flexibility, Odoo allows users more versatility in usage than simply being a business management application. For example, Odoo can be used as a back-office program, mobile software, or website software. By using Odoo, developers can reduce the time spent on structure development, access-control list, and displays which helps prioritizing time to other areas of development. However, due to the complexity of its function, software development using Odoo takes some time to learn and requires a handful of codes to generate the required modules. This article will explore the development of Odoo's modules that allow developers easier and faster usage by allowing them to start developing modules through the display screen [2] and automatically generating codes as a foundation for developers to work on.

II. BACKGROUND KNOWLEDGE AND RELATED WORK

A. Low-Code Development Platform

The low-code development platform is a process tool for speeding up software development as it minimizes manual programming such as program design and behavior of programs through the display screen, which helps developers reduce the structural design process [3]. Users will not be limited by their knowledge and expertise in programming languages, thus making it possible to bring the program for further development or use promptly [2]. When developers design programs using this tool, it enables them to focus on developing important parts of the program. In general, tools have some basic features that can help reduce coding [4], for example, developers can design programs by using the user interface to set properties and run the process through a form or via drag-and-drop. This sequence of actions can be performed on the screen, allowing usage of a developer that might be those with no programming experience. Some tools may also have the ability to connect to data from different databases or sources, so that processes or program decision designs can be created. However, these tools might not suitable for use in all cases. For example, some tools cannot be used to develop complex programs, or might not be further developed with additional codes due to being limited to a custom-made language. The use of these tools requires consideration on the usage context of developers, for them be appropriately designed before being released.

B. Odoo Development

Nowadays, Odoo developers still need to be hired at a fairly high price to develop the Odoo system for it to work according to a business's logic. However, there is also Odoo on Software as a Service (SaaS) that provides tools to assist developers in module creation.

A module is a set of data and/or logic written in MVC model that may contain an object of data and logical processes (Models), user interface (Views), access-control

list (Security), and other static files for a graphical screen. The module may also have dependencies with another modules if it requires other logical processes or capabilities from them. However, these components do not need to be contained in the module, which may have only one or more components [5].

In general, to develop a module for Odoo, the program's command-lines can be used to help create files within the module with only a few code snippets. After that, developers have to write their own codes to develop the module using the steps below.

- Enter the module directory.
- Create Models by making a Python file and inherit class of API and model from Odoo, and then define data objects according to their data structure. This will create ORM API. (Object Relational Mapping Application Programming Interface).
- Create Views by making an XML file, define a record that should be displayed according to its model.
- Create Security by making a CSV file to define who can access the Models.
- Create manifest.py in the root directory of module to define module information.
- Define the module directory in an Odoo server setting and restart the server, then the module is ready-to-go.

The steps above show that developers have to make at least 3 types of files as specified by Odoo, which takes a lot of time to learn. Odoo Community has a code generation tool for modules building, resulting in several modules published by numerous developers. Unfortunately, most of them were not updated for a while and cannot be used with the latest version of Odoo. Outdated versions of tools do neither provide control nor notify when a developer creates wrong attributes or properties. This can cause errors on a module after being installed that the developers will have to debug the source code by themselves.

Another option is to use Odoo Studio, a module offered by Odoo on SaaS as a tool to create a module via a user interface [6], with the highlight being that developers can use drag-and-drop to create a visual interface of forms within a module as needed and define various properties in detail. However, using Odoo Studio is limited, as it is a paid tool and cannot generate source code for further development.

The development method should be neither manually input nor using a code generation tool, as both lead to an issue that when errors occur, the developers have to find and fix them individually. This problem can occur even with experienced developers, especially in manual coding, as developers have to spend more to code.

III. The Proposed Module Generator

From the previous studies mentioned in section II, the objective is to assist developers coding a module (a program) on Odoo with less time required for learning and with less concern about the complexity of its structure.

After developing for various modules, several parts of the module consist of a clear and predictable pattern of codes, which no matter a developer does in coding, they tend to eventually resort to a similar coding style. For example, for coding most models in a Python file, developers have to define their names according to module naming and model naming, using dots to concatenate them together, and then an ORM is defined by a built-in model class.

It was found that several tools were found to be able to assist developers in coding in several parts as mentioned earlier in the concept of low-code development platform. Therefore, a tool named Module Generator was designed to easily update code patterns or code styles of a module to support when an Odoo version is updated. This tool aimed to help reduce module creation complexity and to assist developers in focusing more on the logical side of code generating as much as possible.

A. Introduction of Module Generator

Module Generator is a tool that allows developers to define the various elements needed for creating new modules and obtain results for the module's directory and source code that are similar to manual coding.

In this first version of development, design of Module Generator focuses on helping developers to be able to easily create modules ready for immediate use. It can also be further developed using a user interface to provide the elements and options needed in creating modules and minimizing complex details [7]. Parts of a module that can be generated with this version are listed below;

- manifest.py, the file that provide a module meta data.
- Models, defining data objects as ORM API.
- Views, defining displays for Models.
- Security, defining access-control list for Models.

B. Architecture of Module Generator

To use this tool to automatically create a module, it requires reading of data provided by the developer through the user interface. For ease of development, this tool is designed as a module that can be installed and works on Odoo systems

A module for module creating can be designed to store information that is necessary in creating of a module. This includes the module's metadata, the model to be built, the fields to store the data in that model, as well as the form definitions for display and access to that data. However, the dataset for building the module will be issued to minimize the data entry process. This helps reduce development time and the time required to learn about its operation.

Within a module, there is a set of programs to read data for building a module. It generates source codes and outputs as a module directory as shown in Figure 1. The program is designed to be able to generate source code independently of the user input as the module contains independent components such as Controllers, Models, Views, Security, etc. However, the building blocks do not require all of the above components.

When the program has generated the source code. It is organized into a standard module directory for creating a module (reference an Odoo web) and the source code being generated is similar to manual coding. This enables the developer to use the source code for further development.

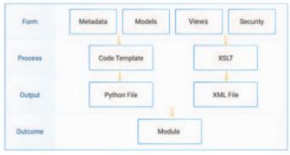


Figure 1. Module Generator's Architecture

C. *Generator Process*

After designing the desired module structure through the user interface, the program will then read the user-defined information to generate the code and save it as a file, arranged according to the standard structure of the module making as shown in Figure 2.

Figure 2. Generator Process

The code generated by the program consists of three file formats: (1) Python file (2) XML file and (3) CSV file. There are two building processes involved:

- A code template is a pre-built source code that contains variables based on various properties of the dataset, such as a Python file that uses the information that the developer specifies according to a variable above to create a ready-to-use file.
- Creating an XML file using XSLT (Extensible Stylesheet Language Transformations) is a language that transforms data from one XML format into another [8-9], whereby the process will arrange the data that the developer defined into categories based on the data hierarchy. In the form of XML, XSLT is then used to transform it into the Odoo format.

In Figure 3, a simple workflow shows how the module is being generated.

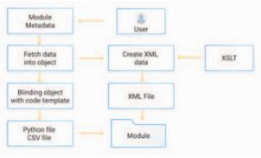


Figure 3. Generator Process Flow

D. *Requirements*

To use Module Generator, it is necessary for users to have a background in developing an Odoo system. This is because it requires basic usage knowledge of the program, such starting the server, restarting the server, or assigning add-on directories to Odoo. After a new module is generated, it can be implemented by assigning an add-on directory to the desired module and restarting the server, so that the existing Odoo system can recognize the new module.

E. *User Interface*

Defining the metadata to create a module will reference the development of modules by manual coding. Therefore, the components are similar to those developed by conventional methods, allowing users to learn how to operate them easily and quickly.

In order to build a module, a developer must define some basic module properties as shown in Figure 4, which is necessary information for building every module.

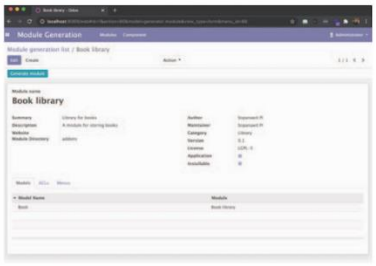


Figure 4. A form for defining a manifest object of module

The users can then define the different parts of the module in mainly 3 menus which are Models to create models and their fields, ACLs to create security, and Menus to create view menu. The data that can be defined in this form includes only basic information. With the complex and logical parts of the module, developers can write further code afterward. Because of this complexity, the user interface might contain components that are difficult to understand and use. Figure 5 shows how little information for creating a model.

Figure 5. A form for defining an object of models

To create an object field, it can be created under each model where developers can define the field's data type and data relationship between models as shown in Figure 6.

Figure 6. A form for defining an object of fields

The function of Module Generator to create Models will automatically generate Views when menu is created and referenced to Models as shown in Figure 7.

Figure 7. A form for defining a menu for model's view

The last part of building a module is to set an access-control list. The program will use the ACLs menu to create a security file for each model in the module as shown in Figure 8.

Figure 8. A form for defining a model's security

IV. Testing Result And Discussion

The purpose of this test is to fundamentally test the functionality of the Module Generator with the goal to create models and views to assist Odoo developers in module creating. The timing method was used to compare the time required to build the module between the use of the normal method and the use of the module generator. The module generated in this test was based on a data structure from the project that aids sufferers from COVID-19 in Thailand by the Ministry of Higher Education, Science, Research and Innovation (MHESI). This data structure was previously created by a module generator and is currently being used in the real situation.

The data structure used for testing consists of 7 models, with a total of 51 data fields. These data relationships between models of sorted data and a number of fields, which are used to record data for this test, are shown in TABLE I. Besides, this data structure has various data types used in the real software project, including by usage in a general software development project, so it might be a good sample for this test.

TABLE I. Number of Field in Each Model

Model No.	No. of fields	Number of unique data types
1	26	6
2	10	4
3	2	2
4	7	4
5	3	2
6	2	2
7	1	1

In the fundamental test, three Odoo developers were invited to be involved, and with the curiosity of whether other user groups would be able to use it or not, an Odoo user and a non-Odoo user were also invited to be involved in this test as follows:

- Three similar-skilled Odoo developers (A)
- An Odoo user with no Odoo development skill (B)
- A Non-Odoo user with no Odoo development skill (C)

Testers were briefly explained how to use the Module Generator, then were given a module document for building the module by themselves. When 3 groups of users (5 individuals) had finished creating a module using the Module Generator, it was found that all groups were able to successfully use the program. However, considering the average module building time, group C was found to take a longer time than the other groups, as shown in Figure 9.

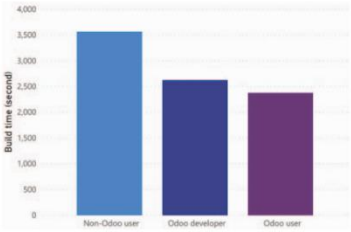


Figure 9. Graph showing average time to create module

After the modules had been created, it was shown that all testers could create the module within approximately 45-60 minutes. However, the general user who did not know much about Odoo was found to take a lot of time to understand data relations between models. In this test the time taken while creating a model is shown in Figure 10.

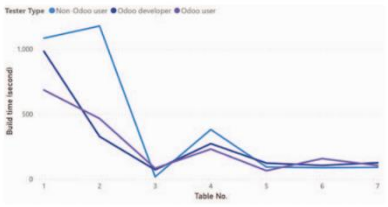


Figure 10. Graph showing average time to create models

As in Figure 10, the general user was confused about data types and how to create data relationships while reading the test document. Consequently, the general user did a trial and error until no errors appeared in the Module Generator. The tester could then create the next models without error and take a time similar to the others. In Model No. 1 and Model No. 2, every tester took a lot of time to create a model because there are a lot of fields of data and the testers except Odoo developers were confused about data relationships. Nonetheless, every module created by the testers was installed to Odoo and usable.

This shows that Module Generator can be used by developers and even "a user", thus it might help Odoo developers to create modules which have a lot of Models or Views without warrying about errors from manual coding.

Next, the comparison of the average building time between creating a module by manual coding and by using Module Generator is shown in Figure 11.

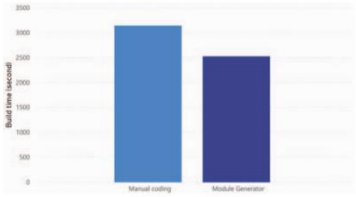


Figure 11. Comparison between creating a module by manual coding and by using tool

According to Figure 11, when Odoo developers created a test module, the average building time in a conventional way (manual coding) is higher than that of using Module Generator. Moreover, the taken time using Module Generator could be reduced if the testers become more familiar with the tool. Furthermore, most of testers also commented that this is a convenient tool.

V. Conclusion

From the testing, it can be seen that the time used to create a module was 20% reduced when using the Module Generator compared with the manual coding method. With the obvious difference being usage of a model with fewer data fields, allowing the Module Generator to quickly create the model. Manual coding is even with a few fields, but creating its view still requires a lot of coding, if copying is used to help. This shows that using Module Generator can allow developers to focus on building modules using only data structures. On the contrary, with manual coding, developers have to focus more on writing code correctly - a main reason for taking more time in development.

In the testing, a general user was invited to collaborate on the test. This shows that the tool can also be used by a beginner Odoo user to start building a module for general propose. This should help making Odoo a software that everyone can use to develop software by themselves. Thus, in certain projects, a required number of experienced developers can be reduced, resulting in cost reduction.

Nonetheless, the limitations of this tool are that the generated codes can deploy only data models with views and the user interfaces are not quite user-friendly. Further research will be carried on in improving this tool to be more user-friendly and providing more code generation usability. It should extend usage capability to general users, including those open to testing this tool in the programmer community.

Acknowledgment

This work is supported by Facgure Company Limited, Program of Science for Industry, Faculty of Science Chulalongkorn University and Innovation and technology assistance program (ITAP) under National Science and Technology Development Agency (NSTDA).

References

[1] Ganesh A, Shanil KN, Sunitha C, Midhundas AM. OpenERP/Odoo - An Open Source Concept to ERP Solution. 2016 IEEE 6th International Conference on Advanced Computing (IACC)2016. p. 112-6.

[2] Adrian B., Hinrichsen S., Nikolenko A. (2020) App Development via Low-Code Programming as Part of Modern Industrial Engineering Education. In: Nunes I. (eds) Advances in Human Factors and Systems Interaction. AHFE 2020. Advances in Intelligent Systems and Computing, vol 1207. Springer, Cham. https://doi.org/10.1007/978-3-030-51369-6_7

[3] Waszkowski R. Low-code platform for automating business processes in manufacturing. IFAC-PapersOnLine. 2019;52(10):376-81.

[4] OutSystems. (2019), The State of Application Development [White paper]. https://t4spartners.com/outsystems-whitepaper-2019-2020

[5] Building a Module – odoo 13.0 documentation, November 2020 [Online]. Available: https://www.odoo.com/documentation/13.0/howtos/backend.html

[6] Create and Customize Your Own Apps with Odoo Studio, November 2020 [Online]. Available: https://www.odoo.com/page/studio

[7] A. Tarasiev, M. Filippova, K. Aksyonov and O. Aksyonova, "Developing Prototype of CASE-Tool to Create Automation Systems Based on Web Applications Using Code Generation," 2018 Dynamics of Systems, Mechanisms and Machines (Dynamics), Omsk, 2018, pp. 1-4, doi: 10.1109/Dynamics.2018.8601443.

[8] Musto J, Dahanayake A. Transforming Object-Oriented Model to a Web Interface Using XSLT. Commun Comput Info Sci 2019;1064:221-231.

[9] Groppe, S. G. J. (2008). Output schemas of XSLT stylesheets and their applications. Information Sciences, 178(21), 3989-4018. doi:https://doi.org/10.1016/j.ins.2008.06.02

บรรณานุกรม

1. Ganesh, A., et al. *OpenERP/Odoo - An Open Source Concept to ERP Solution*. in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*. 2016.
2. Sanchis, R., et al., *Low-Code as Enabler of Digital Transformation in Manufacturing Industry.* Applied Sciences, 2019. **10**: p. 12.
3. Hibernate. *What is Object/Relational Mapping?* [Online] November, 2020]; Available from: https://hibernate.org/orm/what-is-an-orm/.
4. Waszkowski, R., *Low-code platform for automating business processes in manufacturing.* IFAC-PapersOnLine, 2019. **52**(10): p. 376-381.
5. OutSystems. *The State of Application Development*. [Online] November, 2020]; Available from: https://t4spartners.com/outsystems-whitepaper-2019-2020/.
6. Odoo. *Create and Customize Your Own Apps with Odoo Studio*. [Online] November, 2020]; Available from: https://www.odoo.com/page/studio.
7. Jounaidi, A. and M. Bahaj. *Designing and implementing XML schema inside OWL ontology*. in *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 2017.
8. Odoo. *Building a Module - odoo 13.0 documentation*. [Online] November, 2020]; Available from: https://www.odoo.com/documentation/13.0/howtos/backend.html.

ประวัติผู้เขียน

ชื่อ-สกุล	โสภณวิชญ์ พิชิตเธียรธรรม
วัน เดือน ปี เกิด	14 พฤศจิกายน 2534
สถานที่เกิด	ชลบุรี
วุฒิการศึกษา	วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์เพื่ออุตสาหกรรม จุฬาลงกรณ์มหาวิทยาลัย
ที่อยู่ปัจจุบัน	5/2 ม.1 ต.บ้านสวน อ.เมืองชลบุรี จ.ชลบุรี 20000
ผลงานตีพิมพ์	Developing Module Generation for Odoo Using Concept of Low-Code Development Platform and Automation Systems, 2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA), 2021, pp. 529-533, doi: 10.1109/ICIEA52957.2021.9436754.