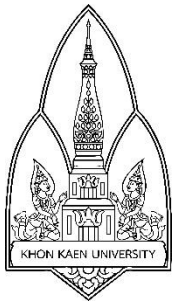


โครงงานเรื่อง

การสร้างฐานข้อมูลและรับส่งข้อมูลแบบ API ด้วย Spring Boot

Create Database and API with Spring Boot

โดย

นางสาวนิภาภรณ์	ขันติกิจ	รหัสนักศึกษา 613020583-5
นางสาวสุภาวี	เกิดแก้ว	รหัสนักศึกษา 613020602-7

อาจารย์ที่ปรึกษา: ผศ.ดร.ปัญญาพล หอระตะ

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา SC313002 หลักการออกแบบพัฒนาซอฟต์แวร์

ภาคเรียนที่ 1 ปีการศึกษา 2563

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยขอนแก่น

(เดือนพฤศจิกายน พ.ศ. 2563)

กิตติกรรมประกาศ

โครงงานเรื่องการสร้างฐานข้อมูลและรับส่งข้อมูลแบบ API ด้วย Spring Boots Framework นี้สำเร็จได้โดยได้ความช่วยเหลืออย่างดียิ่งจาก ผศ.ดร.ปัญญาพล หอระตะ อาจารย์ที่ปรึกษาโครงงาน ผู้สอนรายวิชาหลักการออกแบบพัฒนาซอฟต์แวร์ ที่ได้ให้คำเสนอแนะ แนวคิดและความรู้ในการจัดทำโครงงานจนโครงงานนี้เสร็จสมบูรณ์

ขอขอบคุณ นางสาวดารุณี แสงอุ่นอุไร และนายสุพิจารณ์ วิไลลักษณ์ เพื่อนร่วมชั้นเรียนที่ได้แนะนำ เสนอแนะและให้ความรู้ในการแก้ไขข้อบกพร่องต่าง ๆ ในการทำโครงงานมาโดยตลอด ผู้จัดทำจึงขอขอบคุณเป็นอย่างสูง

ท้ายสุดนี้ผู้จัดทำหวังเป็นอย่างยิ่งว่าโครงงานเรื่องการสร้างฐานข้อมูลและรับส่งข้อมูลแบบ API ด้วย Spring Boot นี้ จะเป็นประโยชน์ต่อการศึกษาค้นคว้า และเป็นประโยชน์ต่อผู้ที่สนใจศึกษาต่อไป

นางสาวนิภาภรณ์ ขันติกิจ

นางสาวสุภาวี เกิดแก้ว

คณะผู้จัดทำ

บทที่ 1

บทนำ

1.1. ความเป็นมาและความสำคัญ

ปัจจุบันมีธุรกิจการค้าขายเกิดขึ้นมากมายไม่ว่าจะเป็นธุรกิจค้าปลีก ค้าส่ง delivery หรือการขายของออนไลน์ในแอปพลิเคชันต่าง ๆ เช่น อินสตาแกรม เฟซบุ๊ก ทวิตเตอร์ ช้อปปี้ ลาซาด้า หรือบางร้านมีเว็บหรือแอปพลิเคชันของร้านตัวเอง แต่แอปพลิเคชันดังกล่าวยังไม่เป็นที่นิยมและเป็นที่รู้จักมากนัก คณะผู้จัดทำจึงได้แรงบันดาลใจในการพัฒนาแอปพลิเคชันในการขายของออนไลน์ขึ้นมาเพื่อกระตุ้นการใช้แอปพลิเคชันในการขายของออนไลน์

1.2. วัตถุประสงค์ของการทำโครงงาน

1.2.1 เพื่อศึกษาหลักการออกแบบพัฒนาซอฟต์แวร์

1.2.2 เพื่อศึกษาการออกแบบและพัฒนาแอปพลิเคชันสำหรับร้านค้าออนไลน์ด้วย Spring Boot

1.3. ขอบเขตของโครงงาน

1.3.1 ออกแบบและพัฒนาแอปพลิเคชันตามหลักการ S.O.L.I.D.

1.3.2 แอปพลิเคชันสามารถใช้งานได้จริง

1.3.2.1 แอปพลิเคชันสามารถเลือกดูสินค้าได้

1.3.2.2 แอปพลิเคชันสามารถซื้อของได้

1.4. ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้ศึกษาหลักการออกแบบพัฒนาซอฟต์แวร์

1.4.2 ได้ศึกษาการออกแบบและพัฒนาแอปพลิเคชันสำหรับร้านค้าออนไลน์ด้วย Spring Boot

1.4.3 ได้ศึกษาการเขียนแอปพลิเคชันให้เป็นไปตามหลักการ S.O.L.I.D.

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

โครงงานเรื่องการสร้างฐานข้อมูลและรับส่งข้อมูลแบบ API ด้วย Spring Boots Framework นี้ทางคณะผู้จัดทำได้ทำการศึกษาค้นคว้าและรวบรวมข้อมูลในด้านต่าง ๆ ในการออกแบบแอปพลิเคชัน ตลอดจนการประยุกต์ใช้ฝนโครงงาน โดยมีรายละเอียดดังนี้

2.1. Flutter Framework

2.2. Spring Boot

2.3. Java language

2.4. GitHub

2.5. Android Studio

2.6. Xampp

2.7. Postman Agent

2.8. Heroku

2.1. Flutter Framework

ฟลัตเตอร์ คือเครื่องมือที่ใช้สร้าง User Interface สำหรับ Mobile application ที่สามารถทำงานได้ทั้งระบบ iOS และ Android ในเวลาเดียวกัน โดยภาษาที่ใช้ใน Flutter นั้นจะเป็นภาษา Dart ซึ่งถูกพัฒนาโดย Google และที่สำคัญคือเป็น Open source ที่สามารถใช้งานได้แบบฟรี ๆ อีกด้วย จุดเด่นของฟลัตเตอร์ คือ ระบบ Hot Reload โดยเมื่อมีการทดสอบ การสร้าง หรือการกระทำต่าง ๆ เกี่ยวกับ User Interface จะต้องมีการโหลดหน้าใหม่เพื่อให้หน้าต่างของแอปพลิเคชันมีการเปลี่ยนแปลง ซึ่งระบบ Hot Reload จะเข้ามาช่วยในส่วนของการโหลดหน้าใหม่ โดยจุดเด่นของระบบนี้คือการย่นระยะเวลาที่ใช้ในการโหลดหน้าใหม่ให้เหลือเพียงเสี้ยววินาทีเท่านั้น ทำให้การพัฒนา User Interface ของแอปพลิเคชันมีความรวดเร็วขึ้น

2.2. Spring Boot

(Assanai Manurat, 2559) Spring Boot เป็น project ส่วนหนึ่งของ Spring Framework ที่ช่วยในการพัฒนาแอปพลิเคชันได้อย่างรวดเร็วโดยที่มีการทำ Auto Configuration ทำให้ไม่ต้องเสียเวลาไป Config ทุกอย่างเอง เหมือนแต่ก่อน และสามารถสร้าง standalone application ที่ export เป็น jar หรือจะทำเป็น war แล้วนำไป deploy ที่ application server เหมือนเดิมได้

2.3. Java language

(พื้นฐานภาษาจาวา, ม.ป.ป.) Java หรือ Java programming language คือภาษาโปรแกรมเชิงวัตถุ พัฒนาโดย เจมส์ กอสลิง และวิศวกรคนอื่น ๆ ที่บริษัท ซัน ไมโครซิสเต็มส์ ภาษานี้มีจุดประสงค์เพื่อใช้แทนภาษาซีพลัสพลัส C++ โดยรูปแบบที่เพิ่มเติมขึ้นคล้ายกับภาษาอ็อบเจกต์ทีฟซี (Objective-C) แต่เดิมภาษานี้เรียกว่า ภาษาโอ๊ก (Oak) ซึ่งตั้งชื่อตามต้นโอ๊กใกล้ที่ทำงานของ เจมส์ กอสลิง แล้วภายหลังจึงเปลี่ยนไปใช้ชื่อ “จาวา” ซึ่งเป็นชื่อกาแฟแทน จุดเด่นของภาษา Java อยู่ที่ผู้เขียนโปรแกรมสามารถใช้หลักการของ Object-Oriented Programming มาพัฒนาโปรแกรมของตนด้วย Java ได้ ภาษา Java เป็นภาษาสำหรับเขียนโปรแกรมที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (OOP : Object-Oriented Programming) โปรแกรมที่เขียนขึ้นถูกสร้างภายในคลาส ดังนั้นคลาสคือที่เก็บเมทอด (Method) หรือพฤติกรรม (Behavior) ซึ่งมีสถานะ (State) และรูปพรรณ (Identity) ประจำพฤติกรรม (Behavior)

2.4. GitHub

(Saixiii, 2560) GitHub คือ website Git (version control repository) ที่อยู่บน internet มีการทำงานแบบเดียวกับ Git แต่สามารถเข้าถึงข้อมูลและจัดการไปผ่าน web โดยไม่ต้องเสียเงิน หรือลงทุนตั้ง server เพื่อติดตั้ง Git เองเลย แต่ code project ทั้งหมดจะถูกแจกจ่ายให้คนอื่น ๆ สามารถเห็นได้ด้วย ซึ่ง GitHub ก็มีการเสนอ plan แบบส่วนตัว ถ้าอยากให้ code ไม่ถูกแจกจ่ายออกไปโดยจะมีค่าใช้จ่ายเพิ่มเติม ปัจจุบันมีมากกว่า 20 ล้าน user รวมกันกว่า 60 ล้าน repository บนระบบ

2.5. Android Studio

(Simple app, ม.ป.ป.) Android Studio เป็นเครื่องมือไว้สำหรับพัฒนาโปรแกรม Android โดยเฉพาะโดยวัตถุประสงค์ของ Android Studio คือต้องการพัฒนาเครื่องมือ IDE ที่สามารถพัฒนา App บน Android ให้มีประสิทธิภาพมากขึ้น ทั้งด้านการออกแบบ GUI ที่ช่วยให้สามารถ Preview ตัว App มุมมองที่แตกต่างกันบน Smart Phone แต่ล่ะรุ่น สามารถแสดงผลบางอย่างได้ทันทีโดนไม่ต้องทำการรัน App บน Emulator รวมทั้งยังแก้ไขปรับปรุงในเรื่องของความเร็วของ Emulator ที่ยังเจอปัญหากันอยู่ในปัจจุบัน

2.6. Xampp

(aosoft, ม.ป.ป.) XAMPP คือโปรแกรมจำลอง web server ทำให้สามารถทดสอบเว็บไซต์ได้โดยที่ไม่ต้องเชื่อมต่ออินเตอร์เน็ต และไม่มีค่าใช่จ่ายใด ๆ XAMPP ประกอบด้วย Apache PHP MySQL PHP MyAdmin Perl ซึ่งเป็นโปรแกรมพื้นฐานที่รองรับการทำงาน CMS ซึ่งเป็นชุดโปรแกรม สำหรับออกแบบเว็บไซต์ที่ได้รับความนิยมในปัจจุบัน ไฟล์สำหรับติดตั้ง Xampp นั้นอาจมีขนาดใหญ่ เนื่องจาก มีชุดควบคุมการทำงานที่ช่วยให้การปรับแต่งส่วนต่าง ๆ ง่ายขึ้น XAMPP นั้นรองรับระบบปฏิบัติการหลายตัว เช่น Windows Linux Apple ทำงานได้ทั้งบนระบบปฏิบัติการแบบ 32 bit และ 64 bit

2.7. Postman Agent

(Ae eiei, ม.ป.ป.) Postman คือเครื่องมือสำหรับช่วยในการพัฒนา API ทดสอบการทำงานของ Service รวมถึงการ Mock Service อีกด้วย ซึ่งช่วยเราทำเรื่องยาก ๆ ให้กลายเป็นเรื่องง่าย โดยความนิยมของ Postman หลัก ๆ มาจาก UI ที่สวยงามใช้งานง่ายกว่า Tools อื่น ๆ โดยผู้ใช้งานไม่จำเป็นต้องมีความเรื่องภาษาโปรแกรมมิ่งก็สามารถใช้งานได้อย่างสบายใจ

2.8. Heroku

(Onamon Ja, 2563) Heroku เป็นบริการคลาวด์แบบ Platform as a Service (Paas) ที่ให้บริการสำหรับนักพัฒนาซอฟต์แวร์ สามารถรองรับภาษาต่าง ๆ เช่น Java Python PHP Ruby Go และ Node.js เป็นต้น โดยนักพัฒนาซอฟต์แวร์ใช้ Heroku ในการ deploy และจัดการแอปพลิชัน เพราะ Heroku มี Add-ons สำหรับเพิ่มเติมบริการอื่น ๆ เช่น PostgreSQL MongoDB Redis เป็นต้น มีให้เลือกใช้งานฟรี และเสียเงิน การให้บริการในรูปแบบของ Cloud มีการให้บริการอยู่ 3 ประเภท หนึ่งในประเภทนั้นคือ “Platform as a Service (Paas)” เป็นการให้บริการด้าน Platform สำหรับผู้ใช้งานที่ทำงานเกี่ยวกับ Software และ Application โดยผู้ให้บริการ Cloud จะจัดเตรียมสิ่งที่จำเป็นต้องใช้ในการพัฒนา เช่น Database Server Web Application เป็นต้น

บทที่ 3
สรุปผลการทำงาน

3.1. สรุปผลการศึกษา

จากการศึกษาการใช้ Spring boot ในการสร้างฐานข้อมูลและทำการรับส่งข้อมูลในรูปแบบของ API นั้น ผลปรากฏว่า Spring boot สามารถสร้าง API และสามารถให้ส่วนของหน้าแอพพลิเคชั่นเรียกใช้และส่งข้อมูลได้จริง อีกทั้งยังสามารถใช้ Spring boot สร้างแอพพลิเนที่มีการออกแบบโครงสร้างของโค้ดให้ข้ากับหลักการ SOLID ได้ โดยเราได้มีการใช้หลักการ SOLID ทั้งหมด 3 ข้อคือ 1. S Single-Responsibility (S.R.P) , 2. O — Open/Closed principle , 3. D — Dependency Inversion principle โดยผู้จัดทำได้มีการสร้างฐานข้อมูลดังนี้

3.1.1 Code พร้อมอธิบายตามหลัก Solid

คลาสที่ใช้ในการสร้างฐานข้อมูล ได้ใช้ Annotation @Entity ในการสร้าง ซึ่งแต่ละคลาสนั้นจะจัดการ Entity อย่างละอัน ซึ่งตรงตามหลักการ SOLID ข้อ S Single-Responsibility (S.R.P) ที่แต่ละคลาสจะมีหน้าที่เพียงอย่างเดียว

```java
@Entity

public class BraceletOrder {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private long bid;
    private long cid;

    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn
    private BraceletProduct b;
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn
    private Customer c;

    @CreationTimestamp
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime dateorder;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }
```

```java
public long getBid() {
    return bid;
}

public void setBid(long bid) {
    this.bid = bid;
}

public long getCid() {
    return cid;
}

public void setCid(long cid) {
    this.cid = cid;
}

public BraceletProduct getB() {
    return b;
}

public void setB(BraceletProduct b) {
    this.b = b;
}

public Customer getC() {
    return c;
}

public void setC(Customer c) {
    this.c = c;
}

public LocalDateTime getDateorder() {
    return dateorder;
}

public void setDateorder(LocalDateTime dateorder) {
    this.dateorder = dateorder;
}
```

ภาพที่ 1 แสดงโค้ดของ Class BraceletOrder

```java
public class BraceletProduct {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long bid;
    private String bname;
    private double price;

    public BraceletProduct(long i, String bn, double p) {
        this.bname = bn;
        this.price = p;

    }

    public BraceletProduct(String bn, double p) {
        this.bname = bn;
        this.price = p;

    }

    public BraceletProduct(double p) {
        this.price = p;
    }

    public BraceletProduct() {

    }

    public void setId(long proid) {
        this.bid = proid;
    }

    public long getId() {
        return bid;
    }

    public void setName(String pron) {
        this.bname = pron;
    }

    public String getName() {
        return bname;
    }

    public void setPrice(double pron) {
        this.price = pron;
    }

    public double getPrice() {
        return price;
    }
```

```java
@Entity
public class Customer {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long cid;
    private String cname;
    private String ctell;
    private String address;

    public Customer() {

    }

    public Customer(long c, String cn, String t,String ad) {
        this.cid = c;
        this.cname = cn;
        this.ctell = t;
        this.address =ad;
    }
    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }
    public void setId(long proid) {
        this.cid = proid;
    }

    public long getId() {
        return cid;
    }

    public void setName(String pron) {
        this.cname = pron;
    }

    public String getName() {
        return cname;
    }

    public String getCtell() {
        return ctell;
    }

    public void setCtell(String ctell) {
        this.ctell = ctell;
    }
```

ภาพที่ 2 แสดงโค้ดของ Class BraceletProduct และ Class Customer

```java
@Entity
public class Invoice {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private long oid;

    @OneToOne (cascade=CascadeType.ALL)
    @JoinColumn
    private BraceletOrder O;

    @CreationTimestamp
    @DateTimeFormat
    @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss")
    private LocalDateTime date;

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    public long getOid() {
        return oid;
    }

    public void setOid(long oid) {
        this.oid = oid;
    }

    public BraceletOrder getO() {
        return O;
    }

    public void setO(BraceletOrder o) {
        O = o;
    }

   public LocalDateTime getDate() {
       return date;
   }

   public void setDate(LocalDateTime date) {
       this.date = date;
   }
```

ภาพที่ 3 แสดงโค้ดของ Class Invoice

ส่วนที่เป็น Class Interface ที่ได้มีการสืบทอด Class JpaRepository ซึ่งเข้ากับหลักการข้อที่ 3. D — Dependency Inversion principle ที่ว่าคลาสแม่จำต้องทำงานเหมือนคลาสลูกได้

```java
import com.example.DB.RealDB.Customer;
import com.example.DB.RealDB.Invoice;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface CustomerRepository extends JpaRepository<Customer,Long> {

}
```

ภาพที่ 4 แสดงตัวอย่าง Class interface ที่สืบทอด Class JpaRepository <Entity , Id>

ส่วนที่เป็น Class ประเภท Service ทำหน้าที่เป็น Service ที่ใช้จัดการกับข้อมูลใน Class ที่เป็น Entity เพื่อให้ Class Controller เรียกใช้งาน

```java
import com.example.DB.RealDB.BraceletProduct;
import com.example.DB.RealDB.Customer;
import com.example.DB.RealDB.Repository.CustomerRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.List;

@Repository
@Service
@Transactional
public class CustomerRepo {
    @Autowired
    private CustomerRepository rp;


    public List<Customer> getAll() {
        return rp.findAll();
    }

    public Customer getByID(long id) {
        return rp.findById(id).get();
    }

    public void add(Customer b) {
        rp.save(b);
    }

    public void delete(long id) {
        rp.deleteById(id);
    }
```

```java
import com.example.DB.RealDB.Invoice;
import com.example.DB.RealDB.Repository.InvoiceRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.List;

@Repository
@Service
@Transactional
public class InvoiceRepo {
    @Autowired
    private InvoiceRepository rp;

    public List<Invoice> getAll() {
        return rp.findAll();
    }

    public Invoice getByID(long id) {
        return rp.findById(id).get();
    }

    public Invoice add(Invoice b) {
        rp.save(b);
        return b;
    }

    public void delete(long id) {
        rp.deleteById(id);
    }
```

ภาพที่ 5 แสดงตัวอย่าง Class ประเภท Service

```java
import org.springframework.stereotype.Repository;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.List;
import java.util.Optional;

@Repository
@Service
@Transactional
public class OrderRepo {

    @Autowired
    private OrderRepository rp;

    public List<BraceletOrder> getAll() {
        return rp.findAll();
    }

    public BraceletOrder getByID(long id) {
        return rp.findById(id).get();
    }

    public BraceletOrder add(BraceletOrder b) {
      return  rp.save(b);
    }

    public void delete(long id) {
        rp.deleteById(id);
    }

}
```

```java
import com.example.DB.RealDB.BraceletProduct;
import com.example.DB.RealDB.Customer;
import com.example.DB.RealDB.Repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;
import org.springframework.stereotype.Service;

import javax.transaction.Transactional;
import java.util.List;

@Repository
@Service
@Transactional
public class ProductRepo {
    @Autowired
    private ProductRepository pr;

    public List<BraceletProduct> getAll() {
        return pr.findAll();
    }

    public BraceletProduct getByID(long id) {
        return pr.findById(id).get();
    }

    public BraceletProduct add(BraceletProduct b) {
        return  pr.save(b);
    }

    public void delete(long id) {
        pr.deleteById(id);
    }

}
```

ภาพที่ 6 แสดงตัวอย่าง Class ประเภท Service

ส่วนที่เป็น Class ประเภท Controller ซึ่งส่วนที่ใช้สร้าง RestFul API และกำหนด Path เพื่อให้สามารถ Requsets มายังเมธอดที่ต้องการให้ทำงานได้ อีกทั้งยังเข้าหลักการข้อ 2. O — Open/Closed principle เนื่องจากว่าในการทำงานของแต่ละเมธอดภายในคลาสนี้ หากต้องการแก้ไขหรือเปลี่ยนแปลง คุณสามารถเปลี่ยนแปลงได้โดยไม่ต้องทำการแก้ไขโค้ดเดิม

```java
@RestController
@RequestMapping("/Customer")
public class CustomerController {
    @Autowired
    private CustomerRepo crp;

    @DeleteMapping(value = "/del/{id}") //delController
    public String delete(@PathVariable long id) {
        crp.delete(id);
        return "ลบลูกค้ารหัส " + id + " เรียบร้อย";
    }

    @GetMapping(value = "/show") //showall
    public List<Customer> show() {
        return crp.getAll();
    }

    @GetMapping(value = "/show/{id}") //delController
    public Customer show(@PathVariable long id) {
        return crp.getByID(id);
    }

    @PostMapping(path="/add")
    public String insert(@RequestBody Customer b) {
        crp.add(b);
        return "เพิ่มลูกค้าเรียบร้อย";
    }
    @PostMapping("/edit")
    public void edit(@RequestBody Customer c){
        Customer ec = crp.getByID(c.getId());
        ec.setName(c.getName());
        ec.setCtell(c.getCtell());
        ec.setAddress(c.getAddress());
        ec.setId(c.getId());
        crp.add(ec);
    }
```

```java
@RestController
@RequestMapping("/Invoice")
public class InvoiceController {
    @Autowired
    private InvoiceRepo irp;
    @Autowired
    private OrderRepo or;

    @DeleteMapping(value = "/del/{id}")
    public String delete(@PathVariable long id) {
        irp.delete(id);
        return "ลบใบเสร็จรหัส " + id + " เรียบร้อย";
    }

    @GetMapping(value = "/show") //showall
    public List<Invoice> show() {
        return irp.getAll();
    }

    @GetMapping(value = "/show/{id}")
    public Invoice show(@PathVariable long id) {
        return irp.getByID(id);
    }

    @PostMapping(path = "/add")
    public String insert(@RequestBody Invoice c) {
        c.setO(or.getByID(c.getOid()));
        c.setDate(c.getDate());
        irp.add(c);
        return "เพิ่มใบเสร็จสั่งซื้อเรียบร้อย";

    }
    @PostMapping("/edit")
    public Invoice edit(@RequestBody Invoice c) {
        Invoice b = irp.getByID(c.getId());
        b.setId(c.getId());
        b.setOid(c.getOid());
        b.setO(or.getByID(c.getOid()));
        b.setDate(c.getDate());
        return irp.add(b);
    }
```

ภาพที่ 7 แสดงตัวอย่าง Class ประเภท Controller

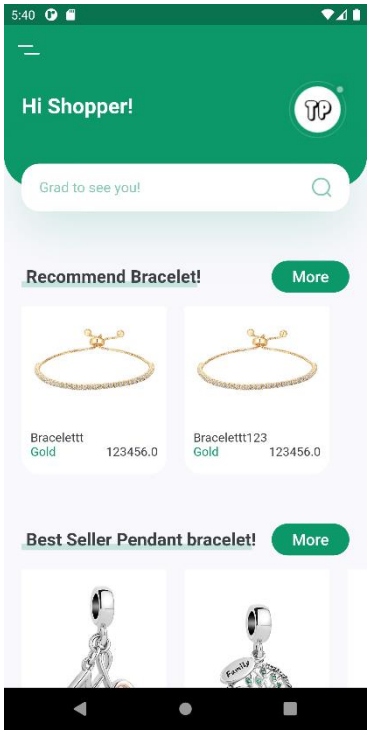
```java
@RestController
@RequestMapping("/Order")
public class OrderController {
    @Autowired
    private OrderRepo orp;
    @Autowired
    private ProductRepo pr;
    @Autowired
    private CustomerRepo cr;

    @RequestMapping(value = "/del/{id}", method = RequestMethod.DELETE)
    public String delete(@PathVariable long id) {
        orp.delete(id);
        return "ลบรายการสั่งซื้อรหัส " + id + " เรียบร้อย";
    }

    @GetMapping(value = "/show") //showall
    public List<BraceletOrder> show() {
        return orp.getAll();
    }

    @RequestMapping(value = "/show/{id}", method = RequestMethod.GET) //
    public BraceletOrder show(@PathVariable long id) {
        return orp.getByID(id);

    }

    @PostMapping(path = "/add")
    public String insert(@RequestBody BraceletOrder c) {
        c.setB(pr.getByID(c.getCid()));
        c.setC(cr.getByID(c.getBid()));
        orp.add(c);
        return "เพิ่มรายการสั่งซื้อเรียบร้อย";

    }

    @PostMapping("/edit")
    public BraceletOrder edit(@RequestBody BraceletOrder c) {
        BraceletOrder b = orp.getByID(c.getId());
        b.setB(c.getB());
        b.setC(c.getC());
        return orp.add(b);
    }
```

```java
@RestController
@RequestMapping("/Product")
public class ProductController {
    @Autowired
    private ProductRepo prp;

    @GetMapping(value = "/show") //showall
    public List<BraceletProduct> show() {
        return prp.getAll();
    }

    @GetMapping(value = "/show/{id}")
    public BraceletProduct show(@PathVariable long id) {
        return prp.getByID(id);
    }

    @PostMapping("/add")
    public String insert(@RequestBody BraceletProduct b) {
        prp.add(b);
        return "เพิ่มสินค้าเรียบร้อย";
    }

    @DeleteMapping(value = "/del/{id}") //delController
    public String delete(@PathVariable long id) {
        prp.delete(id);
        return "ลบสินค้ารหัส " + id + " เรียบร้อย";
    }

    @PostMapping("/edit")
    public BraceletProduct edit(@RequestBody BraceletProduct c) {
        BraceletProduct b = prp.getByID(c.getId());
        b.setName(c.getName());
        b.setPrice(c.getPrice());
        b.setId(c.getId());
        return prp.add(b);
    }
```

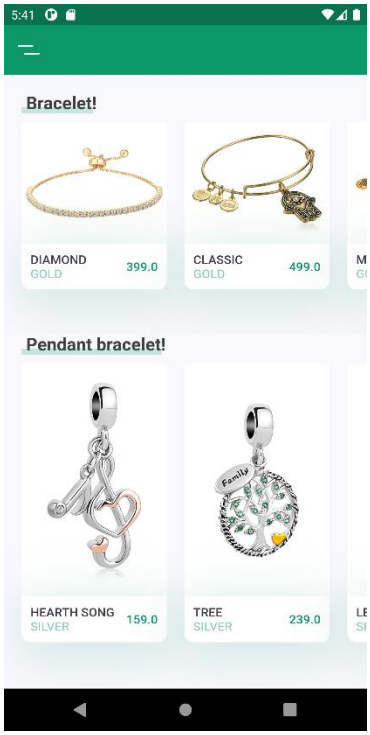
ภาพที่ 8 แสดงตัวอย่าง Class ประเภท Controller

โดยหน้าต่าง GUI ของแอปพลิเคชันมีดังนี้

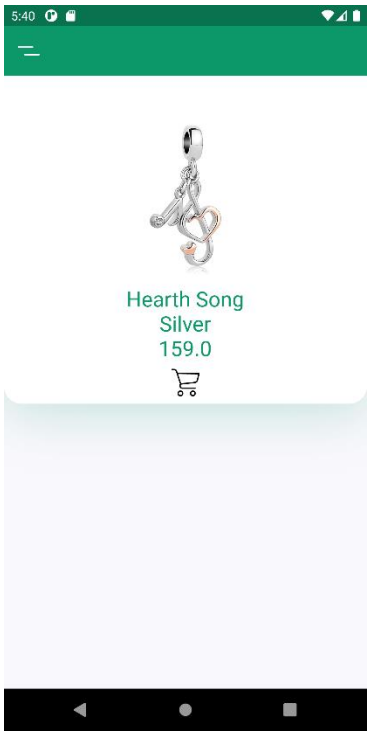
หน้าแรกของแอปพลิเคชัน โดยข้อมูลสินค้าจะรับค่ามาจากฐานข้อมูล โดยส่งมาโดยใช้ API ดังภาพที่ 9



ภาพที่ 9 หน้าแรกของร้านค้า



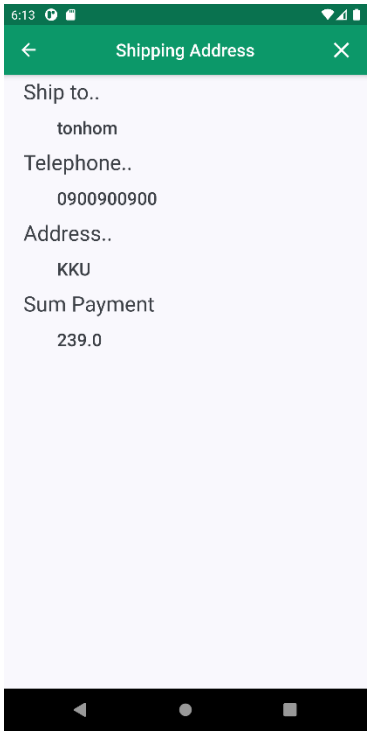
ภาพที่ 10 หน้าต่างสินค้าแยกตามหมวดหมู่



ภาพที่ 11 หน้าข้อมูลสินค้า

ภาพที่ 11 หน้ากรอกข้อมูลการส่งสินค้า

ภาพที่ 12 หน้าสั่งซื้อสินค้า กรอกรายละเอียดข้อมูลการจัดส่ง



ภาพที่ 13 หน้าข้อมูลรายละเอียดการสั่งซื้อสินค้า

เอกสารอ้างอิง

Ae eiei. (ม.ป.ป.). **มาทดสอบ API ด้วย PostMan กันเถอะ.** ค้นเมื่อ 15 พฤศจิกายน 2563, จาก https://bit.ly/35ENr6x

aosoft. (ม.ป.ป.). **XAMPP คืออะไร?.** ค้นเมื่อ 15 พฤศจิกายน 2563, จาก https://bit.ly/2II4Sd0

Assanai Manurat. (2559). **เริ่มต้นทำความรู้จักกับ Spring Boot.** ค้นเมื่อ 15 พฤศจิกายน 2563, จาก http://assanai.com/getting-started-spring-boot/

Onamon Ja. (2563). **Heroku คืออะไร?.** ค้นเมื่อ 15 พฤศจิกายน 2563 จาก https://bit.ly/3f12Y3w

Saixiii. (2560). **Github คืออะไร.** ค้นเมื่อ 15 พฤศจิกายน 2563, จาก https://saixiii.com/what-is-github/

Simple app. (ม.ป.ป.). **รู้จักกับ Android Studio**. ค้นเมื่อ 15 พฤศจิกายน 2563, จาก https://sites.google.com/site/m1929800104/android-studio/android-studio

พื้นฐานภาษาจาวา. (ม.ป.ป.). **พื้นฐานภาษาจาวา.** ค้นเมื่อ 15 พฤศจิกายน 2563, จาก https://sites.google.com/site/beammus55/phun-than-phasa-cawa