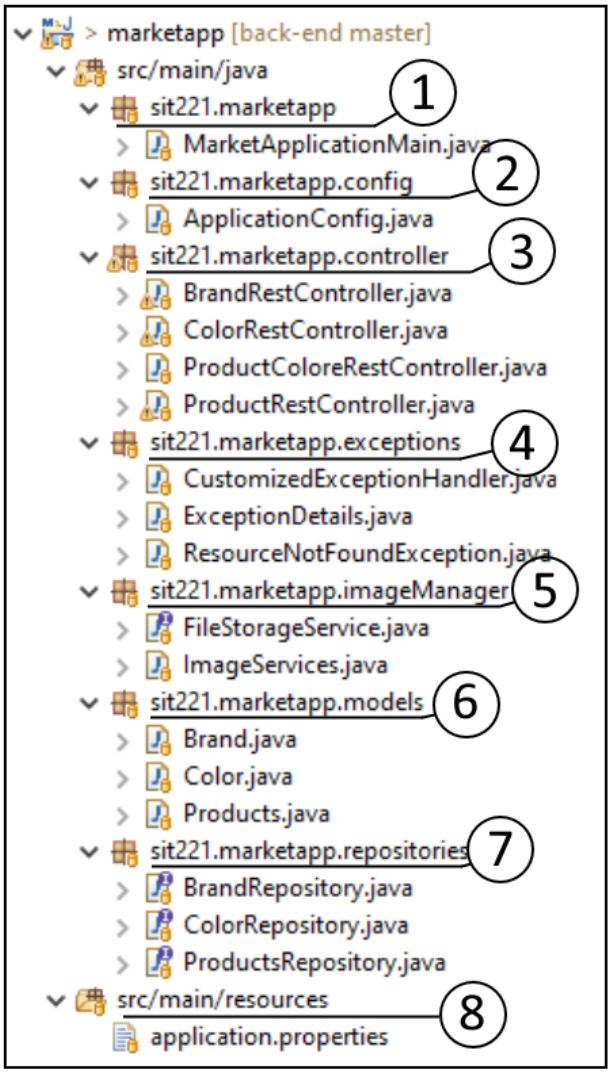
Back-end

เอกสารฉบับนี้แสดงการทำงานของ API ต่าง ๆ ที่ถูกรับผิดชอบโดย Back-end team โดยสังเขป



General Information

Application URL : http://52.139.201.170:8086
รับผิดชอบโดย : ธนพัฒน์ ดารารัตน์ 62130500032

REST API Controller

sit221.marketappcontroller BrandRestController

เรียกผ่าน /api/brands

GET http://52.139.201.170:8086/api/colors
private List<Brand> findAllColor()
เรียกข้อมูลทั้งหมดของ Brands จากฐานข้อมูล แล้วส่ง Object จากโมเดล Brands ไปแสดงเป็น Json

GET http://52.139.201.170:8086/api/colors/{id}
private ResponseEntity<Brand> findColorById(@PathVariable(value = "id") int searchid)
เรียก Object ของ Color ที่มี id ที่ต้องการออกมา โยน ResourceNotFoundException หากไอดีที่ต้องการเป็น null (ไม่มีอยู่ในฐานข้อมูล)

sit221.marketappcontroller ColorRestController

เรียกผ่าน /api/colors

GET http://52.139.201.170:8086/api/brands
public List<Brand> findAllColor()
เรียกข้อมูลทั้งหมดของ Color จากฐานข้อมูล แล้วส่งไปแสดงเป็น Json

GET http://52.139.201.170:8086/api/brands/{id}
private ResponseEntity<Color> findProductById(@PathVariable(value = "id") int searchid)
เรียก Object ของ Color ที่มี id ที่ต้องการออกมา โยน ResourceNotFoundException หากไอดีที่ต้องการเป็น null (ไม่มีอยู่ในฐานข้อมูล)

สังเกตได้ว่าสองคลาสนี้ทำงานคล้ายกัน เพราะไม่ได้ออกแบบให้ผู้ใช้สามารถเพิ่ม Brands หรือ Colors ได้ด้วยตนเอง ให้ใช้ค่าที่มีให้ประกอบข้อมูลอื่นเท่านั้น

sit221.marketappcontroller ProductRestController

เรียกผ่าน /api/product

GET http://52.139.201.170:8086/api/product
public List<Products> findAllProducts()
เรียกข้อมูลทั้งหมดของ product จากฐานข้อมูล แล้วส่งไปแสดงเป็นก้อน Json

GET http://52.139.201.170:8086/api/product/{id}
public ResponseEntity<Brand> findColorById(@PathVariable(value = "id") int searchid)
เรียก Object ของ Products ที่มี id ที่ต้องการออกมา โยน ResourceNotFoundException หากไอดีที่ต้องการเป็น null (ไม่มีอยู่ในฐานข้อมูล)

GET http://52.139.201.170:8086/api/product/image/{filename}
public Resource getImage(@PathVariable String filename)
ค้นหาและเรียกรูปภาพนั้นโดยใช้ชื่อ สำหรับเรียกรูปภาพไปแสดงเท่านั้น

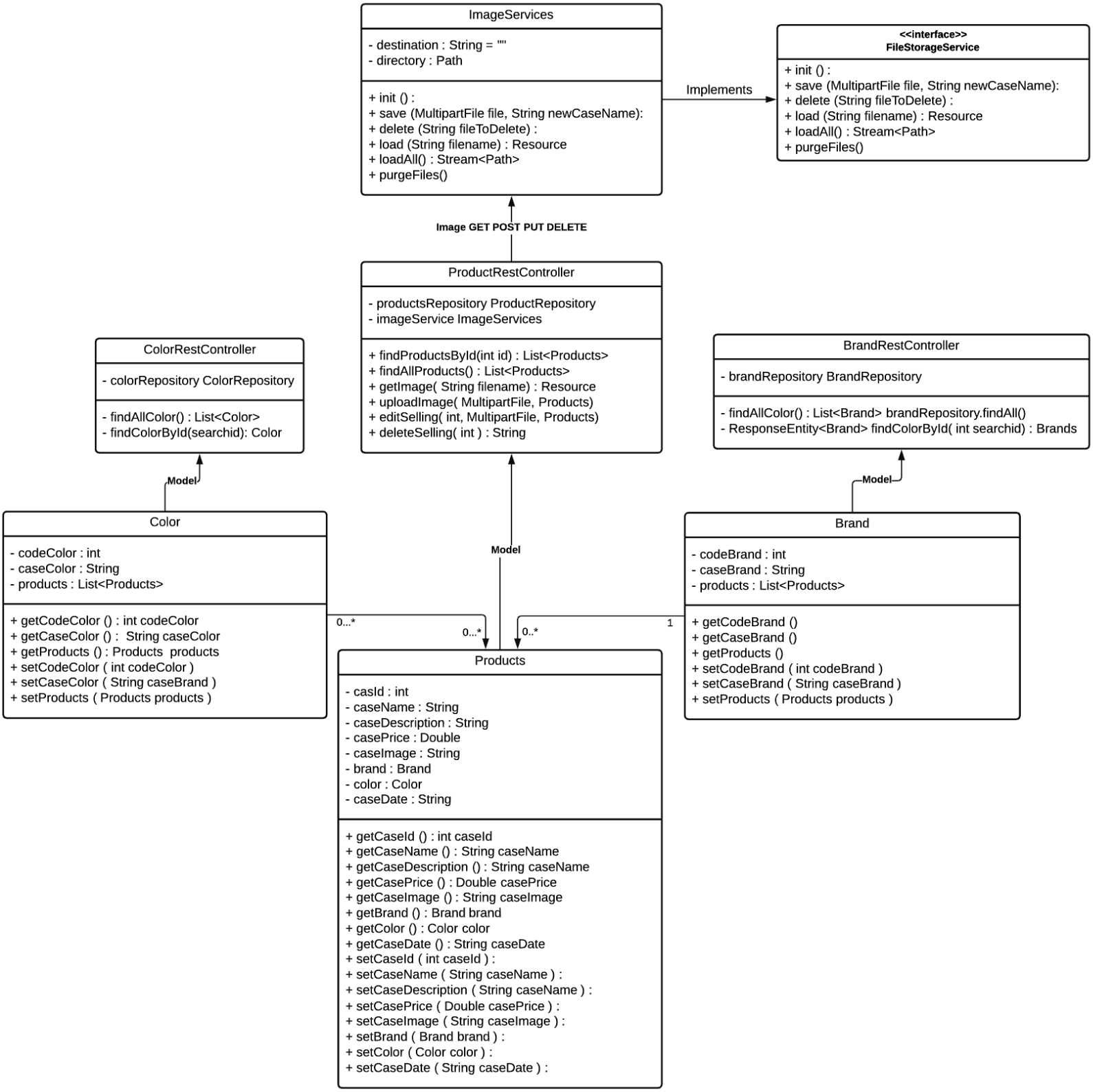
POST http://52.139.201.170:8086/api/product/
public void uploadImage(@RequestParam("imageFile") MultipartFile imageFile,
@RequestPart("body") Products newProduct
นำเข้าข้อมูล Json และรูปภาพจาก Font-End โดยต้องการข้อมูลทั้งสองอย่าง หากขาดอย่างใดอย่างหนึ่งก็จะไม่ทำต่อให้ รูปภาพที่ถูกส่งเข้ามาจะถูกเปลี่ยนชื่อ โดยสุ่มตัวเลข 10 ตัวแล้วทำให้ตัวเลขแต่ละตัวนั้นเป็นตัวอักษรภาษาอังกฤษ แล้วต่อท้ายด้วยชื่อเดิมของไฟล์นั้น ก่อนที่จะส่งชื่อนั้นไปให้ imageService จับยัดลงไปในไฟล์ ซึ่งจะลดโอกาสที่ไฟล์จะซ้ำกันได้มาก หรือแทบจะเป็นไปไม่ได้เลยที่มันจะซ้ำกันอีก ซึ่งหลังจากที่อัพโหลดและผ่าน method แล้ว รูปภาพและตัวแปรที่เก็บชื่อรูปภาพจะมีชื่อนั้นตลอดชีวิตของมัน

PUT http://52.139.201.170:8086/api/product/{id}
public ResponseEntity<Products> editSelling(@PathVariable(value = "id") int editingId,
@RequestParam(required = false, value = "image") MultipartFile imageFile,
@RequestPart("edit") Products newDetails)

นำเข้าข้อมูล Json และรูปภาพจาก Font-End โดยจะไม่เอารูปภาพเข้ามาด้วยก็ได้ (ในกรณีที่จะใช้รูปเดิมไม่เปลี่ยน) แต่หากเปลี่ยนรูป รูปนั้นก็จะได้ชื่อเช่นเดียวกับรูปแรกที่ถูกอัพโหลดเข้ามา ก่อนที่จะถูกเซฟทับ หมายความว่าชื่อรูปภาพจะไม่สามารถเปลี่ยนได้ด้วยตนเอง โยน ResourceNotFoundException และออกจากงานถ้าหากไม่เจอ ID นี้

DELETE http://52.139.201.170:8086/api/product/{id}
public String deleteSelling(@PathVariable(value = "id") int removeId)

Class diagram [UML]



รับคำสั่งลบทั้งข้อมูลและรูปภาพ (หายไปไม่เหลือซาก) ซึ่งจะลบทั้งรูปภาพที่อยู่ในที่เก็บของ Back-end และข้อมูลจากฐานข้อมูลด้วย โยน ResourceNotFoundException และออกจากงานถ้าหากไม่เจอ ID นี้

MarketApplicationMain

```
#Default = 8086 ; FontEnd = 8080 ; Data = 3306
server.port=8086

#The following folder will be generated when exucuted the application.
marketapp.imagepath=product-images

#Allowed Origin ( Only this origin will be able to access BackEnd )
BACKENDV2.origin.host=http://localhost:8080/ , http://52.139.201.170:8080/
BACKENDV2.origin.method=GET, PUT, HEAD, POST, DELETE, OPTIONS

#Import some more stuffs.
spring.jpa.hibernate.naming.implicit-strategy=org.springframework.boot.orm.jpa.hibernate.SpringImplicitNamingStrategy
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

#max file and request size
spring.http.multipart.max-file-size=10MB
spring.http.multipart.max-request-size=11MB

#----------------------------------------------------------------
#!!! W A R N I N G !!!  Sensitive information below,  do not expose
#----------------------------------------------------------------

#Connect to database using the following properties.
spring.datasource.url=jdbc:mysql://52.139.201.170:3306/ProductCase
spring.datasource.username=test221
spring.datasource.password=test

#Default Password
```

```
#url source : jdbc:mysql://138.91.36.165:3306/ProductCase
#User name : testproject221
#User password : test

#Test default
#spring.datasource.url=jdbc:mysql://${MYSQL_HOST:localhost}:3306/productcase
#spring.datasource.username=root
#spring.datasource.password=
```