□ 모듈프로젝트 산출물

모듈프로젝트 보고서

2021 년 9 월 24 일

3 조
권진희
김영모
박현우
윤희상

1. 모듈프로젝트 개요

교육 과정으로 짜여진 모듈 프로젝트 일정에 따라 교육 초기부터 학습했던 지식들을 활용하여 온라인 서점 서비스를 개발하는 모듈 프로젝트를 시작하게 되었습니다.

2. 모듈프로젝트 목적

5 월 31 일부터 진행된 교육 과정에서 배웠던 지식들을 활용하여 MSA 방식으로 주어진 요구사항에 맞게 프로젝트를 진행해 보는 것을 목표로 하며, 최종적으로 컨테이너화 된 서비스들을 클라우드 환경에 배포하는 것으로 합니다. 프로젝트 진행 과정에서 발생하는 여러 이슈들을 바탕으로 향후 진행될 메인 프로젝트 시에 비용 절감의 계기로 삼을 것입니다.

3. 모듈프로젝트 환경

구분	이름	버전
프로그래밍 언어	JAVA	11
	JavaScript	ES6
Web	Spring Boot	2.5.2
	React	17.0.2
	Nginx	1.21.3
	Apache Tomcat	9.0.52
Container Platform	Docker	20.10.7
Server	AWS(ec2)	Amazon Linux 2
Database	RDS : MariaDB	10.4.13
ETL	Kafka	2.13-2.8.0
	Zookeeper	3.5.9
	zipkin	6.1.0
	rabbitmq	jre-8u291-windows-x64
Tool	Git	2.32.0
	Github	-

4. 모듈프로젝트 설계

● 요구사항 분석

업무	요구사항	상세 기능
로그인/ 로그아웃 기능	사용자 로그인	이메일과 비밀번호를 통하여 로그인합니다.
	사용자 로그아웃	로그아웃 버튼을 클릭하여 로그아웃합니다.
	일반/관리자 구분	하나의 로그인 폼에서 일반 사용자와 관리자로 로그인 할 수 있습니다.
회원가입 기능	회원가입	이름, 비밀번호, 이메일, 연락처, 주소를 기입하여 회원가입 합니다.
공통 기능	상품 조회	전체 상품의 목록을 보여줍니다.
	상품 상세보기	상품 정보 : ISBN, 제목, 작가, 갯수 등이 표기됩니다. 현재 상품을 장바구니에 담을 수 있고, 구매할 수도 있습니다.
	상품 검색	상품명, 작가 이름으로 상품을 검색할 수 있습니다.
일반 사용자 기능	장바구니	장바구니에 담긴 상품을 구매 / 수량변경 / 삭제가 가능합니다.
	구매하기	구매페이지에서 상품의 정보와 자신의 정보를 확인 후, 구매할 수 있습니다.
	주문목록 조회	자신이 구매한 상품들의 목록을 볼 수 있습니다.
	회원 정보수정	회원 정보 수정 및 회원 탈퇴를 할 수

		있습니다.
관리자 기능	상품등록	상품 정보를 입력하여 새로운 상품을 등록할 수 있습니다.
	결제 관리	결제 내역을 리스트로 볼 수 있습니다.
	회원 관리	회원 내역을 리스트로 볼 수 있습니다. 회원을 삭제할 수 있습니다.
	회원 상세 조회	회원 정보 리스트

● **API 명세서**

구분	METHOD	/PATH	기능
user-service	GET	/users	전체 회원 조회
	POST	/users	회원 가입
	GET	/users/{userId}	특정 회원 조회
	PUT	/users/{userId}	회원정보 수정
	DELETE	/users/{userId}	회원 탈퇴
order-service	GET	/{userId}/orders	유저 주문 조회
	POST	/{userId}/orders	주문하기
	PUT	/{orderId}/orders	주문 정보 수정
	GET	/orders	전체 주문 조회
catalog-service	GET	/catalogs	전체 상품 조회

	POST	/catalogs	상품 등록
	GET	/catalogs/{productId}	상품 상세 정보
	PUT	/catalogs/{productId}	상품 정보 수정
	DELETE	/catalogs/{productId}	상품 삭제
	GET	/catalogs/search/productname/{productName}	상품 이름으로 상품 조회
	GET	/catalogs/search/writer/{writer}	작가 이름으로 상품 조회
cart-service	GET	/{userId}/carts	유저 장바구니
	POST	/{userId}/carts	장바구니에 상품 추가
	PUT	/{userId}/carts	카트 정보 수정
	DELETE	/carts/{id}	장바구니에 담은 상품 삭제

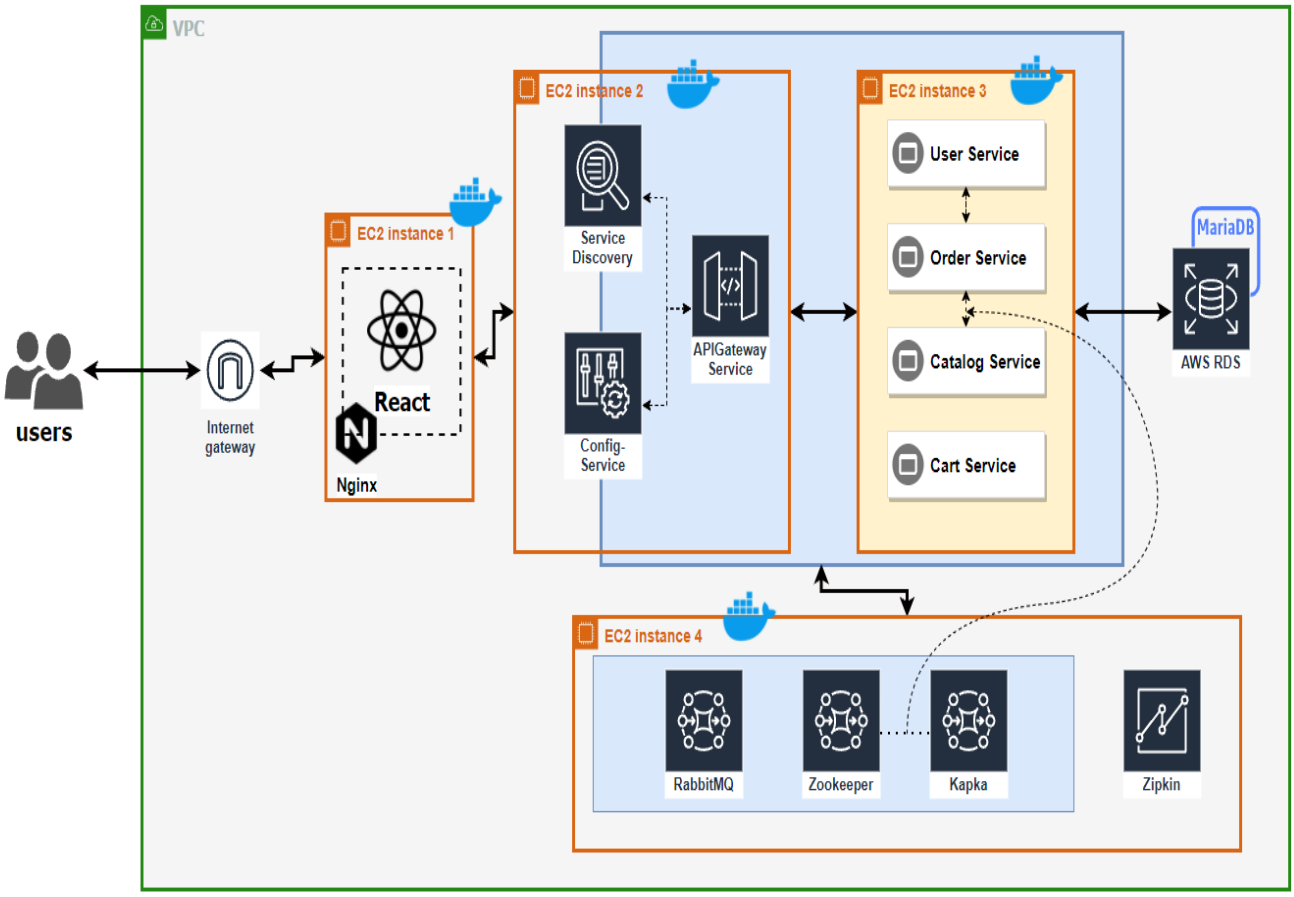
● 개발 표준 정의서

변수 명명 규칙
- 대상 혹은 동사+대상의 형태로 변수를 명명한다. - Java 변수의 명명규칙은 동사 또는 명사를 조합하여 30 자 이내로 명명한다. - 첫 글자는 소문자를 사용하며 이후 용어의 첫 글자만 대문자를 사용하며, 숫자 및 특수문자는 사용하지 않는다.

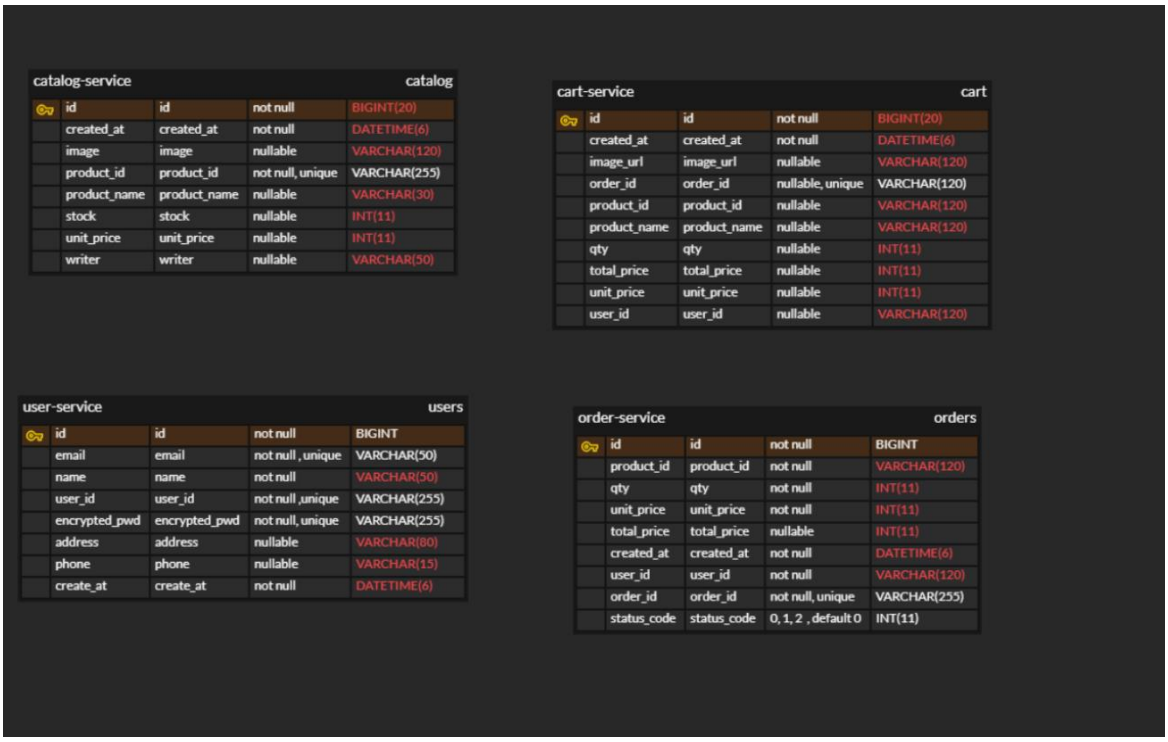
메서드 명명 규칙
- Java method 의 명명규칙은 [동사]로 한다. - 동사 + 대상 형태로 메서드를 명명한다. - 대상의 경우 첫글자를 소문자로 한다. (카멜 표기법 사용) - 대상 값으로 조회할 경우 By 대상 값으로 조회한다. ex) getCatalogsByProductName

구분	유형	동사	비고
상태확인	상태확인	status	
Controller Service ServiceImpl	조회	get	
	모든 조회	get	대상을 복수형태로 작성
	등록(생성)	create	
	수정	update	
	삭제	delete	
VO, DTO	값 읽기	get	
	값 설정	set	

● 시스템 구성도


VPC
EC2 instance 1
React
Nginx
EC2 instance 2
Service Discovery
Config-Service
APIGateway Service
EC2 instance 3
User Service
Order Service
Catalog Service
Cart Service
MariaDB
AWS RDS
EC2 instance 4
RabbitMQ
Zookeeper
Kapka
Zipkin
users
Internet gateway

● ERD



● 테이블

○ USERS

```
CREATE TABLE `users` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `address` VARCHAR(80) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `created_at` DATETIME(6) NOT NULL,
  `email` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_general_ci',
  `encrypted_pwd` VARCHAR(255) NOT NULL COLLATE 'utf8mb4_general_ci',
  `name` VARCHAR(50) NOT NULL COLLATE 'utf8mb4_general_ci',
  `phone` VARCHAR(15) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `user_id` VARCHAR(255) NOT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE INDEX `UK_6dotkott2kjsp8vw4d0m25fb7` (`email`) USING BTREE,
  UNIQUE INDEX `UK_d0xwk3cyqbi384375x5m7ttl7` (`encrypted_pwd`) USING BTREE,
  UNIQUE INDEX `UK_6efs5vmce86ymf5q7lmvn2uuf` (`user_id`) USING BTREE
)
```

○ **ORDERS**

```
CREATE TABLE `orders` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `created_at` DATETIME(6) NOT NULL DEFAULT current_timestamp(6),
  `order_id` VARCHAR(120) NOT NULL COLLATE 'utf8mb4_general_ci',
  `product_id` VARCHAR(120) NOT NULL COLLATE 'utf8mb4_general_ci',
  `qty` INT(11) NOT NULL,
  `status_code` INT(11) NULL DEFAULT '0',
  `total_price` INT(11) NULL DEFAULT NULL,
  `unit_price` INT(11) NOT NULL,
  `user_id` VARCHAR(120) NOT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE INDEX `UK_hmsk25beh6atojvle1xuymjj0` (`order_id`) USING BTREE
)
```

○ **CATALOG**

```
CREATE TABLE `catalog` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `created_at` DATETIME(6) NOT NULL DEFAULT current_timestamp(6),
  `image` VARCHAR(120) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `product_id` VARCHAR(120) NOT NULL COLLATE 'utf8mb4_general_ci',
  `product_name` VARCHAR(30) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `stock` INT(11) NULL DEFAULT NULL,
  `unit_price` INT(11) NULL DEFAULT NULL,
  `writer` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE INDEX `UK_9gggyslu2usn0rxs32mf055wq` (`product_id`) USING BTREE
)
```

○ **CART**

```
CREATE TABLE `cart` (
  `id` BIGINT(20) NOT NULL AUTO_INCREMENT,
  `created_at` DATETIME(6) NOT NULL DEFAULT current_timestamp(6),
  `image_url` VARCHAR(120) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `order_id` VARCHAR(120) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `product_id` VARCHAR(120) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `product_name` VARCHAR(120) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  `qty` INT(11) NULL DEFAULT NULL,
  `total_price` INT(11) NULL DEFAULT NULL,
  `unit_price` INT(11) NULL DEFAULT NULL,
  `user_id` VARCHAR(120) NULL DEFAULT NULL COLLATE 'utf8mb4_general_ci',
  PRIMARY KEY (`id`) USING BTREE,
  UNIQUE INDEX `UK_dqdpmxpsb72di6d4pgpn0qex1` (`order_id`) USING BTREE
)
```

5. 모듈프로젝트 구현 산출물

◯ 개발 단계

■ 소스코드

- CORS 이슈 해결 - nginx 의 config 파일에서 proxy_pass 를 사용하는 방식으로 해결하였습니다.

nginx.conf

```
server {
    listen 80;

    location / {
        root    /app/build;
        index   index.html;
        try_files $uri $uri/ /index.html;
    }

    location /user-service {
        proxy_pass http://172.18.0.5:8000;
    }

    location /order-service {
        proxy_pass http://172.18.0.5:8000;
    }

    location /catalog-service {
        proxy_pass http://172.18.0.5:8000;
    }

    location /cart-service {
        proxy_pass http://172.18.0.5:8000;
    }
}
```

■ 토큰 관련 코드

- 토큰 받아오는 코드 : fetch 함수를 사용하여 서버와의 통신을 하였습니다. 아래 사진은, 로그인 시 header 에 token 이 발급되는 것을 localStorage 를 통하여 저장하는 과정입니다. 외에도 userId 등과 같이 자주 사용되는 정보를 저장하였습니다.

```
fetch(`/user-service/login`,{
    method: "POST",
    headers: {
        "Content-Type": "application/json"
    },
    body: JSON.stringify({
        email: values.email,
        password: values.password
    })
})
.then((res) => {
    if(res.headers.get('token')){
        localStorage.setItem("token", res.headers.get('token'));
        localStorage.setItem("userId", res.headers.get('userId'));
        localStorage.setItem("email", values.email);
        gogo.push("/");
    }
    else{
        alert("로그인 정보를 확인하세요.");
    }
})
```

- 토큰 이용하는 코드 : localStorage 에 저장된 토큰의 값을 활용하여 Bearer token 을 헤더에 포함하여 요청을 보내는 방식을 이용하였습니다.

```
const myHeaders = new Headers();
let token = "Bearer " + localStorage.getItem("token");
myHeaders.append("Authorization", token)

useEffect(()=>{
    fetch(`/user-service/users`,{
        "headers": myHeaders
    })
    .then(res => {
        return res.json();
    })
    .then(data => {
        setUserDatas(data);
    });
},[]);
```

■ 검색 기능 코드

- 상품 이름으로 검색 기능 : 상품명을 받아서 같은 상품명이 있는지 탐색한 후 같은 상품명이 있으면 모두 반환해 줍니다.

```
// 상품 이름으로 검색
@PostMapping("/catalogs/search/productname/{productName}")
public ResponseEntity<List<ResponseCatalog>> getCatalogsByProductName(@PathVariable String productName){

    ModelMapper mapper = new ModelMapper();
    mapper.getConfiguration().setMatchingStrategy(MatchingStrategies.STRICT);
    Iterable<CatalogEntity> catalogList = catalogService.getCatalogsByProductName(productName);
    List<ResponseCatalog> result = new ArrayList<>();
    catalogList.forEach(v -> {
        result.add(new ModelMapper().map(v, ResponseCatalog.class));
    });

    return ResponseEntity.status(HttpStatus.CREATED).body(result);
}
```

- 비슷한 방식으로 작가명으로 검색 기능 또한 구현하였습니다.

```
// 작가명 검색
@PostMapping("/catalogs/search/writer/{writer}")
public ResponseEntity<List<ResponseCatalog>> getCatalogsByWriter(@PathVariable String writer){
    ModelMapper mapper = new ModelMapper();
    mapper.getConfiguration().setMatchingStrategy(MatchingStrategies.STRICT);
    Iterable<CatalogEntity> catalogList = catalogService.getCatalogsByWriter(writer);
    List<ResponseCatalog> result = new ArrayList<>();
    catalogList.forEach(v -> {
        result.add(new ModelMapper().map(v, ResponseCatalog.class));
    });
    return ResponseEntity.status(HttpStatus.CREATED).body(result);
}
```

■ AWS migration 시 config-service 활용

```java
@Bean
public ProducerFactory<String, String> producerFactory() {
    Map<String, Object> props = new HashMap<>();
    props.put(ProducerConfig.BOOTSTRAP_SERVERS_CONFIG, env.getProperty("msg.kafkaip"));
    props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG, StringSerializer.class);
```

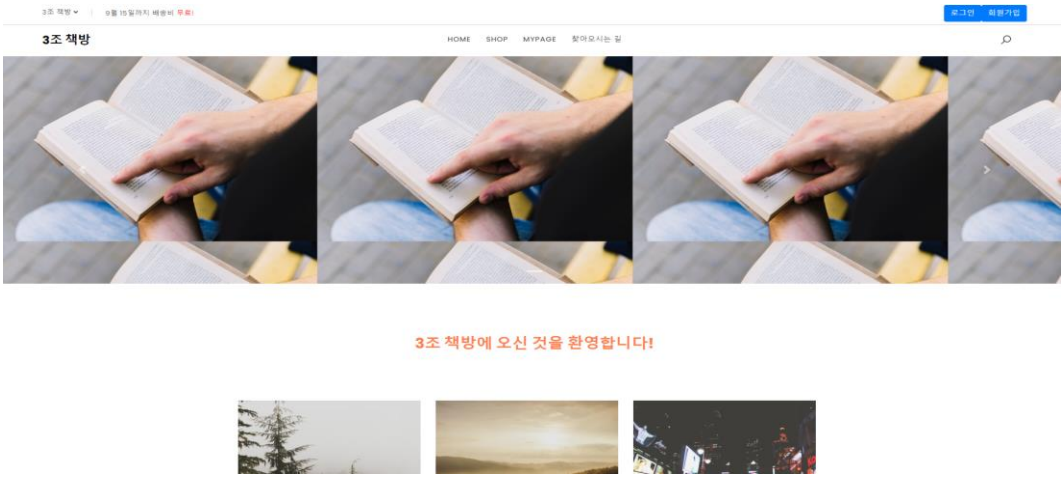
```yaml
msg:
  kafkaip: 3.37.212.247:9092
```

- 운영하는 인스턴스 IP 에 변동이 생길 경우 관리를 쉽게 하기 위하여 github 의 config 관련 코드에 IP 항목을 작성하였습니다. 즉, IP 가 변경되는 경우, config 파일만 수정해주면 됩니다.

○ **시스템 이행 계획서 – freedcamp 를 활용하였습니다.**

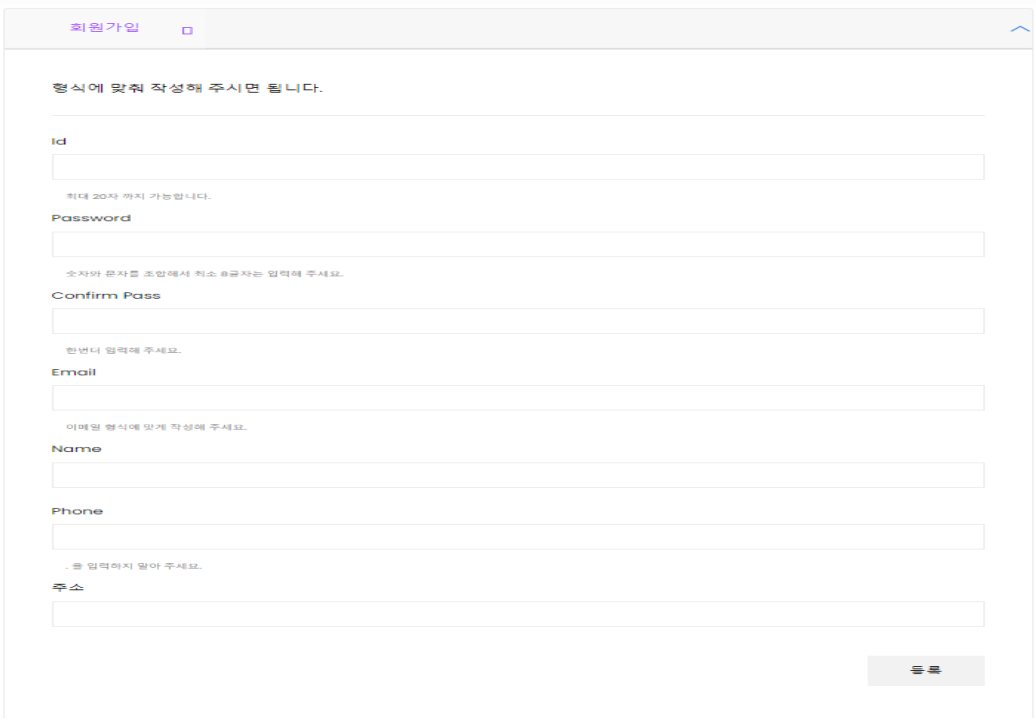
● 시험 단계

○ **공통** – 메인 화면입니다. 오른쪽 상단의 버튼을 통해 로그인과 회원가입을 할 수 있으며, 상단 바의 메뉴들을 이용할 수 있습니다.

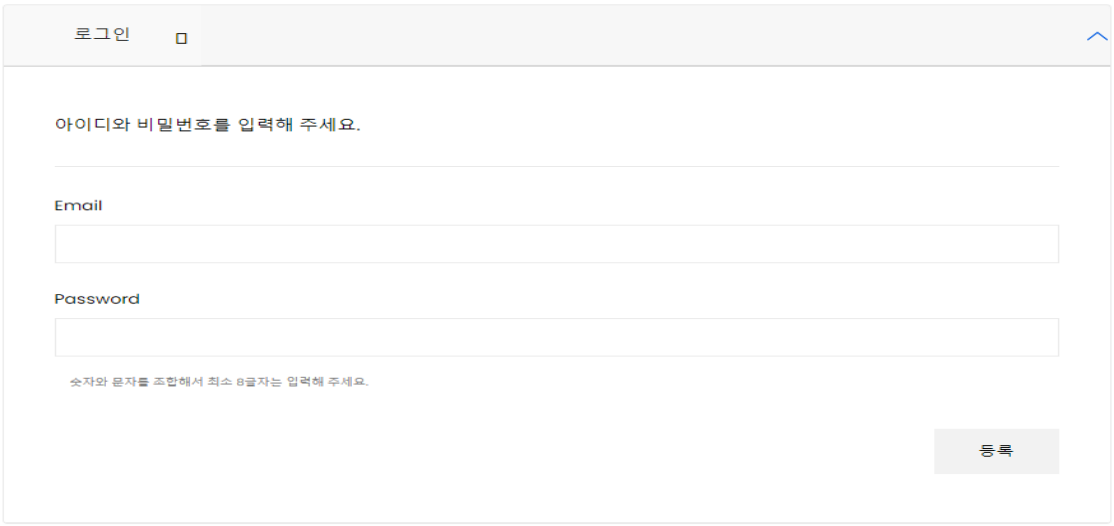


○ 사용자 매뉴얼

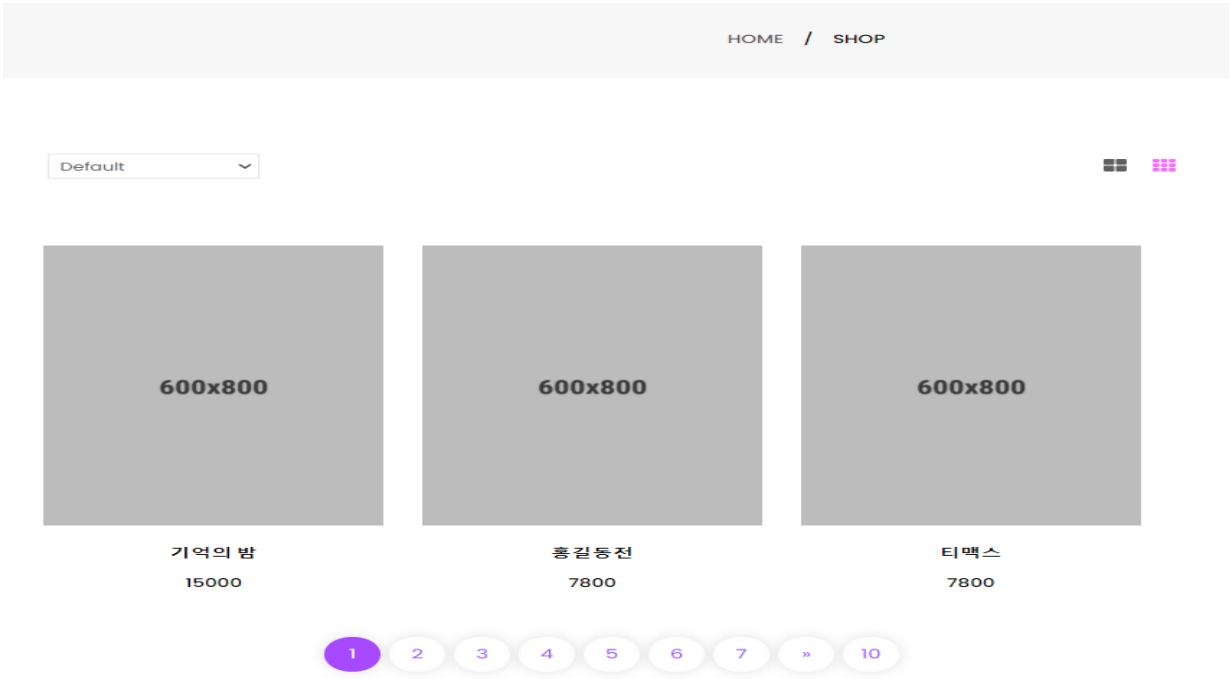
■ **회원가입** - 폼에 맞게 정보를 입력합니다. 아이디, 비밀번호, 이름은 필수 항목이며, 휴대폰 번호와 주소는 선택 항목입니다.



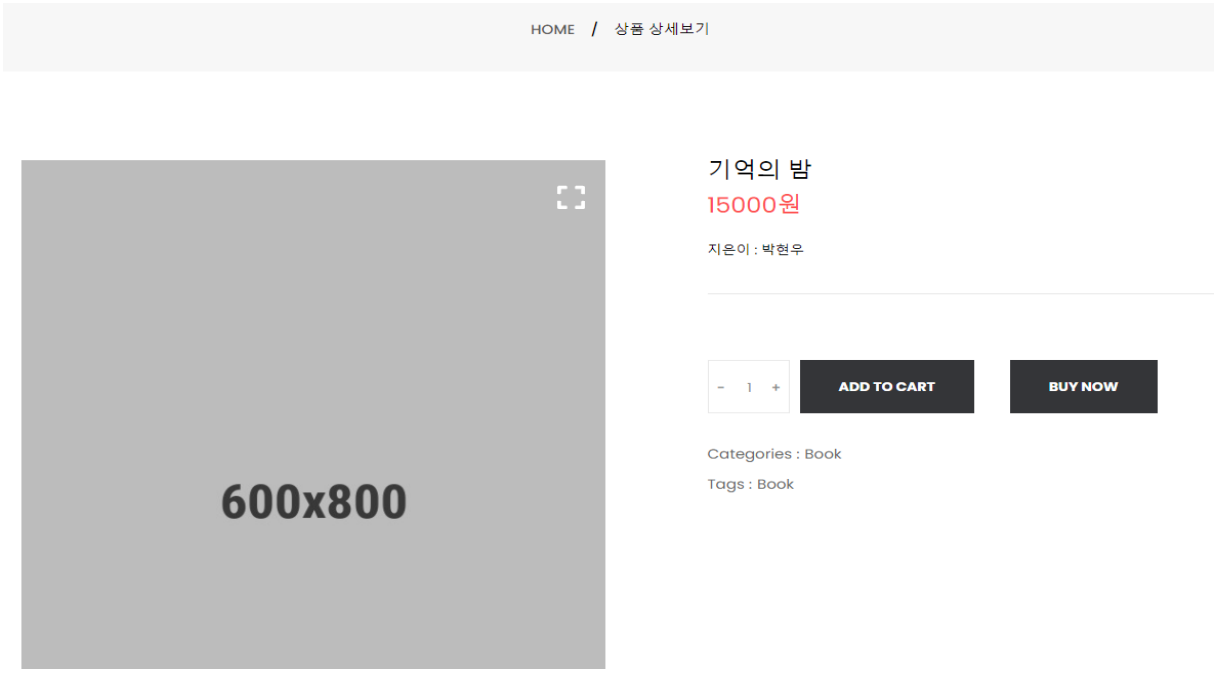
■ **로그인** - 가입한 이메일과 비밀번호로 로그인을 시도합니다. 실패 시 '로그인 정보를 확인하세요' 라는 알림창이 뜨고, 성공한 경우 메인 화면으로 돌아오며, 오른쪽 상단 헤더에 자신의 이메일이 표시됩니다.



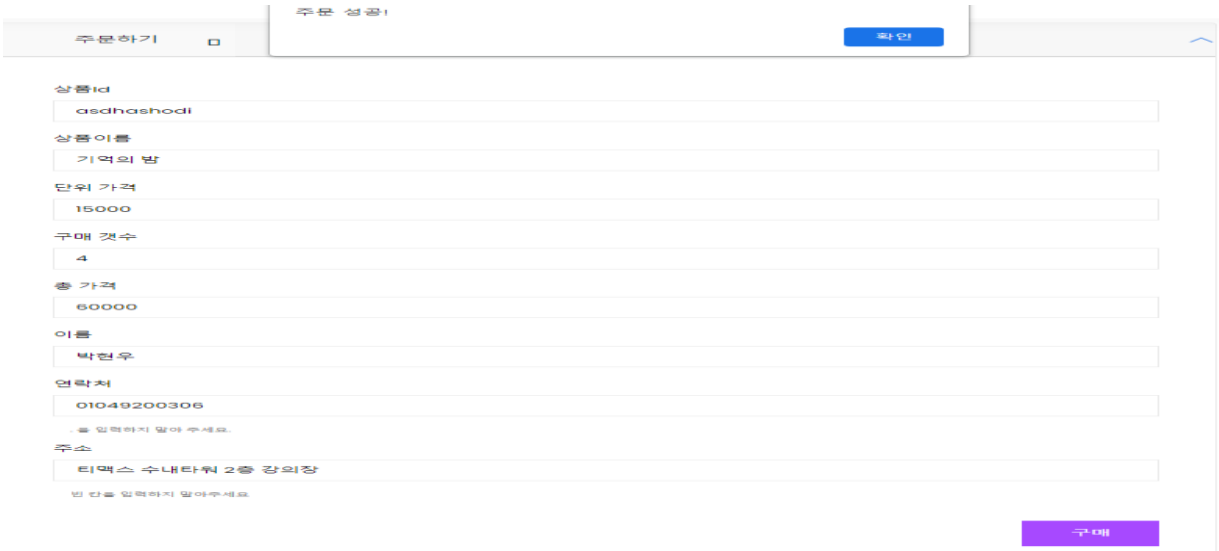
■ **상품 전체 보기**



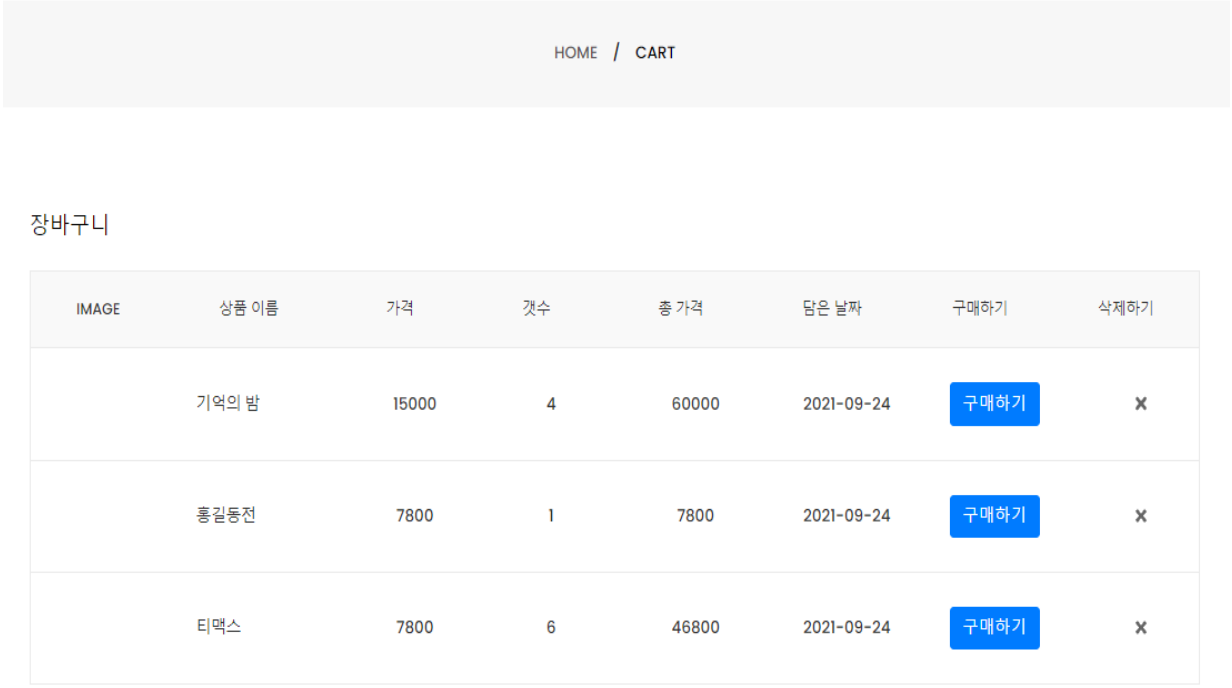
■ **상품 상세 화면** - 수량을 버튼으로 조정할 수 있습니다. ADD TO CART 버튼을 클릭하면 장바구니에 담기게 되고, BUY NOW 버튼을 클릭하면 구매 페이지로 넘어가게 됩니다.



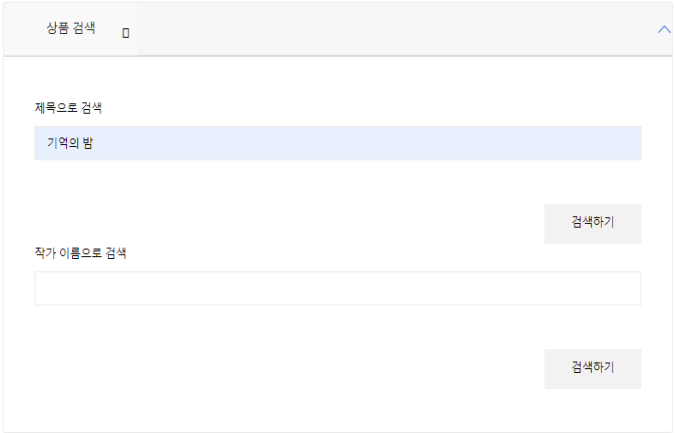
■ **상품 주문하기** - 앞 화면에서 BUY NOW 버튼을 클릭했을 때의 화면입니다. useEffect 함수를 통하여 상세목록의 옵션들이 불러와지고, 개인정보를 입력한 뒤 주문하기 버튼을 클릭하면 주문이 완료됩니다. 아래는 주문이 성공한 화면입니다.

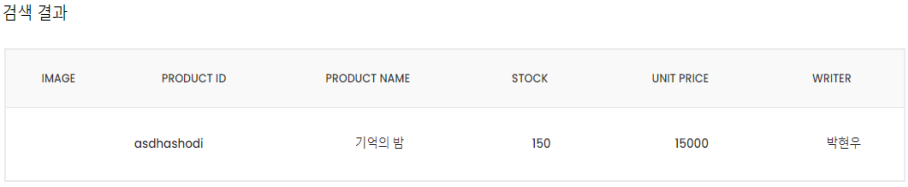


■ **장바구니** - 장바구니에 상품을 담을 수 있으며, 구매하기 버튼을 누르면 3.2.5 의 화면처럼 구매하기 페이지로 이동하고 정보를 불러옵니다.

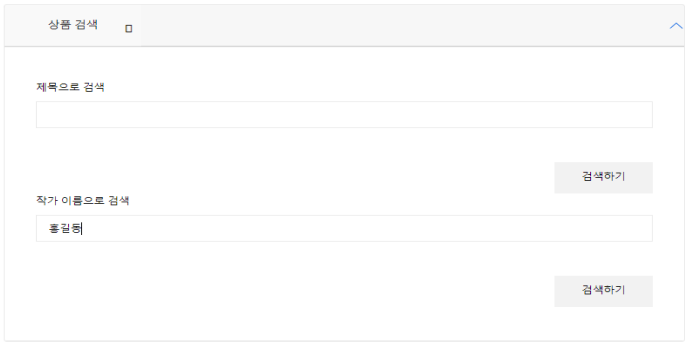


■ **상품 검색 기능** - 상품을 상품명과 작가 이름으로 검색할 수 있습니다. 각각 상품명, 작가 이름으로 검색한 결과입니다.





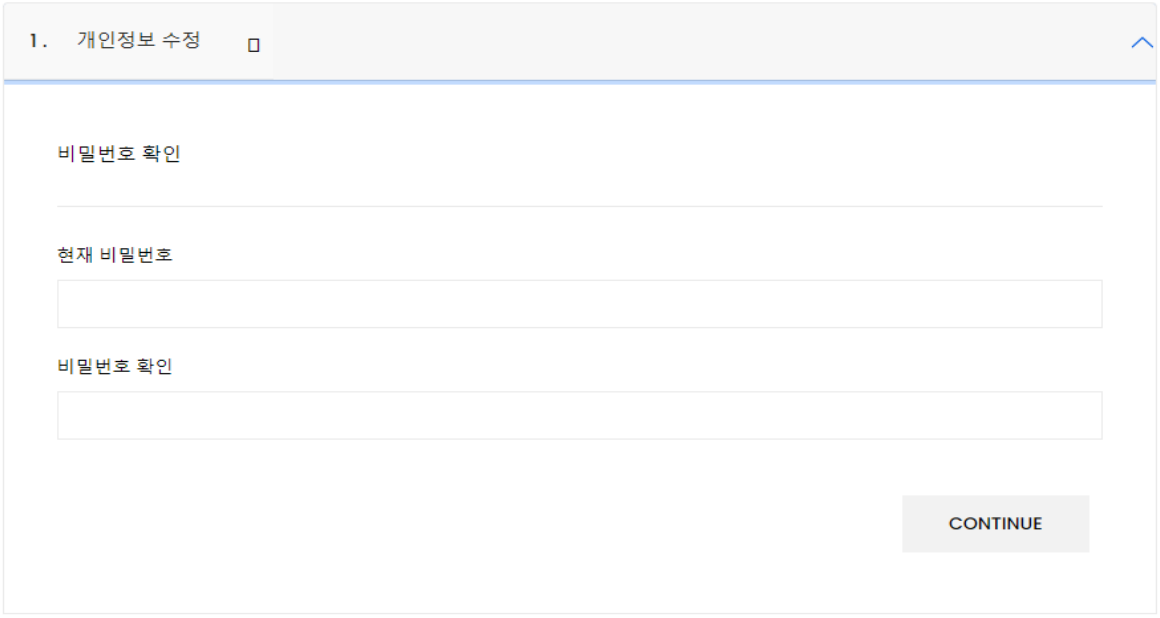
- 이번에는 작가 이름으로 검색해 보았습니다. 홍길동을 입력하고 검색하기를 누르니 아래와 같은 결과가 출력됩니다.



검색 결과

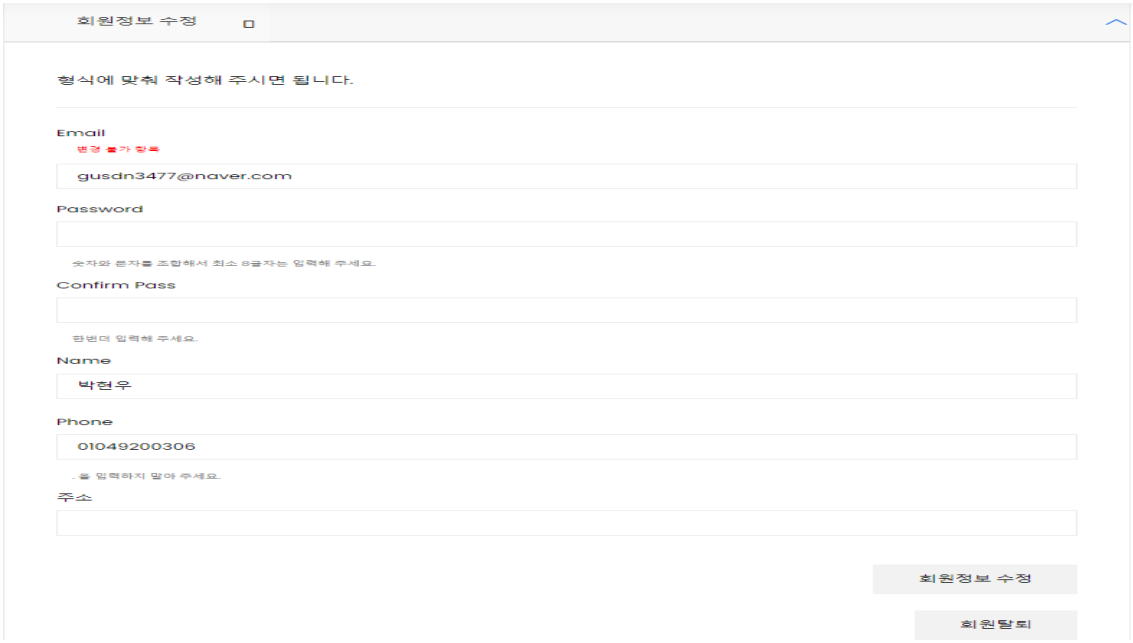
IMAGE	PRODUCT ID	PRODUCT NAME	STOCK	UNIT PRICE	WRITER
	a1b2c3	티맥스	1000	1000	홍길동

■ **회원정보 수정** - 비밀번호 확인 후에 회원정보 수정 페이지로 이동할 수 있습니다.



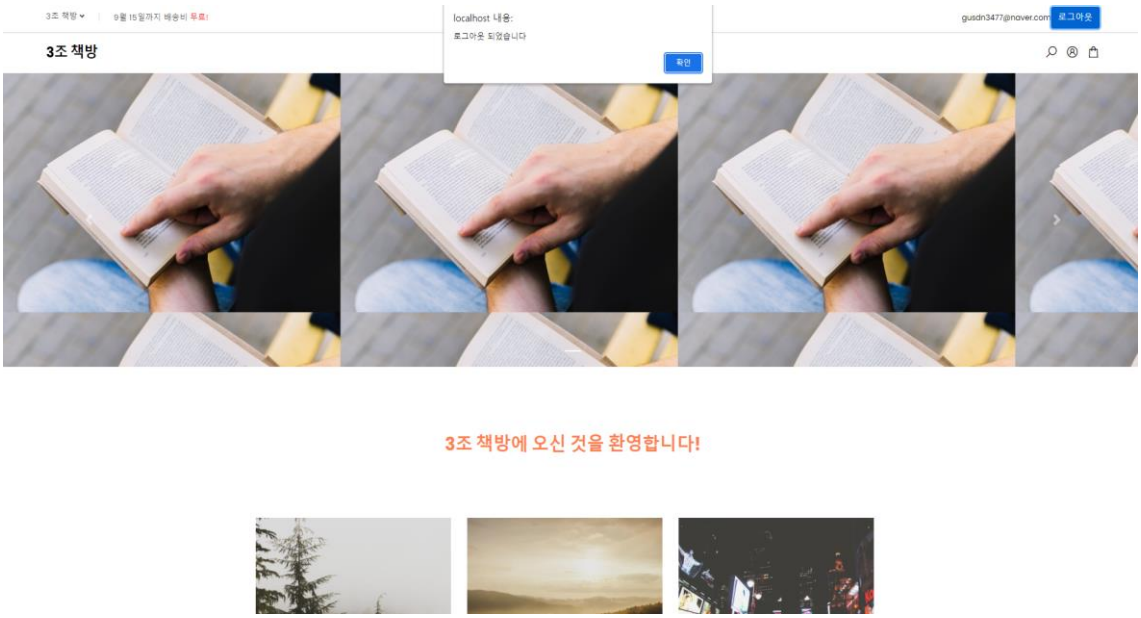
로그인 페이지와 마찬가지로 비밀번호가 일치한 경우 회원정보 수정 페이지로 넘어올 수 있습니다. 현재 회원정보가 useEffect 함수를 통하여 세팅되며, Email 항목은 변경할 수 없으며. (readonly) 나머지 항목들은 변경할 수 있습니다.

Password, Confirm Pass 를 현재 비밀번호와 다르게 설정하면 비밀번호를 변경할 수 있으며, 현재 비밀번호를 유지하고 싶은 경우는 현재 비밀번호를 입력하면 됩니다. 회원정보 수정 버튼을 누르면 수정 절차가 완료됩니다.



■ **회원 탈퇴** – 위의 사진에서 우측 하단에 있는 회원탈퇴 버튼을 클릭하면 탈퇴되며, 자동으로 로그아웃 후 메인 페이지로 이동합니다(DB 와 localStorage 내용을 둘 다 지웁니다.).

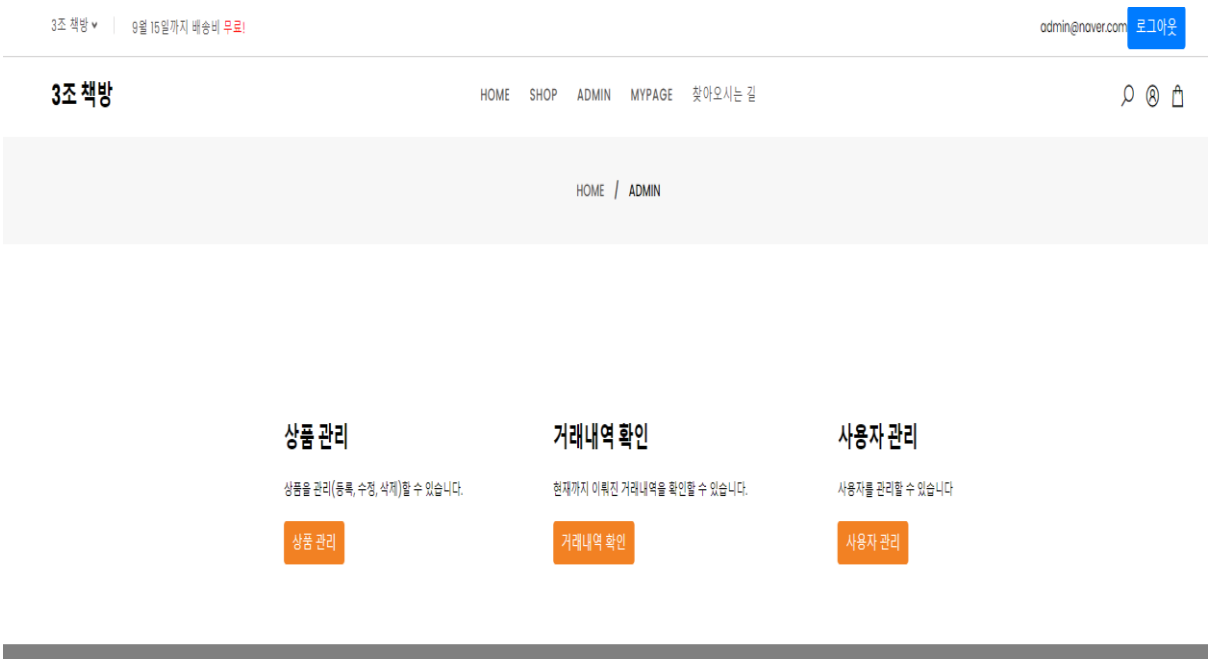
■ **로그아웃** - 로그아웃 버튼을 클릭하게 되면 localStorage 의 정보가 지워지고 메인 화면으로 돌아오게 됩니다.



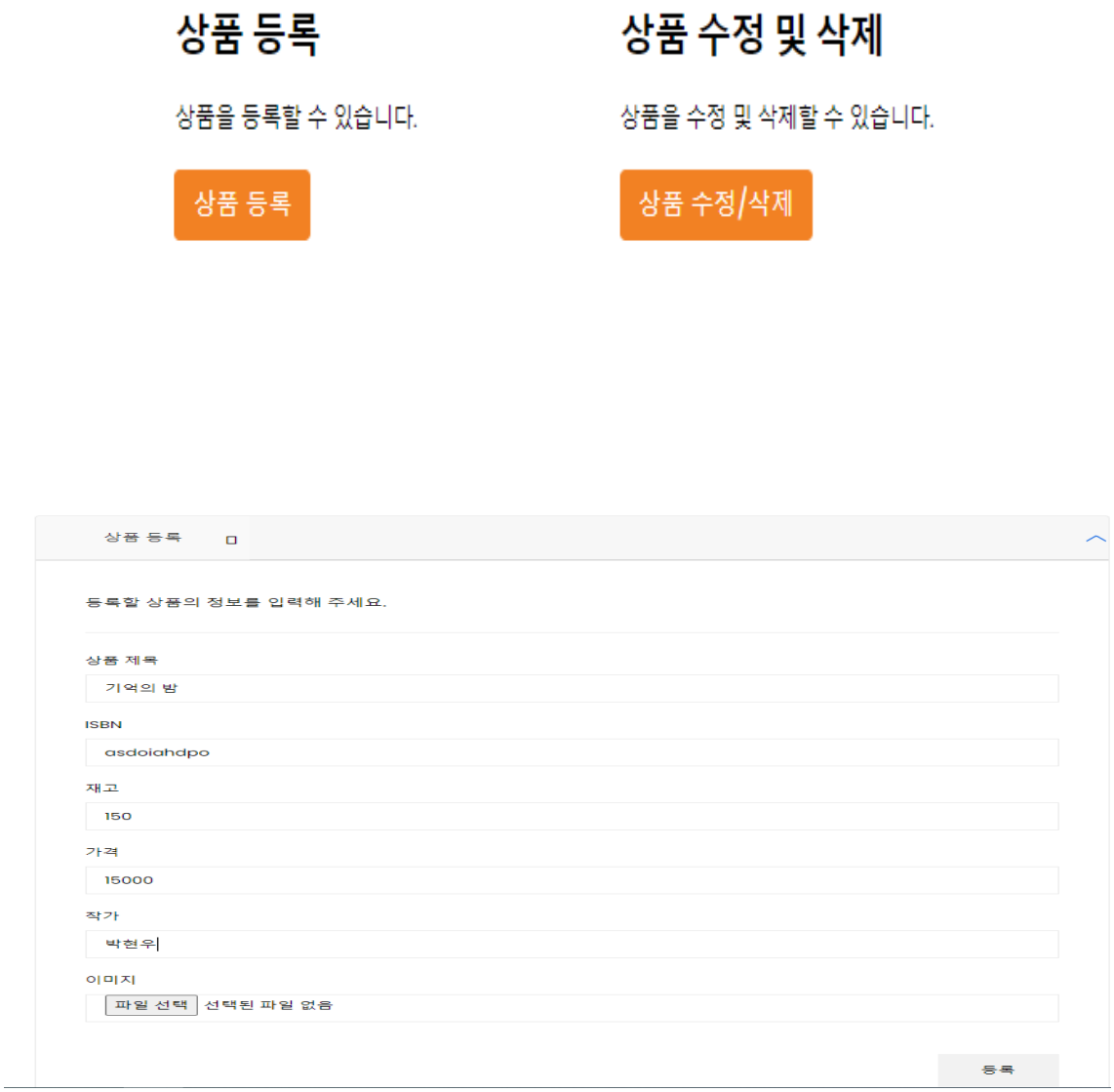
○ 관리자 매뉴얼

■ **관리자 로그인** - admin 으로 시작하는 이메일 계정으로 가입 후 로그인 하게 되면 관리자 로그인으로 처리됩니다. (admin@naver.com 등과 같이 앞 글자 5 자리가 admin 이면 admin 이라는 값을 localStorage 에 저장하고 이용합니다.)

■ **관리자모드** - 관리자로 로그인하게 되면 메뉴 바에 ADMIN 이라는 항목이 생깁니다. ADMIN 항목을 클릭하면 상품 관리, 거래내역 확인, 사용자 관리 기능이 가능합니다.



■ **상품 관리** - 상품 관리로 들어가면 상품 등록, 상품 수정 및 삭제 기능을 이용할 수 있습니다. 상품 등록 버튼을 클릭하면, 등록할 상품에 대한 정보를 입력할 수 있습니다. 제목, ISBN, 재고, 가격, 작가 이름을 입력하고 등록하게 되면 새 상품이 등록됩니다.

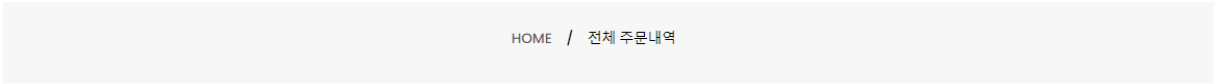


■ **상품 수정 및 삭제 기능**에선 현재 있는 상품들을 삭제할 수 있습니다.

상품에 대해 수정 및 삭제를 할 수 있습니다.

사진	책 제목	작가	가격	갯수	총 가격	등록 일자	삭제하기
	기억의 밤	박현우	15000	- 150 +	2250000	2021-09-24	×
	홍길동전	홍길동	7800	- 10000 +	78000000	2021-09-24	×
	티맥스	티맥스 클라우드	7800	- 14312 +	111633600	2021-09-24	×

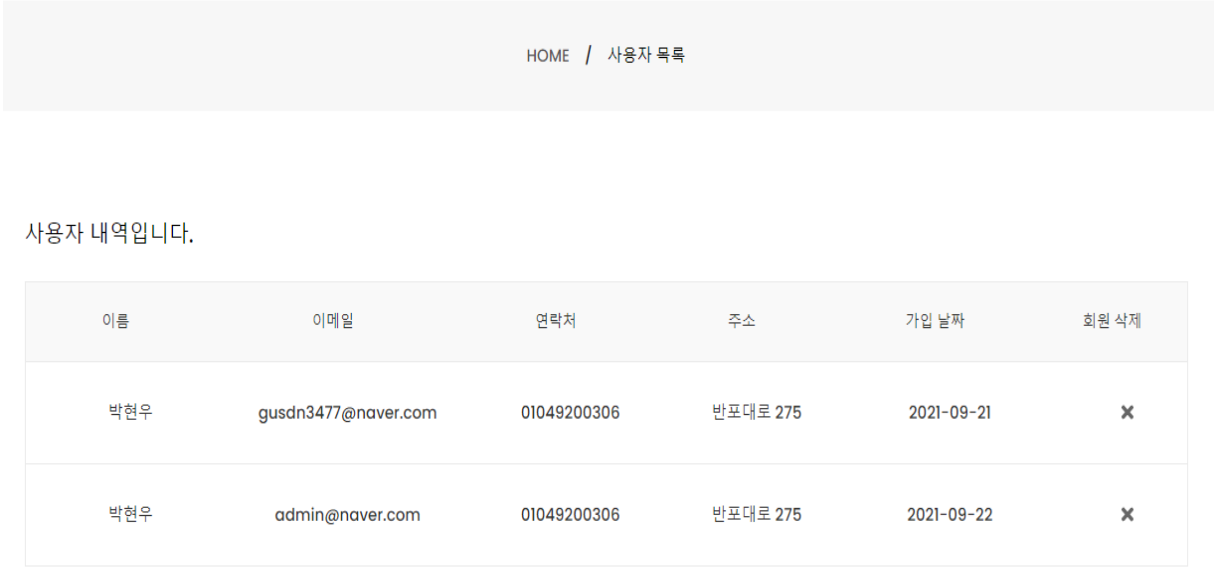
■ **주문 내역**을 클릭하게 되면 지금까지 주문된 항목들을 리스트로 볼 수 있습니다.



전체 주문내역 입니다.

사진	주문번호	가격	갯수	총 가격	주문 일자
	9b3a9ce4-5333-4c9d-b7bd-685e14d00052	15000	1	15000	2021-09-24
	20d89b68-3501-4eca-8165-0b721352b37c	7800	4	31200	2021-09-24
	59a1ef70-ac25-4825-83b5-ae39b085af38	7800	13	101400	2021-09-24

■ **사용자 관리**를 클릭하게 되면, 사용자 목록을 볼 수 있습니다. 가입한 회원들의 목록을 볼 수 있으며, 회원을 삭제할 수 있습니다.



6. 모듈프로젝트 소감

성명	모듈 프로젝트 후 소감
권진희	프로그램을 어느 환경에서 실행하는가에 따라 변경되는 값들을 효율적으로 관리하기 위해 config service 를 적절히 사용하는 것이 효율적임을 직접 느꼈고 다음 프로젝트에 필요하다면 적절히 잘 활용해야겠다 다짐했습니다. 일정 관리를 더 효율적으로 하기 위해서는 초석을 잘 쌓는 것이 매우 중요함을 체감했습니다. MS 간의 통신연결이 생각보다 어려운 점이 많았습니다. 더 다양한 것들을 배워 효과적인 방식을 선택하여 다음 프로젝트에 적용할 수 있게 되면 좋겠다고 생각하였습니다. 비대면 진행인 상황이지만 대면 모임 시 효율이 더 좋았다고 느꼈습니다. 비대면으로도 높은 효율을 낼 수 있는 방법을 다 같이 고민해 봐야겠다 생각하였습니다. 부족한 점이 많았지만 이번을 통해 깨달은 것들을 잘 적용하여 더 나은 결과를 낼 수 있을 것 같아 다음 프로젝트가 기대됩니다.
김영모	문서 작업부터 시작해서 개발의 전반적인 과정을 진행하며 설계 작업의 중요성을 깨달았습니다. 메인 프로젝트에서 향상된 방향으로 적절히 적용할 것 입니다. 개발을 진행하며 배운 내용을 상기하고, 구체화 할 수 있는 좋은 경험이었습니다. 또한, 부족한 부분도 지각할 수 있는 경험이었습니다. 메인 프로젝트 이전에 부족한 부분을 보완하여 팀에 더 많은 도움을 주고 싶고, 프로젝트를 성공적으로 진행할 수 있도록 할 것 입니다.
박현우	지금까지 배운 것들을 간단한 모듈 프로젝트를 통하여 온전히 익힐 수 있는 기회가 되었습니다. 개별로 배웠던 지식들을 프로젝트를 진행하는 과정에서 유기적으로 통합시킬 수 있었고, 결과적으로 정돈되지 않았던 지식들을 조직화하는 기회가 되었습니다. 추가로 일정 조율, 문서 작업 등과 같은 개발을 시작하기 전에 하는 여러 작업들을 직접 주도적으로 경험해 보며 많은 것을 배우게 되었습니다. 모듈 프로젝트 과정에서 배운 여러 경험들을 토대로 메인 프로젝트 시에 더 효율적인 방식으로 프로젝트를 진행할 예정입니다.

윤희상	개인적으로 아쉬움이 많이 남는 프로젝트였습니다. 먼저 개념의 이해와 실습 과정에서 적용 사이의 괴리감이 존재했고, 프로젝트를 진행하며 스스로 부족한 점이 많다는 것을 느꼈습니다. 이후 작업할 메인 프로젝트에서 미숙했던 부분을 보완할 예정입니다. 팀 단위 작업에서 협업 시 일정관리의 중요성 또한 다시금 깨닫게 되었습니다. 비대면으로 진행하며 커뮤니케이션 과정에서 일정 부분 비효율적인 측면이 있었고, 과업 분담에서 차질이 발생한 경우 유연하게 대처하지 못했던 아쉬움이 있습니다. 본 미니 프로젝트를 반면교사 삼아, 메인 프로젝트 진행 시 팀의 구성원으로서 분명한 기여를 할 수 있도록 노력할 것입니다.