Portfolio

SCM/ 자동화와 생산성을 고민하는
개발자 김유석 입니다.

B 1994.01.27

P 010 3582 8143

M itgorae.seok@gmail.com

G github.com/yuseok01

EXPERIENCE

2017.12 GTEP 사업 수료/ 산업통상자원부

2020.08 인제대학교 졸업(4.17/4.5)
국제통상학 / 정치외교학

2021.04 롯데 글로벌로지스 입사
SCM 사업본부 / 유니클로 물류
반품 물류 담당, 도급 관리

2022.10 한일 무역전쟁 TFT 팀 파견
5,000평 임시 센터 운영
고객사 / 도급사 관리 및 대응
RFID / DAS 운영

2024.06 삼성 청년 SW 아카데미 이수
Java, Spring Boot, Vue

2024.06 축구 커뮤니티 프로젝트
Spring Boot , Vue, MySQL

2024.08 WEB WMS 프로젝트
Spring Boot, JWT, REACT

2024.09 무인 매장 관리 프로젝트
Spring Boot, Electron, Next

2024.10 AI Excel, CSV 분석 모듈 (진행중)
Spring Boot, MongoDB

CERTIFICATION

2016.09 국제무역사 2급

2020.08 2종 보통 운전 면허

2021.03 컴퓨터 활용 2급

2023.10 물류관리사

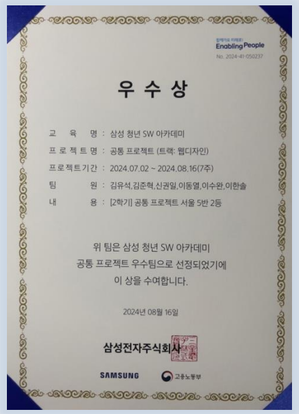
2024.09 SQLD

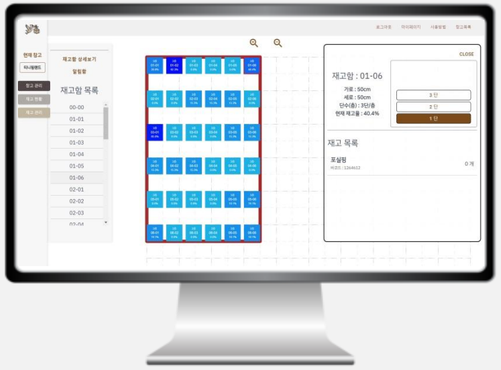
PROGRAM SKILL

Java, spring, JPA, MySQL, React, NEXT.JS, Vue.js, Jira, git

WMS	90%
MS office	80%
DAS	80%

Project: FitBOX





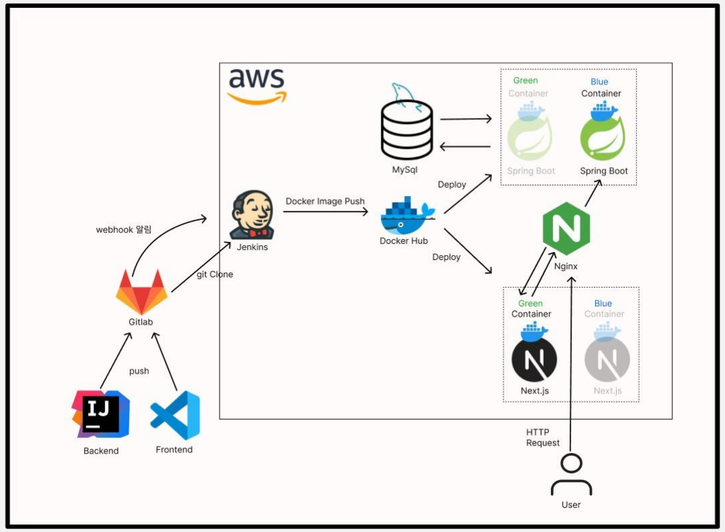
FITBOX
WEB WMS Project

- **창고 재고 관리 서비스**
  - 24/07/08 ~ 24/08/16 (7주)
  - 참여인원 6명 / 팀장
  - 삼성 전자 우수 프로젝트 수상
- **프로젝트 주요 특징**
  - 상품 피킹 / 보관 구역 구현
  - 다중 창고 생성 기능 구현
  - 엑셀 대량 상품 등록 구현
  - 창고 도면 시각화

Back End
- 프레임워크: Spring Boot(3.3.1)
- JVM 버전: Azul Zulu 17.0.11
- 빌드도구: Gradle

Front End
- 프레임워크: Next.js(14.x)
- 빌드도구: npm

Infra
- 웹 서버: AWS EC2
- CI/CD: Jenkins
- Nginx Blue & Green 라우팅



Assigned Role
- 창고 관련 logic 개발(Spring Boot JPA)
- User 관련 logic 개발(Spring Boot JPA)
- 자체 로그인 및 비밀번호 암호화(Spring Boot)
- OAuth 소셜 로그인 기능 구현(Spring Security)
- Email 인증 (Email Provider)
- 회원 가입, 로그인 페이지 제작
- JWT 토큰 관리(Jwt Provider)

Problem
00-00-0의 로케이션을 도면으로 랜더링하고자
X,Y 축(00-00)과 Z 축(0)을 분리 하였고,

사업체-> 창고 -> 로케이션(00-00) -> 층(0) -> 상품
로 테이블을 구성하였습니다.
그로 인해 테이블 조회 관련 다음과 같은 문제 있었습니다.

- N+1 문제 (쿼리 중첩)
- 상품 조회 시 5번의 Join으로 속도 저하

Project: FitBOX_Trouble Shooting

문제 상황

1. N+1

Location을 Warehouse로 조회할 때
N번의 개별 조회 쿼리가 조회되는 문제

2. 속도 개선

사업체 -> 창고 -> 로케이션(00-00) -> 단(0) -> 상품으로 테이블 구성

상품을 조회하기 위해선 5번의 Join이 발생하여 속도 개선이 필요한 상황

Ex) 1개 사업체 -> 3개 창고 -> 1,000로케이션-> 4,000단 -> 4,000개 상품을 가정했을 때 개선 속도

500ms -> **150ms** -> **50ms**

Indexing 반정규화

문제 해결

1-1. 일대다 관계에서 기본 설정 변경

FetchType.LAZY

```
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "warehouse_id", nullable = false)
private Warehouse warehouse;
```

1-2. 특정 쿼리에서 연관된 테이블을 한 번에 가져오도록 설정

Fetch join

```
@Query("SELECT l FROM Location l "  1 usage  rnjsdlf9302
    + "JOIN FETCH l.warehouse w "
    + "WHERE w.id = :warehouseId")
List<Location> findAllByWarehouseId(Long warehouseId);
```

2-1. 삽입이 적은 창고테이블에 Indexing 적용

Indexing

```
@Entity  김유석
@Getter
@Setter
@SQLRestriction("status_enum = 'Active'")
@Table(name = "warehouse", indexes = {
    @Index(name = "idx_business_id", columnList = "business_id")
})
```

2-2 반정규화로 Floor 테이블에 로케이션ID 기록 **반정규화**

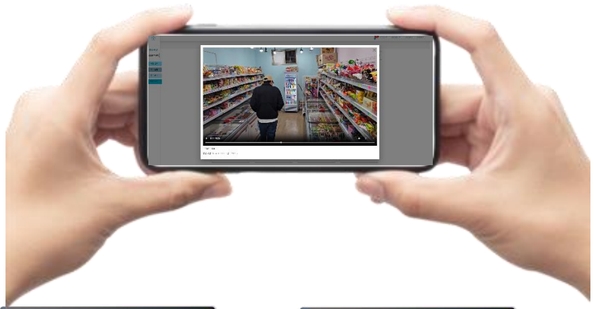
```
/**
 * @param warehouseID
 * @return
 * 창고 상품을 조회하는 쿼리 (반정규화된 필드를 사용)
 */
@Query("SELECT p FROM Product p " +  2 usages
    "JOIN p.floor f " +
    "WHERE f.warehouseId = :warehouseID")
List<Product> findByWarehouseId
(@Param("warehouseID") Long warehouseID);
```

느낀 점

1. JPA의 기본 동작 방식인 지연 로딩(LAZY) 때문에 발생하는 N+1 문제를 이해하게 되었고, 이를 해결하기 위해 Fetch Join, BatchSize, 그리고 EntityGraph 등의 다양한 전략을 상황에 맞게 사용할 수 있다는 것을 배웠습니다.

2. 테이블 설계 시 데이터 조회 성능을 고려해야 한다는 점을 깨달았습니다. 또한, JUnit을 활용하여 성능을 모니터링하고 개선점을 찾는 것이 중요하다는 것을 느꼈습니다.

Project: AutoStore



AutoStore
무인매장관리 시스템

- **무인 매장 관리 시스템**
  - 24/09/19 ~ 24/10/10 (8주)
  - 참여인원 6명 / 팀장
- **프로젝트 주요 특징**
  - RIFD/NFC를 활용한 자동 상품인식
  - 키오스크 exe 파일 배포 환경
  - ML 모델을 활용한 수요예측
  - TorchServe로 학습된 AI 모델
  - CCTV 이상 감지(도난,파손,방화 등)

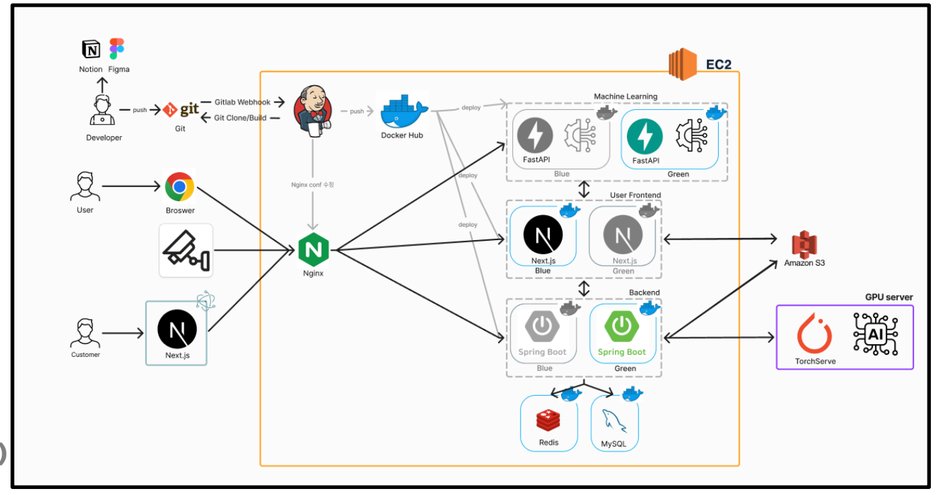
Back End
- 프레임워크: Spring Boot (3.3.1)
- JVM 버전: Azul Zulu 17.0.11
- 빌드도구: Gradle

Front End(PWA)
- 프레임워크: Next.js (14.2.13)
- 빌드도구: npm (10.8.2)

Front End(Electron)
- 프레임워크: Electron (25.0.0)
- 빌드 도구: electron-builder

Machine Learning
- 프레임워크: FastAPI
- Python 버전: 3.12.5
- 모델: PyTorch
- Nginx: Blue-Green 무중단 배포

CCTV 영상 분석
- 모델: TensorFlow
- 데이터셋: AI 이상 행동 데이터
- Nginx: Blue-Green 무중단 배포



Assigned Role
- 키오스크 관리 OTP 생성 (Spring Boot JPA)
- RFID / NFC 칩 상품 자동 인식(PC/SC 라이브러리)
- 키오스크 CPU 배포 환경 구축(Electron)
- 키오스크 화면 구성(Next.js)
- TOSS 결제 API(Next.js)

Problem
- NFC 칩 인코딩 및 상품 자동 인식
- Electron EXE 파일 배포 환경 구축
- Spring Boot 키오스크 OTP 보안 문제

Project: AutoStore_Trouble Shooting

문제 상황

1. 매장 사장님이 생성한 OTP가 영구히 DB에 남아 보안에 취약하다고 생각하여 개선

2. 인코딩한 NFC 칩을 리더기가 상품 인식 하지 못하는 현상

3. 불필요한 서비 비용을 감소하기 위하여 개별 CPU에서 돌아가는 키오스크를 exe 파일로 배포 시도

로컬에서 Index.html 파일을 찾지 못하는 문제

문제 해결

1. Redis와 Random OTP 생성을 결합하여 안전하고 제한된 시간 동안 유효한 OTP를 생성하고 관리하는 방식으로 변경 **Redis + OTP TTL**

```
static final int OTP_TTL = 5 * 60;  1 usage
private final StringRedisTemplate redisTemplate;
public String createDeviceOtp(Long deviceId) {
    log.info("[Service] create device otp");
    try {
        // TODO: 사용자 검증
        String otpString = createRandomOTP();
        String deviceIdString = deviceId.toString();

        ValueOperations<String, String> ops = redisTemplate.opsForValue();
        ops.set(otpString, deviceIdString, OTP_TTL, TimeUnit.SECONDS); // redis s
        return otpString;
    } catch (Exception e) {
        log.error("[Service] create device otp error {}", e.getMessage());
        throw new CommonException(ResponseEnum.DATABASE_ERROR, e.getMessage());
    }
}
```

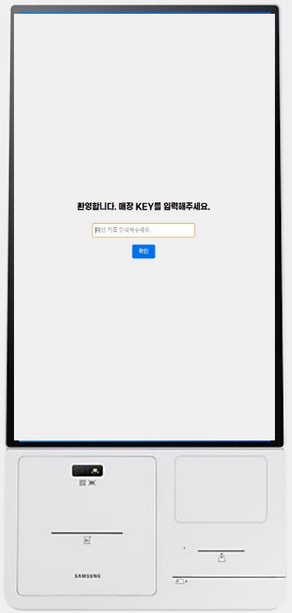
2. NFC 칩 인코딩된 값은 HEX 코드로 4번째 구역에 저장되어 있음을 알게 되었고, 모든 블록을 순회하며 console.log를 사용해 해당 데이터를 출력하여 확인했습니다. 해당 값을 ASCII 형식으로 변환한 후 추출하여 상품인식 하였습니다.

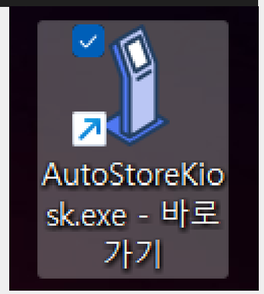
```
function readBlock(reader, protocol, blockNumber) {
  const command = Buffer.from([0xff, 0xb0, 0x00, blockNumber, 0x10]);
```

```
// 특정 데이터 패턴 찾기
if (numericData && numericData.length > 0) {
  console.log(
    `Valid barcode found in block ${blockNumber}: ${numericData}`
  );
  mainWindow.webContents.send("nfc-data", numericData);
  return; // 원하는 데이터를 찾았으므로 종료
}
```

3. index.html 파일과 필요한 모든 리소스를 같이 패키징 및 로컬 파일 경로 수동 설정

```
  const startPath = path.join("C:", "kiosk", "dist", "index.html");
  mainWindow.loadFile(startPath);
}
```





THANK YOU

끝까지 봐주셔서 감사합니다.
끊임없이 도달하는 개발자 김유석의 포트폴리오 였습니다.

CONTACT

Phone Number |
010 3582 8143

e-mail |
itgorae.seok@gmail.com

Git-hub |
github.com/yuseok01