6조 - 3K 프로젝트 발표

음식 주문 관리 플랫폼 개발

Kang Tae Won
Kim Min Chul
Kim Ji Hee

MIRI COMPANY
MIRI GROUP

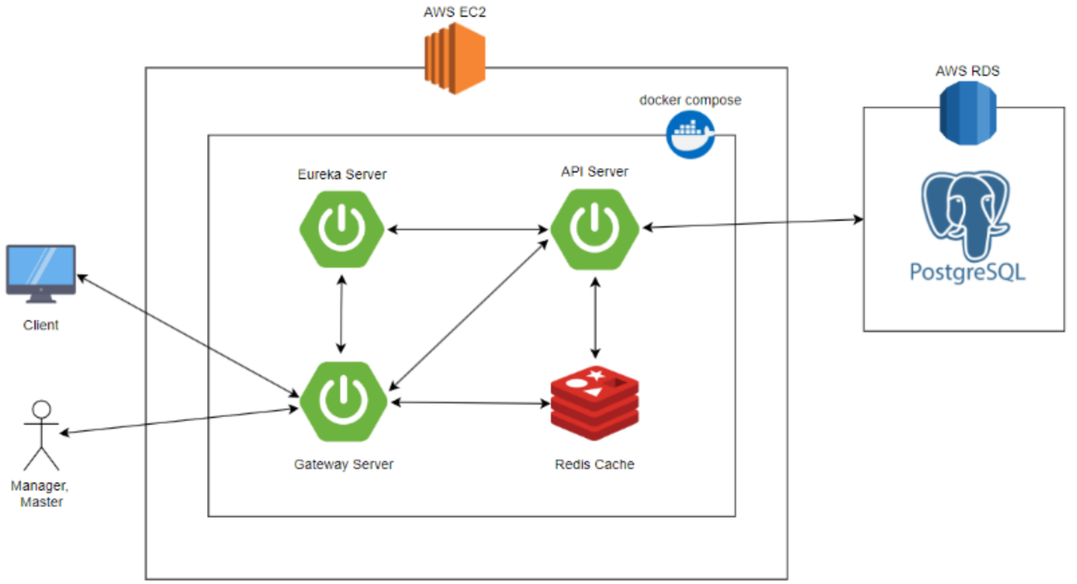
프로젝트 소개

사용 기술 스택

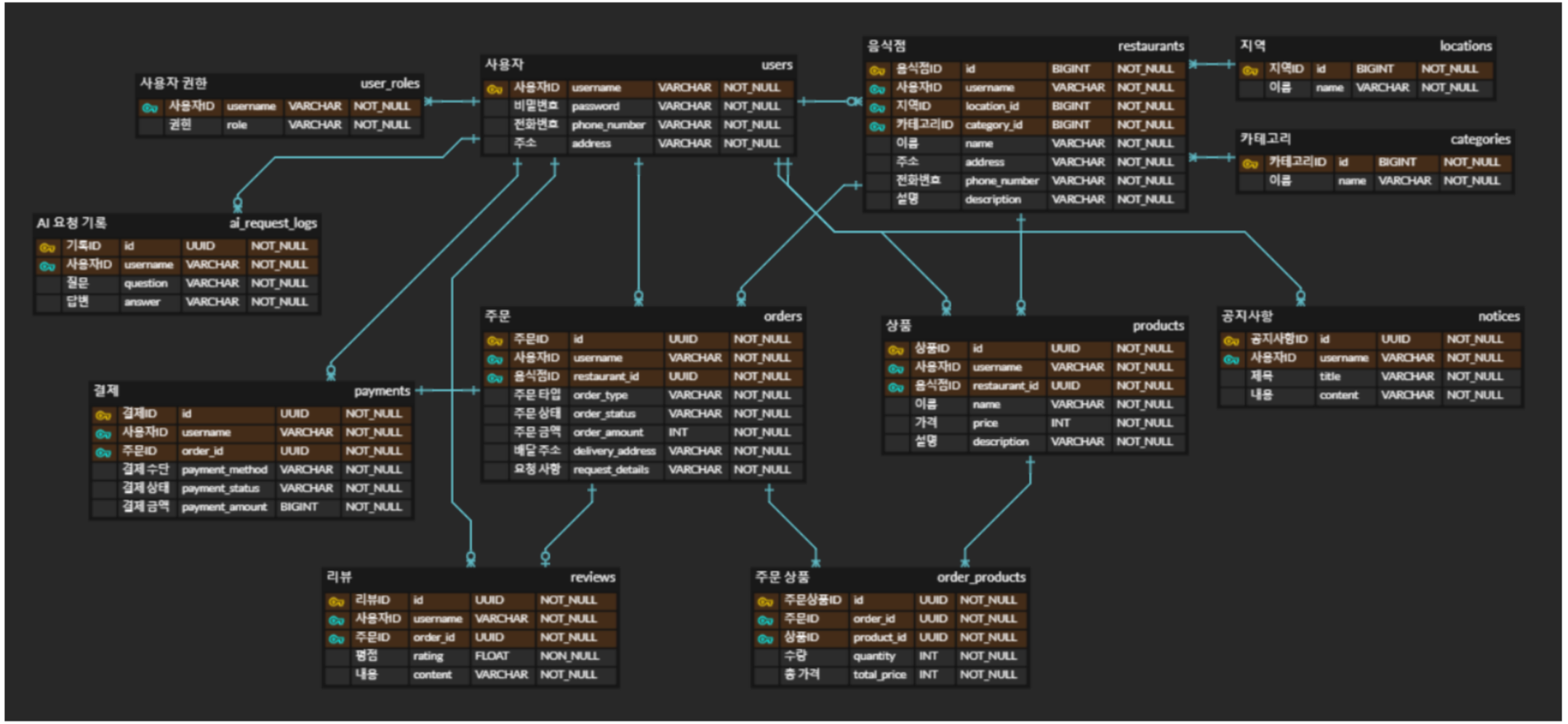
- Spring Boot
- Spring Data JPA, QueryDSL
- Spring Security, JWT
- Spring Cloud
- Swagger

- PostgreSQL, Redis

인프라 아키텍처



ERD


사용자 권한 user_roles
사용자ID username VARCHAR NOT_NULL
권한 role VARCHAR NOT_NULL
사용자 users
사용자ID username VARCHAR NOT_NULL
비밀번호 password VARCHAR NOT_NULL
전화번호 phone_number VARCHAR NOT_NULL
주소 address VARCHAR NOT_NULL
음식점 restaurants
음식점ID id BIGINT NOT_NULL
사용자ID username VARCHAR NOT_NULL
지역ID location_id BIGINT NOT_NULL
카테고리ID category_id BIGINT NOT_NULL
이름 name VARCHAR NOT_NULL
주소 address VARCHAR NOT_NULL
전화번호 phone_number VARCHAR NOT_NULL
설명 description VARCHAR NOT_NULL
지역 locations
지역ID id BIGINT NOT_NULL
이름 name VARCHAR NOT_NULL
카테고리 categories
카테고리ID id BIGINT NOT_NULL
이름 name VARCHAR NOT_NULL
AI 요청 기록 ai_request_logs
기록ID id UUID NOT_NULL
사용자ID username VARCHAR NOT_NULL
질문 question VARCHAR NOT_NULL
답변 answer VARCHAR NOT_NULL
주문 orders
주문ID id UUID NOT_NULL
사용자ID username VARCHAR NOT_NULL
음식점ID restaurant_id UUID NOT_NULL
주문타입 order_type VARCHAR NOT_NULL
주문상태 order_status VARCHAR NOT_NULL
주문금액 order_amount INT NOT_NULL
배달주소 delivery_address VARCHAR NOT_NULL
요청사항 request_details VARCHAR NOT_NULL
상품 products
상품ID id UUID NOT_NULL
사용자ID username VARCHAR NOT_NULL
음식점ID restaurant_id UUID NOT_NULL
이름 name VARCHAR NOT_NULL
가격 price INT NOT_NULL
설명 description VARCHAR NOT_NULL
공지사항 notices
공지사항ID id UUID NOT_NULL
사용자ID username VARCHAR NOT_NULL
제목 title VARCHAR NOT_NULL
내용 content VARCHAR NOT_NULL
결제 payments
결제ID id UUID NOT_NULL
사용자ID username VARCHAR NOT_NULL
주문ID order_id UUID NOT_NULL
결제수단 payment_method VARCHAR NOT_NULL
결제상태 payment_status VARCHAR NOT_NULL
결제금액 payment_amount BIGINT NOT_NULL
리뷰 reviews
리뷰ID id UUID NOT_NULL
사용자ID username VARCHAR NOT_NULL
주문ID order_id UUID NOT_NULL
평점 rating FLOAT NON_NULL
내용 content VARCHAR NOT_NULL
주문 상품 order_products
주문상품ID id UUID NOT_NULL
주문ID order_id UUID NOT_NULL
상품ID product_id UUID NOT_NULL
수량 quantity INT NOT_NULL
총 가격 total_price INT NOT_NULL

API 명세

/api/public/**	ex: 회원 가입, 로그인
/api/master/**	ex: 권한 부여, 회수
/api/admin/**	ex: 회원 정보 조회
/api/owner/**	ex: 가게 주문 조회

역할 분배

강태원
- Spring Security, User, AI 도메인 개발
- User 캐싱, Gateway 인증 필터 개발

김민철
- Restaurant, Product 도메인 개발
- 리뷰 도메인 개발

김지희
- Auditing, Order, Payment 도메인 개발
- 공지사항 도메인 개발, 프로젝트 배포

고민했던 점

강태원

- 사용자가 여러 개의 권한을 가질 수 있게 Collection Table 사용하였다.
- Redis에 저장한 사용자 데이터를 Gateway와 API 서버 둘 다 사용할 수 있게 개발하였다.
- 어떻게 하면 Gateway에서 API 경로를 깔끔하게 처리할 수 있을지 고민하였다.

김민철

- 테스트 코드를 작성 할 때 mock 객체를 어느 정도로 사용하는게 바람직한지 고민하였다.
- 객체지향 개발과 효율적인 코드가 어떤 것인지 고민해보았다.

김지희

- Auditing에 soft delete를 어떻게 구현할지 고민했다.
- 동적 쿼리를 위해 queryDSL을 사용했다.

Auditing 코드 - BaseEntity

```java
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
public class BaseEntity {
    @CreatedDate
    @Column(updatable = false)
    @Temporal(TemporalType.TIMESTAMP)
    private LocalDateTime createdAt;

    @CreatedBy
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "created_by")
    private User createdBy;

    @LastModifiedDate
    @Column(updatable = true)
    @Temporal(TemporalType.TIMESTAMP)
    private LocalDateTime updatedAt;

    @LastModifiedBy
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "modified_by")
    private User modifiedBy;
```

```java
    private LocalDateTime deletedAt;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "deleted_by")
    private User deletedBy;

    mii2026
    public void deleteBy(User user) {
        this.deletedAt = LocalDateTime.now();
        this.deletedBy = user;
    }
}
```

Auditing 코드 - UserAuditorAware

```
mii2026 +1
@Service
public class UserAuditorAware implements AuditorAware<User> {
    mii2026 +1
    @Override
    public Optional<User> getCurrentAuditor() {
        Authentication authentication = SecurityContextHolder.getContext().getAuthentication();

        if (authentication == null || !authentication.isAuthenticated() ||
                authentication instanceof AnonymousAuthenticationToken) {
            return Optional.empty();
        }

        return Optional.of( value: ((UserDetailsCustom) authentication.getPrincipal()).getUser());
    }
}
```

User Caching 코드 - UserDetailsService

```java
@Override
public UserDetailsCustom loadUserByUsername(String username) {
    User user = getUserFromCache(username);
    return new UserDetailsCustom(user);
}

rivertw777
private User getUserFromCache(String username) {
    User user = userCacheRepository.getUserCache(username);
    if (user == null) {
        user = userRepository.findByUsername(username)
                .orElseThrow( exceptionSupplier: () -> new ApplicationException(USER_NOT_FOUND.getValue()));
        userCacheRepository.setUserCache(username, user);
    }
    return user;
}
```

User Caching 코드 - UserCacheRepository

```java
@Slf4j
@Repository
@RequiredArgsConstructor
public class UserCacheRepository {

    private final String USER_CACHE_KEY_PREFIX = "UserCache:";

    private final RedisTemplate<String, User> redisTemplate;

    rivertw777
    public void setUserCache(String username, User user) {
        String key = USER_CACHE_KEY_PREFIX + username;
        redisTemplate.opsForValue().set(key, value: user, timeout: Duration.ofHours( hours: 1));
        log.info("User {} Cache Saved", username);
    }

    rivertw777
    public User getUserCache(String username) {
        String key = USER_CACHE_KEY_PREFIX + username;
        User user = redisTemplate.opsForValue().get(key);
        log.info("User {} Cache find", username);
        return user;
    }

    rivertw777
    public void deleteUserCache(String username) {
        String key = USER_CACHE_KEY_PREFIX + username;
        redisTemplate.delete(key);
        log.info("User {} Cache Deleted", username);
    }
}
```

Gateway 인증 코드 - SecurityConfig (변경 전)

```
rivertw777 *
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
            // JWT 인증
            .csrf( csrfCustomizer: AbstractHttpConfigurer::disable)
            .formLogin( formLoginCustomizer: AbstractHttpConfigurer::disable)
            .httpBasic( httpBasicCustomizer: AbstractHttpConfigurer::disable)
            .sessionManagement( sessionManagementCustomizer: AbstractHttpConfigurer::disable)
            .addFilterBefore(authenticationFilterCustom,  beforeFilter: UsernamePasswordAuthenticationFilter.class)
            // 예외 처리 핸들러
            .exceptionHandling( exceptionHandlingCustomizer: exception -> exception
                    .accessDeniedHandler(accessDeniedHandlerCustom)
                    .authenticationEntryPoint(authenticationEntryPointCustom)
            )
            .authorizeHttpRequests( authorizeHttpRequestsCustomizer: authz -> authz
                    // 회원 가입
                    .requestMatchers( ...requestMatchers: antMatcher( method: HttpMethod.POST,  pattern: "/api/users")).permitAll()
                    // 로그인
                    .requestMatchers( ...requestMatchers: antMatcher( method: HttpMethod.POST,  pattern: "/api/auth/login")).permitAll()
                    // MASTER API
                    .requestMatchers( ...patterns: "/api/master/**").hasRole( role: "MASTER")
                    // MANAGER API
                    .requestMatchers( ...patterns: "/api/admin/**").hasAnyRole( ...roles: "MANAGER", "MASTER")
                    // OWNER API
                    .requestMatchers( ...patterns: "/api/admin/**").hasAnyRole( ...roles: "OWNER", "MANAGER", "MASTER")
                    // Swagger
                    .requestMatchers( ...patterns: "/swagger-ui/**", "/swagger-resources/**", "/v3/api-docs/**").permitAll()
                    .anyRequest().authenticated()
            );
    return http.build();
}
```

Gateway 인증 코드 - SecurityConfig (변경 후)

```java
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
            // JWT 인증
            .csrf( csrfCustomizer: AbstractHttpConfigurer::disable)
            .formLogin( formLoginCustomizer: AbstractHttpConfigurer::disable)
            .httpBasic( httpBasicCustomizer: AbstractHttpConfigurer::disable)
            .sessionManagement( sessionManagementCustomizer: AbstractHttpConfigurer::disable)
            .addFilterBefore(authenticationFilterCustom,  beforeFilter: UsernamePasswordAuthenticationFilter.class)
            // 예외 처리 핸들러
            .exceptionHandling( exceptionHandlingCustomizer: exception -> exception
                    .accessDeniedHandler(accessDeniedHandlerCustom)
                    .authenticationEntryPoint(authenticationEntryPointCustom)
            )
            .authorizeHttpRequests( authorizeHttpRequestsCustomizer: authz -> authz
                    // public api
                    .requestMatchers( ...requestMatchers: antMatcher( pattern: "/api/public/**")).permitAll()
                    // Swagger
                    .requestMatchers( ...patterns: "/swagger-ui/**", "/swagger-resources/**", "/v3/api-docs/**").permitAll()
                    .anyRequest().authenticated()
            );
    return http.build();
}
```

Gateway 인증 코드 - GatewayConfig

```java
@Configuration
@RequiredArgsConstructor
public class GatewayConfig {

    private final JwtAuthorizationFilter jwtAuthorizationFilter;

    rivertw777 *
    @Bean
    public RouteLocator customRouteLocator(RouteLocatorBuilder builder) {
        return builder.routes()
                // public api
                .route( id: "api-server",  fn: r -> r.path( ...patterns: "/api/public/**")
                        .uri( uri: "lb://api-server"))
                // auth api
                .route( id: "api-server",  fn: r -> r.path( ...patterns: "/api/**") BooleanSpec
                        .filters( fn: f -> f.filter( gatewayFilter: (exchange, chain) ->
                                jwtAuthorizationFilter.filter(exchange, chain))) UriSpec
                        .uri( uri: "lb://api-server"))
                .build();
    }

}
```

Gateway 인증 코드 - JwtAuthorizationFilter

```java
public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
    String requestPath = exchange.getRequest().getURI().getPath();
    log.info("요청 API 경로: " + requestPath);

    String token = extractToken(exchange);
    try {
        // 토큰 확인
        if (token == null) {
            return unauthorizedResponse(exchange, errorMessage: ErrorMessage.TOKEN_NOT_FOUND);
        }

        // 토큰 검증
        if (tokenManager.validateToken(token)) {
            Claims claims = tokenManager.parseClaims(token);
            String username = claims.getSubject();
            UserCache user = userCacheRepository.getUserCache(username);

            // UserCache Null
            if (user == null) {
                return unauthorizedResponse(exchange, errorMessage: ErrorMessage.USER_NOT_FOUND);
            }
            List<Role> roles = user.getRoles();
```

Gateway 인증 코드 - JwtAuthorizationFilter

```
            // 권한 없음
            if (!isAuthorized(requestPath, roles)) {
                return unauthorizedResponse(exchange, errorMessage: ErrorMessage.FORBIDDEN);
            }

            // 헤더 추가
            ServerHttpRequest mutatedRequest = exchange.getRequest().mutate()
                    .header( headerName: HEADER_NAME, ...headerValues: username)
                    .build();
            ServerWebExchange mutatedExchange = exchange.mutate()
                    .request(mutatedRequest)
                    .build();
            return chain.filter(mutatedExchange);
        } else {
            return unauthorizedResponse(exchange, errorMessage:
        }
    } catch (Exception e) {
        return responseWriter.setErrorResponse(exchange, Err
                ErrorMessage.INTERNAL_SERVER_ERROR.getMessag
    }
}
```

rivertw777

```
private boolean isAuthorized(String requestPath, List<Role> roles) {
    if (requestPath.startsWith(ApiPrefix.OWNER.getValue())) {
        return roles.contains(Role.OWNER) || roles.contains(Role.MANAGER) || roles.contains(Role.MASTER);
    }
    if (requestPath.startsWith(ApiPrefix.ADMIN.getValue())) {
        return roles.contains(Role.MANAGER) || roles.contains(Role.MASTER);
    }
    if (requestPath.startsWith(ApiPrefix.MASTER.getValue())) {
        return roles.contains(Role.MASTER);
    }
    return true;
}
```

프로젝트 회고

태원

- Mock을 사용한 단위 테스트 작성법을 배워야겠다.
- 권한에 따른 인증 개발을 해볼 수 있어서 좋았다.
- API 경로는 패턴으로 처리하는게 간편하다.

지희

- API 설계, 클린 코드, 단위테스트에 대해 고민하고 논의할 수 있어서 좋았다.
- Auditing과 QueryDSL을 배울 수 있어서 좋았다.
- 이번에는 다른 호스트를 가진 MSA 모듈간의 통신을 해결하지 못하였는데, 원인에 대해 좀 더 알아보고 싶다.

민철

- Swagger를 통해 API 테스트를 간편하게 할 수 있어서 좋았다.
- 연관성이 있는 도메인 간의 의존성을 낮추는 방법에 대해 생각해볼 수 있었다.
- 인증 객체를 가져와 사용만 하다 보니 스프링 시큐리티 학습이 필요하다고 느꼈다.

Thank you*

감사합니다.

MIR COMPANY
MIRI GROUP