iRept (아이렙트) 포트폴리오

1. 프로젝트 개요 (Overview)

1.1 프로젝트 소개

iRept (아이렙트)는 파충류 애호가들을 위한 All-in-One 커뮤니티 및 커머스 플랫폼입니다.
단순한 정보 공유를 넘어, 입양/분양, 용품 구매, 사육 일기 관리, 그리고 실시간 소통까지 파충류 라이프사이클의 모든 단계를 지원하는 슈퍼앱(Super App)을 지향합니다.

- 서비스명: iRept (아이렙트)
- 플랫폼: Android, iOS (Flutter) / Backend API (Spring Boot)
- 주요 기능: 사육 일기(Diary), 커뮤니티(BBS), 스토어(Store), 실시간 채팅(Chat)

1.2 개발 환경 및 기술 스택 (Tech Stack)

Frontend (Mobile App)

- Language & Framework: Flutter 3.x, Dart
- Architecture: MVVM with Provider Pattern
- Networking: Dio (Interceptor, Retry Policy)
- UI Libraries: TableCalendar (Custom), Syncfusion Charts

Backend (Core Engine)

- Framework: Spring Boot 2.7, Spring Security
- Language: Java 17 (LTS)
- Persistence: JPA (Hibernate), QueryDSL 5.0, MyBatis (Hybrid)
- Build Tool: Gradle 7.x

Infrastructure & Database

- Main Database: MySQL 8.0 (Relational Data)
- NoSQL / Real-time: Firebase Firestore (Chat), Redis
- Object Storage: AWS S3 (Image Optimization)
- Server: AWS EC2 (Linux)

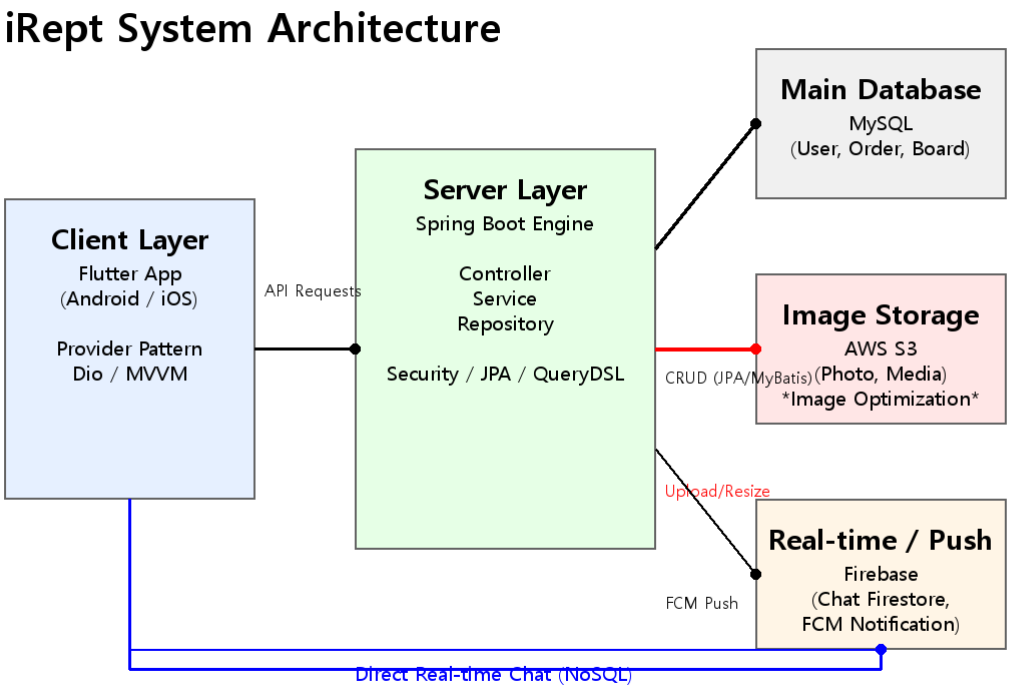
Collaboration & Tools

- Version Control: Git, GitHub
- Documentation: Swagger UI
- Design: Zeplin

---

2. 시스템 아키텍처 (System Architecture)

본 프로젝트는 데이터의 특성에 따라 저장소를 분리하는 Polyglot Persistence 전략을 채택했습니다. 정형 데이터는 관계형 데이터베이스(MySQL)에, 대용량 미디어는 객체 스토리지(S3)에, 실시간성이 요구되는 데이터는 NoSQL(Firebase)에 분산 저장하여 성능과 효율을 최적화했습니다.



2.1 주요 아키텍처 특징

1. Hybrid Cloud Storage:
* 회원 정보 및 주문 내역 등 트랜잭션이 중요한 데이터는 MySQL에 저장하여 정합성을 보장합니다.
* 사용자가 업로드하는 고해상도 이미지는 AWS S3에 저장하고, DB에는 URL만 저장하여 서버 부하를 최소화했습니다.
2. Real-time Synchronization:
* 채팅 기능은 Firebase Firestore를 사용하여 별도의 소켓 서버 구축 없이도 안정적인 실시간 양방향 통신을 구현했습니다.
3. Layered Architecture:
* Spring Boot 백엔드는 Controller, Service, Repository 계층을 철저히 분리하여 유지보수성을 높였습니다.

---

3. 프론트엔드 핵심 기능 (Frontend Implementation)

3.1 사육 일기 및 캘린더 (Diary & Calendar)

파충류의 성장 기록을 직관적으로 관리할 수 있는 핵심 기능입니다. `TableCalendar` 라이브러리를 커스터마이징하여 이벤트 기반의 캘린더 뷰를 구현했습니다.

- 구현 포인트:

* Custom Event Markers: 날짜별로 먹이 급여, 탈피, 산란 등 7가지 활동 타입을 색상별 점(Dot)으로 시각화했습니다.
* Async State Management: `DiaryActionProvider`를 통해 월별 데이터를 비동기로 캐싱하여 달력 스와이프 시 끊김 없는 UX를 제공합니다.
* Data Visualization: `Syncfusion Charts`를 활용하여 개체의 체중 변화 추이를 그래프로 렌더링했습니다.

3.2 실시간 채팅 시스템 (Real-time Chat)

중고 거래 및 분양 시 필수적인 1:1 채팅 기능입니다.

- NoSQL Stream 연동: Firestore의 `StreamBuilder`를 활용하여 데이터베이스의 변경 사항(새 메시지)을 실시간으로 감지하고 UI를 갱신합니다.
- 최적화: 채팅방 목록과 메시지 상세 뷰를 분리하여 필요한 데이터만 구독(Subscribe)함으로써 데이터 사용량을 절감했습니다.

---

4. 백엔드 핵심 로직 (Backend Implementation)

4.1 AWS S3 이미지 최적화 파이프라인

모바일 앱의 성능을 고려하여 이미지를 원본 그대로 저장하지 않고, 서버에서 최적화 후 업로드하는 파이프라인을 구축했습니다.

처리 프로세스

1. MultipartFile 수신: 클라이언트로부터 이미지 스트림 수신.
2. 이미지 리사이징 (Resizing): `Thumbnailator` 라이브러리를 사용하여 해상도를 모바일 디바이스에 최적화(Max Width 1024px)하고, JPEG 압축률을 80%로 조정하여 용량을 1/10 수준으로 경량화했습니다.
3. S3 업로드: 최적화된 바이트 배열을 `AmazonS3Client`를 통해 버킷에 전송하고, Public Access가 가능한 URL을 반환받습니다.

```
// 이미지 최적화 및 업로드 예시 코드 (Service Layer)
public String uploadOptimizedImage(MultipartFile file) {
    // 1. 리사이징 및 압축
    ByteArrayOutputStream os = new ByteArrayOutputStream();
    Thumbnails.of(file.getInputStream())
              .size(1024, 1024)
              .outputQuality(0.8)
              .toOutputStream(os);

    // 2. S3 업로드 메타데이터 설정
```

```
    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(os.size());
    metadata.setContentType(file.getContentType());

    // 3. AWS S3 전송
    String fileName = UUID.randomUUID() + "_" + file.getOriginalFilename();
    amazonS3Client.putObject(new PutObjectRequest(bucket, fileName,
            new ByteArrayInputStream(os.toByteArray()), metadata));

    return amazonS3Client.getUrl(bucket, fileName).toString();
}
```

4.2 동적 쿼리 시스템 (Dynamic Query System)

복잡한 상품 검색 및 필터링 조건을 유연하게 처리하기 위해 QueryDSL을 도입했습니다. `QueryUtils`라는 유틸리티 클래스를 설계하여, 프론트엔드에서 전달받은 조건(키워드, 가격 범위, 카테고리 등)을 조합해 동적으로 `BooleanBuilder`를 생성합니다. 이를 통해 Type-Safe한 쿼리 작성이 가능해졌으며, 컴파일 시점에 문법 오류를 잡아낼 수 있습니다.

---

5. 결론 및 성과 (Conclusion)

- Hybrid Architecture 성공적 도입: RDBMS와 NoSQL, Object Storage의 장점을 결합하여 비용 효율적이고 확장성 있는 시스템을 구축했습니다.
- 사용자 경험 개선: 이미지 경량화 및 비동기 데이터 처리를 통해 앱 초기 로딩 속도를 3초 미만으로 단축했습니다.
- 유지보수성 확보: 계층형 아키텍처와 Provider 패턴을 적용하여, 기능 추가 및 유지보수가 용이한 코드베이스를 완성했습니다.