Species Verifier 개발 계획

🎯 현재 완료된 기능

- ✅ 3개 탭(해양생물, 미생물, COL) 통합 검증 시스템
- ✅ 파일 캐시 관리 시스템 (검증 완료/취소 시 자동 삭제)
- ✅ 취소 시 파일 캐시 삭제 기능
- ✅ 진행률 표시 개선
- ✅ 도움말 서식 개선 및 연락처 추가
- ✅ 텍스트 입력과 파일 입력 우선순위 명확화

🔧 즉시 개선 필요 사항

- 🔄 중단 기능 개선 (현재 불완전)
- 🔄 비동기와 동기 코드 혼재 문제 해결
- 🔄 결과 테이블 셀 선택 방식 개선 (전체 행 → 개별 셀)
- 🔄 심층분석 결과 UI 개선 (JetSpark 디자인 검토)

🚀 향후 개발 계획

Phase 1: 핵심 시스템 개선 (우선순위: 높음)

1.1 DB 캐싱 시스템 구축 💾

- 🔄 Supabase 스키마 설계 및 구축
- 🔄 캐싱 로직 구현 (1개월 캐시 기간)
- 🔄 기존 검증 시스템과 통합
- 🔄 성능 테스트 및 최적화

DB 스키마 설계:

```
-- 검증 결과 캐시 테이블
CREATE TABLE verification_cache (
    id SERIAL PRIMARY KEY,
    scientific_name VARCHAR(255) NOT NULL,
    verification_type VARCHAR(50) NOT NULL, -- 'marine', 'microbe', 'col'
    result JSONB NOT NULL, -- 검증 결과 전체 데이터
    created_at TIMESTAMP DEFAULT NOW(),
    expires_at TIMESTAMP NOT NULL,
    INDEX(scientific_name, verification_type),
    INDEX(expires_at)
);
```

캐싱 시스템 동작 방식:

- 검증 요청 시 DB에서 먼저 조회
- 캐시된 결과가 있고 유효기간 내인 경우 즉시 반환

- 캐시가 없거나 만료된 경우 API 호출 후 DB 저장

Phase 2: DeepSearch 심층분석 시스템 (우선순위: 중간)

2.1 데이터 수집 및 구조화 🔍

- 🔄 위키피디아 정보 + 기타 데이터베이스 통합
- 🔄 심층분석 DB 구축
- 🔄 데이터 품질 관리 시스템
- 🔄 UI/UX 개선 (상세 정보 표시)

수집 정보:

- 위키백과 상세 정보 (한국어/영어)
- 생태학적 특성 (서식지, 분포, 생활사)
- 형태학적 특징
- 경제적/생태적 중요성
- 관련 연구 논문 정보
- 이미지 URL

심층분석 DB 스키마:

```
CREATE TABLE deep_search_results (
    id SERIAL PRIMARY KEY,
    scientific_name VARCHAR(255) UNIQUE NOT NULL,
    korean_name VARCHAR(255),
    taxonomy JSONB, -- 분류학적 정보
    ecology JSONB, -- 생태학적 정보
    morphology TEXT, -- 형태학적 특징
    distribution TEXT, -- 분포 정보
    economic_importance TEXT, -- 경제적 중요성
    conservation_status VARCHAR(100), -- 보전 상태
    images JSONB, -- 이미지 URL 배열
    references JSONB, -- 참고문헌 정보
    last_updated TIMESTAMP DEFAULT NOW(),
    data_quality_score INTEGER DEFAULT 0
);
```

Phase 3: 고급 기능 (우선순위: 낮음)

- ⏳ 배치 처리 시스템 (백그라운드 DeepSearch)
- ⏳ 성능 모니터링 대시보드
- ⏳ 예측 캐싱 시스템
- ⏳ 오프라인 모드 지원

Phase 4: 플랫폼 확장 (장기 계획)

4.1 코드 리팩토링 🔧

- ⏳ 비동기/동기 코드 정리 및 최적화
- ⏳ 모듈화 및 의존성 관리 개선
- ⏳ 테스트 커버리지 확대
- ⏳ 코드 품질 관리 (Linting, 타입 힌트)
- ⏳ API 문서 및 개발자 가이드 작성

4.2 Flutter 앱 개발 📱

- ⏳ 크로스 플랫폼 앱 (Android, iOS, Windows, macOS)
- ⏳ 모바일 친화적 UI/UX 설계
- ⏳ REST API를 통한 백엔드 통신
- ⏳ 오프라인 지원 (로컬 SQLite 캐시)
- ⏳ 푸시 알림 및 다크 모드 지원

4.3 웹 서비스 🌐

- ⏳ 웹 버전 개발
- ⏳ API 서버 분리
- ⏳ 마이크로서비스 아키텍처
- ⏳ 사용자 인증 및 권한 관리

🔧 기술적 고려사항

데이터베이스 최적화

- **인덱싱:** 검색 성능 최적화
- **파티셔닝:** 대용량 데이터 처리
- **백업 전략:** 데이터 안정성 확보

API 통합 및 성능

- **Rate Limiting:** 외부 API 호출 제한 준수
- **Error Handling:** 네트워크 오류 처리 강화
- **Fallback 전략:** API 장애 시 대응 방안

보안 및 개인정보

- **데이터 암호화:** 민감 정보 보호
- **접근 제어:** 사용자 권한 관리
- **로그 관리:** 사용 기록 추적

💡 예상 효과

성능 향상

- **응답 시간:** 캐시된 결과 즉시 반환 (90% 이상 속도 향상)
- **서버 부하:** API 호출 횟수 대폭 감소
- **사용자 경험:** 빠른 검증 결과 제공

데이터 품질 향상

- **정확성:** 검증된 고품질 데이터 제공
- **완성도:** 기본 검증 + 심층 분석 정보
- **최신성:** 주기적 데이터 업데이트

연구 지원 강화

- **상세 정보:** 연구자들이 필요로 하는 깊이 있는 정보
- **참고자료:** 신뢰할 수 있는 출처 정보
- **데이터 활용:** 연구 목적 데이터 내보내기 지원

개발 문의: ecomarine@korea.kr