EVE Abyssal Tracker 기능 명세 (코드 분석 기반)

이 문서는 EVE Abyssal Tracker 애플리케이션의 주요 기능들을 실제 소스 코드 분석에 기반하여 구체적으로 설명합니다.

1. 핵심 기능: 로그 기반 어비설 추적

- **로그 파일 실시간 모니터링**:
    - **구현**: `log_monitor.rs`의 `monitor_loop` 비동기 함수가 2초 간격으로 EVE Online 로그 파일을 폴링합니다.
    - **파일 감지**: `find_latest_local_log` 함수는 설정된 경로에서 `지역_*.txt`(ko) 또는 `Local_*.txt`(en) 패턴과 일치하는 최신 로그 파일을 찾습니다. `detect_character_name` 함수를 통해 로그 파일 내용(`Listener: ...`)을 분석하여 특정 캐릭터의 로그를 식별하거나 자동으로 감지합니다.
    - **내용 처리**: `process_new_lines` 함수는 `UTF-16LE` 인코딩으로 파일을 읽고, 이전에 읽은 위치 이후에 추가된 새로운 라인만 추출합니다. 새 라인이 감지되면 `system_change_processor.rs`로 전달하여 추가 분석을 수행합니다.
    - **UI 연동**: `Settings.tsx` 컴포넌트의 "모니터링 시작/중지" 버튼을 통해 `invoke("start_log_monitor_command")` 및 `invoke("stop_log_monitor_command")` Tauri 커맨드를 호출하여 모니터링을 제어합니다. 모니터의 상태는 `"log_monitor_status"` 이벤트를 통해 UI에 실시간으로 반영됩니다.
- **어비설 런 감지 및 데이터 입력**:
    - **구현**: (현재 코드는 `system_change_processor.rs`에서 성계 이동을 감지하는 로직이 중심이며, 어비설 진입/종료는 이 성계 이동을 기반으로 추정됩니다. 예를 들어 'Abyssal' 문자열이 포함된 성계로의 이동을 감지합니다.)
    - **결과 입력창**: 런이 완료되었다고 판단되면, 백엔드에서 `abyssal-result` 창을 새로 띄웁니다. 이 창은 `AbyssalResultModal.tsx` 컴포넌트를 렌더링합니다.
    - **사용자 입력**: 사용자는 이 모달 창에서 어비설 종류(T1~T6, 날씨), 사용한 함급(입장료 계산용), 그리고 EVE 클라이언트에서 복사한 전리품 목록을 직접 입력합니다.
    - **데이터 저장**: '저장하기' 버튼 클릭 시, 입력된 정보는 백엔드의 `abyssal_data_manager.rs`에 있는 `save_abyssal_result` 함수로 전달되어 `abyssal_results_{YYYY-MM-DD}.csv` 파일에 저장됩니다.

2. 데이터 처리 및 관리

- **데이터 저장 및 관리**:
    - **구현**: `abyssal_data_manager.rs`가 데이터 관리를 전담합니다.
    - **형식**: 모든 런 기록은 Polars `DataFrame`을 사용하여 처리되며, 앱 데이터 폴더 내 `data/abyssal_results_{YYYY-MM-DD}.csv` 파일에 `UTF-8-BOM` 인코딩으로 저장됩니다.
    - **CRUD**:
        - **Create**: `save_abyssal_result` 함수가 새 런 기록을 해당 날짜의 CSV 파일에 추가합니다.

        - **Read**: `load_abyssal_results` 함수가 `data` 폴더의 모든 `abyssal_results_*.csv` 파일을 읽어 하나의 `DataFrame`으로 병합합니다.
        - **Delete**: `delete_abyssal_run` 함수는 `DailyStatsDisplay.tsx`에서 '삭제' 버튼 클릭 시 호출되며, `시작시각(KST)`과 `종료시각(KST)`을 기준으로 특정 런 기록을 CSV 파일에서 찾아 삭제합니다.

- **전리품 파싱**:
    - **구현**: `abyssal_data_manager.rs`의 `parse_items` 함수가 담당합니다.
    - **로직**: 사용자가 붙여넣은 텍스트에서 `아이템명*수량`, `아이템명\t수량`, `아이템명;수량` 등 다양한 형식의 문자열을 정규식으로 분석하여 `(아이템명, 수량)` 형태의 목록으로 변환합니다.

3. 데이터 분석 및 API 연동

- **가격 정보 연동 및 캐싱**:
    - **구현**: `eve_api.rs` 모듈이 외부 API 통신을 담당합니다.
    - **Type ID 변환**: `fetch_type_ids` 함수는 전리품 이름 목록을 ESI API (`/universe/ids/`)에 보내 Type ID로 변환합니다. 이 결과는 `typeid_cache.json`에 영구적으로 캐싱되어 불필요한 API 호출을 최소화합니다.
    - **가격 조회**: `fetch_fuzzwork_prices` 함수는 변환된 Type ID를 Fuzzwork Market API로 보내 Jita 기준 시장 가격을 조회합니다. 이 가격 정보는 `data/price_cache.json`에 30분 유효기간(TTL)으로 캐싱됩니다.

- **데이터 분석 및 통계 생성**:
    - **구현**: `abyssal_data_analyzer.rs` (해당 파일의 코드는 제공되지 않았으나, 호출 관계로 유추) 및 `App.tsx`에서 데이터 처리.
    - **프로세스**: `App.tsx`에서 `invoke("analyze_abyssal_data_command")`를 호출하면 백엔드는 다음 단계를 수행합니다.
        1. `abyssal_data_manager`를 통해 모든 CSV 데이터를 로드.
        2. 모든 전리품 아이템 이름을 수집.
        3. `eve_api`를 통해 아이템 이름들을 Type ID로 변환 (ESI API).
        4. Type ID로 가격 정보 조회 (Fuzzwork API).
        5. 획득한 가격 정보를 바탕으로 각 런의 실수익, ISK/h 등의 통계를 계산.
    - **UI 전달**: 계산된 최종 데이터(원본 데이터프레임, 일별 통계, 전체 통계 등)는 `AbyssalData` 타입으로 프론트엔드에 전달됩니다.
    - **새로고침**: 새 런이 완료되면 `"abyssal_run_completed"` 이벤트가 발생하고, `App.tsx`는 `invoke("light_refresh_abyssal_data_command")`를 호출하여 로딩창 없이 통계를 다시 계산하고 UI를 업데이트합니다.

4. 사용자 인터페이스 (UI)

- **메인 애플리케이션 (`App.tsx`)**:
    - **로딩 화면**: 앱 시작 시 `LoadingProgress.tsx` 컴포넌트를 통해 데이터 분석 각 단계(CSV 로드, API 호출, 통계 생성 등)의 진행 상황을 사용자에게 보여줍니다.
    - **탭 구조**: '분석 대시보드' (`StatsDisplay.tsx`)와 '설정' (`Settings.tsx`) 탭으로 구성됩니다.

    - **알림**: `NotifierPopup.tsx`를 통해 백엔드에서 발생하는 주요 이벤트(예: 트래킹 시작/중지, 에러 발생)를 사용자에게 팝업으로 알립니다.

- **통계 대시보드 (`StatsDisplay.tsx`)**:
    - **일별/전체 통계**: `DailyStatsDisplay.tsx`와 `OverallStatsDisplay.tsx` 컴포넌트가 `recharts` 라이브러리를 사용하여 통계 데이터를 시각화합니다.
    - **차트**: 일별 수익 트렌드, 시간대별 성과, 전체 누적 수익 등 다양한 차트를 제공합니다.
    - **상세 정보**: 각 런 기록을 클릭하면 해당 런의 상세 정보(타임라인, 전리품 목록 및 가치)를 펼쳐 볼 수 있습니다.

- **업데이트 기능**:
    - **구현**: `App.tsx`에서 `invoke("check_for_update_command")`를 호출하여 GitHub 등에서 새 릴리즈가 있는지 확인합니다.
    - **UI**: 새 버전이 있으면 `UpdateDialog.tsx`가 표시되며, 사용자는 이 창을 통해 `invoke("download_and_install_update_command")`를 실행하여 자동 업데이트를 진행할 수 있습니다.