김현서 HYEONSEO KIM

포트폴리오

2026.07
~
Present
3개월

Necrass-Dev/NecrassRS — Rust, GraphQL, Cargo

소개

- GraphQL SDL에서 Rust 타입과 resolver 계약을 생성하고, 스키마 변경을 구현 코드에 반영하는 Rust GraphQL 서버 프레임워크

해결 문제

- 공개 GraphQL 계약과 Rust 구현 사이에서 타입·필드 선언 및 resolver 연결 코드를 반복해서 관리해야 하는 부담을 줄이고자 함.
- SDL을 공개 계약의 기준으로 두고, 코드 생성과 구현 선언 갱신을 일반적인 Cargo 빌드 과정에 통합함.

주요 기능 및 구현 방식

- Cargo 기반 코드 생성
  - Apollo Compiler로 SDL을 파싱·검증하고 Rust 입력 타입, resolver trait, Query 호출 코드를 생성
  - 재생성 가능한 코드는 `OUT_DIR`, 사용자가 구현하는 resolver 본문은 `src/resolvers.rs`에서 관리
- 스키마 변경과 구현 코드 동기화
  - Rust AST를 분석하여 유지되는 resolver 본문과 관련 없는 사용자 코드를 보존
  - 새 필드에 구현 stub을 추가하고, 삭제된 필드의 메서드를 제거
  - 필드 추가·삭제·이름 변경·인자 변경 후 실제 소비자 프로젝트를 재빌드하여 동작 검증
- GraphQL 입력 의미를 Rust 타입에 반영
  - 입력 생략·명시적 null·값을 구분하고, 리스트 nullability와 기본값 처리 지원
  - OneOf 입력을 Rust enum으로 생성하고, 재귀 입력의 순환 참조에 필요한 간접 참조 적용
- 초기화부터 실행까지의 개발 흐름
  - `necrass init`으로 실행 가능한 Axum 프로젝트 생성
  - introspection과 GraphiQL을 제공하고, 실행 핵심을 HTTP 어댑터와 분리하여 Actix 연동도 지원
- 의존성의 명세 처리 문제 재현 및 수정
  - Apollo Compiler의 입력 기본값 순환 검증, OneOf 검증 및 숫자 입력 처리 문제를 로컬 패치와 회귀 테스트로 수정
  - 수정한 의존성을 코드 생성기와 실행기에 함께 적용하고, 생성된 소비자 프로젝트에서도 동작 검증

현황

- 생성된 Query resolver를 통한 scalar·enum·리스트 응답과 일반·OneOf·재귀 입력 지원
- custom scalar, 객체·interface·union 반환 및 Mutation·Subscription 자동 연결은 미지원

2026.04 ~ Present 7개월

gelite-dev/gelite — Rust, SQLite

소개

- Gel에서 영감을 받은 객체 중심 쿼리 언어를 SQLite SQL로 컴파일하고 실행하는 Rust 쿼리 엔진

해결 문제

- SQLite의 테이블과 JOIN 중심 조회 결과를 애플리케이션의 중첩 객체·컬렉션으로 변환하는 작업이 필요함.
- Gel의 객체 중심 쿼리 모델을 SQLite 기반 환경에서 활용하기 위해, 쿼리 언어와 실행 계층을 Rust로 재구현함.
- 쿼리 의미 검증과 SQLite 저장 구조를 분리하기 위해 `AST → Semantic IR → SQLite Plan → SQL → 실행 및 결과 복원` 파이프라인을 구현함.

주요 기능 및 구현 방식

- 스키마 기반 쿼리 컴파일과 CRUD 실행
  - `.geli` 스키마와 `select`, `insert`, `update`, `delete` 쿼리의 파싱부터 SQLite 실행까지 구현
  - 타입·필드·관계 경로·cardinality를 스키마 기준으로 검증하고, 쿼리 의미를 backend-independent Semantic IR로 표현
  - SQLite planner에서 테이블·컬럼·JOIN을 결정하고, SQL과 바인딩 값을 생성하여 실행
- 관계 조회와 중첩 결과 복원
  - `select Post { title, author: { name } }`처럼 필요한 객체 구조를 쿼리에서 선언
  - 단일 관계는 JOIN으로 조회하고, 다중 관계는 부모 ID를 묶은 후속 조회로 가져와 중첩 컬렉션으로 병합
  - 기존 정방향 관계의 저장 구조를 재사용하는 읽기 전용 역방향 link 지원
  - 다중 관계 필터와 `exists` 조건으로 동일한 관계 객체에 여러 조건을 적용하는 조회 지원
- 추가형 스키마 마이그레이션과 이력 검증
  - 새 객체, nullable scalar field, optional single link, multi link 추가를 기존 데이터 보존과 함께 적용
  - 저장된 스키마의 버전·체크섬·논리 catalog를 검증하고, 지원하지 않는 변경은 DDL 실행 전에 거부
  - DDL·메타데이터·스키마 이력을 하나의 트랜잭션으로 적용하고 실패 시 함께 롤백
- CLI/REPL과 트랜잭션
  - 스키마 적용, 쿼리 계획 확인, 쿼리 파일 실행 및 데이터베이스 기반 대화형 REPL 제공
  - 여러 문장으로 구성된 쿼리 파일 전체를 사전 컴파일하고 트랜잭션 구성을 검증
  - `start transaction`, `commit`, `rollback`을 지원하고 실행 오류 시 활성 트랜잭션 롤백
  - `--debug` 옵션으로 쿼리 컴파일 단계의 결과를 확인
- no_std 컴파일러 핵심과 브라우저 WASM 실행 기반
  - parser, schema catalog, resolver, IR, SQLite planner 및 SQL generator를 `no_std` crate로 분리
  - 네이티브 SQLite runner와 브라우저 WASM runner를 분리하고 공통 스키마 적용 로직 재사용
  - WASM의 메모리 SQLite에서 SQL 실행, 초기 스키마·추가형 마이그레이션 적용 및 이력 검증 구현
  - 브라우저 테스트로 스키마 적용·검증과 실패 시 롤백을 확인하도록 CI 구성

현황

- 네이티브 CLI/REPL에서 CRUD, 중첩 관계 조회, 추가형 스키마 마이그레이션 및 트랜잭션 실행 지원
- 브라우저 WASM은 SQL 실행과 스키마 적용·검증까지 지원하며, Gelite 쿼리의 전체 실행·결과 복원은 미구현
- 테이블 재구성·이름 변경·백필을 포함한 마이그레이션, HTTP 서버 및 웹 UI는 미구현

2026.02 ~ 2026.05 2개월

dodok8/gaji — TypeScript, GitHub Actions, Rust

소개

- GitHub Actions 워크플로우를 TypeScript로 타입 안전하게 작성 후 YAML로 컴파일하는 CLI 도구
  - 제작기 및 제작이유
  - 문서

해결 문제

- YAML은 데이터 표현 언어로, CI/CD 동작(behavior)을 표현하기에 부적절함.
- 액션 입력값에 대한 IDE 자동완성 및 타입 검증이 불가능하여, 필수 입력 누락이나 잘못된 값을 사전에 감지할 수 없음.

GitHub Actions의 위와 같은 단점을 TS를 이용해 워크플로우 작성시에 해결하는 것을 목적으로 삼음.

주요 기능

- action.yml 기반 자동 타입 생성
  - action.yml 정의에서 TypeScript 타입을 자동 생성하여, `getAction()` 함수를 통해 액션 입력값의 IDE 자동완성 및 컴파일 시점 타입 체크 제공
  - `gaji dev --watch` 로 파일 변경 감시 시, 새로운 액션 참조를 자동 감지하여 타입 생성
- 빌더 패턴 API
  - `Workflow` → `Job` → `Step` 구조의 빌더 패턴으로 워크플로우를 구성하고, `.build()`로 YAML 컴파일
  - `CompositeAction`, `JavaScriptAction`, `DockerAction` 등 다양한 액션 타입 정의 지원
  - `CallJob`으로 재사용 가능한 워크플로우 호출, `CompositeJob`으로 매개변수화된 작업 패턴 구현
- Rust + QuickJS 아키텍처
  - Rust 단일 바이너리에 QuickJS를 내장하여 Node.js 런타임 의존 없이 동작
- 기존 워크플로우 마이그레이션
  - 기존 YAML 워크플로우를 `Workflow.fromObject()`로 TypeScript로 변환 가능

현황

- gaji 그 자체의 워크플로우를 gaji로 작성하여, 이를 통해 더 효율적인 배포 과정 작성이 가능해짐.

2025.07 ~ Present
1년 4개월

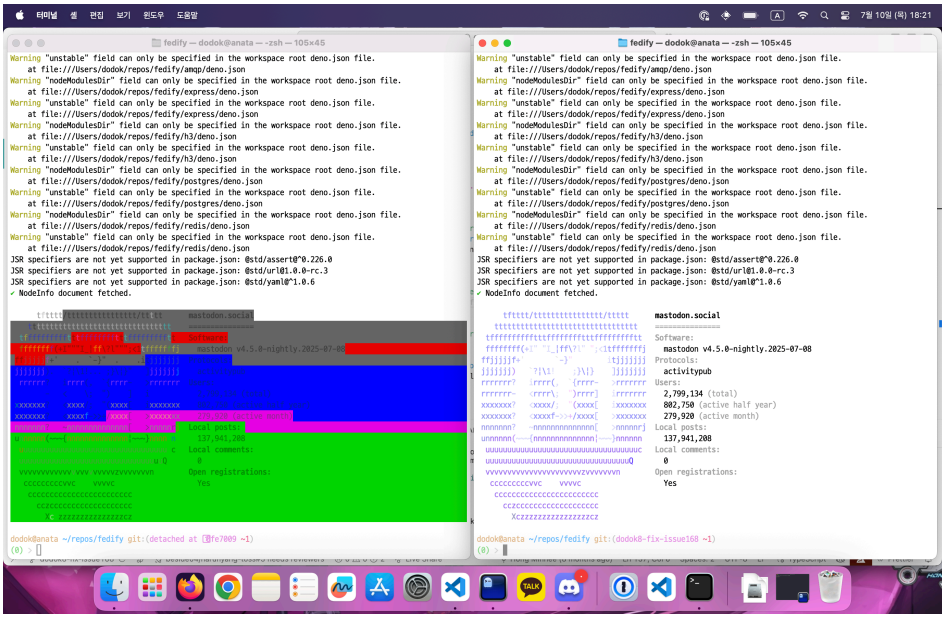
Fedify — TypeScript, ActivityPub

소개

- 2025년 오픈 소스 소프트웨어 컨트리뷰션 아카데미가 계기가 되어, 다음 프로젝트에 기여 중
  - ActivityPub 구현 프레임워크 fedify-dev/fedify — Typescript, ActivityPub
  - Fedify 기반 1인 SNS fedify-dev/hollo — Fedify
  - Fedify 기반 Fediverse 자동 봇 프레임워크 fedify-dev/botkit — Fedify
  - Fedify 기반 IT 전문 오픈소스 SNS hackers-pub/hackerspub — Fedify, Solid, GraphQL

해결 문제

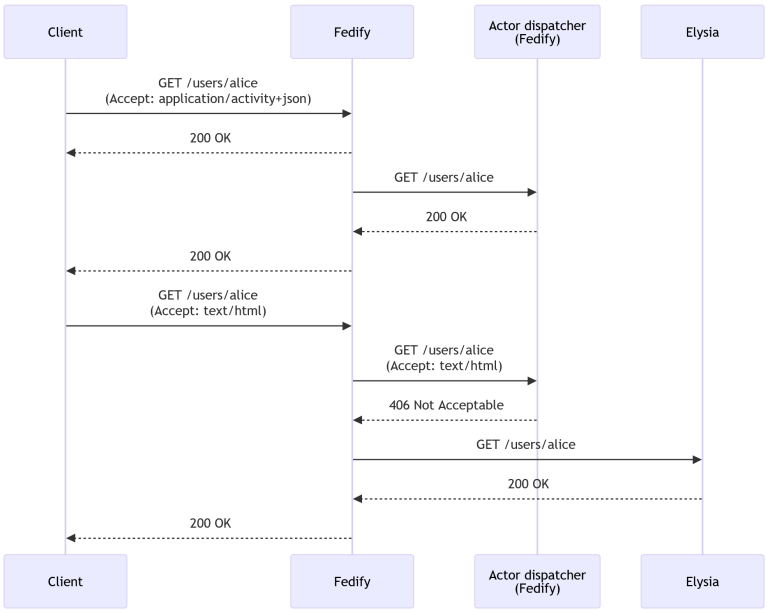
- Fedify CLI nodeInfo 커맨드 개선
  - 문제점: Fediverse 인스턴스의 nodeInfo를 출력해주는 `fedify nodeInfo` 명령어에서 터미널 에뮬레이터의 24bit 컬러 지원 여부에 따라 색상 출력이 붕괴되는 현상이 있었음. (issue #168)
    이 이슈에서 출발하여 점진적으로 커맨드를 개선함.



fedify nodeinfo 색상 개선 결과

  - 개선 과정
    - PR #282: 터미널 에뮬레이터의 24 bit 색상 지원 여부를 감지해 미지원하는 터미널의 경우 ANSI 256 색상을 출력하도록 변경.
    - PR #299: 해당 기능을 포함한 nodeInfo 명령어의 구성 요소들의 유닛 테스트를 작성. 이 과정에서 실제 ANSI 컬러값이 해당값과 매치되도록 색상 비교 로직을 재설계함.
    - PR #327: 테스트 작성 과정에서, `--raw` 옵션이 실제로 존재함에도 불구하고 문서에는 누락되어 있는 것을 발견함. 이를 반영하여 문서를 업데이트함.
    - PR #331: NodeInfo 를 가져오는 명령어 임에도 불구하고, 기존에는 `node` 여서 혼동을 주는 이슈가 있었음. 이를 `nodeinfo`로 명령어를 변경함으로서 해결함.
    - PR #414: 호환성을 위해 전체적으로 CLI를 재작성 하는 과정에서, nodeInfo를 담당하여 CLI를 재작성.

- @fedify/elysia 패키지 제작 (PR #339)
  - Bun 기반 백엔드 프레임워크 Elysia와 Fedify 의 통합 패키지. Elysia의 `onRequest` 라이프 사이클에서 `fedify`의 `federation.fetch`를 호출하여 ActivityPub 요청인지 구별하도록 구현.



@fedify/elysia 패키지 작동 원리

  - Bun이 Fedify의 jsr 의존성을 지원하지 않아서 오류가 발생했음. 플러그인의 빌드 과정에 있어서 Bun 의존성은 없었기에 기존 빌드 체인인 tsdown을 사용하고, 빌드된 패키지만 Bun에서 사용하도록 하는 방식으로 해결함.
- @fedify/fresh 패키지 제작 (PR #478)
  - Deno 기반 웹 프레임워크 Fresh 2 와 Fedify 의 통합 플러그인. 기존 Fresh 연동과 유사한 구조를 가지도록 설계하였음.
  - Fresh 2에서 사용 중인 Vite 서비스와 Fedify의 의존성 사이에 오류가 발생하여, 이를 해결 하는 Vite 설정을 문서화함.
  - 더 이상 사용되지 않은 Preact 의존성으로 인한 오류를 해결함.
- @fedify/solidstart 패키지 제작 참여 (PR #601, PR #652)
  - Solid 기반 웹 프레임워크 SolidStart용 연동 패키지 제작에 참여.
- BotKit, Hackers' Pub 팔로잉 관련 기능 개선 및 Fedify Webfinger 확장 API 추가
  - 문제점
    - BotKit의 기본 봇 페이지는 팔로워 목록을 제공하지 않았음. (issue #2)
    - BotKit의 봇을 팔로우 하기 위해서는 사용자가 자신의 인스턴스로 돌아가서 BotKit의 핸들을 검색해야하는 불편함이 존재하였음. 이를 해결하기 위해 Remote Follow 가 제안 됨. (issue #10)
  - 개선 과정
    - PR #13: 팔로우 목록 제공 프론트엔드 제작.
    - PR #14: Remote Follow 구현
      - 유저로부터 핸들 입력 받음
      - 유저 핸들 정보로 부터 webfinger 정보 수집
      - OStatus 1.0 따른 subscribe 항목이 있을 경우, 그에 해당하는 봇 팔로우 url로 리다이렉트
    - 본 프로젝트인 `fedify`와 이를 기반으로 하는 앱에서 OStatus 1.0을 지원하지 않았기에 이를 확장하는 PR들을 작성함.
      - PR #404 Fedify의 Link의 타입을 확장하여, OStatus 1.0 을 지원하는 mastodon 과 같은 인스턴스의 webfinger를 받을 수 있도록 확장함.

        - PR #407 Fedify에서 Webfinger 커스텀을 보다 더 쉽게 할 수 있도록, `setWebFingerLinksDispatcher(dispatcher)` 추가. OStatus 1.0 과 같은 추가적인 정보를 사용자가 손쉽게 지정할 수 있게 되었음.
    - BotKit에서 구현한 Remote Follow를 Hackers' Pub에도 적용 (issue #173, PR #180)
- Hollo 타임라인 오류 개선
    - 문제점: Hollo의 타임라인에 게시물의 작성시간이 미래로 되어 있는 경우, 타임라인의 최상단에 고정되어 버리는 문제가 존재하였음. (issue #199)
    - 해결책 : PR #201 게시물의 생성 시간과 서버시간을 비교하여, 5분 이후의 미래 게시물의 경우 타임라인에서 숨겨지도록 하였음.
- Fedify CI/CD 워크플로우 개선
    - 문제점: 스폰서 목록을 업데이트 하는 GitHub actions workflow가 포크된 저장소에서도 작동하여, 매시간 마다 알림이 가는 문제가 있었음.
    - 해결책: 해당 워크플로우를 수동으로 전환한 다음, REST API 를 통해 메인 저장소의 액션만 매시간 호출하도록 별도의 저장소와 워크플로우 작성.(fedify-dev/sponsor-automation)
- Fedify NodeInfo 타입 변경
    - 문제점: Fediverse Instance의 서버 정보를 담는 NodeInfo Protocol 2.1의 스펙과 실제 Fedify의 구현이 software version 필드의 타입이 차이가 났음.
    - 개선과정:
        - PR #365: Fedify client 측에서 nodeInfo를 가져올 때, Semver가 아니더라도 파싱할 수 있도록 fallback 을 변경함.
        - PR #433: Fedify의 NodeInfo 구현체의 타입의 software version 필드를 Semver에서 String으로 변경하고, 관련된 테스트들을 수정함.
- Hackers' Pub 해쉬태그 관련 오류 수정
    - 문제점: 알파벳 및 숫자가 아닌 문자가 포함된 해쉬태그 검색이 제대로 되지 않았음.
    - 개선과정:
        - PR #171: 검색 쿼리에서 알파벳과 숫자가 아닌 문자도 지원하도록 정규식 수정. 이를 반영하여 해쉬 태그 페이지 구현

2022.03 ~ 2022.07 4개월

SmartX-Labs/SmartX-Mini

GitHub, Docker

소개

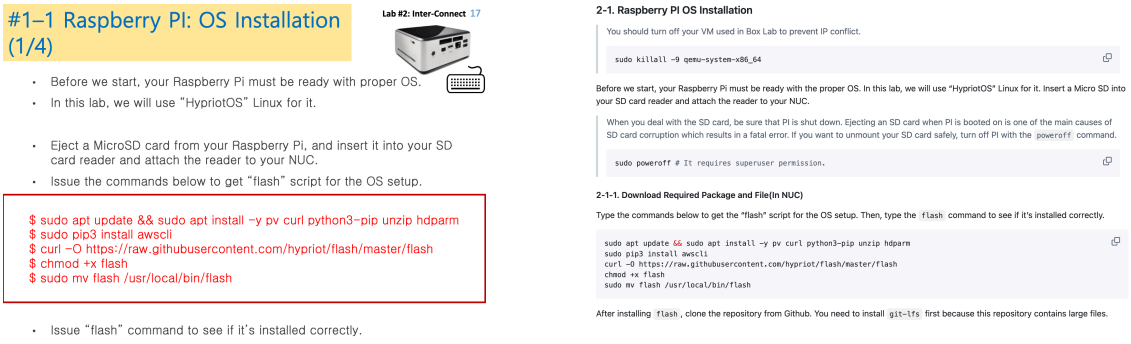
- GIST 전공 필수 IOT 웹 및 네트워크 인프라 실험 SmartX-Mini 실습 교재, 실험 조교로 참여

목표

- 기존 PPT 파일의 문제였던 업무 분업 및 변경 사항 추적 불가 개선
- 년도가 변경되며 생긴 오타 및 사용되는 파일 및 명령어의 버전 차이로 인한 진행 불가 개선

현황

- 원할한 수업 진행
  - 오타가 없는 수업자료가 제공됨으로써 수업 시 실험 성공률이 올라감
  - 매주 스프린트 진행 후, 발생하는 오류들에 대한 경고 및 해결 방식을 메뉴얼에 반영함으로서 다음 수업에서는 같은 오류를 막을 수 있었음
- 원할한 인수인계
  - 2022년 수정 당시는 마크다운 이전 및 버전 변경없는 버그 해결을 목표로 진행되었음
  - 2023년 부터는 지원이 끊긴 버전을 최신 버전으로 올리는 마이그레이션을 진행.
    - 반복되는 오류를 수정하느라 낭비되었을 인력이 버전 마이그레이션 위주로 투자 되었음



SmartX Mini InterConnect Lab 실습 교재(좌: 개선 전 PPT, 우: 개선 후 마크다운)

해결 문제

- 1차 개선
  - 기존 PPT에 존재하는 오타를 수정한 버전을 제작. 이 버전을 기준으로 다시 실습을 진행하면서 발생하는 버전 충돌 과정을 탐색
  - 오타가 수정된 PPT를 기준으로, 원할한 분업 작업 및 변경사항 추적을 위해 Git 과 markdown 조합을 사용하도록 변경
- 버전 오류 수정
  - 분업 후 Kafka와 flume으로 IoT 장비를 연결 실습을 담당하는 InterConnect Lab랩 담당.
  - 해당 수업에 사용되는 지원이 끊긴 flume 버전 1.6.0의 설치 소스를 archive 저장소 옮기도록 Dockerfile 수정