포트폴리오

이영주 · 대용량 렌더링 최적화, 앱 출시·운영, AI 기능을 제품에 넣고 측정까지 해 본 5년차 웹 프론트엔드 개발자

epww1502@gmail.com github.com/epww1502 이력서 → 경력기술서 →

약 10,000 → 100ms
대용량 차트 렌더링을 계열 단위로 바꿔 100배 단축

3개 플랫폼 · 5회
iOS·Android·반응형 웹을 한 코드로, 4개월간 정식 릴리스 5회

이용자 약 2천 명
산돌구름 AI Studio 베타로 공개된 서비스의 백오피스(ADMIN)와 인증 구조

티맥스가이아 · 2022.01 – 2025.01

GAIA OFFICE Chart

웹오피스의 차트를 Victory 라이브러리 위에 만들었습니다. 2022년 9월 팀 개편 이후에는 차트 기능 전체의 개발과 유지보수를 책임졌고, 신규 입사자 약 5명의 멘토링도 맡았습니다.

역할	스택	성과	데모
차트 모델·컴포넌트 설계, 기능 전체 개발·유지보수	React, TypeScript, MobX, Victory(D3 기반)	대용량 렌더링 약 10,000ms → 100ms	직접 해보기

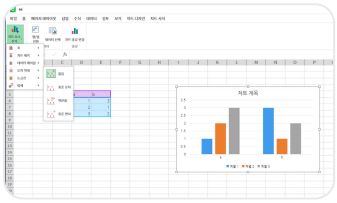
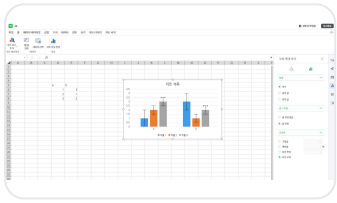


차트 요소 추가 메뉴


오차 막대 서식

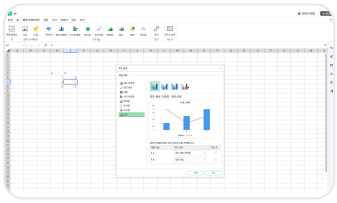
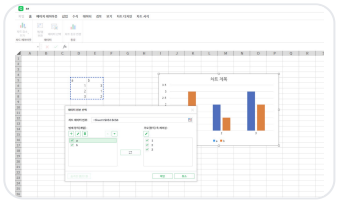


차트 종류 변경 · 콤보 차트


데이터 선택 다이얼로그

대용량 차트 렌더링 개선 (2023.01 – 03)

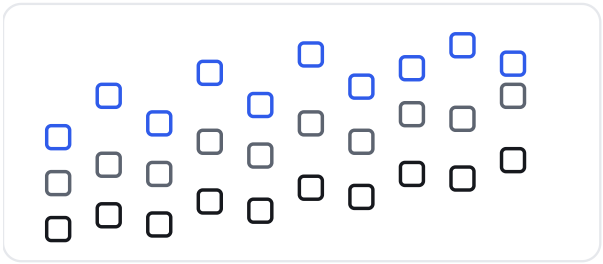
문제 데이터가 많은 차트일수록 급격히 느려져, 100×100 차트 하나를 그리는 데 약 10초가 걸렸습니다. 30×30만 넘어도 삽입과 선택, 크기 조절이 느렸고, 서버로 가는 메시지가 몰려 데드락이 생기기도 했습니다.

원인 가장 작은 단위인 데이터 포인트마다 컴포넌트와 모델을 만들고 있어서, 100×100 차트면 컴포넌트가 10,000개 생겼습니다.

해결 Victory가 포인트마다 계산하는 path를 계열(series) 단위로 모아, 계열 하나를 path 하나로 그리도록 바꿨습니다. 데이터 포인트는 모델의 자식으로 만들지 않고 목록으로 관리해 서버로 보내는 메시지 수도 줄였습니다. 오차 막대, 데이터 레이블, 선택, 행/열 전환, 타입 변경까지 모두 새 구조에 맞췄습니다.

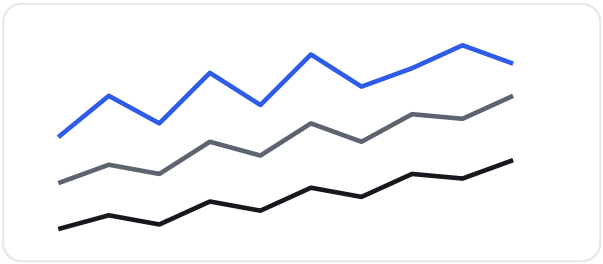
결과 **100×100 차트 렌더링 약 10,000ms → 100ms, 30×30 차트 삽입 11초 → 2초, 서버 데드락 해결**

변경 전 · 포인트마다 컴포넌트



100×100 차트 = 컴포넌트 10,000개

변경 후 · 계열마다 path 하나



100×100 차트 = 컴포넌트 100개

네모 하나가 컴포넌트 하나입니다. 변경 후에는 계열 하나가 선 하나로 그려집니다.

```
// 변경 전: 포인트마다 컴포넌트를 만든다
series.points.map((point) => <DataPoint key={point.id} model={point} />);

// 변경 후: 계열의 포인트 path를 모아 한 번에 그린다
const d = series.points.map(toPath).join(' ');
<path d={d} />;
```

실제 코드가 아니라 구조만 단순화한 예시입니다.

Excel과 같은 오차 막대 구현 (2022.03 – 05)

문제 Excel과 같은 오차 막대가 필요했지만, Victory는 cap 없는 형태나 한 축만 보이는 형태를 지원하지 않았습니다. 막대형 차트에서는 계열 사이 간격 때문에 위치도 어긋났습니다.

해결 고정값, 백분율, 표준오차, 표준편차 4종을 계열 모델 안에 두고, 차트 위에 별도 그룹으로 겹쳐 그렸습니다. Victory의 Error Bar가 만드는 path 값을 받아 수정해서 그렸고, 오차량 계산식도 구현했습니다.

결과 MS Excel 2021과 같은 값과 크기로 표시

그 외 맡은 일

- Chart 모델·컴포넌트 트리·데이터 관리 구조 설계, OOXML 기준 Import/Export
- 차트 종류 변경과 행/열 전환. 모델을 복제·전치해 오차 막대, 데이터 레이블, 스타일을 유지하고 Undo/Redo 지원
- 데이터 참조를 문자열에서 셀 주소 객체로 바꾸고, 여러 범위 선택(Multi-Range)과 다단계 항목(Multi-Level Category) 지원
- 데이터 선택 다이얼로그. 범위를 바꾸면 미리보기에 바로 반영하고, 범위가 맞지 않으면 오류를 띄움
- 미니 셀. Word·Point 문서에 차트를 넣기 위해 데이터용 셀 문서를 링크 방식으로 연결하는 구조 설계
- 테스트 46개 파일 작성(Jest, React Testing Library). 차트 종류별 모델, 오차 막대·범례·축 명령의 실행/Undo/Redo, 렌더링 훅과 컴포넌트를 검증

관악연구소 · 2025.12 – 현재

GLAB · AI 디지털 PB 앱

개인 투자자에게 AI 투자 상담을 제공하는 앱으로, 2026년 7월 정식 출시 후 머니투데이에 소개됐습니다. 프론트엔드의 기획과 디자인 검토부터 빌드, 배포, 운영까지 맡았고(2026년 5월부터 파트장), 코드리뷰와 스토어 출시, AWS 웹 인프라와 배포 자동화를 담당했습니다.

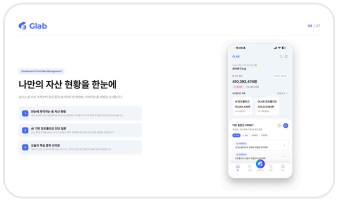
역할
프론트엔드 개발 · 파트장 (2026.05~)
기획 · 빌드 · 배포 · 운영

기여
팀 기능 PR 약 540개 중 274개 작성
동료 PR 265건 중 156건 머지

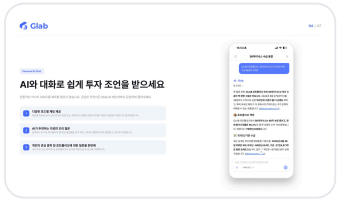
서비스
App Store · Google Play
웹사이트 (PWA) · 머니투데이 보도

스택
Expo, React Native, TanStack Query, Zustand, NativeWind, Firebase Auth

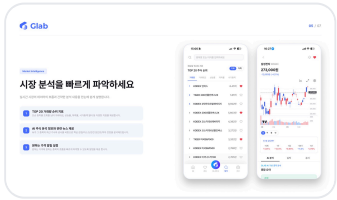
데모
채팅 스트리밍 렌더링 직접 해보기



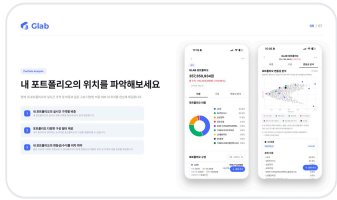
홈 · 자산 현황



AI 채팅



탐색 · 실시간 랭킹



포트폴리오 분석

AI 채팅 스트리밍 렌더링 최적화

문제 답변이 스트리밍되는 동안 토큰이 올 때마다 store와 채팅 화면 전체가 다시 그려졌고, 다른 대화가 스트리밍 중일 때도 보고 있는 화면이 다시 그려졌습니다.

원인 이미 있던 throttle은 대기 값이 할당되지 않아 실제로는 동작하지 않았고, 대화 목록 구독이 채팅 화면 전체에 걸려 있었습니다.

해결 팀원이 먼저 넣은 store 갱신 건너뛰기를 바탕으로 동작하지 않던 throttle을 걷어내 단순화하고, 메시지 컴포넌트 비교 함수로 마지막 메시지만 다시 그리게 했습니다. 대화 목록 구독은 사이드바로 옮겼습니다. 채팅 화면은 탭 밖 별도 스택으로 올리고 `freezeOnBlur` 를 적용해, 보이지 않을 때는 상태를 유지한 채 렌더링만 멈추게 했습니다.

```
// 내용이 바뀐 메시지만 다시 그린다. 스트리밍 중에는 마지막 메시지만 해당된다.
const Message = memo(MessageView, (prev, next) =>
  prev.message.id === next.message.id && prev.message.text === next.message.text,
);

// 채팅은 탭 밖 스택에 두고, 보이지 않는 동안에는 렌더링을 멈춘다.
<Stack screenOptions={{ headerShown: false, freezeOnBlur: true }} />;
```

실제 코드가 아니라 구조만 단순화한 예시입니다.

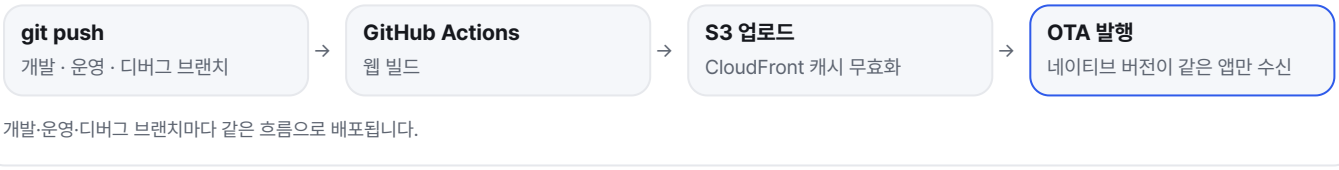
OTA 업데이트와 웹 배포 자동화

상황 작은 팀이 iOS·Android·웹을 동시에 운영해야 했고, 스토어 심사를 거치면 작은 수정도 바로 내보낼 수 없었습니다.

판단 Expo 한 코드베이스로 세 플랫폼을 만들고, 화면·로직 수정은 EAS Update(OTA)로 바로 내보내기로 했습니다.

구현 업데이트는 백그라운드에서 미리 받아 두었다가 앱으로 돌아올 때 적용해 사용 중 화면이 깜빡이지 않게 했고, 복귀할 때의 확인 요청에는 15분 간격을 두었습니다. 네이티브 버전이 다른 앱에는 업데이트를 내려보내지 않아, 옛 앱이 새 네이티브 코드를 찾다가 종료되는 문제를 막았습니다. 웹은 도메인, S3, CloudFront로 AWS 인프라를 구성했습니다. 해시가 붙은 파일은 길게 캐시하고 HTML과 서비스 워커는 매번 확인하도록 나눠, 배포 후에도 옛 화면이 남던 문제를 해결했습니다.

결과 **v1.1 → v1.5까지 4개월간 정식 릴리스 5회, 그 사이 수정은 심사 없이 OTA로 반영**



개발·운영·디버그 브랜치마다 같은 흐름으로 배포됩니다.

소셜 로그인과 Android의 Apple 계정 연결 해제

상황 Kakao·Google·Apple 로그인을 지원해야 했고, App Store는 탈퇴할 때 Apple 계정 연결까지 해제하도록 요구합니다. 그런데 Android에는 Apple 로그인 SDK가 없습니다.

구현 Kakao SDK와 Firebase Auth(Google·Apple)로 로그인을 구성하고, 탈퇴 시 각 IdP의 토큰·코드를 서버에 함께 넘겨 연결을 해제했습니다. Android에서 Apple 인증 창을 띄우면 App Links 없이는 리다이렉트를 기다리다 창이 닫혔고, 쿼리로는 id_token을 받을 수 없었습니다. 그래서 웹 페이지에서 Apple JS로 code와 id_token을 한 번에 받아 `glab://` 딥링크로 앱에 넘기는 브릿지를 만들었습니다.

결과 **세 플랫폼 모두 같은 로그인·탈퇴 흐름으로 스토어 심사 기준 충족**

그 외 맡은 일

- 파트장 업무로 프론트 기획과 디자인 검토, 배포 일정 결정, 팀원 2명의 업무 일정 관리, 출시 전 버튼 위치·화면 흐름 점검, 전체 커밋 검토와 코드리뷰 (동료 PR 265건 중 156건 머지), PR 템플릿 도입, 스토어 출시·OTA 업데이트 관리
- 디자인 산출물 없이 시작해 디자인 시스템(공통 컴포넌트와 테마·컬러 토큰)을 먼저 만들고 화면을 그 위에 구성. 뉴스 기사는 앱 내 웹뷰로 표시
- 실시간 랭킹에서 연결 전에 보낸 구독 해지가 늦게 도착해 로딩 화면에 멈추던 경쟁 조건 수정, 1초 안에 데이터가 없으면 다시 구독
- 실시간 시세 숫자 애니메이션 최적화. 자리마다 29개씩 그리던 행을 바뀌는 구간만 그리게 줄이고(1 차이일 때 3개), 간격을 서버 시세 주기에 맞춤
- 홈 개편(관심 종목 접기, 뉴스 필터, 탭 다시 누르면 맨 위로), 추천 질문을 채팅에서 홈으로 이동, 관심 등록 캐시 정합성
- 포트폴리오 매수·매도·편집·복사와 도넛·샤프 차트
- 로그인 세션 정합성. 서버 로그아웃이 실패하면 로컬 토큰을 지우지 않고 재시도를 안내하고, 로그인 직후 사용자 조회가 실패하면 서버 세션까지 정리
- 다국어(i18n) 체계 도입. 하드코딩된 문구를 한국어·영어 리소스로 옮기고, 기기 언어 감지와 개발 중 누락 키 표시·eslint 검사 추가
- 운영·개발·디버그 앱 환경 분리(앱 설정, EAS 프로필, Firebase 설정, 배포 대상)
- 웹은 웹 표준·접근성과 크로스 브라우징을 고려해 구현
- PostHog 분석, PWA, 모바일 웹에서 스토어 설치 유도, 점검 모드

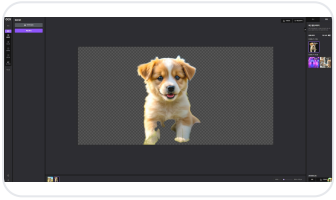
- 앱 만족도 피드백 팝업. 설치 후 3일, 사용 2일이 지난 사용자에게 AI 답변 평가나 채팅 5회 같은 시점에 띄우고, 접수된 피드백은 관리자 웹 고객 지원 화면에서 처리

관악연구소 · LG CNS 고객사 프로젝트 · 2025.01 – 2025.11

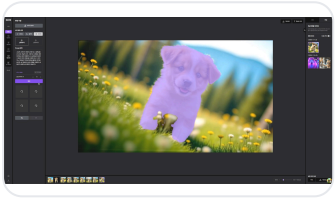
LG CNS COP · 생성형 AI 이미지 편집 플랫폼

텍스트로 이미지를 만들고 배경 확장·제거, 부분 편집, 업스케일링, 이미지 속 텍스트 편집까지 하는 웹 서비스입니다. AI 페스타에서 공개돼 언론에 보도됐고, LG CNS와 산돌구름의 협업으로 산돌구름 AI Studio 베타로 오픈했습니다. ADMIN 프론트엔드 전체를 설계·구현했고, 서비스 프론트엔드에서는 공통 컴포넌트·인증 구조와 개발·배포 자동화를 구축했습니다.

역할	팀 구성	공개	스택
백오피스(ADMIN) 프론트엔드 설계·구현 서비스 공통 기반·배포 자동화	프론트엔드 3명, 백엔드 2명	AI 페스타 · 디지털타임스 보도 산돌구름 AI Studio 베타 · 동작 영상	React, TypeScript, TanStack Query·Table, Zustand, Zod, Tailwind



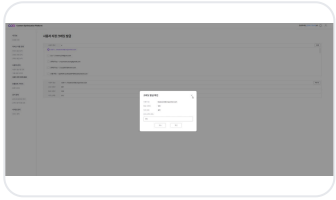
배경 제거



부분 편집 · 영역 선택



ADMIN · 사용 현황



ADMIN · 크레딧 발급

토큰 만료 시점에 동시에 발생하는 401

문제 토큰이 만료되는 순간 이미지 생성·편집 요청이 여러 개 나가면 한꺼번에 401을 받았습니다. 그러면 갱신 요청이 중복되거나, 옛 토큰으로 다시 요청하거나, 실패한 요청이 그대로 유실됐습니다. 내부망 인증 서버와 브라우저 호출이 맞물려 CORS·쿠키 처리도 프론트에 몰려 있었습니다.

해결 refresh token 재발급은 백엔드로 옮겨 인증 흐름을 하나로 만들었습니다. 프론트에서는 공통 인터셉터가 401을 모아 갱신을 한 번만 실행하고, 기다리던 요청은 새 토큰으로 다시 보냅니다. 갱신이 실패하면 대기 요청을 비우고 로그인 만료 상태로 전환합니다.

결과 **동시 요청 상황에서도 인증 실패 복구가 일관되게 동작, 서비스와 ADMIN에 같은 구조 적용**

요청 A · B · C
모두 401 응답
→
첫 401만 갱신 요청
나머지는 같은 갱신을 기다림
→
새 토큰 발급
갱신은 한 번
→
A · B · C 재전송
새 토큰으로 다시 보냄

갱신이 실패하면 대기 중인 요청을 비우고 로그인 만료 화면으로 보냅니다.

```
let refreshing: Promise<void> | null = null;

async function onUnauthorized(request: Request) {
  // 이미 갱신 중이면 그 갱신을 함께 기다린다. 갱신 요청은 한 번만 나간다.
  refreshing ??= refreshToken().finally(() => {
    refreshing = null;
  });

  try {
    await refreshing;
  } catch {
    return expireSession(); // 갱신 실패: 대기 요청을 버리고 로그인 만료로 전환
  }
  return retry(request); // 새 토큰으로 다시 보낸다
}
```

실제 코드가 아니라 구조만 단순화한 예시입니다.

역할별 권한과 크레딧 발급 ADMIN

상황 운영자가 사용자에게 크레딧을 수동으로 발급하고 계정·그룹·공지·정책을 관리해야 했고, 역할마다 볼 수 있는 기능이 달라야 했습니다.

구현 역할 4종에 따라 메뉴·탭·접근 경로를 나누고, 크레딧 발급은 마스터에게만 보이게 했습니다. 발급량은 숫자만 받아 10~9,999 범위를 벗어나면 버튼을 막고, 발급 전에 내용을 확인 모달로 한 번 더 보여 줍니다. 사용 현황 필터는 URL에 담아 새로고침·뒤로 가기에도 유지되게 하고, 그룹에서 사용자로 내려가 보는 드릴다운과 엑셀 내보내기를 넣었습니다.

결과 **ADMIN 전체(사용 현황, 크레딧 발급, 계정·그룹, 공지, 정책)를 테스트 일정 안에 완료**

공통 오류 처리와 배포 자동화

상황 여러 명이 서비스 화면과 ADMIN을 동시에 만들어야 해서, 로그인·인증·오류 처리가 화면마다 달라질 위험이 있었습니다.

구현 로그인과 공통 UI 구조를 설계하고 팀 공통 컴포넌트를 정리했습니다. 서버 오류는 쿼리 캐시에서 한곳으로 모아 상태·서비스 코드별로 토스트를 띄우고, 응답은 Zod로 검증해 화면마다 따로 처리하지 않게 했습니다.

자동화 husky와 lint-staged로 커밋 전에는 lint, push 전에는 lint와 타입 검사를 돌려 깨진 코드가 올라가지 않게 했고, main에 push하면 GitHub Actions가 빌드, S3 배포, CloudFront 캐시 무효화까지 처리하도록 했습니다.

그 외 맡은 일

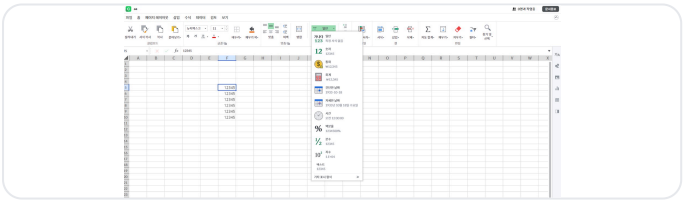
- 마이페이지·텍스트 편집 기능 구현과 세부 기획 참여, 테이블 편의 기능 제안
- DTO·서비스 문서를 작성해 백엔드와 API 형식 합의
- 웹 표준·접근성과 크로스 브라우징을 고려해 서비스와 ADMIN 화면 구현

티맥스가이아 · 2022.06 – 2025.01

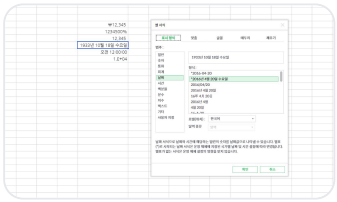
GAIA OFFICE Cell · 웹 스프레드시트

웹오피스의 스프레드시트입니다. 표시 형식, 피벗 테이블, 병합, 그래픽 요소를 개발했고, 2년 동안 이슈 약 555건을 처리했습니다(약 35%는 담당 기능 밖). 과기정통부 SW컴퓨팅산업원천기술개발 과제 「문서 데이터의 정보화 및 협업을 위한 지능형 문서 처리 플랫폼 기술」에 참여연구원으로 참여했습니다(2022.06 – 2023.12, 과제고유번호 1711152861·1711193911).

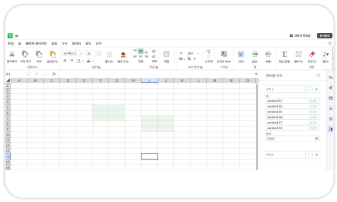
역할	스택	이슈 처리	데모
Cell 기능 개발·유지보수	React, TypeScript, Canvas, Java	약 555건	직접 해보기



홈 탭 · 표시 형식



셀 서식 · 표시 형식



문서 자동 완성 · 구역 설정

시트 전체를 병합하면 멈추는 문제 (2024.03 – 05)

문제 시트 전체(약 170억 셀)나 행·열 전체를 병합하면 오피스가 멈췄습니다.

원인 병합 전 값을 백업할 때 범위 안의 데이터 벡터를 빈 칸까지 전부 복사하고 있었습니다.

해결 빈 데이터는 건너뛰고, Dimension Reference로 실제 데이터가 있는 최소 범위만 복사하도록 바꿨습니다. 셀마다 만들던 서버 메시지는 범위 단위로 묶고, 하나로 섞여 있던 병합·병합 해제 명령을 분리했습니다. 셀·행/열·전체 선택에 따라 스타일 적용 범위도 다시 계산하고, Excel 2021의 병합 스타일 규칙을 분석해 맞췄습니다.

결과 **멈추던 전체 병합을 2초 안에 처리, 넓은 범위 병합 평균 10초 → 1.5초**

변경 전 선택 범위 전체를 빈 칸까지 복사 → **빈 데이터 제외** 값이 있는 셀만 대상 → **최소 범위 계산** 실제 데이터가 있는 영역 → **변경 후** 그 범위만 복사해 병합

복사 대상을 선택 범위 전체에서 실제 데이터 범위로 줄였습니다.

명령마다 반복되던 Canvas 렌더링

문제 명령 하나가 실행될 때마다 Canvas 갱신을 호출해서, 병합이나 스타일 적용처럼 여러 명령이 이어지는 동작에서는 같은 범위를 여러 번 다시 그렸습니다.

해결 명령이 도는 동안에는 갱신할 범위와 조건만 모아 두고, 명령이 끝나는 시점에 한 번만 그리게 했습니다. 처음에는 지연 호출(debounce)을 썼지만 빠르게 입력하면 화면이 멈춘 듯 보여서, 명령 종료 시점에 그리는 방식으로 바꿨습니다.

결과 **병합·스타일 적용 시 렌더 호출 5회 → 1회**

한국 서식을 지원하는 표시 형식(Number Format) (2023.02 – 05)

문제 numfmt 라이브러리를 썼지만 한국 날짜, 경과 시간(`[h]:mm:ss`), 원화 기호, 소수점 증감은 제대로 처리하지 못했습니다.

해결 입력값을 라이브러리가 이해하는 형식으로 바꾸는 변환 로직을 규칙마다 구현했습니다. 회계 서식은 Canvas `measureText` 로 글자 폭을 재서 통화 기호와 숫자 사이를 셀 너비에 맞게 띄웠습니다.

문서 자동 완성 TFT (2023.06 – 08)

- 외부 앱 데이터(사고 접수 등)를 Cell 문서의 지정 구역에 자동으로 채워 문서를 만드는 기능. TFT에서 가장 낮은 연차로 대표 기능을 담당
- Word·Point·Cell 공통 구역 관리 구조와 DB 테이블 설계, MERGE INTO로 중복 태그 Insert/Update를 한 번에 처리
- 배달서비스공제조합 서비스에 제공

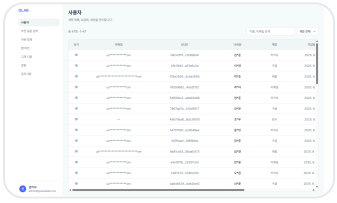
관악연구소 · 2026.08 – 현재

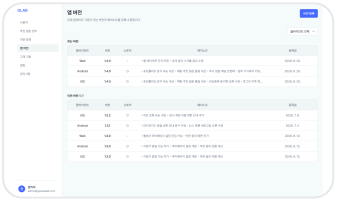
GLAB 백오피스(관리자 웹)

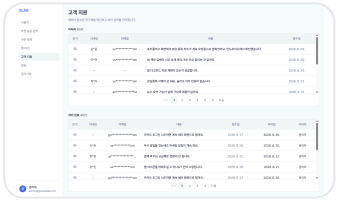
GLAB 앱 운영을 위한 백오피스(관리자 웹)이며 React SPA로 만들었습니다. 사용자, 공지, 약관, 앱 버전, 추천 질문, 고객 지원까지 21개 화면을 설계·구현하고 실제 API와 연동해 운영에 쓰고 있습니다.

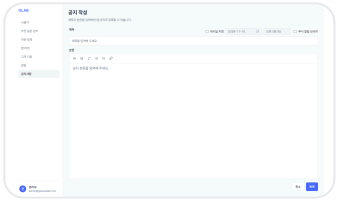
역할
기획·설계·구현

스택
Vite, React 19, React Router 7, Zustand, React Hook Form, Zod, TipTap


사용자 · 개인정보 마스킹


앱 버전 · 최신/이전 분리


고객 지원 · 처리 상태 관리


공지 작성 · 예약 게시·푸시

운영 규칙을 담은 화면과 입력 검증

상황 관리자 웹에서 바꾼 공지·약관·앱 버전은 그대로 앱 사용자에게 나갑니다.

판단 운영 중 실수가 나기 쉬운 지점을 먼저 정리해 화면과 입력 검증에 반영했습니다. 목 데이터로 흐름을 확인한 뒤 실제 API에 연결했습니다.

구현 공지는 지난 시각으로 예약할 수 없고, 이미 게시된 공지는 게시일과 푸시 설정을 바꿀 수 없게 했습니다. 사용 중인 약관 유형은 삭제를 막고, 앱 버전은 iOS·Android·웹별로 등록합니다. 버전 목록은 최신/이전 테이블을 나눠 지금 배포 중인 버전이 묻히지 않게 했습니다. 규칙은 Zod 스키마로 옮겨, 작성과 수정에 다른 규칙을 적용하고 약관 본문은 HTML 안에 실제 글자가 있는지까지 검사했습니다.

```
// 예약 게시는 현재 시각 이후만 받는다. 이미 게시된 공지를 고칠 때만 지난 시각을 그대로 둔다.
const noticeSchema = (options: { allowPast: boolean }) =>
  baseSchema.superRefine((value, ctx) => {
    if (!value.scheduled || options.allowPast) return;
    if (isPast(value.publishAt)) {
      ctx.addIssue({ code: 'custom', path: ['publishAt'], message: '현재 시각 이후로 선택해 주세요.' });
    }
  });
```

실제 코드가 아니라 구조만 단순화한 예시입니다.

에디터 재사용과 개인정보 마스킹

문제 관리자 웹에서 쓴 글이 앱에서 그대로 보여야 하고, 운영자 화면에 사용자 정보가 그대로 노출되면 안 됩니다.

구현 TipTap 에디터 하나를 출력 형식만 바꿔 재사용해, 공지는 Markdown, 약관은 앱의 HTML 렌더러에 맞춘 HTML로 저장하게 했습니다. 사용자 목록과 고객지원 화면에서는 이메일·닉네임을 기본으로 가리고, 필요한 행 하나만 열어 볼 수 있게 했습니다.

그 외 맡은 일

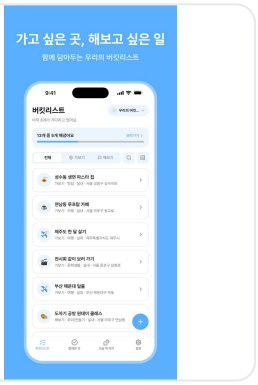
- 목록 화면은 제목·검색·필터와 표 머리글을 고정하고 행만 스크롤. 검색은 버튼을 눌렀을 때만 제목·본문 조회
- 고객 지원은 미처리/처리 완료 목록을 나누고 처리일·처리자 표시
- 브라우저 기본 영문 검증 대신 해당 입력 아래에 한국어 오류 표시. 웹 표준에 맞는 태그를 쓰고, 입력마다 label을 연결하고 오류 상태를 aria-invalid로 알리는 등 접근성 속성 적용

개인 프로젝트 · 2026.10 – 현재 · iOS·Android 출시

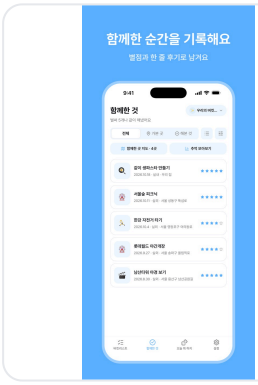
DoIt2 · 함께 쓰는 버킷리스트 앱

연인·친구·가족이 공간을 만들어 가고 싶은 곳과 해보고 싶은 것을 함께 모으고, 다녀온 뒤 사진과 후기를 남기는 앱입니다. 기획, 앱, Supabase 백엔드, App Store 심사 대응과 출시까지 2주 반 만에 진행했고, 이후 AI 추천과 여행 일정 웹 GoIt2 연동을 더했습니다(커밋 140개).

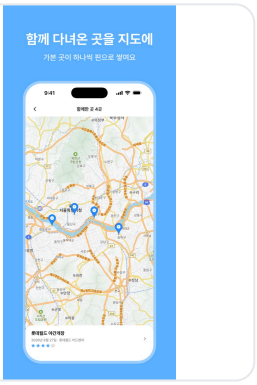
역할	규모	스택	링크
기획·설계·개발 코딩 에이전트(Claude Code) 활용	DB 마이그레이션 43개, RLS 정책 58개 Edge Function 7개, Jest 테스트 251개(40개 파일)	Expo SDK 58, Expo Router, NativeWind, Zustand, Supabase, Claude API, Jest	App Store · Google Play · 소개 사이트 github.com/epww1502/DoIt2



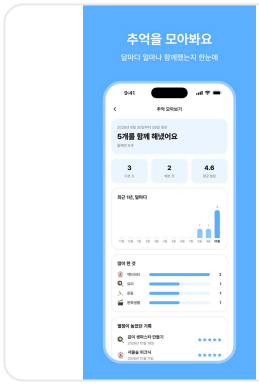
버킷리스트



함께한 기록



다녀온 곳 지도



추억 돌아보기

여러 명이 동시에 수정할 때의 동기화

문제 한 공간을 여러 명이 같이 쓰기 때문에, 두 사람이 동시에 체크리스트를 고치면 한쪽 변경이 덮어써질 수 있었습니다.

해결 목록 전체를 덮어쓰지 않고 체크, 추가 같은 변경 단위로 기록해 서버의 최신 상태 위에 다시 적용했습니다. 같은 변경이 두 번 전송돼도 결과가 달라지지 않고, 오프라인일 때는 변경을 쌓아 두었다가 연결되면 보냅니다.

검증 두 사람이 서로 다른 항목을 체크해도 둘 다 남는지, 같은 변경을 두 번 보내도 결과가 같은지를 Jest 테스트로 작성했습니다.

```
it('두 사람이 다른 할 것을 체크해도 둘 다 남는다', () => {
  const server = [todo('a'), todo('b')];
  // 나는 a, 상대는 b를 체크 (어느 쪽이 먼저 반영되든 같은 결과)
  const result = applyTodoOps(
    applyTodoOps(server, [{ type: 'check', todoId: 'a', done: true }]),
    [{ type: 'check', todoId: 'b', done: true }],
  );
  expect(result.map((t) => t.done)).toEqual([true, true]);
});

it('같은 변경을 두 번 보내도 결과가 같다', () => {
  const ops = [{ type: 'check', todoId: 'a', done: true }, { type: 'add', todo: todo('c') }];
  const once = applyTodoOps([todo('a')], ops);
  expect(applyTodoOps(once, ops)).toEqual(once);
});
```

저장소의 `__tests__/todoOps.test.ts` 에서 가져와 줄였습니다.

RLS와 Edge Function으로 권한 관리

상황 같은 공간 사람끼리만 데이터를 봐야 하는데, 앱 코드만으로 막으면 요청을 직접 보내는 경우를 막을 수 없습니다.

구현 모든 표에 RLS를 걸어 공간 멤버만 읽고 쓰게 하고, 공간 참여·공간장 위임처럼 여러 표를 바꾸는 작업은 DB 함수로 묶었습니다. 카카오 API 키는 Edge Function 프록시 뒤에 두고, 링크 미리보기는 내부망 주소를 리다이렉트 이후까지 막았습니다. 푸시 알림과 계정 삭제도 서버 함수로 처리했습니다.

심사 대응 App Store 심사에서 신고·차단 기능(가이드라인 1.2)을 요구해, 신고는 쓰기만 가능하고 차단한 사람의 댓글은 보이지 않도록 RLS로 추가했습니다.

AI 추천: 장소와 할 것까지 채워 주고, 평가 세트로 측정

도입 목록이 비어 있거나 무엇을 담을지 떠오르지 않을 때를 위해 넣었습니다. 담아 둔 목록과 사용자가 말한 조건("비 오는 날 실내에서", 이어서 "더 저렴하게")을 보고 새 항목 5개를 장소와 할 것까지 채워 추천합니다. Claude Haiku 5.5를 Edge Function에서 호출해 API 키를 서버에만 두고, 한 사람 하루 10회와 전체 하루 2,000회 한도를 두었습니다.

판단 모델의 답을 그대로 쓰지 않도록 했습니다. 답의 모양은 JSON 스키마로 고정하고, 장소는 모델이 적은 이름을 카카오 장소 검색에서 찾아 이름이 맞을 때만 지도에 표시합니다. 맞는 곳이 없으면 지역 글자만 남깁니다. 사용자가 쓴 말은 지시가 아니라 참고할 내용으로만 전달합니다. 첫 추천이 보이기까지 9초쯤 걸려서, 답을 스트리밍으로 받아 진행 단계와 "5개 중 n개"를 보여 줍니다.

측정 출시 초기라 실제 사용 기록이 적어, 요청 24개(조건 하나, 조건 여러 개, 이어서 말하기, 요청 없음, 엉뚱한 말)를 직접 만들어 배포된 함수에 2번씩 보내 추천 240개를 채점했습니다. 개수, 길이, 실내/야외, 지역 같은 규칙은 코드로 검사하고, 요청에 맞는지와 할 것이 쓸모 있는지는 다른 모델(Sonnet 5.5)이 항목마다 근거를 쓰고 판정하게 했습니다. 심사 근거는 결과 파일에 남겨 직접 다시 확인할 수 있게 했습니다.

결과 **요청에 맞음 235/240(97.9%), 할 것 쓸모 235/240, 규칙 130/130, 장소 실재 121/123. 첫 추천까지 가운데값 8.9초, 다섯 개 모두 10.9초, 한 번에 약 $0.001. 48번 모두 실패 없이 끝났고, 지시를 바꾸려는 말에도 2번 모두 평소대로 추천했습니다.**

알게 된 것 모델이 종류를 잘못 적은 10건은 모두 같은 혼동(카테고리 이름 `activity` 와 종류 이름 `activity`)이었고, 앱에서 카테고리에 맞춰 종류를 다시 정하는 코드가 전부 바로잡았습니다. 장소 이름은 정확했지만 구 단위 지역은 가끔 틀려서, 장소를 찾으면 모델이 적은 지역 대신 카카오 주소를 씁니다. 조건이 여러 개인 요청이 94.3%로 가장 약해 다음 개선 대상입니다.

```
// 요청 하나: 담아 둔 목록, 사용자가 말한 것, 코드로 검사할 규칙, 심사 기준
{
  "id": "follow-busan-sights",
  "list": "couple",
  "requests": ["부산 여행", "맛집 말고 구경할 곳 위주로"],
  "check": { "locationIncludes": ["부산"], "categoryNotIn": ["food", "cafe"] },
  "rubric": "다섯 개 모두 부산에서 하는 것이고, 먹는 곳이 아니라 구경하는 곳이어야 한다."
}

// 장소 실재: 앱이 담을 때와 같은 검색, 같은 판정
async function placeFound(it) {
  const { places } = await (await call('kakao-search', { query: it.place })).json();
  const want = norm(it.place);
  return places.some((p) => norm(p.name).includes(want) || want.includes(norm(p.name)));
}
```

저장소의 `scripts/ai-eval` 에서 가져와 줄였습니다. 요청 24개와 결과, 한계는 그 폴더의 README에 있습니다.

그 외 작업

- 소개 사이트와 초대 링크·돌아보기 공유 웹 페이지: 초대 페이지는 코드 추측 방어를 유지하려고 서버 조회 없이 구성하고, 공유 페이지는 고른 사진만 위치 정보(EXIF)를 뺀 사본으로 30일간 공개
- GoIt2와 연동: 항목에서 여행 일정 짜기, 고잇투 일정을 장소마다 또는 여행 하나로 원하는 공간에 담기 (GoIt2 사례에 자세히)
- OTA·강제 업데이트, 오류는 기기에 먼저 저장했다가 서버로 전송, 사진은 올리기 전에 줄이고 저장소 경로 기준으로 캐시
- 인터넷이 잠깐 끊기면 새 계정으로 바뀌던 문제: 토큰 갱신이 네트워크로 실패했을 때는 새 계정을 만들지 않고, 계정 코드를 미리 만들어 두어 로그인이 사라져도 같은 계정으로 돌아가게 함
- 새 버전을 다 받으면 '다시 시작' 띠를 보여 주고, 쓰다가 돌아왔을 때 저절로 다시 켜지지 않게 함
- 목표 날짜(하루 또는 기간): 캘린더에 넣기, 날씨 예보, 다녀온 다음 날 기록 알림, 다가오는 일정 모아보기
- 같이 갈 날 달력: 멤버마다 가능한 날을 고르면 이어진 날을 한 블록으로 묶어, 내 날과 남의 날을 구분해 보여 줌

개인 프로젝트 · 2026.11 – 현재 · 웹 서비스 운영 중

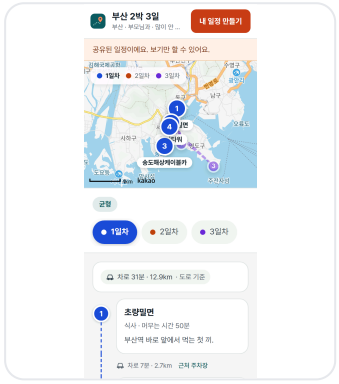
GoIt2 · AI 여행 일정 웹

어디로, 누구와, 어떻게 가고 싶은지 한 줄로 쓰면 지도 위에 날짜별 동선을 그려 주는 여행 일정 웹입니다. AI가 고른 장소는 카카오 장소 검색으로 확인한 곳만 넣고, DoIt2와 이어 담아 둔 곳으로 일정을 짜고 다녀온 뒤 기록까지 남기게 했습니다.

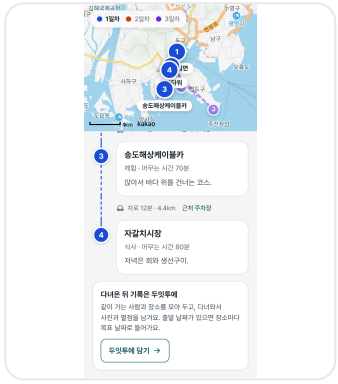
역할	규모	스택	링크
기획·설계·개발 코딩 에이전트(Claude Code) 활용	PR 16개, Edge Function 1개 Jest 테스트 102개, Playwright E2E 19개, PR마다 CI	Next.js 16, React 19, Tailwind, Claude API(Haiku 5.5), Supabase Edge Functions, Kakao Map·Local·Mobility, Vercel	go-it-2.vercel.app



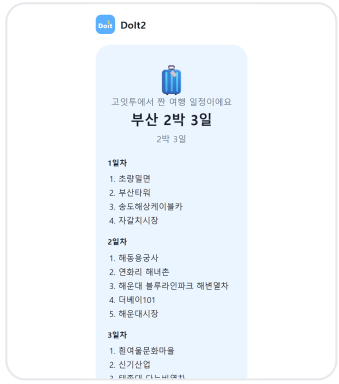
조건 한 줄로 시작



날짜별 동선과 실제 도로 시간



두잇투에 담기



두잇투 담기 페이지

AI가 지어낸 장소를 일정에 넣지 않기

문제 언어 모델은 그럴듯한 가게 이름을 지어내거나, 있는 곳이라도 좌표와 주소를 틀리게 적습니다. 그대로 지도에 올리면 가 보니 없는 곳이 일정에 들어갑니다.

해결 모델은 장소 이름, 종류, 머무는 시간, 고른 이유만 정하게 했습니다. 좌표와 주소는 카카오 장소 검색에서 이름이 맞고 주소가 여행 지역 안인 곳으로만 채우고, 찾지 못한 곳은 빼고 화면에 알립니다. 식당과 카페는 같은 동네의 대안 1~2곳을 함께 받아, 모두 확인한 뒤 블로그 언급이 가장 많은 곳을 고릅니다.

결과 **지도에 오르는 장소는 모두 카카오에서 확인된 곳입니다. 차로 가는 구간은 카카오모빌리티 길찾기로 실제 도로 시간과 거리를 보여 주고, 도착지 근처 주차장도 찾아 줍니다.**

기다리지 않게: 한 줄씩 받아 바로 확인하기

문제 처음 실제로 만든 1박 2일 일정은 26초쯤 걸렸습니다. 답이 다 와야 장소를 확인하기 시작했고, 장소마다 검색을 차례로 기다렸습니다.

해결 모델이 한 줄에 JSON 하나씩 쓰게 하고, 줄이 오는 대로 장소 확인을 시작했습니다. 확인은 동시에 돌리되 화면에는 모델이 쓴 순서대로 내보내고, 장소마다 하던 두 번의 검색도 함께 보냅니다.

판단 하루 장소 수를 지키게 하려고 규칙을 넣자 모델이 장소 수를 세느라 2,164토큰을 생각에 써서 첫 줄이 13.7초로 늦어졌습니다. 규칙은 프롬프트에 두고 생각은 꺼서 장소 수와 첫 줄 속도를 함께 지켰습니다.

결과 **1박 2일이 서버 기준 4.2초(첫 장소까지 1.1초), 화면 기준 5.8초. 만드는 동안 단계와 확인된 장소가 실시간으로 지도에 올라갑니다.**

조건 한 줄
지역·동행·스타일·속도·숙소·날짜
→
Claude
한 줄에 장소 하나 (NDJSON)
→
카카오 확인
이름·지역이 맞는 곳만, 동시에
→
지도
모델이 쓴 순서대로 표시

모델이 한 줄을 쓰면 그 장소의 확인이 바로 시작됩니다.

로그인 없는 공개 웹에서 비용과 권한 지키기

상황 누구나 로그인 없이 쓰는 웹이라 서버 함수를 아무나 부를 수 있고, 한 번 부를 때마다 AI와 지도 API 비용이 듭니다.

구현 Claude와 카카오 키는 Supabase Edge Function 뒤에 두고, 허용한 사이트 주소에서 온 요청만 받습니다. 하루 횟수는 방문자별과 전체 합계를 DB 함수로 함께 세어, 주소가 퍼져도 하루 요금이 합계에서 멈춥니다. 방문자는 IP를 그대로 저장하지 않고 서버 키로 서명한 해시로만 구분합니다.

방어 사용자 글은 여행 조건으로만 읽게 지시하고, 모델의 답은 정해 둔 항목과 길이만 통과시킵니다. 공유 링크는 맞힐 수 없는 id를 쓰고 30일 뒤 만료됩니다.

DoIt2와 잇기: 담아 둔 곳 → 일정 → 기록

고잇투 → 두잇투 일정 화면의 "두잇투에 담기"가 공유 링크를 만들고 두잇투의 웹 페이지를 엽니다. 휴대폰이면 앱을 바로 열고, 컴퓨터에서 열었으면 휴대폰으로 옮길 QR과 링크를 보여 줍니다. 앱에서는 장소마다 담을지 여행 하나로 묶을지, 어느 공간에 담을지 고르고, 출발 날짜가 있으면 목표 날짜로 들어갑니다.

두잇투 → 고잇투 두잇투 항목의 "고잇투로 일정 짜기"가 고른 곳을 "꼭 넣을 곳"으로 실어 고잇투를 엽니다. 서버는 그 곳을 모두 넣고 나머지를 가까운 곳으로 고르라고 모델에 전합니다.

권한 로그인 없이 공유 id로만 잇고, 두잇투 앱과 웹 페이지는 서버 함수에서 공유 일정 읽기만 할 수 있게 막았습니다. 배포 뒤에는 실제 요청으로 읽기는 되고 생성·저장·길찾기는 막히는지 확인했습니다.

그 외 작업

- 편집: 끌어서 순서 바꾸기, 날짜 옮기기, 고정, 되돌리기, 동선 다시 정렬. 말로 고치기는 바뀔 곳을 미리 보여 주고 적용하거나 취소
- 출발 날짜가 16일 안이면 날짜별 예보, 장소별 1인 비용 어림과 합계, 인쇄(PDF)와 캘린더 파일(.ics) 내보내기
- 공유 페이지는 Next.js SSR: 요청 때 서버에서 일정을 읽어, 메신저에 붙이면 일정 제목·날짜별 장소·동선 그림이 미리보기 이미지로 나옴
- JS가 받아지기 전에도 첫 화면의 입력이 동작하게 해 느린 연결에서 첫 요청을 앞당김
- Jest 102개, Playwright E2E 19개(데스크톱·모바일), 외부 API를 가짜로 바꿔 서버 함수를 돌려 보는 점검 스크립트, PR마다 GitHub Actions CI

개인 프로젝트 · 2026.11 – 현재 · macOS·Windows 배포

mochi-pet · 친구들과 함께 쓰는 데스크톱 펫

초대 코드로 같은 방에 들어온 친구들의 픽셀 펫이 화면 아래를 걸어 다니며 접속 상태를 보여 주고, 말풍선 채팅·이모티콘·간식·쪽지를 주고받는 데스크톱 앱입니다. 투명한 창 위에서 다른 앱 작업을 막지 않게 만들고, 태그 하나로 macOS·Windows 빌드와 서명된 자동 업데이트까지 배포되게 했습니다.

역할	플랫폼	규모	링크
기획·설계·개발·배포 코딩 에이전트(Claude Code) 활용	macOS(Apple Silicon·Intel) · Windows	Vitest 테스트 102개, 브랜치마다 두 OS 빌드 검사	GitHub · 다운로드

화면을 덮는 투명 창이 다른 앱 클릭을 막지 않게

문제 펫은 화면 아래를 걸어 다니고 다른 앱 창 위로 올라가기도 해서, 화면 전체를 덮는 투명 창 위에 그립니다. 그런데 이 창이 클릭을 받으면 뒤에 있는 앱을 누를 수 없고, 클릭을 통과시키면 마우스 이벤트가 오지 않아 커서가 펫 위에 왔는지도 알 수 없습니다.

해결 React가 0.2초마다 펫·말풍선·입력창이 차지한 사각형을 Rust 쪽에 넘기고(바뀌었을 때만), Rust는 0.04초마다 전역 커서 위치를 읽어 사각형 안일 때만 창이 클릭을 받게 바꿉니다. 펫을 끌 때는 커서가 펫 밖으로 나가도 놓치지 않게 잠깐 창 전체가 클릭을 받습니다.

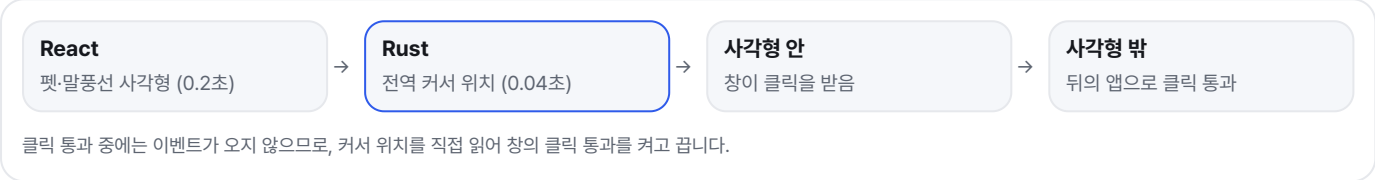
결과 **펫과 말풍선은 누르고 끌 수 있고, 그 밖의 화면은 평소처럼 뒤에 있는 앱이 클릭을 받습니다. 맨 앞 창의 위치만 OS API로 읽어(내용·제목은 읽지 않음) 펫이 그 창 위 모서리로 올라가게 했습니다.**

배율이 다른 외장 모니터에서 펫이 클릭되지 않던 문제

상황 맥북 화면(2배)과 일반 외장 모니터(1배)를 같이 쓰면, 한쪽 모니터에 있는 펫이 눌리지 않았습니다.

원인 맥에서는 커서 좌표가 주 모니터 배율로, 창 위치는 창이 있는 모니터 배율로 곱해져 옵니다. 두 배율이 다르면 그대로 뺀 값이 어긋나, 커서가 펫 위에 있어도 사각형 밖으로 계산됐습니다.

해결 커서와 창 위치를 각자의 배율로 되돌린 뒤에 빼도록 바꾸고, 두 모니터 배치(맥북이 주 모니터일 때와 외장이 주 모니터일 때)를 Rust 테스트로 고정했습니다. 윈도우는 둘 다 실제 픽셀이라 기존 계산을 그대로 씁니다.



펫은 보이는데 채팅만 안 오던 문제

상황 Windows 친구 화면에서 펫은 접속 중으로 보이는데 채팅과 간식이 오지 않았습니다. 앱은 1분마다 서버에 접속 기록을 남기고 그 기록으로도 접속 중을 표시해서, 실시간 연결이 끊겨도 겉으로는 멀쩡해 보였습니다.

원인 WebView2에 원격 디버깅 포트를 열어 실행 중인 앱을 들여다보니, 실시간 채널이 끊긴 뒤 다시 붙지 않고 그대로 남아 있었습니다. 다시 붙더라도 같은 채널 객체라 화면 상태가 바뀌지 않아 내 접속 상태를 다시 알리지 않았습니다.

해결 채널이 오류·시간 초과·닫힘이면 5초 뒤 새로 만들고, 1분마다 보는 지킴이가 두 번 연속 끊긴 상태면 새로 연결합니다. 인터넷이 다시 붙으면 바로 연결하고, 붙을 때마다 내 상태를 다시 보냅니다. 끊긴 동안 보낸 채팅은 대화 기록에 남겨 두고 그 사실을 알립니다.

결과 **잠자기·네트워크 변경 뒤에도 앱이 스스로 다시 붙고, 연결 상태와 상관없이 "접속 중"으로만 보이던 착시를 없앴습니다.**

실시간 상태와 마지막 접속을 한 화면에

구현 지금 켜져 있는 친구는 Supabase Realtime presence로, 꺼진 친구는 서버의 마지막 접속 시각으로 보여 줍니다. 두 목록을 합칠 때 접속 중인 친구를 앞에 세우고, 30일 넘게 안 들어온 친구는 빼고, 내 컴퓨터 시계가 서버보다 빨라도 "앞으로 N분"이 되지 않게 맞췄습니다. 이름으로 같은 사람의 다른 컴퓨터로 보이면 한 번만 보여 줍니다.

방어 친구에게서 오는 채팅·반응·놀이 메시지는 받자마자 모양과 길이를 검사해, 이상한 값 하나로 화면 전체가 멈추지 않게 했습니다. 꺼진 친구에게 남기는 쪽지는 표를 직접 읽고 쓸 수 없게 막고 서버 함수로만 남기고 받게 했으며, 받으면 지워지고 한 사람 앞으로 20개까지만 쌓입니다.

태그 하나로 두 OS에 서명된 업데이트 배포

구현 브랜치마다 GitHub Actions가 macOS·Windows에서 타입 검사·테스트·빌드를 돌립니다. 버전 태그를 올리면 macOS 2종(코드 서명·공증)과 Windows 설치 파일을 만들고, 셋 다 성공해야 자동 업데이트용 latest.json을 공개합니다. 하나라도 실패하면 이전 버전이 그대로 최신으로 남습니다. 앱은 서명을 확인한 업데이트만 받습니다.

개선 배포 직후 "업데이트가 안 뜬다"는 문의를 받고 확인해 보니, 앱이 켤 때와 6시간마다만 새 버전을 확인하고 있었습니다. 확인 간격을 1시간으로 줄이고 펫 메뉴에 지금 버전과 "업데이트 확인"을 넣었으며, 업데이트 알림이 다른 알림에 가려지지 않게 알림을 차례로 쌓도록 바꿨습니다.

그 외 작업

- 화면 배율(레티나, Windows 125% 등)에 맞춰 한 칸을 정수 픽셀로 그리는 캔버스 픽셀아트, 내 그림을 펫으로 쓸 때 투명 여백 자동 정리
- 방장: 친구 내보내기와 방장 넘기기, 방장이 나가거나 30일 넘게 안 오면 가장 먼저 들어온 사람에게 넘김(서버 함수)
- 대화 창: 메신저처럼 보고 보내기, 끌어서 옮기기, 연달아 한 말 쌓아 보이기. 친구 펫은 내 화면에서만 끌어서 자리 옮기기
- 펫과 모자를 클릭으로 골라 끌어서 옮기고 크기를 바꾸는 편집 화면
- 펫 창과 설정 창이 localStorage로 설정과 명령을 주고받는 두 창 구조
- 이벤트: 생일·퇴근 시간·기념일·방 친구가 다 모였을 때, 요일·시간에 따른 펫 기분, 가까운 펫끼리 하이파이브