フロント × バックの型ズレ、ちゃんと倒せてますか？

Orval × Swagger で型同期してわかった
3 つの型設計の学び

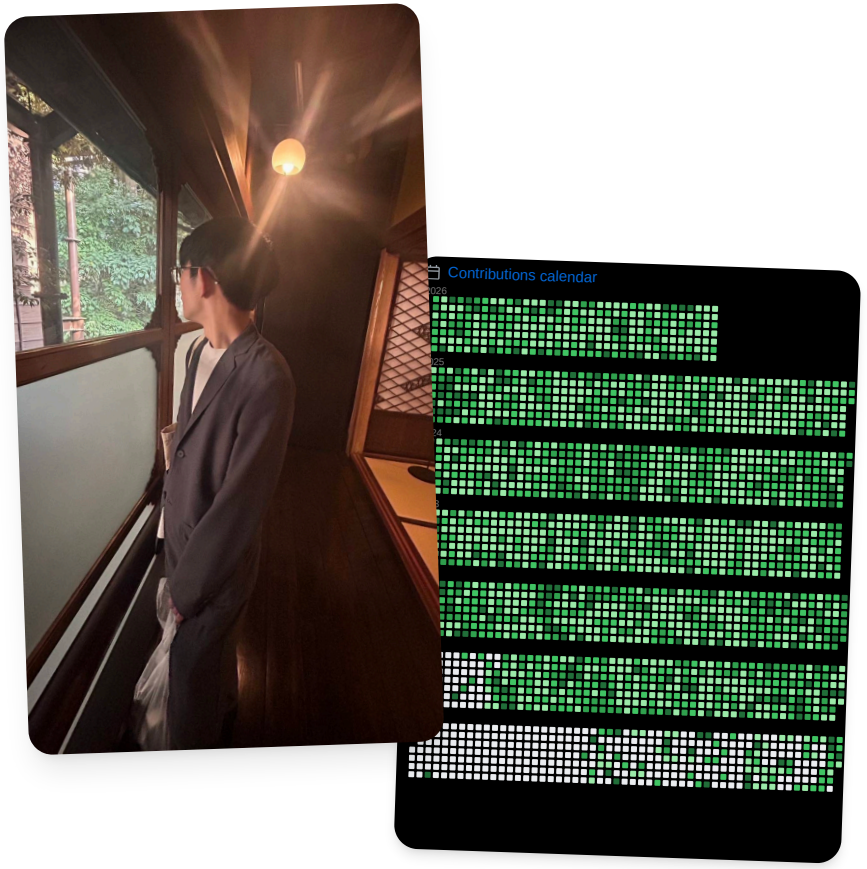
Yoriai.tech LT / @illionillion

自己紹介

セイ

フロントエンドエンジニア・25新卒

- Next.js / TypeScript
- X: `@Sei_engineer`
- 趣味
  - 個人開発・GitHub毎日更新！！
  - 飲み・海鮮
  - シーシャ・カラオケ



FE エンジニアの人〜？ 🙋（BE の方は「FE にどう設計してほしいか」の参考に）

こんな経験、ありませんか？

「バックが変わったら API の型がズレて動かなくなった」

「仕様書通りに実装したのに、綴りミスで型が合わない…」

「言語が違うから、型を手で合わせ続けるのがしんどい」

言語をまたぐと、型は必ずズレる

型同期、どうしてますか？

1️⃣ 仕様書（Swagger）を手で更新

課題: 工数大・実装と乖離しやすい

2️⃣ バックエンドも TS で書く

課題: 既存プロダクト・制約がある現場では難しい

3️⃣ ツールで自動同期

課題: 🤔

```
Go コード →[swaggo]→ swagger.json →[orval]→ TS 型 + API クライアント
```

バックが変わると **コマンド 1 つで型が自動更新** される

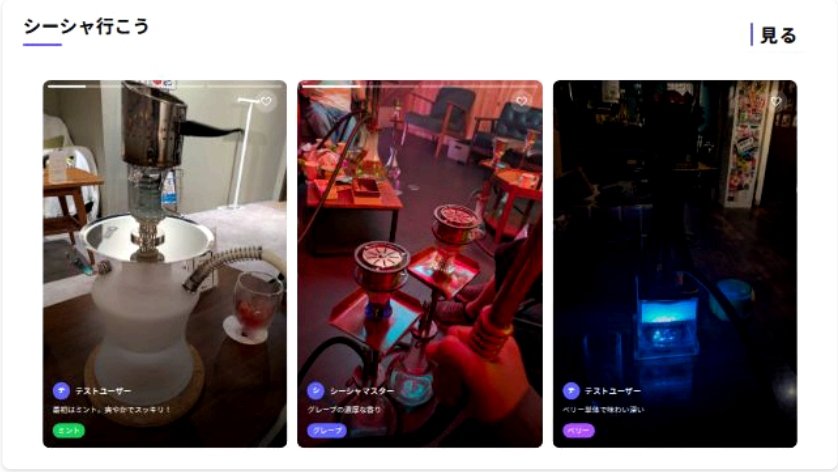
でも ― **ツールを入れたら終わりじゃなかった**

今日の話のベース

https://github.com/illionillion/go-shisha

シーシャ（水タバコ）記録アプリ

- **フロント**: Next.js App Router
- **バック**: Go + Clean Architecture
- Orval × Swago で型を自動同期



ツールを入れた後の
型の扱い方が大事だった

Next.js × Go の個人開発で気づいた 3 つのこと

① 生成型を直接 import しない

生成された型、そのまま使いたいですよね…？

❌ 直接 import

```
// ❌ 生成型をコンポーネントで直接 import
import type { GoShishaBackendInternalModelsUser } from "@/api/model/..."
```

バックが型名を変えると → **直接 import していた全ファイルで型エラーが爆発** 💥

型エラーが爆発

```
- import type { GoShishaBackendInternalModelsUser } from "@/api/model/..."
+ import type { GoShishaV2InternalModelsUser } from "@/api/model/..."
// components/UserCard.tsx ← 型エラー  pages/profile.tsx ← 型エラー  ...
```

`types/domain.ts` **を中間層として挟む → 修正は 1 ファイルだけ**

✅ domain.ts

```
// ✅ ここだけ @/api/model を知っている
export type User = GoShishaV2InternalModelsUser; // ← ここだけ直す
```

ESLint の `no-restricted-imports` で直接 import を **禁止して強制する**

② ドメイン型を丸ごとコンポーネントに渡さない

❌型をそのまま渡す

```
// ❌ 型をそのまま渡す
<UserCard user={user} />
// user.name → user.displayName に変わったら
// コンポーネントの「中」で型エラーが出て、中を直しに行くことになる
```

✅必要なフィールドだけ

```
// ✅ 必要なフィールドだけ渡す
<UserCard
-  username={user.username}
+  username={user.displayName}
/>
// 型エラーは呼び出し元（外側）だけに発生する
```

「壊れる場所」を外側に固定する という考え方

コンポーネントの内側はバックの変更に無関心でいられる

③ エラーメッセージはフロントで管理する

メッセージの文言って、フロントの都合で決めたくないですか？

エラーコードだけ返してもらい、メッセージはフロントで管理する

エラーメッセージ定義

```
const LIKE_ERROR_MESSAGES = {
  [ConflictErrorCode.already_liked]:     "この投稿にはすでにいいねしています",
  [UnauthorizedErrorCode.unauthorized]: "ログインが必要です",
  [NotFoundErrorCode.not_found]:         "投稿が見つかりません",
} satisfies Record<LikeErrorCode, string>;
//  ^^^^^^^^ エラーコードの網羅チェックをコンパイル時に強制
```

- メッセージはフロントの UI・言語・トーンに合わせたい → **バックに任せると辛くなる**
- `satisfies` によって **新しいエラーコードが追加されたら型エラーで気づける**
- エラーの型も Orval で生成できる → catch 内の型が安全になる
- おまけ: バックのメッセージをそのまま出すと **XSS のリスク** にもなる

ツールを入れても、型設計は終わらない

① 中間層を挟む

`domain.ts` で生成型を包む
変更を 1 ファイルに封じ込める

② 型を分解して渡す

ドメイン型を丸ごと渡さない
「壊れる場所」を外側に固定

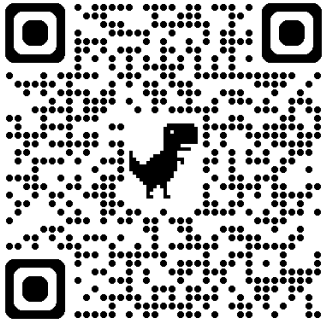
③ メッセージはフロントで管理

バックはコードだけ返す
`satisfies` で網羅漏れを検知

フルスタック TS でなくても、**この 3 つは同じように使える**

ありがとうございました 🙏

👇リポジトリの QR👇



𝕏：Sei_engineer　／　Slack：times_ichinose