# Language agnostic E2E type safety with OpenAPI Generator

## Who am I?

- Patrick Huijten 🇳🇱
- Studied **3D animation** (2011 - 2015)
  - Learned quickly that I needed to learn coding
- Started a Unity dev
  - When they were still **cool** 👀
  - Transitioned to web development
  - Unity dev ➡️ Startup ➡️ digital agency
- Full-stack Software engineer @ **Ikea**
  - Building a full-stack product for receiving returns inside the Ikea store
  - We’re almost live in all stores worldwide (Germany, Denmark & China tbd)
- **Personally note:** I (surprisingly 🇳🇱) very much like mountains and hiking 🏔️🥾

## Raise of hands

- Who does primarily **FE**?
- Who does primarily **BE**?
- Who does both?
- Who uses **different languages** between BE & FE?
- Who has heard of OpenAPI?
- Who actively uses OpenAPI in their project?

## Who is this for?

- Software engineers in **product teams** that consume a REST API

- Back-end engineers building REST APIs
  - **Any** language, not just TS
  - Support for all the major languages

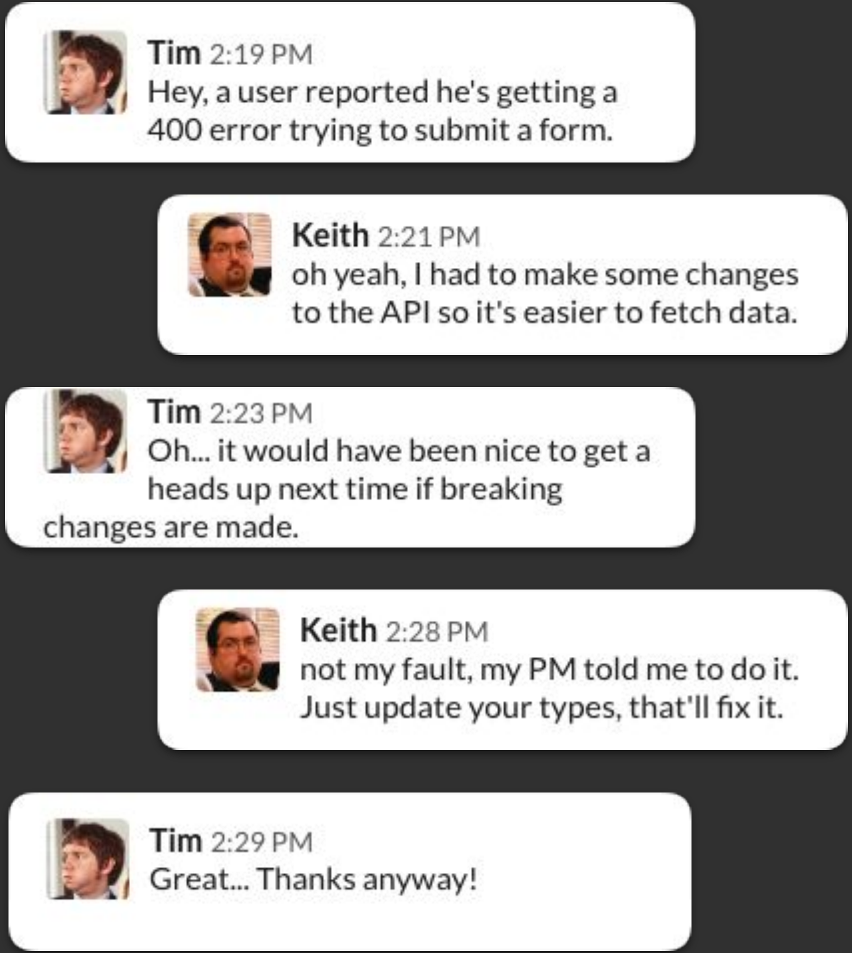
- Front-end engineers **working together** with BE engineers

- SDK maintainers for a public API

## Problem

Have you had this conversation?

- FE is dependent on BE
- API changes are fragile
- Unhappy users
- Unpleasant atmosphere for engineers



**Tim** 2:19 PM
Hey, a user reported he's getting a 400 error trying to submit a form.

**Keith** 2:21 PM
oh yeah, I had to make some changes to the API so it's easier to fetch data.

**Tim** 2:23 PM
Oh... it would have been nice to get a heads up next time if breaking changes are made.

**Keith** 2:28 PM
not my fault, my PM told me to do it. Just update your types, that'll fix it.

**Tim** 2:29 PM
Great... Thanks anyway!

## Common responses

- “Use GraphQL!”
  - If you’re happy using GraphQL, keep on using it 🚀
  - However…
    - Extra layer of complexity & point of failure
    - Non-trivial learning curve
- “Use gRPC!”
  - If you’re happy using gRPC, keep on using it 🚀
    - However…
      - It’s a bit overkill for an intermediate FE/BE application
      - Protobuffers take some getting used to
      - TRPC is nice, but only for Typescript
- You probably have more… (tell me more after the talk!)

## Best of both worlds: OpenAPI (spec first)

- Instead of generating the spec from your BE code…
  - Generate BE (controllers, models, routes, etc) code from the spec
  - Generate FE (types, service methods, etc) from the spec
- Use BE / FE language of **your choice** (java, C#, PHP, Node, Android, etc)
- **TLDR**: When starting work on a full-stack feature, **start with the spec**
- Process
  - Define FE/BE needs **before starting** work
  - Agree on spec **together**
  - Commit / deploy / push spec
  - Generate your boilerplate & start work **in parallel**
  - Merge BE code **then** FE code
    - *Sometimes even possible to do independent deployments*

## Advantages

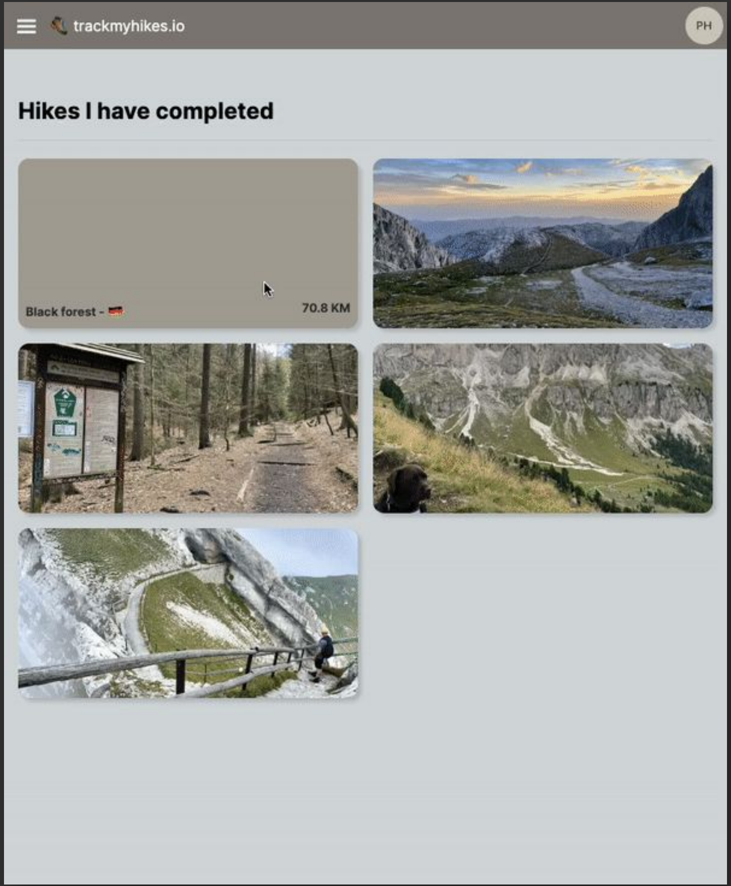
- Start the conversation **early**
- Faster dev cycles by allowing BE & FE to work **in parallel**
- Removes the need to maintain multiple type systems, focus on **business logic**
- Fosters **collaboration** and **aligns** expectations
- OpenAPI has great **tooling** (postman, wiremock, apidoc, etc)

## Acknowledgements

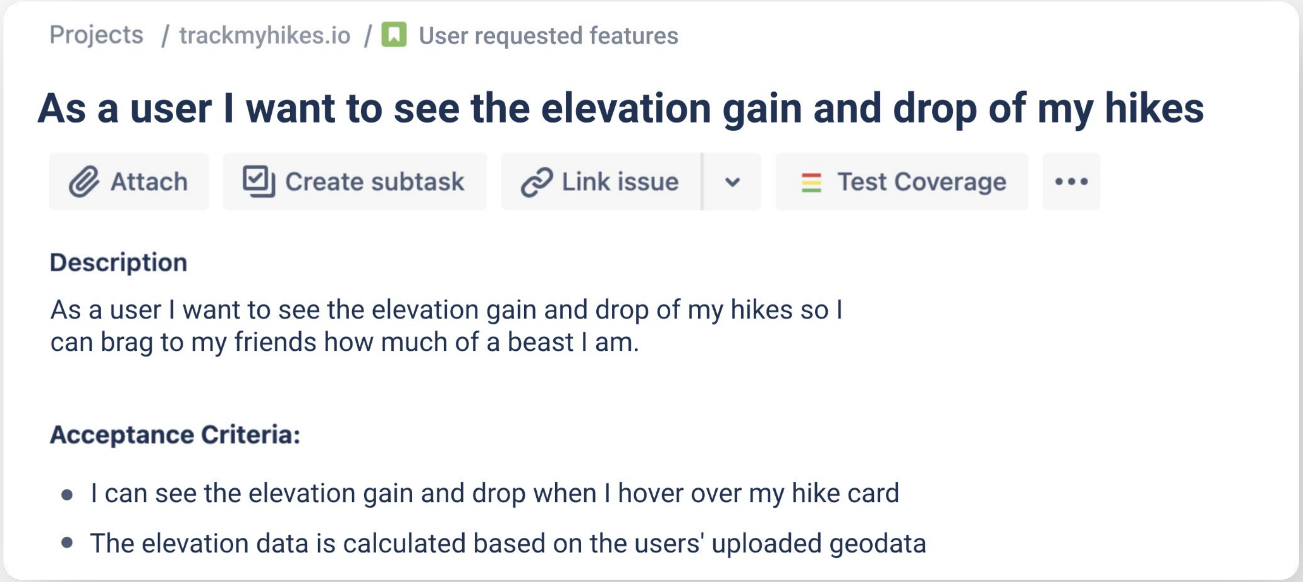
- Biggest productivity gain is if your team owns **both** BE & FE
- Having healthy code review process helps **a lot**
- It’s **not** a silver bullet, challenges like backwards compatibility is still a factor (albeit smaller)

## **Demo** - trackmyhikes.io

- Imaginary app for displaying completed hikes
- Existing OpenAPI spec integration
- BE in **Kotlin** / spring
- FE in **next** / app router
- 💭🥾 Domain is available!



## **Demo** Let's build a **feature** with OpenAPI



Projects / trackmyhikes.io / User requested features

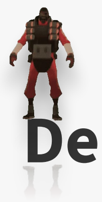
**As a user I want to see the elevation gain and drop of my hikes**

**Description**

As a user I want to see the elevation gain and drop of my hikes so I can brag to my friends how much of a beast I am.

**Acceptance Criteria:**

- I can see the elevation gain and drop when I hover over my hike card
- The elevation data is calculated based on the users' uploaded geodata



## Demo time...

## That’s cool. What’s next?

### Take it further with

- Load spec with **wiremock** to simulate API for E2E tests
- Load spec in **Postman** to make API requests easily
- Use spec to perform **contract testing** to validate your API
- Upload your spec to **backstage** and share it with your org
- Generate and publish client SDK on (private) **NPM** to share with your consumers

## Thank you, Amsterdam!

Github repository
with code + slides

Feel free to connect
via my socials!

patrickhuijten.dev

## Time for questions, maybe? ⏱️