Docs first

from a client's perspective
if a feature is not documented, then it doesn't exist,
if a feature is documented incorrectly, then it's broken

Overview

- why good documentation matters more for apis and microservices

- how I've benefited from a docs first approach

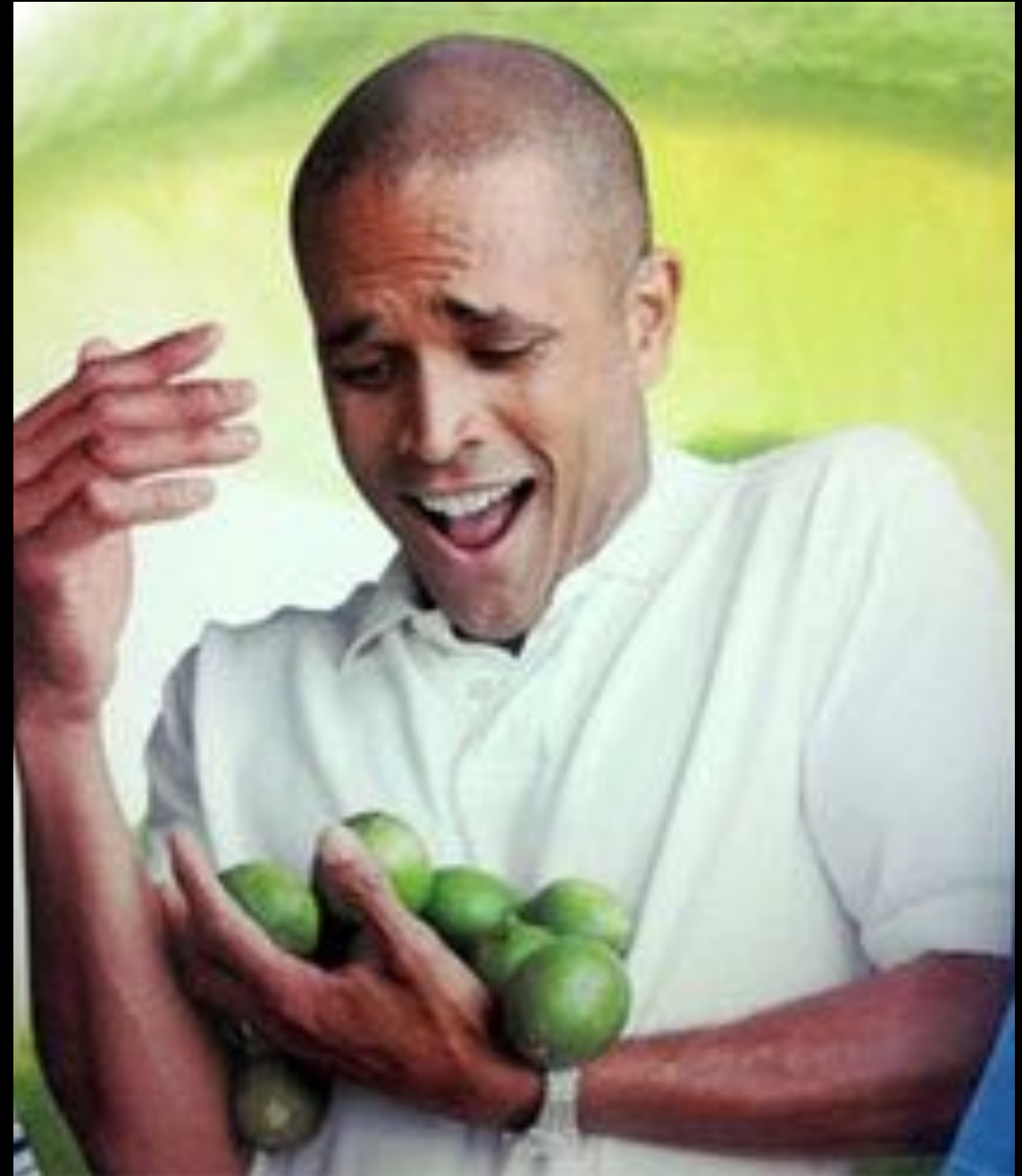- documentation technology (openapi/swagger)

Disclaimer: docs are boring

actual footage of a developer working on documentation

All these clients

- user client

- admin client

- other micro services in ecosystem

- 3rd party clients

When clients read the docs

- point out missing functionality

- complain about the number of requests

- complain about the structure of data

- show ambiguities

- "clarify" the "original" requirements

timeline example

Documentation benefits

- Feedback before committing to a solution

- Thinking the whole project through

- Replace early specifications

Machine readable documentation benefits

- OpenAPI (Swagger) ecosystem

- Spec validation

- UI rendering

- Automated testing

- Code generation

Ui rendering

How it went

- larger OpenApi docs got hard to debug

- spec doesn't cover everything we need right now

- IDE support is lacking

- used machine readable docs with getting a machine to read it

Next steps

- cli watch filesystem and validate docs

- json refs, allOf, oneOf, anyOf

- api scaffolding from spec (swagger -> laravel)

- help build OpenApi ecosystem