From Manual to Velocity: The Spec-Driven Way 🚀

Our Orval-powered journey across BFF and Frontend layout: center class: text-center

The Setup 🧩

NestJS BFF to call **internal + external APIs** via **WSO2**

- 4 APIs (2 external, 2 internal)

layout: center

What we needed 🧰

- **Services** with WSO2 auth
- **DTOs** for controllers

layout: statement class: text-4xl font-black leading-tight

From “manual DTOs” 😵‍💫
to “generated everything” ✨

layout: two-cols-header

The Search 🔎

Left: past experience with “API clients”
Right: can we do it from a Swagger/OpenAPI spec?

layout: center class: bg-gradient-to-tr from-amber-500 to-rose-500 text-white

Why Orval felt like magic 🪄

- Config in, **clients out**
- **Zod** schemas
- **React/Svelte/Vue Query**
- **Mocks** (examples + Faker)
- Full **types** (body/params/query/response)

layout: image-right image: https://source.unsplash.com/collection/190727/1600×900

The Orval config 🧪

Point to input spec, choose outputs.

```
// orval.config.ts (gist)
export default {
  bff: {
    input: './specs/bff.yaml',
    output: {
      client: 'react-query',
      mock: true,
      override: {
        zod: true,
      },
    },
  },
}
```

layout: center

The twist: WSO2 auth 🔐

We needed to attach tokens on every call.

layout: two-cols

layout: center

Services = Generated handlers ✅

Use Orval handlers as **NestJS services** in modules.

layout: center

DTOs without tears 😌

Use **nestjs-zod** to generate DTOs from Zod.

- Validates body/params/query
- Generates Swagger for WSO2

layout: two-cols

layout: center class: bg-gradient-to-br from-emerald-500 to-cyan-600 text-white

Testing at lightspeed ⚡

Use **generated mock handlers** for services.

- Fast happy-path tests
- Consistent data via examples/Faker

layout: center

Impact ⏱️

Days of manual work → **seconds** of generation

- Fewer bugs
- Consistent types
- Same source of truth: the spec

layout: center

Specs must be standard 📏

- Some specs were “valid” but not **standard**
- We pushed for standardization (met some resistance 😅)

layout: center class: bg-gradient-to-tr from-indigo-600 to-fuchsia-600 text-white

Frontend: we kept going 🧭

Consume BFF Swagger, generate:

- **React Query** hooks
- **Types** and **Zod** schemas
- **Mocks** + handlers

layout: two-cols-header

Frontend usage 🎯

layout: center

The spec-driven loop ♻️

Update spec → regenerate → app stays in sync layout: center class: text-5xl font-black

Questions? ❓

Thank you! 🎉

Presenter notes cheatsheet 📝

- BFF via WSO2, 4 APIs
- Services + DTOs
- Orval choice (over Kubb)
- Mutators (internal/external)
- nestjs-zod + query workaround
- Mocks = fast tests
- Standard specs matter
- Frontend hooks/schemas/mocks
- Spec-driven loop