

# Fullstack Development

# Form and validation

Is your app safe?

# Let's make sure your app is safe from

- Data from API call
- User input

# Wait, but I thought I used TypeScript.

- TypeScript catches compile-time errors.
- We are talking about **run-time** errors.

# Validation library

- Zod
- Yup
- Typebox

# Task 1: Validating API

# User server

## Local

- Clone <https://github.com/fullstack-66/user-server>
- `npm install`, `npm start`
- `http://localhost:3001`

## Cloud

- <https://user-server-production-f0b7.up.railway.app>

# Task

- Inspect `User` data from `/users` and `/users_wrong`. Notice
  - Key differences
  - Calendar year difference
- Let's make sure your app can detect and handle these differences at run-time.
- Solution
  - <https://github.com/fullstack-66/form>



# Setting up NextJS

./tsconfig.json

```
{
  // ...
  "baseUrl": ".",
  "paths": {
    "@components/*": ["components/*"],
    "@app/*": ["app/*"]
  }
}
```

# Remove Tailwind preflight

```
./app/globals.css
```

```
/* @tailwind base; */  
@tailwind components;  
@tailwind utilities;
```

# API URL

./utils/index.ts

```
if (!process.env.NEXT_PUBLIC_API_URL) {  
  throw new Error("NEXT_PUBLIC_API_URL is not set");  
}  
const API_URL = process.env.NEXT_PUBLIC_API_URL;  
export const URL_DATA = `${API_URL}/users`;  
export const URL_DATA_WRONG = `${API_URL}/users_wrong`;
```

# Install

- `npm install zod zustand axios`

# Task 2: Validating user input

Using vanilla React

# Endpoints

## POST /users

- Create user
  - I intentionally throttled the server.

```
{  
  "firstName": "Vince",  
  "lastName": "Romaguera",  
  "email": "Albert_Von@gmail.com",  
  "dateOfBirth": "2022-8-1",  
  "password": "12345",  
  "confirmPassword": "12345"  
}
```

# Endpoints

**GET** `/reset`

- Reset data

# Install

- `npm install react-modal`
- `npm install -D @faker-js/faker`



# Form UX improvement

- "Real-time" validation
- Prevent submission if input is not valid.
- Prevent double submission.
- Prevent typing during submission.
- Auto-focus the wrong input.

# Real-time validation

- Use `useEffect` to trigger schema validation
- Store errors in `errors` state.
- Keep track of when user touches the form.
  - Prevent premature validation.
  - Store `touch` state

# Form disable

- Keep track of `valid` state.
- Keep track of `submission` state.

# Spiral out of control

- Too many states
- Too many logics
- Not reusable

# Form library

- Help you handle form states and logics in a reusable manner.
  - It is essentially a custom hook.
  - Integrates seamlessly with validation library.
- Popular libraries
  - Formik
  - React Hook Form

# Install

- `npm install react-hook-form @hookform/resolvers @hookform/error-message`