

# Developing TypeScript APIs with confidence using tRPC

Boris Martinović

**Hi, I'm  
Boris.**

- Software Engineer  
@ Euroherc Insurance
- Lecturer @ Algebra
- Organizer in GDG  
Zagreb chapter

# Some questions for today

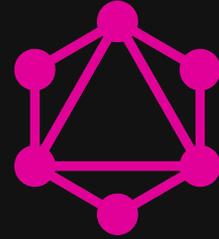
**What** even is tRPC and why do we want to use it?

**Where** to use it?

**How** to use it and improve our dev experience?



**What options do we  
have now?**



GraphQL

{ REST }

**What** even is tRPC  
and why do we want  
to use it?



gRPC



So now that we have  
established what  
tRPC **isn't**...

## ...let's see what it is (and has)

- TS Library that lets us build fully typesafe API without need for schemas or code generation
- Allows for sharing types between client and server (or server to server)

## ...let's see what it is (and has)

- A protocol to expose your backend code to your frontend
- Stripping down REST
- GET for queries (easy caching)
- POST for mutations

# Why would I even use this thing?

- Allows full E2E type safety
- Catching bugs as soon as possible (compile and build time)
- If you are using TypeScript everywhere already, it's easy to plug in
- Autocompletion

**Where to use it?**

# Where to use it?

- Any new or existing full-stack TypeScript project (e.g. Next.js or Node.js)
- In a monorepo or single project (type definitions come from the TS code itself)
- If you want faster and safer development

## Where **not** to use it?

- Projects with many microservices
- Projects that require some other language for the backend (C#, Go, Java...)
- Don't use it to rewrite existing TS codebase - instead add to them

**How** to use it to improve  
our dev experience?

# For starters...

- Controller ---- Router
- Endpoint --- Procedure
- Uses Zod for validation
- React Query for data fetching

# The Example

- T3 stack - tRPC pre-configured with example
- `npm create t3-app@latest`
- `kusur.martinovic.dev`

# The Example

## Povrat € nakon plaćanja u kn

Unesite iznos artikla u €

100,00

Iznos kojeg je kupac dao u €

90,00

Iznos kojeg je kupac dao u kn

100,00

Preostalo za vratiti kupcu: 3,27 €, tj. 24,65 kn

# Defining our router



```
export const appRouter = createTRPCRouter({  
  calculation: calculationRouter,  
});
```

# Defining our procedure

```
const rate = 7.5345;
//api/trpc/calculateRest
export const calculationRouter = createTRPCRouter({
  calculateRest: publicProcedure
    .input(
      z.object({
        priceEur: z.number().nonnegative(),
        givenEur: z.number().nonnegative(),
        givenKn: z.number().nonnegative(),
      })
    )
    .query(({ input }) => {
      //calculating sum and formatting
      return {
        resultText:
          returnSum <= 0
            ? `Preostalo za vratiti kupcu: ${returnSumEur} €, tj. ${returnSumKn} kn`
            : `Kupac duguje: ${returnSumEur} €, tj. ${returnSumKn} kn`,
      };
    }),
});
```

# Calling our procedure



```
const calcRest = api.calculation.calculateRest.useQuery({  
  priceEur: priceEur ?? 0,  
  givenEur: givenEur ?? 0,  
  givenKn: givenKn ?? 0,  
});
```

**Ok... but where's the  
safety part?**

# Autocomplete

```
api.calculation.|
```



```
api.calculation.
```



```
calculateRest (property) calculateRest: { getQueryKey: (input...
```



```
getQueryKey
```

# Wrong property name?

Argument of type '{ price: number; givenEur: number; givenKn: number; }' is not assignable to parameter of type '{ priceEur: number; givenEur: number; givenKn: number; }'.

**Translation:** I was expecting { priceEur: number; givenEur: number; givenKn: number; }, but you passed { price: number; givenEur: number; givenKn: number; }

[See full translation](#)

Object literal may only specify known properties, and 'price' does not exist in type '{ priceEur: number; givenEur: number; givenKn: number; }'.

**Translation:** You can't pass property price to object { priceEur: number; givenEur: number; givenKn: number; }.

[See full translation](#)

(property) price: number

Argument of type '{ price: number; givenEur: number; givenKn: number; }' is not assignable to parameter of type '{ priceEur: number; givenEur: number; givenKn: number; }'.

```
price: priceEur ?? 0,  
givenEur: givenEur ?? 0,  
givenKn: givenKn ?? 0,  
});
```

# Wrong procedure name?

```
Property 'calculateRests' does not exist on type '{ getQueryKey: () => QueryKey; } & DecoratedProcedureRecord<{ calculateRest: BuildProcedure<"query", { _config: RootConfig<{ ctx: {}; meta: object; errorShape: DefaultErrorShape; transformer: { stringify: (object: any) => string; ... 6 more ...; allowErrorProps: (...props: string[]) => void; }; }>; ... 5 more ...; ...'. Did you mean 'calculateRest'?
```

**Translation:** You're trying to access `calculateRests` on an object that doesn't contain it. Did you mean `calculateRest` ?

[See full translation](#)

any

```
Property 'calculateRests' does not exist on type '{ getQueryKey: () => QueryKey; } & DecoratedProcedureRecord<{ calculateRest: BuildProcedure<"query", { _config: RootConfig<{ ctx: {}; meta: object; errorShape: DefaultErrorShape; transformer: { stringify: (object: any) => string; ... 6 more ...; allowErrorProps: (...props: string[]) => void; }; }>; ... 5 more ...; ...'. Did you mean 'calculateRest'? ts(2551)
```

[View Problem \(Alt+F8\)](#) [Quick Fix... \(Ctrl+.\)](#)

You, now • Uncommitted changes

```
import { api } from "../utils/api";
import { InputNumber } from "primereact";
import { useState } from "react";
import { Button } from "primereact";
```

```
const Home: NextPage = () => {
  const [priceEur, setPriceEur] = useState(0);
  const [givenEur, setGivenEur] = useState(0);
  const [givenKn, setGivenKn] = useState(0);
```

```
const calcRest = api.calculation.calculateRests.useQuery({
  priceEur: priceEur ?? 0,
  givenEur: givenEur ?? 0,
  givenKn: givenKn ?? 0,
});
```

# Added a new property? No problem!

```
export const calculationRouter = createTRPCRouter({
  calculateRest: publicProcedure
    .input(
      z.object({
        priceEur: z.number().nonnegative(),
        givenEur: z.number().nonnegative(),
        givenKn: z.number().nonnegative(),
        note: z.string(),
      })
    )
})
```

Argument of type '{ priceEur: number; givenEur: number; givenKn: number; }' is not assignable to parameter of type '{ priceEur: number; givenEur: number; givenKn: number; note: string; }'.

**Translation:** I was expecting { priceEur: number; givenEur: number; givenKn: number; note: string; }, but you passed { priceEur: number; givenEur: number; givenKn: number; }

[See full translation](#)

Property 'note' is missing in type '{ priceEur: number; givenEur: number; givenKn: number; }' but required in type '{ priceEur: number; givenEur: number; givenKn: number; note: string; }'.

**Translation:** You haven't passed all the required properties to { priceEur: number; givenEur: number; givenKn: number; note: string; } - { priceEur: number; givenEur: number; givenKn: number; } is missing the note property

[See full translation](#)

```
const givenKn: number | null
```

```
  givenKn: givenKn ?? 0
```

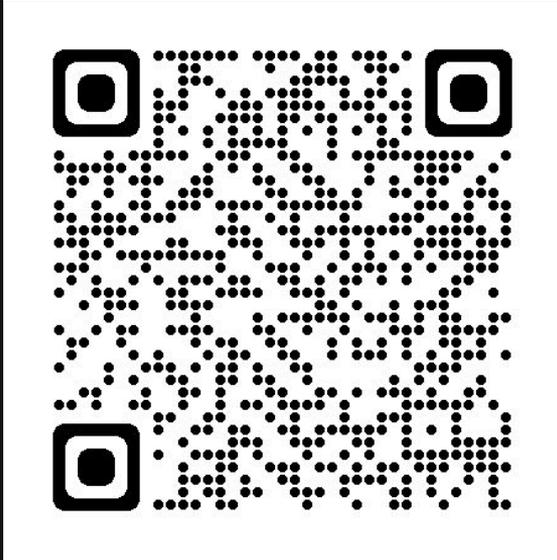
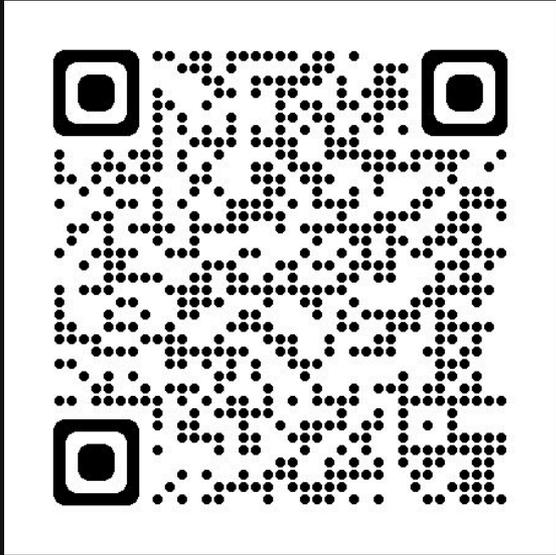
You, last month • Index + calculation ...

```
});
```

## Let's sum it up

- tRPC is best learned by doing
- Excellent tool for fast-moving individuals and teams
- Full autocompletion, type safety and ease of use make it a great tool

**Thank you!**



The example

