

- RestApi GraphQL
- GraphQL
- Hasura
- WebApi Midway/TSRPC
- CQRS
- Vue3 TSPC,Hasura
- :
-

# Restful API

2000

resetful

10

WEB API

API

DNA

```
//  
GET http://www.store.com/products
```

```
//      id?id=12345  
GET http://www.store.com/products/12345
```

```
//  
POST http://www.store.com/orders
```

```
//      id?id=12345  
DELETE http://www.store.com/products/12345
```

# Restful

restful



# Restful

GET http://www.store.com/user/1

user “ ”

```
{
  "nickname": "      ",
  "age": 88,
  "avatar": "      .png",
  "dream": "      ... ",
  "sign": "      ",
  "phone": "      ",
  "homepage": "www.yinzhuoei.com"
}
```

nickname age avatar

“ API”

, Restful API 2 :

1.

2.

N+1

API

API



ui  $\longrightarrow$  front end  $\longrightarrow$  back end

Before

ui  $\longrightarrow$  front end

After



Graphql

API

...

```
query {  
  User(){  
    nickname  
    age  
    avatar,  
    orders{  
      title,  
      time  
    }  
  }  
}
```

```
{  
  "nickname": "      ",  
  "age": 88,  
  "avatar": "      .png",  
  "order": [  
    {  
      "title": "      ",  
      "time": "2021-1-1"  
    }  
  ]  
}
```



# GraphQL

restapi

graphql

gql

restapi

GET http://www.store.com/products

vs

```
query {  
  Product(){  
    title,  
    time  
  }  
}
```

gql

gql

“ ”

gql

```
query {  
  user(id: 10086) {  
    nickname  
    history {  
      title  
    }  
    wallt {  
      amount  
    }  
    collect {  
      title  
    }  
    message {  
      user_id  
    }  
    follow {  
      user_id  
    }  
    likes {  
      user_id  
    }  
  }  
}
```

# GraphQL

graphql

restful

type    resolvers

,

Type

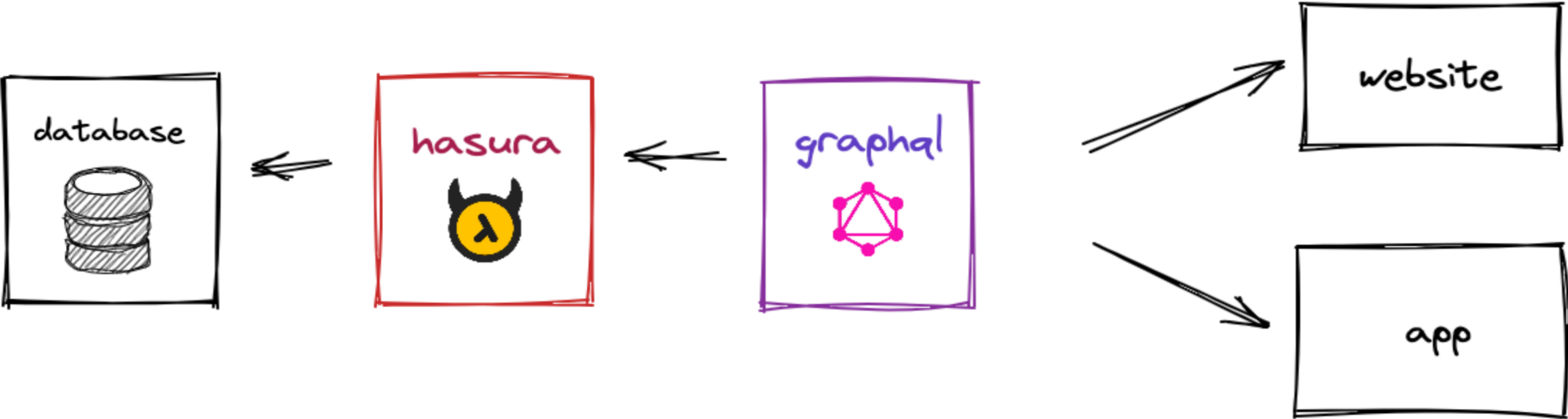
graphql +

0

Hasura

Graphql

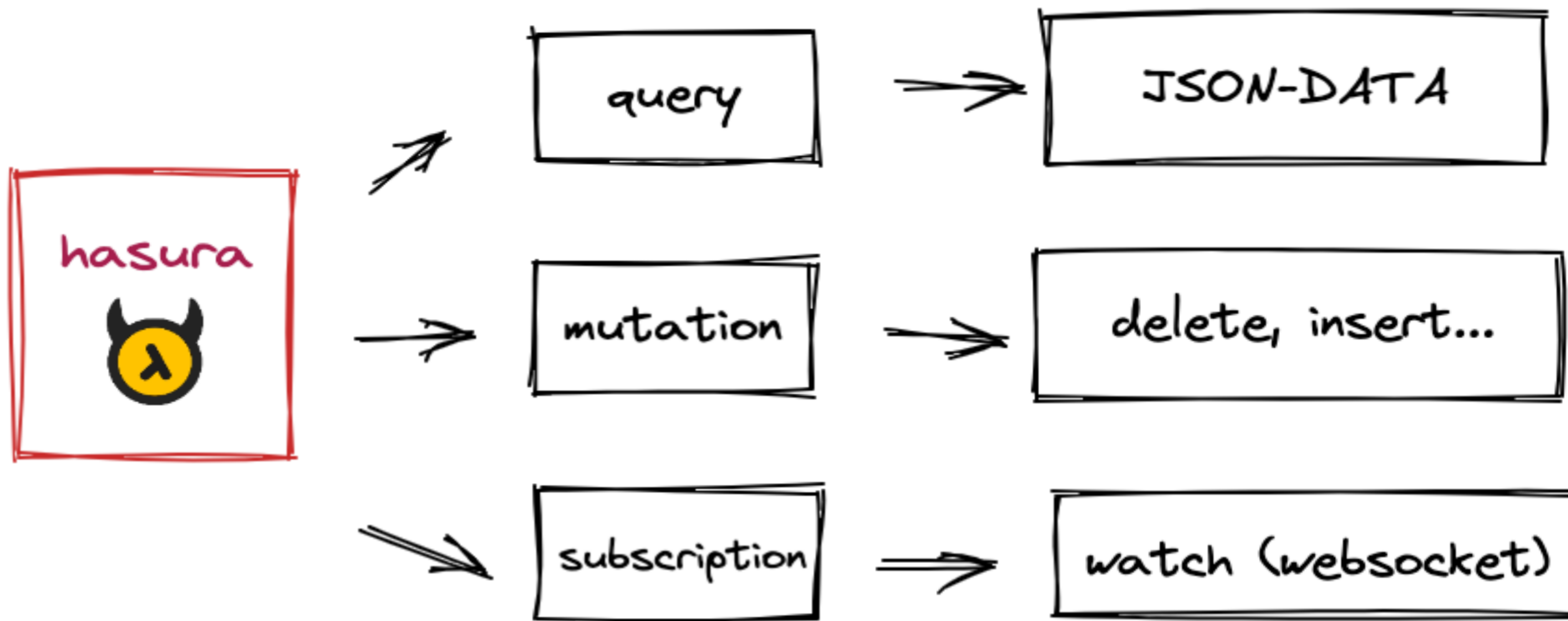
:



# hasura:

: postgres

# hasura



hasura “ ”

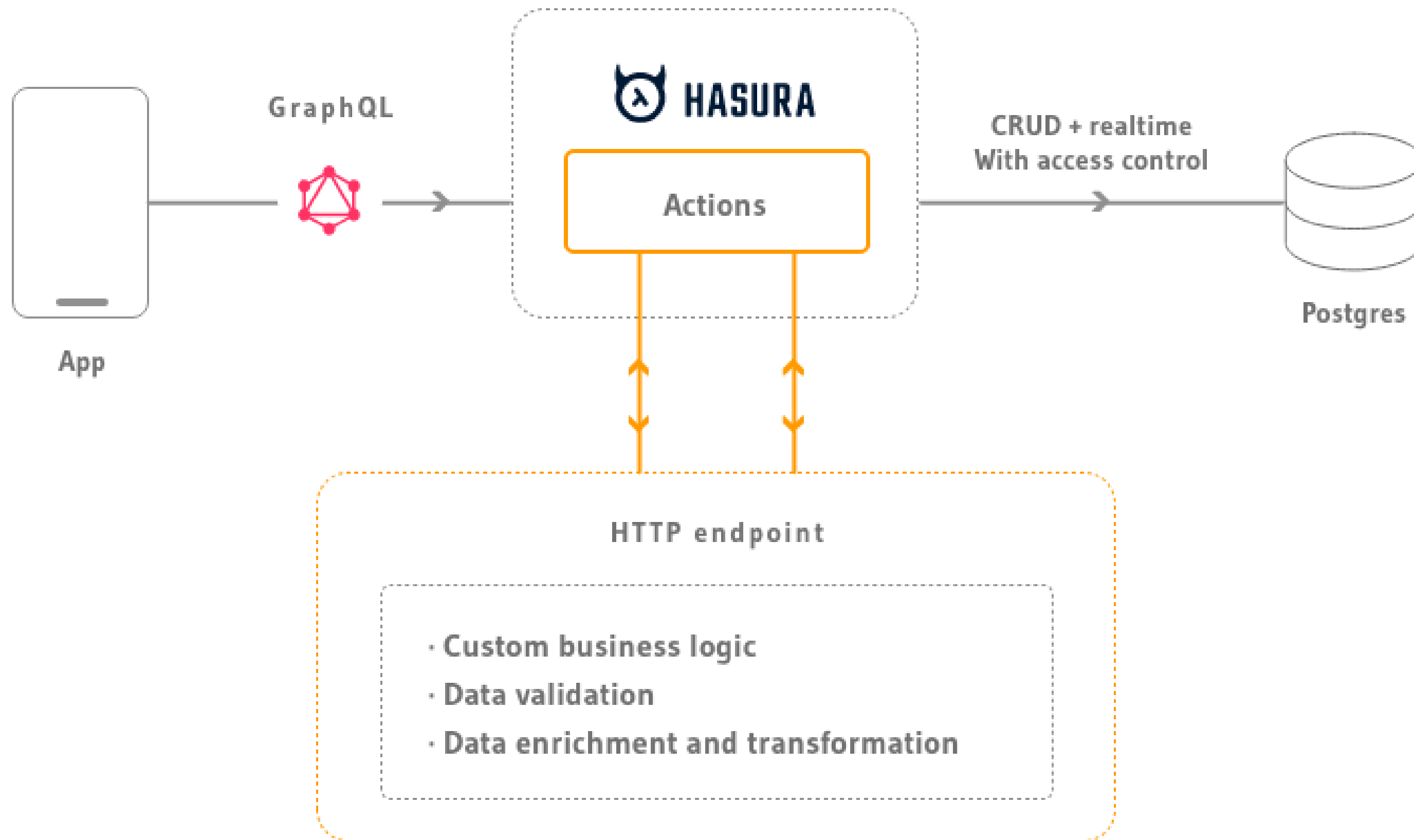
hasura mutation query mutation :

```
mutation MyMutation {
  insert_products(objects: {desc: " ", price: "12", title: " ", push_user_id: 1}) {
    returning {
      id
    }
  }
}
```

[ Action

“ (hasura query)” “ (hasura action)”

hasura action action / /



action

serverless

restapi

<https://hasura.io/docs/latest/graphql/core/actions/index.html>

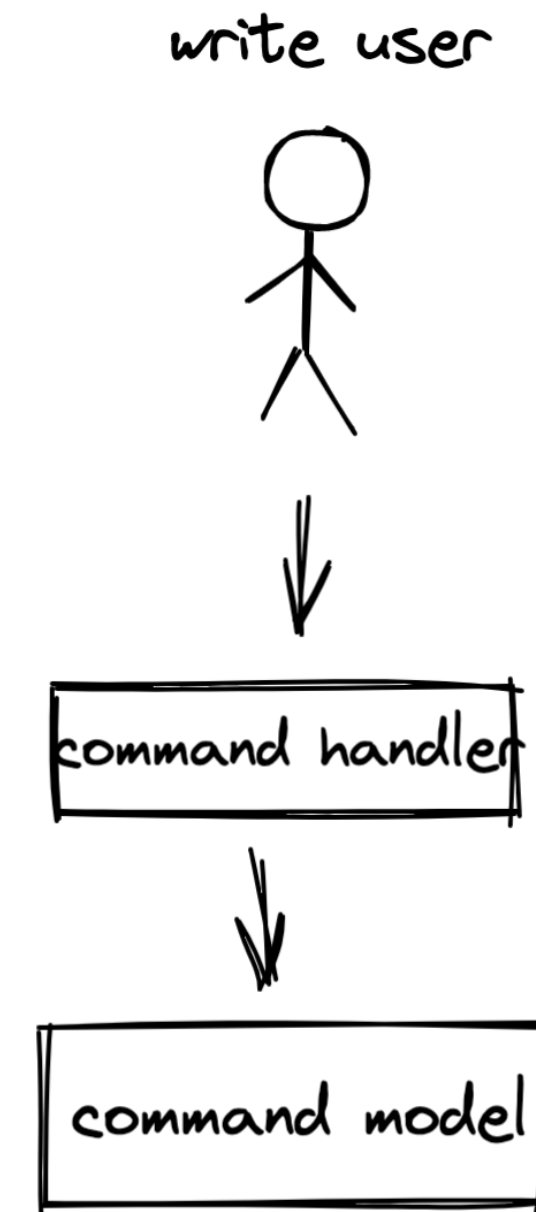
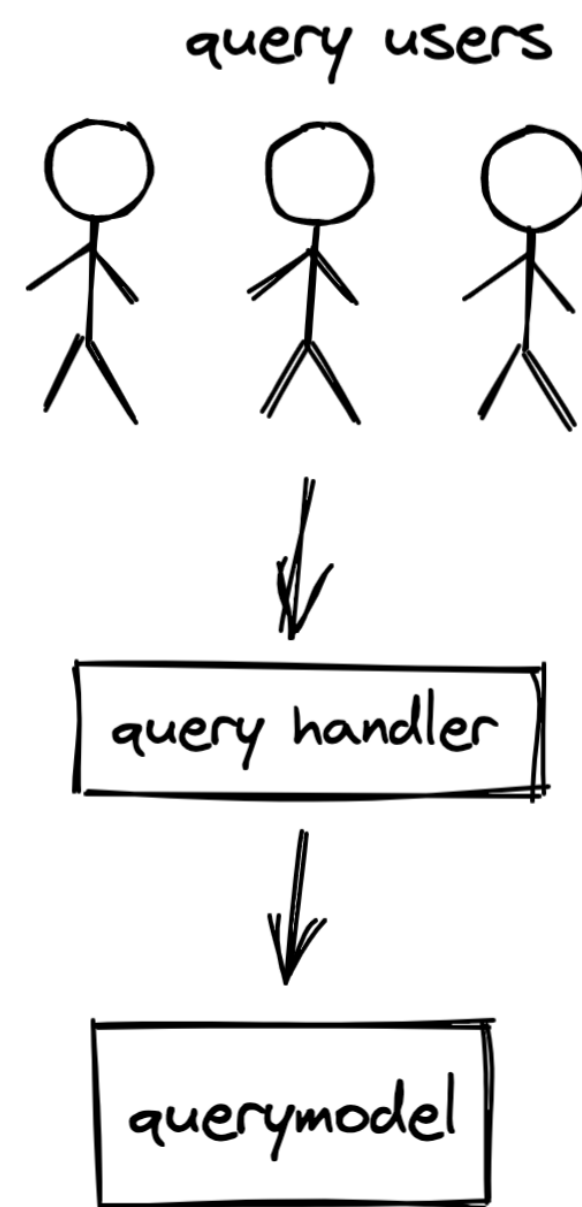


# CQRS

CQRS “ ” Command Query Responsibility Segregation

CQRS

CQRS



# CQRS

CQRS

## Query Model

# Command Model

Redis 😊

/ /

# Command Model

## Query Model

CQRS

2

1.

2

CQRS

2.

2

/

/

Typescript

midway.js, uniapp (unicloud) tsrpc...

2

:

```
// server.js
const main = () => {
  return ["1", "2", "3"]
}
```

```
// app.vue
//
<search-db >

</search-db>
```

10

JS

:

1. TypeScript

2.

3.

# TSRPC

TSRPC 🐶

:

- 1.
- 2. API
- 3. API Action Hasura
- 4. Hasura+TSRPC

ppt	demo
-----	------

# API Protocols

TSRPC protocols

2

API

```
export interface ReqAddProduct {  
  /** */  
  title: string;  
  /** */  
  desc: string;  
  /** */  
  price: string;  
  /** */  
  pushUserId: number;  
}
```

```
export interface ResAddProduct {  
  /** */  
  id?: number  
}
```

# API

ORM    TypeOrm

```
import { ApiCall } from "tsrpc";
import {
  ReqAddProduct,
  ResAddProduct,
} from "../shared/protocols/PtlAddProduct";
import { Products } from "../entities/Products";
import { getRepository } from "typeorm";

export async function ApiAddProduct(
  call: ApiCall<ReqAddProduct, ResAddProduct>
) {
  const res = await getRepository(Products).save(call.req);
  call.succ({
    id: res.id
  })
}
```



TS    TSRPC

“

”

# TS

JS                      TS                      TS                      TS

:

- 1. JSON SCHEMA
- 2. /

TS Interface

TSRPC



```
interface Colorful {
  color: string;
}
interface Circle {
  radius: number;
}
export interface Info {
  title: "h5" | " " | " ";
  desc: Record<string, string>;
  price: string | number;
  other: Colorful & Circle
}
```

# API

hasura

CQRS

api

...

graphql hasura

API

hasura

Action API

API



Action hasura

hasura-action

hasura action



Type (Mutation / Query)



Definition



Handler (webhook ?)



support serverless





Action      hasura

