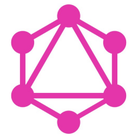


- RestApi GraphQL
- GraphQL
- Hasura
- WebApi Midway/TSRPC
- CQRS
- Vue3 TSRPC,Hasura
- :
-

Restful API

2000 resetful 10 WEB API

API DNA

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

```
//     id12345
GET http://www.store.com/products/12345
```

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

```
//     id12345
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"
}
```

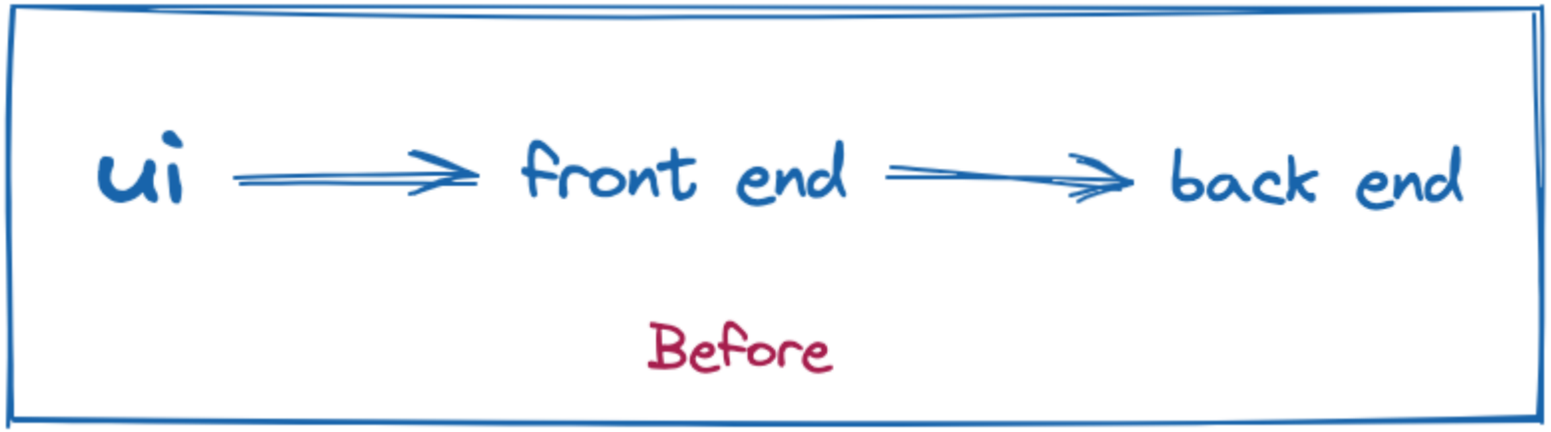
nickname age avatar

“ API”

, Restful API 2 :

1.
2. N+1 API API

😄


ui
front end
back end
Before


ui
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 restapi

gql

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

vs

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

gql gql

“ ”

gql

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


  }
```

Graphql

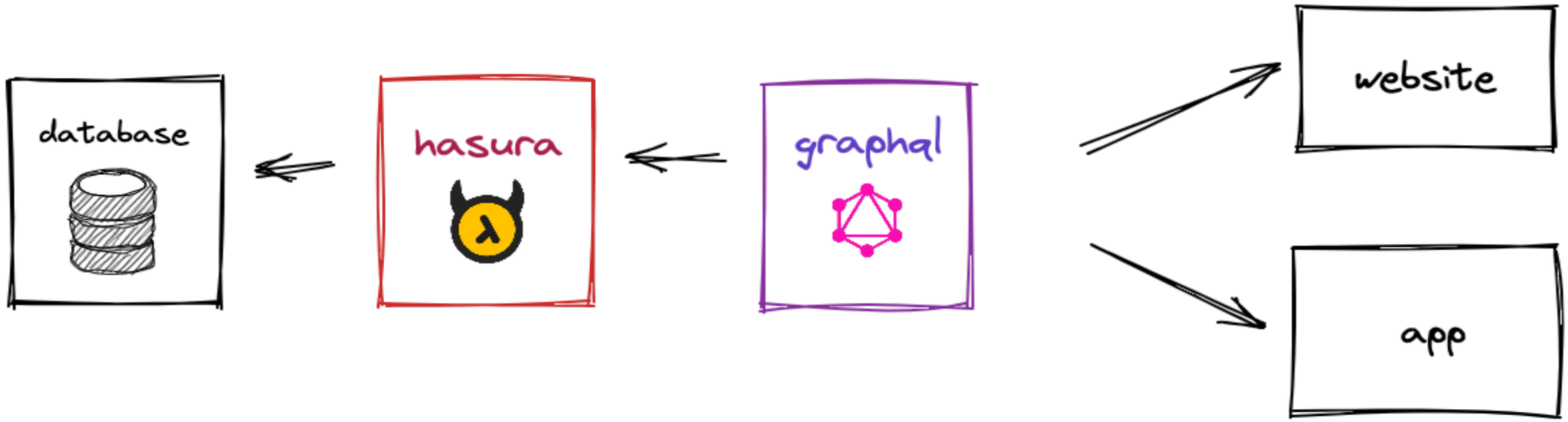
graphql restful

type resolvers

, Type

graphql + 0

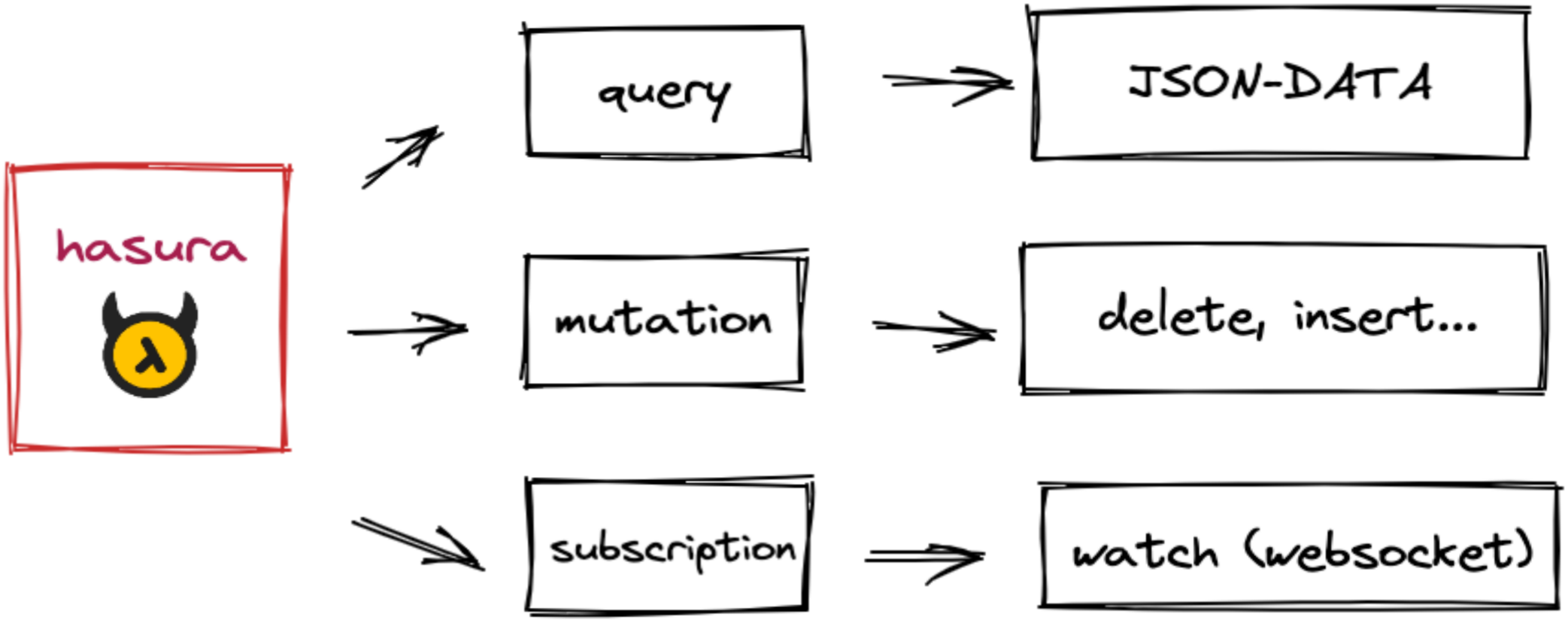
Hasura Graphql :



hasura:

: postgres

hasura


hasura
query
JSON-DATA
mutation
delete, insert...
subscription
watch (websocket)

hasura “ ”

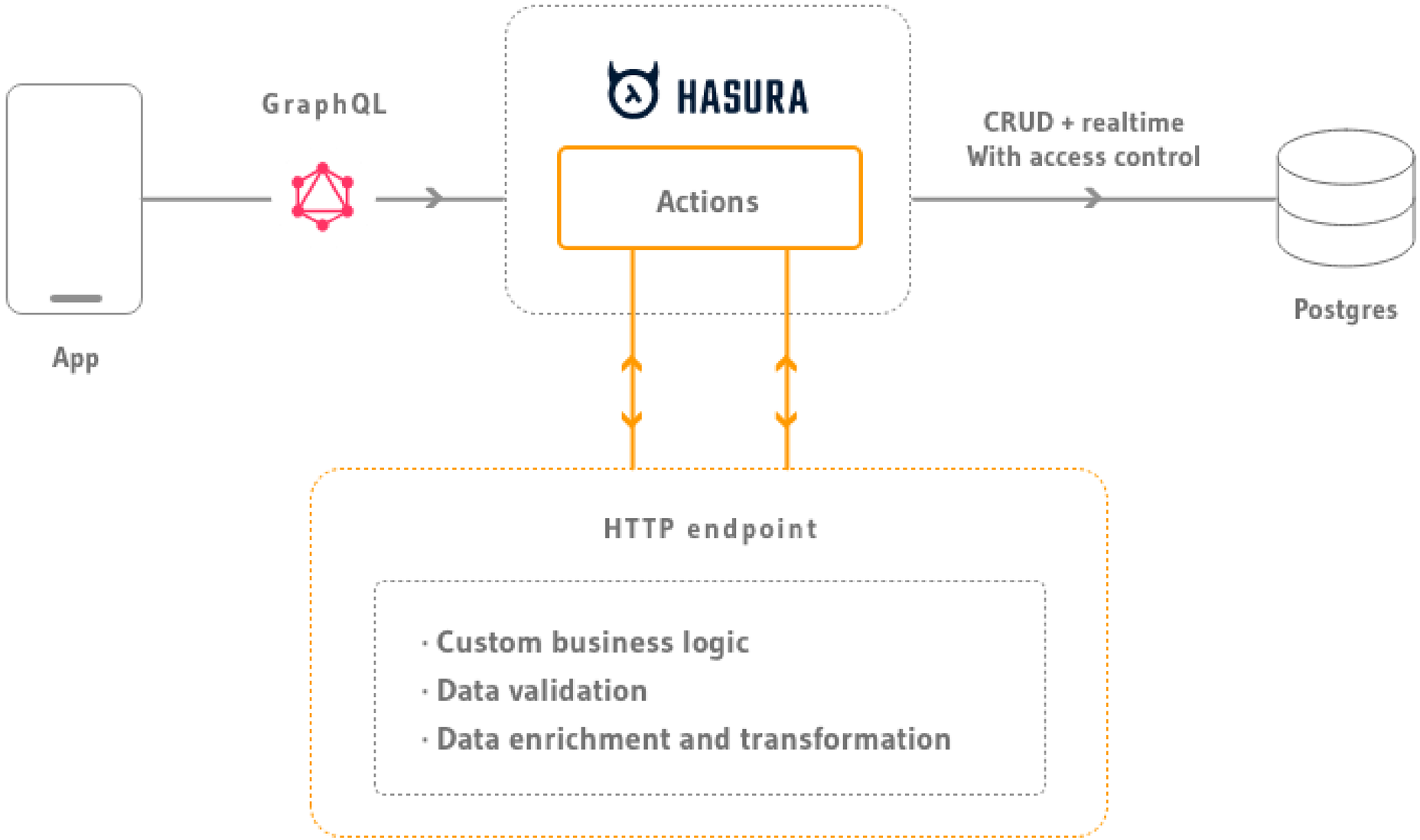
hasura mutation query mutation :

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

Action

“ (hasura query)” “ (hasura action)”

hasura action action / /



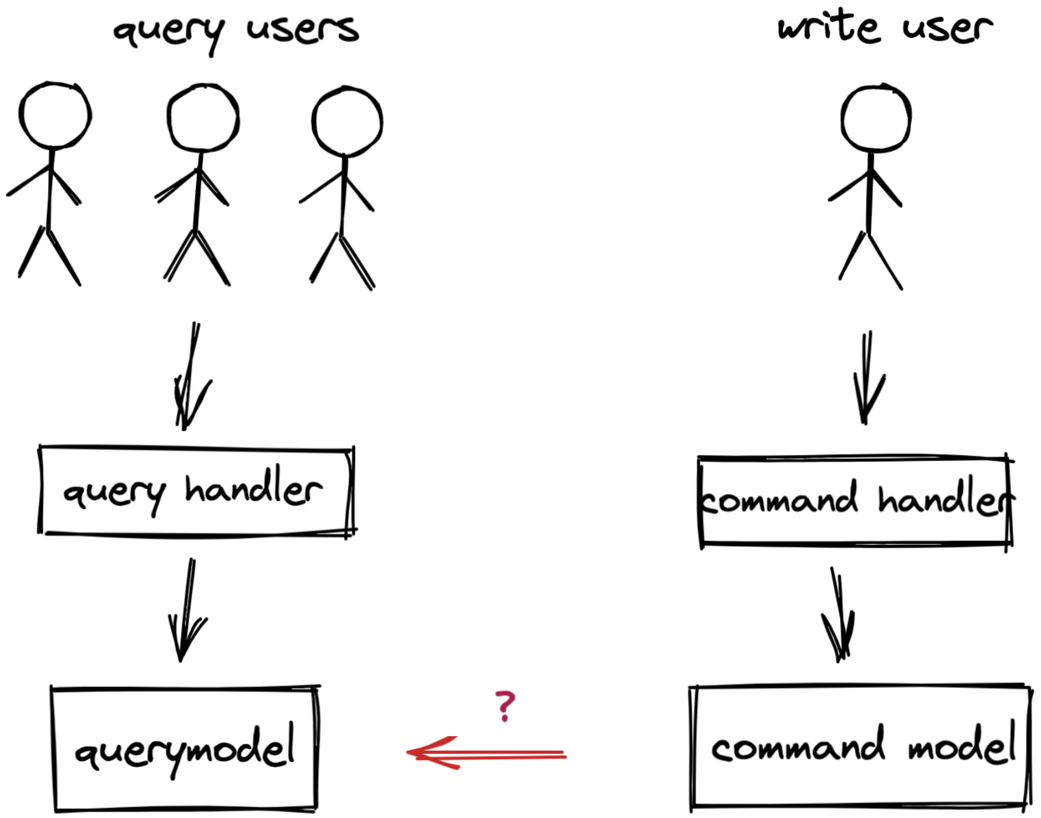
action serverless restapi

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 //
                                10
<search-db> </search-db>
```

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

```ts
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

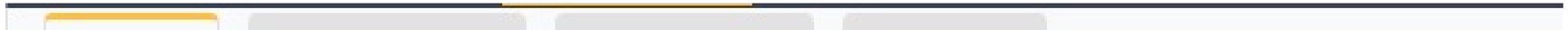
...

hasura Action API API

😣 Action hasura

hasura-action hasura action

Action hasura



VSCode gql (hasura gql)

gql

```
export const GET_PRODUCTS = gql`
  query getProducts(
    $limit: Int = 10
    $offset: Int
    $title: String
    $desc: String
    $minPrice: Int
  ) {
    products(
      where: {
        title: { _like: $title }
        desc: { _like: $desc }
        price: { _gte: $minPrice }
      }
      limit: $limit
      offset: $offset
      order_by: { created_at: desc }
    ) {
      id
      title
      desc
      price
      push_user {
```

```
//
const {
  result: product,
  loading: productLoading,
  fetchMore,
} = useQuery<
  {
    products: FormData[] | null;
    products_aggregate: { aggregate: { count: number } };
  },
  Partial<FormState>
>(GET_PRODUCTS, {
  limit: formState.value.limit,
  offset: formState.value.offset,
});

watch(product, () => {
  formData.value = product.value!.products;
  pagination.value.total = product.value!.products_aggregate.aggregate.count;
});

const getList = () => {
  const { title, minPrice, desc, limit } = formState.value;
  searchState.value = {
    title: title !== "" ? `%${title}%` : null,
```

gql mutation

mutation action api

```
export const ADD_PRODUCT = gql`
  mutation addProduct($params: AddProductInput!) {
    addProduct(params: $params) {
      id
    }
  }
`;
```

hook mutation

```
const {
  mutate: addProductMutate,
  loading: addProductMutateLoading,
  onDone: onAddProductDone,
} = useMutation<{ id: string }, { params: AddFormState }>(ADD_PRODUCT);
```

useMutation useQuery

```
function useMutation<TResult, TVariables>
```

API

```
//
const handleAdd = () => {
  addProductMutate({
    params: {
      ...addFormState.value,
      price: Number(addFormState.value.price),
    },
  });
};

//
onAddProductDone(() => {
  addFormRef.value.resetFields();
  pagination.value.current = 1;
  addVisible.value = false;
  getList();
});
```

1. graphql hasura

2. + + 😄 UI
   css

3. serverless ...

- Hasura
- graphql
- demo tsrpc
- CQRS