

- RestApi GraphQL
- GraphQL
- Hasura
- CQRS
- WEB Nuxt/Next/TSRPC
- Vue3 TSRPC,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 nodejs

# graphql java, go...

java demo: <https://www.graphql-java.com/tutorials/getting-started-with-spring-boot>

```
@Bean
public GraphQL graphql() {
    return graphql;
}

@PostConstruct
public void init() throws IOException {
    URL url = Resources.getResource("schema.graphqls");
    String sdl = Resources.toString(url, Charsets.UTF_8);
    GraphQLSchema graphqlSchema = buildSchema(sdl);
    this.graphql = GraphQL.newGraphQL(graphqlSchema).build();
}

@Autowired
GraphQLDataFetchers graphqlDataFetchers;

private GraphQLSchema buildSchema(String sdl) {
    TypeDefinitionRegistry typeRegistry = new SchemaParser().parse(sdl);
    RuntimeWiring runtimeWiring = buildWiring();
    SchemaGenerator schemaGenerator = new SchemaGenerator();
    return schemaGenerator.makeExecutableSchema(typeRegistry, runtimeWiring);
}
```

graphqlrestful

schema    resolvers

Schema

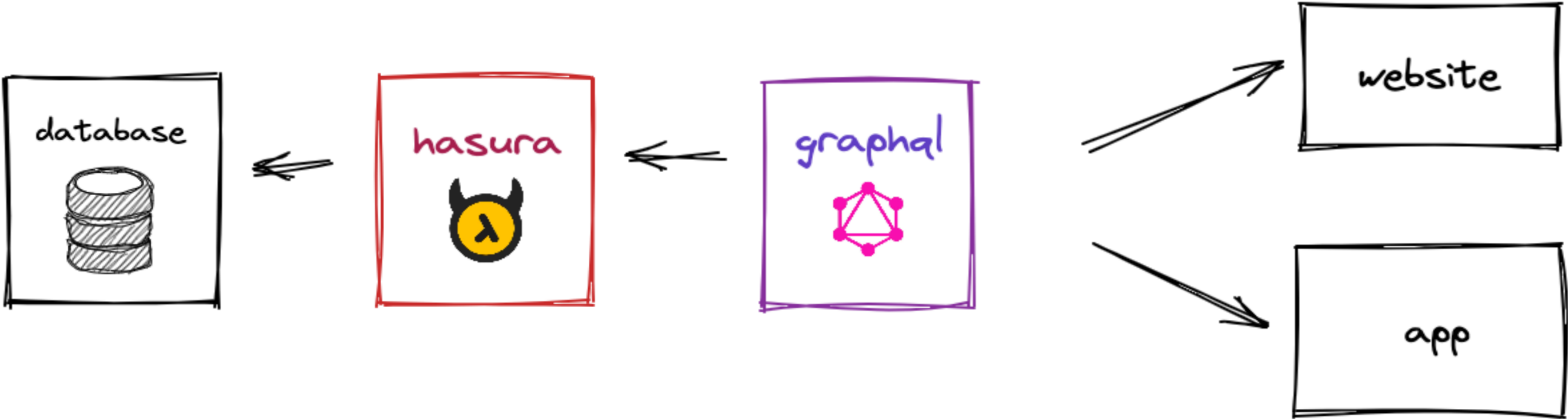
graphql +

0

Hasura

Graphql

:

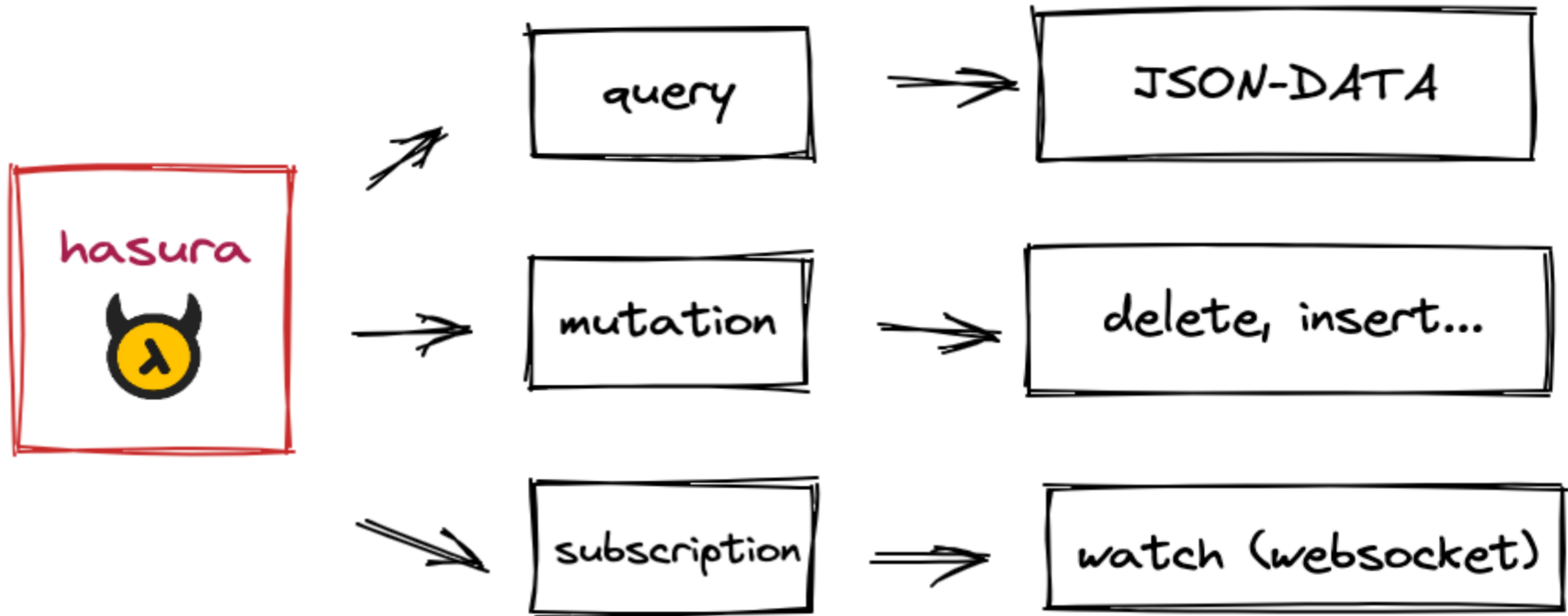


hasura:

: postgres



# hasura



hasura

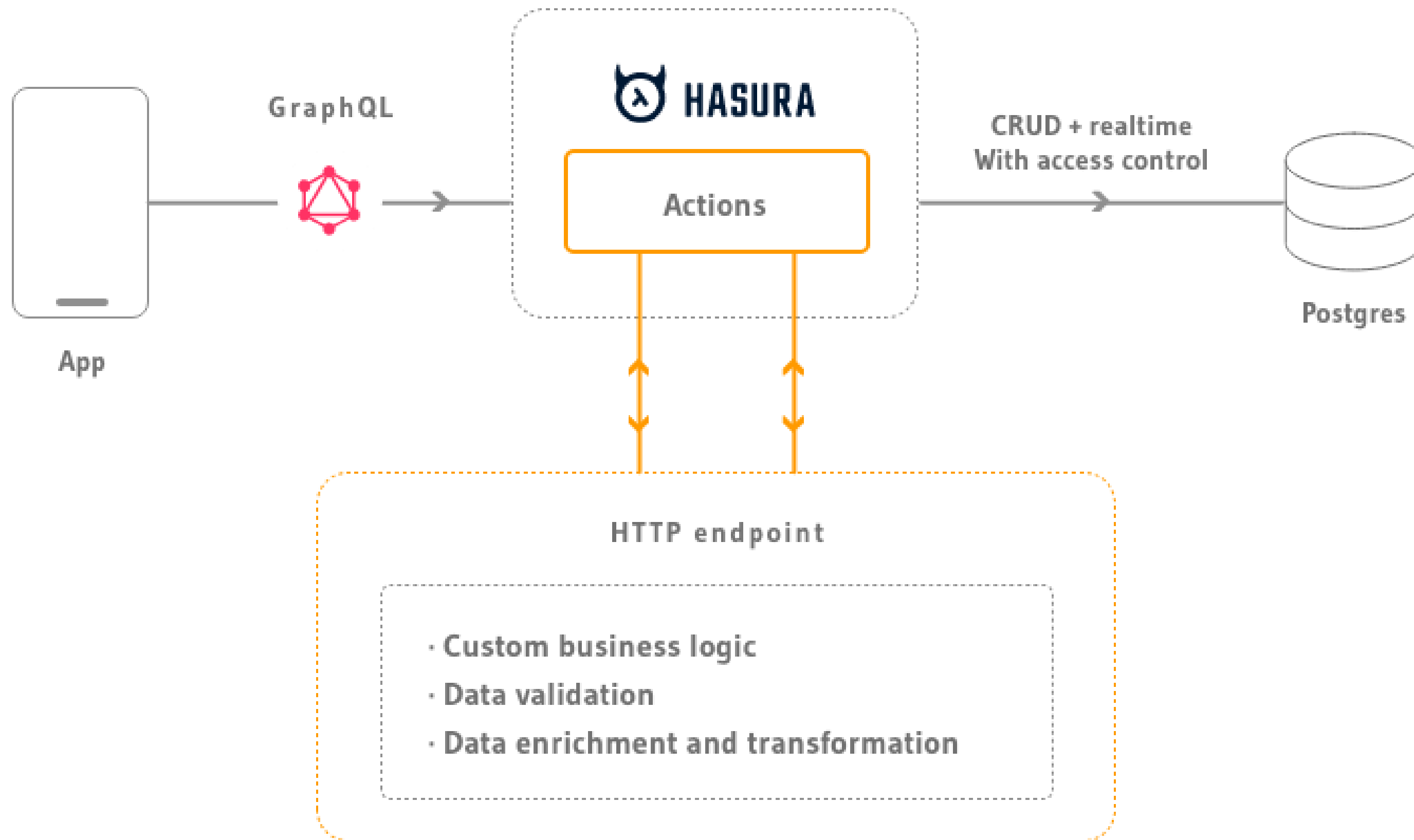
hasura   mutation   query

mutation :

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

hasura action / /





action

serverless

restapi

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

# Query & Action

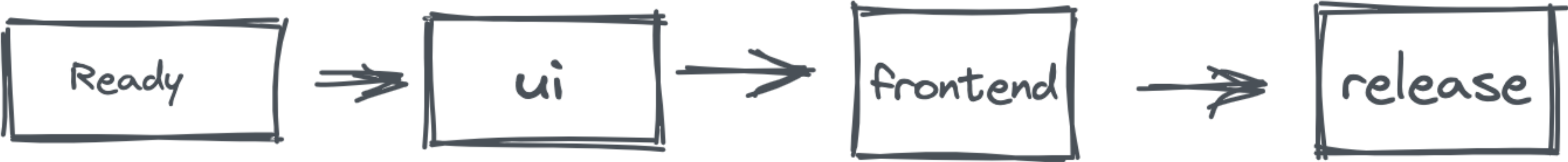
Hasura Action

hasura query

/ui

hasura

API

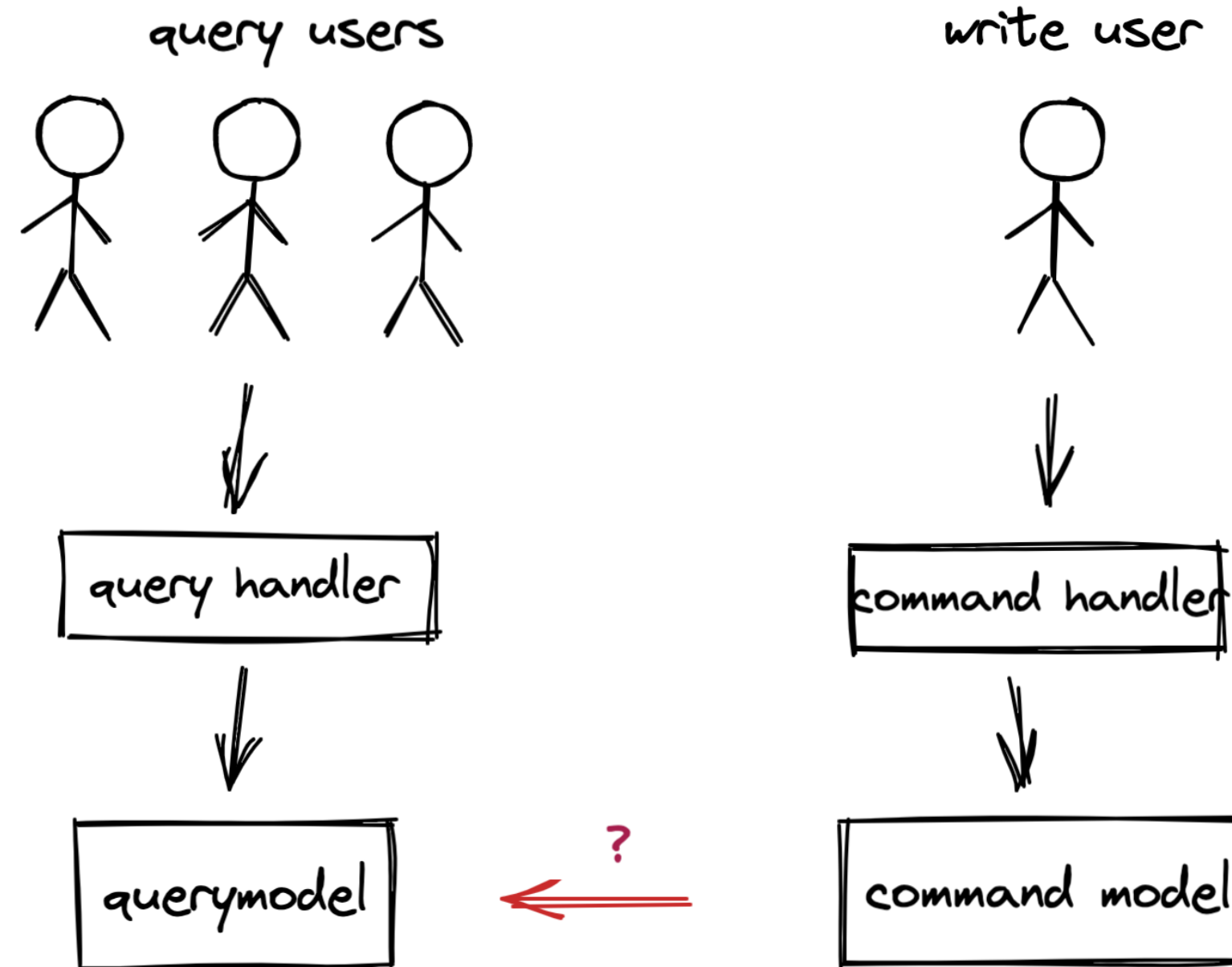


# CQRS

CQRS “ ” Command Query Responsibility Segregation

“ ”

CQRS



# CQRS

CQRS

## Query Model

# Command Model

# Redis 😊

# Command Model

## Query Model

CQRS

2

1.

2

CQRS

2.

2

/

/

nuxt/next/tsrpc...

2

:

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

```
// app.vue
const data = call.api("main")
```

demo

tsrpc

1. TS

2. demo

3. ...

# TSRPC

TSRPC 🐶

:

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

# 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

VSCode

gql ( hasura gql )

```
rontend > src > graphql > TS products.ts > [?] getProducts
1  import gql from 'graphql-tag'
2
3  export const getProducts = gql`
4  query getProducts {
5    pro
6  }`
   products
   products_aggregate
   products_by_pk
```

# 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

