**API-REST**

**Parámetros pasados por url**

/availableSlots?day=12&month=2&year=2022

/availableSlots?month=2&year=2022

const params = getQuery(context, { mergeParams: true });

**Parámetros en el cuerpo (body) de la petición en formato json**

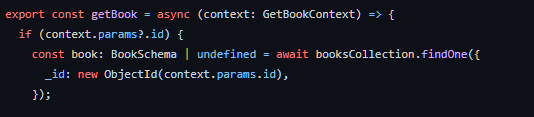
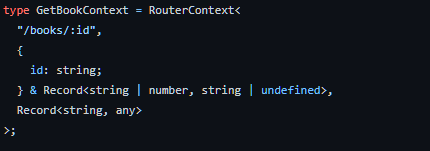
const value = await context.request.body().value;

**Parámetros en json**

const result = context.request.body({ type: "json" });

const value: Slot = await result.value;

const value = await context.request.body({ type: "json" }).value;

**Parámetros pasados por context**

**GRAPHQL**

{

"query": "...",

"operationName": "...",

"variables": { "myVariable": "someValue", ... }

}

operationName y  variables son campos opcionales. operationName solo es necesario si hay varias operaciones presentes en la consulta. Por tanto, casi siempre suelen tener la siguiente estructura:

{

"query": "...",

"variables": { "myVariable": "someValue", ... }

}

export const createCar = async (

  \_: unknown,

  arg: {

    model: string;

    price: number;

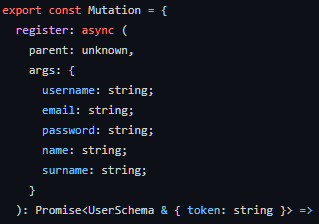
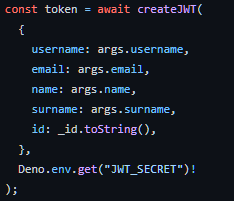
    licensePlate: string;

    salespeopleIDs?: string[];

    dealerID?: string;

  }

): Promise<Car> =>



**ENCADENADOS EN GRAPHQL**

* Devolver un objeto con tipos anidados dentro.
* Sigue la siguiente estrucutra:

export const “nombredeltipo” = {

nombredelapropiedad1 : async (parent : SchemaDelTipo) : Promise <SchemaDeLaPropiedad1/ArrayDeSchemaDeLaPropiedad1/ numero> => {

try{

const loquesea = await ColeccionDeLaPropiedad1.findOne/find ({…});

…

}

catch(e){

throw new Error(e);

}

},

nombredelapropiedad2 : async…

};

