4️⃣ 과제 #4

1. Repository interface

우리 데이터베이스에서 꺼내올 정보이기 때문에 레파지토리에 먼저 메소드를 정의해준다.

```
List<Product> findByProductName(String productName);
```

- productName 을 받아서 product형 List로 반환해준다는 뜻이다.
- Repository는 DTO를 반환형으로 삼을 수 없다.
- 이렇게 레포지토리 인터페이스에 적어두기만 하면 구현은 JPA가 알아서 해준다고 한다..

2. Service

먼저 인터페이스에 메서드를 정의해준다.

반환형은 ProductReturnDto의 리스트이다.

```
//ProductService.java
public List<ProductReturnDto> searchProduct(String productName);
}
```

래포지토리에서 productName으로 데이터를 찾은 다음 `<List> Product`로 반환을 해주는 메서드를 구현했는데, 이것을 productDto 로 변환하고 `<List> productDto`에 add 해줘야한다.

```
public List<ProductReturnDto> searchProduct(String productName){
  try{
      List<Product>productList =productRepository.findByProductName(productName);
        List<ProductReturnDto> productReturnDtoList = new ArrayList<>();
        for(Product product: productList){
                ProductReturnDto productReturnDto = new ProductReturnDto();
                productReturnDto.setId(product.getId());
                productReturnDto.setProductName(product.getProductName());
                productReturnDto.setPrice(product.getPrice());
                productReturnDtoList.add(productReturnDto);
            }
            return productReturnDtoList;

        }catch(Exception e){
            e.printStackTrace();
        }
        return null;
    }
```

`List<Product>productList =productRepository.findByProductName(productName);`

- product 형 List 를 선언해서, 레포지토리의 findByProductName 메서드를 이용해 찾은 데이터를 담아준다.

`List<ProductReturnDto> productReturnDtoList = new ArrayList<>();`

- 결국 우리가 반환해줘야 하는것은 productReturnDto이므로 새로운 List 를 선언한다.

```
for(Product product: productList){
                ProductReturnDto productReturnDto = new ProductReturnDto();
                productReturnDto.setId(product.getId());
                productReturnDto.setProductName(product.getProductName());
                productReturnDto.setPrice(product.getPrice());
                productReturnDtoList.add(productReturnDto);
            }
            return productReturnDtoList;
```

- 찾은 productList를 product 변수에 담고 하나 하나씩 productReturnDto 로 변환하는 반복문을 돌린다.
- `Product product:productList` : productList에 있는 Product를 product에 담고, 그 수만큼 반복문을 진행한다.
- `ProductReturnDto productReturnDto` Prouct 형을 ProductReturnDto으로 변환하기 위한 변수를 선언한다.
- `productReturnDto.set~..` : set 메서드를 통해서 product에서 get메서드를 통해 얻어온 id, price, productName을 productReturnDto 에 넣는다.
- `productReturnDtoList.add(productReturnDto)` : list에 product를 productReturnDto 형태로 변환한 것들을 넣어준다.

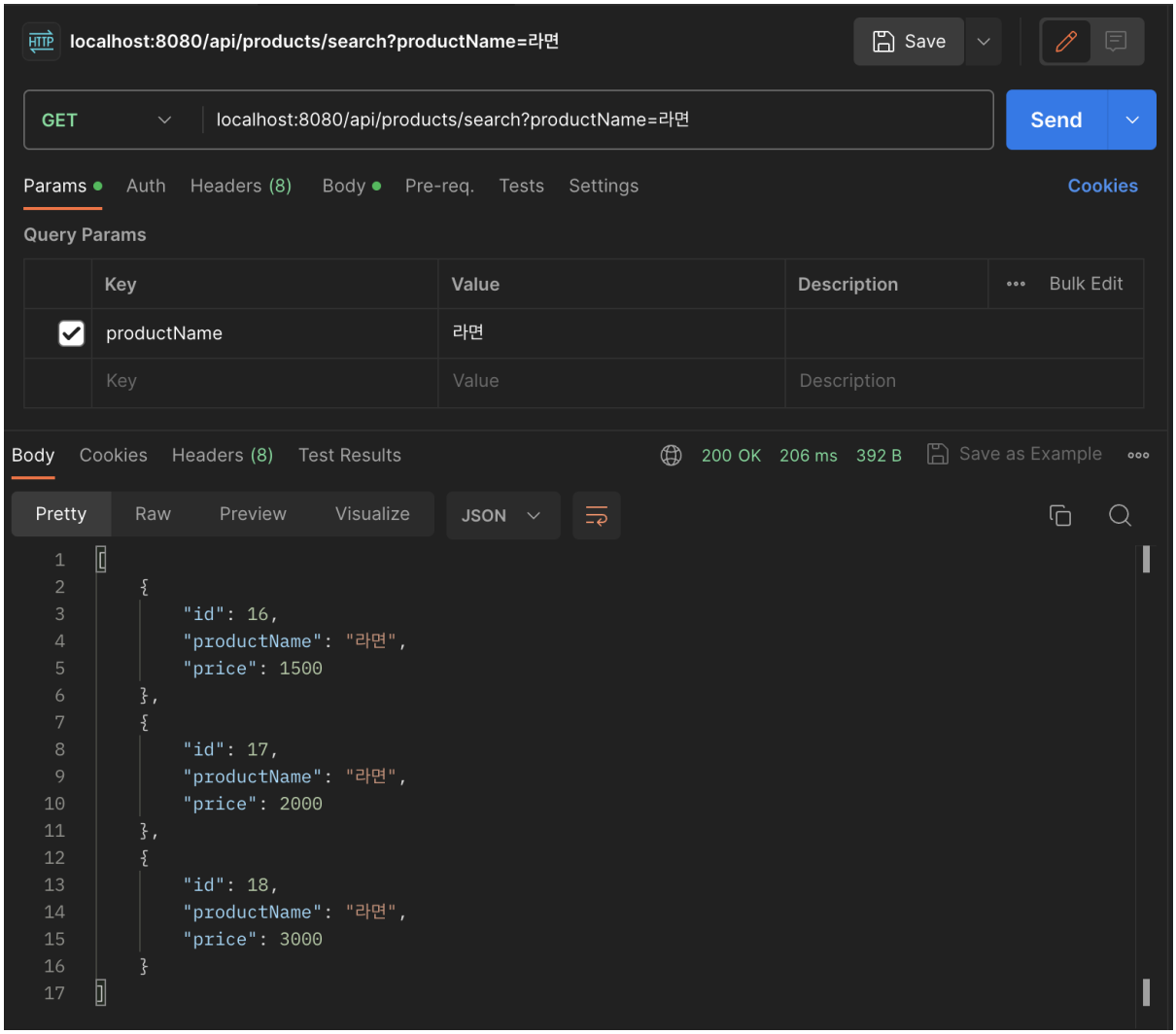
3. Controller

서비스를 다 구현했으니까 이제 사용자의 요청에 맞게 처리해주면 된다.

```
 @GetMapping("/products/search")
    public ResponseEntity <List<ProductReturnDto>> searchProduct(@RequestParam("productName") St
ring productName){
        List<ProductReturnDto> products= productService.searchProduct(productName);
        return ResponseEntity.ok(products);
    }
```

- 정보를 얻어오는 것이기 때문에 GET 메서드이다.
- 또, 요청 URL 에 ?key=value 형태로 전달되기 때문에 @RequestParam 을 사용해 쿼리 파라미터 값을 가져와줘야 한다.
- products/search?productName=".." 이것을 모두 적용하면 형태로 값을 받아오게 된다.

4. 포스트맨 결과



productName= "라면" 인 데이터들만 받아와지는 것을 알 수 있다.