|  |  |
| --- | --- |
| Method | POST |
| URL | /users/join |
| HTTP status code | 성공 201 |
| Request Body | {  email : “사용자가 입력한 이메일”,  password : “사용자가 입력한 비밀번호”  } |
| Response Body |  |

회원 api

1. 회원가입
2. 로그인

|  |  |
| --- | --- |
| Method | POST |
| URL | /users/login |
| HTTP status code | 성공 200 |
| Request Body | {  email : “사용자가 입력한 이메일”,  password : “사용자가 입력한 비밀번호”  } |
| Response cookie | JWT Token |

|  |  |
| --- | --- |
| Method | POST |
| URL | /users/reset |
| HTTP status code | 성공 200 |
| Request Body | {  email : “사용자가 입력한 이메일”,  } |
| Response Body | {  email : “이메일”  } |

1. 비밀번호 초기화 요청

|  |  |
| --- | --- |
| Method | PUT |
| URL | /users/reset |
| HTTP status code | 성공 200 |
| Request Body | {  email : “이전 페이지에서 입력했던 이메일”,  password : “사용자가 입력한 비밀번호”,  } |
| Response Body |  |

1. 비밀번호 초기화(=수정)

도서 API 설계

1. 전체 도서 조회(이미지 경로 n개씩 보내줘야함)

|  |  |
| --- | --- |
| Method | GET |
| URL | /books?limit={page당 도서 수}&currentPage={현재 page} |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body | {//전체도서 목록에는 상세 정보를 포함한다.  {필요한 데이터만 선별하여 구현부탁한다고 FE에게 전달  id : “도서 id”  title : “도서 제목”,  image : 이미지 id(picsum image #id)  summary: “요약 설명”,  author: “도서 작가”,  price : “가격”,  likes: ”좋아요 수”,  pubDate: “출간일”//신간인지 구분하기 위해서 사용”  },  {  id : “도서 id”  title : “도서 제목”,  image : 이미지 id(picsum image #id)  summary: “요약 설명”,  author: “도서 작가”,  price : “가격”,  likes: ”좋아요 수,  pubDate: “출간일”//신간인지 구분하기 위해서 사용”  }  …//나머지 6개 더 있음 총 8개  } |

1. 개별도서조회//이미지 경로 아직 고려하지 않음

|  |  |
| --- | --- |
| Method | GET |
| URL | /books/{bookId} |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body | {  Id : “도서 id”,  title : “도서 제목”,  image : 이미지 id(picsum image #id)  category : ”카테고리”,  format : “포멧”,  isbn : “isbn”,  summary: “요약 설명”,  author: “도서 작가”,  pages : “쪽 수”,  index : “목차”  price : “가격”,  likes : ”좋아요 수”  liked : boolean,  pubDate: “출간일”//신간인지 구분하기 위해서 사용”  } |

1. 카테고리별 도서 목록 조회

//-new : true => 신간조회 (기준 출간일 30일 이내)

//이미지 경로 보내줘야함

//카테고리 id 어떻게 알고 보내줄지 생각해야함

|  |  |
| --- | --- |
| Method | GET |
| URL | /books?categoryId={categoryId}&new={boolean}  // 쿼리스트링으로 카테고리별로 신간을 확인할 수 있음 |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body | {  {  id : “도서 id”  title : “도서 제목”,  image : 이미지 id(picsum image #id)  summary: “요약 설명”,  author: “도서 작가”,  price : “가격”,  likes: ”좋아요 수”,  pubDate: “출간일”//신간인지 구분하기 위해서 사용”  },  {  title : “도서 제목”,  image : 이미지 id(picsum image #id)  summary: “요약 설명”,  author: “도서 작가”,  price : “가격”,  likes: ”좋아요 수”,  pubDate: “출간일”//신간인지 구분하기 위해서 사용”  }  …//여러권이 들어감  } |

카테고리API

1. 카테고리 전체 조회

|  |  |
| --- | --- |
| Method | GET |
| URL | /category |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body | {  {  id : 0,  name : “동화”,  }  {  id : 1,  name : “소설”  }  …  } |

좋아요 API 설계 1차(추후에 설계 예정)—좋아요 테이블을 새로 만들어서 method바꿈

1. 좋아요 추가

|  |  |
| --- | --- |
| Method | POST |
| URL | /likes/{bookId} |
| HTTP status code | 성공 200 |
| Request Body | // 로그인할 때 받은 token >header “Authorization”  // payload값을 읽을 수 있다 => 사용자의 id를 읽어낼 수 있음 |
| Response Body |  |

1. 좋아요 취소

|  |  |
| --- | --- |
| Method | DELETE |
| URL | /likes/{bookId} |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body |  |

장바구니 API 설계

회원 1명당 – 장바구니 1개씩

1. 장바구니 담기

|  |  |
| --- | --- |
| Method | POST |
| URL | /cart |
| HTTP status code | 성공 201 |
| Request Body | {//로그인 할 때 받은 token > header “Authorizaion”  //payload 값을 읽을 수 있다= 사용자의 id를 읽어낼 수 있다.  bookId : “도서 id”,  quantity : 수량  userId : ”회원id”  } |
| Response Body |  |

|  |  |
| --- | --- |
| Method | GET |
| URL | /cart |
| HTTP status code | 성공 200 |
| Request Body | // 로그인할 때 받은 token >header “Authorization”  // payload값을 읽을 수 있다 => 사용자의 id를 읽어낼 수 있음  {  user\_id : 회원id,  selected : [ cartItemId, cartItemId,…]  } |
| Response Body | {  {  id : “장바구니 도서 id”,  bookId : “도서 id”,  title : “도서 제목”,  summary : “도서 요약”,  quantity : 수량,  price : 가격  },  {  id : “장바구니 도서 id”,  bookId : “도서 id”,  title : “도서 제목”,  summary : “도서 요약”,  quantity : 수량,  price : 가격  }  } |

1. 장바구니 조회 / 선택한 장바구니 상품 목록 조회
2. 장바구니 제거

|  |  |
| --- | --- |
| Method | DELETE |
| URL | /cart/{bookId} |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body |  |

1. 선택한 장바구니 상품 목록 조회(2번에 묶어서 들어옴)

|  |  |
| --- | --- |
| Method | GET |
| URL | /carts |
| HTTP status code | 성공 200 |
| Request Body | [  cartItemId , cartitemId  ] |
| Response Body | {  {  id : “장바구니 도서 id”,  bookId : “도서 id”,  title : “도서 제목”,  summary : “도서 요약”,  quantity : 수량,  price : 가격  },  {  id : “장바구니 도서 id”,  bookId : “도서 id”,  title : “도서 제목”,  summary : “도서 요약”,  quantity : 수량,  price : 가격  }  ,…  } |

결제(주문) API

1. 결제하기 = 주문하기 =주문등록 = 데이터베이스 주문 insert

|  |  |
| --- | --- |
| Method | POST |
| URL | /oreders |
| HTTP status code | 성공 200 |
| Request Body | {  items :  [{  cartItemId : 장바구니 도서 id  bookId : “도서id”  count : 수량  },  {  cartItemId : 장바구니 도서 id  bookId : “도서id”  count : 수량  },  …],  delivery : {  address : “주소”,  recevier : “이름”,  contact : “010-0000-0000”,  }  totalPrice : 총 금액  } |
| Response Body |  |

=장바구니에서 주문한 상품은 delete해야함

1. 주문 목록 조회

|  |  |
| --- | --- |
| Method | GET |
| URL | /oreders |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body | [  {  order\_id : 주문id  created\_at : “주문일자”,  delivery :{  address : “주소”,  recevier : “이름”,  contact : “전화번호”  }  bookTitle : “대표 책 제목”,  totalPrice : 결제 금액,  totalCount : 총 수량  }  ] |

1. 주문 상세 상품 조회

|  |  |
| --- | --- |
| Method | GET |
| URL | /oreders |
| HTTP status code | 성공 200 |
| Request Body |  |
| Response Body | [  {  bookId : 도서id,  bookTitle : “도서 제목”,  author : “작가명”,  price : 가격,  count : 수량  },  {  bookId : 도서id,  bookTitle : “도서 제목”,  author : “작가명”,  price : 가격,  count : 수량  },…  ] |