

CARDSELECTER

카드 정보 사이트

선정동기 및 벤치마킹 사이트

선정동기:

대학생들은 사회 초년생으로써 체크카드나 신용카드를 스스로 만드는 경우가 많이 생긴다. 이 경우 소비 패턴에 따라 나에게 알맞는 카드를 만들어야 하는데, 수 많은 카드 중에서 내게 맞는 카드를 찾는 것은 쉽지 않은 일이다. 어떤 카드를 만들어야 할 지 고민되는 사람들을 위해 카드 정보 사이트를 제작하게 되었다.

벤치마킹 사이트:

카드 고릴라: <https://www.card-gorilla.com/home>

신한 카드: <https://www.shinhancard.com/>

구현 목표

1. 학습을 위한 포트폴리오 제작
2. 팀 프로젝트 이전의 개인프로젝트 이므로, 새로운 툴과 라이브러리들의 폭넓은 경험 쌓기
3. 지속적으로 재사용 가능한 코드
4. QnA게시판과 카드 리스트를 활용한 유기적인 이동 구조
5. 카드를 커스텀하는 기능을 통한 차별성 부여
6. 테스트 주도 개발로, 프론트 없이 서버만 만들어보는 경험

사용 기술

백엔드



프론트



개발도구



Springboot 3.0.0
IntelliJ 17.0.4
Java 17.0.4.1
MySQL 8.0.31
React 9.1.3
Mybatis 3.0.1
JWT 0.9.1

주요 기능

로그인 / 로그아웃 / 회원가입

패스워드 암호화 및 액세스 토큰으로 로그인 관리

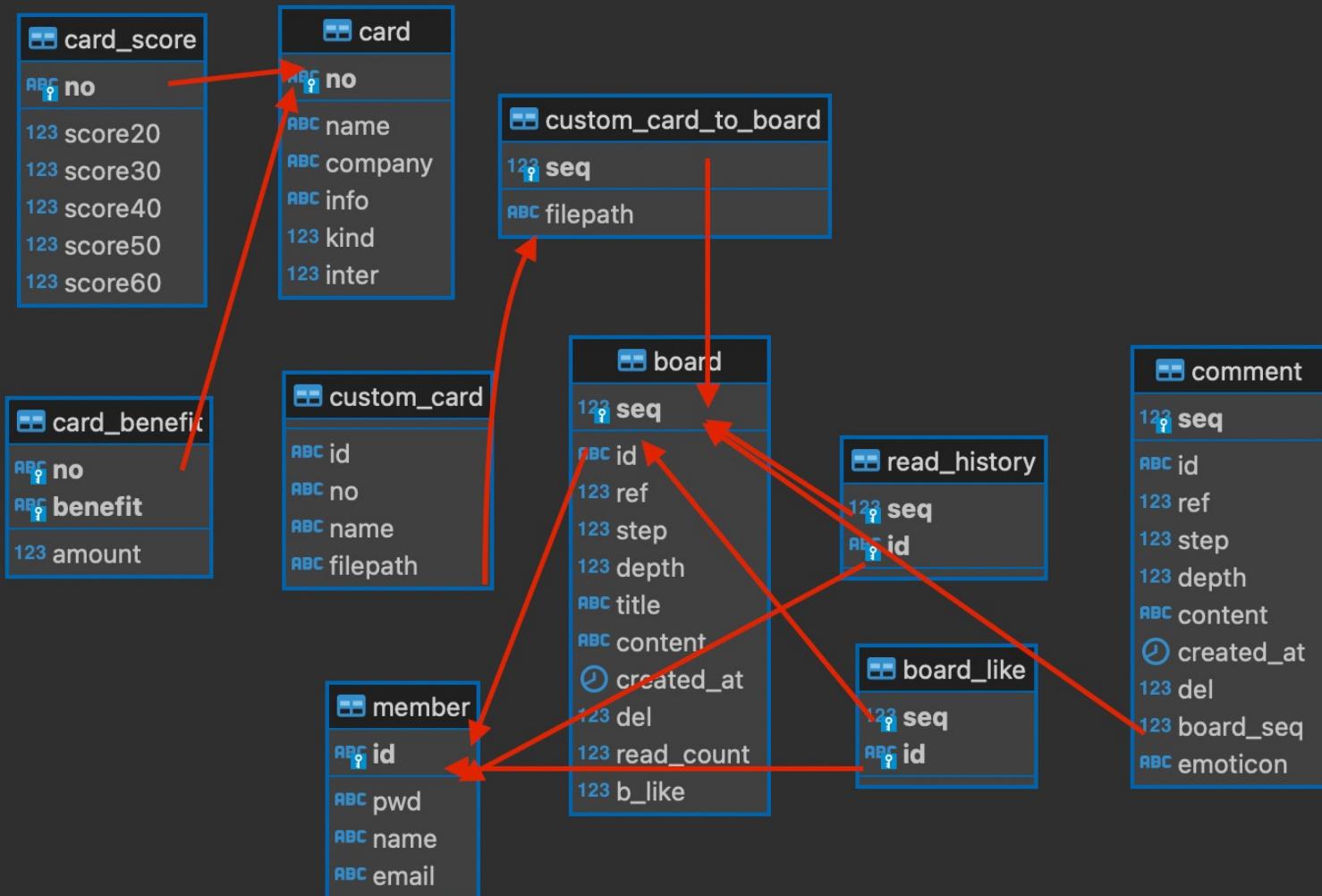
카드 리뷰 및 Q&A 게시판(계층형, 파일 업로드 불가능하지만 즐겨찾기 한 카드의 이미지 게시 가능)

카드 순위별 view(카테고리별)

커스텀 카드 기능(카드의 이미지를 사용자 임의로 변경, 즐겨찾기의 대체)

카드게시판과 QnA게시판의 유기적인 결합

DB 테이블



DB 테이블

ENTITY

Board

- int seq > pk
- String id
- int ref
- int step
- int depth
- String title
- String content
- String createdAt
- int del
- int readCount
- int like

• 읽은거 또 읽으면 조회수 카운트 X
 • 로그인한 사람이 읽었을 때만 조회수 카운트
 • del은 지우기 아닌 숨기기 1이면 숨겨진 상태, 0이면 오픈된 상태(디폴트 0)

BoardHistory

- int seq > fk (board.seq)
- String id > fk (member.id)

• 게시글을 조회하면, 게시글 seq와 id가 테이블에 저장
 • 조회수를 추가할 때, 해당 테이블의 글 번호와 아이디를 비교하여 일치한다면 좋아요 수 증가
 • isExistById와 같이, int형으로 where절 결과값을 count 한다.
 • 두 개 뮤어 primary key하고 insert 안되면 ~ 이라는 조건으로

BoardLike

- int seq > fk (board.seq)
- String id > fk (member.id)

• 게시글에 좋아요를 누르면, 게시글 seq와 id가 테이블에 저장
 • 좋아요 수를 초기화 때, 해당 테이블의 글 번호와 아이디를 비교하여 일치한다면 좋아요 수 증가
 • isExistById와 같이, int형으로 where절 결과값을 count 한다.
 • 두 개 뮤어 primary key하고 insert 안되면 ~ 이라는 조건으로

Comment

- int seq > pk
- int boardSeq > fk (board.seq)
- String id > fk (member.id)
- String content
- String createdAt
- int del
- int ref
- int step
- int depth

• 게시글에 따른 댓글

Member

- String id > pk
- String pwd
- String name
- String email

• 패스워드는 암호화된 상태로 DB에 저장
 • 이미일은 FE에서 형식을 검사하고, 스프링에선 하지 않음

CustomCard

- String no > fk (Card.no)
- String id > fk (Member.id)
- String filepath
- String name

• custom-card 기능에서 카드 이미지를 생성하면, 해당 내용이 저장되는 DB

CustomCardToBoard

- int seq > fk (Board.seq)
- String filepath > fk (MemberLikeCustomCard.filepath)

• 게시판의 기능 중 하나로, 카드의 이미지를 보여준다. 단, 무분별한 이미지 사용을 방지하기 위하여 사용자가 즐겨찾기해놓은 카드만을 가져온다.
 • 카드 업로드 버튼을 누르면, 즐겨 찾기한 카드리스트가 나오고 그 중 하나를 선택하는 방식으로 게시글 사이에 카드를 넣을 수 있으며, 해당 카드는 위에 리스트업되어 클릭시 정보창으로 간다.

Card

- String name
- String no > pk
- String company
- String info
- int kind
- int inter

• 카드 정보
 • no: 회사이니셜+번호
 • filepath: no로 대체
 • kind: 1제크, 2신용
 • inter: 1국내, 2마스터, 3비자, 4유니온

CardScore

- String no > fk, pk(Card.no)
- float score20
- float score30
- float score40
- float score50
- float score60

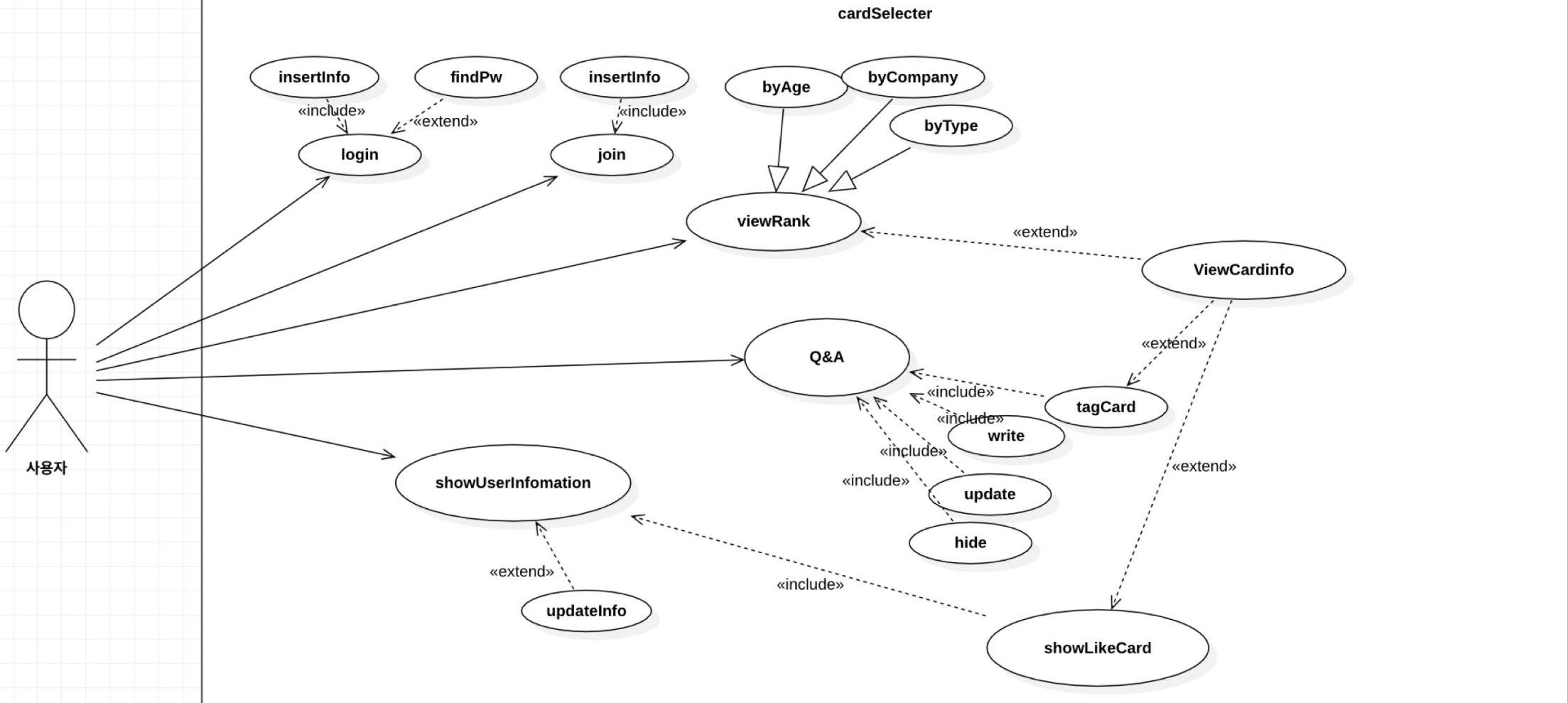
• 연령별 카드 점수(랭킹에 활용)

CardBenefit

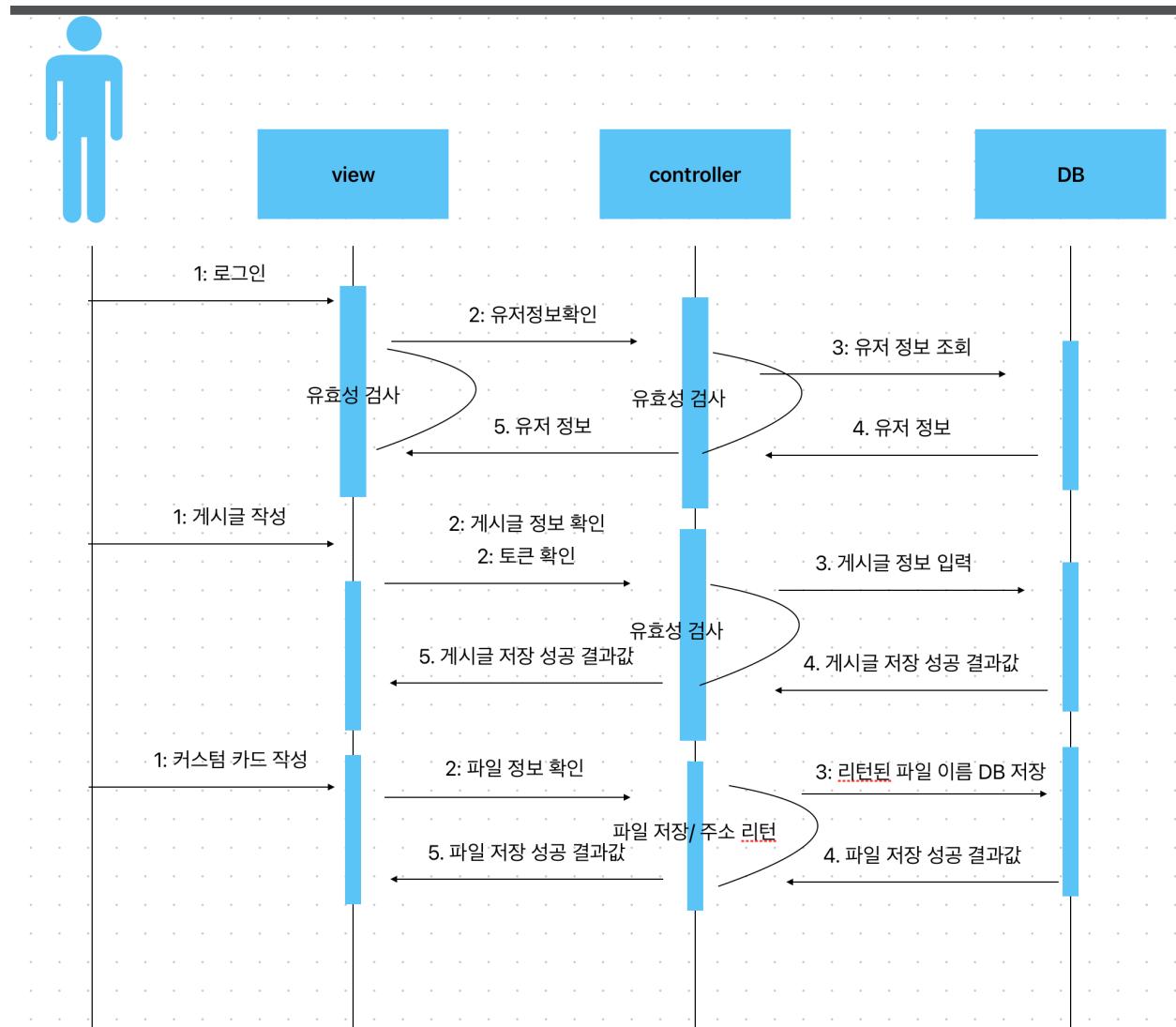
- String no > fk (Card.no)
- String bene
- float amount

• 카드별 혜택과 소수점 한자리 수치

유스케이스 다이어그램



시퀀스 다이어그램



URL Mapping

	URL		Controller
메인페이지	/		/
게시판 리스트	/board	GET	getBoardList
게시판 리스트(id)	/board/my	GET	getBoardListById
게시글상세	/board/{seq}	GET	getBoard
게시글 작성	/board	POST	createBoard
게시글 답글 작성	/board/{pSeq}/answer	POST	createBoardAnswer
좋아요 추가	/{seq}/like	PATCH	updateLike
게시글 수정	/{seq}	PATCH	updateBoard
게시글 삭제	/{seq}	DELETE	deleteBoard
	URL		Controller
댓글목록보기	/comment	get	getCommentList
댓글작성	/comment	post	createComment
답댓글작성	/{parentSeq}/answer	post	createCommentAnswer
댓글수정	/{seq}	Patch	updateComment
댓글삭제	/{seq}	delete	deleteComment
	URL		Controller
카드 리스트 조회 메인	/card	get	getCardList
카드 리스트 조회 차트	/card/chart/{orderBy}	get	getCardChart
카드 정보 확인	/card/{no}	get	getCard
커스텀 카드 조회	/card/customCard	get	getCustomCardList
커스텀 카드 생성	/card/customCard	Post	createCustomCard
커스텀 카드 삭제	/card/customCard	delete	deleteCustomCard
	URL		Controller
아이디 중복체크	/user	get	checkIdDuplicate
회원가입	/user/join	post	join
로그인	/user/login	post	login

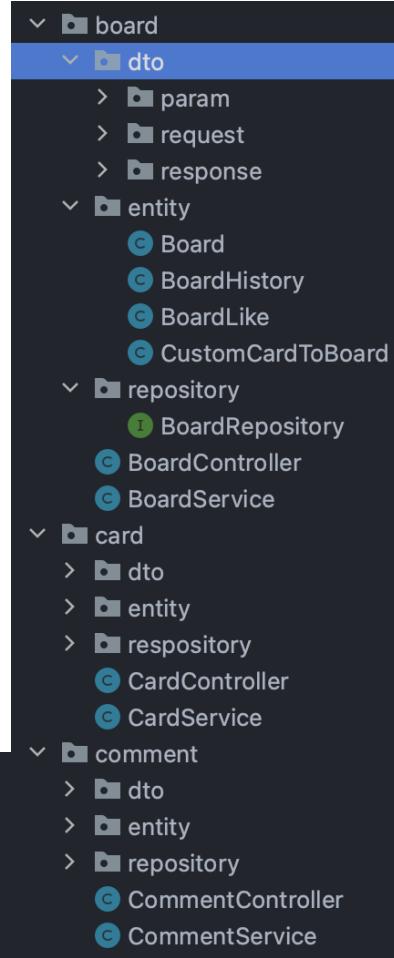
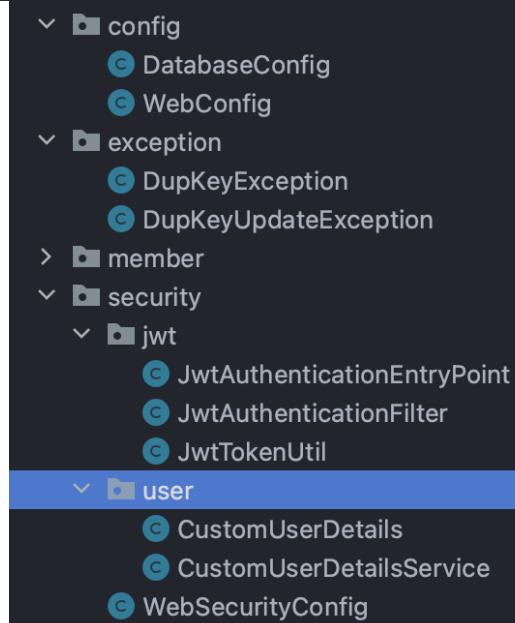
일정표

																			서버(스프링부트)								프론트(리액트)						
11/30	12/1	12/2	12/3	12/4	12/5	12/6	12/7	12/8	12/9	12/10	12/11	12/12	12/13	12/14	12/15	12/16	12/17	12/18	12/19	12/20	12/21	12/22	12/23	12/24	12/25	12/26	12/27	12/28	12/29	12/30	12/31	1/1	1/2
요구사항분석 유스케이스 다이어그램 사용API테스트	와이어프레임 UI ERD제작 사용API테스트(분석)	기말	fe:typescript fe:정규표현식 bfe:로그인 bfe:회원가입 be:암호보안	spring 구조 테스트 및 기말	스프링 구조 및 시큐리티 / jwt토큰 / 스프링 + 리액트 사용법 학습	멤버 시큐리티 jwt 구현	보드	보드 sql로직	코멘트	카드	예외처리	메인 로그인 게시판 카드 카드info	mypage 커스텀 카드 게시판 카드 카드info	DB 최종 정리	발표 자료 준비																		

테스트 주도 개발 및 폴더링

```
23 >   □ TWkim  
24     @Test  
25     void getBoardList() {  
26         BoardListResponse boardList = boardService.getBoardList(new BoardListRequest( choice: "", search: "", page: 1, itemC  
27     }  
28  
29 >   □ TWkim  
30     @Test  
31     void getBoard() {  
32         try{  
33             BoardPostResponse boardPostResponse = boardService.getBoard( seq: 28, id: "id");  
34             System.out.println(boardPostResponse);  
35         } catch (DupKeyException e){  
36             ResponseEntity<String> stringResponseEntity = new ResponseEntity<>(e.getMessage(), e.getStatus());  
37             System.out.println(stringResponseEntity);  
38         }  
39     }  
40 }
```

프론트를 붙이지 않고, 데이터 값을 넣어서 결과물을 확인했다.
장점은 해당 기능을 포함하는 프론트를 제작할 때까지 기다릴 필요가 없다는 장점이 있으며, 짧은 시간 동안 수차례의 테스트를, Jwt토큰이나 세션과 같은 유효성 검사를 피한 상태로 내가 원하는 기능만 추려서 테스트가 가능하기 때문에 시간 절약의 효과가 있다.



구역별로 컨트롤러를 전부 합쳤으며, DB와 1대1 매칭되는 Entity클래스, Sql쿼리문 하나와 1대1 매칭되는 Repository 클래스(dao), 각종 데이터를 요청하거나 응답할 때 DB와 1대1 매칭 되는 엔티티 클래스 대신 필요한 부분만 사용하는 dto 클래스로 나누어 폴더링 했다.
또한 전역적으로 활용되는 Config, exception, security 의 경우에는 따로 폴더를 빼서 정리했다.

프론트엔드 구조

```
function App() {
  return (
    <div className="App">
      <BrowserRouter>
        <ScrollToTop />
        <Header/>
        <Nav/>
        <Main/>
        <Footer/>
      </BrowserRouter>
    </div>
  );
}
```

Index.html -> index.css 이후 실행되는 App.js에는 최상단 로고가 있는 헤더, 메뉴바가 있는 Nav, 화면 가운데인 Main, Footer를 두고 Main만 갈아끼우는 방식으로 구조를 짰기 때문에,

유저 입장에서 일관적인 UI를 경험할 수 있게 한다.

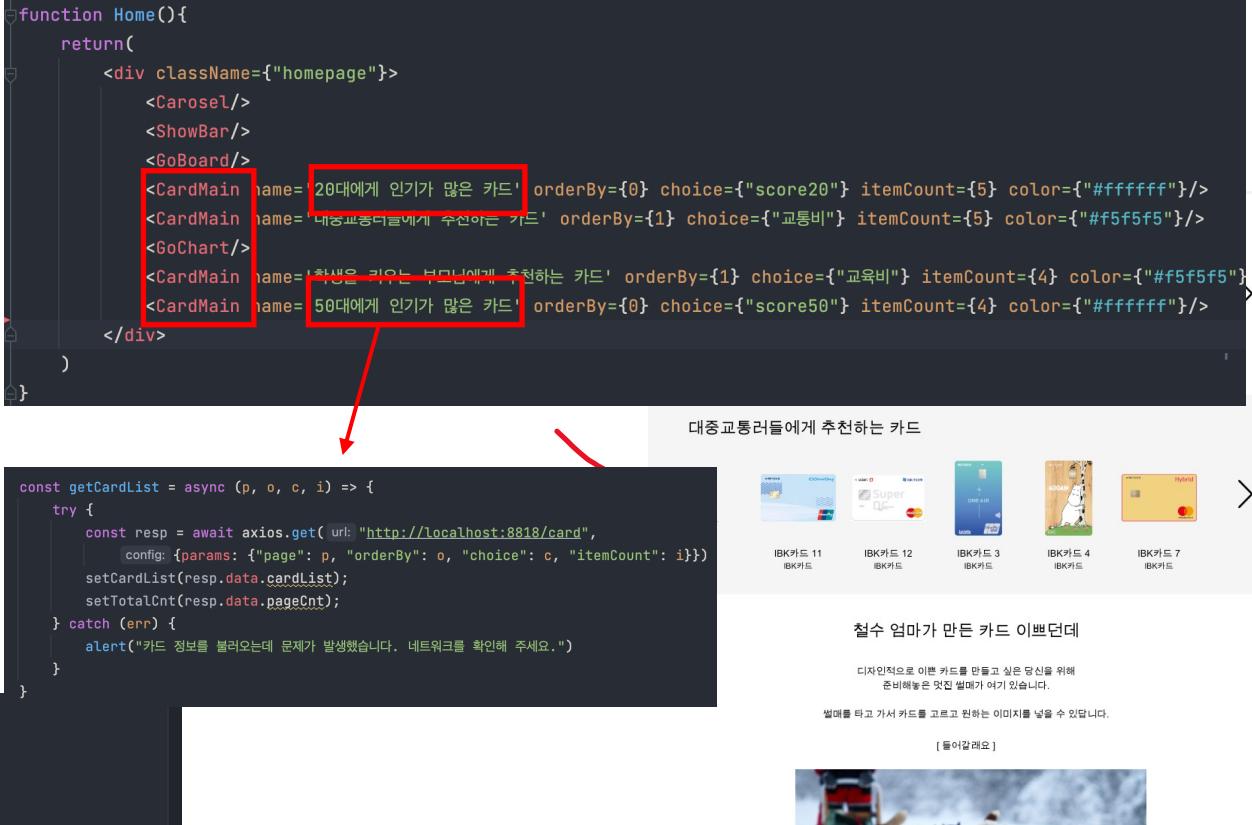
```
function Main() {
  return (
    <main>
      <div className="py-4">
        <div className="container">
          <Router></Router>
        </div>
      </main>
    );
}

function Router(){
  return(
    <Routes>
      <Route path="/" element={<Home/>} />
      <Route path="/join" element={<Join/>} />
      <Route path="/logout" element={<Logout />} />
      <Route path="/card/chart/:orderBy" element={<CardChart/>} />
      <Route path="/card/detail/:no" element={<CardDetail/>} /></Route>
      <Route path="/board/list" element={<BoardList />} /></Route>
      <Route path="/board/write" element={<BoardWrite />} /></Route>
      <Route path="/board/detail/:seq" element={<BoardDetail />} /></Route>
      <Route path="/board/update" element={<BoardUpdate />} /></Route>
      <Route path="/board/answer/:parentSeq" element={<BoardAnswer />} /></Route>
      <Route path={"/mypage/:id"} element={<MyPage/>} />
    </Routes>
  )
}
```

```
function Home(){
  return(
    <div className="homepage">
      <Carousel/>
      <ShowBar/>
      <GoBoard/>
      <CardMain name="20대에게 인기가 많은 카드" orderBy={0} choice={"score20"} itemCount={5} color={"#ffffff"} />
      <CardMain name="내돈짜증들에게 추천하는 카드" orderBy={1} choice={"교통비"} itemCount={5} color={"#f5f5f5"} />
      <GoChart/>
      <CardMain name="학생을 위한 부모님에게 추천하는 카드" orderBy={1} choice={"교육비"} itemCount={4} color={"#f5f5f5"} />
      <CardMain name="50대에게 인기가 많은 카드" orderBy={0} choice={"score50"} itemCount={4} color={"#ffffff"} />
    </div>
  );
}

const getCardList = async (p, o, c, i) => {
  try {
    const resp = await axios.get(`http://localhost:8818/card`, {
      config: {params: {page: p, orderBy: o, choice: c, itemCount: i}}
    });
    setCardList(resp.data.cardList);
    setTotalCnt(resp.data.pageCnt);
  } catch (err) {
    alert("카드 정보를 불러오는데 문제가 발생했습니다. 네트워크를 확인해 주세요.");
  }
}
```

App.js에서 메인이 되는 Main.js에선 라우터의 모임을 가져와서, 이후엔 라우터의 링크를 통해 화면 깜빡임 없이 화면을 이동하게 된다.



Path가 "/"인 Home을 보면, CardMain이 많은 것을 볼 수 있는데 이는 코드를 복록처럼 훌뿌려놓고 조립하는 식으로 짜서 보다 효율적으로 활용하기 위해 하나의 (jsx{html})가 data를 받아서 해당 data값을 뿌려주는 방식으로 설계했다.



View-join

CARDSELECTER 카드를 고르는 현명한 방법

회원가입 로그인

연령별 순위 헤택별 카드 회사별 카드 계시판

회원가입을 하러 오셨나요?

간단한 정보를 입력해주시면
회원가입이 단숨에 완료됩니다.

아이디
 아이디를 확인해주세요(5~20자 내 영문).

닉네임

비밀번호
 비밀번호를 확인해 주세요(8~16자, 영문, 숫자, 특수문자 포함).

비밀번호 확인
 비밀번호를 입력해주세요.

이메일

회원가입

Axios기반 비동기 http통신

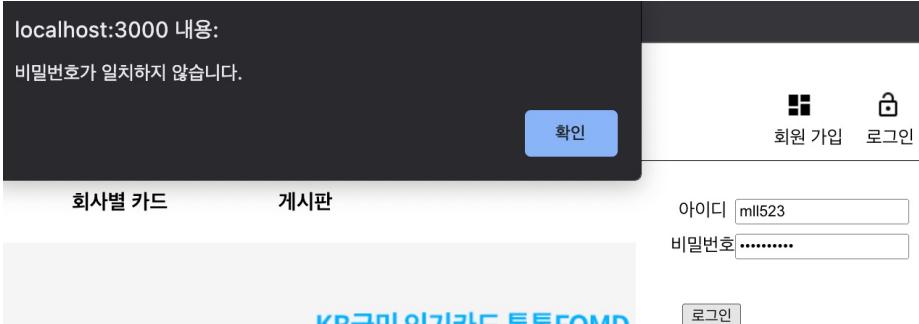
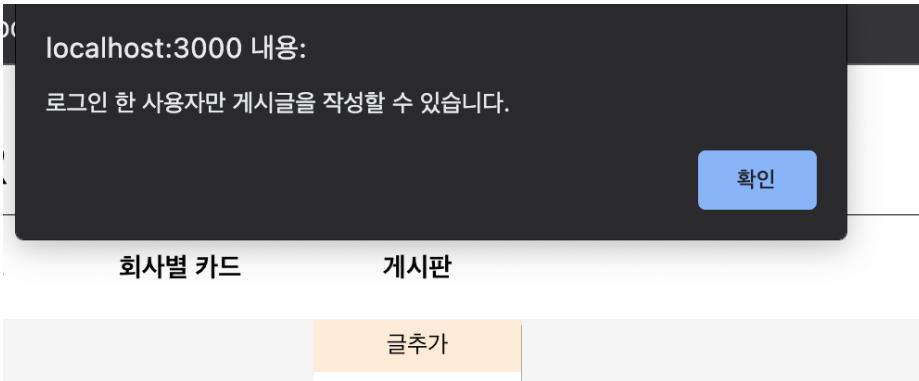
```
@GetMapping
public ResponseEntity<?> checkIdDuplicate(@RequestParam String id) {
    HttpStatus status = service.checkIdDuplicate(id);
    log.info("checkIdDuplicate {}", status);
    return new ResponseEntity<>(status);
}

<select id="isExistUserId" parameterType="String" resultType="java.lang.Integer">
    SELECT count(*)
    FROM member
    WHERE id=#{id}
</select>

/* 아이디 중복 체크 */
const checkIdDuplicate = async () => {
    try {
        if(checkId(id)) {
            const resp = await axios.get( url: "http://localhost:8818/user", config: {params: {id: id}});
            if (resp.status === 200)
                setMsg( value: "사용 가능한 아이디입니다. ")
        } else {
            setMsg( value: "아이디를 확인해주세요(5~20자 내 영문). ")
        }
    } catch (err) {
        const resp = err.response;
        if (resp.status === 400)
            setMsg( value: "이미 있거나 삭제된 아이디입니다. ")
    }
}

const checkId = (id) => {
    const regExp = /^[a-z]+[a-z0-9]{4,19}$/g;
    return regExp.test(id)
}
```

View - security



JWT 토큰을 통한 사용자 요청 관리로,
서버 부담 감소 +
패스워드 인코더를 통한 패스워드 암호화

id	pwd
kkjkkjj	\$2a\$10\$47NBf7rlXhp05M9s/6bZLubHW55jFWP6ql
mll5234	\$2a\$10\$fCXKYG2v3gAEZ/qTQI394u1vfBxoayixiuM9

```
const createBoard = async () => {
  console.log(selectCard);
  console.log(user.jwt);
  const req = {
    id: user.id,
    title: title,
    content: content,
    filePath: selectCard.customCard,
  }
  console.log(req)
  try {
    const resp = await axios.post( url: "http://localhost:8818/board", req, config: {headers: user.jwt})
    alert("게시글을 성공적으로 등록했습니다.");
    navigate(`/board/detail/${resp.data.seq}`);
  } catch (err) {
    alert("게시글을 등록하는데 실패했습니다. \n 원인: " + err.response.data)
  }
}

@Bean
public SecurityFilterChain filterChain(HttpSecurity httpSecurity) throws Exception{
  httpSecurity
    .csrf().disable()
    .cors()
    .and()
    .authorizeHttpRequests()
    .requestMatchers(HttpServletRequest.POST, ...patterns: "/board", "/comment").authenticated()
    .requestMatchers(HttpServletRequest.PATCH, ...patterns: "/board", "/comment").authenticated()
    .requestMatchers(HttpServletRequest.DELETE, ...patterns: "/board", "/comment").authenticated()
    .anyRequest().permitAll();
}
```

1 usage

```
String HEADER_STRING = "Authorization";
```

2 usages

```
String TOKEN_PREFIX = "Bearer ";
```

Jwt 토큰은 Authorization: Bearer ~~~.~~~.~~~ 와 같은 구조다.

View-chart

CARDSELECTER 카드를 고르는 현명한 방법

회원가입 로그인

연령별 순위 헤택별 카드 회사별 카드 게시판

20대 차트
30대 차트
40대 차트
50대 차트
60대 차트

롯데카드 5

카셀차트 TOP 1 ~ 10 총 95개

1 롯데카드 5
광고나 간단한 글이 들어갈 영역

2 신한카드 1
광고나 간단한 글이 들어갈 영역

3 삼성카드 20
광고나 간단한 글이 들어갈 영역

4 국민카드 26
광고나 간단한 글이 들어갈 영역

5 국민카드 24
광고나 간단한 글이 들어갈 영역

6 국민카드 14
광고나 간단한 글이 들어갈 영역

10 국민카드 23
광고나 간단한 글이 들어갈 영역

1 2 3 4 5 > »

다양한 조건으로 정렬해줌(연령별, 혜택별, 회사별)
코드의 재사용성을 최대한 고려하여 설계(페이지는 하나만 사용하고,
State값을 다르게 줌)

```
<Link to={`/card/chart/0`} state={{  
    choice: "score30", itemCount:10}}>  
    <p>40대 차트</p>  
</Link>  
  
li>  
i>  
    <Link to={`/card/chart/0`} state={{  
        choice: "score30", itemCount:10}}>  
        <p>50대 차트</p>  
</Link>  
  
li>  
i>  
    <Link to={`/card/chart/0`} state={{  
        choice: "score30", itemCount:10}}>  
        <p>60대 차트</p>  
</Link>
```

娄 WKim
@GetMapping("/chart/{orderBy}")
public ResponseEntity<CardListResponse> getCardChart(@PathVariable Integer orderBy,
@RequestParam String choice,
@RequestParam Integer page,
@RequestParam Integer itemCount) {

```
CardListRequest cardListRequest = new CardListRequest(page, orderBy, choice, itemCount);  
CardListResponse cardList = service.getCardList(cardListRequest);  
log.info("cardList {}", cardList);  
return ResponseEntity.ok(cardList);  
}
```

```
select no, name, company, info, kind, inter  
<if test="orderBy == 0">  
from  
(  
    select row_number() over (order by ${choice}) as rnum,  
          a.no, name, company, info, kind, inter  
     from card a  
    left join card_score b on a.no = b.no  
) c  
    WHERE rnum BETWEEN ${pageStart} AND ${pageEnd}  
</if>  
<if test="orderBy == 1">  
from  
(  
    select row_number() over (order by name) as rnum,  
          a.no, name, company, info, kind, inter  
     from card a  
    left join card_benefit b on a.no = b.no  
    where b.benefit = #{choice}  
) c  
    WHERE rnum BETWEEN ${pageStart} AND ${pageEnd}
```

View-board(1)

계층형 게시판으로, 검색기능, 답글기능,
한 페이지에 들어오는 게시글 갯수 커스텀 전부 지원
또한 최신글은 맨 위로 정렬하며, 자연스러움을 위해 답글은
최신글이 아래로 가도록 배치함

번호	제목	작성자	조회수	추천수	작성시간
1	L 모 O 곤	kkjkkjji	1	0	2023-01-03 08:23:14
2	국민카드 20줄다	kkjkkjji	1	0	2023-01-03 06:30:20
3	dd	kkjkkjji	1	1	2023-01-03 06:16:32
4	dd	kkjkkjji	1	0	2023-01-03 05:57:54
5	이미지 테스트	kkjkkjji	1	1	2023-01-03 05:39:38
6	글 잘쓰지나	kkjkkjji	1	0	2023-01-03 05:36:58
7	글테스트	kkjkkjji	1	0	2023-01-03 05:36:44
8	asdf	kkjkkjji	1	0	2023-01-03 05:36:05
9	ASF	kkjkkjji	1	0	
10	fas	kkjkkjji	1	0	
11	asdf	kkjkkjji	1	0	
12	aaddfa	kkjkkjji	1	1	
13	ㅁㄴㅇㄹ	kkjkkjji	1	0	
14	L 모 O 곤	kkjkkjji	1	0	
15	L asdf	kkjkkjji	1	0	
16	L ffa	kkjkkjji	1	0	
17	ㅁㄴㅇㄹ	kkjkkjji	1	0	2023-01-02 21:45:23
18	⚠️ 이 글은 작성자에 의해 삭제됐습니다.				
19	L 안녕하세요. 답글입니다.	kkjkkjji	1	0	2023-01-02 21:23:00

```
<!-- 답글 작성 -->
<!-- 부모 게시글 답글 위치 결정(아래에 추가, 가운데에 넣기) -->
<select id="updateBoardCheck" parameterType="Integer" resultType="Integer">
    select ifnull(min(step), 0)
    from board
    where ref = (select ref from board a where seq = #{parentSeq})
    and step > (select step from board b where seq = #{parentSeq})
    and (select depth from board c where seq = ${parentSeq}) >= depth
</select>

<!-- step 업데이트 -->
"updateBoardStep" parameterType="twk.cardselecter.board.dto.param.BoardStep"
    board
    p = step + 1
    ef = (select ref from (select ref from board a where seq=#{parentSeq}) A)
    p >= #{checkResult};
```

```
- 새로운 답글 추가 -->
sert id="createBoardAnswer" parameterType="twk.cardselecter.board.dto.param.BoardAnswer"
      useGeneratedKeys="true" keyProperty="seq">
INSERT board (id, ref, step, depth, title, content, created_at, del, read_count, b_like)
    <if test="checkResult == 0">
        values (#{id}, (SELECT ref FROM board a WHERE seq=#{parentSeq}),
                (select ifnull(max(step), 0) + 1 from board b where ref=(select ref FROM
                (SELECT depth FROM board d WHERE seq=#{parentSeq}) + 1,
                #{title}, #{content}, NOW(), 0, 0, 0);
    </if>
    <if test="checkResult > 0">
        VALUES (#{id}, (SELECT ref FROM board e WHERE seq=#{parentSeq}),
        #{checkResult},
        (SELECT depth FROM board f WHERE seq=#{parentSeq}) + 1,
        #{title}, #{content}, NOW(), 0, 0, 0);
    </if>
```

View-board(2)

The screenshot shows two screens of a mobile application. The top screen is titled 'CARDSELECT' and displays a message: '게시글에 좋아요를 누르는데 문제가 생겼습니다. 원인: 좋아요는 게시글당 한 번만 가능합니다.' It has a '확인' button. The bottom screen shows a board detail with a green header '질문을 하면 아이디어가 샘솟는답니다.' and two icons. It lists posts by 'asdf' and 'kkjkkkj' with their respective creation dates and counts of likes and views.

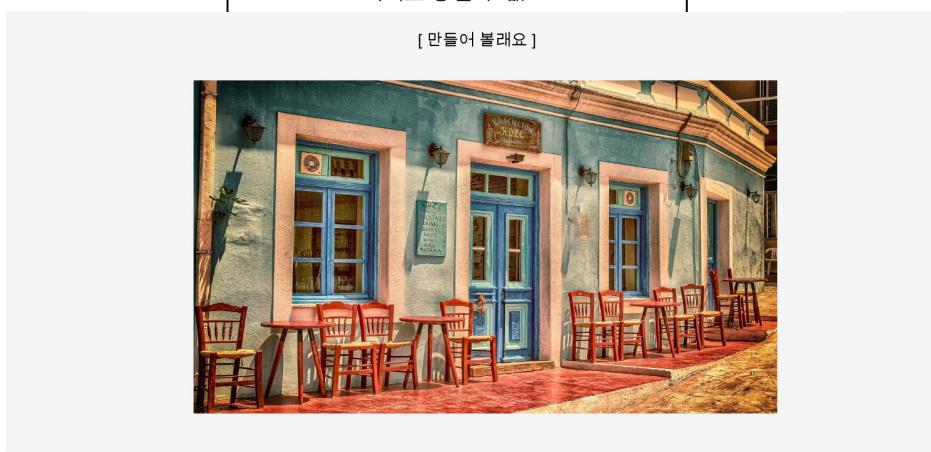
답글달기

수정 삭제

답글달기

View-card&board

언령별 순위 헤택별 카드 회사별 카드 게시판

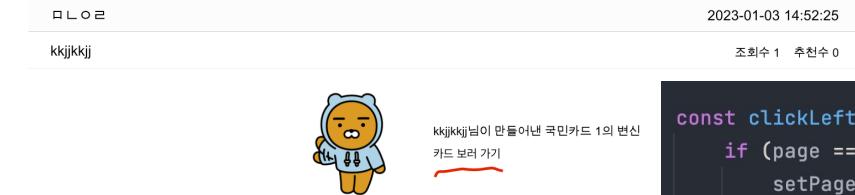
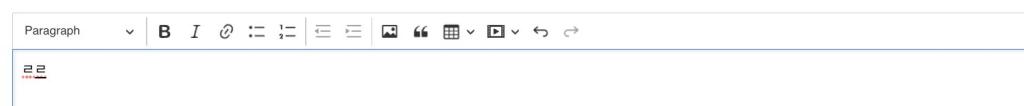
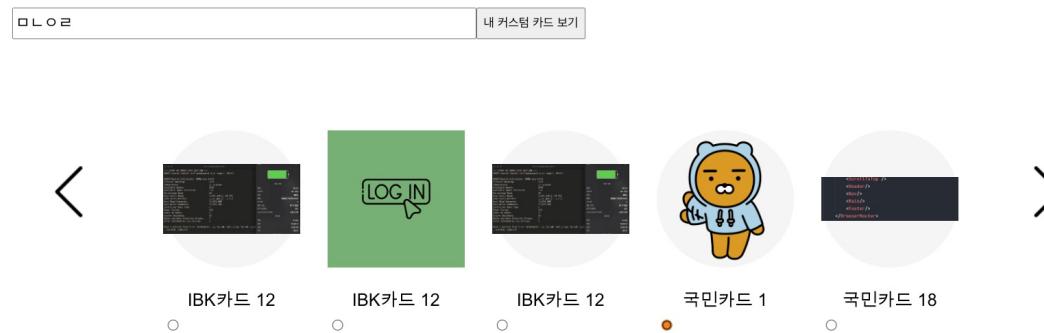


[파일 선택] 002.png [Upload]

멋진 카드가 만들어졌어요!!



카드와 QnA 게시판간의 유기적인 구조를 보여주는 부분이다.
카드 디테일페이지에서 사용자가 파일을 업로드해서 커스텀 카드를
만들 수 있으며, 만든 카드를 질문글에 활용 할 수 있다.
또한 게시글에 카드를 사용시 카드 바로가기 링크가 생성된다.
사진은 커스텀카드 무한화살표 로직



```
const clickLeft = () => {
  if (page === 1) {
    setPage(maxPage);
    getCardlist(page, data.orderBy, data.choice, data.itemCount);
  } else {
    setPage(value: page - 1);
    getCardlist(page, data.orderBy, data.choice, data.itemCount);
  }
}
```



View-card&board



해당 카드가 들어간 QnA글

작성자: kkjkkjj 제목: 모노로 추천도: 0

모노로

kkjkkjj



kkjkkjj님이 만들어낸 글
카드 보러 가기

22



해당 카드가 들어간 QnA글

작성자: kkjkkjj 제목: 모노로 추천도: 1

카드 바로가기 링크를 타고 카드 디테일 창에 가보면 자동으로 해당 카드가 들어간 QnA글 탭이 생긴 것을 확인해 볼 수 있다. 마찬가지로 해당 글을 누르면 이동이 가능하여 게시판과 카드 차트 사이에 짜임새 있는 화면 전환이 가능하다.

```
/**  
 * 커스텀 카드 조회  
 */  
3 usages  ↳ TWkim  
public CardCustomListResponse getCustomCard(CardCustomListRequest req){  
    CardCustomListParam param = new CardCustomListParam(req.getPage(), req.getId());  
    param.setPageParam(req.getPage(), req.getItemCount());  
    List<CustomCard> customCard = repository.getCustomCardList(param);  
    int pageCnt = repository.getCustomCardCount(param);  
    return new CardCustomListResponse(customCard, pageCnt);  
}
```

```
}  
/*updateResult 사용해서 예외처리*/  
/*아래 두개 합치기*/  
String customCardToBoard = boardRepository.getCustomCardToBoard(seq);  
CustomCard customCard = boardRepository.findCustomCardNo(customCardToBoard);  
return new BoardPostResponse(boardRepository.getBoard(seq), customCard);
```

```
CardCustomListResponse  
@Getter  
@AllArgsConstructor  
@ToString  
public class BoardPostResponse {  
    private Board board;  
    private CustomCard customCard;  
}
```

```
/**  
 * 특정 카드 조회(카드 정보, 카드 혜택, 카드 선호도)  
 */  
2 usages  ↳ TWkim  
public CardResponse getCard(String no){  
    Card card = repository.getCard(no);  
    List<Board> board = repository.getBoardToCustomCard(no);  
    List<CardBenefit> cardBenefitList = repository.getCardBenefit(no);  
    Map<Integer, Float> wantAge = wantAge(repository.getCardAge(no).mapScore());  
    return new CardResponse(card, cardBenefitList, wantAge, board);  
}
```

```
@Getter  
@ToString  
@AllArgsConstructor  
public class CardResponse {  
    private Card card;  
    private List<CardBenefit> cardBenefitList;  
    private Map<Integer, Float> wantAge;  
    private List<Board> boardList;  
}
```

View-card&my

카드번호	카드주소	카드주인	카드상태	등록일
1234567890123456	123.123.123.123	IBK카드 12	정상	2023-01-03 14:52:25
1234567890123456	123.123.123.123	IBK카드 12	정상	2023-01-03 08:23:14
1234567890123456	123.123.123.123	국민카드 20좋다	정상	2023-01-03 06:30:20
1234567890123456	123.123.123.123	dd	정상	2023-01-03 06:16:32
1234567890123456	123.123.123.123	dd	정상	2023-01-03 05:57:54
1234567890123456	123.123.123.123	이미지 테스트	정상	2023-01-03 05:39:38
1234567890123456	123.123.123.123	글 잘써지나	정상	2023-01-03 05:36:58
1234567890123456	123.123.123.123	글테스트	정상	2023-01-03 05:36:44
1234567890123456	123.123.123.123	asdf	정상	2023-01-03 05:36:05
1234567890123456	123.123.123.123	ASF	정상	2023-01-03 05:34:58

내가 쓴 글과, 만든 커스텀카드를 볼 수 있는 마이페이지다.
디자인을 보면 알겠지만, 목표사항에 맞게 코드를 효율적으로
재사용한 부분이 많다.

```
{  
    isShow ?  
        <CustomCardList itemCount={5} select={true}/>  
    :  
    null  
}
```

```
function MyPage(){
    return(
        <div>
            <BoardListById/>
            <CustomCardList itemCount={5}/>
        </div>
    )
}

export default MyPage
```

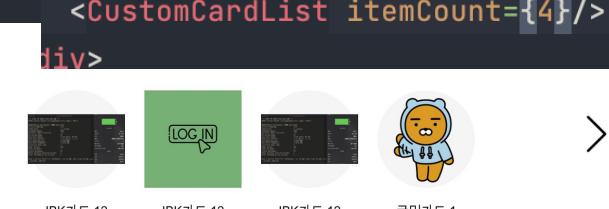
마이페이지에서는 select속성을 <비활성화하여 선택하는 라디오버튼이 없다. itemCount를 조절하여 정렬 방식도

Boardwrite에서는
Select 속성을 활성화
+ 아이템은 5개씩 정렬



국민카드 1

국민카드



IBK카드 12 IBK카드 12 IBK카드 12 국민카드 12

테스트케이스

CSK-CO-001	component	Login	로그인	o	로그인 관련 부분은 기존 세션에서 jwt토큰으로 대체하였기 때문에, 특성을 고려하여 만들지 않은 부분이 많다. 대신 jwt토큰을 활용한 부분을 많이 제작하였다.
CSK-CO-002			아이디 저장	x	
CSK-CO-003			자동 로그인	x	
CSK-CO-004			검	o	
CSK-CO-005			비밀번호 변경	x	
CSK-CO-006			관리자 로그인	x	
CSK-CO-007		SignUp	회원가입	o	
CSK-CA-001	CardDetail	Search	검색	-> 게시글 검색으로 대체	
CSK-CA-002		Chart	카드 차트	o	
CSK-CA-003		RelativeCard	연관 카드	o	
CSK-CA-004		ReviewChart	리뷰 차트	-> 카드 디테일 내 리뷰 및 질문 확인으로 대체	
CSK-CA-005		CustomCard	카드 커스텀	o	
CSK-FO-001	Forum	Reivew Forum	리뷰 게시판	o	
CSK-FO-002		Write	글쓰기	o	
CSK-FO-003		Reply	댓글쓰기	o	
CSK-FO-004		notification	알림	x	
CSK-FO-005		Read	상세글읽기	o	

history

시간 순과 무관하게 정렬

1: boardLike, readHistory => 사용자별로 조회수, 좋아요 수를 체크하기 위해 새로운 테이블을 생성.

기존 누르면 board테이블에 있는 like와 readhistory값이 하나씩 증가했으나,

새로운 테이블 생성 이후에는, primary키인 seq(보드)+id(멤버)값을 추가해보고, insert가 성공하면

조회수 증가, 실패하면 조회수를 증가하지 않는 방식으로 구현하여 중복이 사라짐.

2. customCard: 대학교때 만들어본 머신러닝 모델을 사용하여, 텍스트 기반 랜덤 커스텀 카드 이미지 제작 기능을 넣으려 했으나 개인 컴퓨터로는 성능의 한계가 크게 느껴져서 포기. 대신 차선책으로 파일 이미지 업로드 하는 기능을 제작하여, 커스텀 카드는 사용자가 업로드한 이미지를 기반으로 제작된다.

3. card_benefit, card_score: 시간의 한계로, 카드 혜택 검색은 만들지 않았지만, 해당 기능이 만들어졌을때의 성능적인 부분을 고려하여, 업데이트가 잦은 내용의 테이블을 기능별로 떼어냄.
둘이 따로 떼어낸 이유는, 베네핏은 카드마다 여러개가 있을 수 있고, 연령별 선호도인 score와 다르게 혜택의 가짓수가 무한히 늘어날 수 있는 특징을 가질 수 있기 때문에 따로 떼어둠

4. @validation 검증을 추가 axios의 채용으로, 서버와 프론트간의 연계가 더욱 긴밀해져서,
프론트단에서는 정규식으로, 서버에서는 정규식과 validation검증을 통해 더욱 안정적인 검증이 가능해짐

5. 리액트 특성상 SPA 즉 index.html "/"로만 페이지가 진행되므로 배포시에
새로고침시 주소를 제대로 못찾고 에러페이지가 떠버린다. 에러페이지 출력시 "/"로 들어가게 |바인딩
해준다.

추가 작업 사항

1. 예외처리 보강: 현재 오류가 나와도 관련 메세지가 제대로 안오는 경우가 많음
2. Jwt refresh토큰 사용: 현재는 리프레시토큰에 대한 명확한 정의를 하지 않아서, 사용 중간에 재 로그인을 해야하는 불편함이 있음
3. Spring security 정리: 공부하면서 사용했지만, 아직 부족함이 많음. 내용이 방대해서 천천히 학습할 예정
4. 현재 jwt token을 localstorage에 사용 중인데, httpCookie에 저장.
5. 지금 이미지 업로드할때 id체크를 안 하는데-> 비회원도 가능, 헤더를 같이 보내서 체크하도록 만들기(시간 부족 이슈)