|  |
| --- |
| 리액트와 페이저, 스프링 부트를 이용한  메이플스토리 커뮤니티 사이트 개발 |

텍스트이(가) 표시된 사진

자동 생성된 설명

|  |  |
| --- | --- |
| 프로젝트명 | 메이플스토리 커뮤니티 |
| **개발 기간** | 2022. 01.27~02.11 |
| 작성자 | 성동인(팀장) 이도현(프론트엔드 개발자) 박성철(백엔드 개발자) 탁원석(DB 설계자) |

목차

**1. 서론**

1.1 주제선정 및 배경 4

1.2 기대효과 4

**2. 계획**

2.1 시나리오 및 문제의 정의 4

2.2 일정계획 4

2.3 노력 추정 7

2.4 조직 계획 8

2.5 수행방향 9

2.6 핵심기술 9

2.7 필수기능 10

2.8 시스템 아키텍처 10

2.9 디자인 가이드 10

**3. 요구 분석**

3.1 요구 사항 11

3.2 구조적 분석 11

**4. 화면 설계**

4.1 로그인 13

4.2 맵(전체맵) 13

4.3 맵(상세 맵) 14

4.4 게시판(자유게시판) 14

**5. 화면 구현 / 기능**

5.1 화면 구현 15

**6. 중요 코드**

5.1 화면 구현 22

**7. 정책**

7.1 정책 정의 26

**8. 계획**

8.1 하자 보수 계획 27

8.2 추후 향상 계획 27

**9. 프로젝트 기획안, 시현 영상 링크**

9.1 프로젝트 기획안 링크 28

9.1 시현 영상 링크 28

1. 서론

* 1. **주제선정 및 배경**

최근 다양한 기능을 가진 커뮤니티를 바라는 사용자의 니즈가 꾸준히 상승하고 있다. 옛날부터 사랑을 받아온 메이플스토리의 게임 커뮤니티를 게임적 요소를 추가해 새롭게 만들어 보고자 한다.

* 1. **기대효과**

색다른 기능을 가진 웹페이지로 유저들의 호기심을 자극해 단기간에 상당수의 유저 유입이 기대된다. 또한 다양한 유저 니즈에 맞는 게시판을 다수 지원하여 원활한 유저들의 소통과 정보 공유를 지원한다.

2. 계획

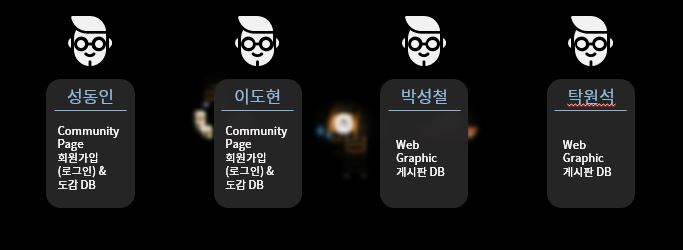
**2.1 시나리오 및 문제의 정의**

오래된 커뮤니티의 기능들을 개선하고 인게임 요소를 추가해 유저들이 같이 참여할 수 있도록 만들어 타 커뮤니티와의 차별화를 두었다.

**2.2 일정계획**

**2.2.1** **작업 분해**

전체 프로젝트의 일정을 예측하기 위하여 소작업들로 분류하였다. WBS를 작성함으로써, 프로젝트 진행에서 일어나는 모든 작업을 찾고 큰 단위의 일을 관리하기 쉬운 작은 일로 나누기 위하여 작성되었다.



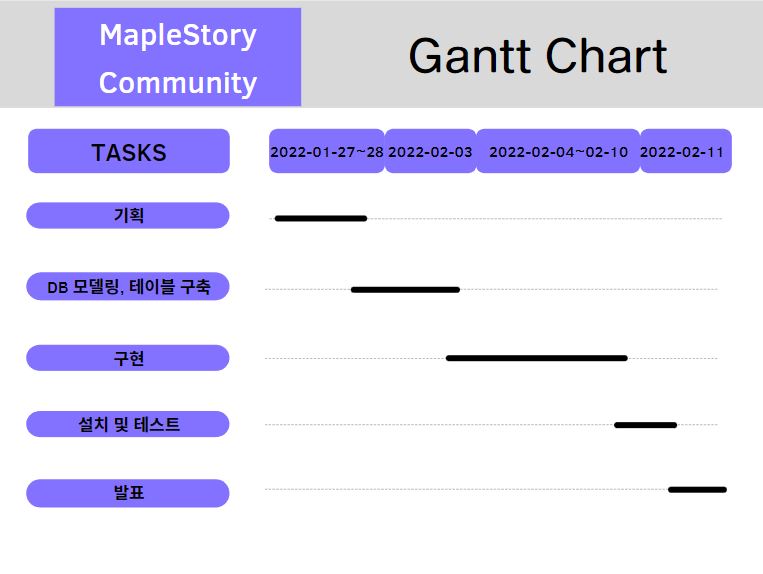
**<그림 1. 작업 분해 (WBS)>**

2.2.2 간트 차트

간트 차트(Gantt chart)는 프로젝트 일정 관리를 위한 바(bar) 형태의 도구이다. 프로젝트의 주요 활동을 파악한 후, 각 활동의 일정을 시작하는 시점과 끝나는 시점을 연결한 막대 모양으로 표시하여 전체 일정을 한눈에 볼 수 있다. 아래처럼 활동은 막대로 표시하고 그 길이는 활동의 작업 시간과 비례한다. 프로젝트를 이루는 소작업 별로 언제 시작되고 언제 끝나야 하는지를 한눈에 볼 수 있도록 프로젝트 일정표를 그렸다.

|  |  |  |
| --- | --- | --- |
| 총 개발 기간 | 2022.01.27 ~ 2022.02.11 (16일) | |
| 프로젝트명 | 메이플스토리 커뮤니티 | |
| 프로젝트 개발 기간 | 작업내용 | 개발 기간 |
| 기획 | 01월 27일 ~ 01월 28일 |
| DB 모델링 구축 | 01월 28일 ~ 02월 03일 |
| 구현 | 02월 03일 ~ 02월 10일 |
| 설치 및 테스트 | 02월 10일 ~ 02월 11일 |
| 제출 | 02월 11일 |
| 개발 순서 | 계획 → 요구 분석 → 설계 → 구현  → 시험 및 유지 보수 → 제출 | |
| 필요 자원 | 개발용 PC, 개발 공간 | |

**<표 1. 메이플스토리 커뮤니티 웹 개발 일정 계획>**



**<그림 2. 간트차트>**

**2.3 노력추정**

커뮤니티 관리 서비스를 개발하기 위하여 드는 비용이 얼마이며 어느 정도의 기간이 걸리는지를 추정할 필요가 있다. 소프트웨어 개발 비용의 대부분은 소요되는 인력 자원이기 때문에 대부분의 비용 예측은 월 인원(Man-Month)으로 표현하는 노력의 추정에 초점을 두고 있다.(\*2021년 SW 기술자 평균 임금 공표를 참고함)

|  |  |  |
| --- | --- | --- |
| 소요내역 | 세부내역 | 산출금액 |
| ① 인건비 | 별첨 참조 | 20,683,136 |
| ② 재경비 | ① × 10% | 2,068,313 |
| ③ VAT | (①+②)×10% | 2,275,144 |
| 총 계 |  | 25,026,593 |

**<표 2. 소요 예산>**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 구분 | 인원 | 일 | 참여율 | 단 가 | 금액 |
| SW개발자 | 4 | 16 | 100% | 323,174(일)/(명) | 20,683,136 |
| 계 |  |  |  |  | 20,683,136 |

**<표 3. 인건비 산정 내역(별첨)>**

**2.4 조직 계획**

프로그래머 팀 구성은 팀 구성원을 회원가입(로그인) & 도감 DB 담당과 게시판 DB 담당을 나누어 배분하였다.



**<그림 3. 조직의 구성>**

위의 노력 추정을 통하여 SW 개발자 4명이 필요하다.

각 기능사들의 업무 기능을 아래의 표로 정리하였다.

|  |  |
| --- | --- |
| **조직 단위** | 업무 기능 |
| SW 개발자 | 요구 분석 및 설계, 중요한 기술적 판단, 작업 |

**<표 4. 조직의 업무 기능>**

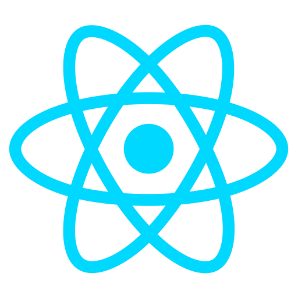
**2.5 수행방향**

이번 프로젝트에서는 실제 업무에서 사용되는 프론트앤드에서의 React와 백앤드에서의 SpringBoot, Node.js, 데이터베이스(Oracle)를 중점적으로 다루고자합니다.

또한 자바스크립트 기반 물리엔진(Phaser)을 사용하여 웹페이지 내에 캐릭터를 생성해 사용자가 조작할 수 있게 함으로써 커뮤니티에서도 게임 내 가상공간을 체험하는 색다른 경험을 제공할 예정입니다.

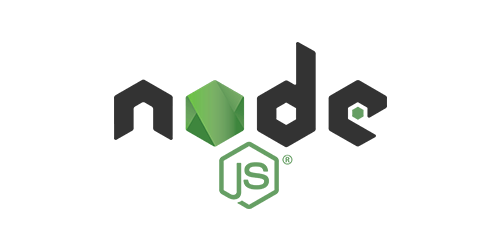
**2.6 핵심기술**

**FrontEnd**

****

**React Phaser**

**BackEnd**

****

**Node.js SpringBoot Oracle**

**2.7 필수기능**

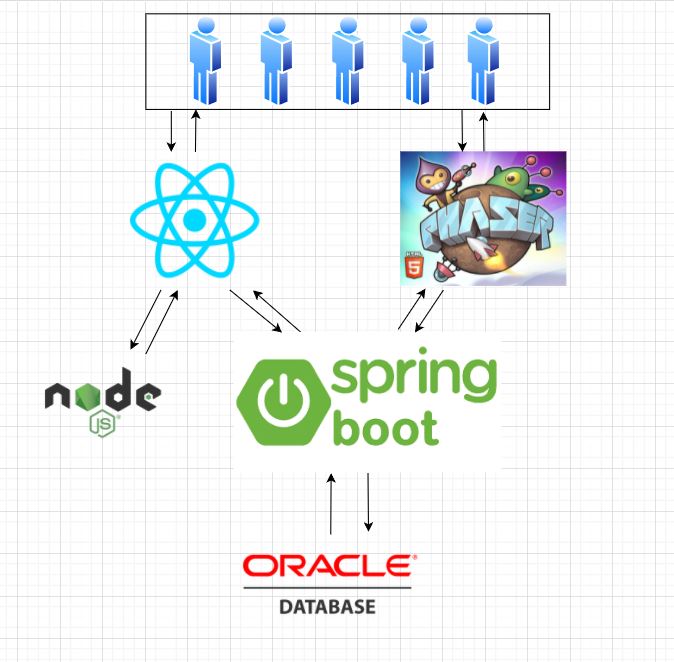
1. 로그인기능

2. 유효성검사

3. 물리엔진(중력 / 속도), 충돌구현(장애물), NPC와 상호작용

4. Router기능, Filter기능(검색, 체크박스), Axios(서버통신)

5. JPA(DB)

**2.8 시스템 아키텍처**

**2.9 디자인 가이드**

주요 게임내 이미지를 가져와서 맵 구현에 활용할 예정입니다.

메인맵 : MapleStory “루디브리엄 메인맵”

서브맵 : MapleStory “루디브리엄 물약상점”, “루디브리엄 장비상점”, “루디브리엄 자유시장”, “루디브리엄 선착장”

NPC : MapleStory “전직NPC 하인즈 외 4명”

로그인 / 회원가입 : MapleStory “(구)로그인화면”

3. 요구 분석

**3.1 요구 사항**

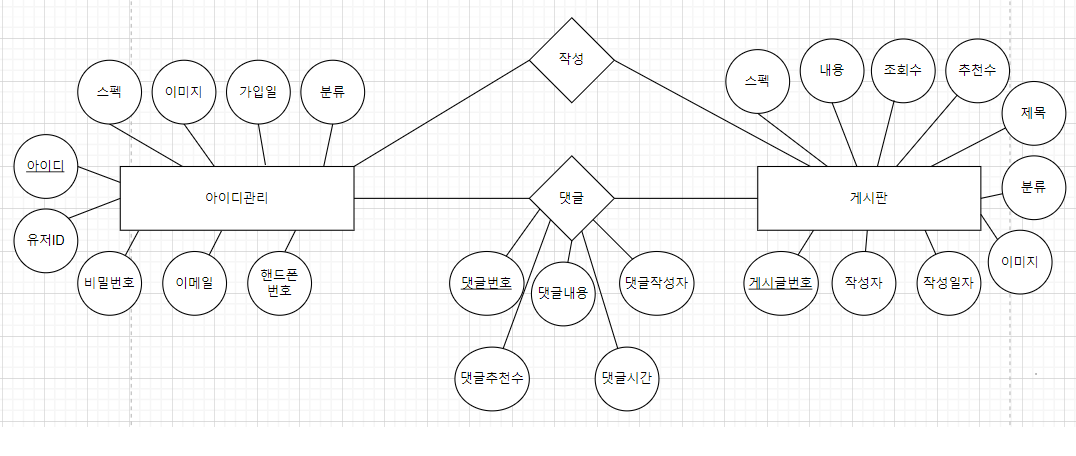
1. UI 재구성

2. 게임플레이에 도움이 되는 공략 및 정보 제공

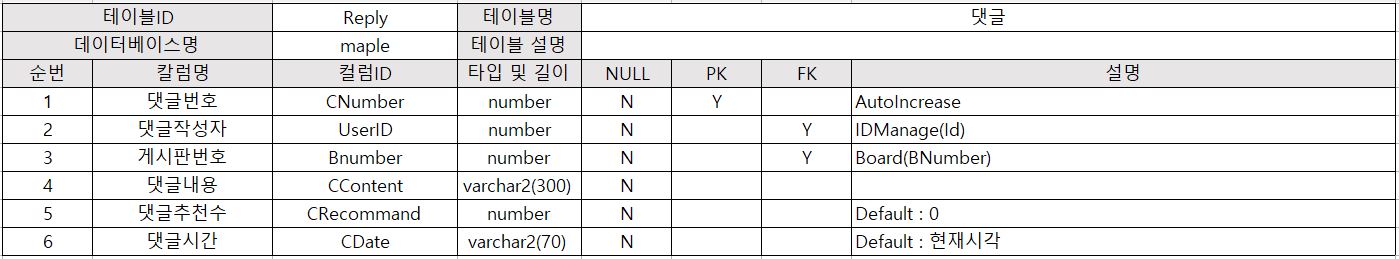
3. 유저간의 소통공간

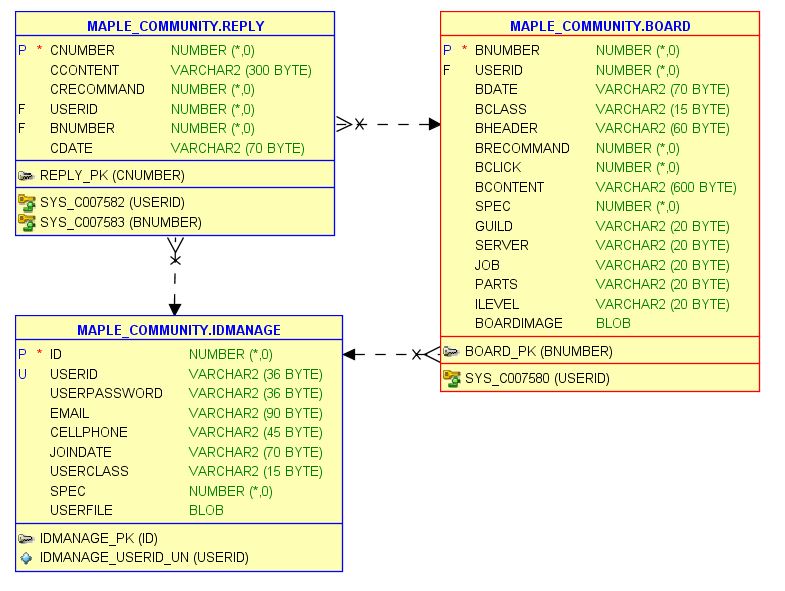
**3.2 구조적 분석**

개념적 설계 단계에서 ERD와 테이블 명세서를 작성하였다. 아이디관리, 게시판, 도감, 이미지, 댓글 테이블이 구성되어 있고, 아이디관리는 아이디를 게시판은 게시판 번호를 댓글은 댓글번호를 도감은 도감 번호를 이미지는 이미지 번호를 기본 키로 설정하였다.



<**그림 4. ERD>**

**<표 5. 테이블명세서>**



**<그림 5. DB모델링>**

4. 화면 설계

**4.1 로그인**

<**그림 6. 로그인>**

**4.2 맵(전체맵)**

<**그림 7. 맵(전체맵)>**

**4.3 맵(상세맵)**

<**그림 8. 맵(상세맵)>**

**4.4 게시판(자유게시판)**

<**그림 9. 게시판(자유게시판)>**

5. 화면 구현 / 기능

|  |
| --- |
| <남자캐릭터>  <여자캐릭터> |
| **로그인 기능**  -유효성 검사 후 로그인  -검사 중 아이디, 비밀번호 DB와 일치 확인은 백엔드에서 함  - 안에 화살표를 클릭하여 캐릭터를 변경 가능  -로그인 성공 시 유저 데이터를 가지고 메인 맵으로 이동  -비회원 로그인 시 유저 데이터 없이 메인 맵으로 이동(게시판 쓰기 X) |
|  |
| **회원가입 기능**  -각 input 값마다 정규 표현식 검사  -아이디 중복 확인 버튼으로 DB 정보와 비교  -회원가입 시 DB에 회원정보 저장 |

|  |
| --- |
|  |
| 팬아트 게시판  -글쓰기 버튼 클릭 시 ‘내 컴퓨터’ 폴더 내에 파일을 불러와 미리 보기 화면을 출력한다.  -미리 보기 된 화면의 글쓰기버튼을 누르면 DB로 전송한다.  -리본돼지사진 을 클릭 시 팝업창에 사진을 불러오고 작성자가 아닌 유저가 수정할 시  ALERT 창을 나타내 경고한다.  - |

|  |
| --- |
|  |
| **게시판 목록**  -제목(하이퍼링크)을 클릭 ‘보기’페이지로 넘어가 글 내용을 확인할 수 있다.  -‘WRITE’ 버튼을 클릭 시 ‘글 게시 작성’ 페이지로 넘어가 글을 작성할 수 있다.  -RADIO 체크를 활용하여 글 목록을 정렬할 수 있다. |

|  |
| --- |
| <보기 페이지>  <글게시작성 페이지> |
| 게시판 쓰기/보기  -서버/레벨 등 선택하는 값은 ‘SELECT’태그 기능을 넣어 선택할 수 있도록 작업했다.  -‘글 게시’ 버튼을 클릭 시 작성한 값을 DB로 전송한다.  -‘보기 페이지’에 추천을 눌러 의 값을 1씩 증가시킬 수 있다.  -‘ |

|  |
| --- |
| <종합게시판>  <뉴스게시판>  <아이템거래게시판>  <팬아트 갤러리> |
| **NPC 별 게시판 기능**  -NPC 클릭 시 특정 게시판 팝업 출력  -게시판 링크 클릭 시 해당 게시판 이동 |
| <앉기>  <공격 모션> |
| **특정 행동 구현**  -의자 위치에서 X 키 누르면 앉기/ Ctrl키 누르면 공격 모션 |

|  |
| --- |
| <전사전직관>  <해적전직관>  <도적 전직관>  <마법사 전직관> |
| **직업 NPC**  -각 직업군의 NPC를 클릭하면 상세 직업을 선택할 수 있는 팝업창이 생성된다.  -상제 직업 팝업창에서 버튼을 클릭 시 이미지 팝업창이 생성된다. |
|  |
| **메이플 도감사전**  -맵에서 도감 표지판을 클릭하여 도감 팝업창을 생성한다.  -검색 기능을 활용하여 각 레벨대의 장비와 몬스터 사냥터를 검색할 수 있다. |
|  |
| 마이페이지  -‘회원정보 변경’버튼을 클릭 시 변경된 회원정보를 DB로 보낸다.  -‘회원 탈퇴’버튼을 클릭 시 DB에 저장되어 있는 값을 삭제한다.  -관리자 아이디로 접속할 시 ‘관리 페이지’에서 회원들의 스펙을 설정해 게시판 이용을 돕는다. |

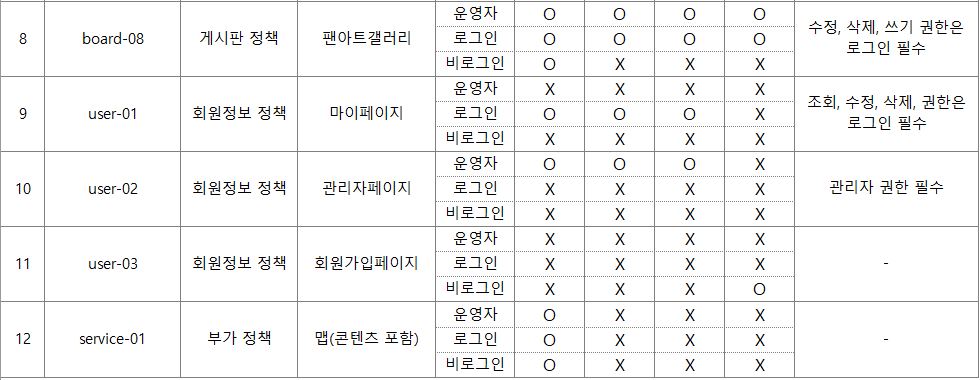
**6. 중요 코드**

|  |
| --- |
| //로그인 정보 받아오는 코드 -  **const** **[**user**,** setUser**]** **=** useState**({**  userid**:** **null,**  userclass**:** **null,**  joindate**:** **null,**  email**:** **null,**  cellPhone**:** **null,**  spec**:** **null**  **});**  **var** url **=** "http://localhost:8088/getLoginData"**;** //backend의 데이터 불러오기  useEffect**(()** **=>** **{**  axios  **.**post**(**url**,** **null,** **{**  params**:** **{**  id**:** `${params}`  **},**  **})**  **.**then**((**response**)** **=>** **{**  console**.**log**(**response**);**  console**.**log**(**response**.**data**);**  console**.**log**(**response**.**data**.**userid**);**  console**.**log**(**response**.**data**.**userclass**);**  console**.**log**(**response**.**data**.**joindate**);**  setUser**(**user **=>** **({**  **...**user**,**  userid**:** response**.**data**.**userid**,**  userclass**:** response**.**data**.**userclass**,**  joindate**:** response**.**data**.**joindate**,**  email**:** response**.**data**.**email**,**  cellPhone**:** response**.**data**.**cellphone**,**  spec**:** response**.**data**.**spec  **}));**  **});**  **},** **[]);**  console**.**log**(**user**);** |
| 첫 번째로 유저 데이터 변수를 선언한 후 axios 비동기 방식을 사용하여 스프링에서 유저 데이터를 받아옴. 로그인한 유저를 url params를 사용하여 특정하고, 스프링으로 보낸 params 값으로 db에서 유저를 찾은 뒤, 특정한 유저의 정보를 응답받아 유저 데이터 변수에 초기화 시킴. |
| // 회원 가입 버튼  $**(**".signUpBtn"**).**click**(function** **(event)** **{**  **let** submitData **=** **{**  userId**:** $**(**".inputId"**).**val**(),**  userPassword**:** $**(**".inputPw"**).**val**(),**  email**:** $**(**".inputEmail"**).**val**(),**  cellPhone**:** $**(**".inputPhoneNum"**).**val**(),**  **class** **:** "회원"**,**  date **:** dateString**,**  **}**  // console.log(`JSON.stringify(submitData) is ${JSON.stringify(submitData)}`);  $**.**ajax**({**  headers**:** **{**  "Accept"**:** "application/json"**,**  "Content-Type"**:** "application/json"  **},**  url**:** "signup/create"**,**  type**:** "POST"**,**  data**:** JSON**.**stringify**(**submitData**),**  dataType**:** "text"**,** // 서버에서 받을 dataType  success**:** **function** **(**data**)** **{**  console**.**log**(**`success!, data = ${data}`**);**  **alert(**"회원가입 성공!"**);**  inputValueInitialize**();** // input 태그에 입력한 값들 초기화  **},**  error**:** **function** **(**request**,** error**)** **{**  console**.**log**(**` error!, requset.status = ${request.status}\n request.responseText = ${request.responseText}\n error = ${error}`**);**  **},**  complete**:** **function** **()** **{**  console**.**log**(**"complete!, bye!"**);**  **}**  **});**  **});** |
| 회원가입 버튼을 누르면 ajax를 사용하여 중복 확인 및 정규 표현식을 충족하는 데이터만 JSON 타입으로 스프링으로 POST 요청을 보냄. 요청 성공 시 회원가입 성공 알림 창과 함께 기존에 입력했던 데이터를 초기화함. |

|  |
| --- |
| //댓글 조회  @PostMapping**(**"/allreply"**)**  **public** List**<**REPLY**>** allreply**(**@RequestParam**(**"BOARDNO"**)** String boardno**)** **{**  System**.**out**.**println**(**"댓글 보드 넘버 ->" **+** boardno**);**  List**<**REPLY**>** replies **=** **new** ArrayList**<**REPLY**>();**  List**<**REPLY**>** replies1 **=** Replyrepository**.**findAll**();**  //System.out.println("리플라이스 1 ->"+replies1.get(0).getCCONTENT());  **for(**REPLY reply **:** replies1**)** **{**  **if(**reply**.**getBoard**().**getBNUMBER**()** **==** Long**.**parseLong**(**boardno**))** **{**  System**.**out**.**println**(**"댓글 트루"**);**  REPLY reply1 **=** **new** REPLY**();**  BOARD board **=** **new** BOARD**();**  BOARD board1 **=** reply**.**getBoard**();**  board**.**setBNUMBER**(**board1**.**getBNUMBER**());**  reply1**.**setBoard**(**board**);**  reply1**.**setCDATE**(**reply**.**getCDATE**());**  reply1**.**setCCONTENT**(**reply**.**getCCONTENT**());**  reply1**.**setCRECOMMAND**(**reply**.**getCRECOMMAND**());**  reply1**.**setCNUMBER**(**reply**.**getCNUMBER**());**  Idmanage idmanage **=** reply**.**getidmanage**();**  System**.**out**.**println**(**"댓글 아이디매니지 ->"**+**idmanage**.**getUSERID**());**  idmanage**.**setUSERFILE**(null);**  reply1**.**setidmanage**(**idmanage**);**  replies**.**add**(**reply1**);**  **}**  **}**  //System.out.println("마지막 댓글 확인 - > " +replies.get(0).getidmanage().getUSERID());  **return** replies**;**  **}** |
| 게시글에 있는 댓글 조회 시 게시글 번호를 axios를 통해 스프링으로 POST 요청을 보냄.  받은 게시글 번호 데이터를 for 문으로 대조하여 요청과 일치하는 데이터를 특정한 뒤  특정한 댓글 데이터를 반환함. |

|  |
| --- |
| //댓글 삭제  @PostMapping**(**"/deletereply"**)**  **public** **void** deletereply**(**@RequestParam**(**"CNUMBER"**)** String cnumber**)** **{**  REPLY reply **=** Replyrepository**.**findById**(**Long**.**parseLong**(**cnumber**)).**get**();**  Replyrepository**.delete(**reply**);**  Replyrepository**.**flush**();**  **}**  //댓글 업데이트  @PostMapping**(**"/updatereply"**)**  **public** **void** updatereply**(**@RequestParam**(**"CNUMBER"**)** String cnumber**,** @RequestParam**(**"content"**)** String content**,** @RequestParam**(**"DATE"**)** String date**)** **{**  REPLY reply **=** Replyrepository**.**findById**(**Long**.**parseLong**(**cnumber**)).**get**();**  reply**.**setCCONTENT**(**content**);**  reply**.**setCDATE**(**date**);**  Replyrepository**.**flush**();**  **}**  //게시글 업데이트 부분 ----------------------------------------------------------  @PostMapping**(**"/freeupdate/{bnum}"**)**  **public** String freeupdate**(**@PathVariable Long bnum**,** BoardDto dto**,**@RequestParam**(**"USERID"**)**String userid**)** **{**  System**.**out**.**println**(**"파라메터 dto 값->"**+**dto**.**toString**());**  Idmanage idmanage **=**IdManagerepository**.**findById**(**Long**.**parseLong**(**userid**)).**get**();**  //1 dto를 엔티티로 변환  BOARD boardEntity **=** dto**.**toEntity**();**  System**.**out**.**println**(**"바꾸려는 값 ->" **+** boardEntity**.**toString**());**  boardEntity**.**setIdmanage**(**idmanage**);**  boardrepository**.**save**(**boardEntity**);**  boardrepository**.**flush**();**  **return** ""**;**  **}** |
| 댓글 삭제  댓글 번호를 변수로 POST 요청을 받아 댓글 데이터를 특정한 뒤 JPA를 사용하여 삭제함.  댓글 업데이트  댓글 번호와 내용, 날짜를 변수로 요청을 받은 뒤 댓글 번호 변수를 사용하여 댓글을 특정한 뒤  내용과 날짜 변수를 특정한 댓글 데이터에 JPA를 통해 저장함.  게시글 업데이트  url 마지막에 게시글 번호를 넣고 PathVariable 로 게시글 번호를 변수로 받은 뒤 JPA를 사용하여 게시글을 특정. 유저가 수정한 데이터를 받은 dto를 게시글 엔티티로 변환 뒤 JPA를 사용하여  저장. |

**7. 정책**

**7.1 정책 정의**

**<표 6. 정책정의서>**

8. 계획

**8.1 하자 보수 계획**

웹사이트 개발한 회사(갑)와 웹사이트 개발을 의뢰한 회사(을)는 다음의 사항을 준수해야

합니다.

갑은 하자 보수 요청 양식을 통해 요청된 웹사이트 변경안만 수용할 수 있습니다. 승인된 보수

계획에는 다음 기준이 적용됩니다.

1. 하자 보수 기간 내 웹사이트

을은 갑이 개발한 하자 보수 기간 내 웹사이트를 소유하고 있어야 합니다. 만약 기간이 명시되지 않은 경우 1년을 기본으로 적용합니다. 기간 내 하자 보수 요청은 횟수의 제한이 없으며 주요하지 않은 기능의 경우 타기능과 병합되거나 요청이 거절될 수 있습니다.

2. 고품질 웹사이트 보증

갑이 개발한 웹사이트는 우수한 사용자 환경을 제공할 수 있도록 충분한 양의 콘텐츠, 최신 상태의 이벤트 및 정보, 명확한 탐색 기능, 명확한 클릭 유도 문안을 제공합니다.

해당 웹사이트는 빠르게 로드되고, 깨진 링크가 포함되어 있지 않으며, HTTPS로 보호됩니다.

3. 새로운 웹사이트 기능 추가

웹사이트의 도메인을 추가하거나 기능을 추가(업데이트) 하려면 추가 웹사이트 개발 요청 양식을 작성하여야 합니다. 해당 요청은 영업일 기준 10일 이내에 검토됩니다.

해당 정책을 위반하는 것으로 확인된 모든 계약은 별도의 통보 없이 자동으로 해지될 수 있습니다. 갑은 어떤 이유로든 언제든지 을의 신청을 승인하거나 거부할 수 있습니다. 또한 언제든지 이 가이드라인을 보완하거나 수정할 수 있습니다.

**7.2 추후 향상 계획**

인게임 디자인 변경 시 실시간으로 메인맵 변경

게시판 UI의 최신화

인기 게시물 게시판 추가

9. 프로젝트 기획안, 시현 영상 링크

**9.1 프로젝트 기획안 링크**

**9.2 시현 영상 링크**