###### 

|  |
| --- |
| **업무기능 : 메인 타임라인(Friends 제외)** |
| Request URL : /news/type/:sort\_type?:my\_id&:page\_cnt |
| METHOD : GET |
| Request Query String :   * sort\_type : popular / recent 정렬 기준 params * my\_id : 로그인한 사용자의 고유 ID query * page\_cnt : 현재 보여주려는 페이지 count query |
| **Server Response JSON Declaration** |
| timelines = [{  var timeline = { if(timeline == null) → 게시글 없음  "id" : 각 게시글의 고유 ID  "wr\_date" : 각 글을 게시한 시간  "user\_id" : 글을 게시한 사용자의 ID  "wr\_category" : 글의 카테고리  "like\_cnt" : 글이 좋아요를 받은 수  "wr\_content" : 글의 내용ow.wr\_content,  "pet\_id" : 글을 등록한 동물의 고유 ID  "pet\_name" : 글을 등록한 동물의 이름  "pet\_photo\_path" : 글을 등록한 동물의 프로필 사진  "wr\_video\_path" : 글에 등록된 동영상의 URL  "wr\_video\_thumb\_path" : 글에 등록된 동영상의 썸네일의 URL  “like\_yes\_no” : 현재접속한 사용자가 해당글을 좋아요 했는지 나타내는 구분자  "reply" : [{  id : 각 댓글의 고유 ID  pet\_id : 댓글을 게시한 동물의 고유 ID  re\_date : 댓글을 게시한 시간  re\_content : 댓글의 내용  pet\_name : 댓글을 게시한 동물의 이름  pet\_photo\_path : 댓글을 게시한 동물의 프로필 사진 URL  }]  };  }] |
| //10개씩 만 뿌려주는기능 추가해야된다 (limit)  popular → sort\_type = like\_cnt;  recent → sort\_type = wr\_date;  like\_yes\_no = false → 좋아요 누르지 않음 / true → 좋아요 누름 |
|  |

|  |
| --- |
| **업무기능 : 친구’들’ 이 쓴 타임라인** |
| Request URL : /news/friends/:my\_id?:page\_cnt |
| METHOD : GET7 |
| Request Query String :   * page\_cnt : 현재 보여주려는 페이지 count //query   + my\_id : 누구의 페이지인지 알기위해 필요하다 //int |
| **Server Response JSON Declaration** |
| timelines = [{  var timeline = {  "id" : 각 게시글의 고유 ID  "wr\_date" : 각 글을 게시한 시간  "user\_id" : 글을 게시한 사용자의 ID  "wr\_category" : 글의 카테고리  "like\_cnt" : 글이 좋아요를 받은 수  "wr\_content" : 글의 내용ow.wr\_content,  "pet\_id" : 글을 등록한 동물의 고유 ID  "pet\_name" : 글을 등록한 동물의 이름  "pet\_photo\_path" : 글을 등록한 동물의 프로필 사진  "wr\_video\_path" : 글에 등록된 동영상의 URL  "wr\_video\_thumb\_path" : 글에 등록된 동영상의 썸네일의 URL  “like\_yes\_no” : 현재접속한 사용자가 해당글을 좋아요 했는지 나타내는 구분자  "reply" : [{  id : 각 댓글의 고유 ID  pet\_id : 댓글을 게시한 동물의 고유 ID  re\_date : 댓글을 게시한 시간  re\_content : 댓글의 내용  pet\_name : 댓글을 게시한 동물의 이름  pet\_photo\_path : 댓글을 게시한 동물의 프로필 사진 URL  }]  };  }] |
| 메인페이지에서 Friend 탭을 눌렀을 때 보내주는 화면이다  like\_yes\_no = false → 좋아요 누르지 않음 / true → 좋아요 누름 |

|  |
| --- |
| **업무기능 : 동물의 프로필 페이지** |
| Request URL : /profile/:pet\_id?:my\_id&:page\_cnt |
| METHOD : GET |
| Request Query String :   * page\_cnt : 현재 보여주려는 페이지 count //query * my\_id : 현재접속한 사용자의 ID //query * pet\_id : 어떤 동물의 프로필을 보여줄지에 대한 ID //params |
| **Server Response JSON Declaration** |
| pet\_page = {  "timelines" : [{  "id" : 각 게시글의 고유 ID,  "wr\_date" : 글을 게시한 시간,  "user\_id" : 글을 게시한 사용자의 고유 ID,  "wr\_category" : 게시된 글의 카테고리 명,  "like\_cnt" : 글이 좋아요 눌린 횟수,  "wr\_content" : 게시글의 내용,  "pet\_id" : 글을 게시한 동물의 ID,  "pet\_name" : 글을 게시한 동물의 이름,  "pet\_photo\_path" : 글을 게시한 동물의 프로필 사진 URL,  "wr\_video\_path" : 게시된 동영상 URL,  "wr\_video\_thumb\_path" : 게시된 동영상의 썸네일 URL  “like\_yes\_no” : 현재접속한 사용자가 해당글을 좋아요 했는지 나타내는 구분자  "reply" : [  id : 댓글의 고유 ID,  pet\_id : 댓글을 작성한 동물의 ID,  re\_date : 댓글을 작성한 시간,  re\_content : 댓글의 내용,  pet\_name : 댓글을 작성한 동물의 이름,  pet\_photo\_path : 댓글을 작성한 동물의 프로필 사진 URL  ]  }],  "pet\_info" : {  "id" : 해당 동물의 고유 ID,  “user\_id” : 동물을 소유하고 있는 사용자의 ID  "pet\_name" : 해당 동물의 이름  "birth" : 해당 동물의 생일  "pet\_speciality" : 해당 동물의 특기  "pet\_region" : 해당 동물이 사는 지역  "pet\_photo\_path" : 해당 동물의 프로필 사진 URL  "sex" : 해당 동물의 성별  "pet\_like" : 해당 동물이 좋아하는 것  "pet\_hate" : 해당 동물이 싫어하는 것  "pet\_love" : 해당 동물의 연애여부  }  }; |
| sex = 0 → 여자 / 1 → 남자  like\_yes\_no = false → 좋아요 누르지 않음 / true → 좋아요 누름 |

|  |
| --- |
| **업무기능 : 홈(사용자) 화면** |
| Request URL : /news/user/:user\_id?:my\_id&:sort\_type&:page\_cnt |
| METHOD : GET |
| Request Query String :   * page\_cnt : 현재 보여주려는 페이지 count //query * my\_id: 현재 클라이언트에서 접속한 유저의 고유 아이디 //int //query * sort\_type : popular / recent 를 구분 //like\_cnt / wr\_date //varchar(20) //query   + user\_id : 누구의 페이지인지 알기위해 필요하다 //int /params |
| **Server Response JSON Declaration** |
| user\_page = {  "timelines" : [{  "id" : 해당 글의 ID  "wr\_date" : 글이 작성된 시간  "user\_id" : 글을 작성한 사용자의 ID  "wr\_category" : 글의 카테고리  "like\_cnt" : 글이 좋아요를 받은 수  "wr\_content" : 글의 내용  "pet\_id" : 글을 작성한 동물의 ID  "pet\_name" : 글을 작성한 동물의 이름  "pet\_photo\_path" : 글을 작성한 동물의 프로필 사진 URL  "wr\_video\_path" : 글에 게시된 동영상의 URL  "wr\_video\_thumb\_path" : 글에 게시된 동영상의 썸네일의 URL  “like\_yes\_no” : my\_id 사용자가 해당 글에 좋아요를 눌렀는지 표시해주는 변수  "reply" : [{ id : 댓글의 고유 아이디  pet\_id : 댓글을 쓴 동물의 아이디  re\_date : 댓글이 게시된 날짜  re\_content : 댓글의 내용  pet\_name : 댓글을 게시한 동물의 이름  pet\_photo\_path : 댓글을 게시한 동물의 사진 URL  }]  }],  "user\_info" : {  "pet\_info" : [{ id : 사용자가 보유한 동물의 고유 아이디 값  pet\_photo\_path : 동물의 사진 URL  pet\_name : 동물의 이름  }],  "wr\_cnt" : 사용자가 쓴 글의 개수를 카운트  "user\_like\_cnt" : 사용자의 글이 좋아요를 받은 개수 카운트  "friends\_cnt" : 사용자의 친구 수  “nickname” : 해당 사용자의 (집주인) 닉네임  "friend\_check" : 남이냐 / 친구냐 / 친구요청중이냐 / 상대방이 나에게 친구신청을 했느냐를 알려주는 값  //남 : 0  //친구 : 1  //내가 친구요청중 : 2  //상대방이 나에게 친구신청을 했느냐 : 3  //자기 자신의 페이지 : 4  }  };  total\_page\_cnt : 전체글의 페이지가 몇페이지 인지 |
| like\_yes\_no = false → 좋아요 누르지 않음 / true → 좋아요 누름 |

|  |
| --- |
| **업무기능 : 홈(사용자) 화면 - Favorite** |
| Request URL : /news/user/favorite/:user\_id?:my\_id |
| METHOD : GET |
| Request Query String :   * page\_cnt : 현재 보여주려는 페이지 count * my\_id: 현재 클라이언트에서 접속한 유저의 고유 아이디 //쿼리로   + user\_id : 누구의 페이지인지 알기위해 필요하다 //int |
| **Server Response JSON Declaration** |
| user\_page = {  "timelines" : [{  "id" : row.id,  "wr\_date" : row.wr\_date,  "user\_id" : row.user\_id,  "wr\_category" : row.wr\_category,  "like\_cnt" : row.like\_cnt,  "wr\_content" : row.wr\_content,  "pet\_id" : row.pet\_id,  "pet\_name" : row.pet\_name,  "pet\_photo\_path" : row.pet\_photo\_path,  "wr\_video\_path" : row.wr\_video\_path,  "wr\_video\_thumb\_path" : row.wr\_video\_thumb\_path,  “like\_yes\_no” : my\_id 사용자가 해당 글에 좋아요를 눌렀는지 표시해주는 변수  "reply" : [{ id : 댓글의 고유 아이디  pet\_id : 댓글을 쓴 동물의 아이디  re\_date : 댓글이 게시된 날짜  re\_content : 댓글의 내용  pet\_name : 댓글을 게시한 동물의 이름  pet\_photo\_path : 댓글을 게시한 동물의 사진 URL  }]  }],  "user\_info" : {  "pet\_info" : [{ id : 사용자가 보유한 동물의 고유 아이디 값  pet\_photo\_path : 동물의 사진 URL  pet\_name : 동물의 이름  }],  "wr\_cnt" : 0, // 사용자가 쓴 글의 개수를 카운트  "user\_like\_cnt" : 0, // 사용자의 글이 좋아요를 받은 개수 카운트  "friends\_cnt" : 0, // 사용자의 친구 수  "friend\_check" : 0 // 남이냐 / 친구냐 / 친구요청중이냐 / 상대방이 나에게 친구신청을 했느냐를 알려주는 값  //남 : 0  //친구 : 1  //내가 친구요청중 : 2  //상대방이 나에게 친구신청을 했느냐 : 3  //자기 자신의 페이지 : 4  }  };  total\_page\_cnt : 전체글의 페이지가 몇페이지 인지 |
| Favorite 은 내가 좋아요 한 글을 최신순으로 보여준다.  like\_yes\_no = false → 좋아요 누르지 않음 / true → 좋아요 누름 |

|  |
| --- |
| **업무기능 : 카테고리 별 타임라인** |
| Request URL : /news/category/:category?my\_id= &sort\_type= &page\_cnt= |
| METHOD : GET |
| Request Query String :   * page\_cnt : 현재 보여주려는 페이지 count * sort\_type : popular / recent 정렬 기준   + my\_id : 접속한 사용자의 ID   + category : 카테고리 별 기준 //우선은 char(1) → 수정가능     - dog = 개     - cat = 고양이     - bug = 곤충     - bird = 새     - reptilia = 파충류     - fish = 물고기     - mammal = 포유류     - amphibian = 양서류 |
| **Server Response JSON Declaration** |
| timelines = [{  var timeline = {  "id" : 각 게시글의 고유 ID  "wr\_date" : 각 글을 게시한 시간  "user\_id" : 글을 게시한 사용자의 ID  "wr\_category" : 글의 카테고리  "like\_cnt" : 글이 좋아요를 받은 수  "wr\_content" : 글의 내용ow.wr\_content,  "pet\_id" : 글을 등록한 동물의 고유 ID  "pet\_name" : 글을 등록한 동물의 이름  "pet\_photo\_path" : 글을 등록한 동물의 프로필 사진  "wr\_video\_path" : 글에 등록된 동영상의 URL  "wr\_video\_thumb\_path" : 글에 등록된 동영상의 썸네일의 URL  “like\_yes\_no" : my\_id 사용자가 해당 글에 좋아요를 눌렀는지 표시해주는 변수  "reply" : [{  id : 각 댓글의 고유 ID  pet\_id : 댓글을 게시한 동물의 고유 ID  re\_date : 댓글을 게시한 시간  re\_content : 댓글의 내용  pet\_name : 댓글을 게시한 동물의 이름  pet\_photo\_path : 댓글을 게시한 동물의 프로필 사진 URL  }]  };  }]  total\_page\_cnt : 전체글의 페이지가 몇페이지 인지 |
| 각 category 별로 popular / recent 두가지 탭이 있다.  popular → sort\_type = like\_cnt;  recent → sort\_type = wr\_date;  like\_yes\_no = false → 좋아요 누르지 않음 / true → 좋아요 누름 |

|  |
| --- |
| **업무기능 : 타임라인에 글 올리기** |
| Request URL : /news/create |
| METHOD : POST |
| Request Query String :   * pet\_id : 글을 게시한 동물의 ID //int //not null * wr\_category : 글의 카테고리 //char(1) //not null * 동영상 파일 upload //not null * wr\_public : 글의 공개범위 //bool //not null   + 0 → 전체공개 1 → 친구에게만 * wr\_content : 글의 내용 //varchar(200) //null |
| **Server Response JSON Declaration** |
| {  ‘result’ : ‘success’  } |

|  |
| --- |
| **업무기능 : 친구목록 표시** |
| Request URL : /list/friends/:my\_id |
| METHOD : GET |
| Request Query String :   * my\_id : 사용자의 친구를 찾기위한 사용자 ID |
| **Server Response JSON Declaration** |
| friendsList: [{  ‘user\_id’ : 5,  ‘nickname’ : name,  ‘pet\_photos’ : [  ‘pet\_photo\_path’ : ‘www.pet\_photo\_path.png’  ]  }] |

|  |
| --- |
| **업무기능 : 반려동물의 프로필 등록** |
| Request URL : /pet/profile/create |
| METHOD : POST |
| Request Query String :   * user\_id : 동물을 소유하는 사용자의 id //not null //int * pet\_name : 반려동물의 이름 //not null //varchar(20) * birth : 반려동물이 생일 //not null //char(8) * sex : 반려동물의 성별 //not null //boo * pet\_region : 사는 지역 //null //varchar(20) * pet\_speciality : 반려동물의 특기 //null //varchar(20) * pet\_like : 반려동물이 좋아하는 것 //null //varchar(20) * pet\_hate : 반려동물이 싫어하는 것 //null //varchar(20) * pet\_love : 반려동물의 연애 유무 //null //bool * 동물의 프로필사진파일 upload //null |
| **Server Response JSON Declaration** |
| pet\_info : {  “pet\_id”:  “pet\_name”  “pet\_photo\_path” :  } |
| **socket IO 해야돼** |

|  |
| --- |
| **업무기능 : 반려동물의 프로필 수정** |
| Request URL : /profile/pet/update |
| METHOD : POST |
| Request Query String :  not null   * isMaintain : 프로필 수정하면서 사진을 새로 등록하지 않으면서 기존의 사진을 유지할지 / 없앨지 구분자   + 유지 → 1   + 없앨 지 → 0 * pet\_id : 반려동물의 id * pet\_name : 반려동물의 이름 //not null //varchar(20) * birth : 반려동물이 생일 //not null //char(8) * sex : 반려동물의 성별 //not null //boo   + 남자 → 1 여자 → 0   null   * pet\_region : 사는 지역 //null //varchar(20) * pet\_speciality : 반려동물의 특기 //null //varchar(20) * pet\_like : 반려동물이 좋아하는 것 //null //varchar(20) * pet\_hate : 반려동물이 싫어하는 것 //null //varchar(20) * pet\_love : 반려동물의 연애 유무 //null //bool * 동물의 프로필사진파일 upload //null |
| **Server Response JSON Declaration** |
| pet\_info : {  pet\_id :  pet\_name :  pet\_photo\_path  } |
|  |

|  |
| --- |
| **업무기능 : 게시된 글을 수정** |
| Request URL : /news/update |
| METHOD : POST |
| Request Query String :   * timeline\_id : 수정할 글의 id * wr\_content : 글의 내용 * wr\_category : 수정할 카테고리 * wr\_public : 변경할 공개여부 |
| **Server Response JSON Declaration** |
| {  ‘result’ : ‘success’  } |
| 글 수정할 때 동물도 변경가능 |

|  |
| --- |
| **업무기능 : 댓글작성** |
| Request URL : /reply/create |
| METHOD : POST |
| Request Query String :   * timeline\_id : 댓글을 작성할 글의 ID * pet\_id : 댓글을 작성한 동물의 ID * re\_content : 댓글의 내용 |
| **Server Response JSON Declaration** |
| {  message : success  } |
|  |

|  |
| --- |
| **업무기능 : 댓글삭제** |
| Request URL : /reply/delete |
| METHOD : POST |
| Request Query String :   * reply\_id : 댓글의 ID |
| **Server Response JSON Declaration** |
| {  ‘result’ : ‘success’  } |
|  |

|  |
| --- |
| **업무기능 : 게시된 글을 좋아요** |
| Request URL : /news/like |
| METHOD : POST |
| Request Query String :   * timeline\_id : 좋아요를 누른 글의 ID * user\_id : 좋아요를 누를 사용자의 ID |
| **Server Response JSON Declaration** |
| {  "message": "like\_timeline complete"  } |
| 좋아요 눌렀을 때는 다시 redirection 해주는 것보다 클라이언트쪽에서 화면에보이는걸로 like\_cnt 값 하나 올려서 표시해주고 좋아요 누른 아이콘으로 바꿔주고  나한테 query string으로 보내주면 내가 디비에 적용하고나서  다음에 화면 다시 요청할때는 내가 가지고있는 정보로 뿌려주는식으로 하는게 어때  redirection 하면 타임라인 맨위페이지로 올라갈거 같아 |

|  |
| --- |
| **업무기능 : 게시된 글을 신고** |
| Request URL : /news/report |
| METHOD : POST |
| Request Query String :   * user\_id : 신고한 사용자의 ID * timeline\_id : 신고할 글의 ID |
| **Server Response JSON Declaration** |
| {  ‘result’ : ‘success’  } |
|  |



|  |
| --- |
| **업무기능 : 사용자가 로그인** |
| Request URL : /login |
| METHOD : POST |
| Request Query String :   * email : 로그인 한 사용자의 email * passwd : 로그인한 사용자의 비밀번호 |
| **Server Response JSON Declaration** |
| user\_info : {  “my\_id” : int,  “nickname” : varchar(10),  “update authkey” : varchar(40)  “pet\_info” : [{  id : 동물의 고유 ID  pet\_name : 동물의 이름  pet\_photo\_path : 동물의 사진  }]  } |
|  |

|  |
| --- |
| **업무기능 : 회원가입** |
| Request URL : /signup |
| METHOD : POST |
| Request Query String :   * email : 사용자가 아이디로 사용할 email * passwd : 사용자가 사용할 비밀번호 * nickname : 사용자가 사용할 닉네임 |
| **Server Response JSON Declaration** |
| {  ‘result’ : ‘success’  } |
|  |

|  |
| --- |
| **업무기능 : 친구관계 변경** |
| Request URL : /friend/update |
| METHOD : POST |
| Request Query String :   * my\_id : 현재 접속한 사용의 ID * user\_id : 친구 관계를 맺으려는 사용자의 ID * accept\_reject : 친구신청을 받은 상태에서 수락을 할지 거절을 할지 정해주는 변수   + 0 → 거절   + 1 → 수락 |
| **Server Response JSON Declaration** |
| {  “message”’ : “Success”  } |
|  |

|  |
| --- |
| **업무기능 : 게시글 삭제** |
| Request URL : /news/remove |
| METHOD : POST |
| Request Query String :   * timeline\_id : 사용자가 지우려는 글의 ID * my\_id : 사용자의 ID |
| **Server Response JSON Declaration** |
| {  ‘result’ : ‘success’  } |
|  |

|  |
| --- |
| **업무기능 : 댓글 페이지** |
| Request URL : /reply?page\_cnt= &timeline\_id= |
| METHOD : GET |
| Request Query String :   * timeline\_id : 사용자가 지우려는 글의 ID * page\_cnt : 댓글의 페이지 |
| **Server Response JSON Declaration** |
| replies[{  id : 댓글의 id  pet\_id :  re\_date :  re\_content :  pet\_name :  pet\_photo\_path :  }] |



|  |
| --- |
| **업무기능 : 동물 삭제** |
| Request URL : /profile/pet/remove |
| METHOD : POST |
| Request Query String :   * my\_id : 해당 사용자의 ID //body * pet\_id : 사용자가 지우려는 동물의 ID //body |
| **Server Response JSON Declaration** |
| message : Remove pet profile success |