**Documentation for Project toDoctoDoc**

* 모든 영문자는 대소문자에 유의한다.
* 리턴값으로 던져지는 객체에 대해서 잘 모르겠으면 콘솔에 로그 찍어보고 JSON 트리 구조를 파악하길 바람.
* <중요!!!> 복잡한 객체, 즉 스키마가 정의된 객체는 스키마 선언 보고 그에 맞는 객체화를 시킬 것!!!!!! 그냥 트리 구조 그대로 활용하지 말고 객체화를 꼭 꼭 시키자.
* 보는 법
  + 요청 인자: 이러이러한 객체를 백엔드로 “넘겨 줘야 함”
  + 반환 인자: 이러이러한 객체가 백엔드에서 “반환 됨” (정상적인 경우라면)
* 에러시 응답 코드는 401로 통일 해놨으니 아래 표준 응답 객체의 result 칸을 보고 어떤 오류인지 판단하여 분기 처리 및 디스플레이 하자. 각종 오류에 따라 상세한 result를 보여주니까 꼭 꼭 참고!

1. 표준 JSON 응답 객체
   1. 텍스트, 스크린샷, 폰트이(가) 표시된 사진

      자동 생성된 설명의 규칙을 따른다.  
      즉 {  
       error: Boolean,  
       result: String,  
       content: Object,  
       }  
      의 형태를 따른다. 그런데 Content에 바로 String이 들어갈 수도 있고 Object로 감쌀 수도 있고
2. REST API :: 환자가 사용하는 API
   1. /mapp/login
      1. POST
      2. 요청 인자: userid,password를 body의 인자로 받음.
      3. 반환 인자: {Access Token, Refresh Token}을 content에 담아 전송
      4. 로그인 하는 기능을 제공함.
   2. /mapp/logout
      1. GET
      2. 로그아웃 하는 기능을 제공함.
   3. /mapp/register
      1. POST
      2. 요청 인자: id,password,password2,nickname,postcode,address,extraaddress,email을 body의 인자로 받음.
      3. 회원가입 하는 기능을 제공함.
      4. 반환 인자: {Access Token, Refresh Token}을 content에 담아 전송
   4. /mapp/query
      1. POST
      2. 로그인이 되어 있어야 함.
      3. 요청 인자: input을 body의 인자로 받음.
      4. 반환 인자: content에 pageContent 객체를 담아서 전송, pageContent 객체의 형태:  
         텍스트, 폰트, 스크린샷, 그래픽이(가) 표시된 사진

         자동 생성된 설명  
         title이 query한 내용의 제목,  
         context가 query한 내용에 대한 답변 내용임.
   5. /mapp/upload
      1. POST
      2. 로그인이 되어 있어야 함
      3. 요청 인자: title, content, content\_additional, tag를 body의 인자로 받음.
      4. 반환 인자: content에 Post 객체를 담아서 전송, Post 객체의 형태:  
         텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

         자동 생성된 설명  
         여기에 고유한 \_id 속성까지 존재한다.
   6. /mapp/view/:id
      1. GET
      2. 로그인이 되어 있어야 함
      3. 요청 인자: id 매개변수로 Post의 \_id 속성을 받음. (예시: /mapp/view/12345)
      4. 반환 인자: content에 pageContent 객체를 담아 반환함:  
         텍스트, 스크린샷, 폰트이(가) 표시된 사진

         자동 생성된 설명  
         isOwner로 편집/삭제 기능을 활성화 시킬 수 있음.
   7. /mapp/edit/:id
      1. PATCH
      2. 로그인이 되어 있어야 함
      3. 로그인한 유저와 게시물의 작성자 id가 맞지 않으면 오류를 반환함
      4. 요청 인자: id 매개변수로 Post의 \_id 속성을 받음
      5. 게시물을 수정하는 역할을 함(따라서 PATCH후 view/로 다시 리다이렉션 해 줘야 함
   8. /mapp/delete/:id
      1. DELETE
      2. 로그인이 되어 있어야 함
      3. 로그인한 유저와 게시물의 작성자 id가 맞지 않으면 오류를 반환함
      4. 요청 인자: id 매개변수로 Post의 \_id 속성을 받음
      5. 게시물을 삭제하는 역할을 함(따라서 DELETE후 myPosts/로 다시 리다이렉션 해 줘야 함)
   9. /mapp/myPosts
      1. GET
      2. 로그인이 되어 있어야 함
      3. 반환 인자: 로그인한 유저가 작성한 전체 게시물을 Posts[ ]의 형태로 리턴함! 배열임에 주의해야 함
   10. /mapp/tagSearch/:tag
       1. GET
       2. 로그인이 되어 있어야 함
       3. 요청 인자: tag 매개변수로 게시물의 태그를 검색하는 역할을 수행함.
       4. 반환 인자: 해당 tag를 가진 게시물을 Posts[ ]의 형태로 리턴함! 배열임에 주의해야 함
   11. /mapp/graphBoard
       1. GET
       2. 로그인이 되어 있어야 함
       3. 그래프보드를 렌더링 하기 위한 객체를 반환하는데, 아래와 같은 형태임  
            
          tagList는 키:값 쌍을 가진 맵 형 자료구조임.  
          값은 그 태그를 가진 게시물의 수를 의미함.  
          즉 값이 클수록 큰 원을 가지게 됨.  
          tagGraph는 키:키 쌍을 가진 자료구조임.  
          중복된 키:키 쌍은(즉, A:B와 B:A)존재하지 않음.  
          서로 연결된 태그를 표시해 주기 위한 자료구조임.  
          이해 안되면 디스코드 남기면 바로 답장 해 줌
   12. /mapp/info
       1. GET
       2. 로그인이 되어 있어야 함
       3. 요청 인자: 없음
       4. 반환 인자: userinfo 객체를 반환하는데, MongoDB Projection이 되어 있어서 일부 필드만 전송됨. 전송되는 필드는 id usernick email address limits isPremium임.  
          userinfo 객체의 스키마는 아래의 스키마를 따르지만, 전송되는 필드가 한정되어 있음을 유념하길 바람. 또 address는 그 자체로 주소가 아니라 AddressSchema로 구성되어 있기에 이 역시 같이 볼 수 있도록 첨부함. 예를 들어 주소에 접근하려면 userinfo.address.address를 해야 한다는 뜻임. limits도 객체를 전송함  
          텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

          자동 생성된 설명  
          텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

          자동 생성된 설명
   13. /mapp/editInfo
       1. PATCH
       2. 로그인이 되어 있어야 함
       3. 요청 인자: {usernick, email, postcode, address, detailAddress, extraAddress, password, password2}를 body에 담아서 전송한다.
       4. 반환 인자: 성공적으로 작동(200)했을 경우 새 newUserInfo 객체를 반환한다. 이 객체의 특성은 /mapp/info에서 봤던 것과 동일하다.