# 기능 정의

* Express.js 및 거기에 따르는 라우터 객체를 사용하여 Back-end 프로그래밍을 진행하라.
* DB를 사용하여 프로그래밍 진행
* (추가 1) Ejs를 사용하여 템플릿으로 출력하라.
* (추가 2) 가상의 사용자를 두 명이상 추가하여 접근 가능하게 하라.
* 유저 검색기능을 통하여 접근 및 추가 가능하게 하였음.
* (추가 3) 문서 첨부의 형태로 블로그의 게시글을 작성 가능하게 하라.
* 파일 첨부 및 textarea를 통하여 작성 가능하게 함.
* (추가 4) 다른 사용자를 친구 추가하면, 블로그 시작 페이지에 가장 최근에 추가한 게시글 3개를 보여줌.
* 시간별로 정렬하여 3개를 보여줌.
* (추가 5) 회원가입과 세션을 구현.
* (추가 6) 전체 시스템은 각종 서버에서 가동.
* (부가기능) 각종 유저의 블로그 작성 시간을 시각화하여 보여줌.
* (부가기능 2) 유저의 랜덤 추천

# 스크립트 설명

* 기본 :

|  |
| --- |
| # 변수의 선언   * App : 서버를 열어야 할 express 객체 * Db : MySQL과 연결하기 위한 “await”기반의 동기형 객체 * db\_common : MySQL과 연결하기 위한 “.then”기반의 비동기형 객체 * mongodb : mongodb와 연결하기 위한 객체. * DATABASE\_URL : Mysql과 연결하기 위한 연결스트링(.env에 정의) * MONGODB\_URL : MongoDB와 연결하기 위한 MongoDB연결 스트링(.env에 정의)   # 문제의 해결 및 일고리즘  app.use(express.json());//json 응답을 사용가능하게 하고  app.use(express.static('static')); // static파일을 쓸 수 있게 활성화 하며,.  app.use(BodyParser.urlencoded({extended:true}));// Body의 사용을 가능하게 하기 위해 bodyparser를 use처리. |
| * MySQL의 경우, 연결스트링(DATABASE\_URL)을 기반으로 createPool을 통해 Connection pool을 만들어 DB와 연결을 시도함. |
| * Mongodb의 경우 연결스트링(MONGODB\_URL)을 기반으로 연결할 수 있게 해주는 객체를 만들어 필요시마다 await mongodb.connect()을 통해 연결을 열고, 쿼리 실행 후 await mongodb.close()를 통해 닫는다. |
| * 이미지등의 파일은 multer를 통하여 저장하고, 불러올때는 static이 아니라면 ‘/file/download’를 통해서 파일을 불러오게 함. |
| * 라우터 객체는 ‘router.js’에 보관. * const appRouter = Router(); 를 통해서 생성하고, * get post put delete등을 사용해 각종 엔드포인트를 붙인 뒤, * export default appRouter;로 보내게 함. * 그후 app.use(‘/’,appRoute)로 라우터 활성화 |

* 추가기능 1 ((추가 1) Ejs를 사용하여 템플릿으로 출력)

|  |
| --- |
| app.set('view engine','ejs'); 으로 EJS를 활성화하고,  app.set('views', 'view'); 으로 view폴더에서 ejs파일을 가져오게 한다.  (그 외 res.render를 통해서 보여줌) |

* 추가기능 2 ((추가 2) 가상의 사용자를 두 명이상 추가하여 접근 가능하게 하라.)

|  |
| --- |
| # 변수의 선언   * 유저의 Table은 이하와 같이 정의한다.   model User {    // 사용자 정보    id                 Int         @id @default(autoincrement()) // 기본 키로 지정    loginId            String // 로그인시 사용할 Id    password           String // 로그인시 사용할 Password, 해쉬화되어 보관됨.    nickname           String // 타 사용자에게 보여질 닉네임.    email              String // 이메일    createdAt          DateTime       @default(now()) // 유저 생성시간    passwordErrorCount Int            @default(0) // 비밀번호 틀린 횟수    posts              Post[] // 작성한 포스트들    schedule           UserSchedule[] // 유저의 스케쥴(미구현)    mainDesc           String @db.Text// 설명 (유저의 메인 프로필에 달린 설명)    mainProfilePicture        String? // 유저의 프로필 사진 경로    mainTitlePicture        String? // 유저의 헤더 사진 경로    @@index([id])    @@index([loginId,password]) // 로그인시 사용    @@map("user") // 실제로는 user에 등록.  }  # 알고리즘   * 유저의 조회 :   유저는 회원가입으로 user에 등록된다.(이는 추가기능 5에서 다룸)  이후, /page/:userid 라우터로 get요청을 보내면 DB에서 유저 프로필과 유저가 작성한포스터들(추가기능 3에서 다룸)을 불러온다. (loadUserPage)   * SELECT id,nickname,mainDesc,mainTitlePicture,mainProfilePicture FROM user WHERE id=?   이후 불러온 정보를 ejs로 로딩해서 보여준다.  # 함수의 사용   * (loadPostModel : SQL에서 불러오는 함수) * loadUserPage : 페이지를 불러오는 라우터 함수 |
| 알고리즘 |

텍스트, 도표, 스크린샷, 평행이(가) 표시된 사진

자동 생성된 설명

* 계속
* 추가기능 3 ((추가 3) 문서 첨부의 형태로 블로그의 게시글을 작성 가능하게 하라.)

|  |
| --- |
| # 변수의 선언   * 게시글은 이하의 Table로 보존한다.   model Post { // 유저의 게시글  id Int @id @default(autoincrement()) // 게시글 ID  user User @relation(references: [id], fields: [userId]) // 유저의 정보  userId Int // 작성자ID  title String // 제목  file String? // 받을수있는파일  desc String? // 글내용이 담긴 mongoDB의 objectId  createdAt DateTime @default(now()) //게시한 시간(DB기준시간)  PostPicture PostPicture[] // Prisma에서 사용하는 관계형. Post에는 PostPicture가 여러 개 있음을 의미.  @@index([userId]) // userID에 index걸어서 검색을 빠르게  }   * 또한, 게시글의 사진은 이하의 형태로 보존한다.   model PostPicture { // 게시글의 사진 정보이다.  id Int @id @default(autoincrement()) //게시글 사진의 ID  post Post? @relation(references: [id], fields: [postId]) // 포스트와의 관계 postId로 이어짐.  userId Int // 작성자ID  postId Int? // 연결된 post의 id  pictureId Int? // 사진의 Id  createdAt DateTime @default(now()) //올라간 시간  path String //경로  @@index([postId]) // PostId를 LEFT로 해서 JOIN할경우 Key.  @@index([pictureId])  }   * MongoDB에는 이하와 같이 보존한다.   {  \_id //mongoID가 부여하는 ID  ...etc, //넘겨받은 기타 정보 (위치정보나, etc…)  context:text, // 글의 본문  picture:{ path // 사진 경로들}  };   * Mongo와 mysql에 따로 저장하는 이유는, “DB의 유연성”으로 인하여 다양한 정보를 담는 가능한지 아닌지에 대한 여부와, 글의 길이를 고려하여 텍스트는 따로 분리하는 것이 검색등의 측면에서 이득이라 판단하였기 때문이다.   # 함수의 사용   * uploadPostModel : DB측에서 포스터를 등록하는 함수 * uploadPost : 포스터 등록 리퀘스트를 받는 라우터 함수 * exceptionHandlerOnUpload : 파일 업로드중 에러 발생시 대응하는 미들웨어 * posterUploader : 파일을 업로드하는 미들웨어 |

텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명# 알고리즘 및 순서도

* 비고 : posterUploader를 통해 업로드된 파일의 경로는 request의 filename(텍스트) / imagefile(이미지)를 통하여 다음 미들웨어인 uploadPost로 전달됨.
* 텍스트, 도표, 스크린샷, 폰트이(가) 표시된 사진

  자동 생성된 설명추가기능 4((추가 4) 다른 사용자를 친구 추가하면, 블로그 시작 페이지에 가장 최근에 추가한 게시글 3개를 보여줌.)

|  |
| --- |
| # 변수의 선언  model Social{ // 관계도  followerId Int // 팔로잉 하는 사람의 UserID  targetId Int // 팔로우 받는 사람  @@id([followerId,targetId]) // 둘의 관계가 고유함으로 설정.  @@index([followerId]) // 검색은 followerId가 잦은 바, index는 이것으로 설정.  }   * User는 추가기능 2에서 갈음 * Post는 추가기능 3에서 갈음   # 함수 사용   * loadSocialPage : 조건에 맞는 게시글들을 가져오는 함수 * displayMainPage : 로그인 되었는지를 판단해서, 로그인 되어있다면 친구들의 동향을, 그렇지 않다면 일반 메인페이지를 보여줌. |

텍스트, 도표, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

* 텍스트, 도표, 폰트, 라인이(가) 표시된 사진

  자동 생성된 설명텍스트, 도표, 폰트, 평행이(가) 표시된 사진

  자동 생성된 설명추가기능 5 ((추가 5) 회원가입과 세션을 구현.)

|  |
| --- |
| # 변수의 선언  model Session { //세션 Table. 세션 저장소는 MYSQL로 지정하였다.  sid String @id  expire String  data String}   * 회원은 2번 추가기능 table로 갈음.   # 함수의 사용   * tryToLogin:로그인을 위한 페이지 로딩 * tryToRegister : 회원가입을 위한 페이지 로딩 * userRegister : 유저의 회원가입을 받는 라우터 함수 * userLogin : user DB에 hash화 된 PW와 유저가 제출한 ID가 같은 행이 있는지 확인하고, 있다면 그 행의 id를 세션에 등록하는 함수. * userLogout : 유저의 로그아웃을 받는 라우터 함수 * isNotLoginState : 유저가 로그인 상태가 아닌지 검증하는 미들웨어 * isLoginState : 유저가 로그인 상태인지 검증하는 미들웨어 * 세션에 passport.js의 값이 등록되어 있다면 로그인 된 상태로 판단하고, 그것이 아니라면 로그인되지 않은 상태로 판단한다. * passport.authenticate : 유저를 로그인 시키는 미들웨어 * PassportJs로 구현하였으며, 전략은 세션에 자체서버에서 발급한 ID를 저장하여 필요한때 호출하는 local 전략을 사용하였다. |

* 추가기능 6 ((추가 6) 전체 시스템은 각종 서버에서 가동.)

|  |
| --- |
| * 이하의 업체들을 통하여 서버를 호스팅하였음. * 서버 : app.cloudtype.io * MySQL : planetscale.com * MongoDB : mongodb.com * SSL인증 등은 자동으로 진행함으로, https접속 대응도 준비되어있음. |

* 추가기능 7((부가기능) 각종 유저의 게시글 작성 시간을 시각화하여 보여줌.)

|  |
| --- |
| # 자료구조   * 추가기능 3으로 갈음   # 함수의 사용   * loadTimeStat : 지정된 날의 게시글 정보들을 가져옴 * searchingPostByDate: DB에서 지정된 날의 게시글을 가져옴 * meta 를 통해서 게시글 작성자를 보여줌 * timetable.js : 클라이언트측에서 이벤트를 시각화 해줌   # 알고리즘   * 유저로부터 GET 요청인 `/datestat` 을 받아온다. 이때, 날짜는 date로, client간의 시차는 delta로 받아온다. * 시간을 보정하고, 그날의 게시글들을 가져오기위해 Date를 설정한다. * 원하는 날짜의 게시글을   SELECT A.\*,B.nickname FROM Post AS A INNER JOIN user AS B ON A.userId=B.id WHERE A.userId != (자기 자신의 ID) AND A.createdAt BETWEEN (시작 시간) AND (끝시간) ORDER BY A.createdAt을 통하여 가져온다.   * 이후 Ejs에는 가져온 포스트들을 렌더릴하며 스크립트로 추가, * 이후 timetable.js로 보여준다. |

* 추가기능 8((부가기능 2) 유저의 랜덤 추천)

|  |
| --- |
| # 자료구조   * 추가기능 2로 갈음   # 함수의 사용   * searchingPeopleRouter : 유저 검색을 받는 라우터 * searchingRandomPeople : 랜덤 유저를 검색하는 함수   # 알고리즘   * 이하의 쿼리를 실행해 랜덤유저를 뽑아온다.   SELECT mainDesc,nickname,mainProfilePicture,mainTitlePicture,id FROM user WHERE id != (자신의 ID) AND id NOT IN (SELECT targetId FROM Social WHERE followerId=(자신의 ID)) ORDER BY rand() Limit 불러오는 사람 수   * 이후 검색결과를 res.render('search.ejs',{data,randomData})를 통해 ejs로 렌더링한다. |

* 추가기능 9 : 날짜로 포스터 검색

|  |
| --- |
| # 자료구조   * 추가기능 3으로 갈음   # 함수의 사용   * loadTimeTableMap : 특정 날짜를 입력받으면 그 게시글들을 출력해주는 라우터 * searchingPostByDate : 특정 날짜로 게시글을 검색해주는 함수 * loadPictureList : 게시글 ID로부터 사진정보를 받아오는 함수.   # 알고리즘   * 유저로부터 end(유저가 찾는 마지막 날짜) , start(유저가 찾는 시작하는 날짜), delta(유저와 UTC간의 시차) 를 입력받아온다. * 시차를 계산(UTC 기준화)하고, 이후 이하의 쿼리로 자신의 포스트가 아닌 포스트를 검색해온다.   SELECT A.\*,B.nickname FROM Post AS A INNER JOIN user AS B ON A.userId=B.id WHERE A.userId != (자기ID) AND A.createdAt BETWEEN (시작일) AND (종료일) ORDER BY A.createdAt   * 이후, 거기에 있는 사진정보를 불러오고, res.render('timetable.ejs',{ret})를 통해 렌더링하여 보여준다. |

(MySql 구조)

텍스트, 도표, 평면도, 평행이(가) 표시된 사진

자동 생성된 설명

# 실행결과

스크린샷, 텍스트, 일렉트릭 블루, 블루이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명<데모영상 첨부>

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 웹 페이지, 웹사이트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 웹 페이지, 웹사이트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 직사각형, 디자인이(가) 표시된 사진

자동 생성된 설명