



실전 프로젝트1



11

TodoList SQLite Version (2)

잡플래닛-한동대학교

- 학교 이메일 계정으로 잡플래닛 가입하면 무료로 취업 정보 무제한 열람
- 기업리뷰, 연봉정보, 면접후기, 채용정보, 취업리포트센터 무제한

<https://www.jobplanet.co.kr/>

www.jobplanet.co.kr

기업 입지원의 솔직한 기업 속 이야기

취업!? 물어볼 선배가 필요해? JOB플래닛에게 물어봐!

우리학교는 잡플래닛 제휴 대학교!



이용방법

학교 이메일 계정으로
잡플래닛에 회원가입한다.



문의사항은 학교 취업센터나 잡플래닛 (haveajob@jobplanet.co.kr)으로!

혜택 1

PC/모바일 모두 가능

잡플래닛에 로그인하고
기업리뷰, 연봉정보, 면접후기 보기



혜택 2

PC만 가능

메인 페이지에서 취업리포트센터
버너 클릭하고 리포트 다운받기



채용 공고 둘러보기 1

전형절차

서류 접수 > 넷마블 테스트 > 1차 면접 > 2차 면접 > 채용 검진 > 최종합격 및

- 모집 직무에 따라 전형 단계가 다를 수 있으며, 대상자에게 개별 안내 예정입니다.
- 소프트웨어 개발, 빅데이터, 클라이언트, 서버 직군은 서류 접수 이후 온라인 코딩테스트가 진행됩니다.
- 서류발표 후 온라인 인성검사가 진행될 예정입니다.

Jobplanet

넷마블 자소서 작성 꿀팁 대방출

[JP요원의 취업 tip] 게임 산업에 대한 깊은 이해와 관심을 보여라

상세모집요강

소프트웨어 개발

[클라이언트 개발]

- 글로벌 게임 퍼블리싱 플랫폼 서비스 개발
- 다양한 게임 엔진/플랫폼을 지원하는 SDK 개발

[서버 개발]

- 게임 플랫폼/웹 서비스 Back-end 서버 개발
- 대용량 트래픽 분산 처리 아키텍처 설계/구축

[웹 프론트엔드 개발]

- 게임 플랫폼/웹 서비스 Front-end 서버 개발
- 다양한 서비스의 Web App 개발 및 Front-end 서버 구축

[DBA]

- 플랫폼 및 게임 RDB/NoSQL 구축, 운영 및 성능 최적화
- RDB/NoSQL 운영 자동화 시스템 구축

[성능 엔지니어]

- 시스템 아키텍처 분석을 통한 서버 컨설팅
- 모바일 게임 유저 플레이 패턴 분석 후 자동 플레이봇 설계/구현
(C/C++, C#, Java 활용)
- 서버 성능 테스트 수행을 통한 대용량 서비스 및 빅데이터 시스템의 처리량 산정, 운영 계획 수립

자격요건

- 프로그래밍 기초 지식
- 프로그래밍 능력
- 논리적, 분석적 사고력
- 원활한 협업 및 커뮤니케이션 역량

우대사항

- 컴퓨터 공학/소프트웨어 관련 전공
- 알고리즘 대회 출전 및 입상 경험
- 게임 개발 경험
- 본인이 직접 작성한 github 공개 코드

채용 공고 둘러보기2

전형절차 4학년 여름방학때 채용준비 완료 해야함

9/28(화) - 10/14(수) 17:00 까지

지원서 접수

10월 말

서류전형 합격자 발표

11월 초

사전과제 제출

11월 초 - 중순

면접 전형

11월 말

최종 합격자 발표

Info
모집분야

Server

서비스를 안정적으로 운영할 수 있는 백엔드 서비스를 개발합니다.
> JAVA, Go, Spring Framework

Client(모바일)

모바일 환경에 최적화된 어플리케이션을 개발합니다.
> Android(Java,Kotlin) / iOS(Objective-C,Swift)

Client(Window)

원활한 게임 플레이를 위한 PC 클라이언트를 개발합니다.
> C,C++, 네트워크 프로그래밍

Client(Web)

스토브에서 제공하는 다양한 웹 서비스를 개발합니다.
> HTML, Javascript, TypeScript, CSS,
React, Vue.js(nuxt.js) 등 framework



Singleton pattern

- 인스턴스를 1개만 생성해서 사용하는 패턴
- 단 한번의 객체 생성으로 메모리에 로드되어 재사용이 가능하므로 메모리 낭비를 방지
- 여러 스레드(thread)가 동시에 인스턴스를 공유하여 사용
- DB 접속을 위한 Connection에서 효율적인 사용 가능
- **문제점**
 - 객체의 역할이 복잡한 경우 다른 객체와의 결합도가 높아져서 문제 발생 가능
 - 싱글톤 객체를 수정할 경우 사용하고 있는 다른 객체에도 영향 발생 가능성 있음
 - 멀티쓰레드 환경에서 동기화 처리 문제 발생 가능성

Singleton pattern 예제

```
public class SingletonClass {  
  
    private static SingletonClass instance = new SingletonClass();  
    private SingletonClass() {}  
  
    public static SingletonClass getInstance() {  
        return instance;  
    }  
}
```

```
public class SingletonClass {  
  
    private static SingletonClass instance;  
    private SingletonClass() {}  
  
    static {  
        try { instance = new SingletonClass();}  
        catch(Exception e) {  
            throw  
                new RuntimeException("[Error] " + e.getMessage());  
        }  
    }  
  
    public static SingletonClass getInstance() {  
        return instance;  
    }  
}
```

- **static variable**
 - static 메모리 영역에 할당
 - 모든 객체에서 공유 가능
 - GC에서 관리하지 않는 영역이므로 프로그램 종료시까지 존재
- 생성자는 private 이므로 외부 접근 불가
- **static block**
 - 클래스가 로딩될 때 한번만 실행

Java – SQLite 연동 Review

1. 사전 준비 - DB설계, DB 파일 생성, Java Project 내에 DB 파일 복사

2. DB Connection 생성

```
String dbFile = "<db filename>";  
conn = DriverManager.getConnection("jdbc:sqlite:" + dbFile);
```

3. SQL 실행

(R) Statement stmt = conn.createStatement();
String sql = "SELECT * FROM <table name> WHERE <conditions>";
ResultSet rs = stmt.executeQuery(sql);

(C)(U)(D)

```
Statement stmt = conn.createStatement();  
String sql = "INSERT INTO <table name> (fields...) VALUES (values...);  
int count = stmt.executeUpdate(sql);
```

4. SQL 결과 사용

(R) while (rs.next()) {
 int id = rs.getInt("<numeric field>");
 String str = rs.getString("<string field>");
 System.out.println(id + " " + str);
}

```
if(count > 0)  
    ;// Query 성공!  
else  
    ;// Query 실패!
```

5. Statement closing

```
stmt.close();
```

(C)(U)(D)

SQL Test

- SELECT * FROM list
- SELECT count(id) FROM list
- SELECT count(id) FROM list WHERE title = '산책하기'
- SELECT * FROM list WHERE title LIKE '%과제%'
- SELECT * FROM list WHERE title LIKE '%과제%' OR memo like '%과제%'
- SELECT * FROM list ORDER BY title
- SELECT * FROM list ORDER BY title DESC
- SELECT * FROM list ORDER BY due_date
- SELECT * FROM list ORDER BY due_date DESC
- SELECT * FROM list WHERE category = '과제'
- SELECT DISTINCT category FROM list

검색 : find <keyword>

SELECT * FROM list WHERE title LIKE '%keyword%' OR memo like '%keyword%'

```
case "find":  
    String keyword = sc.nextLine().trim();  
    TodoUtil.findList(l, keyword);  
    break;
```

TodoMain

```
public static void findList(TodoList l, String keyword) {  
    int count=0;  
    for (TodoItem item : l.getList(keyword)) {  
        System.out.println(item.toString());  
        count++;  
    }  
    System.out.printf("총 %d개의 항목을 찾았습니다.\n", count);  
}
```

TodoUtil

```
public ArrayList<TodoItem> getList(String keyword) {  
    ArrayList<TodoItem> list = new ArrayList<TodoItem>();  
    PreparedStatement pstmt;  
    keyword = "%" + keyword + "%";  
    try {  
        String sql = "SELECT * FROM list WHERE title like ? or memo like ?";  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setString(1, keyword);  
        pstmt.setString(2, keyword);  
        ResultSet rs = pstmt.executeQuery();  
    } catch (SQLException e) {  
        e.printStackTrace();  
    }  
    return list;  
}
```

TodoList

검색

Command > `ls`

[전체 목록, 총 8개]

- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
- 2 [기타] 동아리모임 - 동아리 동기들과 밥먹기 - 2021/09/22 - 2021/09/02 12:10:21
- 3 [취미] 산책하기 - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
- 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
- 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 6 [기타] 동아리회식 - 동아리 동기들과 밥먹기 - 2021/09/29 - 2021/09/12 12:10:21
- 7 [취미] 운동 - 1킬로미터 빠르게 걷기 - 2021/09/15 - 2021/09/13 20:30:31
- 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45

Command > `find 과제`

- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
 - 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 총 2개의 항목을 찾았습니다.

Command > `find 하기`

- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
 - 3 [취미] 산책하기 - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
 - 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
 - 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
 - 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45
- 총 5개의 항목을 찾았습니다.

카테고리 : ls_cate, find_cate <keyword>

SELECT DISTINCT category FROM list

SELECT * FROM list WHERE category = 'keyword'

```
case "ls_cate":  
    TodoUtil.listcateAll(l);  
    break;
```

TodoMain

```
case "find_cate":  
    String cate = sc.nextLine().trim();  
    TodoUtil.findcateList(l, cate);  
    break;
```

```
public static void listcateAll(TodoList l) {  
    int count=0;  
    for (String item : l.getCategories()) {  
        System.out.print(item + " ");  
        count++;  
    }  
    System.out.printf("\n총 %d개의 카테고리가 등록되어 있습니다.\n", count);  
}
```

TodoUtil

```
public static void findcateList(TodoList l, String cate) {  
    int count=0;  
    for (TodoItem item : l.getListCategory(cate)) {  
        System.out.println(item.toString());  
        count++;  
    }  
    System.out.printf("\n총 %d개의 항목을 찾았습니다.\n", count);  
}
```

```
public ArrayList<String> getCategories() {  
    ArrayList<String> list = new ArrayList<String>();  
    Statement stmt;  
    try {  
        stmt = conn.createStatement();  
        String sql = "SELECT DISTINCT category FROM list";  
        ResultSet rs = stmt.executeQuery(sql);  
    }
```

TodoList

```
public ArrayList<TodoItem> getListCategory(String keyword) {  
    ArrayList<TodoItem> list = new ArrayList<TodoItem>();  
    PreparedStatement pstmt;  
    try {  
        String sql = "SELECT * FROM list WHERE category = ?";  
        pstmt = conn.prepareStatement(sql);  
        pstmt.setString(1, keyword);  
        ResultSet rs = pstmt.executeQuery();  
    }
```

카테고리

Command > `ls`

[전체 목록, 총 8개]

- 1 [과제] 2주차과제하기 – 운영체제, 알고리즘 과제하기 – 2021/09/15 – 2021/09/01 10:00:01
- 2 [기타] 동아리모임 – 동아리 동기들과 밥먹기 – 2021/09/22 – 2021/09/02 12:10:21
- 3 [취미] 산책하기 – 평봉필드 2바퀴 돌기 – 2021/09/05 – 2021/09/03 20:30:31
- 4 [과제] 세미나자료찾기 – 랩실 세미나 자료 준비하기 – 2021/09/17 – 2021/09/07 09:20:45
- 5 [과제] 3주차과제하기 – 운영체제, 알고리즘 과제하기 – 2021/09/21 – 2021/09/10 10:00:01
- 6 [기타] 동아리회식 – 동아리 동기들과 밥먹기 – 2021/09/29 – 2021/09/12 12:10:21
- 7 [취미] 운동 – 1킬로미터 빠르게 걷기 – 2021/09/15 – 2021/09/13 20:30:31
- 8 [과제] 세미나발표 – 랩실 세미나 준비하기 – 2021/09/27 – 2021/09/14 09:20:45

Command > `ls_cate`

과제 기타 취미

총 3개의 카테고리가 등록되어 있습니다.

Command > `find_cate 과제`

- 1 [과제] 2주차과제하기 – 운영체제, 알고리즘 과제하기 – 2021/09/15 – 2021/09/01 10:00:01
- 4 [과제] 세미나자료찾기 – 랩실 세미나 자료 준비하기 – 2021/09/17 – 2021/09/07 09:20:45
- 5 [과제] 3주차과제하기 – 운영체제, 알고리즘 과제하기 – 2021/09/21 – 2021/09/10 10:00:01
- 8 [과제] 세미나발표 – 랩실 세미나 준비하기 – 2021/09/27 – 2021/09/14 09:20:45

총 4개의 항목을 찾았습니다.

정렬 : ls_name, ls_date, ls_name_desc, ...

SELECT * FROM list ORDER BY title

```
case "ls_name":  
    System.out.println("제목순으로 정렬하였습니다.");  
    TodoUtil.listAll(l, "title", 1);  
    break;  
  
case "ls_name_desc":  
    System.out.println("제목역순으로 정렬하였습니다.");  
    TodoUtil.listAll(l, "title", 0);  
    break;  
  
case "ls_date":  
    System.out.println("날짜순으로 정렬하였습니다.");  
    TodoUtil.listAll(l, "due_date", 1);  
    break;  
  
case "ls_date_desc":  
    System.out.println("날짜역순으로 정렬하였습니다.");  
    TodoUtil.listAll(l, "due_date", 0);  
    break;
```

TodoMain

SELECT * FROM list ORDER BY title DESC

```
public static void listAll(TodoList l, String orderby, int ordering) {  
    System.out.printf("[전체 목록, 총 %d개]\n", l.getCount());  
    for (TodoItem item : l.getOrderedList(orderby, ordering)) {  
        System.out.println(item.toString());  
    }
```

TodoUtil

```
public ArrayList<TodoItem> getOrderedList(String orderby, int ordering) {  
    ArrayList<TodoItem> list = new ArrayList<TodoItem>();  
    Statement stmt;  
    try {  
        stmt = conn.createStatement();  
        String sql = "SELECT * FROM list ORDER BY "+ orderby;  
        if (ordering==0)  
            sql += " desc";  
        ResultSet rs = stmt.executeQuery(sql);
```

TodoList

제목 정렬

Command > `ls_name`

제목순으로 정렬하였습니다.

[전체 목록, 총 8개]

- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
- 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 2 [기타] 동아리모임 - 동아리 동기들과 밥먹기 - 2021/09/22 - 2021/09/02 12:10:21
- 6 [기타] 동아리회식 - 동아리 동기들과 밥먹기 - 2021/09/29 - 2021/09/12 12:10:21
- 3 [취미] 산책하기 - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
- 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45
- 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
- 7 [취미] 운동 - 1킬로미터 빠르게 걷기 - 2021/09/15 - 2021/09/13 20:30:31

Command > `ls_name_desc`

제목역순으로 정렬하였습니다.

[전체 목록, 총 8개]

- 7 [취미] 운동 - 1킬로미터 빠르게 걷기 - 2021/09/15 - 2021/09/13 20:30:31
- 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
- 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45
- 3 [취미] 산책하기 - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
- 6 [기타] 동아리회식 - 동아리 동기들과 밥먹기 - 2021/09/29 - 2021/09/12 12:10:21
- 2 [기타] 동아리모임 - 동아리 동기들과 밥먹기 - 2021/09/22 - 2021/09/02 12:10:21
- 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01

날짜 정렬

Command > `ls_date`

날짜순으로 정렬하였습니다.

[전체 목록, 총 8개]

- 3 [취미] 산책하기 - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
- 7 [취미] 운동 - 1킬로미터 빠르게 걷기 - 2021/09/15 - 2021/09/13 20:30:31
- 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
- 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 2 [기타] 동아리모임 - 동아리 동기들과 밥먹기 - 2021/09/22 - 2021/09/02 12:10:21
- 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45
- 6 [기타] 동아리회식 - 동아리 동기들과 밥먹기 - 2021/09/29 - 2021/09/12 12:10:21

Command > `ls_date_desc`

날짜역순으로 정렬하였습니다.

[전체 목록, 총 8개]

- 6 [기타] 동아리회식 - 동아리 동기들과 밥먹기 - 2021/09/29 - 2021/09/12 12:10:21
- 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45
- 2 [기타] 동아리모임 - 동아리 동기들과 밥먹기 - 2021/09/22 - 2021/09/02 12:10:21
- 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
- 1 [과제] 2주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
- 7 [취미] 운동 - 1킬로미터 빠르게 걷기 - 2021/09/15 - 2021/09/13 20:30:31
- 3 [취미] 산책하기 - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31

제목 중복 체크

Command > `ls`

[전체 목록, 총 8개]

- 1 [과제] 2주차과제하기 – 운영체제, 알고리즘 과제하기 – 2021/09/15 – 2021/09/01 10:00:01
- 2 [기타] 동아리모임 – 동아리 동기들과 밥먹기 – 2021/09/22 – 2021/09/02 12:10:21
- 3 [취미] 산책하기 – 평봉필드 2바퀴 돌기 – 2021/09/05 – 2021/09/03 20:30:31
- 4 [과제] 세미나자료찾기 – 랩실 세미나 자료 준비하기 – 2021/09/17 – 2021/09/07 09:20:45
- 5 [과제] 3주차과제하기 – 운영체제, 알고리즘 과제하기 – 2021/09/21 – 2021/09/10 10:00:01
- 6 [기타] 동아리회식 – 동아리 동기들과 밥먹기 – 2021/09/29 – 2021/09/12 12:10:21
- 7 [취미] 운동 – 1킬로미터 빠르게 걷기 – 2021/09/15 – 2021/09/13 20:30:31
- 8 [과제] 세미나발표 – 랩실 세미나 준비하기 – 2021/09/27 – 2021/09/14 09:20:45

Command > `add`

[항목 추가]

제목 > `산책하기`

제목이 중복됩니다!

```
public static void createItem(TodoList l) {  
  
    String title, desc, category, due_date;  
    Scanner sc = new Scanner(System.in);  
  
    System.out.print("[항목 추가]\n"  
        + "제목 > ");  
    title = sc.nextLine();  
    if (l.isDuplicate(title)) {  
        System.out.println("제목이 중복됩니다!");  
        return;  
    }  
}
```

TodoUtil

완료 체크 필드 추가 (선택 미션)

이미 생성된 테이블명을 변경하려면 ALTER TABLE 문을 사용한다. 형식은 다음과 같다.

- Table 필드 추가하기
 - ~~ALTER TABLE list ADD COLUMN is_completed integer default 0;~~
 - value : 0 not completed, 1 completed
- Command 추가하기
 - 항목 완료하기 : `comp <번호>`
 - 완료된 항목만 출력하기 : `ls_comp`
- 코드 수정
 - TodoItem : ~~멤버 is_completed 추가, setter/getter 추가, 생성자 변경, toString() 변경~~
 - TodoMain : ~~명령어 처리 구문 2가지 추가~~
 - TodoUtil : ~~completeItem(), listAll(숫자) 추가~~
 - TodoList : ~~completeItem(), getL ist(숫자) 추가~~

Command > `comp 3`

완료 체크하였습니다.

Command > `ls`

[전체 목록, 총 8개]

- 1 [과제] 2주차과제하기 [V] - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
- 2 [기타] 동아리모임 - 동아리 동기들과 밥먹기 - 2021/09/22 - 2021/09/02 12:10:21
- 3 [취미] 산책하기 [V] - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
- 4 [과제] 세미나자료찾기 - 랩실 세미나 자료 준비하기 - 2021/09/17 - 2021/09/07 09:20:45
- 5 [과제] 3주차과제하기 - 운영체제, 알고리즘 과제하기 - 2021/09/21 - 2021/09/10 10:00:01
- 6 [기타] 동아리회식 - 동아리 동기들과 밥먹기 - 2021/09/29 - 2021/09/12 12:10:21
- 7 [취미] 운동 - 1킬로미터 빠르게 걷기 - 2021/09/15 - 2021/09/13 20:30:31
- 8 [과제] 세미나발표 - 랩실 세미나 준비하기 - 2021/09/27 - 2021/09/14 09:20:45

Command > `ls_comp`

- 1 [과제] 2주차과제하기 [V] - 운영체제, 알고리즘 과제하기 - 2021/09/15 - 2021/09/01 10:00:01
 - 3 [취미] 산책하기 [V] - 평봉필드 2바퀴 돌기 - 2021/09/05 - 2021/09/03 20:30:31
- 총 2개의 항목이 완료되었습니다.

개인프로젝트 - SQLite 기반 TodoList App

- 제출물 : 제작보고서(PDF) + 데모(5분 동영상 또는 원페이지 포스터 PDF)
- 제출 마감 : 2021년 10월 17일(일) 밤 12시 - 각자는 별도의 과제 없음
- 수행 아이템 (100점 만점기준, 원하는 항목 선택할 것)

구분	기본필드(6개)	완료체크필드 추가	필드 2개 이상 추가 <small>적은 사용 이유 or 위선 분위 or 함께 하는 사람</small>	레코드 50개 이상 입력/사용 <small>[10점]</small>	Category Table 별도 생성/사용 <small>[10점]</small>
Item 추가	5	5	5		
목록	5	5	5		
카테고리	5	5			
수정	5	5	5		
삭제	5	5	5		

50점

예) 주기적인 항목을 한번에 추가하기
매달 1일에 고향 다녀오기

필드 추가

- with_who (text) default null ;
- priority (integer) default 0 ;

Item 추가

목록

수정

삭제

작은 사람 이름이 enter 만 들어오면

6주차 실습과제 제출

- Due date - 2021년 10월 10일 (일) 밤 11시 59분
- 제출방법 - LMS 과제로 PDF 파일 제출
- PDF 파일 내 포함 내용
 - 파일명 형식 : 이름(학번)_6주차.pdf
 - 6주차 실습 수행을 통해 새로 제작한 프로젝트 결과
 - 프로젝트 최종 소스를 Commit & Push 된 자신의 Github URI 기입(새로운 Repo 필요)
 - 필수 미션 수행 완료를 보여주는 프로그램 실행 이미지 첨부 및 간단 설명 입력
 - todolist.db로 옮긴 데이터를 사용하여 기존의 모든 명령어가 수행되도록 프로젝트 리뉴얼할 것
 - 데이터를 10개 이상 등록하여 실행해 볼 것
 - 선택 미션 수행 완료를 보여주는 프로그램 실행 이미지 첨부 및 간단 설명 입력
 - 6주차 실습을 수행하면서 느낀 점 (50자 이상)
 - 개인프로젝트 수행 전략/계획 → → 나쁠 수도 있지만 전략적 예상하기