

1. Know-How	2
1.1 Android	2
1.1.1 Android Calender Library (참조용)	2
1.1.2 Android Graph library	4
1.1.3 Android HTTP Request	10
1.1.4 Custom ActionBar	14
1.1.5 Custom Calendar View API	14
1.1.6 Get filename and path from URI from mediastore	15
1.1.7 JWT	15
1.1.7.1 [JWT] 토큰(Token) 기반 인증에 대한 소개	15
1.2 AWS	35
1.2.1 AWS signIn and Installation	35
1.3 Git	35
1.3.1 Git Configuration From Ubuntu 16.04 Server	35
1.3.2 git public key	36
1.3.3 jslim public key	36
1.4 JIRA	36
1.4.1 Mastering Jira	36
1.5 mysql	36
1.5.1 mySQL utf8	36
1.5.2 mysql UTF-8 setting	38
1.6 nodejs	39
1.6.1 NVM installation	39
1.7 Restful	39
1.7.1 RESTful Web Services	39
1.7.1.1 Restful 블로그자료	39
1.8 Server	44
1.8.1 SMS Server Configuration	44
1.8.2 포워드 프록시(forward proxy) 리버스 프록시(reverse proxy) 의 차이	45
1.9 공공기관 API 특일 서비스	47
1.10 메일보내기 관련 Gmail	47
2. Library	47
3. Meeting	47
3.1 비트 프로젝트 07/10 (월)	47
3.2 비트 프로젝트 07/11 (화)	48
3.3 비트 프로젝트 07/12 (수)	49
3.4 비트 프로젝트 07/14 (금)	54
3.5 비트 프로젝트 07/17 (월)	63
3.6 비트 프로젝트 07/18 (화)	71
3.7 비트 프로젝트 07/26 (수)	72
3.7.1 Android prototype code	73
3.8 비트 프로젝트 08/23 (수)	74
4. Project	74
4.1 ANDROID prototype	74
4.2 aws key	75
4.3 Coding Standards Guide	75
4.4 Dailyreport confirm field value rule	89
4.5 DB Excel정리 및 eXERD 파일	89
4.6 email tamplet	90
4.7 Internal Site Information	91
4.8 JIRA Standards Guide	91
4.9 node.js와 android 통신규약	103
4.10 Project Topic	104
4.11 restful 사양서	104
4.12 WEB prototype	105
4.13 발표자료	105
4.14 팀장에게 보고서의 승인요청된 데이터체크 쿼리	106

Know-How

Android

Android Calender Library (참조용)

안드로이드 UI 관련 라이브러리 입니다. (참조용)

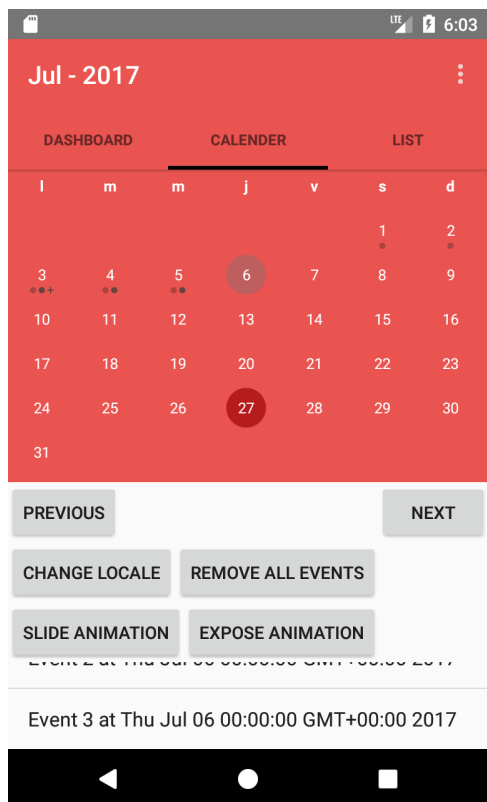
소스 코드



CompactCalenda...iew-master.zip

Hello ListView.

Hello ListView.



Android Graph Library

Create Charts and Graphs Easier With These Android Libraries

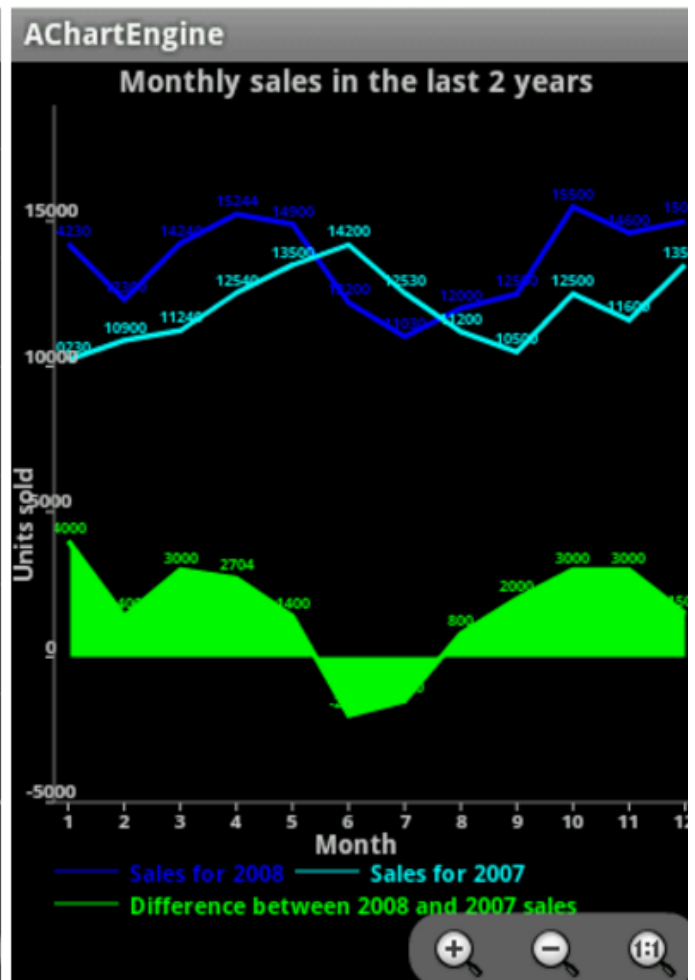
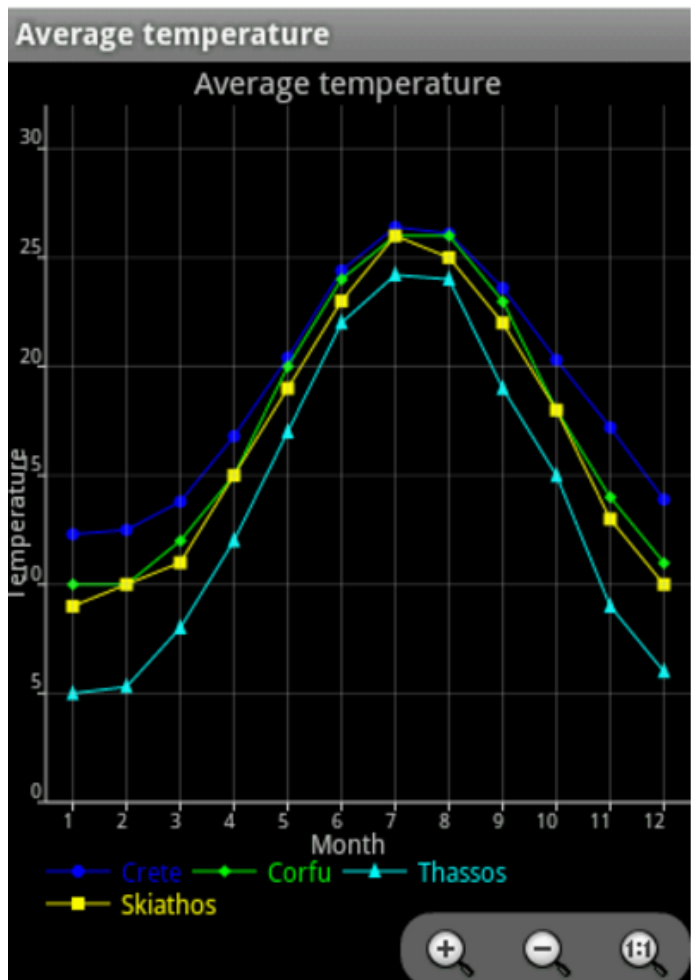
Creating chart has never been a fun task to do. These libraries will help developer avoid this horrendous task.

[AChartEngine](#)

AChartEngine is a charting library for [Android app development](#) which supports a large ranges of chart types, from simple ones such as line,

bar and pie to complicated ones such as combined chart. It has the biggest number of supported chart types compared to other libraries mentioned in this post.

The charts can be built as a view which can be added to any view groups or as an intent, such as it can be used to start an activity.

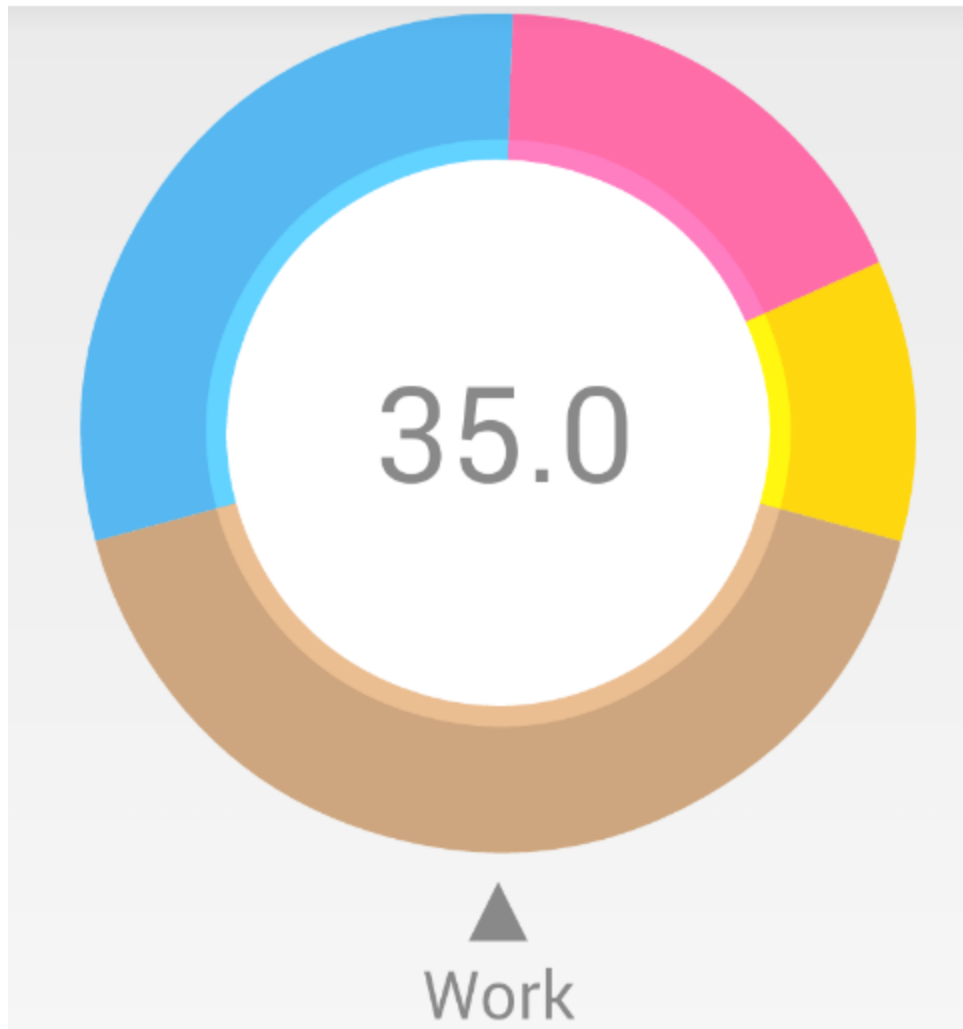


[EazeGraph](#)

EazeGraph can help creating beautiful and fancy charts.

Its main goal was to create a lightweight library which is easy to use and highly customizable with a modern look.

EazeGraph currently supports 4 different chart types which are Bar Chart, Stacked Bar Chart, Pie Chart and Line Chart.

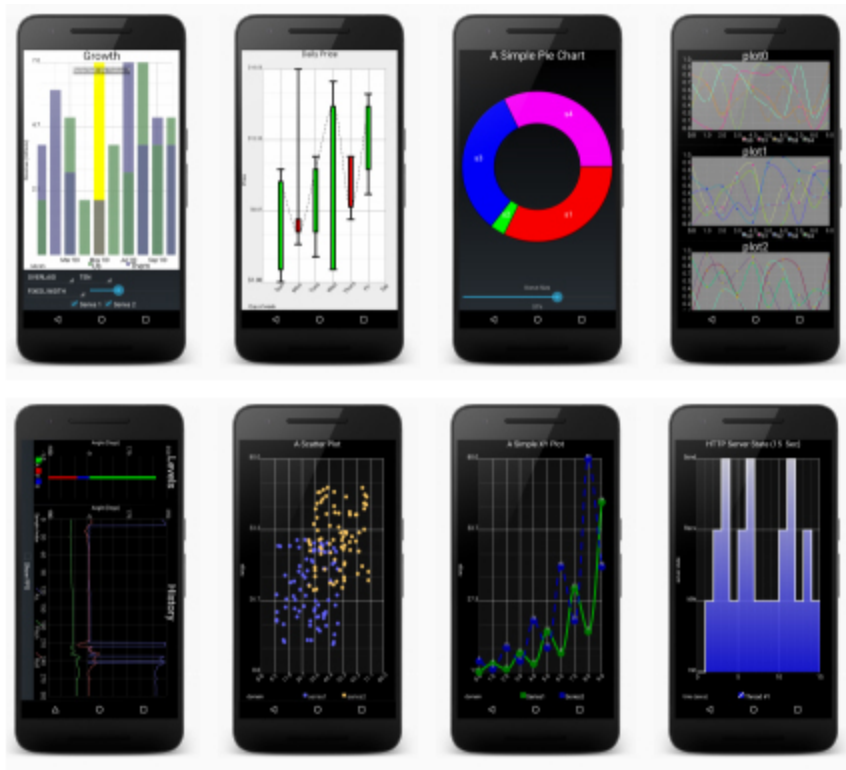


[Androidplot](#)

Androidplot is used to develop dynamic and static charts.

The library is compatible with [Android 1.6+](#) and is used by over 1,000 apps on Google Play.

It supports Line, Bar, Pie, Scatter, Step and Candlestick Charts.

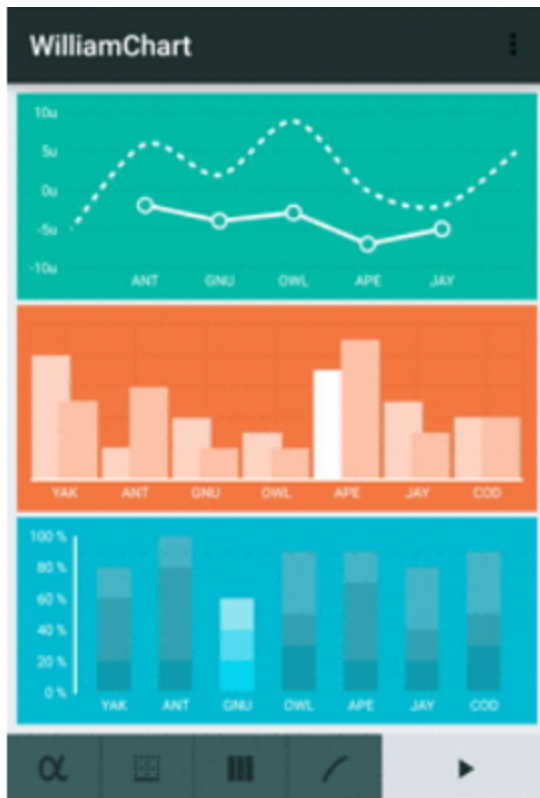


WilliamChart

WilliamChart is another library which presents chart in a pleasant and intuitive way.

Supported charts:

- LineChartView
- BarChartView
- HorizontalBarChartView
- StackBarChartView
- HorizontalStackBarChartView

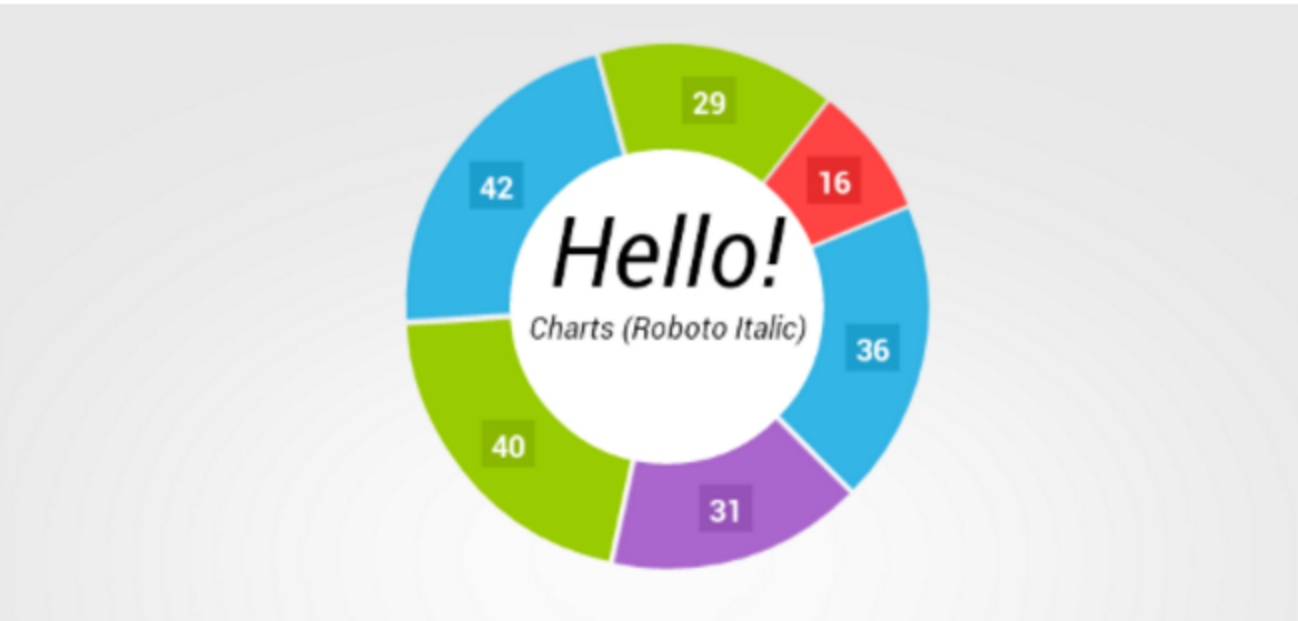
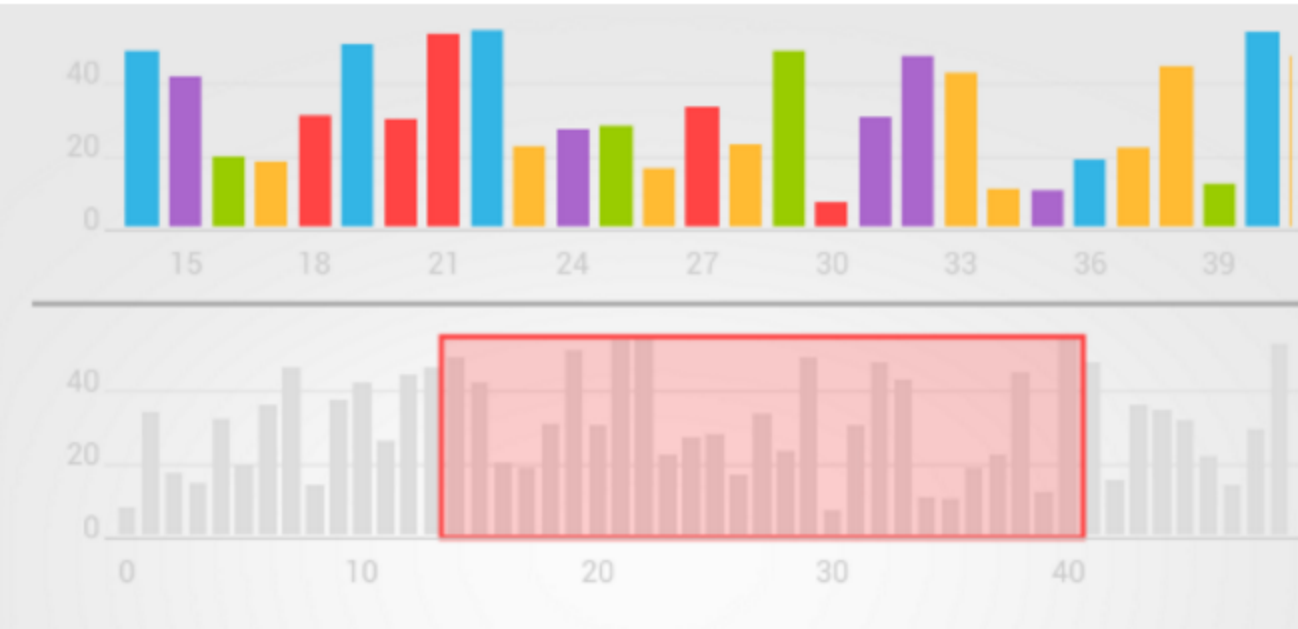


HelloCharts for Android

The library is compatible with API 8+(Android 2.2). It works the best when hardware acceleration is available so API 14+(Android 4.0) is recommended.

It supports:

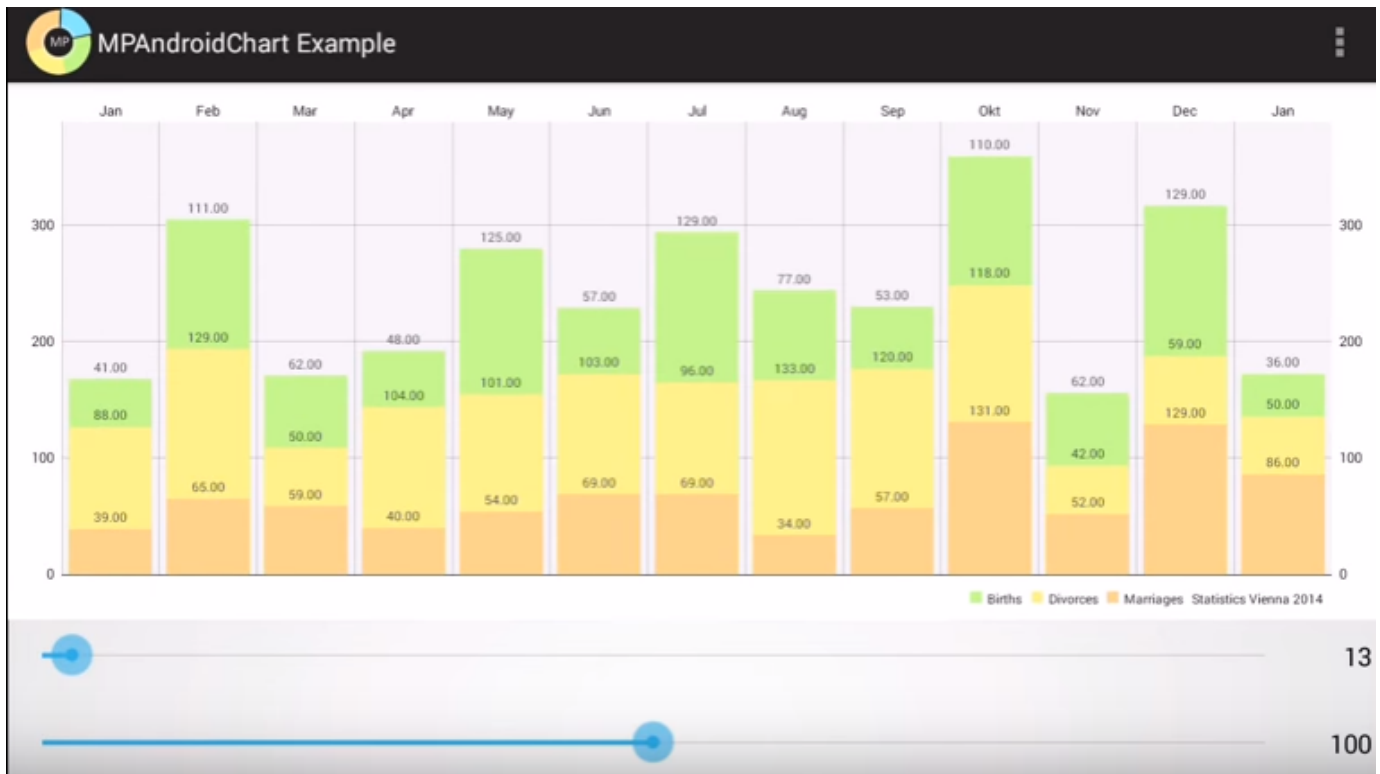
- Line chart(cubic lines, filled lines, scattered points)
- Column chart(grouped, stacked, negative values)
- Pie chart
- Bubble chart
- Combo chart(columns/lines)
- Preview charts(for column chart and [line chart](#))
- Zoom(pinch to zoom, double tap zoom), scroll and fling
- Custom and auto-generated axes(top, bottom, left, right, inside)
- Animations



MPAndroidChart

It is a powerful & easy to use chart library for Android version which has API level 8 and upwards.

MPAndroidChart allows cross-platform development between Android and iOS. It supports 8 different chart types.



Android HTTP Request

<https://github.com/kevinsawicki/http-request>

Http Request

A simple convenience library for using a [HttpURLConnection](#) to make requests and access the response.

This library is available under the [MIT License](#).

Usage

The http-request library is available from [Maven Central](#).

```
<dependency>
  <groupId>com.github.kevinsawicki</groupId>
  <artifactId>http-request</artifactId>
  <version>6.0</version>
</dependency>
```

Not using [Maven](#)? Simply copy the [HttpRequest](#) class into your project, update the package declaration, and you are good to go.

Javadocs are available [here](#).

FAQ

Who uses this?

See [here](#) for a list of known projects using this library.

Why was this written?

This library was written to make HTTP requests simple and easy when using a `HttpURLConnection`.

Libraries like [Apache HttpComponents](#) are great but sometimes for either simplicity, or perhaps for the environment you are deploying to (Android), you just want to use a good old-fashioned `HttpURLConnection`.

This library seeks to add convenience and common patterns to the act of making HTTP requests such as a fluid-interface for building requests and support for features such as multipart requests.

Bottom line: The single goal of this library is to improve the usability of the `HttpURLConnection` class.

What are the dependencies?

None. The goal of this library is to be a single class class with some inner static classes. The test project does require [Jetty](#) in order to test requests against an actual HTTP server implementation.

How are exceptions managed?

The `HttpRequest` class does not throw any checked exceptions, instead all low-level exceptions are wrapped up in a `HttpRequestException` which extends `RuntimeException`. You can access the underlying exception by catching `HttpRequestException` and calling `getCause()` which will always return the original `IOException`.

Are requests asynchronous?

No. The underlying `HttpURLConnection` object that each `HttpRequest` object wraps has a synchronous API and therefore all methods on `HttpRequest` are also synchronous.

Therefore it is important to not use an `HttpRequest` object on the main thread of your application.

Here is a simple Android example of using it from an [AsyncTask](#):

```
private class DownloadTask extends AsyncTask<String, Long, File> {
    protected File doInBackground(String... urls) {
        try {
            HttpRequest request = HttpRequest.get(urls[0]);
            File file = null;
            if (request.ok()) {
                file = File.createTempFile("download", ".tmp");
                request.receive(file);
                publishProgress(file.length());
            }
            return file;
        } catch (HttpRequestException exception) {
            return null;
        }
    }

    protected void onProgressUpdate(Long... progress) {
        Log.d("MyApp", "Downloaded bytes: " + progress[0]);
    }

    protected void onPostExecute(File file) {
        if (file != null)
            Log.d("MyApp", "Downloaded file to: " + file.getAbsolutePath());
        else
            Log.d("MyApp", "Download failed");
    }
}

new DownloadTask().execute("http://google.com");
```

Examples

Perform a GET request and get the status of the response

```
int response = HttpRequest.get("http://google.com").code();
```

Perform a GET request and get the body of the response

```
String response = HttpRequest.get("http://google.com").body();  
System.out.println("Response was: " + response);
```

Print the response of a GET request to standard out

```
HttpRequest.get("http://google.com").receive(System.out);
```

Adding query parameters

```
HttpRequest request = HttpRequest.get("http://google.com", true, 'q', "baseball gloves", "size", 100);  
System.out.println(request.toString()); // GET http://google.com?q=baseball%20gloves&size=100
```

Using arrays as query parameters

```
int[] ids = new int[] { 22, 23 };  
HttpRequest request = HttpRequest.get("http://google.com", true, "id", ids);  
System.out.println(request.toString()); // GET http://google.com?id[]=22&id[]=23
```

Working with request/response headers

```
String contentType = HttpRequest.get("http://google.com")  
    .accept("application/json") //Sets request header  
    .contentType(); //Gets response header  
System.out.println("Response content type was " + contentType);
```

Perform a POST request with some data and get the status of the response

```
int response = HttpRequest.post("http://google.com").send("name=kevin").code();
```

Authenticate using Basic authentication

```
int response = HttpRequest.get("http://google.com").basic("username", "p4ssw0rd").code();
```

Perform a multipart POST request

```
HttpRequest request = HttpRequest.post("http://google.com");  
request.part("status[body]", "Making a multipart request");  
request.part("status[image]", new File("/home/kevin/Pictures/ide.png"));  
if (request.ok())  
    System.out.println("Status was updated");
```

Perform a POST request with form data

```
Map<String, String> data = new HashMap<String, String>();  
data.put("user", "A User");  
data.put("state", "CA");  
if (HttpRequest.post("http://google.com").form(data).created())  
    System.out.println("User was created");
```

Copy body of response to a file

```
File output = new File("/output/request.out");  
HttpRequest.get("http://google.com").receive(output);
```

Post contents of a file

```
File input = new File("/input/data.txt");
int response = HttpRequest.post("http://google.com").send(input).code();
```

Using entity tags for caching

```
File latest = new File("/data/cache.json");
HttpRequest request = HttpRequest.get("http://google.com");
//Copy response to file
request.receive(latest);
//Store eTag of response
String eTag = request.eTag();
//Later on check if changes exist
boolean unchanged = HttpRequest.get("http://google.com")
    .ifNoneMatch(eTag)
    .notModified();
```

Using gzip compression

```
HttpRequest request = HttpRequest.get("http://google.com");
//Tell server to gzip response and automatically uncompress
request.acceptGzipEncoding().uncompress(true);
String uncompressed = request.body();
System.out.println("Uncompressed response is: " + uncompressed);
```

Ignoring security when using HTTPS

```
HttpRequest request = HttpRequest.get("https://google.com");
//Accept all certificates
request.trustAllCerts();
//Accept all hostnames
request.trustAllHosts();
```

Configuring an HTTP proxy

```
HttpRequest request = HttpRequest.get("https://google.com");
//Configure proxy
request.useProxy("localhost", 8080);
//Optional proxy basic authentication
request.proxyBasic("username", "password");
```

Following redirects

```
int code = HttpRequest.get("http://google.com").followRedirects(true).code();
```

Custom connection factory

Looking to use this library with [OkHttp](#)? Read [here](#).

```

HttpRequest.setConnectionFactory(new ConnectionFactory() {

    public HttpURLConnection create(URL url) throws IOException {
        if (!"https".equals(url.getProtocol()))
            throw new IOException("Only secure requests are allowed");
        return (HttpURLConnection) url.openConnection();
    }

    public HttpURLConnection create(URL url, Proxy proxy) throws IOException {
        if (!"https".equals(url.getProtocol()))
            throw new IOException("Only secure requests are allowed");
        return (HttpURLConnection) url.openConnection(proxy);
    }
});

```

Contributors

- Kevin Sawicki :: contributions
- Eddie Ringle :: contributions
- Sean Jensen-Grey :: contributions
- Levi Notik :: contributions
- Michael Wang :: contributions
- Julien HENRY :: contributions
- Benoit Lubek :: contributions
- Jake Wharton :: contributions
- Oskar Hagberg :: contributions
- David Pate :: contributions
- Anton Rieder :: contributions
- Jean-Baptiste Lièvremont :: contributions
- Roman Petrenko :: contributions

Custom ActionBar



Custom Calendar View API

Stack over flow 자료

<https://stackoverflow.com/questions/21951638/how-to-use-views-month-day-week-from-android-stock-calendar-in-a-custom-appl>

Google 소스 경로

<https://developers.google.com/google-apps/calendar/>

<https://android.googlesource.com/platform/packages/apps/Calendar/>

<https://developer.android.com/guide/topics/providers/calendar-provider.html>

Android week view 관련 api

<https://github.com/alamkanak/Android-Week-View>

UI 참조 용 Calender app

<https://android.gadgethacks.com/news/5-best-android-calendar-apps-replace-your-stock-one-0173013/>

참조

<http://android-pratap.blogspot.kr/2016/04/calendarview-like-google-calendar-in.html>

깃허브 Calender api 관련

<https://stackoverflow.com/questions/39031661/android-custom-calendar-with-month-view>

<https://github.com/wasabeef/awesome-android-ui/blob/master/pages/Calendar.md>

Get filename and path from URI from mediastore

```
public String getRealPathFromURI(Context context, Uri contentUri) {
    Cursor cursor = null;
    try {
        String[] proj = { MediaStore.Images.Media.DATA };
        cursor = context.getContentResolver().query(contentUri, proj, null, null, null);
        int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
        cursor.moveToFirst();
        return cursor.getString(column_index);
    } finally {
        if (cursor != null) {
            cursor.close();
        }
    }
}
```

JWT

[JWT] 토큰(Token) 기반 인증에 대한 소개

Git 라이브러리 (안드로이드) → <https://github.com/jwtk/jjwt>

Introduction

웹 / 앱 개발을 하면 로그인 과정에서 반드시 만나게 되는 개념이 -이다.

이미 많은 자료와 경험으로 인해 쿠키는 세션은 , 로그인은 일단 세션으로 해야지라는 개념이 개발자들의 머릿속에 자리 잡혀있다.

그러나, 최근 들어 IT 인프라 구성에 많은 변화가 생겼다. 웹 기반의 서비스들은 웹과 앱을 함께 서비스하는 것을 넘어 'Mobile First' 앱이 먼저라는 인식까지 생겨났다.

또한, AWS, Azure 와 같은 IaaS 클라우드 서비스가 대중화 되면서 고사양 단일 서버 아키텍처에서 중-저사양 다중 서버 아키텍처로 변화하고 있다.

이러한 상황에서 더 이상 쿠키-세션 기반 인증 아키텍처는 현재의 요구사항을 만족하지 못하고 있다.

현재의 요구 사항을 그나마 충족시키는 Web Token 기반 **JWT**에 대해서 알아보고 web Token이 나타난 배경과 장단점에 대해서 알아보겠다.

Part 1. JWT's Basic Information

JSON Web Token

- JSON 포맷을 이용한 Web Token
- Claim based Token
- 두 개체에서 JSON 객체를 이용해 Self-contained 방식으로 정보를 안전하게 전달
- 회원 인증, 정보 전달에 주로 사용
- [RFC 7519](#)

Claim based ?

- Claim : 사용자에게 대한 프로퍼티 / 속성
- 토큰 자체가 정보
- Self-contained : 자체 포함, 즉 토큰 자체가 정보
- key / value

Part 2. Web Token

웹 토큰의 필요성

- CSRF, 기존 시스템의 보안 문제
- CORS, 도메인 확장시 api로서의 문제
- Web, Mobile 등 다양한 클라이언트
- Session 의 한계
- Scale out 의 한계
- REST API : REST API는 Stateless를 지향

현재 우리는?

- 그런데 아직도 Cookie??
- 쿠키 기반 로그인 문제점은 너무 많다
- Server side Session 기반으로 넘어갈 것인가?

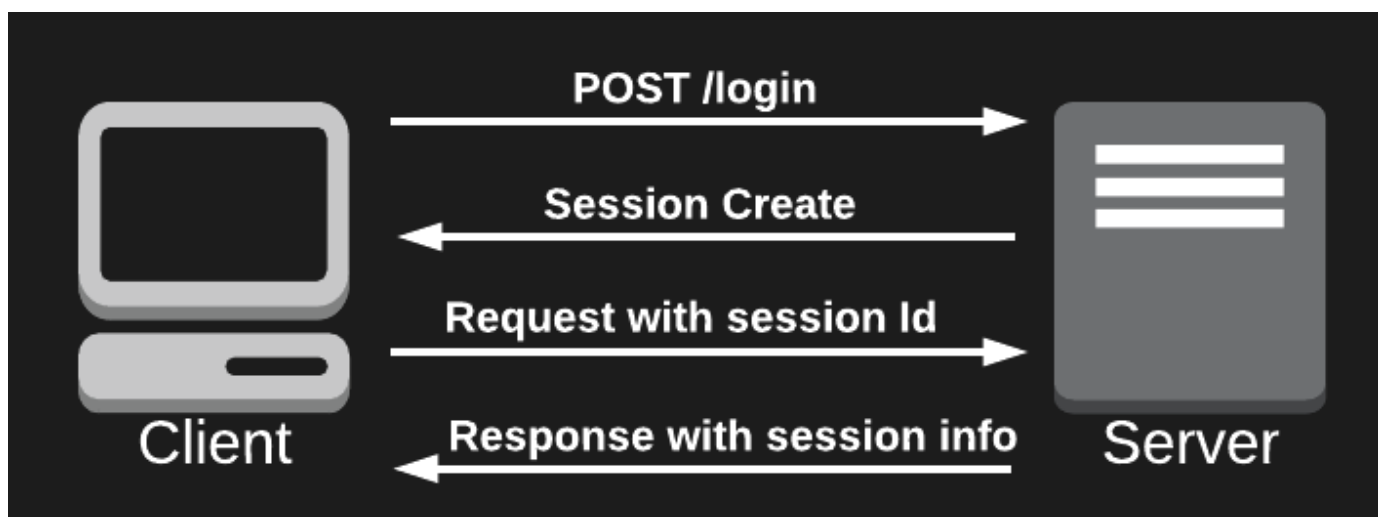
쿠키 기반 인증은 두말 할것 없이 변경해야 하고, 불편적 Session으로 갈것인가?

Part 3. Type of Authorization

Cookie - Client Side Storage

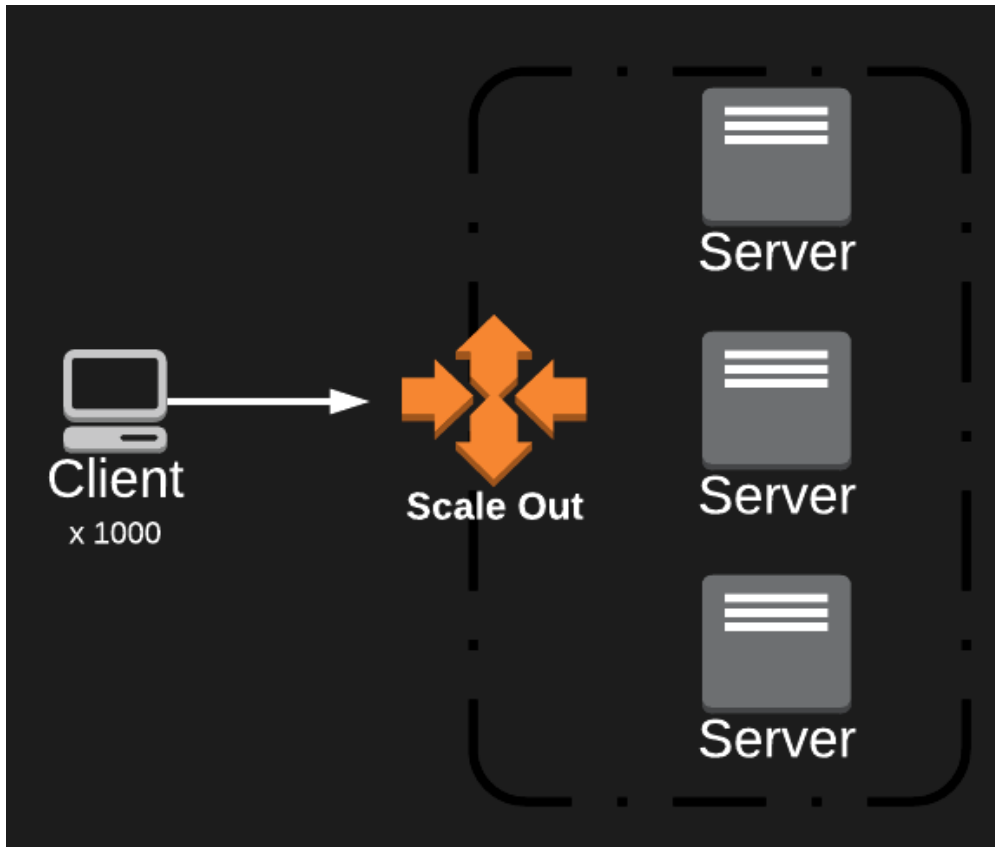
| 문제점 투성이

Session - Server Side Authorization



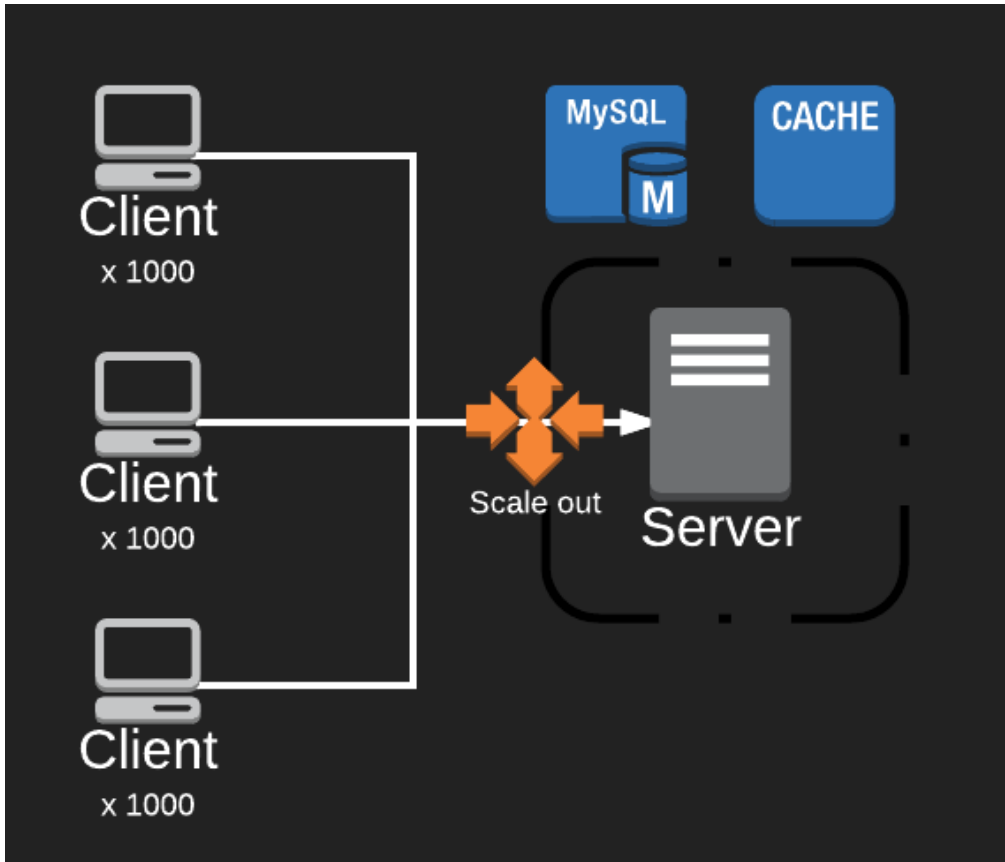
| Cookie와 차이점은 Cookie는 정보를 클라이언트에 저장하고 Session은 정보를 서버에 저장한다.

Session Problem 1 - 서버 확장시 세션 정보의 동기화 문제



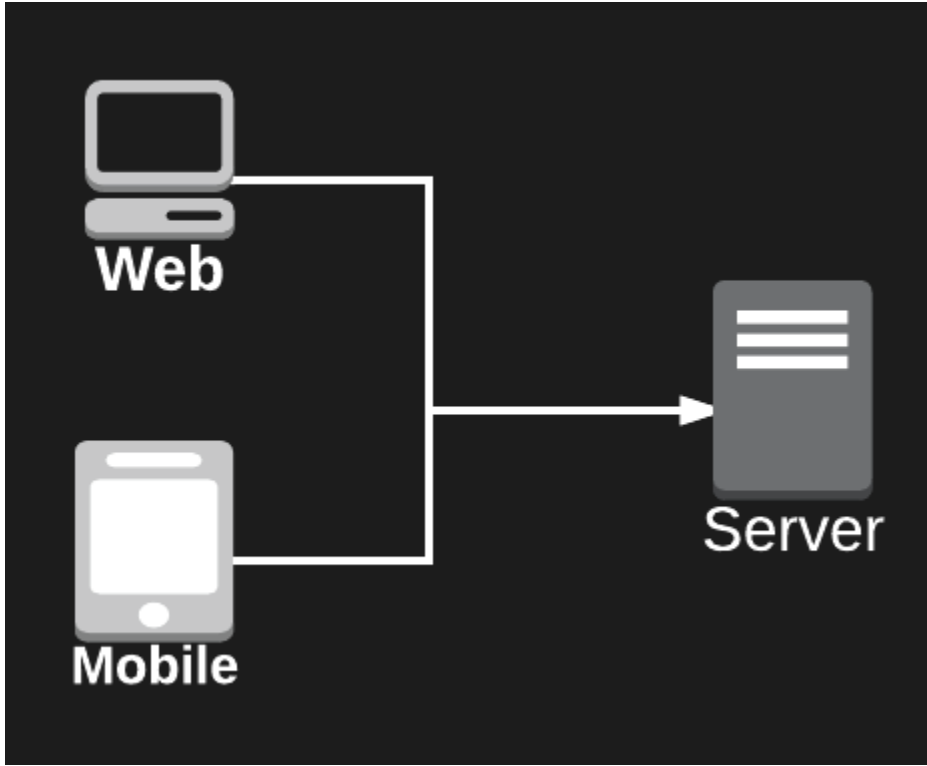
서버가 스케일아웃 돼서 여러 개가 생기면 각 서버마다 세션 정보가 저장된다.
로그인시(서버1), 새로고침(서버2) 로 접근하면 서버는 인증이 안됐다고 판단한다.

Session Problem 2 - 서버 / 세션 저장소의 부하



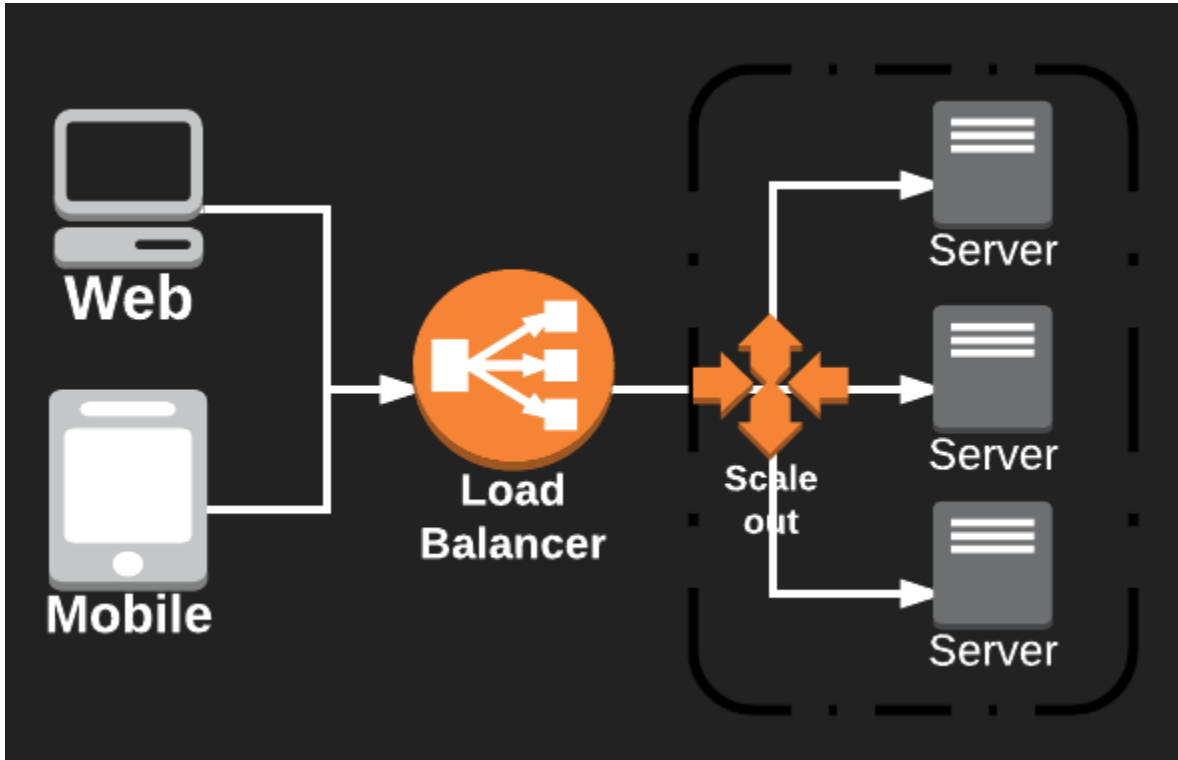
세션을 각 서버에 저장하지 않고 세션 전용 서버, DB에 저장해도 문제가 생긴다.
모든 요청 시 해당 서버에 조회해야 한다. DB 부하를 야기할 수 있다.

Session Problem 3 - 웹 / 앱 간의 상이한 쿠키-세션 처리 로직



기존의 Client는 웹 브라우저가 유일했다. 그러나 이제는 모바일로 접근하는 경우도 처리해야 한다.
웹 / 앱 의 쿠키 처리 방법이 다르고 또 다른 Client 가 생겨나면 쿠키-세션에 맞게 처리해야 한다.

Token - Self-contained & Stateless



앞의 문제를 해결하는 최선의 방법은 토큰이다.
토큰은 서버의 상태를 저장하지 않는다. 토큰 자체로 정보를 가지고 있기 때문에 별도의 인증서버가 필요없다. 따라서 요청을 받을 서버 자체에서 인증 프로세스를 수행할 수 있다.
또한, JSON 포맷으로 통신하기 때문에 어떤 Client 에서든 Data 통신에 JSON을 이용하면 토큰을 이용할 수 있다.

Part 4. JWT' s Architecture

JWT의 기본 구조

Header . Payload . Signature

Header

JWT 웹 토큰의 헤더 정보

- `typ` : 토큰의 타입, JWT만 존재
- `alg` : 해싱 알고리즘. (HMAC SHA256 or RSA). 헤더를 암호화 하는게 아니다. 토큰 검증시 사용.

```
{
  "alg" : "HS256",
  "typ" : "JWT"
}
```

위의 내용을 base64 로 인코딩한다. => eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
base 64 는 암호화된 문자열이 아니다. 같은 문자열에 대해서는 항상 같은 인코딩 문자열을 반환한다.

Payload

실제 토큰으로 사용하려는 데이터가 담기는 부분. 각 데이터를 Claim이라고 하며 다음과 같이 3가지 종류가 있다.

- **Reserved claims** : 이미 예약된 Claim. 필수는 아니지만 사용하길 권장. key 는 모두 3자리 String이다.
 - iss (String) : issuer, 토큰 발행자 정보
 - exp (Number) : expiration time, 만료일
 - sub (String) : subject, 제목
 - aud (String) : audience,
 - [More](#)
- **Public claims** : 사용자 정의 Claim.
 - Public 이라는 이름처럼 공개용 정보
 - 충돌 방지를 위해 [URI](#) 포맷을 이용해 저장한다.
- **Private claims** : 사용자 정의 Claim
 - Public claims 과 다르게 사용자가 임의로 정한 정보
 - 아래와 같이 일반 정보를 저장한다.
- ```
{
 "name" : "hak",
 "age" : 26,
}
```

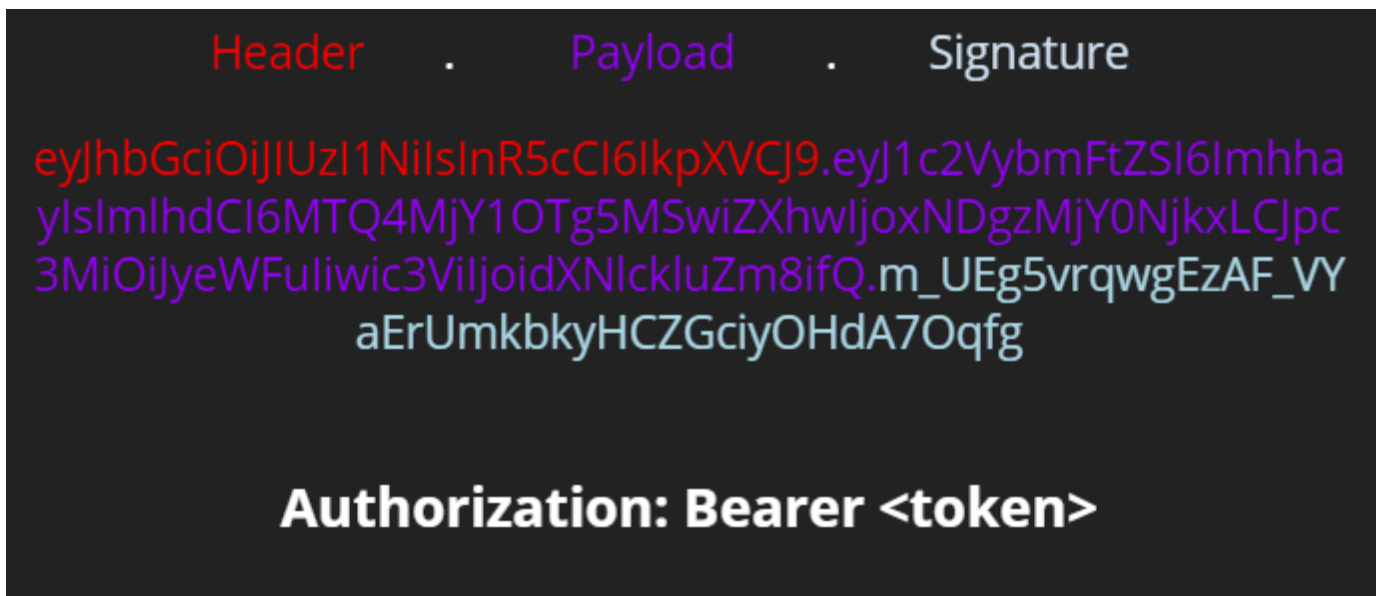
## Signature

Header와 Payload의 데이터 무결성과 변조 방지를 위한 서명  
Header + Payload 를 합친 후, Secret 키와 함께 Header의 해싱 알고리즘으로 인코딩

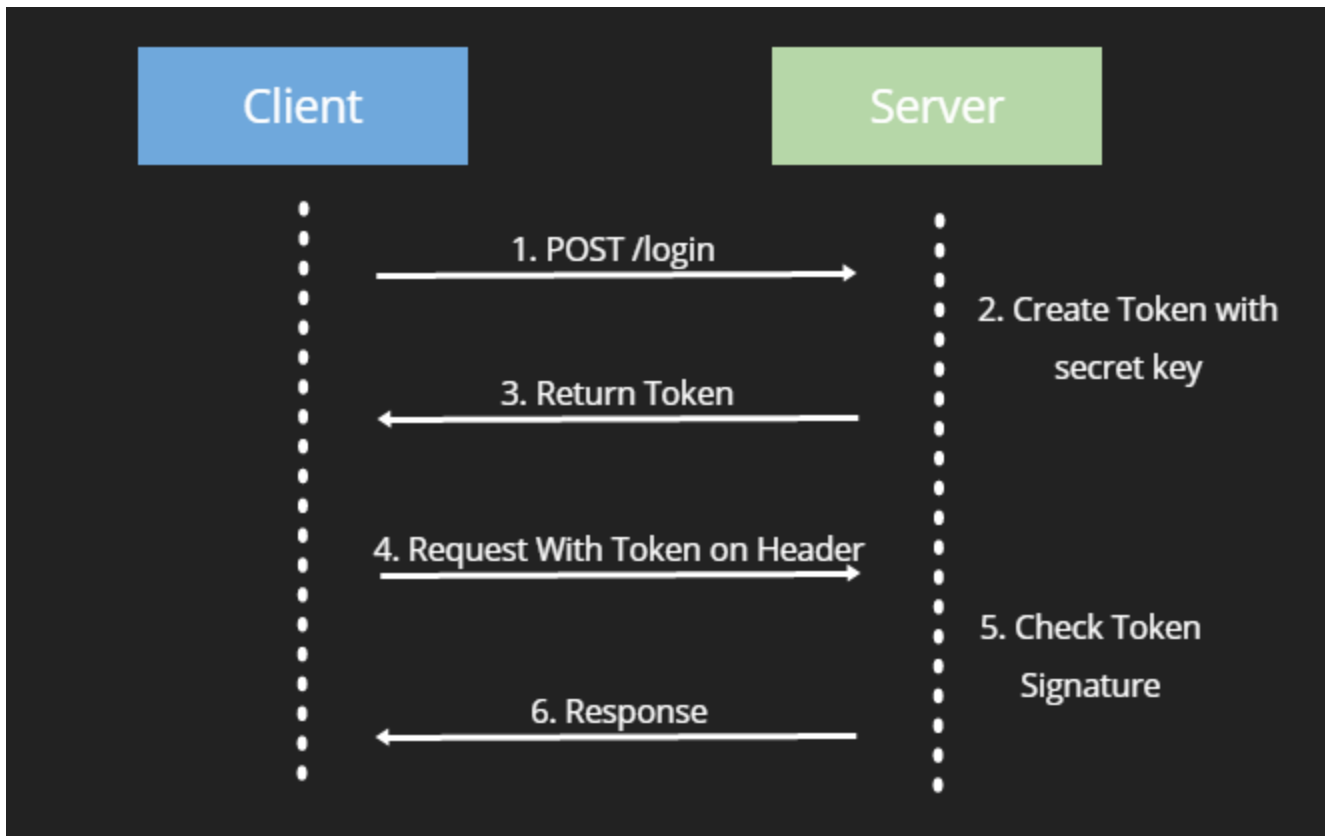
```
HMACSHA256(
 base64UrlEncode(header) + "." +
 base64UrlEncode(payload),
 secret)
```

## JWT 구조

JWT는 [Header Payload Signature] 각각 JSON 형태의 데이터를 *base 64* 인코딩 후 합친다.  
아래와 같은 순서로 . 을 이용해 합친다.  
최종적으로 만들어진 토큰은 HTTP 통신 간 이용되며, *Authorization* 이라는 *key*의 *value*로서 사용된다.



## JWT 인증 과정



## Part 5. Is JWT a Silver bullet?

그렇다면 모든 서비스들은 기존의 쿠키-세션 기반에서 웹 토큰 기반의 인증으로 변경해야할까?

### JWT 의 단점 & 도입시 고려사항

- **Self-contained** : 토큰 자체에 정보가 있다는 사실은 양날의 검이 될수 있다.
  - : 토큰 자체 payload 에 Claim set을 저장하기 때문에 정보가 많아질수록 토큰의 길이가 늘어나 네트워크에 부하를 줄 수 있다.
  - **payload** : payload 자체는 암호화 되지 않고 base64로 인코딩한 데이터다.
    - 중간에 payload를 탈취하면 디코딩을 통해 데이터를 볼 수 있다.
    - **JWE** 를 통해 암호화하거나, payload에 중요 데이터를 넣지 않아야 한다.
- **Stateless** : 무상태성이 때론 불편할 수 있다. 토큰은 한번 만들면 서버에서 제어가 불가능하다.
  - 토큰을 임의로 삭제할 수 있는 방법이 없기 때문에 토큰 만료시간을 꼭 넣어주는게 좋다.
- **token Token** : 토큰은 클라이언트 side에서 관리해야하기 때문에 토큰을 저장해야한다.

### Conclusion

HTTP, REST API 의 공통적인 특징은 **stateless**(무상태)를 지향한다는 것이다.

**Stateful**, 즉 상태를 저장하는 서버는 많은 **Side-effect**를 발생시킬 수 있다.

또한, 서론에 말했듯이 현재의 IT 인프라 구조는 유연한 확장 가능성이 있어야 하는데 기존의 쿠키-세션 기반의 인증을 사용하면 확장 가능한 인프라를 구성하기 힘들다.

기존의 로그인 / 인증을 모두 Web Token 기반으로 변경할 수는 없지만, 앞으로 만들게 될 서비스 특히 **RESTful**한 API의 인증에는 JWT를 사용해보는 것이 좋을 것이라 생각한다.

( JWT 인증 예제 - <https://github.com/SangHakLee/nodejs-jwt-example-ryan>).

### References

- [JWT] 토큰(Token) 기반 인증에 대한 소개
- [JWT] JSON Web Token 소개 및 구조
- [node.js] Token 기반 인증

- REST JWT(JSON Web Token)소개 - #1 개념 소개
  - JWT(JSON Web Token)에 대해서...
  - <https://jwt.io/introduction/>
- 

## [JWT] 토큰(Token) 기반 인증에 대한 소개

### 소개

토큰(Token) 기반 인증은 모던 웹서비스에서 정말 많이 사용되고 있습니다. 여러분이 API 를 사용하는 웹서비스를 개발한다면, 토큰을 사용하여 유저들의 인증작업을 처리하는것이 가장 좋은 방법입니다.

토큰 기반 인증 시스템을 선택하는데에는 여러가지 이유가 있는데요, 그 중 주요 이유들은 다음과 같습니다

#### Stateless 서버

Stateless 서버를 이해하려면 먼저 Stateful 서버가 무엇인지 알아야합니다. Stateful 서버는 클라이언트에게서 요청을 받을 때 마다, 클라이언트의 상태를 계속해서 유지하고,

이 정보를 서비스 제공에 이용합니다. stateful 서버의 예제로는 세션을 유지하는 웹서버가 있습니다.

예를들어 유저가 로그인을 하면, 세션에 로그인이 되었다고 저장해 두고, 서비스를 제공 할 때에 그 데이터를 사용하죠.

여기서 이 세션은, 서버컴퓨터의 메모리에 담을 때도 있고, 데이터베이스 시스템에 담을 때도 있습니다. Stateless 서버는 반대로, 상태를 유지 하지 않습니다. 상태정보를 저장하지 않으면, 서버는 클라이언트측에서 들어오는 요청만으로만 작업을 처리합니다.

이렇게 상태가 없는 경우 클라이언트와 서버의 연결고리가 없기 때문에 서버의 확장성 (Scalability) 이 높아집니다.

#### 모바일 어플리케이션에 적합하다

만약에 Android / iOS 모바일 어플리케이션을 개발 한다면, 안전한 API 를 만들기 위해선 쿠키같은 인증시스템은 이상적이지 않습니다. (쿠키 컨테이너를 사용해야하죠). 토큰 기반 인증을 도입한다면, 더욱 간단하게 이 번거로움을 해결 할 수 있습니다.

#### 인증정보를 다른 어플리케이션으로 전달

대표적인 예제로는, OAuth 가 있습니다. 페이스북/구글 같은 소셜 계정들을 이용하여 다른 웹서비스에서도 로그인 할 수 있게 할 수 있습니다.

#### 보안

토큰 기반 인증 시스템을 사용하여 어플리케이션의 보안을 높일 수 있습니다. 단, 이 토큰 기반 인증을 사용한다고 해서 무조건 해킹의 위험에서 벗어나는건 아닙니다.

#### 토큰 기반 인증 시스템을 사용하는 서비스들

토큰 기반 인증 시스템은 여러분이 알고있는 많은 서비스들에서 사용되고있습니다.

왜 토큰을 사용하게 되었을까?

토큰 기반 인증 시스템이 어떻게 작동하고, 또 이로 인하여 얻을 수 있는 이득에 대하여 알아보기전에, 이 토큰 기반 인증 시스템이 어쩌다가 나타났는지 알아보시다. 이를 위해선, 과거의 인증시스템이 어떤 방식으로 작동했는지 살펴볼 필요가 있습니다.

## 서버 기반 인증

기존의 인증 시스템에서는 서버측에서 유저들의 정보를 기억하고 있어야합니다. 이 세션을 유지하기 위해서는 여러가지 방법이 사용됩니다. 메모리 / 디스크 / 데이터베이스 시스템에 이를 담곤 하죠.

서버 기반 인증 시스템의 흐름을 보자면 다음과 같습니다.

이런 방식의 인증 시스템은 아직도 많이 사용 되고 있습니다. 하지만, 요즘 웹 / 모바일 웹 어플리케이션들이 부흥하게 되면서, 이런 방식의 인증 시스템은 문제를 보이기 시작했습니다. 예를 들자면, 서버를 확장하기가 어려웠죠.

## 서버 기반 인증의 문제점

### 세션

유저가 인증을 할 때, 서버는 이 기록을 서버에 저장을 해야합니다. 이를 세션 이라고 부릅니다. 대부분의 경우엔 메모리에 이를 저장하는데, 로그인 중인 유저의 수가 늘어난다면 어떻게될까요? 서버의 램이 과부화가 되겠지요? 이를 피하기 위해서, 세션을 데이터베이스에 시스템에 저장하는 방식도 있지만, 이 또한 유저의 수가 많으면 데이터베이스의 성능에 무리를 줄 수 있습니다.

### 확장성

세션을 사용하면 서버를 확장하는것이 어려워집니다. 여기서 서버의 확장이란, 단순히 서버의 사양을 업그레이드 하는것이 아니라, 더 많은 트래픽을 감당하기 위하여 여러개의 프로세스를 돌리거나, 여러대의 서버 컴퓨터를 추가 하는것을 의미합니다. 세션을 사용하면서 분산된 시스템을 설계하는건 불가능한것은 아니지만 과정이 매우 복잡해집니다.

## CORS (Cross-Origin Resource Sharing)

웹 어플리케이션에서 세션을 관리 할 때 자주 사용되는 쿠키는 단일 도메인 및 서브 도메인에서만 작동하도록 설계되어있습니다. 따라서 쿠키를 여러 도메인에서 관리하는것은 좀 번거롭습니다.

## 토큰 기반 시스템의 작동 원리

토큰 기반 시스템은 stateless 합니다. 무상태. 즉 상태유지를 하지 않는다는 것이죠.

이 시스템에서는 더 이상 유저의 인증 정보를 서버나 세션에 담아두지 않습니다.

이 개념 하나만으로도 위에서 서술한 서버에서 유저의 인증 정보를 서버측에 담아둠으로서 발생하는 많은 문제점들이 해소됩니다.

세션이 존재하지 않으니, 유저들이 로그인 되어있는지 안되어있는지 신경도 쓰지 않으면서 서버를 손쉽게 확장 할 수 있겠죠?

토큰 기반 시스템의 구현 방식은 시스템마다 크고작은 차이가 있겠지만, 대략적으로 보면 다음과 같습니다:

- 유저가 아이디와 비밀번호로 로그인을 합니다
- 서버측에서 해당 계정정보를 검증합니다.
- 계정정보가 정확하다면, 서버측에서 유저에게 signed토큰을 발급해줍니다.
- 여기서 signed 의 의미는 해당 토큰이 서버에서 정상적으로 발급된 토큰임을 증명하는 signature 를 지니고 있다는 것입니다
- 클라이언트 측에서 전달받은 토큰을 저장해두고, 서버에 요청을 할 때 마다, 해당 토큰을 함께 서버에 전달합니다.
- 서버는 토큰을 검증하고, 요청에 응답합니다.



웹서버에서 토큰을 서버에 전달 할 때에는, HTTP 요청의 헤더에 토큰값을 포함시켜서 전달합니다.

## 토큰의 장점

무상태(stateless) 이며 확장성(scalability)이 있다

이 개념에 대해선 지금 반복적으로 이야기하고 있죠? 이는 그만큼 토큰 기반 인증 시스템의 중요한 속성입니다.

토큰은 클라이언트사이드에 저장하기때문에 완전히 stateless 하며, 서버를 확장하기에 매우 적합한 환경을 제공합니다.

만약에 세션을 서버측에 저장하고 있고, 서버를 여러대를 사용하여 요청을 분산하였다면, 어떤 유저가 로그인 했을땐,

그 유저는 처음 로그인했었던 그 서버에만 요청을 보내도록 설정을 해야합니다. 하지만, 토큰을 사용한다면,어떤 서버로 요청이 들어가던, 이제 상관없죠.

## 보안성

클라이언트가 서버에 요청을 보낼 때, 더 이상 쿠키를 전달하지 않음으로 쿠키를 사용함으로 인해 발생하는 취약점이 사라집니다.

하지만, 토큰을 사용하는 환경에서도 취약점이 존재 할 수 있으니 언제나 취약점에 대비해야 합니다([참조](#)).

## Extensibility (확장성)

여기서의 확장성은, Scalability 와는 또 다른 개념입니다. Scalability 는 서버를 확장하는걸 의미하는 반면,

Extensibility 는 로그인 정보가 사용되는 분야를 확장하는것을 의미합니다. 토큰을 사용하여 다른 서비스에서도 권한을 공유 할 수 있습니다.

예를 들어서, 스타트업 구인구직 웹서비스인 [로켓펀치](#)에서는 Facebook, LinkedIn, GitHub, Google 계정으로 로그인을 할 수 있습니다.

토큰 기반 시스템에서는, 토큰에 선택적인 권한만 부여하여 발급을 할 수 있습니다 (예를들어서 로켓펀치에서 페이스북 계정으로 로그인을 했다면,

프로필 정보를 가져오는 권한은 있어도, 포스트를 작성 할 수 있는 권한은 없죠)

여러 플랫폼 및 도메인

서버 기반 인증 시스템의 문제점을 다룰 때 CORS 에 대하여 언급 했었죠? 어플리케이션과 서비스의 규모가 커지면,

우리는 여러 디바이스를 호환 시키고, 더 많은 종류의 서비스를 제공하게 됩니다. 토큰을 사용한다면, 그 어떤 디바이스에서도,

그 어떤 도메인에서도, 토큰만 유효하다면 요청이 정상적으로 처리 됩니다. 서버측에서 어플리케이션의 응답부분에 다음 헤더만 포함시켜주면 되지요.

`Access-Control-Allow-Origin: *`

이런 구조라면, assets 파일들(이미지, css, js, html 파일 등)은 모두 CDN 에서 제공을 하도록 하고, 서버측에서는 오직 API만 다루도록 하도록 설계 할 수도 있지요.

## 웹 표준 기반

토큰 기반 인증 시스템의 구현체인 JWT는 웹 표준 RFC 7519 에 등록이 되어있습니다.

따라서 여러 환경에서 지원이 되며 (.NET, Ruby, Java, Node.js, Python, PHP ...) 수많은 회사의 인프라스트럭처에서 사용 되고 있습니다 (구글, 마이크로소프트 ...)

마치면서..

이번 포스트에서는 토큰 기반 인증 시스템에 대한 소개, 그리고 기존의 시스템과 어떠한 차이가 있는지에 대하여 알아보았습니다.

다음 포스트에서는, 이번에 배운 시스템의 구현체인 JWT (JSON Web Token) 에 대해서 알아보고, 그 시스템에서 사용하는 토큰의 구조를 살펴해보도록 하겠습니다.

## Reference

Stateless vs Stateful Servers

[http://orca.st.usm.edu/~seyfarth/network\\_pgm/net-6-3-3.html](http://orca.st.usm.edu/~seyfarth/network_pgm/net-6-3-3.html)

Cookies vs Tokens. Getting auth right with Angular.JS

<https://auth0.com/blog/angularjs-authentication-with-cookies-vs-token/>

확장성 있는 웹 아키텍처와 분산 시스템

<http://d2.naver.com/helloworld/206816>

What is token based authentication?

<http://stackoverflow.com/questions/1592534/what-is-token-based-authentication>

REST API의 이해와 설계 #1-개념 소개

<http://bcho.tistory.com/953>

The Ins and Outs of Token Based Authentication

<https://scotch.io/tutorials/the-ins-and-outs-of-token-based-authentication#introduction>

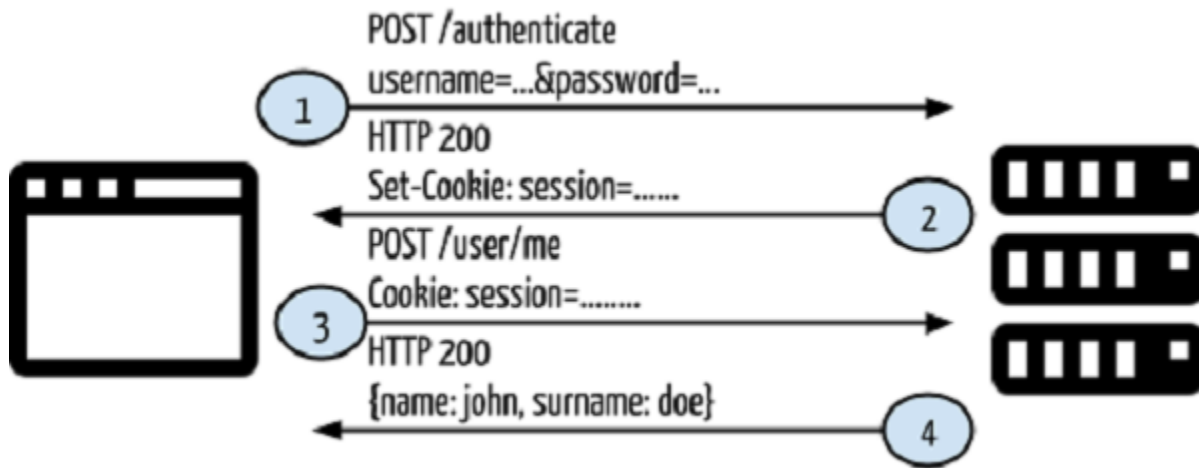
Token-based authentication: security considerations

<http://www.kieranpotts.com/blog/token-authentication-security>

## [node.js] Token 기반 인증

### 전통적인 인증 시스템

토큰 기반 인증 시스템을 설명하기 전에, 먼저 전통적인 인증 시스템을 살펴보자.



- 사용자는 username과 password를 로그인 폼에 입력하고 Login 버튼을 클릭한다.
- 서버는 요청이 들어오면 DB를 쿼리하여 user를 검증한다. 만약 요청이 유효하면 세션을 생성하고 세션 정보를 Response 헤더에 포함시켜 반환한다.
- 클라이언트는 제한된 end points에 접근할 때 모든 Request Header에 세션 정보를 포함시킨다.
- 만약 세션 정보가 유효하면 서버는 사용자가 특정 end point에 접근하는 것을 허용하고 렌더링된 HTML 내용을 반환한다.

여기까지는 문제가 없다. 웹 어플리케이션은 잘 동작하고, 사용자 인증을 거쳐서 특정 endpoint로의 접근을 제한할 수 있다.

하지만 안드로이드와 같은 다른 클라이언트에서 동작하는 어플리케이션을 만들고 싶다면 어떨까?

전통적인 인증 방식으로 동작하는 어플리케이션을 모바일 클라이언트에서도 동일하게 사용할 수 있을까? 그렇지 않다. 그 이유는 다음과 같다.

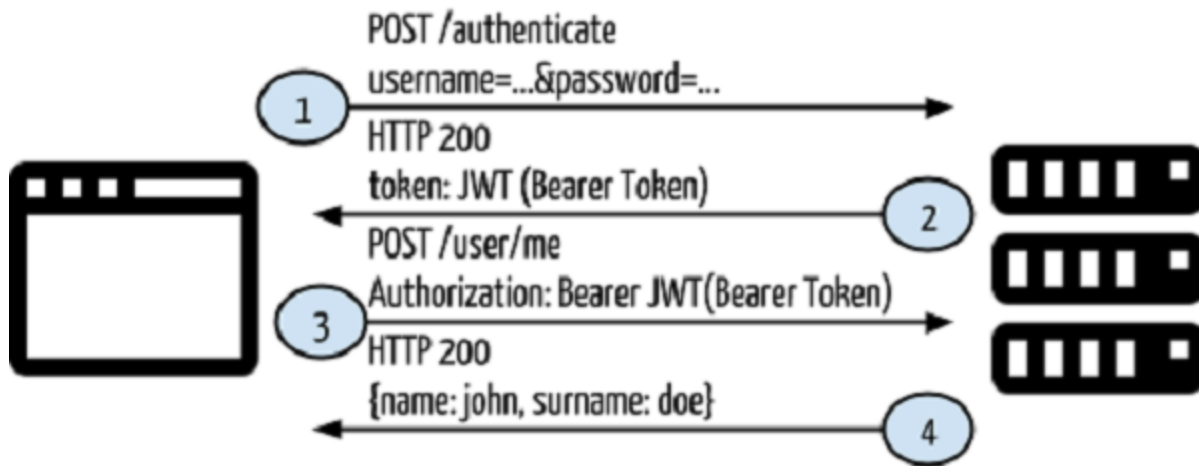
- 세션과 쿠키는 모바일 어플리케이션에서 make sense하지 않다. 서버에서 생성된 세션과 쿠키를 모바일 클라이언트에서는 공유할 수 없다.
- 현재 어플리케이션에서 렌더링된 HTML이 반환되었다. 모바일 클라이언트에서는 JSON 또는 XML과 같은 포맷의 응답이 필요하다.

이러한 경우 클라이언트 독립적인 어플리케이션이 필요하다.

### 토큰 기반 인증 시스템

토큰 기반 인증에서는 쿠키와 세션은 사용되지 않는다. 토큰은 서버로의 모든 요청에 대해 사용자를 인증하기 위해 사용된다.

위의 시나리오를 토큰 기반 인증 시스템으로 다시 설계해보자.



1. 사용자는 username과 password를 로그인 폼에 입력하고 Login 버튼을 클릭한다.
2. 서버는 요청이 들어오면 DB를 쿼리하여 user를 검증한다. 만약 요청이 유효하면 토큰을 생성하고 토큰 정보를 Response 헤더에 포함시켜 반환한다. 이로 인해 우리는 로컬 스토리지에 토큰을 저장할 수 있다.
3. 클라이언트는 제한된 end points에 접근할 때 모든 Request Header에 토큰 정보를 포함시킨다.
4. 만약 Request Header에 포함된 토큰이 유효하면 서버는 사용자가 특정 end point에 접근하는 것을 허용하고 JSON 또는 XML 포맷으로 응답한다.

그렇다면 JWT가 무엇일까?

## JWT

JWT는 JSON Web Token의 약자이며 인증 헤더 내에서 사용되는 토큰 포맷이다.

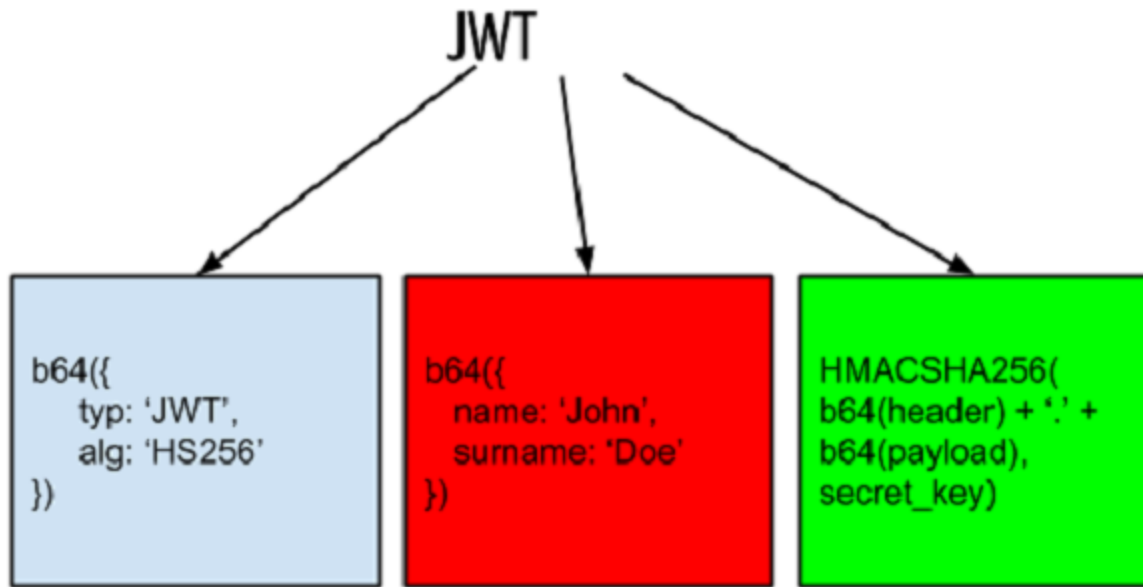
이 토큰은 두 개의 시스템끼리 안전한 방법으로 통신할 수 있도록 설계하는 것을 도와준다.

이 튜토리얼에서는 JWT를 "Bearer 토큰"으로 부르도록 하겠다. Bearer 토큰은 3가지 요소로 구성된다 : Header, Payload, Signature.

- Header는 토큰 타입과 암호화 방법을 보관하는 토큰의 한 부분이며, Base-64로 인코딩된다.
- Payload에는 유저 정보, 상품 정보 등의 다양한 종류의 정보를 넣을 수 있다. Base-64로 인코딩된다.
- Signature는 Header, Payload, Secret key의 조합이다. Secret key는 반드시 서버에 안전하게 보관되어야 한다.

JWT의 장점은 계정 서버와 API 서버가 분리되어 있을 때, API 서버가 계정 서버에게 토큰의 유효성 여부를 물어보지 않고도 스스로 판단할 수 있다는 것이다.

JWT 스키마와 토큰 예제는 아래와 같다.



```

eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6IjUzZTI4MjJiYmQxNzM1MDIwMDBINjZhMiIsIm5hbWUiOiJlW7xzZXlpbiBCYWwJhbGlzInVzZXJuYVw1IljoiaHVzZXIpbmJhYmFslwiw9sZSI6eyJoXRsZSI6ImFkbWluliwiYmloTWZFzayl6NHosImhhdCI6MTQxMTMzNjM1NSwiZXhwIjoxNDExMzU0MzU1fQ.GENSdtYlekoLEzMgdRSmoU1MTVENfMrBqXnaqO3HMHw

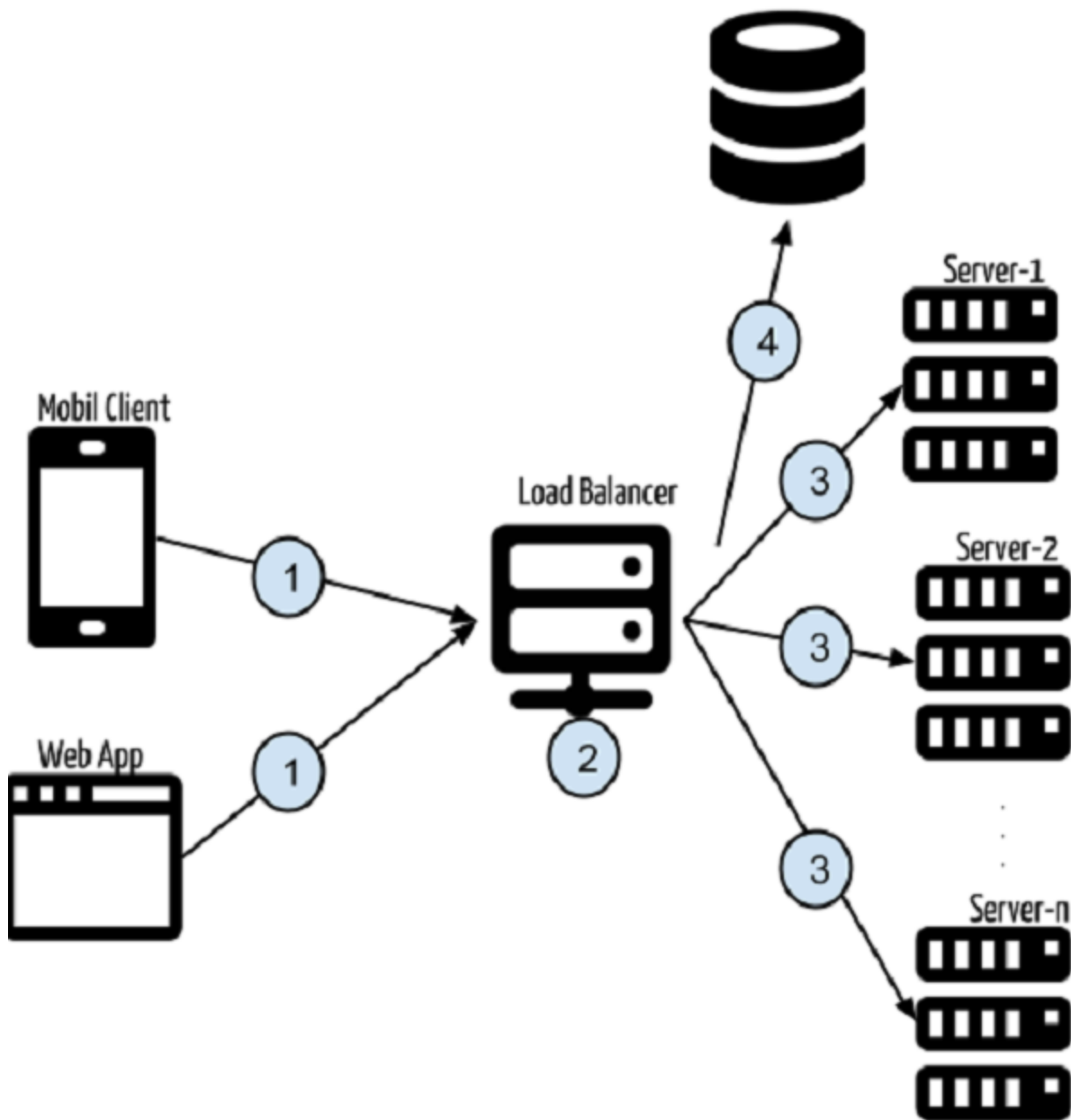
```

몇가지 언어에서 이미 구현되어 있기 때문에 Bearer 토큰을 생성하는 코드를 직접 구현할 필요는 없다.

| Language | Livbary URL                                                                                                                                                                           |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NodeJS   | <a href="http://github.com/auth0/node-jsonwebtoken">http://github.com/auth0/node-jsonwebtoken</a>                                                                                     |
| PHP      | <a href="http://github.com/firebase/php-jwt">http://github.com/firebase/php-jwt</a>                                                                                                   |
| Java     | <a href="http://github.com/auth0/java-jwt">http://github.com/auth0/java-jwt</a>                                                                                                       |
| Ruby     | <a href="http://github.com/progrium/ruby-jwt">http://github.com/progrium/ruby-jwt</a>                                                                                                 |
| .NET     | <a href="http://github.com/AzureAD/azure-activedirectory-identitymodel-extensions-for-dotnet">http://github.com/AzureAD/azure-activedirectory-identitymodel-extensions-for-dotnet</a> |
| Python   | <a href="http://github.com/progrium/pyjwt/">http://github.com/progrium/pyjwt/</a>                                                                                                     |

## 연습 예제

토큰 기반 인증에 대해 몇가지 기본 정보를 다뤘기 때문에 이제 연습 예제를 진행해보자. 아래 스키마를 잘 봐두길 바란다.



1. API에 대한 요청은 여러 클라이언트에 의해 생성된다. (웹 어플리케이션, 모바일 클라이언트, ...)
2. 요청은 <https://api.yourexampleapp.com>과 같은 서비스에게 전달된다. 많은 사람들이 어플리케이션을 사용하면 요청을 수행하기 위해 여러개의 서버가 필요할 것이다.
3. 여기서 가장 적합한 서버로 요청하기 위해 로드 밸런서가 사용된다. <https://api.yourexamplapp.com>으로 요청 발생하면 먼저 로드 밸런서가 요청을 제어하고 특정 서버로 클라이언트를 redirect 시킨다.
4. 하나의 어플리케이션이 있고 이 어플리케이션은 여러 서버로 배치된다 (server-1, server-2, ..., server-n). 요청이 만들어질 때마다 백엔드 어플리케이션은 요청 헤더를 가로채서 인증 헤더 내의 토큰 정보를 추출해낸다. DB 쿼리는 이 토큰을 사용하여 만들어질 것이다. 만약 토큰이 유효하고 end point에 접근하기 위해 요구되는 권한을 갖고 있다면 문제없이 진행될 것이다. 만약 그렇지 않다면 403 응답 코드(forbidden status)를 반환할 것이다.

## 장점

토큰 기반 인증은 심각한 문제들을 해결해주는 몇가지 이점을 갖는다. 그 중 일부는 아래와 같다.

- 클라이언트 독립적인 서비스. 토큰 기반 인증에서 토큰은 요청 헤더를 통해 전달된다. 이것은 stateless를 의미한다. HTTP 요청을 만들 수 있는 클라이언트라면 누구든지 서버로 요청을 보낼 수 있다.
- CDN. 대부분의 현재 웹 어플리케이션 내에서 view는 백엔드 상에서 렌더링되고 브라우저로 HTML이 반환된다. 프론트엔드의 로직이 백엔드 코드와 의존성이 있는 것이다. 이렇게 의존성이 생기면 몇가지 문제가 발생한다. 예를 들어, 프론트엔드 HTML, CSS, JS 등을 구현하는 디자인 에이전시와 함께 일을 한다고 가정해보자. 우리는 일부 렌더링 또는 생성 동작을 수행하기 위해 프론트엔드 코드를

가져와서 백엔드 코드에 통합시켜야 한다. 어쩌면 렌더링된 HTML 콘텐츠는 디자인 에이전시가 구현했던 것과 아주 많이 다를 것이다. 토큰 기반 인증에서는 프론트엔드 프로젝트를 백엔드로부터 독립적으로 개발하도록 할 수 있다. 백엔드 코드는 렌더링된 HTML 대신 JSON 응답을 반환할 것이고, 경량화되고 압축된 버전의 프론트엔드 코드는 CDN에 넣어둘 수가 있다. 누군가 웹페이지에 방문하면 HTML 콘텐츠는 CDN에서 제공되고 페이지의 내용은 인증 헤더의 토큰을 사용하는 API 서비스에 의해 생성될 것이다.

- No Cookie-Session (or NO CSRF). [CSRF\(사이트간 요청 위조\)](#)는 세션 유지에 일반적으로 사용되는 쿠키 정보만 만족하면 요청이 수행되는 취약점을 이용한 공격이다. 예를 들어, 이미 사이트에 로그인하여 쿠키를 들고 있는 사용자가 공격자가 유도한 링크(회원 탈퇴 링크)에 노출되면 회원 탈퇴 요청이 서버로 전달되고 탈퇴가 되어버린다. 이미 쿠키가 사용자 정보를 포함하고 있기 때문에 회원 탈퇴 요청 URL로 접속만하면 웹서버는 요청을 신뢰하고 명령을 수행하는 것이다. 이 문제를 해결하기 위해 탈퇴 시 비밀번호를 한번 더 요구하거나 요청에 토큰과 같은 credential을 포함하는 방법을 사용한다. 토큰 기반 인증에서 토큰은 인증 헤더 내에 포함되기 때문에 CSRF를 방지할 수 있다.
- 지속적인 토큰 저장. 어플리케이션 내에서 세션 읽기, 쓰기, 삭제 동작이 발생하면 최소 1회 OS의 temp 폴더에 file 관련 동작이 발생한다. 여러개의 서버를 갖고 있고 한 세션이 첫번째 서버에 생성되었다고 해보자. 이 상태에서 새로운 요청이 발생하고 그 요청이 다른 서버에 전달되면, 해당 서버에는 세션 정보가 없을 것이기 때문에 "unauthorized" 응답을 받을 것이다. 물론 sticky 세션(처음에 접속했던 서버와 같은 서버에 계속 연결시키는 것)를 사용하여 이 문제를 해결할 수 있다. 하지만 토큰 기반 인증에서는 이 문제 자연스럽게 해결된다. 요청 토큰은 모든 요청, 모든 서버가 가로채기 때문이다.

이것들이 토큰 기반 인증의 가장 일반적인 장점이다. 이로써 이론적인 설명을 마치고 연습 예제를 보도록 하자.

## 예제 어플리케이션

토큰 기반 인증을 시연하기 위해 2개의 어플리케이션을 살펴보자.

1. 토큰 기반 인증 백엔드 (서버)
2. 토큰 기반 인증 프론트엔드 (클라이언트)

백엔드 프로젝트 예제에서는 서비스가 실행되고 서비스의 결과가 JSON 포맷으로 반환될 것이다.

서비스에서는 결코 view가 리턴되지 않는다. 프론트엔드 예제는 HTML을 위한 AngularJS 프로젝트이며,

프론트엔드 앱은 백엔드 서비스로 요청을 보내는 AngularJS 서비스에 의해 작성될 것이다.

## 토큰 기반 인증 백엔드

./models/User.js

// mongoose를 사용하기 위해 해당 모듈을 import

var mongoose = require('mongoose');

// 스키마 정의

// email, password, token 필드를 가지며 각각의 필드는 string 타입이다.

var Schema = mongoose.Schema;

var UserSchema = new Schema({

email: String,

password: String,

token: String

});

// 스키마를 이용해서 모델을 정의

// 'User' : mongodb에 저장될 collection 이름(테이블명)

// UserSchema : 모델을 정의하는데 사용할 스키마

module.exports = mongoose.model('User', UserSchema);

./server.js

// 필요한 모듈 import

```
var express = require("express");
var morgan = require("morgan");
var bodyParser = require("body-parser");
var jwt = require("jsonwebtoken");
var mongoose = require("mongoose");
var app = express();
```

```
var port = process.env.PORT || 3001;
var User = require('./models/User');
```

// DB 연결

```
mongoose.connect(process.env.MONGO_URL); //mongodb://localhost/dbname
```

```
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.use(morgan("dev")); // 모든 요청을 console에 기록
```

```
app.use(methodOverride()); // DELETE, PUT method 사용
app.use(function(req, res, next) {
```

```
 //모든 도메인의 요청을 허용하지 않으면 웹브라우저에서 CORS 에러를 발생시킨다.
 res.setHeader('Access-Control-Allow-Origin', '*');
 res.setHeader('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE,OPTIONS');
 res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type, Authorization');
 next();
});
```

// 로그인

// 다른 endpoint에 접근할 수 있는 토큰을 얻는다.

```
app.post('/authenticate', function(req, res) {
 User.findOne({email: req.body.email, password: req.body.password}, function(err, user) {
 if (err) {
 res.json({
 type: false,
 data: "Error occurred: " + err
 });
 } else {
 if (user) {
 res.json({
 type: true,
 data: user,
 token: user.token
 });
 } else {
 res.json({
 type: false,
 data: "Incorrect email/password"
 });
 }
 }
 });
});
```

// 신규가입

// 계정과 토큰을 생성한다.

```
app.post('/signin', function(req, res) {
 User.findOne({email: req.body.email, password: req.body.password}, function(err, user) {
 if (err) {
 res.json({
 type: false,
 data: "Error occurred: " + err
 });
 } else {
 if (user) {
```



```

 res.json({
 type: false,
 data: "User already exists!"
 });
 } else {
 var userModel = new User();
 userModel.email = req.body.email;
 userModel.password = req.body.password;
 userModel.save(function(err, user) { // DB 저장 완료되면 콜백 함수 호출
 user.token = jwt.sign(user, process.env.JWT_SECRET); // user 정보로부터 토큰 생성
 user.save(function(err, user1) {
 res.json({
 type: true,
 data: user1,
 token: user1.token
 });
 });
 });
 }
}
});
});

```

// 나의 정보

// 토큰 검사 후 계정 정보 반환

// 토큰 추출하기 위해 ensureAuthorized 먼저 실행

```

app.get('/me', ensureAuthorized, function(req, res) {
 User.findOne({token: req.token}, function(err, user) {
 if (err) {
 res.json({
 type: false,
 data: "Error occurred: " + err
 });
 } else {
 res.json({
 type: true,
 data: user
 });
 }
 });
});

```

// 요청 헤더 내의 authorization 헤더에서 토큰 추출

// 토큰이 존재하면, 토큰을 req.token에 할당

```

function ensureAuthorized(req, res, next) {
 var bearerToken;
 var bearerHeader = req.headers["authorization"];
 if (typeof bearerHeader !== 'undefined') {
 var bearer = bearerHeader.split(" ");
 bearerToken = bearer[1];
 req.token = bearerToken;
 next(); // 다음 콜백함수 진행
 } else {
 res.send(403);
 }
}

```

```

process.on('uncaughtException', function(err) {
 console.log(err);
});

```

// Start Server

```

app.listen(port, function () {
 console.log("Express server listening on port " + port);
});

```

## 결론

클라이언트 독립적인 서비스를 구현할 때 토큰 기반 인증/인가 방식은 인증 시스템을 구축하는데 많은 도움을 준다.

이 기술을 사용함으로써 서비스(또는 API) 개발에만 집중할 수 있다. 인증/인가 부분은 토큰 기반 인증 시스템에 의해 서비스의 앞단에서 하나의 '레이어'로써 핸들링될 것이다.

웹 브라우저, 안드로이드, iOS, 데스크탑 클라이언트 등 어떤 클라이언트를 통해서도 서비스에 접근하고 서비스를 사용할 수 있다.

데모 사이트 : <http://token-based-auth.herokuapp.com/>

## 개선사항

1. 비밀번호가 해쉬 암호화되어 저장되어야하고 로그인시 해쉬값을 전달받아 비교해야 한다.

// bcrypt 사용

```
UserSchema.pre('save', function (callback) {
 // salt와 암호화된 비밀번호 생성

 var user = this;
 if (!user.isModified('password')) return next();

 bcrypt.genSalt(SALT_WORK_FACTOR, function(err, salt) {
 if (err) return next(err);

 bcrypt.hash(user.password, salt, function(err, hash) {
 if (err) return next(err);
 user.password = hash;
 next();
 });
 });
});
```

// 비밀번호 검증

```
UserSchema.methods.verifyPassword = function(password, cb) {

 bcrypt.compare(password, this.password, function(err, isMatch) {
 if (err) return cb(err);
 cb(isMatch);
 });
};
```

// 로그인

```
app.post('/authenticate', function(req, res) {

 ...

 user.comparePassword(password, function(isMatch) {
 if (!isMatch) {
 console.log("Attempt failed to login with " + user.username);
 return res.send(401);
 }
 });

 ...

}
```

2. 토큰의 유효성 검사가 수행되어야 하며, 토큰이 만료되어야 한다.

```
function ensureAuthorized(req, res, next) {
```

```
...
jwt.verify(bearerToken, process.env.JWT_SECRET)
...
}
```

3. 위의 개선사항들이 반영된 소스들은 다음 사이트들을 참고하길 바란다.

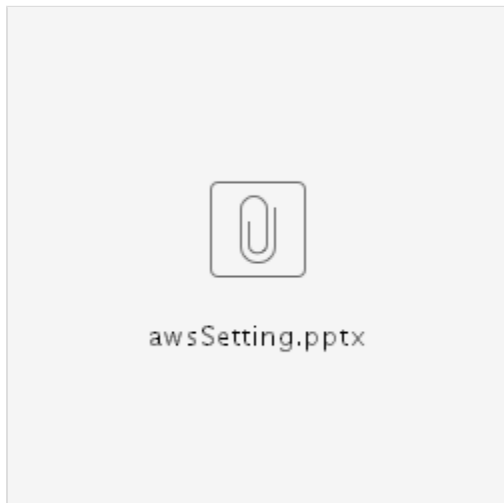
1. [Architecting a Secure RESTful Node.js app](#)
2. [Authentication with AngularJS and a Node.js REST api](#)

#### 참고자료

1. Using JSON Web Tokens with Node.js
2. mongoose ODM 을 이용한 MongoDB 연동
3. REST JWT(JSON Web Token)소개
4. MSA 아키텍처 구현을 위한 API 게이트웨이의 이해 (API GATEWAY)
5. OAuth 2.0 Compliant REST API

## AWS

### AWS signIn and Installation



## Git

### Git Configuration From Ubuntu 16.04 Server



Git server setting.pptx

git public key

shmoon@desktop

jslim public key

jslim@desktop

JIRA

Mastering Jira



Mastering JIRA.pdf

mysql

mySQL utf8

MySQL charset encoding

Charset과 Collation의 차이

- [Character Set General](#)
- [MySQL에서 문자셋과 Collation의 차이](#)

A character set is a set of symbols and encodings. A collation is a set of rules for comparing characters in a character set.

(character set) ( ) , Collation .

A=0, B=1, a=10, b=11 'A' '0' . 'A' 'a' .  
(case insensitive) collation 'A'='a' . Collation ''\_ci'' .

## 기본 설정

- UTF-8 설정 (Ubuntu : /etc/mysql/conf.d/encoding.cnf, RedHat /etc/my.cnf)

```
#
[client]
character-sets-dir=utf8

[mysqld]
init_connect=SET collation_connection = utf8_general_ci
init_connect=SET NAMES utf8
#default-character-set=utf8 # deprecated
character-set-server=utf8
collation-server=utf8_general_ci
```

- 이후 service mysql restart

## DB 테이블 생성시 문자셋 지정

참조 :MySQL and UTF-8

Database:  
(CREATE | ALTER) DATABASE ... DEFAULT CHARACTER SET utf8

Table:  
(CREATE | ALTER) TABLE ... DEFAULT CHARACTER SET utf8

## 현재 문자셋 정보 보기

```
show variables like 'c%';
```

결과 ->

```
character_set_client : utf8
character_set_connection : utf8
character_set_database : utf8
character_set_results : utf8
character_set_server : utf8
character_set_system : utf8
character_sets_dir : /usr/share/mysql/charsets/
collation_connection : utf8_general_ci
collation_database : utf8_general_ci
collation_server : utf8_general_ci
```

## 이미 생성된 데이터베이스의 문자셋 바꾸기

```
--
INFORMATION_SCHEMA.TABLES or SHOW CREATE TABLE [TABLENAME];
```

```
mysql> SET character_set_client = utf8;
mysql> SET character_set_results = utf8;
mysql> SET character_set_connection = utf8;
mysql> ALTER DATABASE [DB] DEFAULT CHARACTER SET utf8;
```

이미 데이터가 들어간 테이블의 문자셋 변환

```
create table test (merong varchar(20) collate latin1_general_ci);

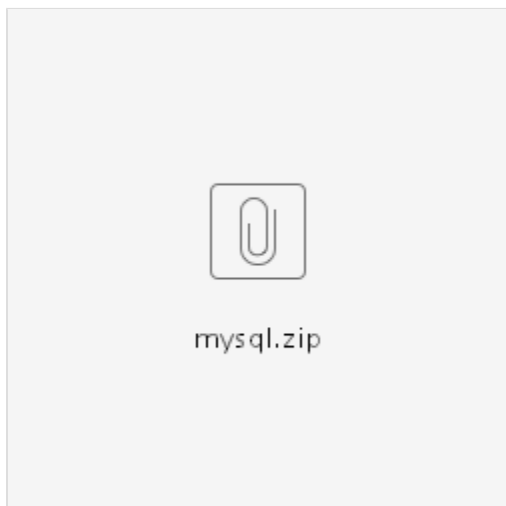
-- euckr
--

alter table test modify merong binary(100);
alter table test modify merong varchar(20) collate euckr_korean_ci;

-- binary
-- ().
```

참조 : [MySQL 4.1.8 RPM 설치시 한글설정](#). [Database.sarang.net](#)

mysql UTF-8 setting



```
[mysql]
default-character-set = utf8
```

```
[client]
default-character-set = utf8
```

```
[mysqld]
character-set-server = utf8
```

```
init_connect=SET collation_connection=utf8_general_ci
init_connect=SET NAMES utf8
collation-server=utf8_general_ci
transaction-isolation = REPEATABLE-READ
```

/var/lib/apt/lists/ppa.launchpad.net\_ondrej\_mysql-5.6\_ubuntu\_dists\_xenial\_main\_binary-i386\_Packages  
/var/lib/apt/lists/ppa.launchpad.net\_ondrej\_mysql-5.6\_ubuntu\_dists\_xenial\_main\_binary-amd64\_Packages  
/var/lib/apt/lists/ppa.launchpad.net\_ondrej\_mysql-5.6\_ubuntu\_dists\_xenial\_InRelease  
/var/lib/apt/lists/ppa.launchpad.net\_ondrej\_mysql-5.6\_ubuntu\_dists\_xenial\_main\_i18n\_Translation-en

## nodejs

### NVM installation



nodejsInstallation.pptx

## Restful

### RESTful Web Services



RESTful Web Services.pdf



RESTful Web Ser...es Cookbook.pdf

Restful 블로그자료



면접볼때 RESTful 한 개발을 할줄 아는사람들 우선시한다고합니다.

Restful 하게 , Restful API 이 뭔가요 ?

많이들들어봤을겁니다. 도대체 이게 뭔지

wiki를 찾아보면 도대체 뭔말인지 모르겠습니다.

# RESTful하게 라는건 ? ( 로다주님의 이야기를 빌려서 )

아키텍처를 말합니다. RESTful 하게 주세요 라는말부터 맞지않습니다.

면접시 혹은 RESTful한개발을 지향한다는건 웹의 특징을 잘 살린다는것을 말하고

이를 위한 아키텍처에 관한 몇가지 룰이 있습니다. 이것을 따른다면 RESTful하게 만든다고 볼수있습니다.

# 왜 RESTful하게 해야하나 ?

"표준" 을 따른다면 다른사람들이 내 코드를 이해 하기 쉬울 것 입니다.

# 현업에서 REST 라고 말하는것들은 ?

이미 당신이 하고 계신겁니다 XML 로 하고계시지않다면요. 현업에서 REST란 URL로콜해서 json형태로 주는 형태 지금 이미 사용하고 계신걸 말합니다. SOAP 는 XML로 기반으로합니다. 이것만해도 형태가 많이 다르다는것을 알수 있습니다.

#왜사용하는지를 알면 이해가쉽습니다.

이게 왜생겨났으면

쉽게 설명하면 스마트폰 개발을 하게되면서 급속도로 RESTful API 사용이 많아졌습니다. 이전에는 "개발자" 한명에서 DB도 만지고 화면도 제작했었습니다. 그런데 업무 양이 많아지고 분야가 넓어지면서 한 사람이 혼자 커버하기 어려워졌고 분업화가 필요 해졌고 백엔드 개발자가 DB접속권한등을 가진 코드등 분리가가능한것을 분리 했고 작업을 나누게 되었습니다.

2008년쯤부터 국내에서 "프론트개발자/백엔드개발자"라고 나누고 지금 처럼 불리우게 되었습니다. 프론트개발자에게 API url를 넘겨주면 그것가지고 "화면"을 제작하고 api url에 파라미터를 넣어주면 의도한 값들이 나오면서 화면에 뿌리게되었습니다.

# 이 게

원 장 점 이 있 으 면 이렇게 분리해서 개발하게되면 두명에서 동시에 개발이가능하므로 (1+1 =2 는아닙니다...) 업무속도를 높히게되고 분리가 되므로 코드가 꼬인다거나 다른 사고들을 미연에 방지 할수있게 되었습니다. 무진장 좋지않나요 ? 이걸 우리는 restful api 설계개발방식 이라고 합니다. RESTful 하게 개발한다는것을 생산성 측면에서 본다면 협업을 효율적으로 하는것과, 유지보수를 체계화 할수있다는것에서 점수를 높게 주게됩니다. (SOAP 이야기는 아예 빼버리겠습니다.)



#API 가뭔가요 ?

REStful하게는 알겠는 API는 뭔가 할수있는데요 현업에서는 "URL"와 거의 동어로 사용되고있습니다.

API 와 파라미터가 있어야 프론트에서 입히고 테스트 할수있거든요 이걸 API라고 하고 보통은 URL형태로 되어있어서

API 달라곤 합니다.

#구직시 restful api 아세요 라고물어보는건?

전 사실 의미없다고 생각합니다. 설계를 할정도의 사람이라면 당연히 알고있는것이고, 모른다고 하더라도 이제는 너무 일반화되어있어서 물어볼필요도 없다고 생각합니다.

다만 물어보는건 알고리즘 외적으로 개발인프라와 협업이런 관계들을 경험을 해보았는지를 확인하는차원으로서 의미가있다고봅니다..

참고: <http://www.smallake.kr/?p=21358>

#알고리즘 VS 개발인프라

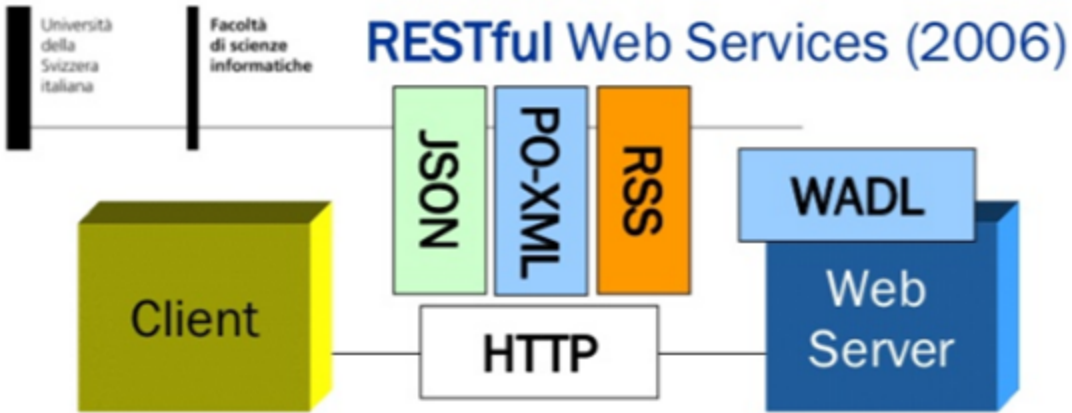
개발인프라는 사실 실무에서 사용하는거라면 금방 배우는것들이라 크게 중요하지는 않습니다. 마찬가지로 REStful API라는거를 잘몰라도 문제는 없다고 생각합니다. SOAP형태로 개발하는 사람은 없으니까요, 그런의미로서 디자인 패턴도 사실은 의미가없다고 생각합니다. 회사마다 "코드컨벤션"이 다를텐데 없을수도 있겠지만요. 협업을 해야만 하는환경이라면 디자인 패턴을 공부해서 적용 할필요는 없다고

생각합니다. 단, 코드컨벤션을 정의하고 설계를 해야하는 사람이라면 다르겠지요.

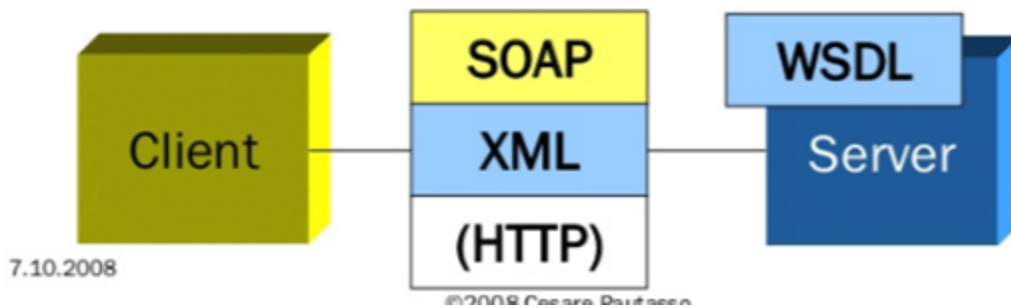
어쩌면 그런의미일까요 지금회사의 코드패턴이 구식이다가 아니라 디자인패턴을 공부함으로서 다른사람들의 코드를 존중하자는 취지 ? 글썄요 .. 전 불필요하다고 생각합니다. 코드 리뷰하는게 빨라질수는있겠네요 정답을 따라 한다가보다는 "협업을 위한" 더 나은 코드를 연구하는데 참고 자료가 될테니까요

#REStful API디자인및설계

이정도면 알면 restful api를 설계하고 개발하고 해야겠지요 그건 다른 문서들이 잘설명해주니 패스 하겠습니다.



## WS-\* Web Services (2000)



5

#RESTful API를 한줄로 요약하자면

"더좋은 설계방식"입니다.

"기존보다 가볍다" 라고 알면됩니다.

#Restful 뭐보다 좋은거지 ?

100% 좋다고 할순없지만 사용하기 편하고 사용점유율도 올라가고있으니

개발자들에게선호대상은 맞는것 같습니다.

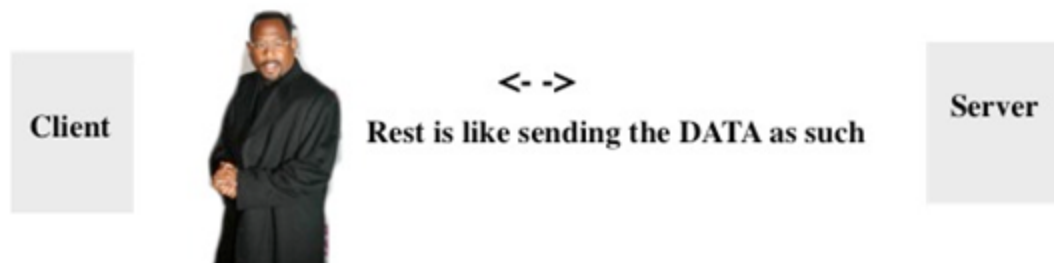
보통 SOAP와 비교를 많이합니다.

Consider "Martin Lawrence" as your data

## SOAP

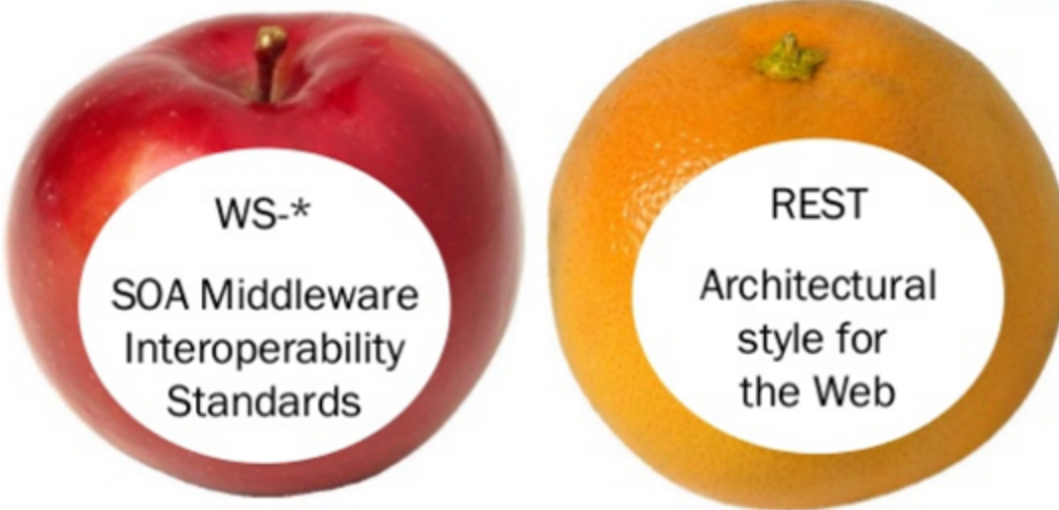


## REST



soap가 뭐냐면 위처럼 무거운놈입니다. 그리고 이렇게 표현을 하는걸봐선 개발자들이 REST를 정말 좋아하는걸알수 있을것 같습니다.

## Can we really compare WS-\* vs. REST?



7.10.2008

SOA Symposium 2008, Amsterdam  
©2008 Cesare Pautasso

9

#Restful api 뭐냐 라고 물어보면

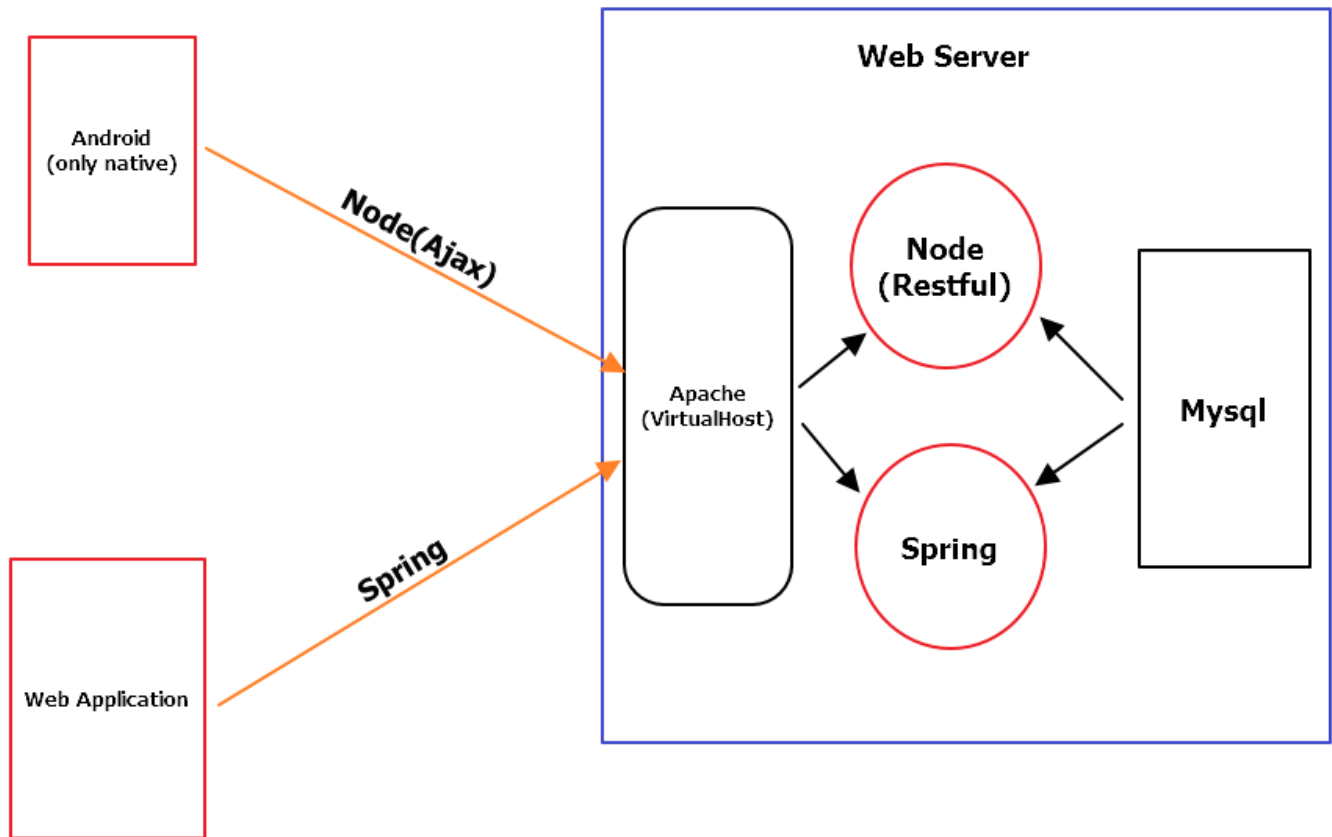
설계 방식인겁니다. 그리고 좋은겁니다. 가볍고, 개발자들이 좋아하고

경영자들은 개발속도를 올릴수있는 좋은 개발방법인겁니다.

[출처] Restful 하게 , Restful API 이 뭔가요 ? |작성자 세줄

## Server

### SMS Server Configuration



## 포워드 프록시(forward proxy) 리버스 프록시(reverse proxy) 의 차이

아파치 웹서버(apache web server) 에는 mod\_proxy 라는 프록시 기능을 하는 모듈이 내장되어 있고 이 모듈은 forward proxy 와 reverse proxy 두 가지 기능을 다 수행한다.

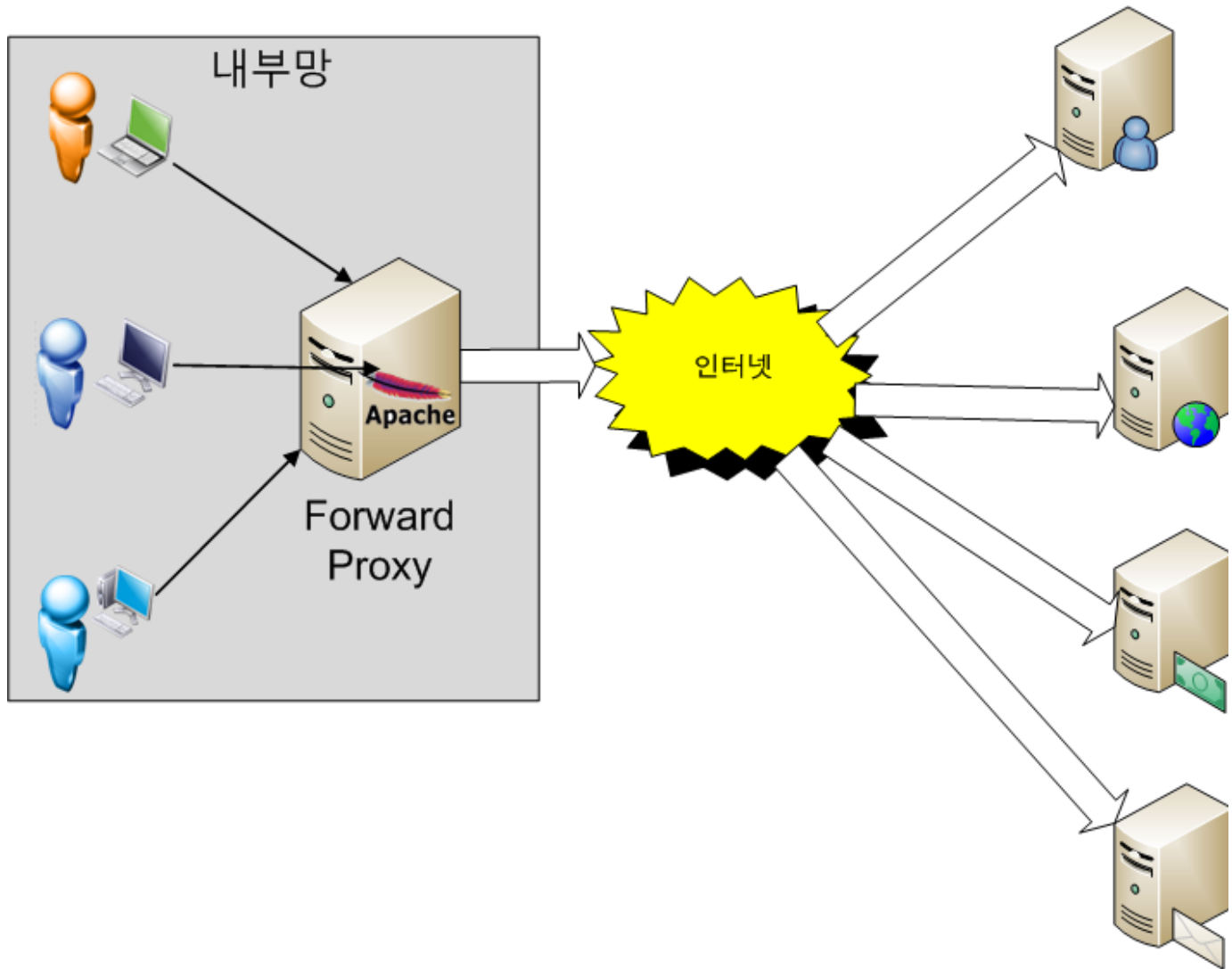
그러면 forward 와 reverse 방식의 차이점은 무엇일까.

아래 그림을 보면 쉽게 이해할 수 있을 것이다.

### Forward Proxy

클라이언트가 [example.com](http://example.com) 에 연결하려고 하면 사용자 PC 가 직접 연결하는게 아니라 포워드 프록시 서버가 요청을 받아서 [example.com](http://example.com) 에 연결하여 그 결과를 클라이언트에 전달(forward) 해 준다.

포워드 프록시는 대개 캐싱 기능이 있으므로 자주 사용되는 콘텐츠라면 월등한 성능 향상을 가져올 수 있으며 사용자의 정해진 사이트만 연결할수 있는등 웹 사용 환경을 제한할수 있으므로 기업 환경등에서 많이 사용한다.



## Reverse Proxy

클라이언트가 [example.com](http://example.com) 웹 서비스에 데이터를 요청하면 Reverse Proxy는 이 요청을 받아서 내부 서버에서 데이터를 받은 후에 이 데이터를 클라이언트에 전달하게 된다.

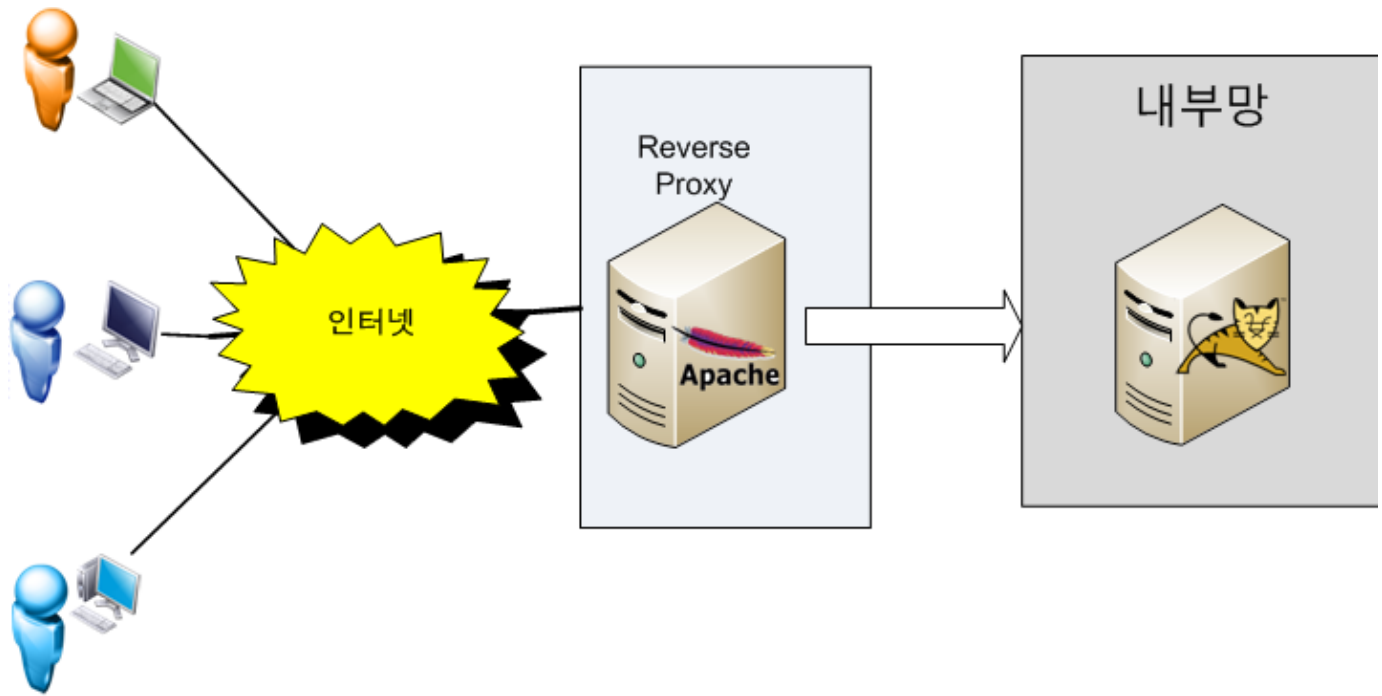
내부 서버가 직접 서비스를 제공해도 되지만 이렇게 구성하는 이유는 보안때문이다.

보통 기업의 네트워크 환경은 DMZ 라고 하는 내부 네트워크와 외부 네트워크 사이에 위치하는 구간이 존재하며 이 구간에는 메일 서버, 웹 서버, FTP 서버등 외부 서비스를 제공하는 서버가 위치하게 된다.

[example.com](http://example.com) 사는 서비스를 자바로 구현해서 WAS 를 DMZ 에 놓고 서비스해도 되지만 WAS 는 보통 DB 서버와 연결되므로 WAS 가 최전방에 있으면 WAS 가 털릴 경우 DB 서버까지 같이 털리는 심각한 문제가 발생할 수 있다.

이때문에 리버스 프락시 서버를 두고 실제 서비스 서버는 내부망에 위치시키고 프락시 서버만 내부에 있는 서비스 서버와 통신해서 결과를 클라이언트에게 제공하는 방식으로 서비스를 하게 된다.

특히 리눅스 환경이라면 리버스 프락시로 아파치 웹 서버를 사용한다면 SELinux 를 켜 놓으면 SELinux 의 기본 정책이 웹 서버는 톱켓의 8080, 8009 포트만 접근 할 수 있으므로 아파치 웹 서버가 해킹당해도 웹 서버 권한으로는 내부망으로 연결이 불가하다.



Reverse Proxy 로 서비스 제공시 WAS 에서 REMOTE\_ADDR 을 가져오면 Reverse Proxy 서버의 IP 를 얻게 되므로 원하는 결과가 나오지 않는다.

Proxy(프락시) 환경에서 client IP 를 얻기 위한 X-Forwarded-For(XFF) http header 를 참고해서 XFF 헤더를 사용하자.

## 공공기관 API 특일 서비스

<https://www.data.go.kr/dataset/15012690/openapi.do>

## 메일보내기 관련 Gmail

test 계정

account : smsprojectleader@gmail.com

password : biztech112

security 설정 → connection app → 낮은 수준의 보안 연결 허용

## Library

## Meeting

## 비트 프로젝트 07/10 (월)

Jira , Confluence , Jenkins, git 설치 및 세팅 완료

미팅 진행 ( 5:45 ~ 6:25 )

팀명

– SASI (System Admin & System Integration)

#### 프로젝트명

- SMS (Sales Management System)

#### 파트 나누기

- 양승근 - Spring (Web 작업 , FrontEnd , BackEnd)
- 문성훈 - Node (RestFull API 서버)
- 임진숙 - Android

#### +α

- 엑셀 , CSV 등 데이터 내보내기
- 추가적으로 하나 더 구현하기

#### 과제 ( Due Date -> 대략 07/12(수) 오후 13:00 까지 )

- SASI 프로젝트의 큰틀 구상하기

#### EX

- UI ,UML , Site Tree , DB 구성, 데이터 전달 방식, 암호화(보안)방식 등등. . .

#### Why?

- 서로가 구상하는 모습이 다르고 구상을 해봐야 대화가 되니깐~~

#### 메모 사항

- 성훈씨 ( 07/13(목) 예비군 훈련으로 인하여 자리 비움 )

## 비트 프로젝트 07/11 (화)

### 기획 ( 총 2주 - 10일 )

1. 요구사항 분석 (1PD)
2. 벤츠마킹 (1PD)
3. UI + UML (5PD)
4. DB설계 (3PD)

---

## JIRA

프로젝트 명 - SMS

팀 이름 - SASI

---

## workflow

1. ToDo - Daily 로 일기 형식으로 본인이 한 업무를 기록하여 작업의 시간을 측정하는 Issue Type ( 낭비하는 시간을 파악하고 개발의 힘을 쏟음)
  2. Bug - 개발 진행시 ( weekly release ) Test후 발생하는 BUG Issue를 관리하는 Issue Type
  3. Task - 일반적으로 진행되어야 하는 업무들을 주로 다루는 Issue Type ( ex > 서버작업 , 기획 등등 ...)
- 

## Confluence



1. 회의록 작성 진행 - 주로 회의후 PM(양승근)이 진행하지만 부재시 또는 특별한 경우는 팀원중 한명이 작성 진행
  2. Study 및 개발시 습득한 자료 update - 개발진행에 있어 서로 공유 하여야 하거나 세미나를 진행함에 있어 자료들을 공유하는 차원.
  3. 개발 라이브러리 사용시 기록 - 라이브러리 관련하여 기업용 라이선스를 제공하지 않는 경우도 있기에 (라이선스 정책, 사용법, 사용 버전) 등등을 등록함
- 

## 회의 내용

+α

(엑셀 + CSV) 는 기본 기능으로 들어감

기존에 가지고 있던 데이터를 기반으로 (약 한달이라고 하면) 다음달의 데이터를 예측 하는 기능(그래프로 뿌려줌)

LDAP 연동 방식 지원을 통하여 AD로 부터 사용자의 정보를 손쉽게 가져오는 기능

## 그래프 관련

그래프를 유동적으로 또는 관리자나 사용자에게 맞도록 설정하게 하는 UI

## 금일 진행할 작업

벤츠 마킹 및 요구사항 분석 부분을 진행

오늘 같이 토의한 내용을 바탕으로 좀더 정교하게 기획을 다져서 내일 다시 미팅하기로 결정

# 비트 프로젝트 07/12 (수)

## 진행 상황

07/11(월) 시작 ~ 07/12(수) = 3PD

1. 요구사항 분석, 벤츠마킹 완료 (2PD)
2. UI + UML 부분 작업 진행중 (1PD)

기획 부분 산정 기간과 오차가 없도록 작업 진행 중

## 기획

1. 요구사항 분석 - 07/11 보다 좀더 구체적인 형태를 분석 (100%)
2. 벤츠마킹 - 참조할 사이트 또는 app을 조사 (100%)
3. UI + UML - 사양서에 나와있는 간단한 tree 구조를 추출 및 UI 뼈대 설계 (20%)

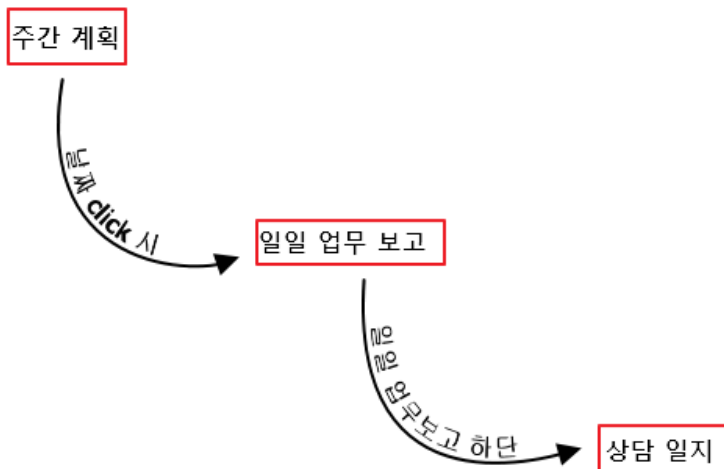
## 추가 계획

1. 07/14(금) 오후 2시까지 UI 뼈대 디자인 + Site Tree 구성한 후 회의 진행
  2. 07/14(금) UI + Site Tree 부분을 완전히 정의 하기로 결정 → 색상 까지 결정
  3. 07/17(월) 프로토타입 (카카오 오픈APP으로) 만들기 → (모바일 , WEB) 각각 파트를 나누어서 만들기
  4. 07/17(월) jenkins + Apache tomcat + Maven 으로 web 환경 구성
- 

## 회의 내용

각자 준비해온 UI + UML , 벤츠마킹 관련하여 발표를 진행

## Tree 구조



## 메뉴구성

1. Dashboard (데이터 표)
  2. Calendar (달력)
  3. Interview (상담 일지)
- 

## Dashboard

그래프로 데이터를 뿌려주는 부분 (통계 , 예측 통계)

매출 , 실적 , 주행거리 등에 대한 통계를 나타내는 부분

이때 통계 부분은 사용자가 (주 , 월 , 분기 , 년도) 별로 원하는 통계를 볼수 있도록 설정을 할수 있으며

설정된 내용은 DB에 저장되어 항상 유지가 되며 설정변경시 다시 그래프를 뿌려 주는 부분은 Ajax로 구현

이부분은 Rest를 이용해서 Node 서버로 부터 데이터를 전달 받음.

팀장은 직원들의 실적을 그래프로 볼수 있도록 해야함 ( 직원들은 자신의 실적과 팀별 실적만 그래프로 볼수 있음 )



## Calendar

Javascript를 이용해서 달력 기능을 넣은 화면 으로써

날짜를 Click 시 일일 업무 일지로 바로 갈수 있다.

라이브 러리 참조 사이트 URL → <https://fullcalendar.io/>

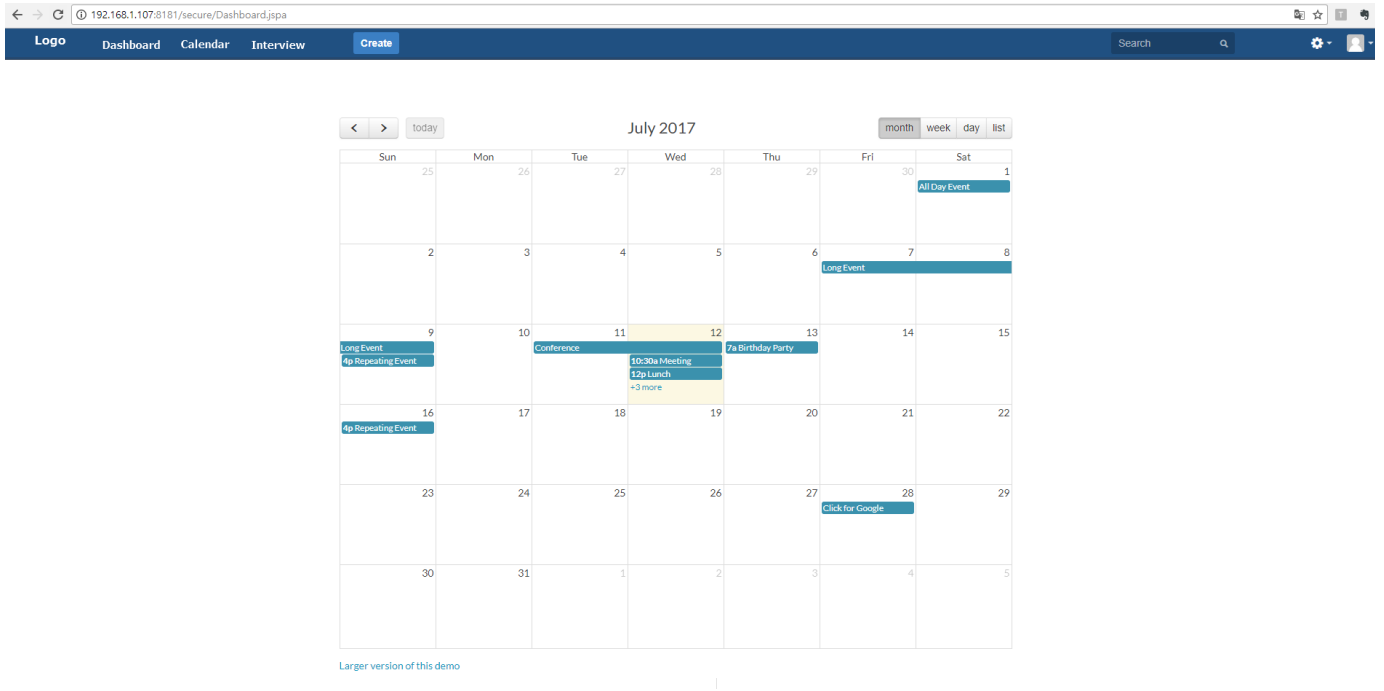
### 해당 페이지 직급별 기능

#### 사원

- 주간 계획 , 일일 업무 보고서 ( 작성 및 수정 )
- 사원은 보고서를 작성시 바로 결제를 올리지 않을수 있기에 보고서 기능에 (저장, 결제) 등의 상태가 있어야함.
- 사원이 결제를 누르는 경우 팀장에게 메일이 발송이 됨 이때 메일은 HTML 문서로 발송이 되며 (JIRA의 메일을 참조 )
- 메일 내용안에 사이트 바로가기 버튼을 넣어서 사이트로 접근이 가능하게 설정
- 일일 보고서 작성시 하단에 상담일지를 추가 할 수 있음 (이부분은 상담일지를 첨부 파일 처럼 추가 할수 있도록 설정 → Ajax 사용)

#### 팀장

- 팀장으로 로그인시 사원에 따른 일일 업무 보고서나 주간 계획을 봐야 하므로 사원에 따라서 검색을 할수 있도록 검색 기능이 필요함.
- 팀장으로 로그인시 일일 보고서에 코멘트 추가 기능 및 승인하기 버튼이 보여짐 .



아래 화면은 Jira에 원하는 유저를 검색시 자동으로 찾아주는 부분 (Ajax 사용)

이처럼 일일 업무 보고서 작성시 상담일지를 추가 할 수 있도록 input 칸을 만들고

해당 부분에 상담일지의 키를 검색하면 등록을 할 수 있도록 설정 (이것은 Jira의 Ticket 마다 고유의 key 가 있는것 처럼)

한번 링크가 걸린 상담 일지는 ( 일일 보고서와 연결이 된 ) 검색이 안되도록 설정

Ajax를 사용한다는 것은 Rest 부분에서 처리를 해줘야 함.


## People

Assignee:

 Unassigned

Reporter:

### Suggestions

 Yang SeungGeun - tmdrms0817@naver.com (sgyang)

Votes:

 Automatic


Watchers:


*Start Typing for Users*


### All Users

 admin - admin@jx372.com (admin)

 Lee HarkSu - leehacksue@naver.com (hslee)

 Lim JinSuk - dlawlstnr21@hanmail.net (jslim)

 No KyungWook - nogang0618@naver.com (kwno)

 Moon SeongHoon - akkgkm@naver.com (shmoon)

 Park YeongSeok - ys40013339@gmail.com (yspark)

## Dates

Created:

Updated:

## People

Assignee:

sg

Reporter:

### Suggestions

 Yang SeungGeun - tmdrms0817@naver.com (sgyang)

Votes:

0

Watchers:

1 Stop watching this issue

## Interview

상당 일지를 조회 및 수정이 가능하도록 하는 부분으로 List 형식이며

영업 사원이 일일 업무 보고서를 작성시에만 상당 일지를 관리 할수 있다면 매우 불편하므로

영업사원이 상담시 바로 작성이 가능하게 따로 관리를 진행

| T                                   | Key    | Summary                   | Assignee       | Reporter       | P ↓ | Status | Resolution | Created   | Updated   | Due |
|-------------------------------------|--------|---------------------------|----------------|----------------|-----|--------|------------|-----------|-----------|-----|
| <input checked="" type="checkbox"/> | TEST-8 | 요요요                       | Unassigned     | Yang SeungGeun | ↑   | TO DO  | Unresolved | 11/Jul/17 | 11/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-7 | 테스트                       | Unassigned     | Yang SeungGeun | ↑   | TO DO  | Unresolved | 11/Jul/17 | 11/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-1 | This is your first task   | Yang SeungGeun | Yang SeungGeun | ↑   | TO DO  | Unresolved | 06/Jul/17 | 11/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-2 | Workflows and statuses    | Yang SeungGeun | Yang SeungGeun | ↑   | TO DO  | Unresolved | 06/Jul/17 | 06/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-3 | Editing tasks             | Yang SeungGeun | Yang SeungGeun | ↑   | TO DO  | Unresolved | 06/Jul/17 | 06/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-4 | Searching for information | Yang SeungGeun | Yang SeungGeun | ↑   | TO DO  | Unresolved | 06/Jul/17 | 06/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-5 | Keyboard shortcuts        | Yang SeungGeun | Yang SeungGeun | ↑   | TO DO  | Unresolved | 06/Jul/17 | 06/Jul/17 |     |
| <input checked="" type="checkbox"/> | TEST-6 | What's next?              | Yang SeungGeun | Yang SeungGeun | ↑   | TO DO  | Unresolved | 06/Jul/17 | 06/Jul/17 |     |

+α

1. (엑셀 + CSV) 는 기본 기능으로 들어감
2. 메일 기능을 추가 하여 사용자들에게 알림을 진행
3. 기존에 가지고 있던 데이터를 기반으로 (약 한달이라고 하면 ) 다음달의 데이터를 예측 하는 기능(그래프로 뿌려줌)
4. LDAP 연동 방식 지원을 통하여 AD로 부터 사용자의 정보를 손쉽게 가져오는 기능

## 과제

1. 07/14(금) 오후 2시까지 UI 뼈대 디자인 + Site Tree 구성한 후 회의 진행

## 비트 프로젝트 07/14 (금)

### 진행 상황

07/13(목) 시작 ~ 07/14(금) = 2PD

1. Web Site UI + Site Tree
2. 모바일 UI + Site Tree

금요일 오후 Web UI를 통일화 하여 큰 틀은 맞춤.

### 회의 내용

로그인 (일반적인 로그인 UI)

- ID , PW 입력 받아서 로그인
- 권한은 " 관리자 , 팀장 , 팀원 " 총 3단계

## 상단 메뉴 (Jira의 상단 메뉴와 유사함)

- Logo (클릭시 홈으로 이동할수 있도록 링크 처리)
- 버튼 형식으로 "Dashboard , Calender, Interview" 페이지 버튼을 생성
- Create 버튼 → 모든 페이지 생성은 create 버튼으로 할수 있도록 만듦 (Dialog 방식 ) → 해당 page 별로 default 설정을 해줘야함 (데이터를 전달 받아서)
- 설정 버튼 → 관리자로 로그인시 보여지는 버튼으로 click시 관리자 페이지로 이동
- 프로필 버튼 → 자신의 계정의 (사용자 지정 홈페이지, 이메일 , 성명, 전화번호 등등) 을 관리할수 있는 버튼

## Dashboard ( View 부분만 수행하는 페이지 )

- Calender → ( Click 시 Calender 페이지로 이동 )
- Graph → (Click 시 Graph 상세 페이지로 이동)
- 상담일지 - ( 해당일의 상담일지를 list 형식으로 뿌려줌 , Jira의 프레임 틀을 참조 → click시 상담일지 상세 popup이 뜸 )

## Calender → javascript 라이브러리 사용

- mon → Default 화면이며 방향버튼 클릭시 월 전환이 가능하며 , 일일을 클릭시 일일 업무 보고 dialog 화면이 뜸
- week → list 형식으로 주간 데이터를 뿌려줌 week 화면에서 click 시 주간 계획을 수정할수 있도록 처리 ( mysql 에 일일 단위로 컬럼을 생성할수 있도록 처리)
- export → 버튼을 누르면 엑셀로 데이터를 뽑아 낼수 있음 , 데이터는 월단위로 뽑아 낼수 있음.

## 상담 일지

리스트 형식인데 검색 기능을 제공하고 list 는 25개를 한정하여 페이지징 기능을 제공한다.

상담일지 상세 내용은 popup을 이용해서 페이지 전환이 없도록 만든다.

## Graph ( 그래프 )

데쉬보드에서 그래프를 누르면 해당 페이지로 들어올수 있음

검색 부분에 Type 에서 "매니저 , 지점별" 두가지를 선택가능

매니저를 선택시 담당자 검색 박스가 생성됨

지점별 선택시 지점 검색 박스가 생성됨

기준을 통해서 예측 통계인지 일반 통계인지 설정 가능

기간을 통해서 "주,월,분기,년" 순으로 검색이 가능

매니저 선택시 그래프는 "월간 실적 , 계량거리 , 상담시간" 등을 그래프로 나타냄

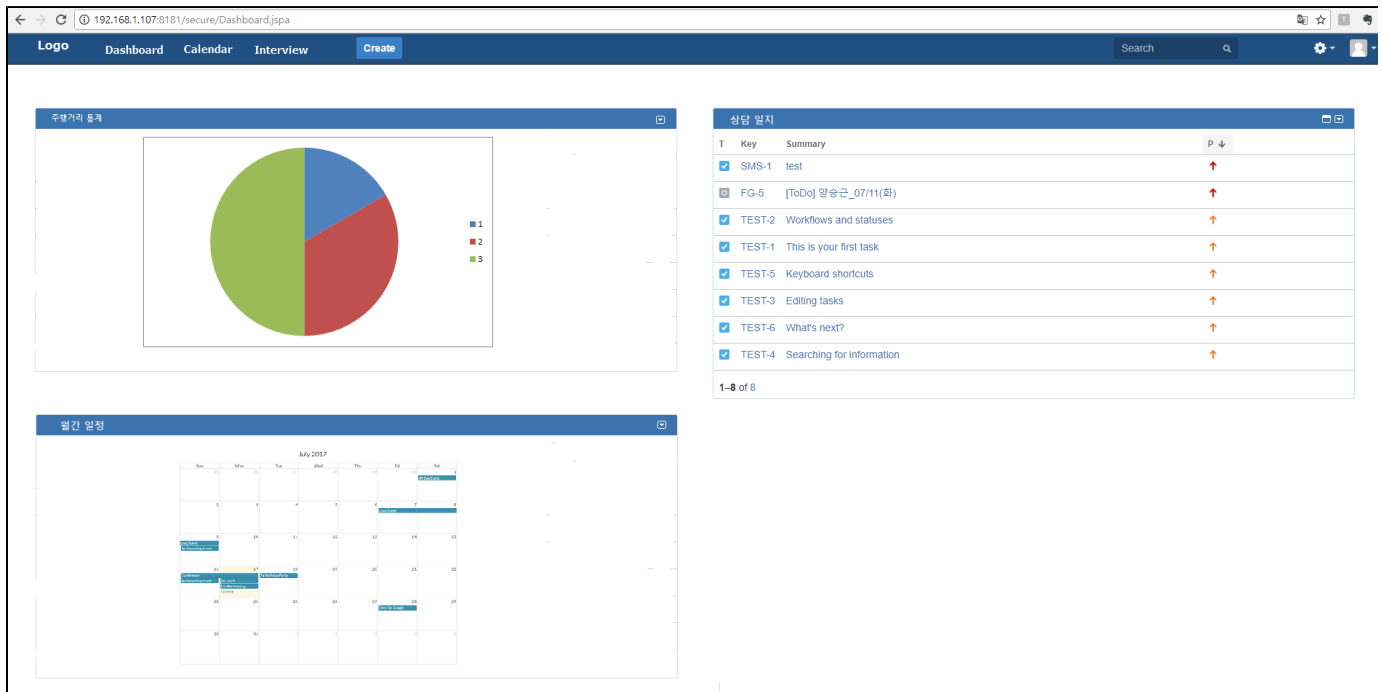
지점별일 경우 우리가 설정한 매출액 대비 어느 정도 실적을 냈는지 그래프로 통계를 보여줌

그래프는 mouse over 시 jquery 로 데이터를 표 형식으로 뿌려줌

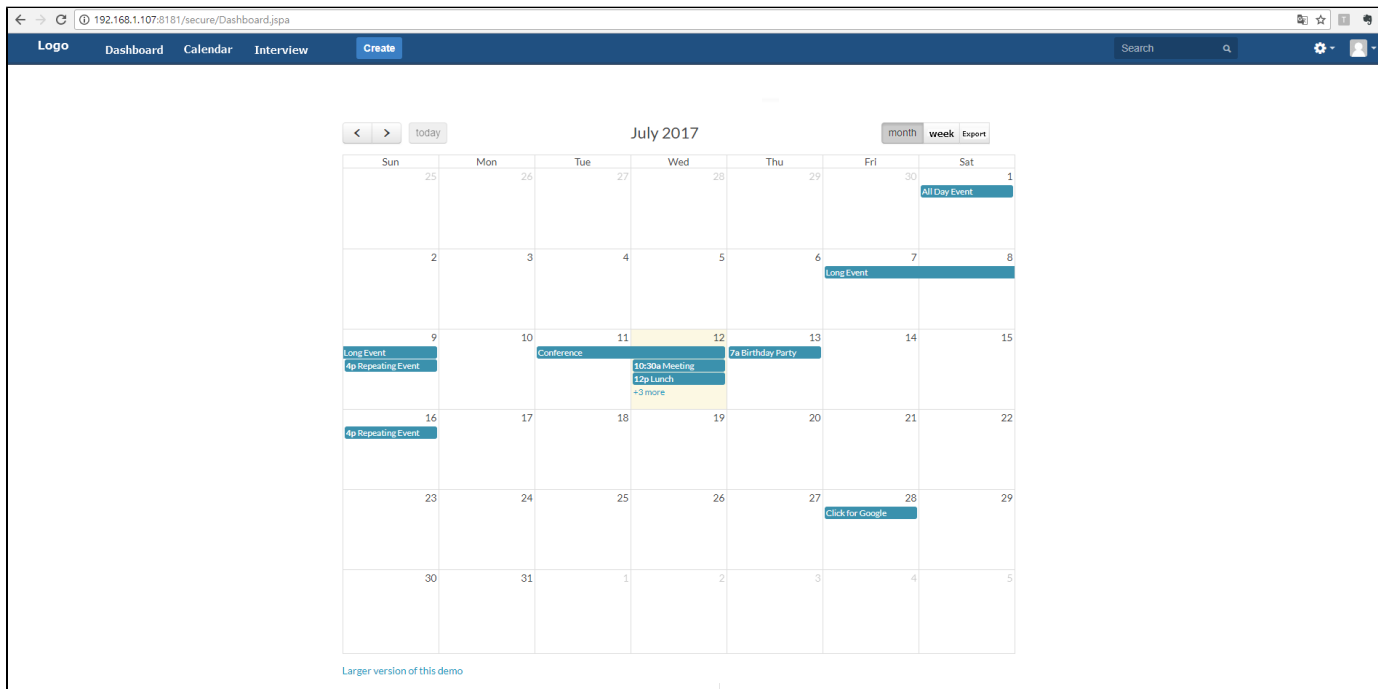
## 관리자 페이지

- 유저 관리
- 그룹 관리 (팀 관리)
- 고객 관리 (ex> 동작 대리점 → 최경호)
- LDAP 기능 on , off 설정
- CSV 데이터 import 기능

## Dashboard



## Calendar



## Interview



Logo
Dashboard
Calendar
Interview
Create
Search

소속: 영업 1팀
성명: 천영진
1차 거래선: 동작 테라점
2차 거래선: 동작 슈퍼
제목: [상당 일지]

Created Date: All

Within the last minutes
More than minutes ago
Between 11-Jan-2012 and 30-Jan-2012
In range -3w 4d to 3w 4d

Update Close

1-8 of 8

| T                                   | Key    | Summary                   | Assignee       |         | Created   | Updated   | Due        |
|-------------------------------------|--------|---------------------------|----------------|---------|-----------|-----------|------------|
| <input checked="" type="checkbox"/> | TEST-8 | 프로젝트                      | Unassigned     |         | 11/Jul/17 | 11/Jul/17 |            |
| <input checked="" type="checkbox"/> | TEST-7 | 테스트                       | Unassigned     |         | 11/Jul/17 | 11/Jul/17 |            |
| <input checked="" type="checkbox"/> | TEST-1 | This is your first task   | Yang SeungGeun |         | 06/Jul/17 | 11/Jul/17 |            |
| <input checked="" type="checkbox"/> | TEST-2 | Workflows and statuses    | Yang SeungGeun | ↑ TO DO | 06/Jul/17 | 06/Jul/17 | Unresolved |
| <input checked="" type="checkbox"/> | TEST-3 | Editing tasks             | Yang SeungGeun | ↑ TO DO | 06/Jul/17 | 06/Jul/17 | Unresolved |
| <input checked="" type="checkbox"/> | TEST-4 | Searching for information | Yang SeungGeun | ↑ TO DO | 06/Jul/17 | 06/Jul/17 | Unresolved |
| <input checked="" type="checkbox"/> | TEST-5 | Keyboard shortcuts        | Yang SeungGeun | ↑ TO DO | 06/Jul/17 | 06/Jul/17 | Unresolved |
| <input checked="" type="checkbox"/> | TEST-6 | What's next?              | Yang SeungGeun | ↑ TO DO | 06/Jul/17 | 06/Jul/17 | Unresolved |

1-8 of 8

## Graph

Logo
Dashboard
Calendar
Interview
Create
Search

Type: 매니지
Assignee: All
기준: 예측 통계
기간: 월

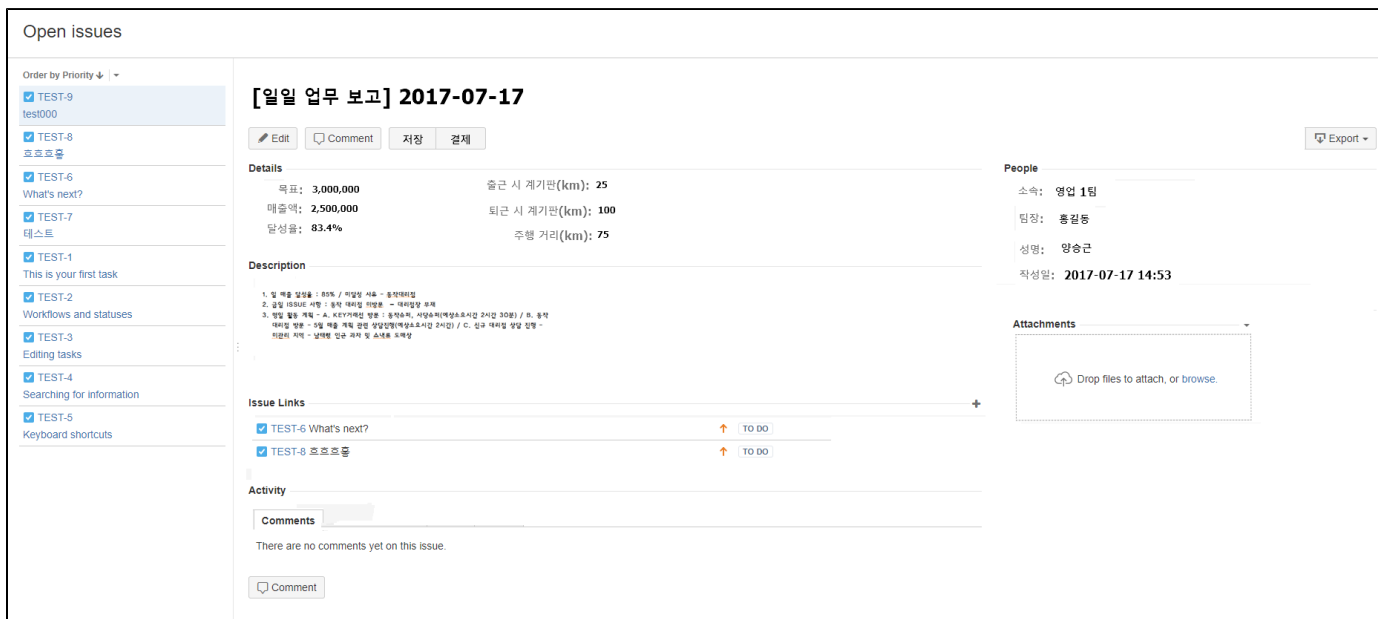
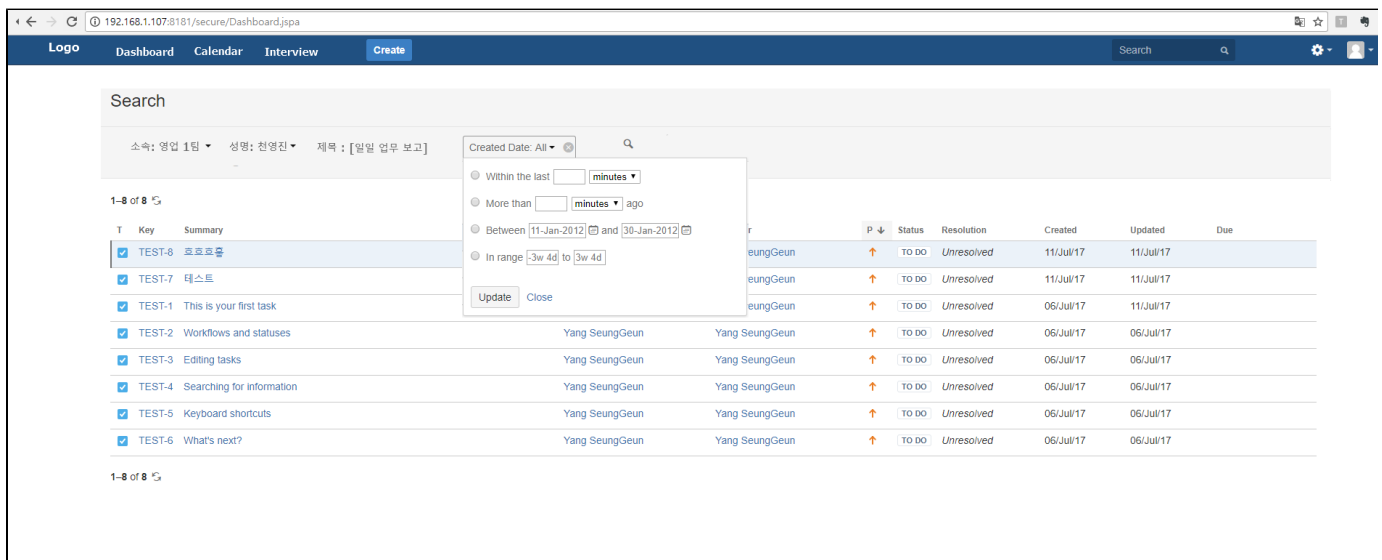
Find Users/Groups...
Current User
Unassigned
Suggested Users
admin
Lee HarkSu
Lim JinSuk
Moon SeongHoon
No KyungWook

주말가미 통계

월간 실적

상당 시간

## 일일 업무 보고



## Popup UI

## 공통 부분

- popup 윗 레이어 부분은 " Type , 소속, 팀장, 성명, 작성일 " 로 DB에서 들고 와서 자동 입력
- Type 변경시 레이아웃 변경 → 이때 기존에 입력한 값은 삭제됨
- 뒤로가기 (브라우저 백) 누를시 javascript confirm 을 이용해서 데이터가 날라가는데 뒤로 갈건지 사용자에게 물어봄.
- cancel 버튼을 누르면 javascript confirm 을 이용해서 데이터가 날라가는데 괜찮은지 물어봄. ( 데이터를 입력한 경우 )

## 주간 계획

- Calender 부분에서 week를 클릭하고 create 버튼을 누르는 경우 바로 생성 뜨는 화면

Create Issue

TYPE : \* 

주간 계획

소속: 영업 1팀    팀장: 홍길동    성명: 양승근    작성일: 2017-07-17 14:53

제목:

목표:

매출액:

달성율:

| 구분   | 월    | 화     | 수                              | 목                         | 금                 |
|------|------|-------|--------------------------------|---------------------------|-------------------|
|      | 5/5  | 5/6   | 5/7                            | 5/8                       | 5/9               |
| 매출   |      |       | 3,000,000                      | 5,000,000                 | 4,500,000         |
| 활동계획 | 어린이날 | 석가탄신일 | 본사출근<br>동작슈퍼<br>동작대리점<br>신갈대리점 | 강남대리점<br>방배대리점<br>신규거래처상당 | KEY거래선방문<br>서초대리점 |

수정

저장

결제

Cancel

## 일일 업무 보고

Create Issue

TYPE : \* 일일 업무 보고

소속: 영업 1팀

팀장: 홍길동

성명: 양승근

작성일: 2017-07-17 14:53

제목:

목표:


매출액:

달성율:

출근 시 계기판(km):

퇴근 시 계기판(km):

주행 거리(km):

첨부파일:  Drop files to attach, or [browse](#).

상담일지: TEST-6 × TEST-8 × +

업무 보고 :

Style ▾


B


I


U


A ▾

☞A ▾

 ▾

 ▾

 ▾



 ▾

+

⌵

Visual

Text

팀장 의견:

수정

저장

결제

Cancel

JIRA 의 이슈를 첨부하는 필드

선택전 검색시 Ajax로 자동 완성을 진행

Issue t

History Search (Showing 9 of 9 matching issues)

- ☒ TEST-8 - ㅎㅎㅎㅎ
- ☒ SMS-1 - test
- ☒ TEST-6 - What's next?
- ☒ TEST-5 - Keyboard shortcuts
- ☒ TEST-4 - Searching for information
- ☒ TEST-2 - Workflows and statuses
- ☒ TEST-7 - 테스트
- ☒ TEST-1 - This is your first task
- ☒ TEST-3 - Editing tasks

T (Enter issue key)

선택 후 화면

해당 이슈의 Key가 추가된 모습 이며 x를 누르면 삭제가 가능

Issue TEST-6 x TEST-8 x

생성 후 티켓 화면

첨부된 이슈를 볼수 있으며 링크가 걸려 있어 바로 갈수 있다.

Issue Links

blocks ☒ TEST-6 What's next? ↑ TO DO

☒ TEST-8 ㅎㅎㅎㅎ ↑ TO DO

연결된 이슈에서도 마찬가지로 해당 이슈로 갈수 있음

Issue Links

is blocked by ☒ TEST-9 test000 ↑ TO DO



## Create Issue

TYPE : \* 상담 일지 ▼

소속: 영업 1팀    팀장: 홍길동    성명: 양승근    작성일: 2017-07-17 14:53

제목:

고객 코드:

고객명:

대표자:

2차 거래선:

주소:

상담 내역:

Style ▼
B
I
U
A
↻A ▼
🔗 ▼
☰
☷
🗨 ▼
+
⌵

Visual

Text

↶ ↷

수정

저장

결제

Cancel

## 비트 프로젝트 07/17 (월)

### 진행 상황

07/10(월) 시작 ~ 07/14(금) = 5PD

Web 의 UI 콘틀은 완성이 됨.

07/17(월)

모바일 UI의 콘틀을 완성했고 각자 DB 스키마 설계를 진행

### 벤츠 마킹 (잡플래닛)

로그인 (UI 는 기존과 동일)

### Create 버튼

버튼을 누르면 선택이 가능하게

### 메뉴

Viewpager 사용해서 슬라이드시 전환이 되도록 설정

- 그래프
- 달력
- 리스트

### 그래프

매니저 , 지점별을 선택하여 그래프를 볼수 있음

매니저로 선택시 매니저의 이름을 선택하는 리스트에서 선택하여 해당 매니저의 통계 출력 가능 ( 팀 선택 리스트가 있어도 좋음→ 매니저가 많은 경우 )

지점별로 선택시 지점 이름을 선택하는 리스트가 있어서 리스트에서 원하는 지점을 선택하여 출력이 가능

예측 통계인지 실사 통계인지 선택하는 리스트가 있어서 데이터를 뿌려줄수 있음 ( 데이터가 없는 경우 토스트 또는 페이지에 텍스트를 이용해서 알려줘야함. )

매니저

- 월간 실적

- 주행거리 통계
- 상담 시간

#### 지점별

- 우리가 산정한 매출액 대비 실적

## 달력

갤럭시 기본 달력을 벤치 마킹하며 영업 사원의 일정을 좀더 편리하게 관리하기 위하여 사용하는 부분

- 달력의 해당일을 클릭시 일일 업무 보고서 상세 Activity로 넘어간다.
- 좌우로( 이전달, 다음달 ) 넘길수 있는 달력의 버튼이 있고 평소에는 잘보이지 않으나 (흐릿하게 보임) 사용자가 클릭시 진하게 표시된다.
- 상단에 오늘 이라는 버튼이 있어서 어느 페이지에서든 해당 월로 돌아올수 있다. (주일 경우 해당 주로)
- 보기 방식이라는 버튼을 클릭시 리스트가 나오며 " 주 , 월 " 등을 선택하여 달력의 형식을 변경해서 볼수 있다.

## 리스트

주간 계획 , 일일 업무 보고 , 상담 일지 등의 정보를 볼수 있는 Activity 이다.

주간 계획 , 일일 업무 보고 , 상담 일지 등을 선택할수 있는 리스트가 있다.

해당 매니저의 팀을 선택 할수 있는 리스트가 있다.

해당 매니저의 이름을 선택 할수 있는 리스트가 있다.

리스트의 정렬 순서는 최신순 이며 총 리스트는 15개 씩 뿌려주며 (데이터를 가지고 있는게)

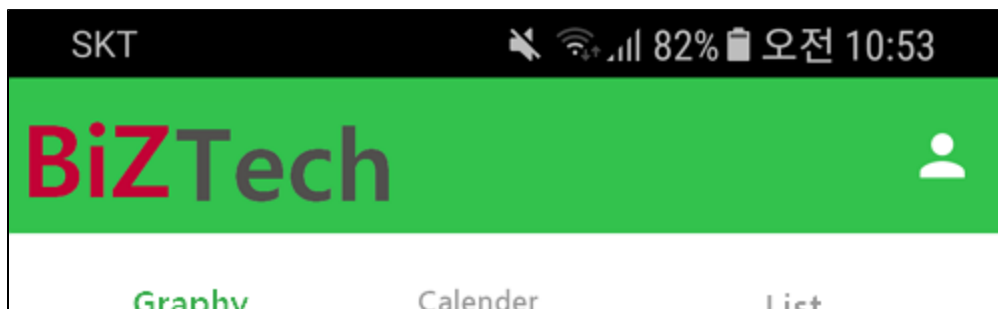
그 이상 화면을 내리는 경우 Ajax 방식으로 Rest 서버로 부터 데이터를 받아서 아래에 계속 추가해준다. ( json 방식 )

주간 계획 같은 경우 계획일 이후로는 수정이 불가능 하므로 수정하는 해당일이 계획일 보다 이후인경우 Fixed라고 리스트에 표시해준다.

일일 업무 보고 리스트는 동일하게 해당 보고서의 상태를 리스트에 뿌려준다. ( 팀장이 승인 완료시 Fixed 라고)

상담일지는 일일 보고서와 연결이 된 녀석인 경우 연결이 된 녀석이라는 것을 알려줘야 한다. ( 연결이 된 녀석만 link 라고 표시 하거나 따른 방식으로 표시함 → 자물쇠 아이콘 사용도 좋아 보임.)

## 그래프 화면



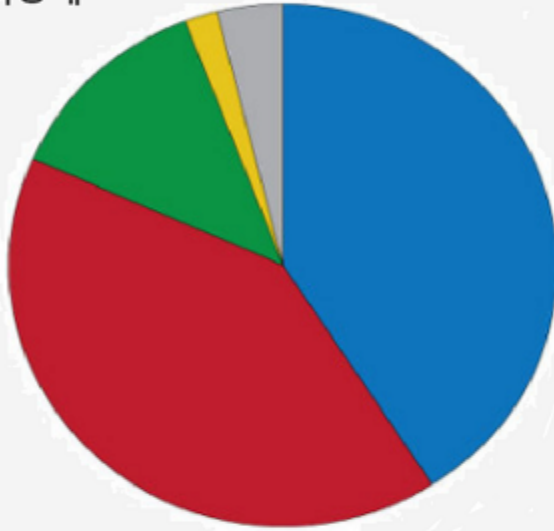


매니저 ▼

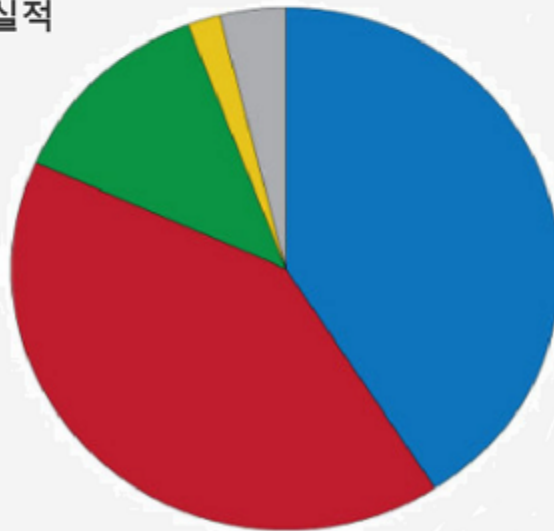
홍길동 ▼

예측 통계 ▼

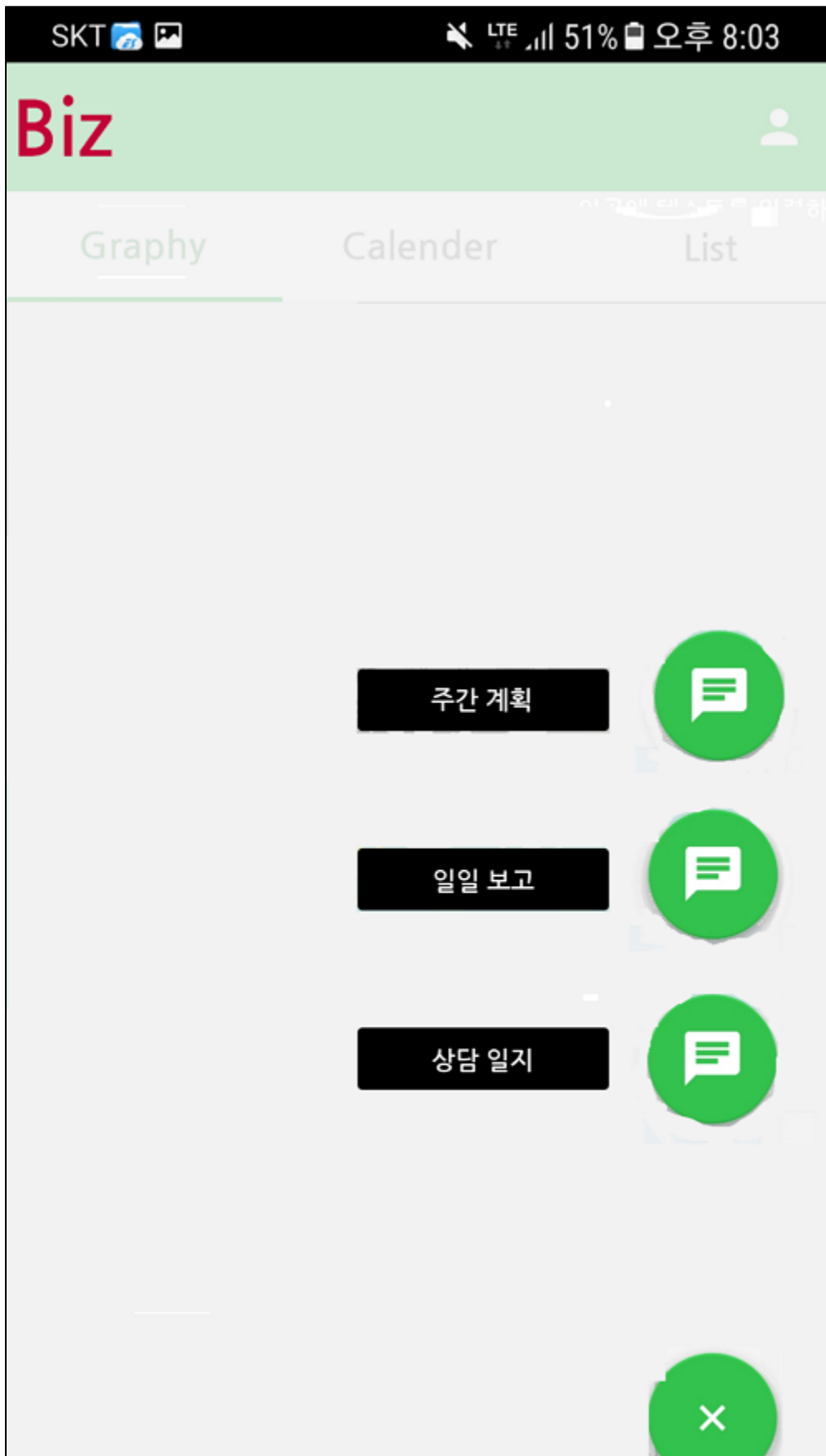
주행 거리통계



월간 실적



생성 버튼 클릭시



전체 화면으로 보려면 여기를 누르세요



문서 작성 부분

SKT

82% 오전 10:54

## ← 주간 계획 작성

소속: 영업 1팀

이름: 양승근

팀장: 홍길동

작성일: 2017-07-19 12:59

제목

제목을 입력해 주세요.

목표

목표를 입력해 주세요.

매출액

매출액을 입력해 주세요.

달성율

달성율 (자동입력)

5/5 (월)

여리가 난 이니다



Graphy

Calender

List




7월  
2017년

오늘 보기 방식


| 일             | 월  | 화  | 수             | 목  | 금  | 토  |
|---------------|----|----|---------------|----|----|----|
| 25<br>윤달 5/2  | 26 | 27 | 28            | 29 | 30 | 1  |
| 2<br>윤달 5/9   | 3  | 4  | 5             | 6  | 7  | 8  |
| 9<br>윤달 5/16  | 10 | 11 | 12            | 13 | 14 | 15 |
| 16<br>윤달 5/23 | 17 | 18 | 19<br>윤달 5/26 | 20 | 21 | 22 |
| 23<br>6/1     | 24 | 25 | 26            | 27 | 28 | 29 |
| 30<br>6/8     | 31 | 1  | 2             | 3  | 4  | 5  |



SKT

 82% 오전 10:53

BiZTech



Graphy

Calender

List

주간 계획 ▼

영업 1팀 ▼

홍길동 ▼

[주간 계획] 2017-07-17

작성일 : 2017-07-13 09:15

[주간 계획] 2017-07-10


작성일 : 2017-07-08 12:08 Fixed

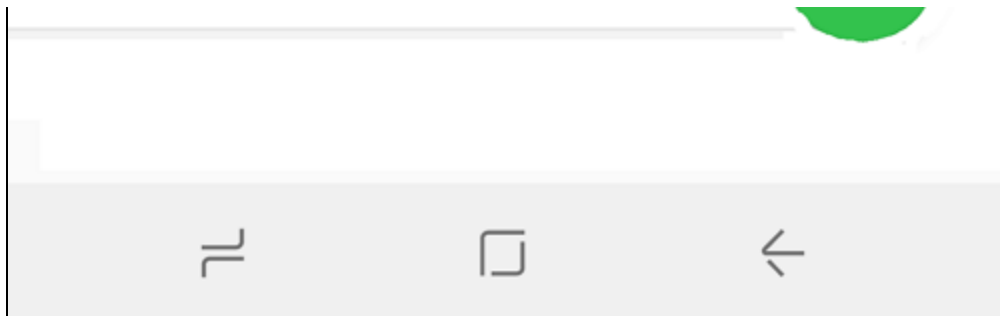
[주간 계획] 2017-07-03

작성일 : 2017-06-27 14:18 Fixed

[주간 계획] 2017-06-26

작성일 : 2017-06-22 10:25 Fixed





## 비트 프로젝트 07/18 (화)

### 진행 상황

07/10(월) 시작 ~ 07/17(월) = 6PD

Web 의 UI 컨트롤은 완성

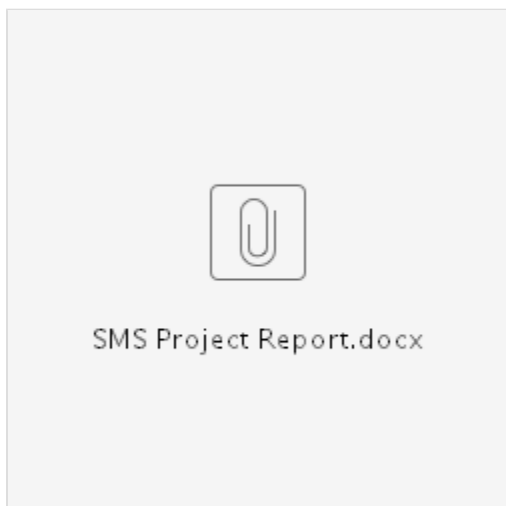
Android 의 UI 컨트롤은 완성

### 07/18(화)

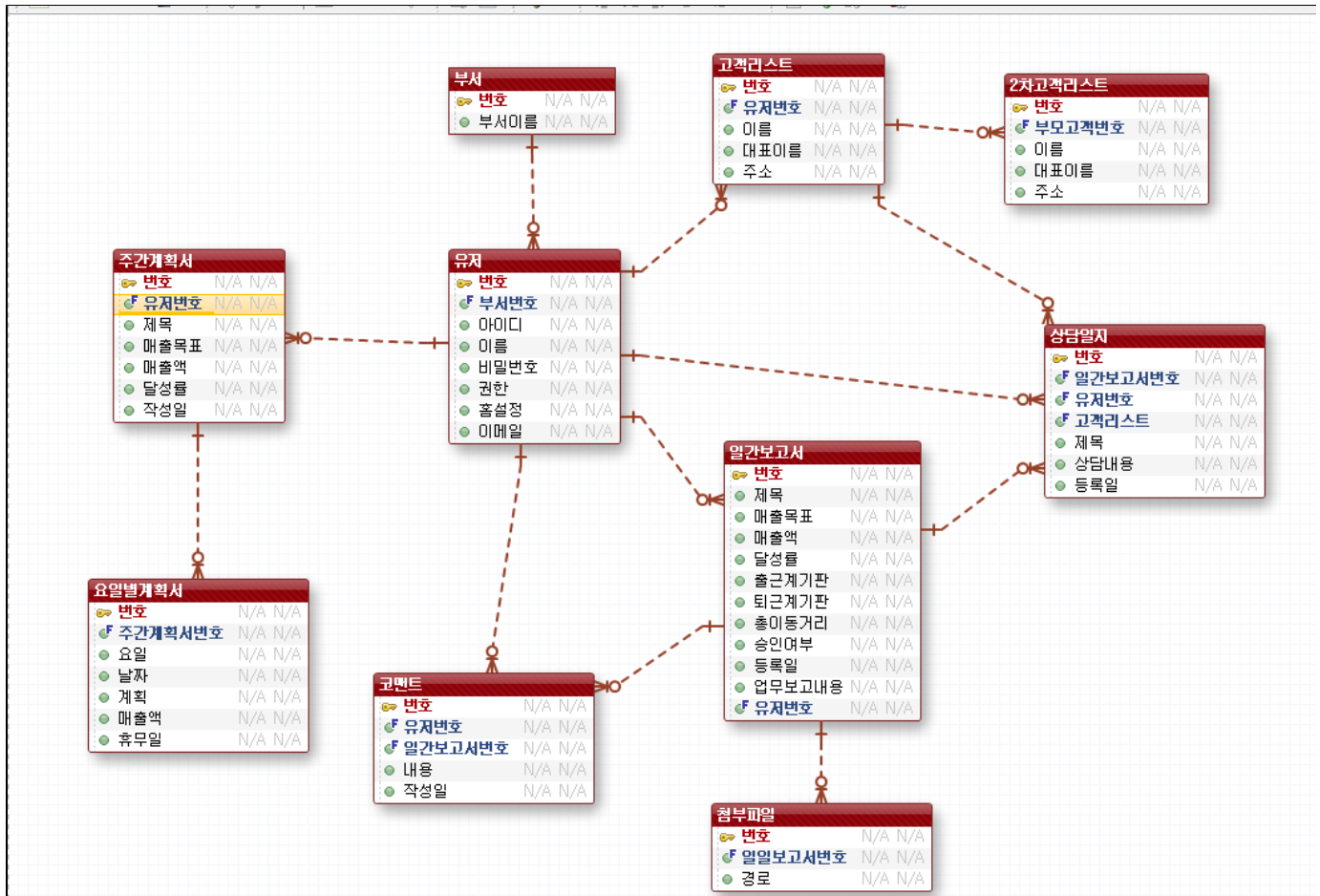
17일 각자 설계한 DB를 바탕으로 통합작업 진행 (DB 스키마 완전 완료)

프로젝트 개요서를 같이 만들며 일정을 명확히 잡음 (프로젝트 개요서 감사님에게 제출)

### 프로젝트 개요서



DB모형 (EXERD)



## 비트 프로젝트 07/26 (수)

### 진행 상황

- 07/24 ~ 07/26 → 상세 일정 설정 진행 (티켓팅)

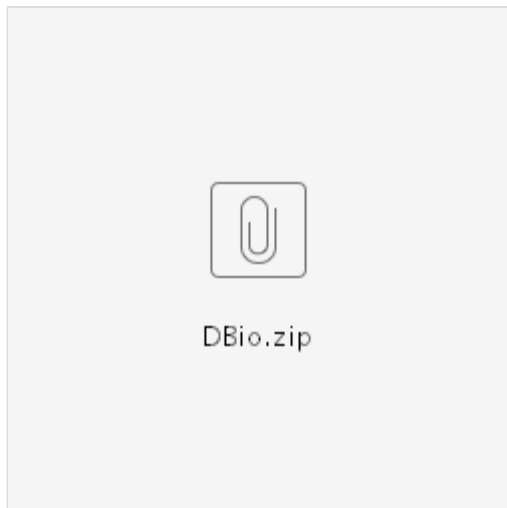
## Android Basic Network Technology Study

### Android

- Async Task 를 활용한 Background 작업 방법
- Custom list view 만드는 방법 → ( list , view , adapter )
- HttpClient 객체를 사용하여 데이터를 서버에 전송하는 방법
- 전송 후 결과를 json으로 받아서 처리하는 방법 (JSONArray → JsonObject, Gson)
- 객체 직렬화를 통해 데이터를 전달하는 방법 (Serializable 인터페이스 → CData)
- 이미지를 카메라, 앨범에서 들고와서 Crop APP을 호출하여 처리하는 방법
- HttpURLConnection 과 DataOutputStream 를 이용해서 바이너리 파일을 HTTP 통신으로 전달하는 방법

### 코드 파일





## Spring

- Mybatis 를 활용하여 DB 처리 방법
- Jackson 과 Message Converter 를 이용하여 Rest 처리 방법
- MultipartResolver 를 이용해서 바이너리 파일을 받아서 처리하는 방법
- LogBack을 활용하여 Network 통신시 error를 디버깅 하는 방법

코드 경로 → <https://github.com/tmdrms0817/async>

## DB정보

```
CREATE TABLE test (
idx INT(10) unsigned NOT NULL AUTO_INCREMENT,
name VARCHAR(100) NOT NULL,
color VARCHAR(100) NOT NULL,
tips VARCHAR(100) NOT NULL,
contents mediumtext NOT NULL,
images VARCHAR(100) NOT NULL,
PRIMARY KEY (idx)
);
```

---

## 회의 내용

- Rest 관련 데이터 전달 부분 회의 → (인증 방식 및 로그 처리 방법에 관하여 고민이 필요)
- 스케줄 관련 회의 ( 전체적인 구성을 알수가 없기에 15PD로 잡고 기본기능을 먼저 구현하기로 결정 )

## 과제

15PD를 기준으로 기본 기능 스케줄 (Task)를 금주 안으로 정하기로 결정

## Android prototype code



DBio.zip

## 비트 프로젝트 08/23 (수)

### 주간계획

1. 현재주(생성일) 이전의 주간 계획은 작성이 불가능. (UI상에서 클릭을 못하도록 설정 ~!)
2. 주간 계획이 작성된 주는 작성불가 (계획서 업데이트로 진행)
3. 주간계획서 상세 화면에서 일일계획서 리스트가 있고 보고서가 작성이 된 녀석은 click 시 일일 보고서 화면으로 전환. (없으면 ui상 제어를 하게 설정)
4. web측 수정은 popup형식이며 전체 업데이트 (일일 계획이랑 주간계획을 전부 update 하는 형식)
5. 주간 계획의 경우 미리 작성이 가능하므로 미래에 작성한 계획은 삭제가 가능하게 설정.

### 일일 계획

1. 주간계획 작성시 같이 작성이 되도록 설정.
2. 현재일 이전의 계획은 ui상에서 누르지 못하게 설정하여 수정 및 삭제 등의 모든 것이 불가능.

### 일일업무보고

1. 일일업무보고서에선 해당 일일보고서의 부모 task(주간 계획) 으로 이동이 가능하게 기능을 제공하여야 한다. (버튼 또는 리스트)
2. 여러개 작성이 가능하나 결제를 올리는 것은 1개만 가능하다. (상태가 → 미승인, 승인요청, 승인, 반려)
3. 결제가 진행된 후에는 수정이 불가능하다. ( 승인요청, 승인, 반려 상태이면 )
4. 결제를 넣지 않은 문서는 삭제가 가능하다.

### 상담일지

1. 상담일지 하나당 하나의 일일 업무 보고만 연결이 가능하게 설정.
2. link된 상담일지는 삭제가 불가능하게 설정 (일일 보고서의 결제가 진행된 경우 )

### 로그인

1. 5회이상 실패시 admin에게 문의하라고 띄워주고 lock을 건다. (비밀번호 관련)

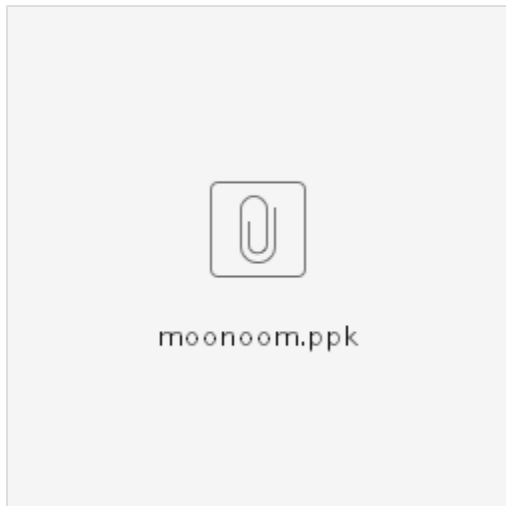
### 첨부파일

1. MD5로 암호화된 문자를 HTTP로 같이 넘겨 무결성 검사를 진행 후 첨부파일을 서버에 올린다.

## Project

## ANDROID prototype

## aws key



## Coding Standards Guide

### 개요

개발 과정에 일련의 규칙을 적용하여 협업 진행시 짧은 시간 안에 산출물을 생산 하는 것에 목적을 둠.

### 코딩 규약이 필요한 이유

소프트웨어를 개발하는 일련의 모든 과정에 들어가는 비용 중 80%가 유지보수에 쓰여지고 소프트웨어의 유지보수를 그 소프트웨어를 직접 개발한 개발자가 담당하는 경우는 거의 보기 힘들다.

코딩 규칙을 지키면 다른 개발자가 그 소스 코드를 처음 보았을 때, 더 빠른 시간 안에 완벽하게 이해할 수 있도록 도와주기 때문에, 소프트웨어의 가독성이 높아진다.

개발자가 자신의 소스 코드를 제품으로 팔려고 한다면, 자신이 작성한 다른 소스 코드들과 잘 어울리도록 패키지(package)를 적절하게 구성할 필요가 있다.

---

## 버전 통일

### Android

- 디바이스 - 삼성 갤럭시 S5
- 최소 지원 SDK 버전 - 5.0(롤리팝)

Node - NVM 버전 6.11.1

Spring - 4.2.1.RELEASE

JAVA - 1.8.0\_141

Apache2 - Apache/2.4.18 (Ubuntu)

Maven - Apache Maven 3.3.9

OS - "linux", version: "4.4.0-62-generic", arch: "amd64", family: "unix"

Tomcat - tomcat8

## Tool

Git - 2.13.0

Confluence - 6.2.3

JIRA - 7.4.0

Jenkins - 1.0

## 파일 이름

### 파일 확장자

| 파일 형태    | 확장자    |
|----------|--------|
| 자바 소스    | .java  |
| 자바 바이트코드 | .class |

### 공통으로 사용되는 파일 이름

| 파일 이름       | 사용                                             |
|-------------|------------------------------------------------|
| GNUmakefile | make파일 이름으로 사용. 소프트웨어를 빌드할 때는 gnumake 명령어를 사용. |
| README      | 특정 디렉토리의 내용을 요약하는 파일 이름으로 사용.                  |

## 파일 구조

파일은 빈 줄이나 다른 구역임을 나타내주는 주석으로 나누어지는 여러 구역(section)들로 구성되어 있다.

2000 라인을 넘는 파일은 이해하기가 쉽지 않기 때문에 될 수 있으면 피해야 한다.

### 자바 소스 파일

|                                                                                                  |
|--------------------------------------------------------------------------------------------------|
| 각각의 자바 소스 파일은 하나의 public 클래스 또는 인터페이스를 가진다.                                                      |
| Private 클래스들과 인터페이스들이 public클래스와 연결되어 있을 때,<br>public 클래스와 같은 파일에 private 클래스들과 인터페이스들을 넣을 수 있다. |
| Public 클래스는 파일에서 첫번째 클래스 또는 인터페이스이어야 한다.                                                         |

자바 소스 파일은 다음과 같은 순서를 가진다.

- 시작 주석
- Package문과 Import 문
- 클래스와 인터페이스 선언

## 시작 주석

```
ex)
/*
 * 클래스 이름
 *
 * 버전 정보
 *
 * 날짜
 *
 * 저작권 주의
 */
```

## Package 문과 Import 문

대부분의 자바 소스 파일에서 주석이 아닌 첫번째 줄은 package 문이며, 그 후에, import문이 뒤따라 나온다.

```
ex)
package java.awt;

import java.awt.peer.CanvasPeer;
```

### 주의

Package는 한 번만 선언되어야 한다.

package 이름의 첫번째 부분은 모두 소문자 ASCII 문자를 사용해야 한다.

첫번째 레벨의 도메인 이름들(com, edu, gov, mil, net, org)중에 하나이거나,

1981년 ISO Standard 3166에서 정의된 영어 두 문자로 표현되는 나라 구별 코드 중에 하나이어야 한다.

## Class와 Interface 선언

|   | 클래스/인터페이스 선언의 구성요소                 | 설명                                                                |
|---|------------------------------------|-------------------------------------------------------------------|
| 1 | 클래스/인터페이스 문서화 주석 (/**...*/)        | 5) 주석 → 문서화(Documentation) 주석을 참고.                                |
| 2 | 클래스/인터페이스 문                        |                                                                   |
| 3 | 필요할 경우, 클래스/인터페이스 구현 주석 (/*...*/), | 이 주석은 클래스/인터페이스 문서화 주석(적합하지 않은 하나의 클래스/인터페이스에만 해당하는 정보들을 포함해야 한다. |

|   |                |                                                                                                                                                                         |
|---|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 4 | 클래스(static) 변수 | <p>첫번째로는 public 클래스 변수들이 나오고, 그 다음에 protected 클래스 변수들,</p> <p>그 다음에 package(접근 제한자(access modifier)가 없는 경우) 클래스 변수들,</p> <p>그 다음에 private 클래스 변수들이 나온다.</p>             |
| 5 | 일반 변수          | 작성하는 순서는 클래스 변수와 동일하다.                                                                                                                                                  |
| 6 | 생성자            |                                                                                                                                                                         |
| 7 | 메서드            | <p>메서드들은 범위나 접근성을 기준으로 나누기 보다는 기능성에 의해서 구성되어야 한다.</p> <p>예를 들어, private 클래스 메서드가 개의 public 메서드들 사이에 존재할 수도 있다.</p> <p>이렇게 하는 목적은 코드가 더 쉽게 읽히고, 더 쉽게 이해되도록 돕기 위해서이다.</p> |

## 들여쓰기

4개의 빈 칸(space)을 들여쓰기 단위로 사용한다.

### 한 줄의 길이

한 줄에 80자 이상 쓰는 것은 대부분의 터미널(terminal)과 툴에서 다룰 수 없기 때문에 피해야 한다.

주의 : 문서화 주석을 가지고 문서를 만들 때, 일반적으로 한 줄에 70자 이상을 가지지 않는다.

### 줄 나누기

하나의 식이 한 줄에 들어가지 않을 때에는, 다음과 같은 일반적인 원칙들을 따라서 두 줄로 나눈다.

- 콤마 후에 두 줄로 나눈다.
- 연산자(operator) 앞에서 두 줄로 나눈다.
- 레벨이 낮은 원칙 보다는 레벨이 높은 원칙에 따라 두 줄로 나눈다.
- 앞줄과 같은 레벨의 식(expression)이 시작되는 새로운 줄은 앞줄과 들여쓰기를 일치시킨다.
- 만약 위의 원칙들이 코드를 더 복잡하게 하거나 오른쪽 끝을 넘어간다면, 대신에 8개의 빈 칸을 사용해 들여쓴다.

ex) 메서드 호출을 두 줄로 나누어 쓰는 예제

```
someMethod(longExpression1, longExpression2, longExpression3,
 longExpression4, longExpression5);
```

```
var = someMethod1(longExpression1,
 someMethod2(longExpression2,
 longExpression3));
```

ex) 수학 표현식을 두 줄로 나누어 작성하는 두 개의 예제

```
// 괄호로 싸여진 표현식 밖에서 줄 바꿈이 일어나고 더 높은 레벨이기 때문에 될 수 있으면 이 방법을 사용
longName1 = longName2 * (longName3 + longName4 - longName5)
 + 4 * longname6;
```

// 될 수 있으면 피하는방법

```
longName1 = longName2 * (longName3 + longName4
 - longName5) + 4 * longname6;
```

ex) 메서드 선언을 들여쓰는 예제

// 일반적인 들여쓰기

```
someMethod(int anArg, Object anotherArg, String yetAnotherArg,
 Object andStillAnother) {
 ...
}
```

// 너무 멀리 들여쓰는 것을 피하기 위해 8개의 빈 칸으로 들여쓰기

```
private static synchronized horkingLongMethodName(int anArg,
 Object anotherArg, String yetAnotherArg,
 Object andStillAnother) {
 ...
}
```

일반적으로 메서드 본문이 시작할 때 4개의 빈 칸을 사용

메서드 본문과 구분하기 위해서 줄을 나누는 경우의 들여쓰기는 일반적으로 8개의 빈 칸 원칙을 사용

// 아래와 같은 들여쓰기는 사용하지 않는 것이 좋다.

```
if ((condition1 && condition2)
 || (condition3 && condition4)
 ||!(condition5 && condition6)) { // 좋지 않은 들여쓰기
 doSomethingAboutIt(); // 메서드 본문 시작이 명확하지가 않다.
}
```

// 대신에 아래와 같은 들여쓰기를 사용한다.

```
if ((condition1 && condition2)
 || (condition3 && condition4)
 ||!(condition5 && condition6)) {
 doSomethingAboutIt();
}
```

// 또는 아래와 같은 들여쓰기를 사용한다.

```
if ((condition1 && condition2) || (condition3 && condition4)
 ||!(condition5 && condition6)) {
 doSomethingAboutIt();
}
```

ex) 삼항식(ternary expression)에서 사용 가능한 세 가지 방법

```
alpha = (aLongBooleanExpression) ? beta : gamma;
```

```
alpha = (aLongBooleanExpression) ? beta
 : gamma;
```

```
alpha = (aLongBooleanExpression)
 ? beta
 : gamma;
```

## 주석

자바 프로그램은 두 가지 종류의 주석을 가진다.

주어 : 때로는 코드에 대한 주석이 많이 필요하다는 것은 코드의 품질이 좋지 않다는 것을 의미

## 구현 주석 형식

### 블록(block) 주석

- 블록 주석은 파일, 메서드, 자료 구조, 알고리즘에 대한 설명을 제공할 때 사용
- 블록 주석은 각각의 파일이 시작될 때와 메서드 전에 사용
- 메서드 안에서와 같이 다른 장소에서 사용될 수도 있다.
- 메서드 안에 존재하는 블록 주석은 주석이 설명하는 코드와 같은 단위로 들여쓰기를 해야 한다.

ex) 블록 주석은 다른 코드들과 구분하기 위해서 처음 한 줄은 비우고 사용

```
/*
 * 블록 주석 작성
 */
```

ex) 블록 주석의 들여쓰기를 다시 고치지 못하도록 하기 위한 특별한 블록 주석은 /\*- 으로 시작할 수 있다.

```
/*-
 * 여기에 들여쓰기를 무시하는 특별한 블록 주석을 작성한다.
 *
 * one
 * two
 * three
 */
```

### 한 줄(single-line) 주석

짧은 주석은 뒤따라 오는 코드와 같은 동일한 들여쓰기를 하는 한 줄로 작성할 수 있다.

만약 주석이 한 줄에 다 들어가지 않는다면, 블록 주석 형식을 따라야 한다.

한 줄 주석은 빈 줄로 시작되어야 한다.

ex) 자바 코드에서 한 줄 주석의 예제

```
if (condition) {
 /* 이 조건을 만족하면 실행된다. */
 ...
}
```

### 꼬리(trailing) 주석

아주 짧은 주석이 필요한 경우 주석이 설명하는 코드와 같은 줄에 작성

실제 코드와는 구분될 수 있도록 충분히 멀리 떨어뜨려야 한다.

ex) 자바 코드에서 꼬리 주석을 사용하는 예제

```
if (a == 2) {
 return TRUE; /* 특별한 경우 */
} else {
 return isPrime(a); /* a 가 홀수인 경우 */
}
```



줄 끝(end-of-line) 주석

주석 기호 // 는 한 줄 모두를 주석 처리하거나 한 줄의 일부분을 주석 처리해야 할 때 사용할 수 있다.

긴 내용을 주석에 포함하기 위해서 연속되는 여러 줄에 이 주석을 사용하는 것은 금지

어떤 코드의 일부분을 주석 처리하기 위해서 여러 줄에 연속해서 사용하는 것은 허락된다.

```
ex) 줄 끝 주석을 사용하는 세가지 스타일
if (foo > 1) {
 // double-flip을 실행한다.
 ...
}
else {
 return false; // 이유를 여기에 설명한다.
}
//if (bar > 1) {
//
// // double-flip을 실행한다.
// ...
// }
//else {
// return false;
//}
```

문서화(Documentation) 주석

문서화 주석은 자바 클래스, 인터페이스, 생성자, 메서드 그리고 필드들을 설명

각각의 문서화 주석은 주석 경계 기호인 `/**...*/` 안으로 들어간다.

각각의 문서화 주석은 클래스, 인터페이스 그리고 멤버 당 하나씩 가진다.

문서화 주석은 선언 바로 전에 나와야 한다.

```
ex)
/**
 * Example 클래스는 ...
 */
public class Example { ...
```

처음 나오는 클래스와 인터페이스들은 들어 쓰지 않는 반면에 그들의 멤버들은 들어쓰기를 한다.

클래스에 대한 문서화 주석(`/**`)의 첫 번째 줄은 들어 쓰지 않는다.

; 그 다음에 나오는 문서화 주석은 별표를 수직으로 맞추기 위해 각각 1개의 빈 칸을 들어쓰기 한다.

생성자를 포함한 멤버들은 문서화 주석 첫 줄에서는 4개의 빈 칸을 들어쓰기 하고, 그 이후에는 5개의 빈 칸으로 들어쓰기를 한다.

만약 클래스, 인터페이스, 변수 또는 메서드에 대한 어떤 정보를 제공하고 싶지만 문서화 주석에 어울리지 않는다고 생각된다면, 클래스 선언 바로 후에 블록 주석 또는 한 줄 주석을 사용한다.

자바는 문서화 주석을 주석 이후에 처음 나오는 선언문과 연결시키기 때문에 문서화 주석은 메서드 또는 생성자 구현 안에 위치해서는 안 된다.

## 선언

한 줄당 선언문의 수

한 줄에 하나의 선언문을 쓰는 것이 주석문 쓰는 것을 쉽게 해주기 때문에 한 줄에 하나의 선언문을 쓰는 것이 좋다.

ex) 좋은 방법  
int level; // 들여쓰기 단위  
int size; // 테이블 크기

int level, size //비추

같은 줄에 서로 다른 타입을 선언하면 안 된다.

ex) int foo, fooarray[]; //절대 이렇게 사용하지 말자!

ex) 탭을 사용하는 방법  
int        level;                // 들여쓰기 단위  
int        size;                // 테이블 크기  
Object     currentEntry;        // 테이블에서 현재 선택된 데이터

## 초기화

지역 변수의 경우, 지역 변수를 선언할 때 초기화 하는 것이 좋다.

변수의 초기화 값이 다른 계산에 의해서 결정되는 경우라면, 변수를 선언할 때 초기화 하지 않아도 괜찮다.

## 배치

선언은 블록의 시작에 위치해야 한다.

ex) 보통 블록은 중괄호인 "{" 과 "}"로 둘러싸인 코드를 이야기한다.

변수를 처음 사용할 때까지 변수의 선언을 미루지 말아야 함.

: 부주의한 프로그래머들을 혼돈에 빠뜨릴 수 있고, 영역내에서 코드를 이동해야 하는 경우에 문제를 야기시킬 수 있다.

ex)  
void myMethod() {  
    int int1 = 0;        // 메서드 블록의 시작  
    if (condition) {  
        int int2 = 0;    // "if" 블록의 시작  
        ...  
    }  
}

이러한 원칙에도 예외는 존재

ex) for 문에서 선언하는 반복문을 위한 인덱스  
for (int i = 0; i < maxLoops; i++) { ... }

상위 영역에서 선언된 것을 숨기기 위해서 블록의 처음 부분에서 다시 선언하는 것은 피해야 한다.

ex) 블록 안의 블록에서 동일한 변수 이름을 사용해서 선언하지 말아야 함.

int count;  
...  
myMethod() {  
    if (condition) {  
        int count = 0;    // 이렇게 사용하지 말 것!  
        ...  
    }  
    ...  
}

## 클래스와 인터페이스의 선언

자바 클래스와 인터페이스를 선언할 때의 원칙

메서드 이름과 그 메서드의 파라미터 리스트의 시작인 괄호 "(" 사이에는 빈 공간이 없어야 한다.

여는 중괄호 "{" 는 클래스/인터페이스/메서드 선언과 동일한 줄의 끝에 사용

닫는 중괄호 "}" 는 "}" 가 여는 중괄호 "{" 후에 바로 나와야 하는 null 문인 경우를 제외하고는, 여는 문장과 동일한 들여쓰기를 하는 새로 줄에서 사용하자.

```
ex) class Sample extends Object {
 int ivar1;
 int ivar2;

 Sample(int i, int j) {
 ivar1 = i;
 ivar2 = j;
 }

 int emptyMethod() {}

 ...
}
```

메서드들을 구분하기 위해서 각 메서드들 사이에는 한 줄을 비우도록 하자.

## 문 (Statements)

### 간단한 문

각각의 줄에는 최대한 하나의 문(statement)만 사용하도록 한다.

```
ex)
argv++; // 올바른 사용법
argc--; // 올바른 사용법
argv++; argc--; // 이렇게 작성하는 것은 피해라!
```

### 복합문

복합문은 중괄호 "{ statements }"로 둘러싸여진 문들의 리스트를 포함하는 문이다.

- 둘러싸여진 문들은 복합문보다 한 단계 더 들여쓰기를 한다.
- 여는 중괄호 "{"는 복합문을 시작하는 줄의 마지막에 위치해야 한다.
- 닫는 중괄호 "}"는 새로운 줄에 써야 하고, 복합문의 시작과 같은 들여쓰기를 한다.
- 중괄호들이 if-else 문이나 for 문 같은 제어 구조의 일부분으로 사용되어질 때에는 이러한 중괄호들이 모든 문들을 둘러싸는데 사용되어야 한다.
- 이렇게 사용하는 것이 중괄호를 닫는 것을 잊어버리는 것 때문에 발생하는 버그를 만들지 않고, 문을 추가하는 것에 큰 도움을 준다.

### return 문

ex)값을 반환하는 return 문은 특별한 방법으로 더 확실한 return 값을 표현하는 경우를 제외하고는 괄호를 사용하지 않는 것이 좋다.

```
return;
```

```
return myDisk.size();
```

```
return (size ? size : defaultSize);
```

if, if-else, if else-if else 문

```
ex)
if (condition) {
 statements;
}

if (condition) {
 statements;
} else {
 statements;
}

if (condition) {
 statements;
} else if (condition) {
 statements;
} else {
 statements;
}
```

if 문은 항상 중괄호를 사용한다. 다음과 같은 에러가 발생할 수 있는 상황은 피해야 한다.

```
ex)
if (condition) // 이렇게 중괄호 {}를 생략해서 사용하지 말자!
 statements;
```

for 문

```
ex)
for (initialization; condition; update) {
 statements;
}
```

빈 for 문(모든 작업이 initialization, condition, update 에서 완료되는)은 다음과 같은 형태를 가져야 한다.

```
ex) for (initialization; condition; update);
```

for 문의 initialization또는 update 부분에서 콤마(,) 연산자를 사용할 때에는, 세 개 이상의 변수들을 사용하는 복잡성은 피해야 한다.

만약 필요하다면, for 문 이전에 문을 분리시켜 사용(initialization절의 경우)하거나 for 문의 마지막에 문을 분리시켜 사용(update절의 경우)한다.

while 문

```
ex)
while (condition) {
 statements;
}

while (condition);
```

do-while 문

```
ex)
do {
 statements;
} while (condition);
```

switch 문

```

ex)
switch (condition) {
case ABC:
 statements;
 /* 다음줄로 계속 진행한다. */
case DEF:
 statements;
 break;
case XYZ:
 statements;
 break;
default:
 statements;
 break;
}

```

모든 case를 수행해야 하는 경우에는 break 문을 사용하지 않으면 된다.

모든 switch 문은 default case를 포함해야 한다.

어떤 경우에 default case에서 break는 중복적이지만, 이후에 또 다른 case가 추가되어질 경우 에러를 방지할 수 있다.

## try-catch 문

```

ex)
try {
 statements;
} catch (ExceptionClass e) {
 statements;
}

```

try-catch 문은 try 블록이 성공적으로 완료되든지,

아니면 중간에 에러가 발생하든지에 상관없이 실행되어야 하는 부분을 추가하기 위해서 finally 부분을 사용할 수 있다.

```

ex)
try {
 statements;
} catch (ExceptionClass e) {
 statements;
} finally {
 statements;
}

```

## 빈 공간 (White Space)

### 한 줄 띄우기 (Blank Lines)

한 줄을 띄우고 코드를 작성하면, 논리적으로 관계가 있는 코드들을 쉽게 구분할 수 있기 때문에 코드의 가독성(readability)을 향상시킨다

다음과 같은 경우에는 두 줄을 띄어서 코드를 작성한다.

- 메서드들 사이에서
- 소스 파일의 섹션들 사이에서
- 클래스와 인터페이스의 정의 사이에서

다음과 같은 경우에는 한 줄을 띄어서 코드를 작성한다.

- 메서드 안에서의 지역 변수와 그 메서드의 첫 번째 문장 사이에서
- 블록(Block) 주석 또는 한 줄(Single-Line) 주석 이전에
- 가독성을 향상시키기 위한 메서드 내부의 논리적인 섹션들 사이에

## 공백 (Blank Spaces)

괄호와 함께 나타나는 키워드는 공백으로 나누어야 한다.

```
ex)
 while (true) {
 ...
 }
```

메서드 이름과 메서드의 여는 괄호 사이에 공백이 사용되어서는 안 된다.  
(메서드 호출과 키워드를 구별하는데 도움을 준다)

공백은 인자(argument) 리스트에서 콤마 이후에 나타나야 한다.

.을 제외한 모든 이항(binary) 연산자는 연산수들과는 공백으로 분리되어야 한다.

```
ex)
 a += c + d;
 a = (a + b) / (c * d);
 while (d++ = s++) {
 n++;
 }
 printSize("size is " + foo + "Wn");
```

for 문에서 사용하는 세 개의 식(expression)들은 공백으로 구분해서 나누어야 한다.

```
ex) for (expr1; expr2; expr3)
```

변수의 타입을 변환하는 캐스트(cast)의 경우에는 공백으로 구분해야 한다.

```
ex)
 myMethod((byte) aNum, (Object) x);
 myMethod((int) (cp + 5), ((int) (i + 3))
 + 1);
```

## 명명(Naming) 규칙

이름을 정하는 규칙은 프로그램을 더 읽기 쉽게 만들어 줌으로써 더 이해하기 쉽게 만들어 준다.

식별자(identifier)를 통해서 기능에 대한 정보도 얻을 수 있으며, 코드를 이해하는데 큰 도움이 된다.

| 식별자<br>타입 | 명명 규칙                                                                                                                                                                                                                                                                                                       | 예제                                                               |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------|
| Packages  | 패키지 이름의 최상위 레벨은 항상 ASCII 문자에 포함되어 있는 소문자<br>가장 높은 레벨의 도메인 이름 중 하나<br>현재는 com, edu, gov, mil, net, org, 또는<br>1981년 ISO Standard 316에 명시된 영어 두<br>문자로 표현되는 나라 구별 코드가 사용<br>패키지 이름의 나머지 부분은 조직 내부의<br>명명 규칙을 따르면 된다.<br>이러한 규칙을 따라 만들어진<br>이름은 디렉토리 이름으로도 사용된다.<br>ex) 부서명, 팀명, 프로젝트명, 컴퓨터 이름,<br>로그인 이름 등이다. | com.sun.eng<br>com.apple.quicktime.v2<br>edu.cmu.cs.bovik.cheese |

|            |                                                                                                                                                                                                                                                                                                                                                             |                                                                                                           |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|
| Classes    | 클래스 이름은 명사<br>복합 단어일 경우 각 단어의 첫 글자는<br>대문자(CamelCase 방식)<br>클래스 이름은 간단하고 명시적이 되도록<br>작성<br>완전한 단어를 사용하고 두 문자어와 약어는<br>피하도록 하자.<br>(만약, 약어가 URL이나 HTML과 같이 더<br>많이, 더 넓게 사용되고 있다면 약어를<br>사용하는 것도 괜찮다)                                                                                                                                                        | class Raster;<br>class ImageSprite;                                                                       |
| Interfaces | 인터페이스 이름도 클래스 이름과 같은<br>대문자 사용 규칙을 적용.                                                                                                                                                                                                                                                                                                                      | interface RasterDelegate;<br>interface Storing;                                                           |
| Methods    | 메서드의 이름은 동사<br><br>복합 단어일 경우 첫 단어는 소문자로<br>시작하며, 그 이후에 나오는 단어의 첫<br>문자는 대문자로 사용(CamelCase 방식)                                                                                                                                                                                                                                                               | run();<br>runFast();<br>getBackground();                                                                  |
| Variables  | 변수 이름의 첫 번째 문자는 소문자로 시작<br><br>각각의 내부 단어의 첫 번째 문자는 대문자로<br>시작<br><br>변수 이름이 언더바(_) 또는 달러 표시<br>문자로 시작하지 않도록 주의<br><br>변수 이름은 짧지만 의미 있어야 한다.<br><br>변수 이름의 선택은 그 변수의 사용 의도를<br>알아낼 수 있도록 의미적<br><br>한 문자로만 이루어진 변수 이름은<br>임시적으로만 사용하고 버릴 변수일 경우를<br>제외하고는 피해야 한다.<br>보통의 임시 변수들의 이름은 integer일<br>경우에는 i, j, k, m, n을 사용하고,<br>character일 경우에는 c, d, e를 사용한다. | Int i;<br>char c;<br>float myWidth;                                                                       |
| Constants  | 클래스 상수로 선언된 변수들과 ANSI<br>상수들의 이름은 모두 대문자로<br>각각의 단어는 언더바("_")로 분리<br>(디버깅을 쉽게 하기 위해서 ANSI 상수들의<br>사용은 자제하도록 하자.).                                                                                                                                                                                                                                           | static final int MIN_WIDTH = 4;<br>static final int MAX_WIDTH = 999;<br>static final int GET_THE_CPU = 1; |

## 좋은 프로그래밍 습관

인스턴스 변수와 클래스 변수를 외부에 노출하지 말고 대신 접근을 제공

인스턴스 변수 또는 클래스 변수를 합당한 이유없이 public으로 선언하지 말 것.  
(인스턴스 변수들은, 명시적으로 선언될 필요가 없는 경우도 많다)

인스턴스 변수가 public으로 선언되는 것이 적절한 경우는 클래스 자체가 어떤 동작(behavior)을 가지지 않는 데이터 구조(data structure)일 경우이다.

만약 class 대신 struct를 사용해야 하는 경우라면, class의 인스턴스 변수들을 public으로 선언하는 것이 적합하다.

클래스 변수와 클래스 메서드는 클래스 이름을 사용하여 호출!

클래스(static) 변수 또는 클래스 메서드를 호출하기 위해서 객체를 사용하는 것을 피하고 클래스 이름을 사용

```
ex)
classMethod(); // 좋은 사용법
AClass.classMethod(); // 좋은 사용법
anObject.classMethod(); // 나쁜 사용법
```

숫자는 바로 사용하지 말고 선언해서 변수 이름으로 접근!

숫자 상수는 카운트 값으로 for 루프에 나타나는 -1, 0, 1을 제외하고는, 숫자 자체를 코드에 사용하지 말 것.

변수에 값을 할당할 때 주의할 것들!

하나의 문(statement)에서 같은 값을 여러 개의 변수들에 할당하면 읽기가 어렵게 된다.

```
ex) fooBar.fChar = barFoo.lchar = 'c'; // 이렇게 사용하지 말자!
```

비교 연산자(equality operator: ==)와 혼동되기 쉬운 곳에 할당 연산자(assignment operator: =)를 사용하지 말 것

```
ex)
if (c++ = d++) { // 이렇게 사용하지 말자! (자바가 허용하지 않음)
 ...
}
```

```
ex) 아래처럼 작성할 것
if ((c++ = d++) != 0) {

}
```

실행시 성능 향상을 위해서 할당문(assignment statement)안에 또 다른 할당문을 사용하지 말 것.

```
ex) d = (a = b + c) + r; // 이렇게 사용하지 말자!
```

```
// 아래처럼 쓸 것
a = b + c;
d = a + r;
```

그 외 신경써야 할 것들

괄호

연산자 우선순위 문제를 피하기 위해서 복합 연산자를 포함하는 경우에는 자유롭게 괄호를 사용하는 것이 좋은 생각이다. 나는 연산자 우선 순위를 확실하게 알고 있다고 할지라도, 다른 프로그래머에게는 생소할 수 있다는 것을 기억하자.

```
ex)
if (a == b && c == d) // 이렇게 사용하지 말자!
if ((a == b) && (c == d)) // 이렇게 사용하자!
```



반환 값  
프로그램의 구조와 목적이 일치해야 한다.

```
ex)
if (booleanExpression) {
 return true;
} else {
 return false;
}
```

```
// 다음과 같은 형식이 더 좋음
return booleanExpression;
```

```
ex)
if (condition) {
 return x;
}
return y;
```

```
// 다음과 같은 형식으로 작성
return (condition ? x : y);
```

조건 연산자 '?' 이전에 나오는 식(expression)  
삼항 연산자(ternary operator - ?:) 에서 ? 이전에 이항 연산자(binary operator)를 포함하는 식(expression)이 있는 경우에는, 꼭 괄호를 사용해야 한다.

```
ex) (x >= 0) ? x : -x;
```

## 참고문헌

[국내. 저자 있음] Kwangshin(2015). "자바 코딩 규칙", [참조 사이트](#) 주석. (2016-05-09 방문)

## Dailyreport confirm field value rule

- 0: 미승인
- 1: 승인요청
- 2: 반려
- 3: 승인

## DB Excel정리 및 eXERD 파일



2017-09-22 최종 수정

## email tamplet

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="Content-Script-Type" content="text/xxxxjavascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
</head>
<body style="width: 730px;color: #010101;font-family: '돋움';font-size: 12px;">
<div style="width: 810px; background: #fff; border: 1px solid #ccc; border-radius: 4px 4px 4px 4px;overflow: hidden; cursor: default;">
<div style="background: #f5f5f5;border-bottom: 1px solid #ccc;box-sizing: border-box;height: 56px;margin: 0;overflow:
hidden;padding: 15px 20px;position: relative;text-overflow: ellipsis:white-space: nowrap;">
<h2 style="color: #333; font-weight: normal; font-size: 20px; line-height: 1.5; margin: 0; overflow: hidden; padding: 0; text-overflow:
ellipsis; white-space: nowrap;border: 0; vertical-align: baseline; font-family: Arial, sans-serif; visibility: visible;">
${header}
</h2>
</div>
<div style="position: relative; overflow: auto; padding: 20px; max-height: 500px; margin: 0;border: 0;font: inherit;vertical-align:
baseline;font-family: Arial, sans-serif;font-size: 14px">
<div style="border-bottom: 1px solid #ddd;padding-bottom: 10px;margin-bottom: 10px;">
<div style="margin-bottom: 10px;">
<div style="box-sizing: border-box;clear: both;position: relative;display: inline-block;width: 22%;cursor: context-menu; text-align:
center;">
<label style="word-wrap: break-word;color: #707070;">Team:<label>
${team}
</div>
<div style="box-sizing: border-box;clear: both;position: relative;display: inline-block;width: 22%;cursor: context-menu; text-align:
center;">
<label style="word-wrap: break-word;color: #707070;">Leader:<label>
${leader}
</div>
<div style="box-sizing: border-box;clear: both;position: relative;display: inline-block;width: 22%;cursor: context-menu; text-align:
center;">
<label style="word-wrap: break-word;color: #707070;">Reporter:<label>
${name}
</div>
<div style="box-sizing: border-box;clear: both;position: relative;display: inline-block;width: 22%;cursor: context-menu; text-align:
center;">
<label style="word-wrap: break-word;color: #707070;">Report Date:<label>
${date}
</div>
</div>
</div>
</div>
<div style="box-sizing: border-box;clear: both;padding: 4px 0 4px 145px;position: relative;margin: 1px 0;width: 100%;margin-bottom: 0
```

```

!important;">
<label style="word-wrap: break-word;color: #707070;float: left;text-align: right;width: 130px;margin-left: -145px;position:
relative;padding: 5px 0 0 0;">Title:</label>
<span style="max-width: 500px;height: 2.14285714em;line-height: 1.4285714285714;padding: 4px 5px;font-size: inherit;margin:
0;vertical-align: baseline;width: 100%;background: #fff;color: #333;font-family: inherit;">
${title}

</div>
<div style="box-sizing: border-box;clear: both;padding: 4px 0 4px 145px;position: relative;margin: 1px 0;width: 100%;margin-bottom: 0
!important;">
<label style="word-wrap: break-word;color: #707070;float: left;text-align: right;width: 130px;margin-left: -145px;position:
relative;padding: 5px 0 0 0;">Description:</label>
<span style="max-width: 500px;height: 2.14285714em;line-height: 1.4285714285714;padding: 4px 5px;font-size: inherit;margin:
0;vertical-align: baseline;width: 100%;background: #fff;color: #333;font-family: inherit;">
${description}

</div>
</div>
</div>
<div style="overflow: visible;min-height: 51px;height: 100%;margin: 0;padding: 10px;box-sizing: border-box;clear: both;position:
relative;width: 100%;text-align: right;white-space: nowrap;border-top: 1px solid #ccc;background: #f5f5f5;">
<div style="margin-top: 0; float: right;">
<a href="${url}" target="_blank"
style="cursor: pointer;display: inline-block;font-family: inherit;font-size: 14px;font-variant: normal;font-weight: normal;height:
2.14285714em;line-height: 1.42857143;margin: 0;padding: 4px 10px;text-decoration: none;vertical-align: baseline;white-space:
nowrap; border-radius: 3.01px;box-sizing: border-box;background: #f5f5f5;border: 1px solid #ccc;margin-bottom: 0;color:
#000;font-family: Arial, sans-serif;">
보고서 바로가기

</div>
</div>
</div>
</div>
</body>
</html>

```

일일 보고서			
Team: 영업1	Leader: test	Reporter: test1	Create Date: 2017-10-14
Title: asdfasdfasd Description: asdfasdfasd			
			<div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">보고서 바로가기</div>

## Internal Site Information

## JIRA Standards Guide

### Jira 개요

Jira는 웹 기반의 프로그램으로 **프로젝트 관리** 와 **버그** 추적 기능을 제공하는 도구이다. 통합된 프로젝트관리 기능과 **이슈추적**, 여러 가지 **형상 관리** 기능을 제공하며 달력과 **간트 차트**, 일정관리 기능도 제공한다.

Jira(지라)라는 이름은 ‘고지라(Gojira)’에서 따왔다. 2002년부터 Atlassian, Inc. 에서 개발하고 있다.

## Jira 사용 대상

개발 프로세스 내에 관여된 PM 및 모든 개발자를 대상으로 한다.

## 용어 정의

Abbreviations	Descriptions
사양(Specification)	제품의 기능 및 동작과 관련된 요구를 정의한 것을 의미한다.
문제점	<ul style="list-style-type: none"><li>- 동작 이상 조작 불능, 시스템 멈춤, 화면/소리 깨짐, 시스템 느려짐 등과 같이 쉽게 확인 가능하고, 고객 불만을 일으킬 수 있는 모든 것이 대상이다.</li><li>- 사양(Specification)적인 문제 (→ SW기준 ) 사용자 편이를 위한 동작을 포함하여 모든 동작이 사양적으로 정의되어 있고, 정의된 동작을 만족하지 못하면 모두 대상이다.</li></ul>
Ticket	Jira에서 issue라고 정의되는 내용. 본 문서에서는 issue type 'ISSUE' 와 구분하기 위해 ticket이라고 명명함.

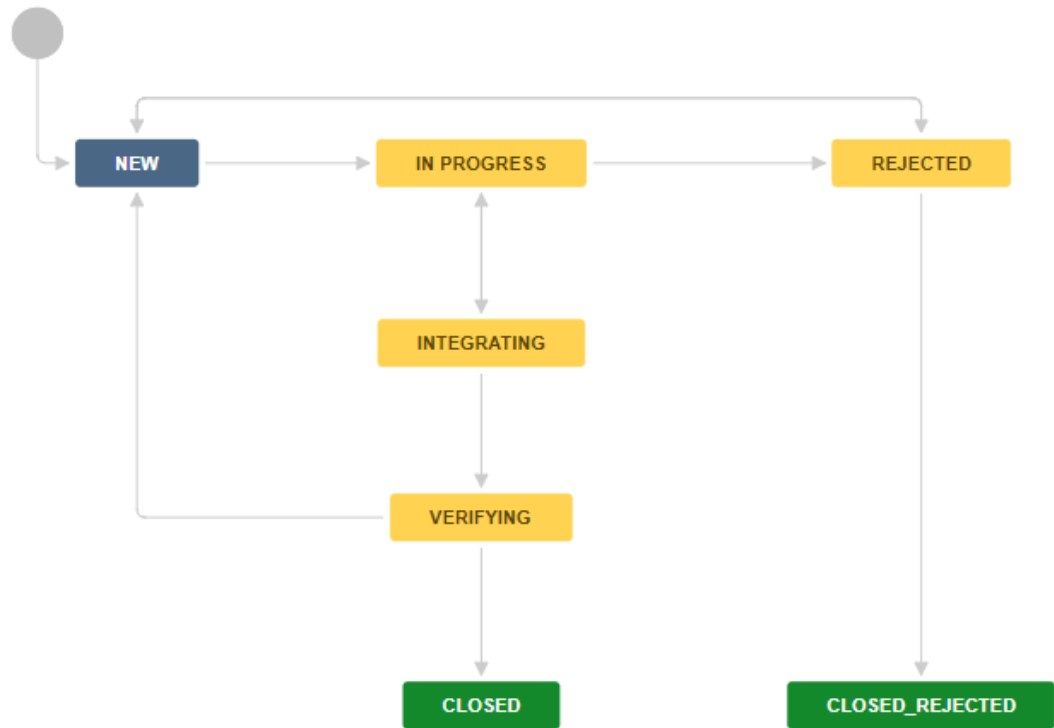
## SMS Project Ticket

### Bug Type

#### Ticket Flow 개요

기본적으로 다음 5단계를 거쳐서 Ticket은 생성되고 종료된다. (정상적인 Flow 로써 Reject 고려하지 않은 상황)

New → In Progress → Integrating → Verifying → Closed



Step Name (id)	Linked Status	Transitions (id)
Open (1)	NEW	In Progress (91) >> IN PROGRESS Rejected (111) >> REJECTED
In Progress (3)	IN PROGRESS	Integrating (31) >> INTEGRATING Processor_rejected (61) >> REJECTED
Integrating (4)	INTEGRATING	Verifying (41) >> VERIFYING Integrating_rejected (71) >> IN PROGRESS
Verifying (5)	VERIFYING	Closed (51) >> CLOSED Reopen (101) >> NEW
Closed (6)	CLOSED	
Rejected (7)	REJECTED	Closed_Rejected (81) >> CLOSED_REJECTED Reopen (101) >> NEW
Closed_Rejected (8)	CLOSED_REJECTED	

그림 1 : Bug Ticket Flow

## New (Ticket의 생성)

누가 생성하는가?

문제점을 발견한 모든 사람이 Ticket을 생성할 수 있는 Reporter가 될 수 있다.

문제의 기준은 무엇인가?

동작이상 또는 사양을 만족하지 못하는 모든 것을 말하며, 사양적으로 정의되지 않았지만 개선이 요구된다면 문제가 될 수 있다.

### 입력 항목

Summary (제목)

문제에 대한 간략한 설명

Description (설명)

아래 사항에 대한 기술이 반드시 포함되어야 한다.

- 문제 발생하기 전 조건(pre-condition)
- 재현 방법
- 문제 상술

Found Date

문제 발견 일자를 기록한다.

Priority (우선순위)

Top

시스템 Down, Booting 하지 못함과 같이 더 이상의 사용자 조작이 불가능한 경우 및 system이 Reset되는 경우.  
(Always)

A

시스템 Down, Booting 하지 못함과 같이 더 이상의 사용자 조작이 불가능한 경우 및 system이 reset되는 경우.  
(Sometimes, Once)

시스템은 동작 하지만 주 기능이 정상적으로 동작하지 않는 경우, Recovery를 위해 System rebooting을 해야 하는 문제

B

시스템은 동작하고, 주기능 이외에 문제가 발생은 하나, recovery를 위해 System rebooting을 하지 않아도 되는 경우.

글자 깨지는 문제 등 화면 표시상의 문제

C

문제점이 B정도이나, 재현하는 방법이 많은 조작을 요구 하는 경우.

System 메뉴 조작을 3~4회 이상 반복해야 재현 가능한 문제

D

사용자 눈에 띄지 않고, System에 영향이 미미한 문제.

Assignee (담당자)

각 상태에서 ticket을 처리하고 있는 사람

Occurrence

문제 발생 빈도를 말하는 것으로, Always/Sometimes/Once 가 있다.

SW Version

문제가 발생한 시스템의 SW Package 버전 정보

무엇을 어떻게 하는가?

해당 티켓을 할당 받은 담당자가 작업을 진행시 In progress로 변경하고

만약 해당 작업이 본인의 작업이 아니라고 판단이 되면 티켓의 상태를 Rejected로 변경하고

Assignee(담당자)를 본인 → PM 으로 변경한다. ( 이때 Comment에 반드시 Rejected로 변경한 사유를 작성한다. )

---

## In Progress (Ticket 처리 중 - Bug Fixing)

누가 하는가?

일반적으로 개발자가 진행하며, Spec 변경과 같이 특별한 경우에 한해 PM과 상담하여 진행 할 수 있다.

입력 항목

Fix Version/s

수정목표 version을 기입한다.

Due Date

Project 일정을 감안하여 기한을 기입

Planned Start, Planned End, Original Estimate

Due data 기한 내에 본인 일정을 감안하여 시작일, 완료일, 추정시간을 기입

Integrated CW

Integrate 예정Calendar Week를 기입한다.

SW\_Version\_Fix\_Planned

수정 예정 SW version을 기입한다.

Root cause

문제의 원인을 기입한다.

Counter Measure

문제 수정을 위한 방법을 제시한다.

Comment, Velocity(진척도), Log work(소요시간)

문제를 수정해 가는 모든 과정을 관련 근거와 함께 기입해야 한다.

예를 들어, 지연이 발생하여 일정이 변경 되었다면 원인과 내용을 상세히 기록해야 한다.

더불어, Ticket 과 관련되어 회의를 했다면 회의록을 첨부한다. 투입된 소요시간 및 진척도도 update 한다.

문제 수정이 완료되어 Integration을 요청할 때는, 문제의 원인과 수정 내용을 상세히 기입한다.

source code일 경우 Git에 Jira reference number 를 포함한 change-list 를 작성하여 commit한다.

#### Risk

문제 수정 시 따르는 위험도에 대해 기입한다.

Risk의 기준은 해당 문제처리 담당자가 판단한다.

통상적인 기준은 아래를 참조한다.

#### 상

설계/Interface/사양 변경이 필요한 경우

문제처리 담당자 / 프로젝트 일정에 5일 이상 영향을 주는 경우

#### 중

문제처리 담당자 / 프로젝트 일정에 2~3일 영향을 주는 경우

#### 하

문제처리 담당자 / 프로젝트 일정에 1일 이하 영향을 주는 경우

#### 무엇을 어떻게 하는가?

계획된 Plan 내에 수정을 하고, comment 항목에 수정 내용을 상세히 기입한 후, Integration 단계로 넘긴다.

만약 해당 작업이 본인의 작업이 아니라고 판단이 되면 티켓의 상태를 Rejected로 변경하고

Assignee(담당자)를 본인 → PM 으로 변경한다. ( 이때 Comment에 반드시 Rejected로 변경한 사유를 작성한다. )

추가로 일정 수정이 필요하면, PM과 협의하여 진행한다.

---

#### Integrating (수정된 사항을 SW Package에 적용 요구)

##### 누가 하는가?

PM에 의해 진행된다.

##### Integrated Version

정의된 SW 버전을 사용한다.

##### 무엇을 어떻게 하는가?

Weekly release 진행시 통합 빌드를 진행.

빌드시 문제가 있을 경우 Integrating\_rejected로 다시 In progress 상태로 처리한다. ( 이때 Comment 에 상태를 변경한



사유를 적는다. )

빌드시 이상이 없는 경우 Ticket을 Verification단계로 넘긴다.

#### 결과물

버전 생성시 각 모듈 별로 수정된 change-list를 Confluence에 정리한다.

Package 된 SW를 생성하고, 이를 이용하여 smoke-test 를 진행한 후 report로 정리한다.

마찬가지로 Git에 특정 폴더를 산출물 관리 폴더로 지정하여 생성된 버전을 등록하여 관리 한다.

---

## Verifying

### 누가 하는가?

Reporter에 의해 행해지며, 보통 PM주도하에 진행한다.

### 입력 항목

Test Result

테스트 결과를 기록한다.

SW\_Version\_Fixed

수정 및 검증이 완료된 SW의 version을 기입한다.

Test Delay Reason

테스트가 지연되었을 경우 지연 사유를 기록한다.

### 무엇을 어떻게 하는가?

Ticket에 주어진 Pre-condition 상태에서 문제의 재현을 검증한다.

검증시 문제가 발생하면 Reopen을 통하여 Ticket 상태를 다시 New로 변경한다. ( 이때 Comment 에 Reopen한 사유를 적는다. )

#### 결과물

Integration 후에 주어지는 change-list 검증 보고서

Verification 결과 통계 보고서

---

## Rejected

### 누가 하는가?

PM 이 Rejected 처리된 결과물과 Comment의 사유를 확인함.

### 입력 항목

Reject reason

Reject한 사유를 명확히 입력한다.

무엇을 어떻게 하는가?

PM이 해당 티켓을 확인하여 Reopen ( NEW )할지 Rejected( Closed\_rejected ) 할지 결정후 상태를 변경한다.

---

### Closed\_Rejected

Ticket에 언급된 문제점이 해당 담당자의 문제가 아니라 판단하거나 잘못 생성된 티켓의 경우 티켓이 종료되는 상태

---

### Closed

Ticket에 언급된 문제점이 수정되고 검증된 상태

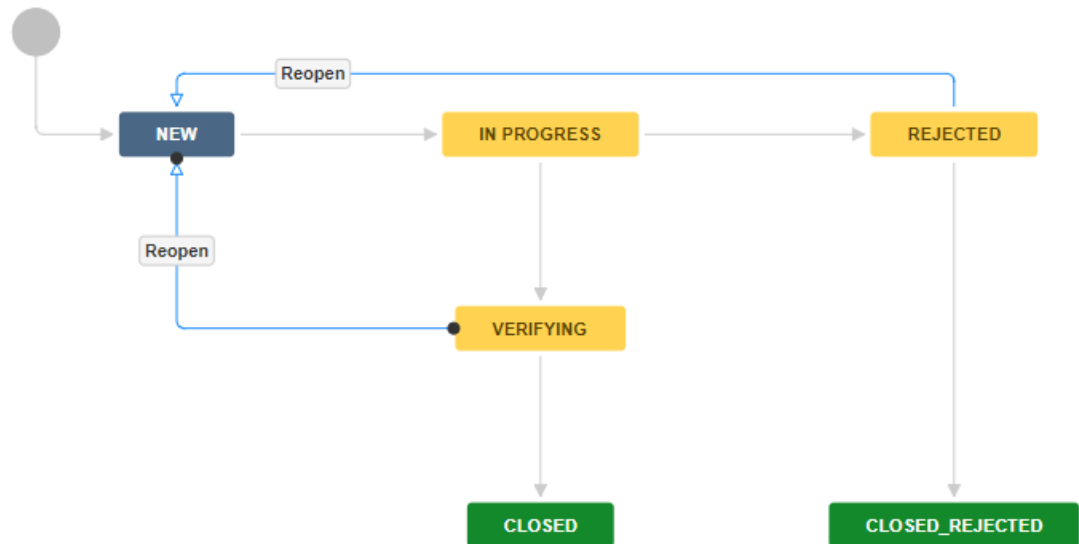
---

## Task Type

Ticket Flow 개요

개발 업무 이외의 업무 기록을 Task로 분류하며 기본 처리 flow는 아래와 같다. ( ex > Server Setting , DB 모델링 , UI 작업 등등 .... )

New → In Progress → Verifying → Closed



Step Name (id)	Linked Status	Transitions (id)
Open (1)	NEW	In Progress (11) >> IN PROGRESS
In Progress (2)	IN PROGRESS	Processor_rejected (21) >> REJECTED Verifying (71) >> VERIFYING
Rejected (3)	REJECTED	Closed_Rejected (31) >> CLOSED_REJECTED Reopen (61) >> NEW
Closed_Rejected (4)	CLOSED_REJECTED	
Closed (6)	CLOSED	
Verifying (7)	VERIFYING	Closed (81) >> CLOSED Reopen (61) >> NEW

그림 1 : Task Ticket Flow

New

누가 하는가?  
업무 담당자가 PM으로부터 업무를 할당 받아 본인의 업무에 대하여 생성한다.

입력 항목

Summary (제목)  
업무에 대한 간략한 설명

Description  
목표  
해당 업무의 목표 기입  
진행계획  
상세 업무 순서 및 내용 기입

Assignee  
각 상태에서 ticket을 처리하고 있는 사람

Priority  
Top  
프로젝트, 타 팀의 일정과 연결되어 있는 경우  
A

팀 내의 타인의 일정과 연결되어 있는 경우

B

본인의 일정끼리만 연결되어 있는 경우

Origin of Task

업무의 원인에 대해 기록한다.

Difficulty

업무 난이도 기입

업무 담당자 판단 하에 상, 중, 하 중 기입.

---

## In Progress

누가 하는가?

담당자가 본인의 업무에 대해 진행한다.

입력 항목

Due Date

Project 일정 및 팀 일정을 감안하여 기한을 기입 ( PM 작성 )

Planned Start, Planned End, Original Estimate

Due Date 기한 내에 본인 일정을 감안하여 시작일, 완료일, 추정시간을 기입 ( 담당자 작성 )

Comment

진행상황

업무 진행 상황 기입

결과

목표 대비 업무 결과 기입

성과물

업무 결과 성과물 기입 (ex) Git commit # 기록, document 첨부 등)

무엇을 어떻게 하는가?

계획된 Due Date 내에 업무를 수행 하고, comment 항목에 진행 상황 / 결과를 상세히 기입한 후, Verifying 단계로 넘긴다.

만약 해당 작업이 본인의 작업이 아니라고 판단이 되면 티켓의 상태를 Rejected로 변경하고

Assignee(담당자)를 본인 → PM 으로 변경한다. ( 이때 Comment에 반드시 Rejected로 변경한 사유를 작성한다. )

일정 수정이 필요하면, PM과 협의하여 진행한다.

결과물

업무 결과 성과물에 대해 comment에 기록 및 첨부한다.

---

## Verifying

누가 하는가?

PM이 담당의 업무에 대해 확인한다.

무엇을 어떻게 하는가?

계획된 Due Date 내에 업무를 수행 하였는지, 업무의 결과는 어떠한지 확인 후 Closed 단계로 넘긴다.

업무가 완료되지 않았다고 판단될시 Reopen ( NEW ) 하여 다시 업무를 진행할 수 있도록 한다. ( 이때 Comment에 Reopen한 사유를 적는다. )

---

## Rejected

누가 하는가?

PM 이 Rejected 처리된 결과물과 Comment의 사유를 확인함.

입력 항목

Reject reason

Reject한 사유를 명확히 입력한다.

무엇을 어떻게 하는가?

PM이 해당 티켓을 확인하여 Reopen ( NEW )할지 Rejected( Closed\_rejected ) 할지 결정후 상태를 변경한다.

---

## Closed\_Rejected

Ticket에 언급된 문제점이 해당 담당자의 문제가 아니라 판단하거나 잘못 생성된 티켓의 경우 티켓이 종료되는 상태.

---

## Closed

업무가 완료된 상태이다.

---

## SASI Project Ticket ( First Group )

### ToDo Type

Ticket Flow 개요

하루 업무에 대한 기록을 ToDo로 분류하며 기본 처리 flow는 아래와 같다.

New → In Progress → Verifying → Closed



Step Name (id)	Linked Status	Transitions (id)
Open (1)	NEW	In Progress (11) >> IN PROGRESS
In Progress (2)	IN PROGRESS	Verifying (21) >> VERIFYING
Verifying (3)	VERIFYING	Closed (31) >> CLOSED
Closed (4)	CLOSED	

그림 1 : ToDo Ticket Flow

New

누가 하는가?  
개발자 본인의 업무에 대하여 생성한다.

입력 항목

Summary (제목)

양식 → [ToDo]본인이름\_날짜(요일) 형식으로 작성한다. ( ex → [ToDo] 양승근\_07/11(화) )

#### Description

자유 양식이며 자신이 하루간 진행한 작업에 대하여 작성을 하며  
PM에게 알릴 내용 또는 막힌 부분등 일기 처럼 작성 하면 된다.

#### Assignee

티켓 생성자 자신을 지정하면 된다.

---

### In Progress

#### 누가 하는가?

담당자가 본인의 업무에 대해 진행한다.

#### 입력 항목

##### Log work(소요시간)

자신이 하루동안 소요한 업무 시간을 기록한다.

#### 무엇을 어떻게 하는가?

업무가 완료가 되었을 경우 담당자( Assignee )를 자신에서 PM으로 변경한다.

하루의 업무를 완벽히 수행하고 , Description 또는 Comment 항목에 진행 상황을 상세히 기입한 후, Verifying단계로 넘긴다.

---

### Verifying

#### 누가 하는가?

PM이 담당의 업무에 대해 확인한다.

#### 무엇을 어떻게 하는가?

특이 사항이 있는지 확인하고 별다른 문제가 없으면 티켓을 Closed 상태로 넘겨서 종료한다.

---

### Closed

업무가 완료된 상태이다.

## node.js와 android 통신규약

android가 반환받는 값 :

mysql DB의 테이블 데이터가 성공적으로 수정될 때 : 'changed'

mysql DB의 테이블에 데이터가 성공적으로 삽입 될 때 : 'inserted'

mysql DB의 테이블에 데이터가 성공적으로 삭제 될 때 : 'deleted'

token이 만료될때 : 'false'

token 발행에 성공하면 토큰값을 받는다.

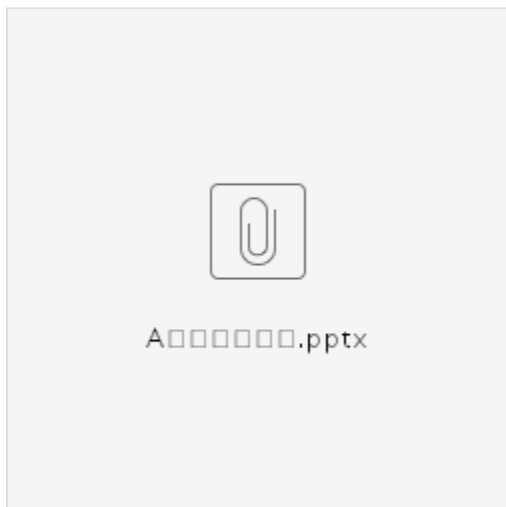
프로필 비밀번호 수정시 :

토큰의 정보가 유효하면 토큰에 있는 비밀번호와 입력받은 비밀번호를 비교 비교값이 다르면 'incorrect' 비교값이 맞다면 'correct' 응답.

토큰값이 유효하다면 작업이 바로 해당하는 작업을 실행 유효하지 않다면 android에서 새로운 토큰을 발행 받을 수 있도록 해주어야 한다. 즉, false값을 받으면 토큰을 재발행 하고 다시 작업을 실행해야 한다.

모든 통신에는 jwt 토큰으로 유저정보를 받아서 검증

## Project Topic



## restful 사양서





restful□□□.xlsx

WEB prototype



smsPrototype(web).pdf

<https://goo.gl/ZZKaCv>

발표자료



web□□□□.mp4



presentatioin.pptx



□□ □□□□.docx

팀장에게 보고서의 승인요청된 데이터체크 쿼리

```
select a.no as dayNo, a.confirm, b.dep_no from day_report a, user b where b.dep_no=1 and a.confirm=1 group by a.no;
```