

이 글을 보시는 수많은 고수님들께 부탁드립니다.

아래 글 및 GitHub 소스에서 잘못된 내용이 있거나 개선이 필요한 부분을 지적해 주시면 즉시 수정하도록 하겠습니다.

혹시, 당장 수정이 어려운 상황이라면 개선 포인트로 반드시 메모하고 향후에도 적용하도록 노력하겠습니다. 감사합니다!!

<https://sendjay.com> by hushsbay@gmail.com

1. What is sendjay ?

sendjay는 사내 메신저 PC Web 및 Mobile App 전체 소스를 GitHub에서 지속적으로 제공하는 '프로젝트'입니다.

sendjay는 GitHub 소스를 이용하여 본인의 회사 환경에 맞게 사내 메신저를 직접 구축하는 개발자를 위한 프로젝트입니다.

https://github.com/hushsbay/sendjay_tiny (Server & Web Client)

https://github.com/hushsbay/sendjay_and (Android Client)

1. 멀티소켓서버(NodeJS, socket.io, Redis)를 지원합니다.
2. 귀사의 포털 사이트에서 Web 메신저를 자동으로 백그라운드(Embedded) 실행 및 소켓 재연결처리를 가능하게 합니다.
3. Mobile App에서는 socket.io 라이브러리를 사용하며 UI는 Webview로 Web 모듈을 재사용하여 구현합니다.
 - 1) Android : Kotlin. Foreground Service 자동 재시작 및 소켓연결 유지, FCM Push 메시지, Battery Optimization
 - 2) iOS : Swift. 2022년 개발 예정입니다.

sendjay project는 iOS/Android 초기와는 달리, 현재의 디바이스/네트워크/OS/Language 등 환경에서 PC용(앱) 메신저를 충분히 커버하는 Web용 사내메신저 개발이 가능하다고 보고, 별도 로컬DB를 full로 사용하지 않고도 라이브러리(socket.io) + Webview UI 기반으로 충분히 구축할 수 있을 것이라는 판단으로 시작한 것입니다.

sendjay project는 사내메신저용 GitHub 소스를 제공하는 것이 최종 목표이기 때문에, 구조와 기능을 파악하고 비즈니스 로직이 어떻게 구성되어 있는지 이해하여, 개발자분들께서 소스를 활용해 자체 개발이 가능하도록 하는데 이 글의 목적이 있습니다.
따라서, 여기서는 메신저의 일반적인 사용법과 (구글링으로 쉽게 찾아볼 수 있는) 프로그래밍 방법에 대해서는 설명하지 않습니다.

<https://sendjay.com>은 실제 메신저 운영을 위한 사이트가 아닌 sendjay project 목적을 위한 테스트 사이트로 이해해 주시길 바랍니다.
물론, 이 사이트는 GitHub에 최종 업데이트된 소스와 100% 동일하게 구성되어져 있고 실제 사용처럼 테스트가 가능합니다.
다만, 특정 개발 조직이 아닌 개인 한명이 자비로 만든 소규모 사이트이므로 성능, 용량 부족뿐 만 아니라 보안적으로도 취약할 수 밖에 없음을 다시 한번 양지해 주시기 바랍니다. (예: 파일업로드 용량을 테스트 용도로만 10MB, 저장시간도 1시간으로 제한 등)
또한, **sendjay의 모든 데이터는 테스트 목적이므로 별도의 공지없이 삭제될 수 있음을 양해해 주시면 감사하겠습니다.**

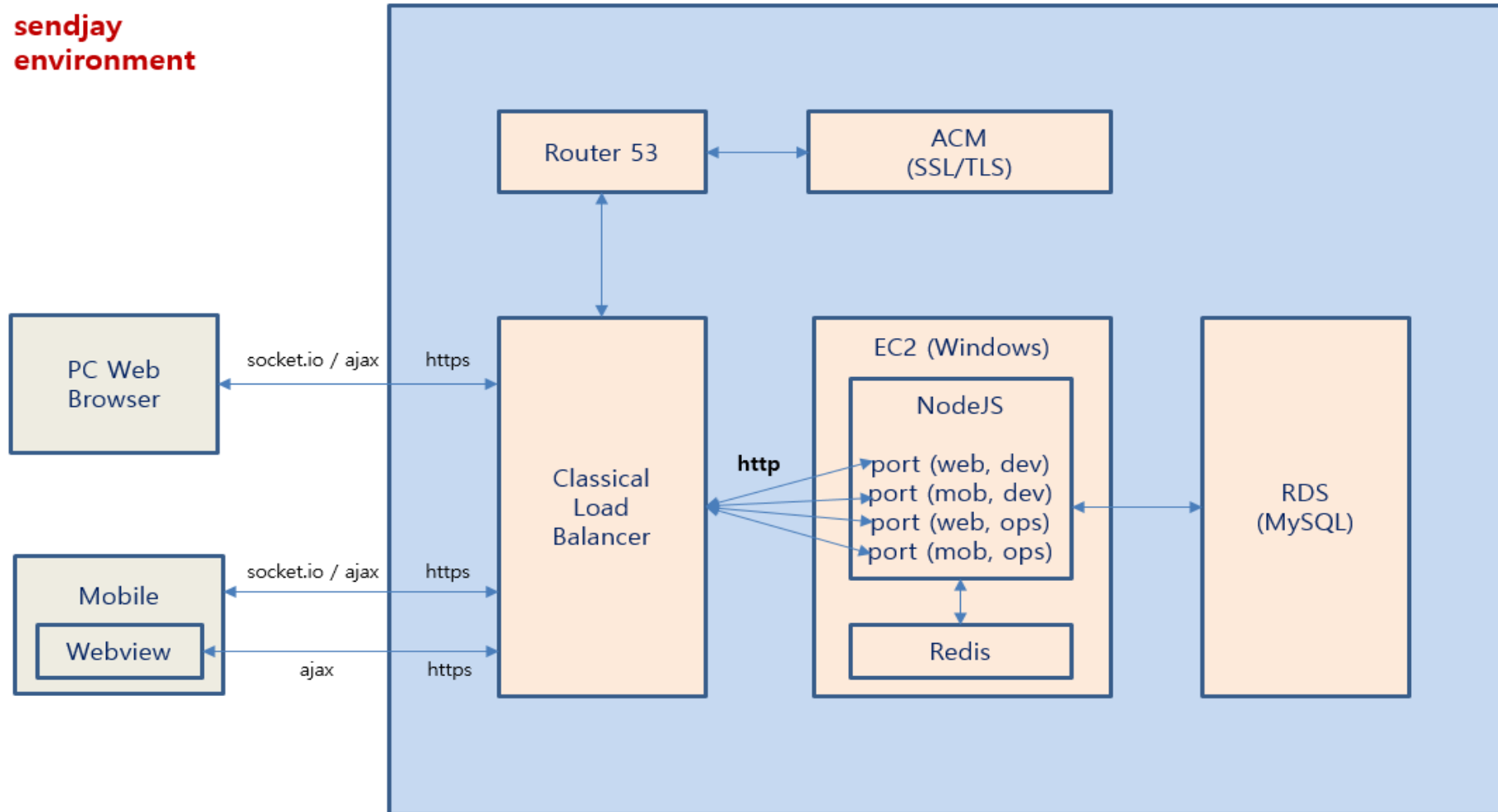
sendjay project는 socket.io 라이브러리를 사용하는데, 소켓 통신을 한번도 경험하지 않은 개발자분들께 미리 말씀드립니다.

1. Web을 먼저 개발하여 안정화한 후 Mobile을 진행하면 좋겠습니다.
2. 사내메신저가 'Mobile 지원이 반드시 필요로 하는 것은 아니다'라고 생각합니다. 각각 환경마다 다를 수 밖에 없지만 실제로 제가 속한 조직(3천여명)에선 수년동안 Web 메신저만 활발하게 사용중이며 아직까지는 Mobile에 대한 사용자 요구가 그리 높지 않다는 말씀을 드리고 싶습니다.
3. PlayStore/AppStore와는 달리, In-House(Enterprise) App 업데이트의 경우 사용자의 개입(Action)이 반드시 필요한 부분인데 개발자로서는 늘 불편하고 부담스러운 이슈입니다. 아시다시피, Web은 배포의 부담을 없애줍니다.
하지만, 모바일 환경에서 네트워크가 항상 재연결되는 환경을 고려하면서 소켓통신을 구현해야 하는 사내메신저를, 그것도 Webview를 UI로 하여 사용자 요구에 맞게 개발하는 것이 과연 만족스러운 지에 대한 부담감(소켓통신과 Webview의 궁합)은 현재 1.0을 올리는 시점에서도 일부 남아 있긴 합니다. 하지만, IT 환경은 늘 발전하며 더 만족스러운 결과가 나올 때까지 계속 개선, 보완, 도전해 나가면 못 할 것도 없다는 자신감도 생긴다는 것을 말씀드리고 싶습니다.
4. sendjay는 AWS 인프라를 사용하는데 그것은 개인적인 상황에 따라 구성한 환경일 뿐임을 알려 드립니다.
5. 참고로, 총 소스 길이는 xml(설정) 등을 제외한 직접 코딩한 부분 약 1만2천Lines 정도 됩니다.

2. 아키텍처 (Tiny version)

sendjay
environment

AWS



* Window Server 2019 (AWS EC2 t3 micro 1대 – 2 Core CPU, 1GB RAM) : https on Router53, ACM(TLS) and Classic Load Balancer
- sendjay는 테스트 목적으로 1대의 서버에 개발/운영서버용 포트와 웹/모바일용 포트를 모두 관리하고 있습니다.

* MySQL (ver 5.7, AWS RDS)

* Redis (ver 3.0.504)

* NodeJS (ver 14.15.4) : express, socket.io(3.1.0), socket.io-redis, ioredis, firebase-admin, jsonwebtoken..

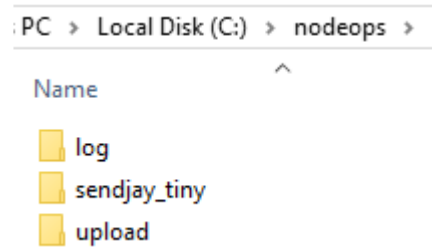
* jQuery (ver 3.4.1, tested on Chrome and Edge browser only)

* Kotlin (ver 1.4.32, AndroidStudio 4.1.3)

* iOS (Swift) : not developed yet

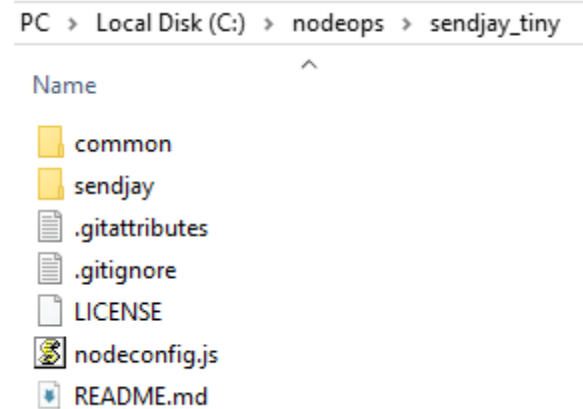
3. Server & Web Client (NodeJS)

* AWS EC2 Window Server에 아래와 같이 폴더가 구성되어 있습니다.



upload 폴더는 메신저에서 파일 전송시 저장되는 공간입니다.

* sendjay_tiny는 스케일이 크지 않은 소규모용이라고 생각하시면 되겠습니다.



1. 여기에는 common과 sendjay라는 각각의 NodeJS 프로젝트가 있습니다.
2. GitHub에서도 위와 동일한 구조로 확인 가능합니다. (아래 참조)

→ ↺ 🏠 github.com/hushsbay/sendjay_tiny



Search or jump to...



[Pull requests](#)

[Issues](#)

[Marketplace](#)

[Explore](#)

🔒 [hushsbay](#) / [sendjay_tiny](#) Private

[Code](#)

[Issues](#)

[Pull requests](#)

[Actions](#)

[Projects](#)

[Security](#)

[Insights](#)

🔗 [main](#) ▾

🔗 1 branch

🔖 0 tags



hushsbay m



common

m



sendjay

m



.gitattributes

Initial commit



.gitignore

m



LICENSE

Initial commit



README.md

Initial commit

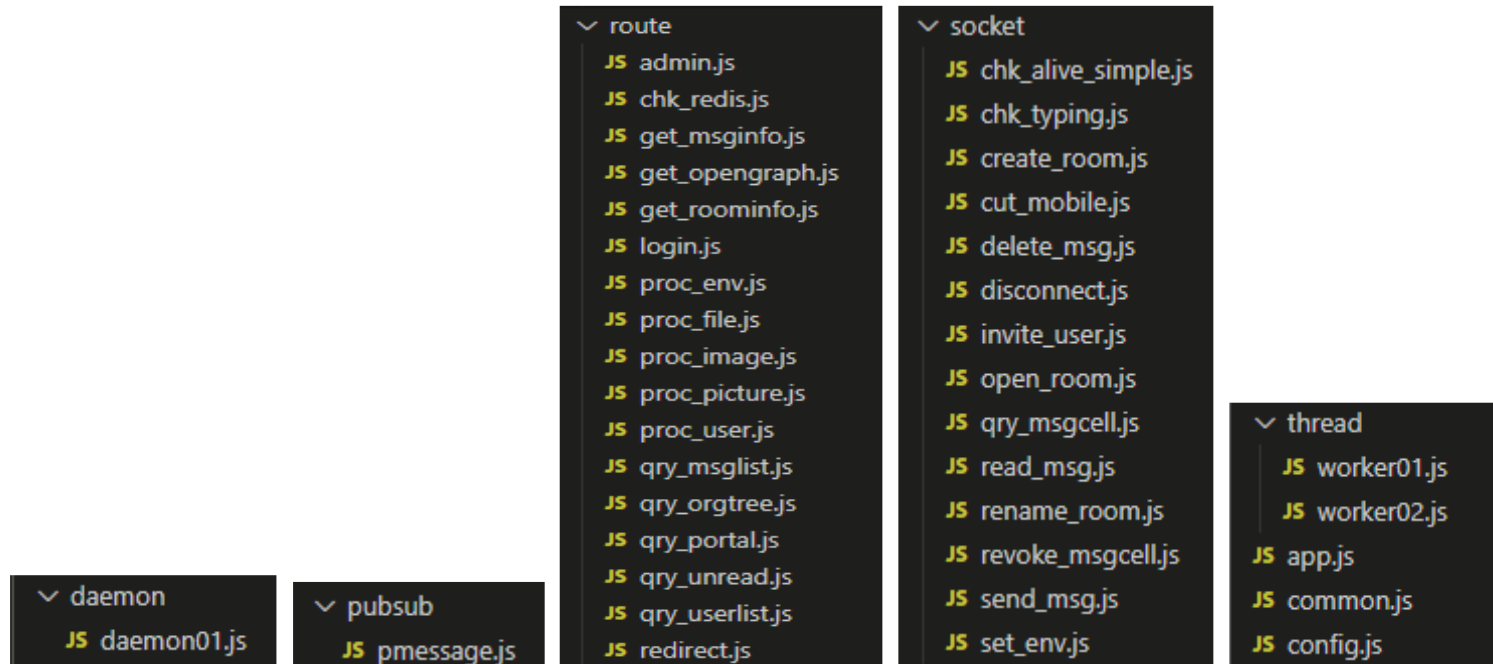
3. 위 파일중에서 nodeconfig.js는 향후 프로젝트가 여러 개일 때 공유하려고 만든 환경설정파일로 아래와 같습니다.
- nodeconfig.js.github를 nodeconfig.js로 rename하여 사용하시면 됩니다.

```
module.exports = {
  "mysql": {
    "hushsbay": {
      "host": "hushsbay.ap-northeast-2.rds.amazonaws.com",
      "poolsize": 20,
      "port": 3306,
      "admin": {
        "user": " ",
        "pwd": " "
      }
    },
  },
  "redis": {
    "host": "127.0.0.1",
    "port": 6379,
    "pwd": " "
  },
  "jwt": {
    "algo": " ",
    "key": " ",
    "expiry": "36500 days" //for not only web but also mobile
  },
  "crypto": {
    "key": " "
  }
}
```

```
▼ SENDJAY_TINY
  ▼ common
    JS config.js
    JS wscommon.js
    JS wsmysql.js
```

4. common 프로젝트에는 프로젝트가 여러 개일 때 공유하려고 만든 wscommon.js(공통함수)와 MySQL 관련 함수를 모아 놓은 wsmysql.js가 있습니다. (위 오른쪽 그림 참조)

* sendjay project에는 서버와 웹클라이언트 파일이 혼재되어 있는데 아래가 서버용 폴더(파일)들입니다.



1. 아래는 package.json 파일입니다. (package.json.github를 package.json으로 rename하여 사용하시면 됩니다)

```
{
  "name": "sendjay",
  "version": "1.0.0",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.17.1",
    "firebase-admin": "^9.6.0",
    "fluent-ffmpeg": "^2.1.2",
    "fs-extra": "^9.1.0",
    "ioredis": "^4.17.3",
    "jsonwebtoken": "^8.5.1",
    "mime": "^2.5.0",
    "multer": "^1.4.2",
    "open-graph-scraper": "^4.7.1",
    "redis": "^2.8.0",
    "socket.io": "^3.1.0",
    "socket.io-redis": "^6.0.0"
  },
  "devDependencies": {}
}
```


2. 아래는 각각 start.bat (Web용), start1.bat (Mobile용) 입니다. 앞서 말씀드린 대로 sendjay project에서는 소규모 테스트 사이트를 운영하므로 Redis를 이용한 Multi Socket Server를 Window Server 1대로만으로 포트를 나누어 Web과 Mobile로 단순 분리한 것입니다.

- 1) 아래 https가 아니고 http인 것은 AWS에서 Router53, ACM(Amazon Certificate Manager), CLB(Classic Load Balancer)가 SSL/TLS를 맡고 내부적으로는 EC2 Window Server와는 http로 통신하는 구성으로 되어 있기 때문입니다.
- 2) 여기서는 PM2 등의 다른 모듈에 대한 추가 설명없이 node app으로 스타트하는 것만 진행합니다. (.bat 이용)
 - start.bat.github를 start.bat으로 rename하여 사용하시면 됩니다.

```
set NODE_CONFIG=c:/nodeops/sendjay_tiny/nodeconfig.js
set MODULE_COMMON=c:/nodeops/sendjay_tiny/common/wscommon.js
set MODULE_MYSQL=c:/nodeops/sendjay_tiny/common/wsmysql.js
set MYSQL_DBINST=
set MYSQL_USER=
set MYSQL_SCHEMA=jayops
set HTTP_METHOD=http
set HTTP_PORT=80
set SOCK_PORT=3050
set SOCK_NAMESPACE=jayops
set REDIS_DB=0
set REDIS_FLUSH_SERVER=Y
set FCM_KEY_FILE=c:/keystore/fcm_key.json
set LOG_PATH=c:/nodeops/log/sendjay_tiny
set UPLOAD_PATH=c:/nodeops/upload/sendjay_tiny
node app
```

```
set NODE_CONFIG=c:/nodeops/sendjay_tiny/nodeconfig.js
set MODULE_COMMON=c:/nodeops/sendjay_tiny/common/wscommon.js
set MODULE_MYSQL=c:/nodeops/sendjay_tiny/common/wsmysql.js
set MYSQL_DBINST=
set MYSQL_USER=
set MYSQL_SCHEMA=jayops
set HTTP_METHOD=http
set HTTP_PORT=81
set SOCK_PORT=3051
set SOCK_NAMESPACE=jayops
set REDIS_DB=0
set REDIS_FLUSH_SERVER=N
set FCM_KEY_FILE=c:/keystore/fcm_key.json
set LOG_PATH=c:/nodeops/log/sendjay_tiny
set UPLOAD_PATH=c:/nodeops/upload/sendjay_tiny
node app
```

3. 데몬 실행을 위한 .bat 파일은 아래와 같습니다. (예: 저장기한이 만료된 파일 및 폴더 삭제)

PC > Local Disk (C:) > nodeops > sendjay_tiny > sendjay > daemon

Name

daemon01.js

start_d01.bat

start_d01.bat - Notepad

File Edit Format View Help

```
set NODE_CONFIG=c:/nodeops/sendjay_tiny/nodeconfig.js
set MODULE_COMMON=c:/nodeops/sendjay_tiny/common/wscommon.js
set MODULE_MYSQL=c:/nodeops/sendjay_tiny/common/wsmysql.js
set MYSQL_DBINST=
set MYSQL_USER=
set MYSQL_SCHEMA=jayops
set LOG_PATH=c:/nodeops/log/sendjay_tiny
set UPLOAD_PATH=c:/nodeops/upload/sendjay_tiny
node daemon01
```

* sendjay 프로젝트에서 웹클라이언트용 폴더(파일)은 아래와 같습니다.

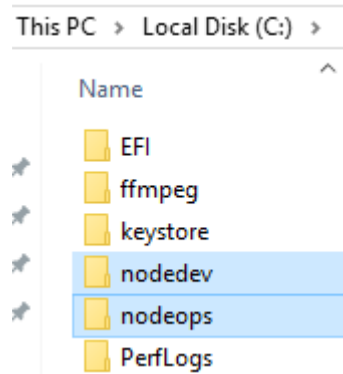
```

  public
  > app
  > common
    JS jay_chat.js
    # jay_common.css
    JS jay_common.js
    JS jay_main.js
    JS jay_worker.js
  > img
  > jay
    <> chat.html
    <> dummy.html
    <> index.html
    <> popup.html
  > plugin
    > images
    JS jquery-3.4.1.min.js
    # jquery-ui-smoothness-1.12.1.min.css
    JS jquery-ui-smoothness-1.12.1.min.js
    JS jquery.cookie.js
    JS lodash.min.js
    JS moment-timezone-with-data-1970-2...
    JS moment.min.js
    JS socket.io.min.js
    JS socket.io.min.js.map
```

```

    <> appdown.html
    {} applist.json
    <> demoplay.html
    <> erp_portal.html
    <> healthcheck.html
    <> portal.html
    <> user_manage.html
```

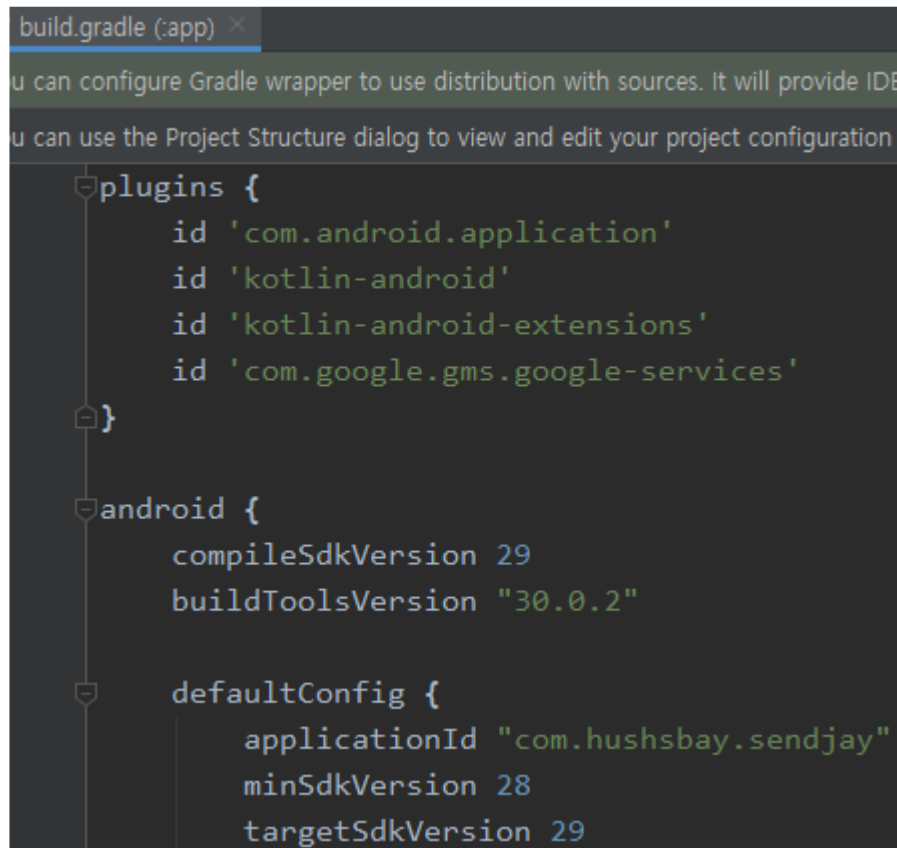
* 위는 서버에서 nodeops 폴더로 구성된 내용입니다. (Web용 포트와 Mobile용 포트 구분해서 1대의 서버에서 운영)



1. nodeops 폴더는 포트만 다를 뿐 nodeops와 동일하게 동일하게 구성된 개발용 환경입니다.
2. 운영서버는 nodeops이고 개발서버는 nodeops인 것입니다.
3. 테스트 및 비용 관계상 1대의 서버로 구성한 것이므로 실제 일반적인 로드밸런싱 환경에서는 맞지 않는 구성입니다.

4. Android (Kotlin)

* framework

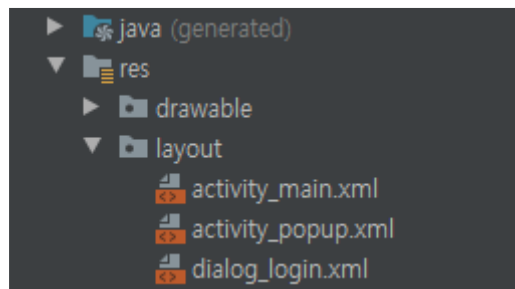
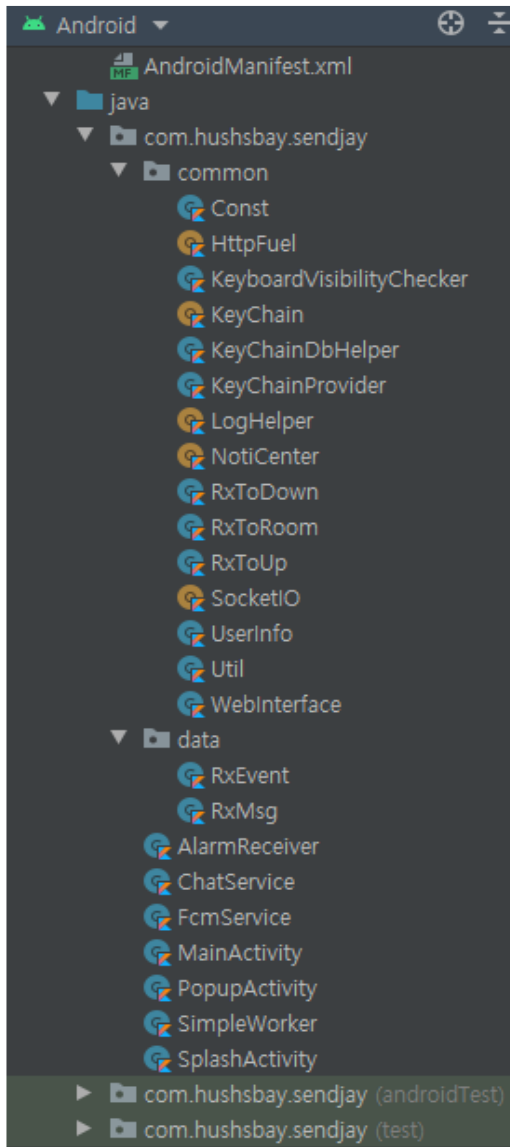


The screenshot shows the 'build.gradle (:app)' file in an IDE. It contains the following Kotlin code:

```
build.gradle (:app) ×  
You can configure Gradle wrapper to use distribution with sources. It will provide IDE  
You can use the Project Structure dialog to view and edit your project configuration  
  
plugins {  
    id 'com.android.application'  
    id 'kotlin-android'  
    id 'kotlin-android-extensions'  
    id 'com.google.gms.google-services'  
}  
  
android {  
    compileSdkVersion 29  
    buildToolsVersion "30.0.2"  
  
    defaultConfig {  
        applicationId "com.hushsbay.sendjay"  
        minSdkVersion 28  
        targetSdkVersion 29
```

```
dependencies {  
  
    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"  
    implementation 'androidx.core:core-ktx:1.3.2'  
    implementation 'androidx.appcompat:appcompat:1.2.0'  
    implementation 'com.google.android.material:material:1.3.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'  
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'  
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'  
    implementation 'com.google.firebase:firebase-messaging:21.1.0'  
    implementation "androidx.work:work-runtime-ktx:2.5.0"  
  
    testImplementation 'junit:junit:4.13.1'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'  
  
    implementation('io.socket:socket.io-client:2.0.0') { exclude group:  
    implementation 'com.google.code.gson:gson:2.8.6'  
    implementation 'com.github.kittinunf.fuel:fuel:2.2.1'  
    implementation 'com.github.kittinunf.fuel:fuel-android:2.2.1'  
    implementation 'com.github.kittinunf.fuel:fuel-coroutines:2.2.1'  
    implementation 'com.github.kittinunf.fuel:fuel-gson:2.2.1'  
    implementation "io.reactivex.rxjava2:rxandroid:2.0.2"  
    implementation "io.reactivex.rxjava2:rxjava:2.1.13"  
    implementation "log4j:log4j:1.2.17"  
    implementation "de.mindpipe.android:android-logging-log4j:1.0.3"  
  
}
```

* file structure : GitHub에서도 아래와 동일한 구조로 확인 가능합니다. (다음 페이지 참조)



→ ↻ 🏠 github.com/hushsbay/sendjay_and



Search or jump to...

[Pull requests](#)

[Issues](#)

[Marketplace](#)

[Explore](#)

🔒 [hushsbay](#) / [sendjay_and](#) Private

[Code](#)

[Issues](#)

[Pull requests](#)

[Actions](#)

[Projects](#)

[Security](#)

[Insights](#)

🔗 [main](#) ▾

🔗 1 branch

🏷 0 tags



hushsbay m



.idea

m



app

m



gradle/wrapper

Initial commit



.gitattributes

Initial commit



.gitignore

Initial commit



LICENSE

Initial commit



README.md

Initial commit



build.gradle

m

* 주요 파일 내용

1. MainActivity.kt

- 1) Socket 및 필요한 Rest 통신만 Kotlin에서 담당하고 UI는 모두 Webview에서 지원합니다.
- 2) 이는 PC Web Browser용으로 개발된 Web 모듈을 Mobile에서 재사용하고자 하는 것이 목적입니다.
- 3) 채팅방 목록보기용 및 채팅 전송용 Webview는 각각 분리하지 않고 하나의 MainActivity에 통합해 관리합니다.

2. ChatService.kt (Foreground Service, Daemon Thread)

- 1) socket.io-client를 이용해 NodeJS 소켓서버와 데이터 통신 및 Mobile Connection을 관리합니다.
- 2) PublishSubject, Observable을 이용해 Activity와 통신합니다.
- 3) Notification (from socket or FCM)을 처리합니다.
- 4) 사용자에게 의한 App 중지시 ChatService 본인을 리스타트시키도록 요청합니다.

3. PopupActivity.kt

- 1) Webview를 통해 채팅방에서 선택한 이미지를 보여주거나 다운로드합니다.
- 2) Webview를 통해 채팅방에서 선택한 mp4 동영상을 스트리밍합니다.
- 3) 참고로, 파일 다운로드를 MainActivity내 Webview를 통해 처리합니다.

4. AlarmReceiver.kt (Broadcast Receiver)

- 1) 디바이스 재부팅후 ChatService 리스타트합니다.
- 2) 사용자에게 의한 App 중지시 ChatService 본인을 리스타트시킵니다.

5. SimpleWorker.kt (Work Manager)

- 1) ChatService가 비정상적으로 종료될 경우를 대비하여 최소 주기로 체크하여 멈춰 있으면 리스타트합니다.

6. FCMService.kt

- 1) Push Token 등록 이벤트 핸들링합니다.
- 2) Push 메시지 도착시 Notification 처리합니다. (socket-disconnected client에게만 발송)

* App 다운로드 및 업데이트 (In-House App)

1. sendjay는 회사(조직)내 구성원들을 대상으로 하므로 PlayStore를 통하지 않고 자체 서버에 App을 올리고 버전을 관리하여 App에서 그 버전을 체크하고 자동으로 다운로드하게 하여 App 업데이트를 진행합니다.
2. Android는 iOS와는 달리 현재 비용없이 자체 서버에서 배포 가능합니다.
3. App 배포 (from Android Studio) – 최초 배포시에는 버전은 1.0 그대로 유지하면 될 것입니다.

1) 버전 설정

```
android {  
    compileSdkVersion 29  
    buildToolsVersion "30.0.2"  
  
    defaultConfig {  
        applicationId "com.hushsbay.sendjay"  
        minSdkVersion 28  
        targetSdkVersion 29  
        versionCode 1  
        versionName "1.0"  
    }  
}
```

버전 업데이트시 먼저 build.gradle에 버전을 변경합니다 (예: 1.0 -> 1.1)

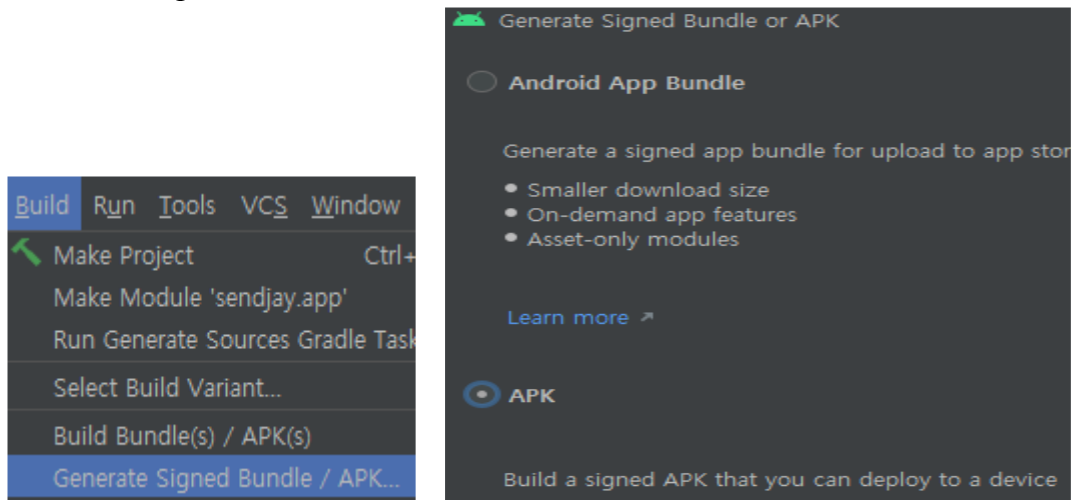
```

{} applist.json X
sendjay > public > {} applist.json > ...
1  {
2      "sendjay_and" : { "version" : "1.0", "filename" : "sendjay.apk", "path" : "/app/" },
3      "sendjay_ios" : { "version" : "1.0", "filename" : "sendjay.plist", "path" : "/app/" },
4      "webview_main_version" : "1.0",
5      "webview_chat_version" : "1.0",
6      "webview_popup_version" : "1.0",
7      "etc" : { "screenoff" : 10000, "screenon" : 3000 },
8      "code" : "0",
9      "msg" : ""
10 }

```

서버에 저장할 applist.json내 app version도 위 build.gradle내 버전과 동일하게 맞추어 줍니다. (sendjay_and의 version)

2) 아래처럼 Signed Bundle을 생성합니다.



Module sendjay.app

Key store path

Create new... Choose existing...

Key store password

Key alias

Key password

☐ Remember passwords

Key store path: C:\keystore\and\hushsbaykey.jks

Password: Confirm:

Key

Alias: hushsbay_key

Password: Confirm:

Validity (years): 25

Certificate

First and Last Name: SB Lee

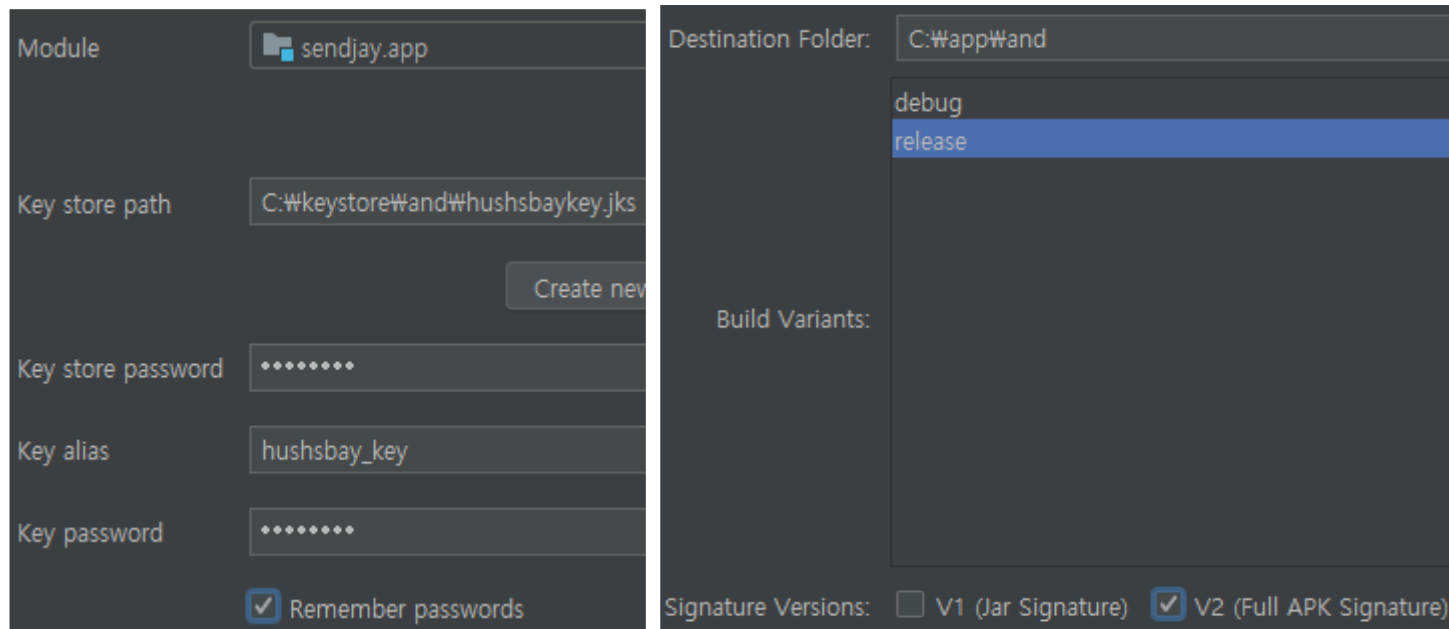
Organizational Unit: hush

Organization: sbay

City or Locality: seoul

State or Province: seoul

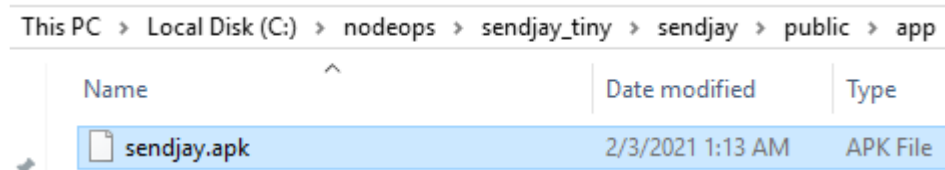
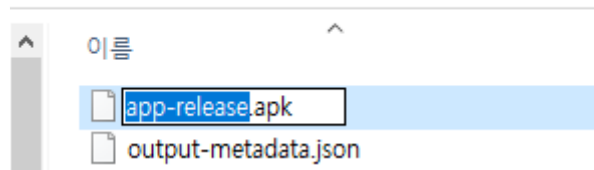
Country Code (XX): 82



3) 아래 그림처럼 app-release.apk를 sendjay.apk로 rename후 서버 업로드합니다. (복사 등)

- 기존에 같은 이름이 있다면 먼저 삭제후 rename. output-metadata.json은 특별히 신경쓸 필요 없습니다.

내 PC > 로컬 디스크 (C:) > app > and > release



- 위 오른쪽 그림처럼 운영서버에만 파일 복사하면 되며 필요시 .gitignore에 *.apk 넣으면 됩니다.

4) applist.json을 서버에 전송합니다. (git push 등)

5) 이제 모바일 웹에서 appdown.html을 열어서 최초 App 다운로드 받으면 됩니다.

6) 버전체크/App업데이트는 안드로이드 소스내 MainActivity.kt의 chkUpdate() 참조하시기 바랍니다.

5. Web & Webview

- * PC 애플리케이션, Android/iOS App으로 개발되는 일반적인 메신저와는 달리 사내메신저용 sendjay project는 Web으로 개발된 UI를 Mobile Webview를 통해 모두 재사용함으로써 **IT 규모가 작은 기업에서 저비용으로 Web/Android/iOS를 모두 개발/운영할 수** 있도록 부담을 줄여줍니다. (물론, 개발자가 Web/Android/iOS 지식을 어느 정도는 고루 갖춰야 할 필요는 있습니다)
- * 또한, PC App 및 Mobile App은 항상 배포 절차의 번거로움과 사용자의 불편이 항상 개발자의 부담으로 남아 있는데, PC Web 및 Mobile Webview는 배포(업데이트)의 부담을 대폭 줄여줍니다. (Native Mobile App 수정후엔 App 업데이트 필요)
- * 특히, Web 메신저는 마우스 움직임 감지 등 일부 Native한 구현 기능을 제외하고는 PC 애플리케이션 못지 않은 기능과 성능을 보장되는데 주요 기능을 나열하면 아래와 같으며 PC 애플리케이션과 별 다른 차이가 없음을 알 수 있습니다.

1. 자동 실행

sendjay는 사내메신저로서 독립적으로 로그인하는 것보다 기업 ERP나 Groupware 포털에서 SSO로 자동으로 실행되도록 하면 더 효율적인데 별도 실행되지 않고 브라우저 탭에 임베디드되어 백그라운드에서 실행되는 방식으로 처리 가능합니다. (PC용 메신저가 시스템트레이에 들어가 있는 모양과 유사하며 HTML5 indexDB와 Worker로 구현되어 있습니다)

2. sendjay에서도 일반적인 Multi Chat Room, isOnline, 파일전송, 파일 만기관리, 초대, 강퇴, 방이름변경, Copy, Reply, Revoke 등 PC용 메신저가 가진 대부분의 기능들이 구현되어 있습니다.

3. 그 외에도 mp4 스트리밍, 안읽은 멤버 조회, 특정멤버 톡만 보기, 목록에서 바로 읽음 처리, 분실신고시 사용중지처리, 사내 조직도(Tree) 등 사내 메신저에 필요한 기능도 추가로 구현되어 있습니다.

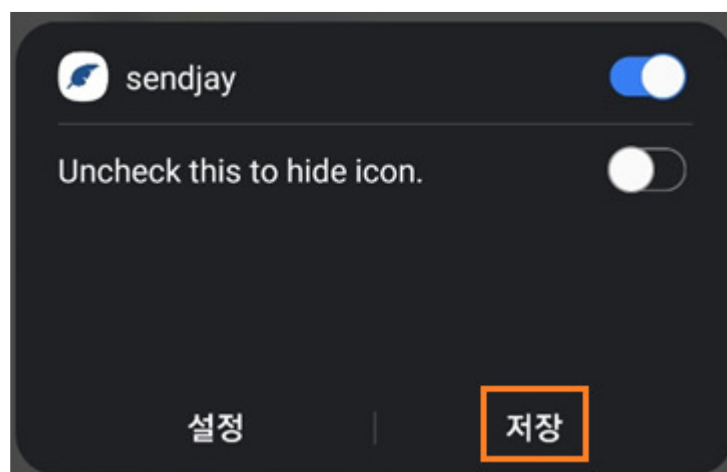
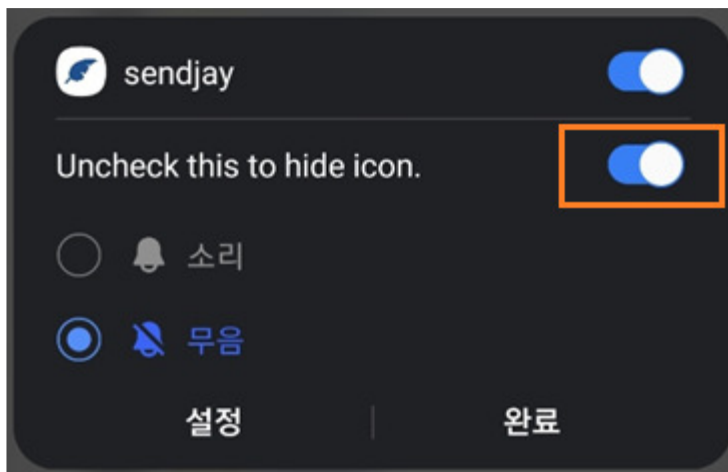
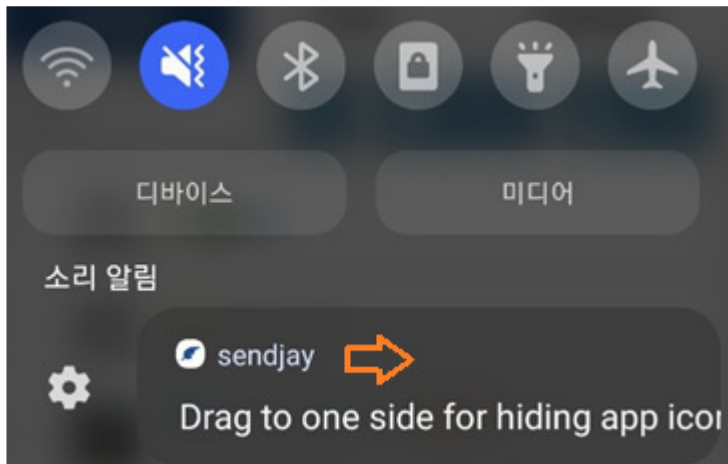
6. Andriod socket.io, Foreground Service, FCM Push and Battery Optimization

* Mobile 환경에서는 PC와는 달리 네트워크(5G, LTE, Wifi 등) 재연결이 자주 반복, 발생하게 되고 디바이스가 리부팅되어도 서비스가 바로 실행되어야 하며 스크린 잠금후 Idle Time이나 Doze mode에서도 네트워크가 끊어지는 경우가 생기게 되는데 이를 고려하면서 App 개발을 진행하게 됩니다.

* Android에서는 **socket.io + Foreground Service (Daemon Thread) + Worker (Periodic) + FCM Push + Battery Optimization**으로 sendjay App의 기본적인 구성이 이루어져 있습니다.

1. 사실, sendjay는 In-House App이기 때문에 Android에서 권장하는 Battery Optimization을 해제하고 FCM을 사용하지 않고 소켓서버와의 연결을 최대한 유지하면서 App을 운영해도 재연결시 그 전까지 전송된 토크를 읽어 알려 주면 되긴 합니다. 하지만, 이 경우엔 배터리 소모율이 크기 때문에 이슈가 됩니다.
2. sendjay에서는 (카카오톡의 배터리 소모율에 근접하기 위해서도) 기본적으로 Battery Optimization 모드를 채택하므로 Idle Time이나 Doze Mode에서도 메시지를 받을 수 있도록 FCM Push를 사용하며 Foreground Service가 사용자에게 의해 강제 종료되는 경우에도 스스로 재시작하며 혹시라도 비정상적인 오류로 인해 종료된 경우에도 Worker가 주기적으로 실행되면서 Service가 중지된 상태이면 재시작하도록 합니다.
3. 또한, 서버에서 메시지를 전송할 때 소켓서버에 접속되어 있으면 소켓으로 보내고 그렇지 않으면 FCM으로 전송합니다. 사실, FCM의 소요비용이 없거나 크게 문제가 되지 않는다면 무조건 FCM으로도 보내는 것이 좋을 것입니다.
4. 그런데, Mobile 네트워크가 모두 끊어진 상태에서는 디바이스에서 FCM도 받을 수 없으며 네트워크가 다시 연결되어야 FCM이 도착하는데 어쨌든 FCM의 Delivery Rate가 100%가 아니고 Delay도 발생하므로 별도의 추가 코딩으로 소켓 재접속전까지 추가로 전송받은 메시지에 대해 Notify하는 로직을 Socket.EVENT_CONNECT에 구현해 놓았습니다.
5. Foreground Service에서는 Daemon Thread가 돌면서 '스크린온 상태에서 소켓접속이 안되어 있으면 바로 재접속'하여 사용중인 상태에서는 최대한 소켓접속을 유지하려 합니다.

* Foreground Service를 구현하면 시작시 실행 Notification이 수반되어야 하는데 이를 프로그래밍에서 제거하는 Nice한 방법을 찾지 못했습니다. (예: Music Player시 Foreground Service용 Notification Icon 표시됨)
따라서, Notification을 보이지 않게 하기 위해서는 최초 실행시 사용자의 설정이 한번 있어야 하는데 아래와 같습니다.



7. 소켓 접속 시작점

* PC Web

1. erp_portal.html – await initMain(“auto”)에서 시작
 - erp_portal.html을 각 회사의 포털이라 가정하고 해당 포털페이지(탭)에 Embedded되어 실행되는 방식임
2. index.html – await initMain(param.launch, param.winid)에서 시작
 - erp_portal.html에서 수동으로 버튼을 누르면 Standalone으로 별도 브라우저탭으로 실행되는 방식
 - index.html의 Settings 탭에서 옵션으로 Standalone 선택시 실행되는 방식

* Android

1. MainActivity.kt
 - onCreate(), start()에서 시작하고 ForeGround Service (ChatService.kt) 호출
2. ChatService.kt
 - onCreate()에서 소켓서버에 접속하고 쓰레드데몬을 시작
 - procSocketEmit()에서 각 업무로직 등에서 요청한 소켓통신 내용을 서버로 전송
 - procSocketOn()에서 서버로부터 전송된 소켓통신내용을 각 업무 로직으로 전달

* Server : app.js - global.jay.on('connection', async (socket) => { })

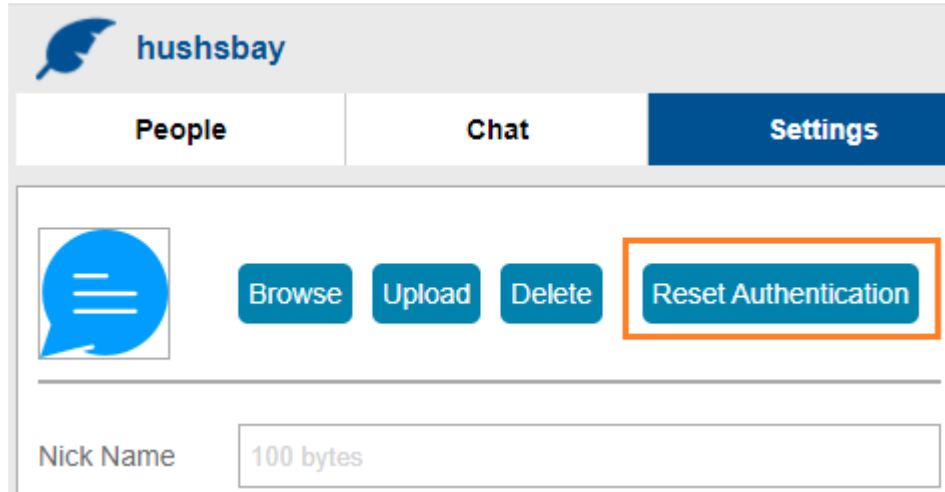
8. Multi Socket Server with Redis

- * sendjay는 Redis를 이용한 Multi Socket Server를 Window Server 1대로만으로 포트를 나누어 Web과 Mobile로 단순 분리한 것입니다.
- * Redis를 위한 npm으로는 socket.io-redis와 ioredis를 이용하며 효율적인 소켓접속정보를 관리하기 위해 connect시 com.multiSetForUserkeySocket()을 이용하여 Redis에 소켓정보를 저장하고 disconnect시 com.multiDelForUserkeySocket()로 소켓정보를 제거하며 필요시 ioredis 모듈이 제공하는 scanStream() API로 해당 정보를 찾아 처리합니다.
 - \$\$S + User_Key + SocketID (\$\$는 향후 만일의 확장을 대비하기 위해 사용하는 단순 Prefix임)
- * socket.io-redis를 사용하므로, 예를 들어, room join/leave가 아닌 remoteJoin/remoteLeave를 사용하는 com.joinRoomWithUserkeySocketArr()/com.leaveRoomWithUserkeySocketArr()로 처리합니다. (remoteDisconnect는 현재 미사용)
- * 또한, com.pub()을 사용하여 다른 소켓서버에 있는 소켓정보를 읽어서 처리하고 있습니다.
예를 들어, Web이나 Mobile에서 사용자가 직접 수동으로 로그인하여 실행하는 경우 com.pub()을 이용하여 다른 소켓서버에 있는 동일 사용자의 키(User_Key)를 찾아서 강제로 연결을 끊습니다.
- * Redis에 저장하는 또 하나의 정보는 PC Web 메신저를 (자동/수동 and Emebded/Standalone) 실행하는 방식과 관련됩니다.
 - \$\$W + User_Key + UniqueID
- * 현재로서는 \$\$S와 \$\$W만 사용중이고 (\$\$US는 현재 Reserved), 톡 읽음처리 등 추가적으로 Redis를 이용해 관리할 대상을 고민했으나 향후 다시 검토하는 것으로 마무리하였습니다.
 - 현재는 소켓서버1을 리스타트하면 리셋을 위해 Redis db0에 저장된 모든 데이터를 삭제(flush db0)하여 초기화합니다.

9. 주요 기능 및 Business Logic

토큰 인증 (자동 로그인)

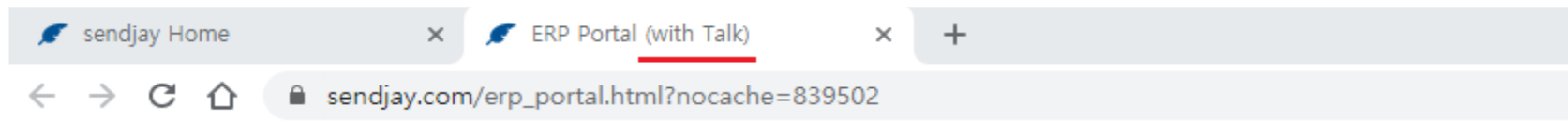
1. sendjay 인증은 세션이 아닌 토큰 방식으로 처리하고 있습니다. (jwt - jsonwebtoken npm 모듈 사용)
2. 토큰은 만기설정이 중요한데, sendjay에서는 jwt의 만기를 10년(36500days)으로 길게 한 이유는 모바일에서 만기가 짧게 도래시어차피 (사용자 불편없이) 연장해야 하기 때문에 아예 길게 설정한 것입니다. (모바일은 특히 자동로그인이 필요)
3. 대신, 토큰 탈취, 기기 분실 등에 대비해 Web 메시저에 Reset Authentication 버튼을 누르면 Web/Mobile 모두 개인별로 관리되는 passkey라는 키값을 변경해 jwt를 invalidate시키며 자동로그인이 해제됩니다. (참고로, 로그아웃은 해당 디바이스에서만 처리됨)



4. 참고 : app.js, com.makeToken(), com.verifyToken()

자동 실행 및 디바이스별 실행

1. sendjay는 Web/Mobile별로 실행가능합니다. 예를 들어, Web에서는 모든 브라우저에서 단 하나의 Web 메신저가 자동실행됩니다. 만일, 크롬 브라우저에서 Web 메신저가 실행중인데 엣지 브라우저에서 동일 사용자로 (수동)실행하면 기존의 크롬에서의 메신저는 강제종료됩니다. 이 때 Mobile에서 실행중인 App이 있다면 그대로 유지되지만 다른 모바일기기에서 실행된다면 기존 모바일기기에서 역시 강제종료됩니다. (모바일은 로그아웃까지 처리됨)
2. Mobile에서는 기본적으로 자동로그인되며 Web에서는 기업 ERP나 Groupware에 로그인한 후 해당 포털에서 자동 실행됩니다. 특히, PC Web에서는 자동실행시 기본적으로 해당 포털페이지에 임베디드됩니다. (포털 종속적) 임베디드시 포털페이지 백그라운드에서 실행되며 도착하는 알림만 표시합니다. 이는 PC용 메신저가 시스템 트레이에 들어가는 것과 유사한 것으로 아래와 같이 표시됩니다. (Standalone 실행옵션도 별도로 지원)



For test, now you can regard this page as your company's portal. (Like ERP or Groupware..)

Userid : **hushsbay**

Tab ID : 20210428173644103 (winid) **Embeded messenger started at this window tab**

Status : Messenger running from this tab / competeWinner: checking as winner: 3

Launch messenger

Add another portal page

Logout

3. Web 메신저 자동실행 로직은 다음과 같습니다.

- 1) 동일 웹 브라우저내에서 상기 그림과 같은 ERP Portal 탭이 여러 개 열려 있는 경우는 여러 탭 각각 서버와 통신하지 않고 로컬 HTML5 indexDB와 Worker를 이용해 탭끼리 경합합니다.
- 2) 경합해서 선점한 탭 한 개만이 서버통신으로 다른 웹브라우저에서 먼저 실행중인 것이 있는지 체크하여 없으면 메신저를 실행(socket 접속)합니다. (자동실행이 아닌 수동실행의 경우는 기존 접속을 강제로 끊고 메신저를 종료시킴)

4. 참고 : erp_portl.html내 jay_main.js, index.html내 jay_main.js, jay_worker.js, chk_redis.js, app.js내 'disconnect_prev_sock' / cut_mobile.js / ChatService.kt내 cut_mobile

데이터 접근 권한

1. sendjay는 어느 정도 웹을 잘 안다면 웹 주소창이나 외부 툴을 이용해 데이터를 가공하여 요청할 수 있을 것입니다. 메신저에서는 대부분 사용자 본인 데이터에 접근하는 경우가 많은데 예를 들어, 서버에서 사용자 본인이 아닌 다른 사용자의 데이터를 요청한다면 막아야 합니다.
2. 조직도(트리) 조회 등 누구에게나 오픈된 자료를 제외하고 아래와 같은 함수로 서버에서 데이터 접근 권한을 처리합니다.
3. 또한, 개인별로 Role을 부여하여 서버 및 클라이언트에서 편리하게 체크할 수 있도록 함수를 제공합니다.
4. com.verifyWithSocketUserId(), com.verifyWithRestUserId(), com.chkAccessUserWithTarget(), com.getRole(), com.chkRole()

캐싱 관리

sendjay project에서는 static 파일(html, js, css, image)들이 그리 많지 않으며 다음과 같은 방식으로 캐싱을 관리합니다.
(캐싱 제거할 필요 없는 plugin js 모듈, image 등은 제외)

캐싱 관리 및 제거를 위한 여러가지 방식이 있겠지만

1. Web에서는 간단히 location.search를 versioning하는 방법을 사용합니다. (?nocache=12345)

1) html : redirec.js 및 jay_common.js내 removeCache() 참조

- /appdown.html과 같이 브라우저에서 요청하면 redirect.js를 통해 /appdown?nocache=34567로 redirect처리합니다.
- 웹브라우저에서 즐겨찾기 등으로 한번 사용한 /appdown?nocache=34567인 경우 /appdown?nocache=67890으로 redirect됩니다.
- /jay/chat.html은 아예 /jay/chat? 으로 프로그램에서 호출됩니다.

2) \$.ajax 호출시 cache option을 false로 설정하였습니다.

3) 브라우저 공통 모듈인 .js는 아래와 같이 캐싱을 제거하는 \$.getScript를 사용하였습니다. (dynamic import)

예) await \$.getScript("/common/jay_common.js")

4) css는 현재로선 각 파일마다 <header>에 선언되어 있어 수정후엔 (그리 많지는 않지만) 캐싱 제거에 번거로울 수 있습니다.

2. Mobile Webview에서는 기본적으로 `webview.settings.cacheMode = WebSettings.LOAD_CACHE_ELSE_NETWORK`으로 설정합니다.


1) 참고로, HttpFuel.kt에서도 `get` 요청시마다 `?nocache=34567` 형식으로 versioning 처리하였습니다.

2) 코딩 수정후 필요시마다 `applist.json`에 각 웹뷰 버전을 업데이트하여 캐시를 제거할 수 있도록 구현하였습니다.


```
{ } applist.json X
sendjay > public > { } applist.json > ...
1  {
2      "sendjay_and" : { "version" : "1.0", "filename" : "sendjay.apk", "path" : "/app/" },
3      "sendjay_ios" : { "version" : "1.0", "filename" : "sendjay.plist", "path" : "/app/" },
4      "webview_main_version" : "1.0",
5      "webview_chat_version" : "1.0",
6      "webview_popup_version" : "1.0",
7      "etc" : { "screenoff" : 10000, "screenon" : 3000 },
8      "code" : "0",
9      "msg" : ""
10 }
```


`webview_main_version` (index.html), `webview_chat_version` (chat.html), `webview_popup_version` (popup.html)


메인 화면 (People탭)


 hushsbay [whowho🙄]


People





 Team

 Tree









 COMPANY_000000

 HQ_000001 (1)


☐





  user000 [Hi~]


 DEPT_000002 (1)

☐






  user001 [Hi~]
job not described yet

☐


 TEAM_000003 (8)



☐




  ay




☐




  hmson



☐



  hushsbay  everyday
programmer

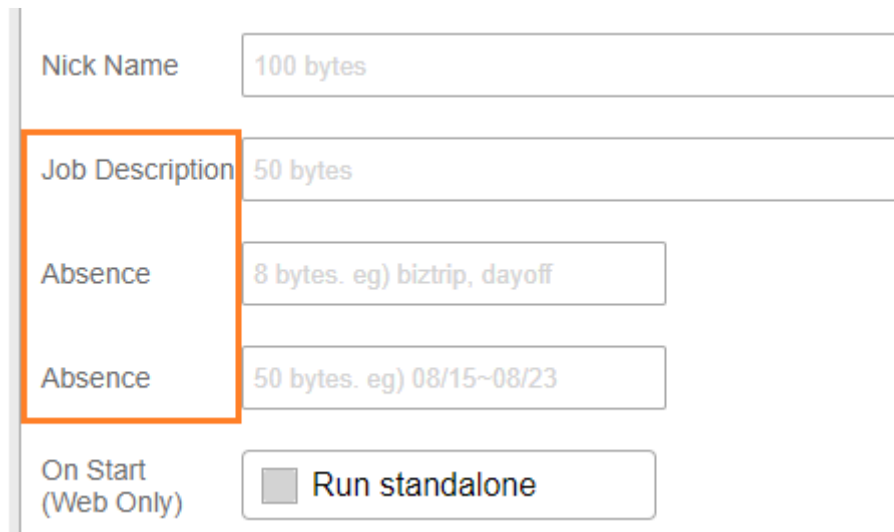
☐



  jspark

- * 부서 등 조직 트리는 위 그림처럼 메인화면 및 사용자등록시 부서 선택에서 사용되는데 sendjay에서는 admin.js에서 회사, 본부, 부서 등 조직 및 인사 정보를 (테스트 목적으로) 생성할 수 있도록 지원합니다.
- * 실제로, 각 회사는 개발팀에서 사내메신저 전용 조직/인사 테이블과 연동하든 아니면 기존 테이블에 직접 붙여서 사용하는 방식을 채택하여 개발을 진행하게 될 것입니다.

메인 화면 (Settings탭)



Nick Name 100 bytes

Job Description 50 bytes

Absence 8 bytes. eg) biztrip, dayoff

Absence 50 bytes. eg) 08/15~08/23

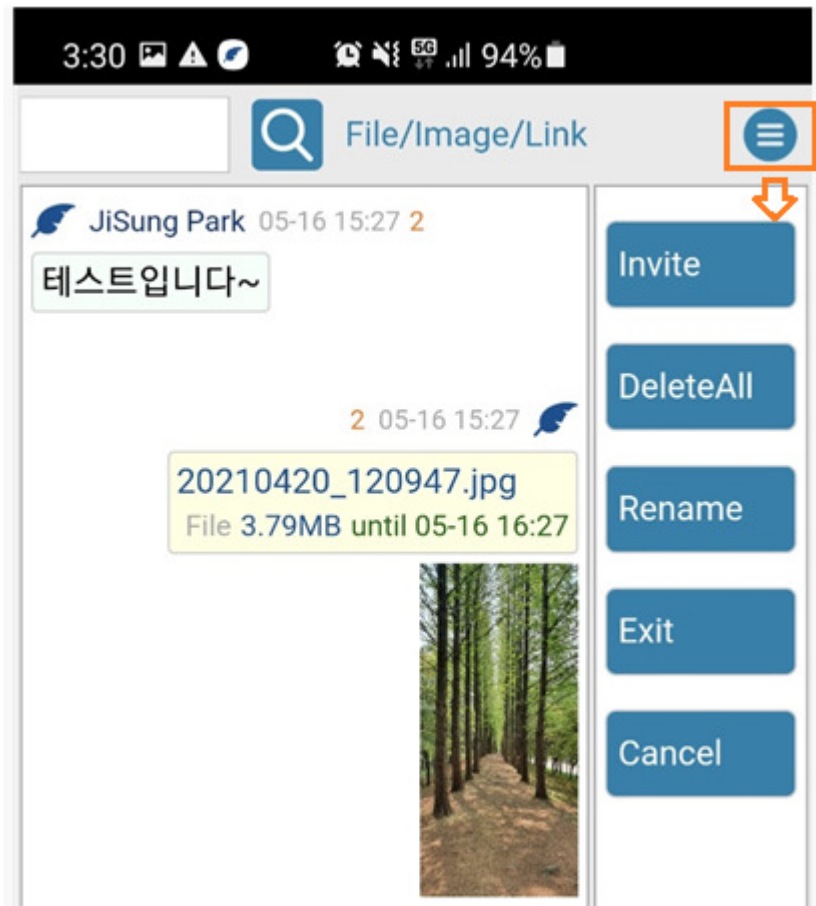
On Start (Web Only) ☐ Run standalone

- * sendjay는 기본적으로 Job(담당업무), 부재 등을 사용자입력으로 처리하고 있으나 실제 사내메신저는 이러한 정보를 내부 데이터와 연동해 Read Only로 표시가능합니다. 예를 들어, 휴가/출장/재택근무 등을 위 부재필드에 연동하며 인사시스템의 직무정보를 Job 필드에 연동 가능합니다. 여기에는 없지만 전화번호도 연동해 표시 가능할 것입니다.

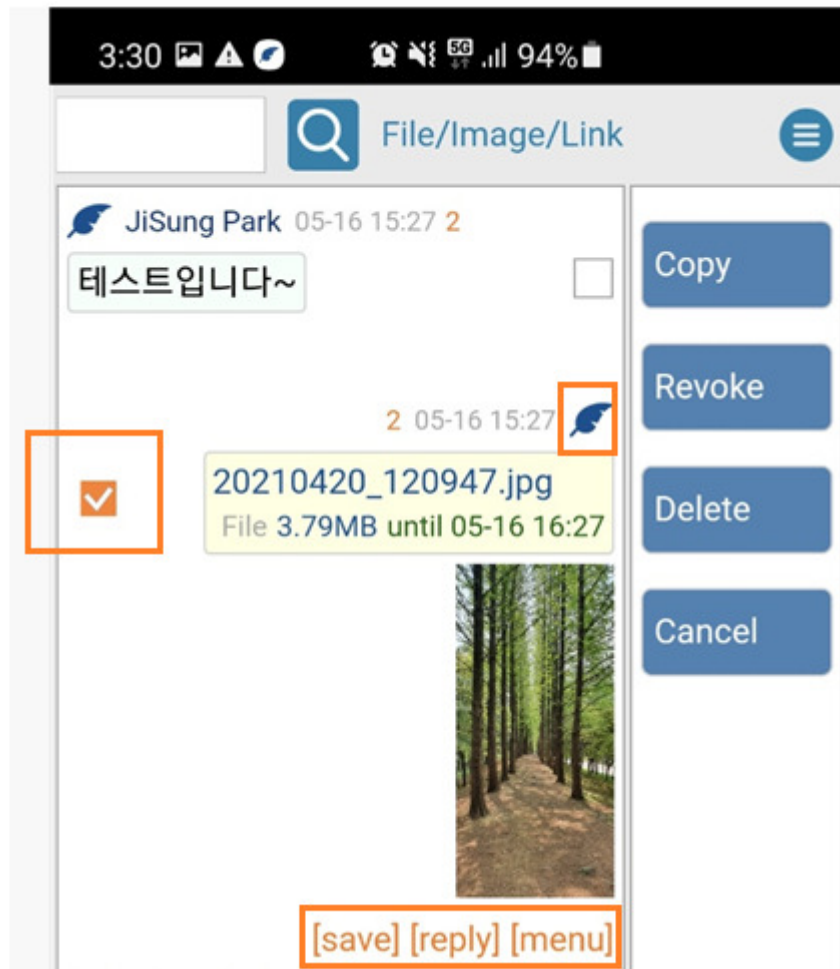
채팅방



- * Mobile 채팅방내 멤버보기는 아이콘을 눌러 토글 가능합니다.
- * 메신저는 본인만 설치한다고 되는 것이 아닙니다.
특히, 사내메신저는 모든 임직원분들이 접속 가능해야 메시지를 주고 받을 수 있습니다.
- * 따라서, 최초 시행시 Web 메신저인 경우는 기업내 포털페이지에 임베디드되므로 자동(강제)실행되고 모든 임직원이 연결되도록 해야 문제가 없을 것이며, Mobile인 경우는 사용자가 설치했는지 On-Line 상태인지를 표시하여 상대방이 톡을 보내야 할지 말지 판단하도록 도와줘야 합니다.
- * Android App을 설치하고 로그인하면 FCM 토큰이 DB에 등록되며 퇴직하거나 톡 전송시 (Invalid Token) 오류가 나지 않는 한 정상적인 상태이므로 초록색 M으로 App 설치여부를 표시합니다.
- * PC Web에서는 기업 사이트에서 로그인하면 접속(W)으로 표시되고 로그아웃되면 같이 종료되고 그 때 같이 접속이 끊깁니다.



* PC Web 메신저에서는 하단에 있던 초대, 모두삭제 등의 버튼들이 Mobile에서는 위 아이콘으로 토글 가능합니다.



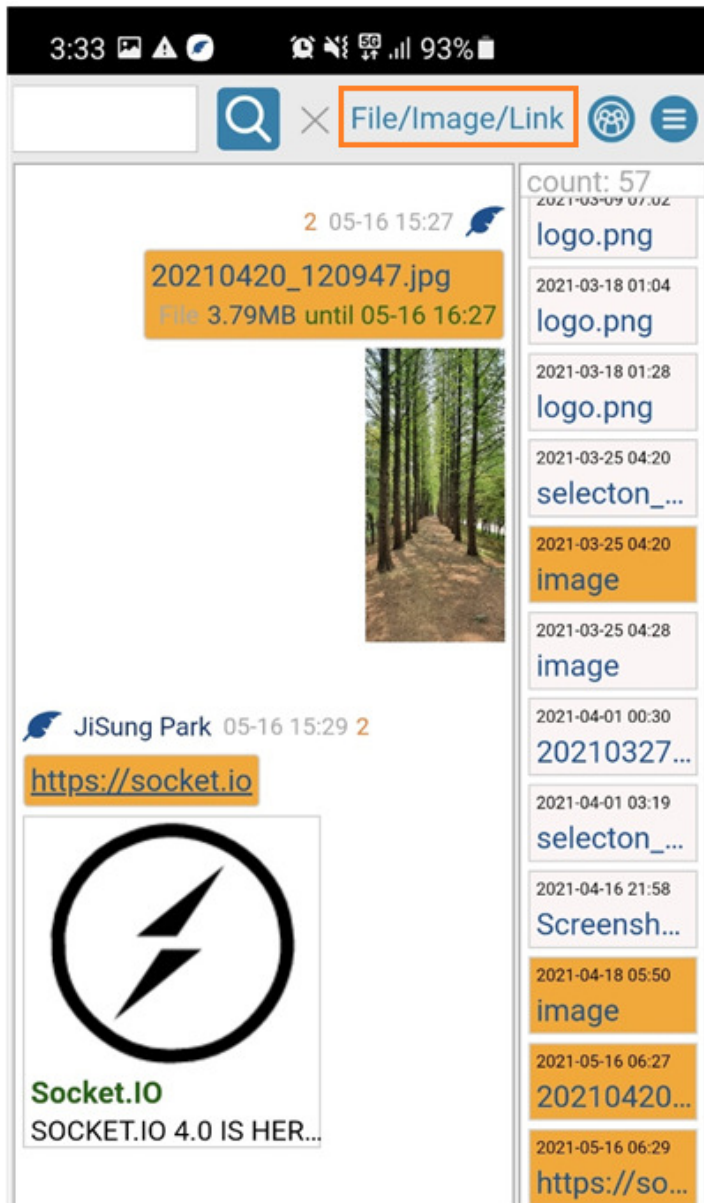
* 카카오톡에서 taphold(꾸욱누르기) 이벤트로 처리되었던
톡 셀 단위의 처리는 해당 셀 가볍게 터치시 하단
[menu]로 탭하거나 sendjay 기본 아이콘(깃털)을 누르면
우측에 메뉴가 나타납니다.

* [menu]뿐 아니고 자주 사용하게 되는 file download를
처리하는 [save]와 멤버를 지정하여 답장하는 [reply]는
같이 보이게 처리하였습니다

* 여기에 특히 복사(Copy) 버튼은 텍스트, 이미지 뿐만 아니라
파일도 다운로드하지 않고 바로 복사하여 다른 톡방을 열어
붙여 넣고 전송이 가능합니다. 특히, mp4 동영상 등
GB가 넘는 파일을 다운로드하지 않고 바로 복사/붙이기로
전송이 가능합니다.



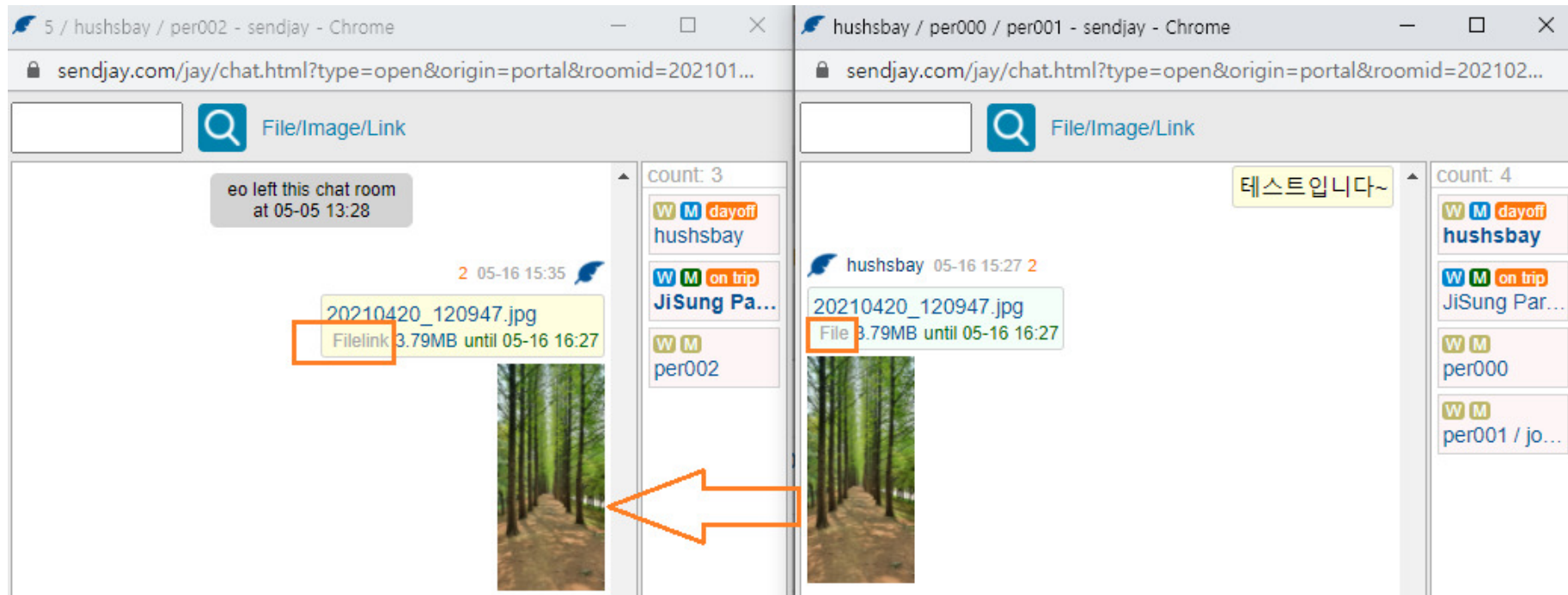
- * 멤버를 탭하면 해당 멤버의 특만 별도로 조회 가능합니다.
- * 강퇴(MakeLeave)를 누르면 특방에서 강제로 내보낼 수도 있는데 현재는 퇴직일 때만 가능하도록 해 놓았습니다.
(퇴직은 db table z_user_tbl의 del_date 필드값 참조)



* 그림과 유사한 UI는 3가지가 있습니다.

- 1) File/Image/Link 조회
- 2) 검색 결과
- 3) 특정멤버 특만 조회

* 그림의 오른쪽 count: 57 아래 결과를 터치하면 해당 셀로 이동해서 보여주면서 전후의 문맥을 알려 주기 위해 그 사이의 특을 모두 표시합니다.



* File을 위 복사 버튼과 동일하게 PC Web에서 다른 톡방으로 Drag & Drop하면 Filelink라는 타입으로 복사되어 링크만 전송하고 그 링크를 베이스로 파일정보를 표시합니다.

* 이상은 카카오톡과는 전혀 다른 기능이나 UI/UX인 경우만 말씀드린 것이며 일반적인 톡과 관련된 설명은 굳이 하지 않았음을 양지하여 주시기 바랍니다.

특방 열린 직후 이미지 로딩중에 스크롤 올리는 경우 자동스크롤 방지

- * 특방이 처음 열릴 때 이미지가 없다고 가정하면 특을 모두 가져와서 표시한 후 맨 아래로 스크롤링 처리하면 됩니다.
- * 그런데, 이 때 단순 이미지, 이미지파일용 미리보기, mp4 이미지, OpenGraph 이미지는 비동기로 가져옵니다.
- * 그러다보니, 사용자가 모든 이미지 내리기 전에 스크롤하여 올라가는 경우엔 이미지가 로드완료되고 난 후 맨 아래로 스크롤링 처리하면 사용자 불편이 발생합니다. 이를 방지하기 위해 (사용자만이 가능한) 위로 스크롤시 `g_stopAutoScrollDown=true`로 주고 이미지 로드 완료까지 자동스크롤 안되게 합니다.

전송오류, 무응답, 재전송 – Mobile을 중심으로

- * 사용자가 톡을 전송시 서버로 전송되지 못하거나 무응답되는 경우가 있습니다.
- * 카카오톡의 경우 모바일 네트워크 연결이 해제되었을 때 톡을 계속 보내면 각각 '계속 전송시도중'이라는 느낌을 주는 종이비행기 아이콘을 표시하고 타임아웃이든 App종료 또는 톡방을 다시 들어오면 재전송할 건지 물어봅니다.
- * sendjay도 이 부분을 Webview내에서 유사하게 구현했습니다.

1. 전송오류 (sending failure)

- 1) 서버로 전송요청이 이미 전달되어 서버에서 업무적인 사안, 논리적인 오류 등으로 클라이언트에게 전달되는 메시지입니다.
- 2) 예를 들어, 특정 요청에 대해 권한이 없는 사용자가 요청시 서버에서 체크하여 sending failure를 반환하는데 이 경우 클릭시 재전송이 아닌 원인 파악과 삭제만 가능하게 됩니다.

2. 무응답 (no response)

- 1) 말 그대로 톡 전송 직후 서버의 응답이 없어 톡이 전송되었는지 여부를 알 수 없을 때 전달되는 메시지입니다.
- 2) 무응답의 경우 크게 보면 네트워크 연결이 끊어졌을 때, 서버가 다운되었을 때, 앱이 (강제)종료되었을 때일 것입니다.
 - 예를 들어 Android App에서는 socket.io로 전송시 volatile(휘발성)로 보내는 경우가 없어서 그런지 소켓전송 결과의 무응답이 네트워크 해제 및 서버 리스타트의 경우는 App socket.io-client가 가지고 있다가 재연결시 다른 소켓통신이 나갈 때 함께 전송되는 것으로 확인하였습니다.
 - 다만, App 종료후 재시작시는 재연결되더라도 자동 재전송되지 않는 것을 확인하였습니다.

3. 재전송 (retry sending)

- 1) 원인이야 어쨌든, 특히 모바일 사용자 입장에서는, 입력한 톡을 전송했는데 전송이 안되고 톡조차 사라져 버리면 무척 당황스러울 것이므로 전송되지 않은 톡은 다시 전송되게 하는 UX/UI가 필요할 것입니다.
- 2) sendjay에서는 App이 아닌 Web 구성요소인 HTML5 IndexedDB로 재전송을 구현했습니다.
 - 일단 톡이 전송될 때 로컬DB에 같이 저장합니다.
 - 서버에서 해당 톡이 전송된 결과가 정상적으로 오면 로컬DB에 해당 톡은 삭제합니다.
 - 그렇지 않으면 톡에 무응답(hush.cons.no_response) 표시합니다.
 - 톡방을 나가지 않고 다시 hush.cons.no_response에 탭하면 무응답을 다시 표시하거나 정상 전송된 결과를 보여줍니다.
 - 톡방을 나갔다가 다시 들어온 경우엔 로컬DB에 있는 메시지ID를 서버로 보내서 아직 도착하지 않은 ID만 받아서 재전송(retry sending)할 것인지 삭제할 것인지 물어봅니다.
 - 로컬DB에도 가비지가 쌓이지 않게 이미 전송된 톡으로 확인되면 제거합니다. (톡방 열 때)
 - 이미지 및 파일은 사용자가 디바이스에서 선택해 전송하는 것이므로 재전송처리 대상에서 제외시켰습니다.

읽음처리 (Make Unread)

* 톡방에서의 읽음 처리(조회)는 다음과 같은 경우에 실행되는데 Webview안에 대부분의 로직이 있습니다.

1. updateAllUnreads() : 무조건 톡방내 읽지 않은 톡 모두를 읽음으로 처리

1) 맨 처음 톡방을 열 때 (first_queried 값 체크)

2) 열린 톡방 다시 포커싱 될 때 (Android에서는 onResume시)

3) 톡방에서 스크롤이 맨 아래 있지 않고 중간에 있을 때 (사용자가 중간 부분을 읽고 있을 때) 톡이 오면

자동으로 맨 아래로 내려가 톡이 온 것을 보여 주면 사용자 입장에서는 무척 불편한데 이 경우는 톡이 왔다고

알려 주기만 합니다 (_isStickyNeeded 값 참조) 이 알림을 클릭시 맨 아래로 스크롤이 가면서 updateAllUnreads()을 호출합니다.

2. procQueryUnread() : 화면에서 눈에 보이는 (아직 안읽은 사용자가 있는) 톡에 대해서만 Query

1) updateAllUnreads() 후

2) 스크롤이 멈추는 경우

3) 톡방이 열린 상태에서 소켓통신 재연결시

3. 단순 Update/Query : 한개의 톡에 대해서만 처리

1) 일반적으로 열린 톡방에서 톡 받았을 때

Notification

- * 크롬 웹브라우저 계열에서는 window.Notification 객체를 이용하는데 최근에는 윈도우10의 알림센터와 연동되어 표시되며 일정시간이 지나면 시스템트레이 바로 위(화면 우측 하단)에서의 알림이 계속 유지되지 않고 알림센터내로 들어가 버려 알림이 잘 보이지 않는 이슈가 많습니다. (네이버 웨일 브라우저는 계속 유지를 지원하고 있으며 엣지와 크롬도 곧 지원할 것으로 기대합니다)
- * 아래에서는 일반적인 로직의 Notification (이하 noti)은 배제하고 Web/Mobile 동시 사용의 경우에 대해 설명합니다.
 1. Web과 Mobile이 모두 접속되어 있고 해당 톡방이 열려 있지 않으면 Web에서 먼저 알려 주고 몇초후에도 읽음처리되어 있지 않으면 Mobile에서도 noti 됩니다.
 2. Web이든 Mobile이든 한 쪽 톡방이 열려 있는 상태에서 톡이 오면 noti는 없습니다.
 3. noti가 떠 있는 상태에서 Web이든 Mobile이든 한쪽이 열리면 다른 쪽의 noti는 제거됩니다.
- * 특히, Mobile의 경우 소켓통신 재연결시 qry_unread.js를 호출하여 연결이 끊어진 동안 새로운 톡 도착정보가 있으면 보여줍니다.

데몬

sendjay에서 현재 주기적으로 실행되고 있는 데몬은 아래와 같습니다.

* daemon01.js : 저장만기된 파일 및 빈 폴더 삭제, 파일전송중 웹브라우저 닫기 등으로 발생한 가비지 제거

* worker01.js : 미등록(등록취소 or 퇴직)된 사용자에 대한 소켓 및 Redis 데이터 삭제

* worker02.js : 간단한 소켓 접속 현황 제공

98. MySQL 테이블 명세

* sendjay에서는 개발/운영서버를 별도의 DB 인스턴스가 아닌 스키마로 구분하여 관리함 (AWS 1 EC2, 1 RDS)

- EC2의 nodedev 폴더 -> jaydev 스키마 (개발서버), nodeops 폴더 -> jayops 스키마 (운영서버)로 관리함

- **sendjay의 모든 데이터는 테스트 목적이므로 별도의 공지없이 삭제될 수 있음을 알려 드립니다.**

* 테이블 명세

```
-- drop table if exists jayops.Z_ORG_TBL;
```

```
CREATE TABLE jayops.Z_ORG_TBL (
```

```
ORG_CD VARCHAR(20) NOT NULL COMMENT 'Organization Code',
```

```
ORG_NM VARCHAR(50) NOT NULL COMMENT 'Organization Name',
```

```
SEQ VARCHAR(8) NOT NULL COMMENT 'Organization Seq from 000001',
```

```
LVL INTEGER NOT NULL COMMENT 'Organization Level',
```

```
ISUR VARCHAR(20) NOT NULL DEFAULT '' COMMENT 'creator',
```

```
ISUDT CHAR(26) NOT NULL DEFAULT '' COMMENT 'created datetime',
```

```
MODR VARCHAR(20) NOT NULL DEFAULT '' COMMENT 'modifier',
```

```
MODDT CHAR(26) NOT NULL DEFAULT '' COMMENT 'modified datetime' );
```

```
CREATE UNIQUE INDEX Z_ORG_IDX0 ON jayops.Z_ORG_TBL (ORG_CD) ;
```

```
CREATE INDEX Z_ORG_IDX1 ON jayops.Z_ORG_TBL (SEQ) ;
```

```
-- drop table if exists jayops.Z_USER_TBL;
CREATE TABLE jayops.Z_USER_TBL (
USER_ID VARCHAR(20) NOT NULL COMMENT "",
PWD VARCHAR(128) NOT NULL COMMENT "",
PASSKEY VARCHAR(6) NOT NULL DEFAULT "" COMMENT 'Manual login needed when passkey changed',
USER_NM VARCHAR(50) NOT NULL COMMENT "",
ORG_CD VARCHAR(20) NOT NULL DEFAULT "" COMMENT 'Organization Code',
ORG_NM VARCHAR(50) NOT NULL DEFAULT "" COMMENT 'Organization Name',
TOP_ORG_CD VARCHAR(20) NOT NULL DEFAULT "" COMMENT 'Top Organization Code',
TOP_ORG_NM VARCHAR(50) NOT NULL DEFAULT "" COMMENT 'Top Organization Name',
PICTURE longblob NULL,
MIMETYPE VARCHAR(25) NOT NULL DEFAULT "" COMMENT 'mimetype for picture',
NICK_NM VARCHAR(100) NOT NULL DEFAULT "" COMMENT 'alias or user's state',
JOB VARCHAR(50) NOT NULL DEFAULT "" COMMENT "",
TEL_NO VARCHAR(20) NOT NULL DEFAULT "" COMMENT 'phone number',
AB_CD VARCHAR(7) NOT NULL DEFAULT "" COMMENT 'Absense Code eg) dayoff, biztrip',
AB_NM VARCHAR(50) NOT NULL DEFAULT "" COMMENT 'Absense Detail eg) 20210101~20210103',
STANDALONE CHAR(1) NOT NULL DEFAULT "" COMMENT 'Y if true (for web only)',
NOTI_OFF CHAR(1) NOT NULL DEFAULT "" COMMENT 'Y if true and turn notification off (Notification Setting)',
SOUND_OFF CHAR(1) NOT NULL DEFAULT "" COMMENT 'Y if true and turn noti sound off (Notification Setting)',
TM_FR VARCHAR(4) NOT NULL DEFAULT "" COMMENT 'Noti time (from)',
TM_TO VARCHAR(4) NOT NULL DEFAULT "" COMMENT 'Noti time (to)',
BODY_OFF CHAR(1) NOT NULL DEFAULT "" COMMENT 'Y if true and hide body (Notification Setting)',
```

```
SENDER_OFF CHAR(1) NOT NULL DEFAULT " COMMENT 'Y if true and hide sender (Notification Setting)',
LASTCHKDT CHAR(26) NOT NULL DEFAULT " COMMENT 'last disconnected dt ',
ROLE VARCHAR(30) NOT NULL DEFAULT " COMMENT 'eg) admin (rolename as you want)',
UID VARCHAR(30) NOT NULL DEFAULT " COMMENT 'IP on web browser, UUID on mobile',
OS_INUSE CHAR(3) NOT NULL DEFAULT " COMMENT 'ios or and when login (mobile)',
PUSH_IOS VARCHAR(256) NOT NULL DEFAULT " COMMENT 'apns token',
PUSH_AND VARCHAR(256) NOT NULL DEFAULT " COMMENT 'fcm token – 163bytes',
DEL_DATE VARCHAR(26) NOT NULL DEFAULT " COMMENT 'unregistered datetime',
ISUR VARCHAR(20) NOT NULL DEFAULT " COMMENT 'creator',
ISUDT CHAR(26) NOT NULL DEFAULT " COMMENT 'created datetime',
MODR VARCHAR(20) NOT NULL DEFAULT " COMMENT 'modifier',
MODDT CHAR(26) NOT NULL DEFAULT " COMMENT 'modified datetime' ) ;
CREATE UNIQUE INDEX Z_USER_IDX0 ON jayops.Z_USER_TBL (USER_ID) ;
CREATE INDEX Z_USER_IDX1 ON jayops.Z_USER_TBL (ORG_CD, USER_NM) ;
CREATE INDEX Z_USER_IDX2 ON jayops.Z_USER_TBL (USER_NM, USER_ID) ;
CREATE INDEX Z_USER_IDX3 ON jayops.Z_USER_TBL (DEL_DATE, USER_ID) ;
```



```
-- drop table if exists jayops.A_ROOMMST_TBL;  
CREATE TABLE jayops.A_ROOMMST_TBL (  
  ROOMID VARCHAR(40) NOT NULL COMMENT "  
  ROOMNM VARCHAR(800) NOT NULL DEFAULT " COMMENT "  
  MASTERID VARCHAR(40) NOT NULL DEFAULT " COMMENT "  
  MASTERNM VARCHAR(50) NOT NULL DEFAULT " COMMENT "  
  MEMCNT SMALLINT UNSIGNED NOT NULL DEFAULT 0 COMMENT 'total count of embers',  
  CDT CHAR(26) NOT NULL DEFAULT " COMMENT 'created datetime',  
  UDT CHAR(26) NOT NULL DEFAULT " COMMENT 'updated datetime',  
  STATE CHAR(1) NOT NULL DEFAULT " COMMENT "  
  NICKNM VARCHAR(100) NOT NULL DEFAULT " COMMENT " );  
CREATE UNIQUE INDEX A_ROOMMST_IDX0 ON jayops.A_ROOMMST_TBL (ROOMID) ;
```

```
-- drop table if exists jayops.A_ROOMDTL_TBL;
CREATE TABLE jayops.A_ROOMDTL_TBL (
ROOMID VARCHAR(40) NOT NULL COMMENT "",
USERID VARCHAR(40) NOT NULL COMMENT "",
USERNM VARCHAR(50) NOT NULL COMMENT "",
NOTI CHAR(1) NOT NULL DEFAULT " COMMENT 'X (no notification)',
DISPMEM CHAR(1) NOT NULL DEFAULT " COMMENT 'X (no member display)',
CDT CHAR(26) NOT NULL DEFAULT " COMMENT 'created datetime',
UDT CHAR(26) NOT NULL DEFAULT " COMMENT 'modified datetime',
STATE CHAR(1) NOT NULL DEFAULT " COMMENT 'L(Leave), 2(LeaveInCaseOf2Members)',
NICKNM VARCHAR(100) NOT NULL DEFAULT " COMMENT " );
CREATE UNIQUE INDEX A_ROOMDTL_IDX0 ON jayops.A_ROOMDTL_TBL (ROOMID, USERID) ;
CREATE INDEX A_ROOMDTL_IDX1 ON jayops.A_ROOMDTL_TBL (ROOMID, STATE) ;
```

```
-- drop table if exists jayops.A_ROOMMEM_TBL;
CREATE TABLE jayops.A_ROOMMEM_TBL (
ROOMID VARCHAR(40) NOT NULL COMMENT "",
MEMBERS VARCHAR(1500) NOT NULL DEFAULT " COMMENT "",
CDT CHAR(26) NOT NULL DEFAULT " COMMENT 'created datetime' );
CREATE UNIQUE INDEX A_ROOMMEM_IDX0 ON jayops.A_ROOMMEM_TBL (ROOMID) ;
CREATE INDEX A_ROOMMEM_IDX1 ON jayops.A_ROOMMEM_TBL (MEMBERS) ;
```

```
-- drop table if exists jayops.A_MSGMST_TBL;
CREATE TABLE jayops.A_MSGMST_TBL (
MSGID VARCHAR(40) NOT NULL COMMENT "",
ROOMID VARCHAR(40) NOT NULL COMMENT "",
SENDERID VARCHAR(40) NOT NULL COMMENT "",
SENDERNM VARCHAR(40) NOT NULL COMMENT "",
BODY VARCHAR(4000) NOT NULL COMMENT "",
BUFFER LONGBLOB,
REPLY VARCHAR(100) NOT NULL DEFAULT "" COMMENT "",
TYPE VARCHAR(20) NOT NULL DEFAULT "" COMMENT 'file, flink, image, invite, leave',
CDT CHAR(26) NOT NULL DEFAULT "" COMMENT 'datetime for sending',
UDT CHAR(26) NOT NULL DEFAULT "" COMMENT 'datetime for update',
STATE CHAR(1) NOT NULL DEFAULT "" COMMENT 'D(deleted)/R(read)',
FILESTATE VARCHAR(19) NOT NULL DEFAULT "" COMMENT 'expiry(YYYY-MM-DD hh:mm:ss) ');
CREATE UNIQUE INDEX A_MSGMST_IDX0 ON jayops.A_MSGMST_TBL (MSGID, ROOMID) ;
CREATE INDEX A_MSGMST_IDX1 ON jayops.A_MSGMST_TBL (ROOMID, TYPE, CDT) ;
CREATE INDEX A_MSGMST_IDX2 ON jayops.A_MSGMST_TBL (ROOMID, CDT, SENDERID) ;
CREATE INDEX A_MSGMST_IDX3 ON jayops.A_MSGMST_TBL (TYPE, FILESTATE) ;
```

```
-- drop table if exists jayops.A_MSGDTL_TBL;
CREATE TABLE jayops.A_MSGDTL_TBL (
MSGID VARCHAR(40) NOT NULL COMMENT "",
ROOMID VARCHAR(40) NOT NULL COMMENT "",
SENDERID VARCHAR(20) NOT NULL COMMENT "",
RECEIVERID VARCHAR(20) NOT NULL COMMENT "",
RECEIVERNM VARCHAR(20) NOT NULL COMMENT "",
CDT CHAR(26) NOT NULL DEFAULT "" COMMENT 'datetime for sending',
UDT CHAR(26) NOT NULL DEFAULT "" COMMENT 'datetime for update',
STATE CHAR(1) NOT NULL DEFAULT "" COMMENT 'D(deleted)/R(read)',
PUSH_ERR VARCHAR(100) NOT NULL DEFAULT "" COMMENT 'push error cd/msg');
CREATE UNIQUE INDEX A_MSGDTL_IDX0 ON jayops.A_MSGDTL_TBL (MSGID, ROOMID, SENDERID, RECEIVERID) ;
CREATE INDEX A_MSGDTL_IDX1 ON jayops.A_MSGDTL_TBL (MSGID, RECEIVERID) ;
CREATE INDEX A_MSGDTL_IDX2 ON jayops.A_MSGDTL_TBL (MSGID, SENDERID) ;
CREATE INDEX A_MSGDTL_IDX3 ON jayops.A_MSGDTL_TBL (ROOMID, RECEIVERID, STATE, CDT) ;
CREATE INDEX A_MSGDTL_IDX4 ON jayops.A_MSGDTL_TBL (RECEIVERID, STATE, CDT) ;
```

```
-- drop table if exists jayops.A_FILELOG_TBL;
CREATE TABLE jayops.A_FILELOG_TBL (
MSGID VARCHAR(40) NOT NULL COMMENT "",
ROOMID VARCHAR(40) NOT NULL COMMENT "",
SENDERID VARCHAR(40) NOT NULL COMMENT "",
BODY VARCHAR(1000) NOT NULL COMMENT "",
CDT CHAR(26) NOT NULL DEFAULT " COMMENT 'datetime for sending',
UDT CHAR(26) NOT NULL DEFAULT " COMMENT 'datetime for update' );
CREATE UNIQUE INDEX A_FILELOG_IDX0 ON jayops.A_FILELOG_TBL (MSGID, ROOMID) ;
CREATE INDEX A_FILELOG_IDX1 ON jayops.A_FILELOG_TBL (UDT, CDT) ;
```

```
-- drop table if exists jayops.Z_LOG_TBL;
-- CREATE TABLE jayops.Z_LOG_TBL – not used for now
-- USER_ID VARCHAR(20) NOT NULL COMMENT "",
-- IP VARCHAR(40) NOT NULL DEFAULT " COMMENT 'Remote IP',
-- PGM_ID VARCHAR(20) NOT NULL DEFAULT " COMMENT 'program(page) ID',
-- CDT CHAR(26) NOT NULL DEFAULT " COMMENT 'created datetime'
-- CREATE INDEX Z_LOG_IDX0 ON jayops.Z_LOG_TBL (CDT) ;
```

* 아래는 데이터 샘플입니다.



1 • SELECT * FROM jayops.Z_USER_TBL

2

3

4



5

Result Grid																				
Filter Rows: <input type="text"/>		Edit:				Export/Import:		Wrap Cell Content: T												
	USER_ID	PWD	PASSKEY	USER_NM	ORG_CD	ORG_NM	TOP_ORG_CD	TOP_ORG_NM	PICTURE	MIMETYPE	NICK_NM	JOB	TEL_NO	AB_CD	AB_NM	STANDALONE	NOTI_OFF	SOUND_OFF	TM_FR	TM_TO
▶	hmson	40da18b6...	782026	hmson	000003	TEAM_000003	000000	COMPANY_000000												
	jspark	61f6fc278...	598628	jspark	000003	TEAM_000003	000000	COMPANY_000000												
	yakim	84002c09...	244404	yakim	000003	TEAM_000003	000000	COMPANY_000000		image/jpeg						Y				
*																				

M_TO	BODY_OFF	SENDER_OFF	LASTCHKDT	ROLE	UID	OS_INUSE	PUSH_IOS	PUSH_AND	DEL_DATE	ISUR	ISUDT	MODR	MODDT
				main	222.108.42.175					hmson	2021-06-18 08:07:47.532548		
			2021-06-27 09:23:21.012793	main	222.108.42.175	and		ex5L0hYLT32hxy63Yq4-X:APA91bGsgKumCO...		jspark	2021-06-18 07:56:26.938271		
				main	222.108.42.175					yakim	2021-06-25 22:38:25.934083	yakim	2021-06-26 00:55:26.711799
1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

1 • SELECT * FROM jayops.Z_ORG_TBL;

Result Grid

  Filter Rows:

	ORG_CD	ORG_NM	SEQ	LVL
	000000	COMPANY_000000	000000	0
	000001	HQ_000001	000001	1
	000002	DEPT_000002	000002	2
	000003	TEAM_000003	000003	3
	000004	TEAM_000004	000004	3
	000005	DEPT_000005	000005	2
	000006	TEAM_000006	000006	3
	000007	TEAM_000007	000007	3
	000008	HQ_000008	000008	1

1 • SELECT * FROM jayops.A_ROOMMST_TBL;

Result Grid										Filter Rows:		Edit:		Export/Import:		Wrap Cell Content:	
	ROOMID	ROOMNM	MASTERID	MASTERNM	MEMCNT	CDT	UDT	STATE	NICKNM								
▶	20210620133248724000359755YLZtG5IIK	{roomnm:"ay / hmson / hushsbay / jspark / user002 / more..",userid:"doob / hm...	hushsbay	hushsbay	7	2021-06-20 04:32:49.347506											
	20210620134441579000353126iYKGILL5tz	{roomnm:"hmson / hushsbay / jspark / user030 / more..",userid:"hmson / hushs...	hushsbay	hushsbay	5	2021-06-20 04:44:41.677693	2021-06-27 08:13:20.390131										
	20210620141536800000356509J1CtBFQoz	{roomnm:"hmson / hushsbay",userid:"hmson / hushsbay"}	hushsbay	hushsbay	2	2021-06-20 05:15:40.515191											
	202106260748247910003854956qct3RPY0	{roomnm:"ay / hmson / hushsbay / user002 / more..",userid:"doob / hmson / hu...	yakim	yakim	5	2021-06-25 22:48:23.782108	2021-06-26 00:50:30.279213		최종테스트01								
	202106261043051340004062896Dub4tuJUn	{roomnm:"ay / yakim",userid:"doob / yakim"}	yakim	yakim	2	2021-06-26 01:43:05.891230											
	20210626104322838000992690JuDtDUB6n4	{roomnm:"ay / user001 / user002 / user003 / more..",userid:"doob / user001 / u...	yakim	yakim	7	2021-06-26 01:43:23.567807	2021-06-26 02:44:26.595856										
	20210627200933825000710887dGweRXe66A	{roomnm:"ay / hushsbay",userid:"doob / hushsbay"}	doob	ay	2	2021-06-27 11:09:34.454396											
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL								

1 • `SELECT * FROM jayops.A_ROOMDTL_TBL;`

Result Grid



Filter Rows:

Edit:



Export/Import:



Wrap Cell Content:



ROOMID	USERID	USERNM	NOTI	DISPMEM	CDT	UDT	STATE	NICKNM
20210620133248724000359755iYLztG5IiK	doob	ay			2021-06-20 04:32:49.342959			
20210620133248724000359755iYLztG5IiK	hmson	hmson			2021-06-20 04:32:49.345612			
20210620133248724000359755iYLztG5IiK	hushsbay	hushsbay			2021-06-20 04:32:49.346000			
20210620133248724000359755iYLztG5IiK	jspark	jspark			2021-06-20 04:32:49.346323			
20210620133248724000359755iYLztG5IiK	user002	user002			2021-06-20 04:32:49.346611			
20210620133248724000359755iYLztG5IiK	user003	user003			2021-06-20 04:32:49.346884			
20210620133248724000359755iYLztG5IiK	user004	user004			2021-06-20 04:32:49.347174			
20210620134441579000353126iYKGiIL5tz	hmson	hmson			2021-06-20 04:44:41.676558			
20210620134441579000353126iYKGiIL5tz	hushsbay	hushsbay			2021-06-20 04:44:41.676933			
20210620134441579000353126iYKGiIL5tz	jspark	jspark			2021-06-20 04:44:41.677357			
20210620134441579000353126iYKGiIL5tz	user030	user030			2021-06-27 08:13:20.387618			
20210620134441579000353126iYKGiIL5tz	user045	user045			2021-06-27 08:13:07.557360			

1 • SELECT * FROM jayops.A_MSGMST_TBL;

MSGID	ROOMID	SENDERID	SENDERNM	BODY	BUFFER	REPLY	TYPE	CDT	UDT	STATE	FILESTATE
20210620133302170000719635LK5itGizY	20210620...	hushsbay	hushsbay	Hi~ sendjay !!!	NULL		talk	2021-06-20 04:33:02.013732			
2021062013444971300071074915IYLgZkt	20210620...	hushsbay	hushsbay	hi, there~	NULL		talk	2021-06-20 04:44:49.561100			
202106201345013430003217825IYzGKtL	20210620...	hushsbay	hushsbay	whowho~	NULL		talk	2021-06-20 04:45:01.185978			
20210620134543152000928774RfLQUAv_XS	20210620...	jspark	jspark	what's up?	NULL		talk	2021-06-20 04:45:42.994683			
20210620134615136000507152XLv_fRSAQ	20210620...	jspark	jspark	hello, there~	NULL		talk	2021-06-20 04:46:14.981085			
20210620134705467000746296fJvXA_SRLQ	20210620...	jspark	jspark	hihi~	NULL		talk	2021-06-20 04:47:05.310617			
20210620134902350000239817zKIJ5tYGLI	20210620...	hushsbay	hushsbay	message cancelled	NULL		talk	2021-06-20 04:49:02.008210	2021-06-20 04:49:45.472641		
20210620134957812000209299Qfs_XLvRUA	20210620...	jspark	jspark	https://socket.io	NULL		talk	2021-06-20 04:49:57.676840			
20210620135408588000810977681XhjDaC8	20210620...	hushsbay	hushsbay	20210620134441579000...	NULL		file	2021-06-20 04:54:08.711115			expired
202106201415477950008292169dKQCKYy	20210620...	hushsbay	hushsbay	옥아~	NULL		talk	2021-06-20 05:15:48.170886			
202106201416188410001534219YkDQdCkKy	20210620...	hushsbay	hushsbay	여기 특에 찍히는 시각이 ...	NULL		talk	2021-06-20 05:16:19.290183			
20210620141634207000869894K9DkYkCyQd	20210620...	hushsbay	hushsbay	20210620141536800000...	NULL		file	2021-06-20 05:16:35.369245			expired
20210620173758171000600348y0efoVKd4d	20210620...	hushsbay	hushsbay	20210620134441579000...	NULL		file	2021-06-20 08:37:58.257207			expired

1 • SELECT * FROM jayops.A_MSGDTL_TBL;

MSGID	ROOMID	SENDERID	RECEIVERID	RECEIVERNM	CDT	UDT	STATE	PUSH_ERR
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	doob	ay	2021-06-20 04:33:02.013732		R	
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	hmson	hmson	2021-06-20 04:33:02.013732			
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	hushsbay	hushsbay	2021-06-20 04:33:02.013732	2021-06-27 08:32:38.521785	D	
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	jspark	jspark	2021-06-20 04:33:02.013732		R	
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	user002	user002	2021-06-20 04:33:02.013732			
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	user003	user003	2021-06-20 04:33:02.013732			
20210620133302170000719635LK5itGizY	20210620133248724000359755iYLztG5IiK	hushsbay	user004	user004	2021-06-20 04:33:02.013732			

1 • `SELECT * FROM jayops.A_FILELOG_TBL;`

Result Grid						
Filter Rows:						
Edit:						
Export/Import:						
Wrap Cell Content:						
MSGID	ROOMID	SENDERID	BODY	CDT	UDT	
20210620135408588000810977681XhjDaC8	20210620134441579000353126iYKGiIL5tz	hushsbay	test\$\$20210620045408826965.mp4	2021-06-20 04:54:08.494999	2021-06-20 04:59:11	
20210620141634207000869894K9DkYkCyQd	20210620141536800000356509J1CtiBFQoz	hushsbay	1623935296343\$\$20210620051634489015.jpg	2021-06-20 05:16:34.623060	2021-06-20 05:19:11	
20210620173758171000600348y0efoVKId	20210620134441579000353126iYKGiIL5tz	hushsbay	selecton_oragne\$\$20210620083758695174.png	2021-06-20 08:37:58.247050	2021-06-20 08:39:12	
202106240103051800004116298iwouQXti4	20210620134441579000353126iYKGiIL5tz	hushsbay	test\$\$20210623160305135539.mp4	2021-06-23 16:03:05.361490	2021-06-23 16:09:53	

99. 향후 해결해야 할 이슈

1. sendjay는 특정 개발 조직이 아닌 개인 한명이 자비로 만든 소규모 사이트이므로 성능/용량/보안상의 이슈가 발생할 수도 있습니다. 1개의 서버로 개발/운영서버를 같이 사용하므로 배포시 서버다운이나 GitHub소스 Push등에 일시적인 문제가 발생할 수도 있습니다. 메신저 서버에 저장되는 각종 데이터는 테스트 용도임을 전제로 하며 통보없이 언제든지 삭제될 수도 있음을 말씀 드립니다.
2. AWS에서 Application Load Balancer (ALB)가 아닌 Classic Load Balancer (CLB)를 사용한 것은 ALB에서는 1 EC2로 구성하지 못해 비용 문제가 발생했기 때문입니다. ALB에서는 http -> https 리다이렉션이 편리하게 가능한데 CLB with NodeJS (without Nginx)에서는 쉽게 해결하지 못했습니다. (현재 Nginx 서비스 없음)
3. 모든 웹 브라우저 지원이 현실적으로 쉽지 않아서 크롬,엣지,네이버웨일 웹브라우저에서만 테스트하였습니다. (iOS는 별도)

4. 프로그래밍

- 1) Android에서는 ForegroundService를 사용하는데 이 때 Notification Icon을 자동으로 제거하지 못해
사용자로 하여금 최초 1회 수동으로 제거하도록 안내합니다.
- 2) 현재 서버아이디로 클라이언트 소켓으로 전송하는 기능은 지원하지 않고 있습니다.
 - 전체 또는 특정 룸에 브로드캐스트 메시지 전송
- 3) 파일 업로드
 - 업로드 중에 토크방을 닫으면 업로드가 중지됩니다. 중지되고 재전송시 계속 이어서 보내기가 구현되어 있지 않습니다.
- 4) 모바일에서 Webview를 사용하므로 카카오톡과 비교해 보면 아래와 같은 이슈가 발생합니다.
 - sendjay에서는 재전송 처리 관련해서만 로컬DB(HTML5 IndexedDB)를 사용합니다.
 - 네트워크 연결이 안 될 경우엔 토크방을 열 수 없습니다. 토크방이 열리고 데이터를 모두 가져오기 전에 연결이 끊기면 토크방이 열리지 않고 재시도(Retry)화면이 열립니다. Android의 경우 WebviewClient object의 OnReceiverError()에서 오류 처리화면을 열거나 안내메시지를 주는 방식으로 처리합니다. (카카오톡은 연결여부는 알 수 없지만 반드시 토크방은 열림)
- 5) 투표, 비밀채팅 등 투입비용에 비해 (사내메신저에서는) 효용이 적다고 판단한 구성요소들은 개발 제외하였습니다.
- 6) 카카오톡처럼 다양한 이모티콘 생태계를 이용할 수 없습니다. 대신, Emoji를 지원하는데,
이모지를 체크하는 함수(chkEmoji)가 현재 완벽하지 않습니다. (regex에 korean으로만 패턴 체크하는 것은 개선이 필요합니다)

끝까지 읽어 주셔서 감사합니다 !!