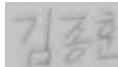
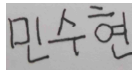



캡스톤 디자인 결과보고서

<차량 운행 로그와 고객 정보 분석을 통한 앱 서비스>

2022년 04월 25일

팀장: 김종훈 

팀원: 민수현 

지도교수: 강현수 

목 차

1. 연구 계획	1
2. 관련 지식	2
3. 업무 분담	3
4. 문제점 및 개선사항	4
5. 작품 설명서	5
6. 결과 분석 및 고찰	6
7. 회의록	7
8. 작품 설명서	8
9. 하드웨어 설계도 또는 프로그램 소스	10

연구계획

연구제목	차량 운행 로그와 고객 정보 분석을 통한 앱 서비스
연구의 필요성	<p>4차산업혁명으로 인한 자동차 관련 산업은 계속 발전하고 있다. 하지만, 자동차 관련한 산업에서 실시간으로 발생하는 대량의 데이터를 처리하기 위한 플랫폼과 서비스가 부족하다.</p> <p>실시간으로 발생하는 데이터는 자동으로 수집 할 수 있어야 하며, 수집 과정 데이터의 손실을 최소화 해야한다. 이렇게 수집된 데이터를 가지고 고객에게 서비스를 제공할 수 있어야 한다. 우리는 이러한 플랫폼의 구축과 앱서비스의 구현을 목적으로 이 연구의 필요성을 제시한다.</p>
연구 내용	<p>로그 시뮬레이터를 활용해 가상의 차량에 대한 속도, 위치, 차량번호, 고객 정보, 고객의 쇼핑 정보를 제공받아서 하둡 에코 시스템을 활용해서 데이터에 대한 수집, 적재, 탐색, 응용을 적용해 본 결과 고객에 대한 정보와 차량의 정보를 조사해서 인사이트를 얻어냈습니다. 이러한 결과를 가지고 고객에 대한 차량 용품 추천 및 과속 차량 정보를 실시간으로 고객에게 제공함에따라 빅데이터를 활용한 고객 맞춤 서비스를 가능하게 만들었습니다.</p>
참고 자료	<p>1. 연구보고서 : Hadoop eco-system 기반 고급 빅데이터 분석 플랫폼 개발 (등록번호: TRKO201600005426)</p> <p>2. 관련 특허 : 빅데이터 처리 시스템 및 처리 방법 (Bigdata processing system and method) (출원번호: 1020160008634)</p> <p>3. 관련 특허 : 리뷰정보 기반 음식점 추천 방법 및 시스템 (RESTAURANT RECOMMENDATION METHOD BASED ON REVIEW DATA AND SYSTEM THEREOF) (출원번호: 1020200055704)</p>

관련 지식

연구제목	차량 운행 로그와 고객 정보 분석을 통한 앱 서비스
연구의 필요성	<p>4차산업혁명으로 인한 자동차 관련 산업은 계속 발전하고 있다. 하지만, 자동차 관련한 산업에서 실시간으로 발생하는 대량의 데이터를 처리하기 위한 플랫폼과 서비스가 부족하다.</p> <p>실시간으로 발생하는 데이터는 자동으로 수집 할 수 있어야 하며, 수집 과정 데이터의 손실을 최소화 해야한다. 이렇게 수집된 데이터를 가지고 고객에게 서비스를 제공할 수 있어야 한다. 우리는 이러한 플랫폼의 구축과 앱서비스의 구현을 목적으로 이 연구의 필요성을 제시한다.</p>
연구 내용	<p>로그 시뮬레이터를 활용해 가상의 차량에 대한 속도, 위치, 차량번호, 고객 정보, 고객의 쇼핑 정보를 제공받아서 하둡 에코 시스템을 활용해서 데이터에 대한 수집, 적재, 탐색, 응용을 적용해 본 결과 고객에 대한 정보와 차량의 정보를 조사해서 인사이트를 얻어냈습니다. 이러한 결과를 가지고 고객에 대한 차량 용품 추천 및 과속 차량 정보를 실시간으로 고객에게 제공함에따라 빅데이터를 활용한 고객 맞춤 서비스를 가능하게 만들었습니다.</p>
참고 자료	<p>1. 연구보고서 : Hadoop eco-system 기반 고급 빅데이터 분석 플랫폼 개발 (등록번호: TRKO201600005426)</p> <p>2. 관련 특허 : 빅데이터 처리 시스템 및 처리 방법 (Bigdata processing system and method) (출원번호: 1020160008634)</p> <p>3. 관련 특허 : 리뷰정보 기반 음식점 추천 방법 및 시스템 (RESTAURANT RECOMMENDATION METHOD BASED ON REVIEW DATA AND SYSTEM THEREOF) (출원번호: 1020200055704)</p>

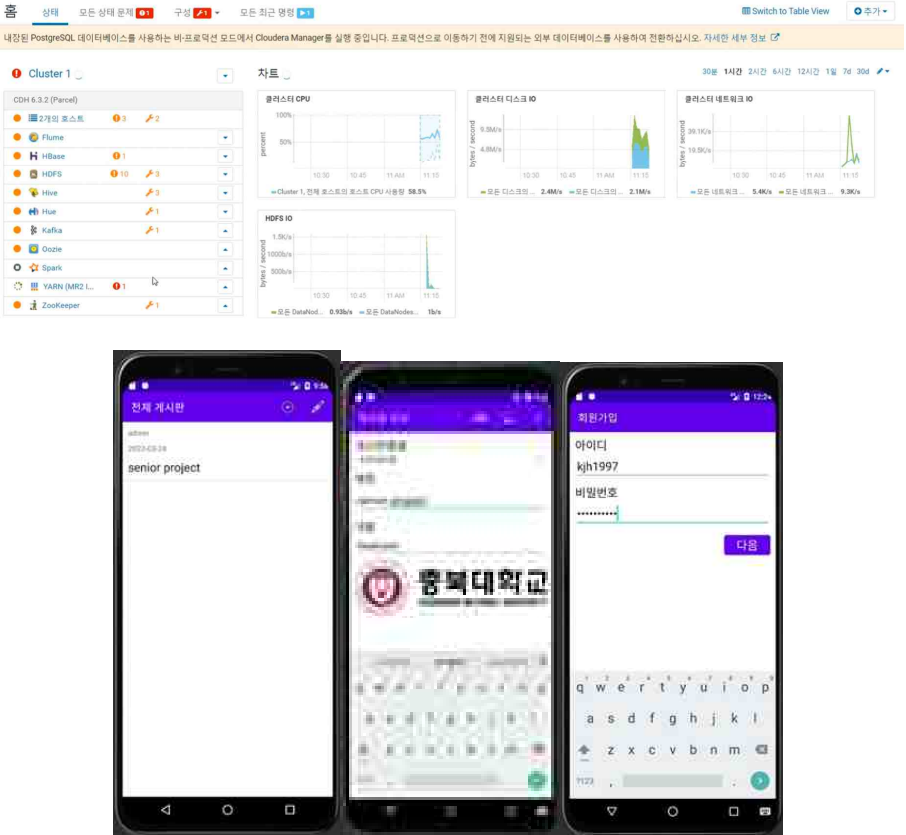
업무 분담 (Member's Mission)

역할	업무	성명	진행도 (100%)
빅데이터 플랫폼 개발	빅데이터의 수집, 적재, 탐색, 응용을 위한 하둡 에코 시스템 설계	김종훈	100%
협업 필터링 기반 머신러닝 모델 개발	고객의 차량 용품 구매내역을 활용하여 collaborative filtering기반 용품 추천 서비스	김종훈	100%
안드로이드 애플리케이션 개발	코틀린을 활용한 안드로이드 애플리케이션 개발	민수현	80%
서버 개발 및 데이터베이스 설계	flask, mysql, firebase를 활용한 서버 및 데이터베이스 설계	김종훈, 민수현	100%

문제점 및 개선사항

일자	문제점	개선사항	담당
21/09/13	리눅스 ubuntu 이미지를 받아 개발환경 구성 중 각 노드의 네트워킹이 원활하지 않아 문제 발생.	리눅스 os를 centos로 변경 후 문제 해결	김종훈
21/10/18	hdfs의 datanode가 2개밖에 되지 않아 data flow의 과부하가 걸려서 다운되는 문제가 발생.	datanode를 3개까지 늘리고 복제 계수와 블록 크기를 조정해서 해결	김종훈
21/12/22	머신러닝 모델을 도커를 활용해서 쿠버네티스 클러스터에 배포하려는 도중 flask와 nginx의 연동이 안되는 문제 발생	Dockerfile에서의 문법 오류. 각 이미지의 port를 매핑해주어야 하는데, port mapping을 해주고 문제 해결	김종훈
22/01/27	redis에서 firebase로의 data export가 안되는 문제 발생	google firebase authorization key를 새로 발급받아 문제 해결	김종훈
22/02/11	firebase authentication 기능을 이용하여 회원가입 기능을 구현하는데 문제 발생	firebase 대신 mysql을 이용하여 회원가입과 로그인 기능을 구현하여 문제 해결	민수현
22/03/17	앱의 모든 기능을 activity로 구성하였더니 앱이 동작하는데 무겁고 느리다는 문제 발생	main activity를 제외한 activity들을 fragment로 대체하여 구성하여 문제 해결	민수현

작품 설명서

작품명	차량 운행 로그와 고객 정보 분석을 통한 앱 서비스	작성일자	2022/04/25
팀명	가짜개발자	작성자	김중훈
제작기간	2021/09/01 ~ 2022/04/25	제작비	0₩
작품 사진			
작품 기능	<ol style="list-style-type: none"> 1. 안드로이드 애플리케이션에서 과속차량들의 정보를 볼 수 있다. 2. 안드로이드 애플리케이션에서 고객들과 상담 및 소통할 수 있는 게시판 기능을 구현한다. 3. 안드로이드 애플리케이션에서 고객들에게 차량과 관련된 용품을 추천하여 홍보효과를 줄 수 있다. 		
우수한 점	<ol style="list-style-type: none"> 1. 도커와 쿠버네티스를 활용하여 대량의 traffic이 들어와도 로드 밸런싱 기능을 활용하여 뛰어난 안정성과 확장성을 보장한다. 2. 빅데이터 클러스터를 구성하여 하나의 노드가 다운되어도 안정성이 보장된다. 3. 데이터의 수집 및 분석의 자동화로 빠르게 변화하는 사용자의 요구사항에 적응할 수 있도록 한다. 		
개선할 점	<p>이 프로젝트에 사용되는 데이터는 가상의 데이터이므로, 실제 데이터를 가지고 테스트하는 작업이 필요하다. 또한 애플리케이션 개발이 아직 완료되지 않았으므로 추후적인 개발이 필요하다.</p>		

결과 분석 및 고찰

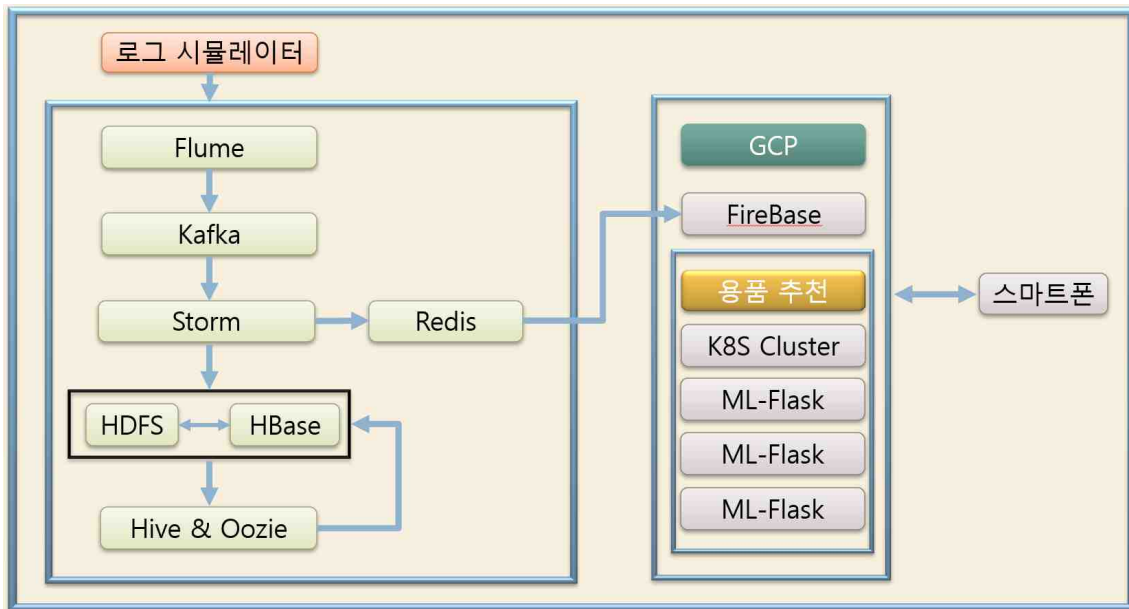
작품명	차량 운행 로그와 고객 정보 분석을 통한 앱 서비스	작성일자	2022/04/25
팀명	가짜개발자	작성자	김중훈
제작기간	2021/09/01 ~ 2022/04/25	제작비	0₩
결과 분석 및 고찰	<p>이번 프로젝트를 통해 빅데이터로부터 인사이트를 창출하여 다양한 서비스를 제공해봤습니다.</p> <p>실제 실시간 데이터와 실제 서버를 빌려 빅데이터 플랫폼을 구성하려면 비용이 많이 필요해서 이번 프로젝트에는 적용하지 못하고 Virtual box를 활용해서 구현하였습니다.</p> <p>하지만, 로그 시뮬레이터를 활용하여 실제 데이터와 비슷한 양상을 보이는 반정형 데이터를 가지고 다양한 하둡 생태계의 오픈소스를 활용해보으로써 빅데이터와 반정형, 비정형 데이터에 대한 접근 방식을 이해했습니다.</p> <p>또한 협업 필터링 알고리즘을 사용하여 사용자와 자기 자신이 남긴 용품에 대한 별점을 가지고 용품을 추천하는 머신러닝 모델을 개발하고 머신러닝 모델에 대한 지속, 확장 가능한 서비스를 제공하기 위해 쿠버네티스와 도커를 이용하여 버전 관리를 할 수 있게 구현하였습니다.</p> <p>이러한 서비스를 가지고 코틀린을 활용한 안드로이드 애플리케이션을 구현하고 있고, 구글 Play store에 출시하는 경험까지 생각하고 있습니다.</p> <p>이번 프로젝트를 진행하면서 4차 산업혁명의 빅데이터, 인공지능 분야를 모두 경험해보았습니다. 실제 현업에서 사용하는 빅데이터 플랫폼 아키텍처는 더욱 복잡하고 설계하기 힘들겠지만 이번 프로젝트의 경험을 원동력으로 삼아 노력하고 발전하겠습니다.</p>		

회의록

팀명	가짜개발자		
회의일시	2022/04/17	회의장소	온라인
참석자	김종훈, 민수현		
<div>1. 개발 진행 상황 검토</div> <div>- 빅데이터 플랫폼 개발 진행 상황 : Flume, Kafka, Hadoop 클러스팅 노드, Storm 토폴로지, Redis와 Firebase 클라이언트, Hbase까지 구현한 상태</div> <div>- 안드로이드 개발 진행 상황 : 로그인과 회원가입 기능, 게시판 기능까지 구현한 상태</div> <div>- 개발하면서 생겼던 문제점들에 대해서 논의</div> <div>2. 추가로 구현할 기능들에 대해서 논의</div> <div>- 애플리케이션 디자인 변경 필요</div> <div>- 내비게이션 뷰를 구현할지 탭 레이아웃을 구현할지에 대해 토의하였고 탭 레이아웃으로 결정</div> <div>- 용품 추천 기능과 과속 차량 정보를 알려주는 기능 구현 필요</div> <div>3. 결과보고서 작성</div> <div>- 연구 계획, 관련 지식, 업무 분담, 문제점 및 개선사항, 작품 설명서, 결과 분석 및 고찰, 회의록, 작품 설명서, 하드웨어 설계도 또는 프로그램 소스 작성</div> <div>4. 향후 일정 논의</div> <div>- 미개발된 안드로이드 애플리케이션 개발을 마무리 해야 함.</div> <div>- 안드로이드 애플리케이션 개발 완료 후 통합 테스트를 진행해야 함.</div>			

프로그램 설계도 및 소스코드

https://github.com/JaeYeopHan/Interview_Question_for_Beginner



Redis에서 Firebase로 실시간 data expose

```
try {
    while(true) {

        overSpeedCarList = jedis.smembers(key);

        System.out.println("=====");
        System.out.println("=====");
        System.out.println("=====");

        System.out.println("\n[ Try No. " + cnt++ + " ]");

        if(overSpeedCarList.size() > 0) {
            for (String list : overSpeedCarList) {
                System.out.println(list);
                insert(list);
            }
            System.out.println("");
            jedis.del(key);
        }else{
            System.out.println("\nEmpty Car List...\n");
        }

        System.out.println("=====");
        System.out.println("=====");
        System.out.println("=====");
        System.out.println("\n\n");

        Thread.sleep(10 * 1000);
    }
} catch (Exception e) {
    e.printStackTrace();
} finally {
    if( jedis != null ) jedis.close();
}
```

Firestore접속을 위한 Client 구현

```
public static void init() throws Exception{
    FileInputStream refreshToken = new FileInputStream(PATH);
    option = new FirebaseOptions.Builder()
        .setCredentials(GoogleCredentials.fromStream(refreshToken))
        .setDatabaseUrl("https://kjh-prj-default-rtdb.firebaseio.com")
        .build();
    FirebaseApp.initializeApp(option);
}

public static void makeDatabaseConn(){ //Firestore 인스턴스 생성
    db = FirestoreClient.getFirestore();
}

public static void select(){ //선택
    db.collection(COLLECTION_NAME).addSnapshotListener( (target, exception)->{
        System.out.println(" - insert start - ");
    });
}

public static void insert(String a){ //삽입
    Map<Object, Object> item = new HashMap<Object, Object>();
    item.put("test", a);
    db.collection(COLLECTION_NAME).add(item);
}
}
```

Flume driver 구현

```
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.DriverCarInfo_TailSource
SmartCar_Agent.channels.SmartCarInfo_Channel.DriverCarInfo_Channel
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.DriverCarInfo_KafkaSink

SmartCar_Agent.sources.SmartCarInfo_SpoolSource.type = spooldir
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.spoolDir = /home/kjh-prj/working/car/batch-log
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.deletePolicy = immediate
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.batchSize = 1000

SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.timestamp.type = timestamp
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.timestamp.preserveExisting = true

SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.typeInterceptor.type = static
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.typeInterceptor.key = logType
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.typeInterceptor.value = carbatch-log

SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.collectDayInterceptor.type =
com.wikibook.bigdata.smartcar.flume.CollectDayInterceptor$Builder

SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.filterInterceptor.type = regex_filter
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.filterInterceptor.regex = "(\\d{14})"
SmartCar_Agent.sources.SmartCarInfo_SpoolSource.interceptors.filterInterceptor.excludeEvents = false

SmartCar_Agent.channels.SmartCarInfo_Channel.type = memory
SmartCar_Agent.channels.SmartCarInfo_Channel.capacity = 100000
SmartCar_Agent.channels.SmartCarInfo_Channel.transactionCapacity = 10000

SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.type = hdfs
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.path = /kjh-prj/collect/${logType}/wri_data/${Y}/m/${d}
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.filePrefix = ${logType}
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.fileSuffix = .log
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.fileType = DataStream
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.writeFormat = Text
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.batchSize = 10000
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.rollInterval = 0
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.rollCount = 0
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.idleTimeout = 100
SmartCar_Agent.sinks.SmartCarInfo_HdfsSink.hdfs.callTimeout = 600000
```

안드로이드 앱 주요 소스코드

1. 회원가입

```
joinFragmentBinding.joinNextBtn.setOnClickListener{  
    .....val joinId = joinFragmentBinding.joinId.text.toString()  
    .....val joinPw = joinFragmentBinding.joinPw.text.toString()  
  
    .....if(joinId == null || joinId.length == 0){  
        .....val dialogBuilder = AlertDialog.Builder(requireContext())  
        .....dialogBuilder.setTitle("아이디 입력 오류")  
        .....dialogBuilder.setMessage("아이디를 입력해주세요")  
        .....dialogBuilder.setPositiveButton(text: "확인"){ dialogInterface: DialogInterface, i: Int ->  
            .....joinFragmentBinding.joinId.requestFocus()  
            .....}  
        .....dialogBuilder.show()  
        .....return@setOnClickListener  
    .....}  
  
    .....if(joinPw == null || joinPw.length == 0){  
        .....val dialogBuilder = AlertDialog.Builder(requireContext())  
        .....dialogBuilder.setTitle("비밀번호 입력 오류")  
        .....dialogBuilder.setMessage("비밀번호를 입력해주세요")  
        .....dialogBuilder.setPositiveButton(text: "확인"){ dialogInterface: DialogInterface, i: Int ->  
            .....joinFragmentBinding.joinPw.requestFocus()  
            .....}  
        .....dialogBuilder.show()  
        .....return@setOnClickListener  
    .....}
```

2. 로그인

```
loginFragmentBinding.loginLoginbtn.setOnClickListener{ it: View!
    val loginId = loginFragmentBinding.loginId.text.toString()
    val loginPw = loginFragmentBinding.loginPw.text.toString()
    val chk = loginFragmentBinding.loginAutologin.isChecked

    var loginAutoLogin = 0
    if (chk == true){
        loginAutoLogin = 1
    } else{
        loginAutoLogin = 0
    }

    // 유효성 검사
    if (loginId == null || loginId.length == 0){
        val dialogBuilder = AlertDialog.Builder(requireContext())
        dialogBuilder.setTitle("아이디 입력 오류")
        dialogBuilder.setMessage("아이디를 확인해주세요")
        dialogBuilder.setPositiveButton(text: "확인"){ dialogInterface: DialogInterface, i: Int ->
            loginFragmentBinding.loginId.requestFocus()
        }
        dialogBuilder.show()
        return@setOnClickListener
    }

    if (loginPw == null || loginPw.length == 0){
        val dialogBuilder = AlertDialog.Builder(requireContext())
        dialogBuilder.setTitle("비밀번호 입력 오류")
        dialogBuilder.setMessage("비밀번호를 확인해주세요")
        dialogBuilder.setPositiveButton(text: "확인"){ dialogInterface: DialogInterface, i: Int ->
            loginFragmentBinding.loginPw.requestFocus()
        }
        dialogBuilder.show()
        return@setOnClickListener
    }
}
```

3. Recycler View를 활용한 게시판 목록

```
val boardMainRecyclerAdapter = BoardMainRecyclerAdapter()
boardMainFragmentBinding.boardMainRecycler.adapter = boardMainRecyclerAdapter

boardMainFragmentBinding.boardMainRecycler.layoutManager = LinearLayoutManager(requireContext())
boardMainFragmentBinding.boardMainRecycler.addItemDecoration(DividerItemDecoration(requireContext(), orientation: 1))
boardMainFragmentBinding.boardMainRecycler.addOnScrollListener(object :: RecyclerView.OnScrollListener(){ // 스크롤 발생 시 발생
    override fun onScrolled(recyclerView: RecyclerView, dx: Int, dy: Int) {
        super.onScrolled(recyclerView, dx, dy)

        // 현재 화면에 보이는 항목 중 제일 마지막 항목의 인덱스를 가지고 올
        val index1 = (recyclerView.layoutManager as LinearLayoutManager).findLastVisibleItemPosition()

        // recycler view가 관리하는 항목의 총 개수
        val count1 = recyclerView.adapter?.itemCount

        if(index1 + 1 == count1){
            act.nowPage = act.nowPage + 1
            getContentList(clear: false) // 화면 내려도 안없어지게 함
        }
    }
})
```

GCP 서버 및 데이터베이스 구현

K8S Cluster

```
kjh09047@cloudshell:~/dockertest (shaped-snowfall-347701)$ k get svc
NAME                TYPE                CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
flask                LoadBalancer       10.8.1.144    34.71.135.217  80:31808/TCP     4m15s
kubernetes           ClusterIP           10.8.0.1      <none>         443/TCP          22h
```

```
kjh09047@cloudshell:~/dockertest (shaped-snowfall-347701)$ k get pod
NAME                                READY   STATUS    RESTARTS   AGE
flask-deploy-746b58bbb4-g56xs      1/1     Running   0          6m30s
flask-deploy-746b58bbb4-rrzjx      1/1     Running   0          6m30s
flask-deploy-746b58bbb4-rtvn8      1/1     Running   0          6m30s
```

Firebase

