

Holiday Notifier

2020학년도

전공 종합 설계 최종보고서

비정기 휴업 공지 시스템

팀장 : 도우진

팀원 : 강민철, 김범승, 장예찬, 정현석

목 차

1. 요약문	02
2. 서론	02
가. 추진 배경 및 목적	02
나. 개발 목표와 내용	02
3. 관련 연구	02
가. 기존 개발연구 동향 파악	02
나. 우리 시스템과 비교	02
4. 설계 및 구현	03
가. 개발환경	03
나. 시스템 설계	04
다. 하위 프로젝트 설계 및 디자인	05
5. 구현 결과	09
가. 클라이언트 구현 결과	09
나. 개인별 작업 내용	13
6. 결론	29
가. 문제점 분석	29
나. 향후 계획	30
다. 기대효과	30
라. 총론	30
7. 참고문헌	30

Holiday Notifier

1. 요약문

이 프로젝트는 식당이나 가게, 공공시설 등의 운영상 발생할 수도 있는 비정기 휴업을 시설 관리자가 시설을 이용하고자 하는 이용자들에게 쉽게 공지를 할 수 있게 하는 시스템을 제공한다. 시설의 관리자가 부득이한 일정(경조사, 여행, 내부 공사 등)으로 인해 휴업하게 되는 경우, 소규모 업장은 이를 공지할 수 있는 수단이 문 앞이나 창문에 안내문을 붙이는 정도로 제한되어 있다. 그렇기에 해당 시설의 이용자는 해당 시설을 방문하고 나서야 휴업 사실을 인지할 수 있다. 이 점을 해결하고자 해당 프로젝트를 시작하게 되었으며 기본적인 시스템의 골격을 완성하는 성과를 거두었다.

2. 서론

가. 추진 배경 및 목적

주변 맛집, 카페, PC방, 도서관이 갑자기 여행이나 공사 등의 이유로 임시 휴일일 때 헛걸음한 경험을 한 번쯤은 갖고 있을 것이다. 가게의 문이나 창문에 휴업 공지를 하고, 도서관 등의 큰 시설은 자체적인 홈페이지 등의 플랫폼으로 공지하고 있지만, 통합적인 공지 플랫폼이 없어 일일이 확인하기 귀찮고, 자그마한 가게나 식당의 경우 이러한 공지 플랫폼마저 없어 직접 가거나 전화하지 않고는 확인할 방법이 없다. 이 프로젝트를 통해 이런 난처한 경험을 줄이고, 일정과 동선을 계획할 때 방문할 시설의 휴업 여부를 고려할 수 있도록 하여 시간과 비용을 절약할 수 있을 것이다.

나. 개발 목표와 내용

관리자용 앱을 통해 관리자가 시설의 기본적인 영업시간, 영업 요일 등을 등록하고 관리할 수 있다.

관리자는 공사, 여행 등으로 인한 임시 휴업 시 앱에 등록하여 고객들이 볼 수 있도록 공지한다.

고객은 고객용 앱을 통해 원하는 시설(식당, 공원, 도서관 등)을 조회하여 영업시간을 볼 수 있고, 즐겨찾기로 등록하여 관리자가 임시 휴업 등록 시 해당 고객들에게 푸시 알림을 제공하는 것을 목표로 한다.

3. 관련 연구

* 기존 개발연구 동향 파악

구글 지도 등에서 휴업 여부를 표시하는 예가 있음

* 우리 시스템과 비교

개인, 소상공인이 운영하는 점포나 공공시설 등 규모·용도에 관계없이 모든 시설에 대해 공지하고 공지 받을 수 있도록 하며, 이러한 공지 등록 시 즐겨 찾기 한 사용자에게 푸시 알림을 제공

Holiday Notifier

4. 설계 및 구현

사용대상	관리용 앱: 사장, 직원 등, 고객용 앱: 일반 사용자
기능	1. 시설의 기본적인 정보와 영업시간, 영업 요일 등을 등록하여 고객이 해당 정보를 볼 수 있음 2. 임시 휴업 시 앱에 등록하여 고객들이 볼 수 있도록 공지
범위	시설: 유형의 고정된 장소(공원 등), 수단(기차, 항공 등) 관리자: 대표자(업주) 및 대표자가 등록한 사람(직원 등)

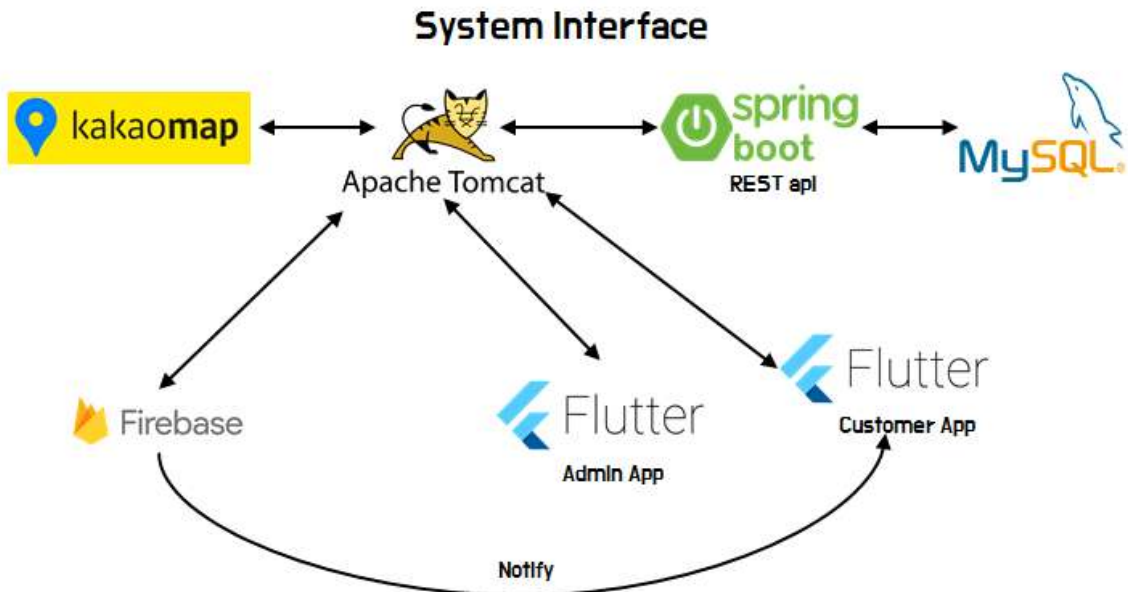
가. 개발환경

CLIENTS	
Framework	Flutter 1.17.1 • channel stable
IDE	Android Studio 3.6.3
Language	Dart 2.8.2
App native languages	
android	Open JDK 11 (Gradle 6.3 build, plugin 3.6.3), Kotlin 1.3.10
android-ndk	21.0.6113669
iOS	Swift
SERVER	
Framework	Spring boot Starter Parent 2.3.0.RELEASE
IDE	IntelliJ IDEA 2019.3.3 (Ultimate Edition)
Language	Open JDK 11 (Maven 3.6.1 build)
DB	5.7.28 MySQL Community Server
WAS	Apache Tomcat 9.0.27
Host	Google Cloud Platform
Server OS	CentOS Linux release 7.7.1908
VCS	
VCS	Git, GitKraken(GUI)
Host	GitHub (https://github.com/NeoMindStd/HoliNoti)

Holiday Notifier

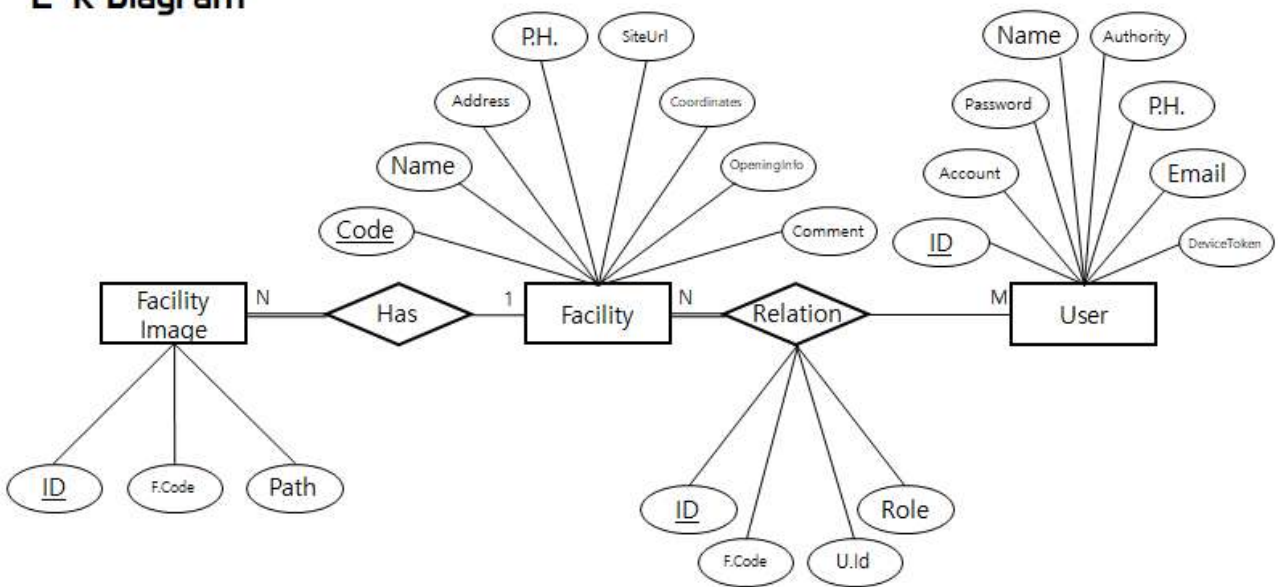
나. 시스템 설계

* 시스템 인터페이스



* E-R 다이어그램

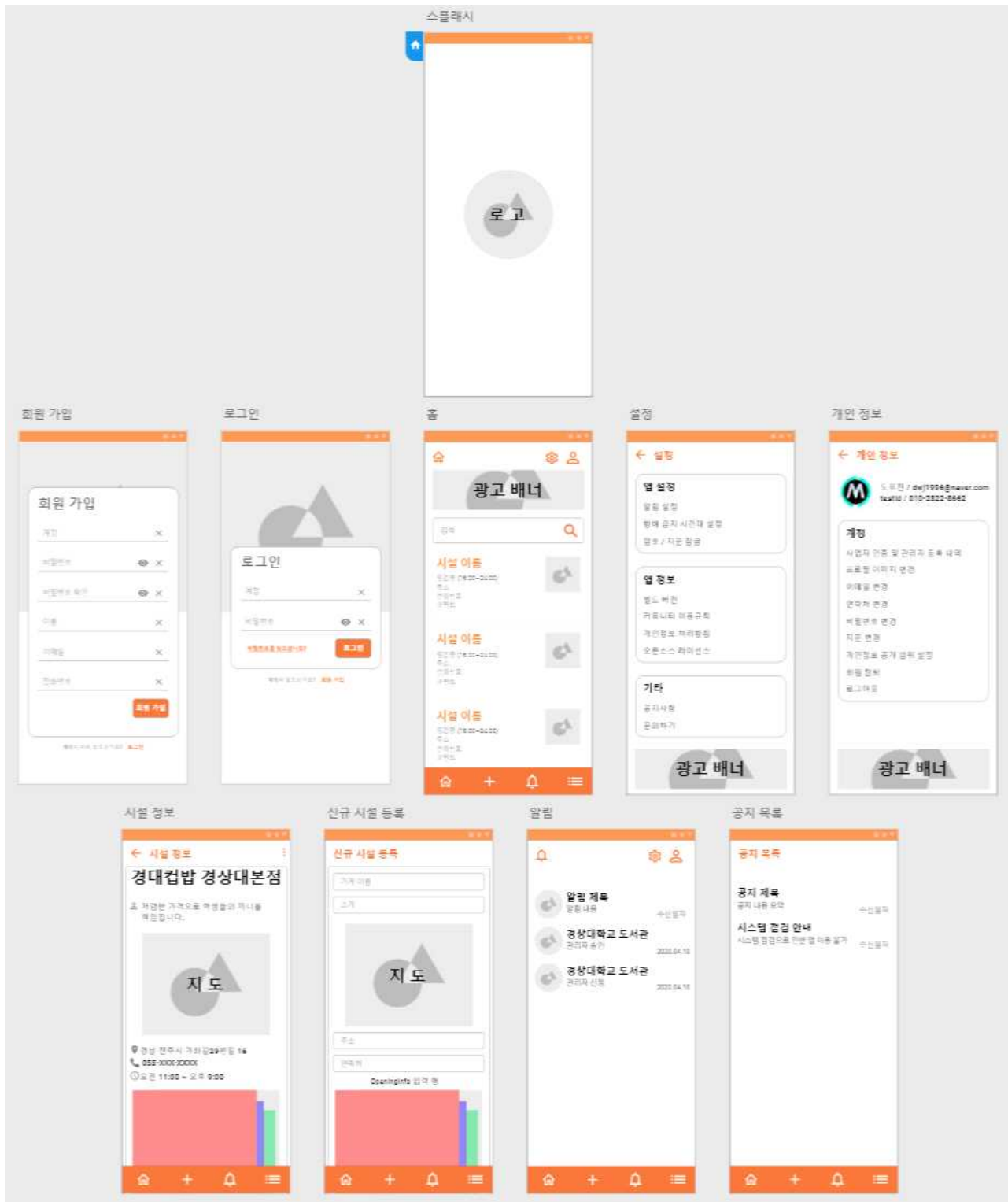
E-R Diagram



Holiday Notifier

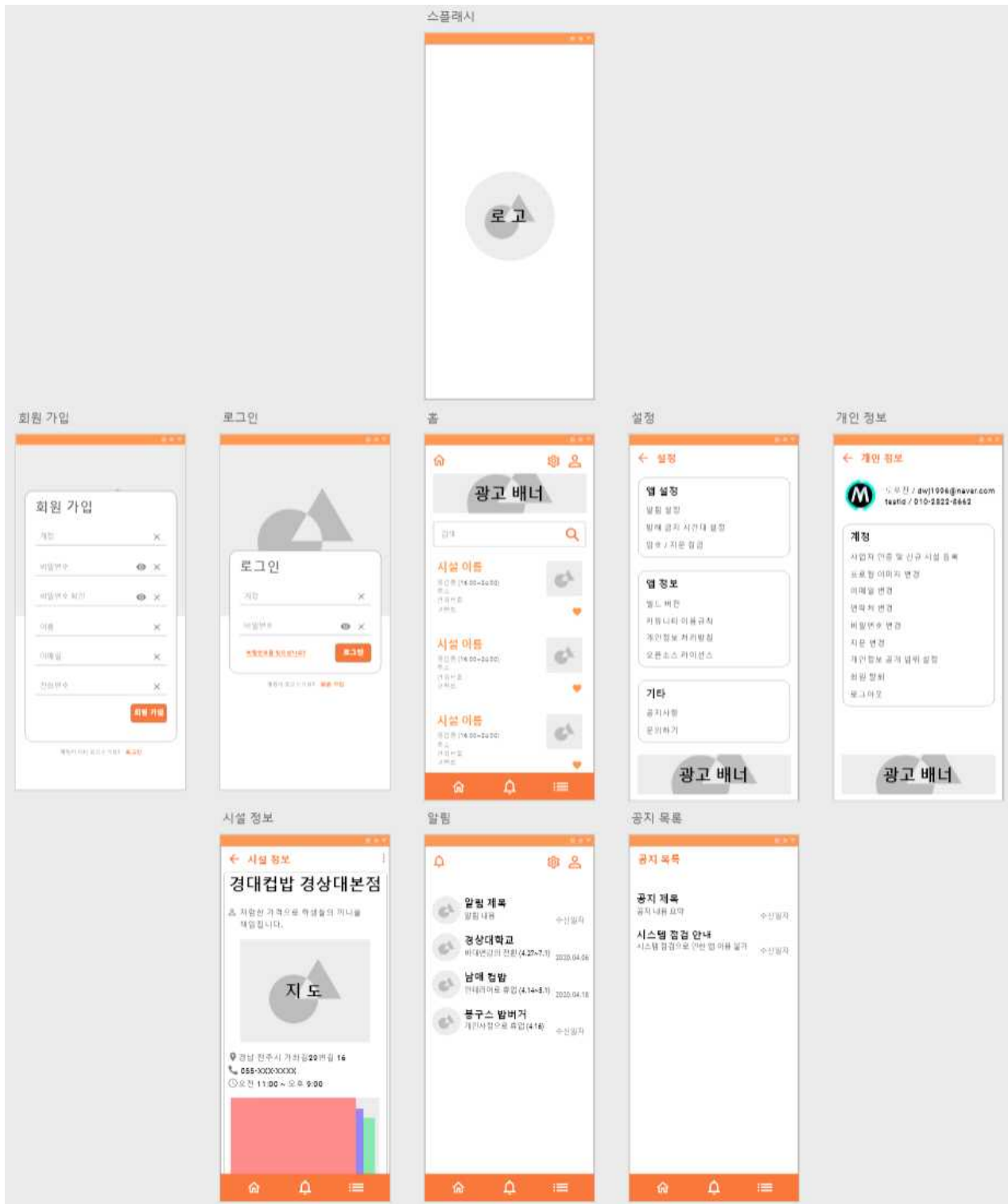
가. 하위 프로젝트 설계 및 디자인

* 관리자용 앱 UI 설계



Holiday Notifier

* 고객용 앱 UI



Holiday Notifier

* 네트워크 URI 설계

전체 URL

- GCP 서버(도메인): http://holinoti.tk:8080/holinoti/
- GCP 서버(IP주소, 변경될 수도 있습니다): http://35.234.41.179:8080/holinoti/
- 로컬 테스트 시(.war 배포 후 톰캣 서버 따로 구동): http://localhost:8080/holinoti/
- 로컬 테스트 때(인텔리)에서 바로 실행): http://localhost:8080/

Authority 위계(계정 유형)

- admin(시스템 관리 계정)
- normal(일반 계정)

Role 위계(권한)

- supervisor(점검 등 관리자 관리 권한을 갖는 관리자)
- manager(직원, 알바 등)
- customer(해당 시설을 구독한 고객)

기능별 URI

getAllFacilities(): /facilities(전체 접근 가능)

- GET

전체 접근 가능

* getFacility(@PathVariable("facilityCode") int code): /code={facilityCode}

* getFacilityByPhoneNumber(@PathVariable("phoneNumber") String phoneNumber):

/phone_number={phoneNumber}

* getFacilitiesByCoordinates(@PathVariable("x") double x, @PathVariable("y") double y,

@PathVariable("distanceM") int side): /x={x}/y={y}/distance_m={distanceM}

* getFacilitiesByCoordinatesAndName(@PathVariable("x") double x, @PathVariable("y") double y,

@PathVariable("distanceM") int side, @PathVariable("name") String name):

/x={x}/y={y}/distance_m={distanceM}/name={name}

* getFacilitiesByName(@PathVariable("name") String name): /name={name}

- POST

Authority: normal 이상 접근 가능 (Role 규칙이 없는 이유는, 추후 사업자 인증 등을 통해 해결할 예정이기 때문)

* addFacility(@RequestBody Facility facility)

- PUT

Authority: normal, Role: manager 이상 접근 가능

* updateFacility(@RequestBody Facility facility,

@PathVariable("facilityCode") int code): /code={facilityCode}

- DELETE

Authority: normal, Role: supervisor 이상 접근 가능

* deleteFacility(@PathVariable("facilityCode") int code): /code={facilityCode}

getAllFacilityImages(): /facilities/facility_images(Authority: admin만 접근 가능)

- GET

전체 접근 가능

* getFacilityImage(@PathVariable("facilityImageId") int id): /id={facilityImageId}

* getFacilityImagesByFacilityCode(@PathVariable("facilityCode") int facilityCode): /facility_code={facilityCode}

- POST

Authority: normal, Role: manager 이상 접근 가능

* addFacilityImage(@RequestBody FacilityImage facilityImage)

- PUT

Authority: normal, Role: manager 이상 접근 가능

* updateFacilityImage(@RequestBody FacilityImage facilityImage,

@PathVariable("facilityImageId") int id): /id={facilityImageId}

- DELETE

Authority: normal, Role: manager 이상 접근 가능

* deleteFacilityImage(@PathVariable("facilityImageId") int id): /id={facilityImageId}

getAllRelationAF(): /relation_afs(Authority: admin만 접근 가능)

- GET

전체 접근 가능

* getRelationAFById(@PathVariable("relationAFId") int id): /id={relationAFId}

* getRelationAFsByFacilityCode(@PathVariable("facilityCode") int facilityCode): /facility_code={facilityCode}

(자신의 것만 가능)

* getRelationAFsByUserId(@PathVariable("userId") int userId): /user_id={user_id}

- POST

Authority: normal, Role: Role필드가 customer는 전체 접근 가능, manager 이상은 기존 supervisor이상 권한자 또는 사업자 인증 등을 통해 가능

Holiday Notifier

```
* addRelationAF(@RequestBody RelationAF relationAF)
- PUT
  Authority: normal, Role: Role필드가 customer는 전체 접근 가능, manager 이상은 supervisor이상 권한자 또는
  자신만 가능
  * updateRelationAF(@RequestBody RelationAF relationAF,
    @PathVariable("relationAFid") int id): /id={relationAFid}
- DELETE
  Authority: normal, Role: Role필드가 customer는 전체 접근 가능, manager 이상은 supervisor이상 권한자 또는
  자신만 가능
  * deleteRelationAF(@PathVariable("relationAFid") int id): /id={relationAFid}

getAllUsers(): /users(Authority: admin만 접근 가능)
- GET
  전체 접근 가능
  * login(): /login
  (비밀번호 초기화 등 특수한 경우만, 그 외에는 Authority: admin만 접근 가능)
  * compareUser(@PathVariable("account") String account, @PathVariable("password") String password):
  /compare/{account}/{password}
  * getUserByAccount(@PathVariable("userAccount") String account): /account={userAccount}
  (비밀번호 초기화 등 특수한 경우만, 그 외에는 Authority: admin만 접근 가능)
  * getUserByEmail(@PathVariable("email") String email): /email={email}
  (비밀번호 초기화 등 특수한 경우만, 그 외에는 Authority: admin만 접근 가능)
  * getUserByPhoneNumber(@PathVariable("phoneNumber") String phoneNumber):
  /phone_number={phoneNumber}
  Authority: admin만 접근 가능
  * getUserById(@PathVariable("userId") int id): /id={userId}
- POST
  전체 접근 가능
  * addUser(@RequestBody User user): /register
- PUT
  Authority: normal 이상 접근 가능
  * updateUser(@RequestBody User user,
    @PathVariable("userId") int id): /id={userId}
- DELETE
  Authority: normal 이상인 자신의 계정만 접근 가능
  * deleteUser(@PathVariable("userId") int id): /id={userId}

기타
- GET
  전체 접근 가능
  * map(@PathVariable("x") double x, @PathVariable("y") double y, ModelMap modelMap):
  /kakao_map/x={x}/y={y}
- POST
  Authority: normal, Role: manager 이상 접근 가능
  * send(@PathVariable("facilityCode") int facilityCode, @RequestBody Notification notification):
  /notifications/facility_code={facilityCode}

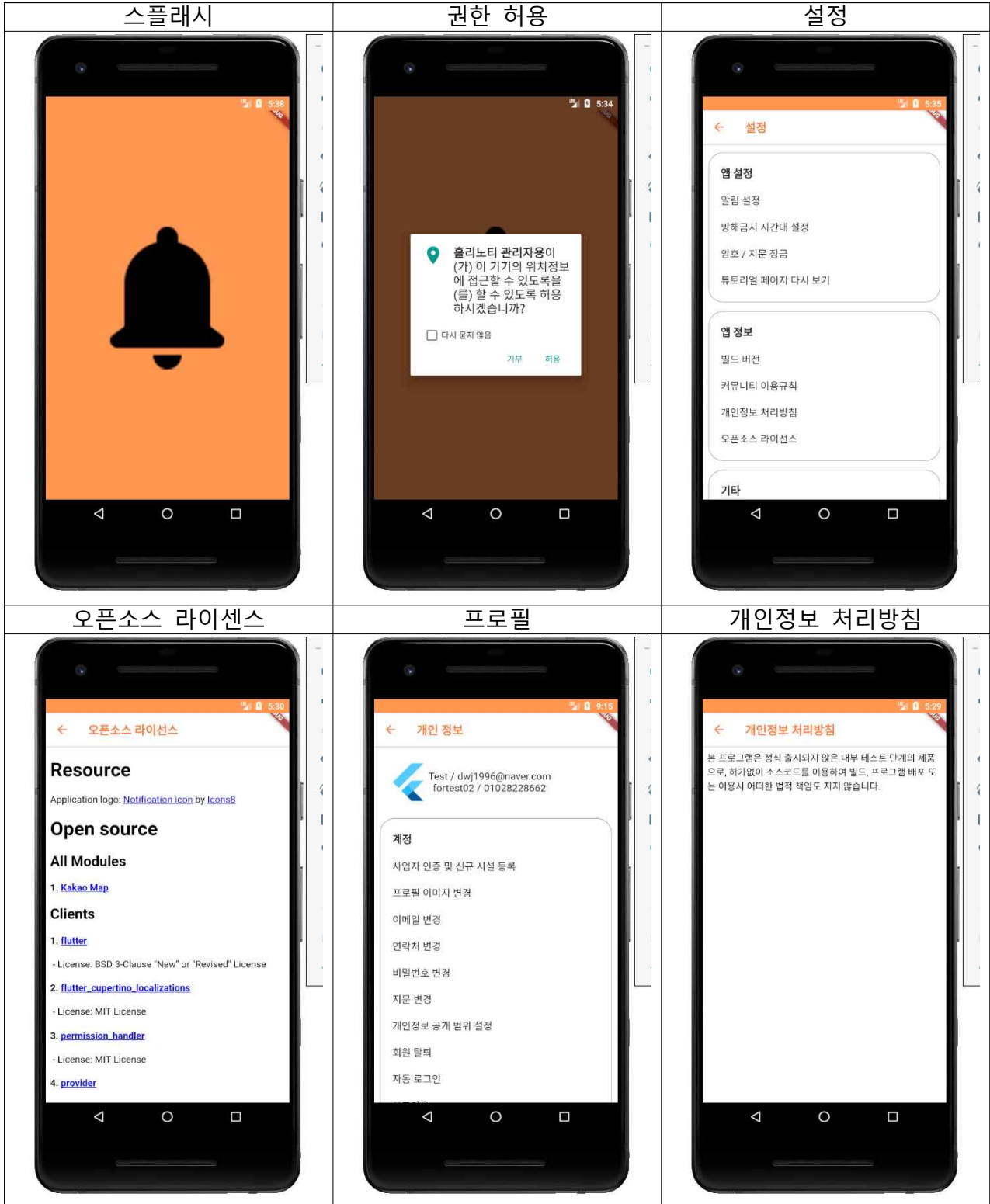
예시)
- 전체 매니저 목록을 GCP서버에서 받아오기: http://holinoti.tk:8080/holinoti/users (GET 메소드)
- 새로운 매니저 추가를 로컬 톰캣 서버에서 하기: http://localhost:8080/holinoti/users/register (POST 메소드)
  * 바디
  {
    "account": "fortest01",
    "password": "password0!",
    "name": "Test",
    "authority": "admin",
    "email": "dwj1996@naver.com",
    "phoneNumber": "01028228661"
  }
- 권한이 필요한 요청은 basic auth로 인증 후에 접근 가능합니다.
```

Holiday Notifier

5. 구현 결과

가. 클라이언트 구현 결과

클라이언트 공통부분

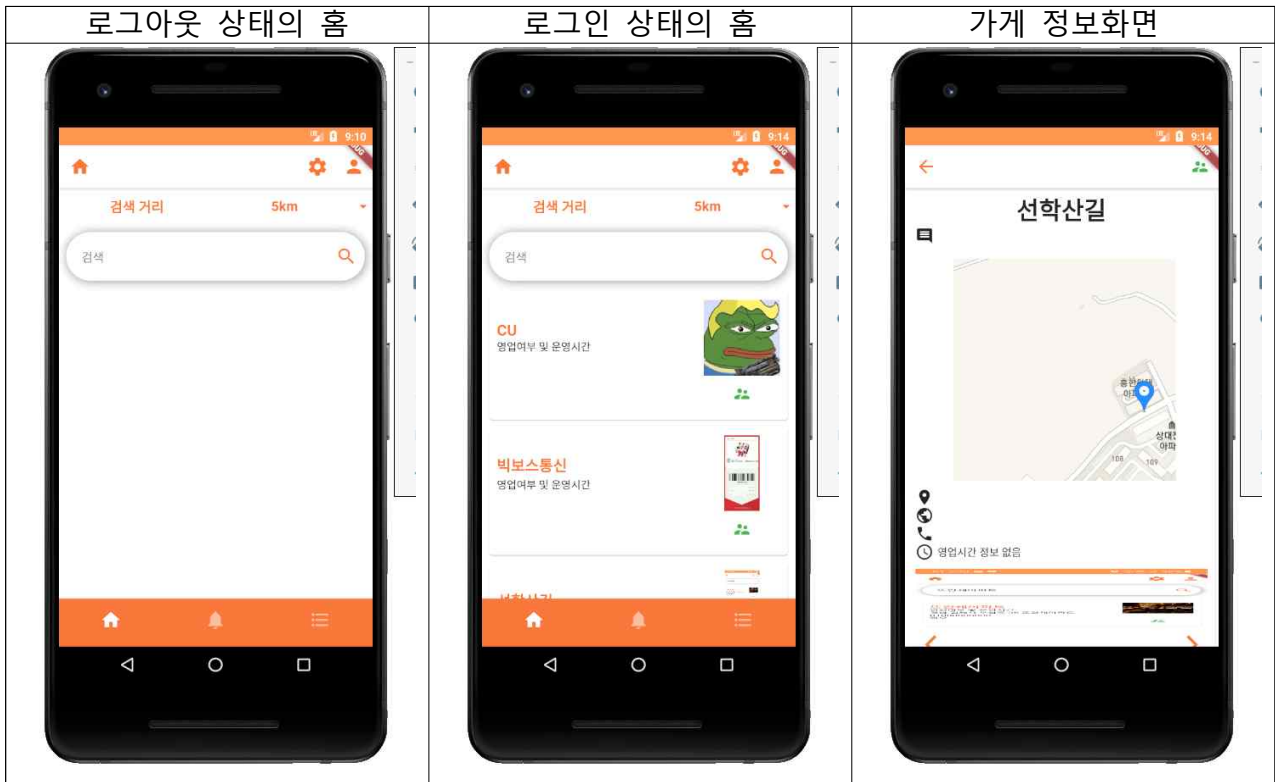


Holiday Notifier

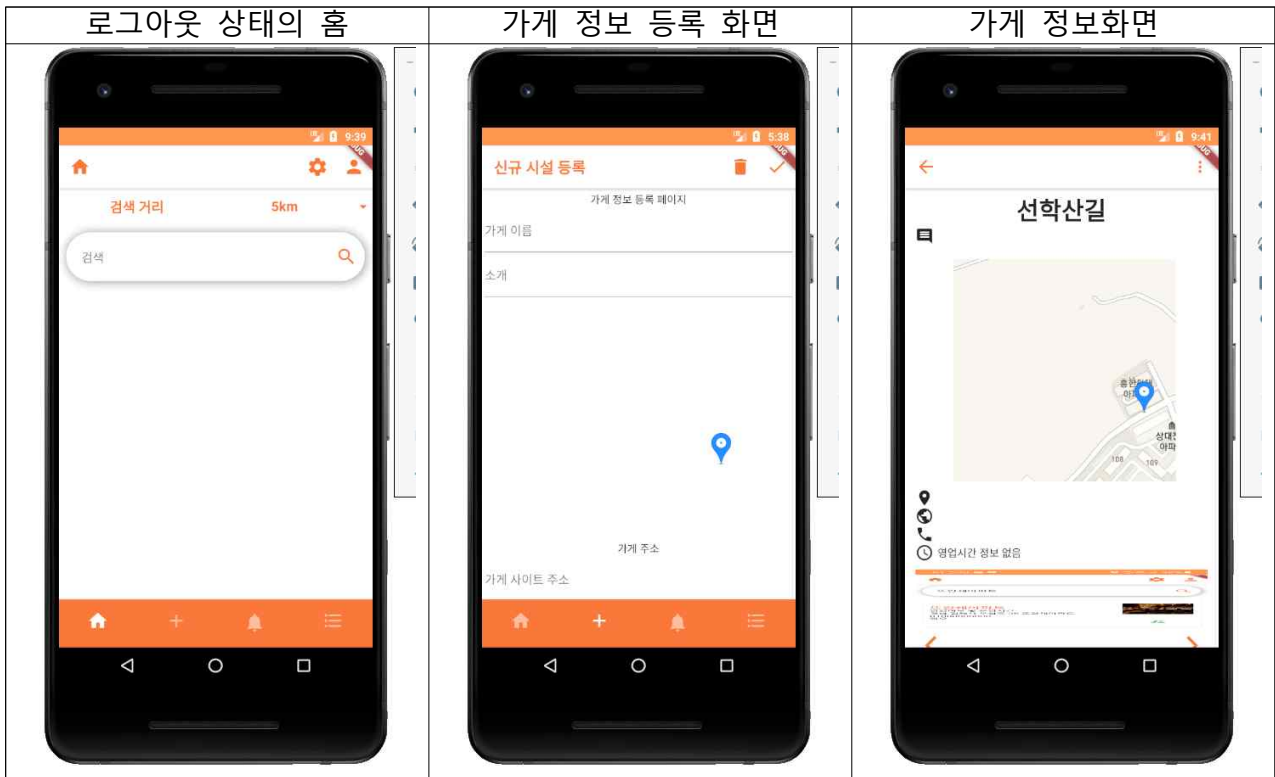
로그인	회원가입	공지사항
튜토리얼	자동 로그인 설정	튜토리얼 재시작

Holiday Notifier

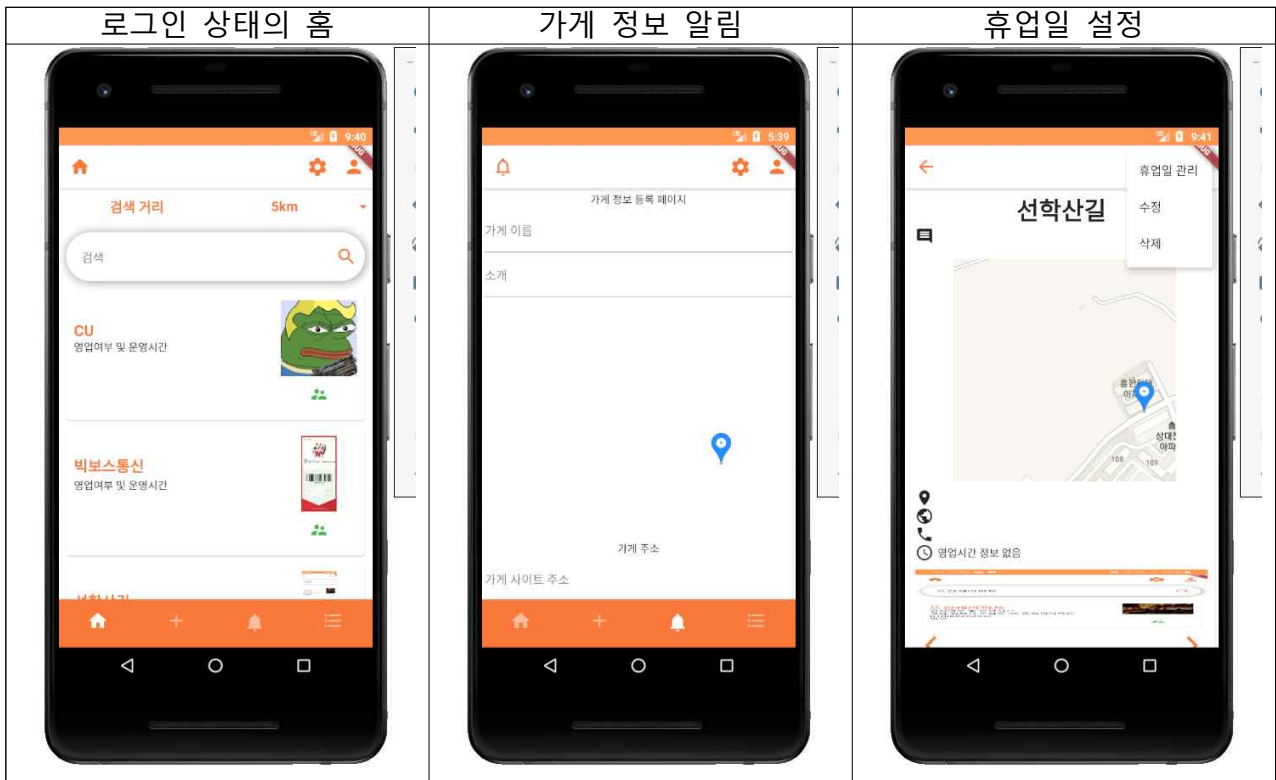
클라이언트 고객 부분



클라이언트 관리자 부분



Holiday Notifier



Holiday Notifier

나. 개인별 작업 내용

◎ 도우진 - 통합, 기능개발, 버그 수정, 프로젝트 관리 등 프로젝트 전반

- 주도적으로 팀장이 되어 팀을 관리하는 법과 백로그 작성, 짧은 주기의 반복 등 애자일 개발 프로세스에 대한 이해를 할 수 있었음.
- MVC 디자인 패턴, 싱글톤 패턴 등 여러 디자인 패턴을 실제 활용하며 학습하였음.
- HTTP 메소드를 이용한 통신에 대한 이해가 증가하였음.

클라이언트 상태관리

Subject를 이용한 Stream 패턴으로 Widget 상태관리

모두 StatelessWidget을 사용: 위젯의 멤버변수가 변하지 않는 정적인 클래스

StreamBuilder를 통해 Stream에 변한 값 수신 시 해당 Widget을 재생성

BLoC 패턴에 잘 어울리고, StatefulWidget보다 퍼포먼스가 좋다는 장점이 있음

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:holiday_notifier_admin/bloc/facility_input_bloc.dart';
import 'package:holiday_notifier_admin/bloc/home_bloc.dart';
import 'package:holiday_notifier_admin/constants/strings.dart' as Strings;
import 'package:holiday_notifier_admin/screens/home.dart';
import 'package:rxdart/rxdart.dart';
import 'package:shared_preferences/shared_preferences.dart';

class TutorialBloc {
  final SharedPreferences _preferences;
  double _currentPage;

  TutorialBloc(this._preferences) {
    _currentPage = 0;
  }

  final _currentPageSubject = PublishSubject<double>();

  get currentPageStream => _currentPageSubject.stream;
  get currentPage => _currentPage;

  void setCurrentPage(ValueNotifier<double> notifier, currentPage) {
    _currentPage = currentPage;
    notifier?.value = _currentPage;
    _currentPageSubject.add(_currentPage);
  }

  moveToHomePage(BuildContext context) async {
    _preferences.setBool(Strings.Preferences.SHOW_TUTORIAL, false);
    Navigator.of(context).pushReplacement(MaterialPageRoute(
      builder: (context) => HomePage(HomeBloc(), FacilityInputBloc())); // MaterialPageRoute
  }

  dispose() {
    _currentPageSubject.close();
    _currentPage = 0;
  }
}
```

Holiday Notifier

MVC, BloC 패턴 적용

	<p>서버</p> <ul style="list-style-type: none"> • Model(Data, VO): 자료의 구조를 담당 • View: 보여지는 페이지. (카카오톡/Static Page를 제외하고는 클라이언트에게 위임) • Controller: HTTP 메소드의 컨트롤러 • Service: 비즈니스 로직을 담당 • Repository(DAO): DB의 접근을 담당 • Constant: 상수 리소스 • Util: 범용적으로 응용하는 로직 		<p>클라이언트</p> <ul style="list-style-type: none"> • Data(Model): 자료의 구조를 담당 • Screen(View): 보여지는 페이지 • BloC: 비즈니스 로직을 담당하는 컴포넌트 • Constant: 상수 리소스 • Util: 범용적으로 응용하는 로직
--	---	--	---

클라이언트 데이터 모델

Default Value를 통한 Null 관련 Exception 예방
JSON 통신에 응용하기 쉽도록 관련 메소드 작성

```

class Facility {
  int code;
  String name;
  String address;
  String phoneNumber;
  String title;
  String comment;
  String openingInfo;
  double x;
  double y;

  /// json -> 객체로 변환
  List<FacilityImage> facilityImages;

  Facility() {
    this.code = Non.Global.NOT_ASSIGNED_ID;
    this.name = '';
    this.address = '';
    this.phoneNumber = '';
    this.title = '';
    this.comment = '';
    this.openingInfo = '';
    this.x = 0;
    this.y = 0;
  }

  List<FacilityImage> ?? = [];

  factory Facility.fromJson(Map<String, dynamic> json) => Facility(
    code: json['code'] as int ?? Non.Global.NOT_ASSIGNED_ID,
    name: json['name'] as String ?? '',
    address: json['address'] as String ?? '',
    phoneNumber: json['phoneNumber'] as String ?? '',
    title: json['title'] as String ?? '',
    comment: json['comment'] as String ?? '',
    openingInfo: json['openingInfo'] as String ?? '',
    x: json['coordinates'] as Map ?? {
      'coordinates': [0, 0],
      'coordinates': ??
    },
    y: json['coordinates'] as Map ?? {
      'coordinates': [0, 0],
      'coordinates': ??
    },
  );

  Map<String, dynamic> toJson() => {
    'code': code,
    'name': name,
    'address': address,
    'phoneNumber': phoneNumber,
    'title': title,
    'comment': comment,
    'openingInfo': openingInfo,
    'facilityImages': [],
    'type': 'Point',
    'coordinates': {
      x:
        y:
      },
    };
  }

  @override
  String toString() =>
    'Facility{code: $code, name: $name, address: $address, phoneNumber: $phoneNumber, title: $title, comment: $comment, openingInfo: $openingInfo, x: $x, y: $y}';
}

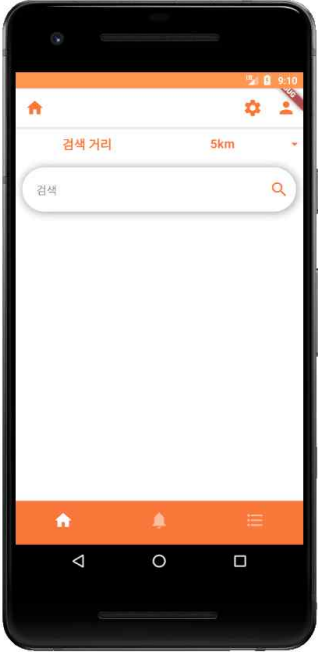




Facility facilityFromJson(String string) =>
  Facility.fromJson(json.decode(string));

String facilityToJson(Facility facility) => json.encode(facility.toJson());
  
```


Holiday Notifier

◎ 강민철 - 고객용 클라이언트

작업 내용

		
<p>홈에서의 각 버튼을 구현하였고 여기서 하단 바의 버튼과 상단 바의 버튼을 담당함</p>	<p>공지사항 화면을 구현하였고 임시 내용을 입력해 놓은 상황으로, 자세한 구현은 아래에 후술</p>	<p>프로필 화면 화면에서는 설정 화면과 동일하게 구현되어 있으며 이미지와 텍스트를 표시하는 부분과 아래에 놓여있는 상자는 설정부분과 동일하게 구현함.</p>
		
<p>설정 화면. 자세한 내용은 아래에 후술.</p>	<p>스플래시 화면으로, 여러 데이터를 로드할 수 있도록 비동기로 동작, 완료되면 메인 홈 화면으로 이동합니다.</p>	

Holiday Notifier

아래는 설정 화면 구현 코드로, Scaffold의 AppBar에서 상단 바를 형성하고 Text로 내용을, body 부분에서 각각의 화면에 표시될 내용을 표시함. AppSetting이나 AppInfo와 같은 함수들은 사용자 정의 함수로 구현되어 있으며 각 부분은 MenuContent 함수로 구현되어 있음. MenuContent 함수에서의 파라미터는 Text와 넘어가는 다른 페이지의 함수로 되어있고 예시 부분에서는 튜토리얼 활성화를 하는 부분과 아직 기능이 없는 알림 설정 등으로 되어있음. 여기서 Text 부분은 String 파일에 정적으로 선언됨.

```
class SettingsPage extends StatelessWidget {
  final SettingsBloc _settingsBloc;

  SettingsPage(this._settingsBloc);

  @override
  Widget build(BuildContext context) => Scaffold(
    appBar: AppBar(
      backgroundColor: Themes.Colors.WHITE,
      title: const Text(
        Strings.SettingPage.SETTINGS,
        style: TextStyle(
          color: Themes.Colors.ORANGE,
          fontWeight: FontWeight.bold,
        ), // TextStyle
      ), // Text
      iconTheme: IconThemeData(size: 28, color: Themes.Colors.ORANGE),
    ), // AppBar
    body: SingleChildScrollView(
      child: Container(
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            AppSetting(_settingsBloc),
            AppInfo(_settingsBloc),
            EtcSetting(_settingsBloc),
          ], // <Widget>[]
        ), // Column
      ), // Container
    ), // SingleChildScrollView
  ); // Scaffold
}
```

```
class AppSetting extends StatelessWidget {
  final SettingsBloc _settingsBloc;

  AppSetting(this._settingsBloc);

  @override
  Widget build(BuildContext context) => MenuBlock(
    title: MenuTitle(Strings.SettingPage.APP_SETTING),
    children: <Widget>[
      MenuContent(Strings.SettingPage.ALARM_SETTING),
      MenuContent(Strings.SettingPage.SLEEP_TIME_SETTING),
      MenuContent(Strings.SettingPage.PASSWORD_FINGERPRINT_LOCK),
      MenuContent(
        Strings.SettingPage.SHOW_TUTORIAL,
        onTap: () => AppDialog(context).showYesNoDialog(
          Strings.SettingPage.SHOW_TUTORIAL_DIALOG_YES_NO,
          onConfirm: _settingsBloc.initTutorial,
        ),
      ), // MenuContent
    ], // <Widget>[]
  ); // MenuBlock
}
```

Holiday Notifier

아래는 공지사항 구현 코드로 각 화면에 표시할 내용을 호출하는 상황으로, 내용 부분은 Container를 이용해서 씌워놓고 내부에서는 Text를 이용하여 글자를 표시함. InkWell은 제스처를 감지하는 이벤트가 있는 위젯으로, Tap 하면 onTap 메소드가 호출되어 다른 화면으로 넘어가거나 다른 내용을 호출 할 수 있음. 예시에서는 snackbar형식으로 호출하고 있음.

```
class NoticeColumn extends StatelessWidget {
  @override
  Widget build(BuildContext context) => Container(
    child: NoticeList(),
  ); // Container
}

class NoticeList extends StatelessWidget {
  final size = [];

  Widget build(BuildContext context) => Column(
    mainAxisAlignment: MainAxisAlignment.min,
    crossAxisAlignment: CrossAxisAlignment.start,
    children: <Widget>[
      NoticeListContent(),
    ], // <Widget>[]
  ); // Column
}

class NoticeCard extends StatelessWidget {
  @override
  Widget build(BuildContext context) => Card(
    child: Container(
      child: Column(
        children: <Widget>[
          Text("Title", style: GlobalPage.bLockTitle),
          Text("data", style: GlobalPage.bLockContents),
        ], // <Widget>[]
      ), // Column
    ), // Container
  ); // Card
}

class NoticeListContent extends StatelessWidget {
  @override
  Widget build(BuildContext context) => InkWell(
    child: Container(
      padding: EdgeInsets.all(8),
      child: Column(
        mainAxisAlignment: MainAxisAlignment.start,
        crossAxisAlignment: CrossAxisAlignment.start,
        children: <Widget>[
          Text("Title",
            textAlign: TextAlign.left,
            style: Themes.GlobalPage.bLockTitle), //
          Text("Content",
            textAlign: TextAlign.left,
            style: Themes.GlobalPage.bLockContents), //
          Row(
            mainAxisAlignment: MainAxisAlignment.end,
            crossAxisAlignment: CrossAxisAlignment.end,
            children: <Widget>[
              Text("date", textAlign: TextAlign.right),
            ], // <Widget>[]
          ), // Row
        ], // <Widget>[]
      ), // Column
    ), // Container
    onTap: () {
      Scaffold.of(context).showSnackBar(SnackBar(
        content: NoticeCard(),
      )); // SnackBar
    },
  ); // InkWell
}
```

호출

호출

호출

호출

내용 구성

버튼 누를 때 함수 호출

- 전반적으로 플러터 프레임워크와 내부의 위젯들에 관한 공부, 플러터에서 위젯이 어떻게 표시되는지에 관한 공부를 할 수 있었음.

Holiday Notifier

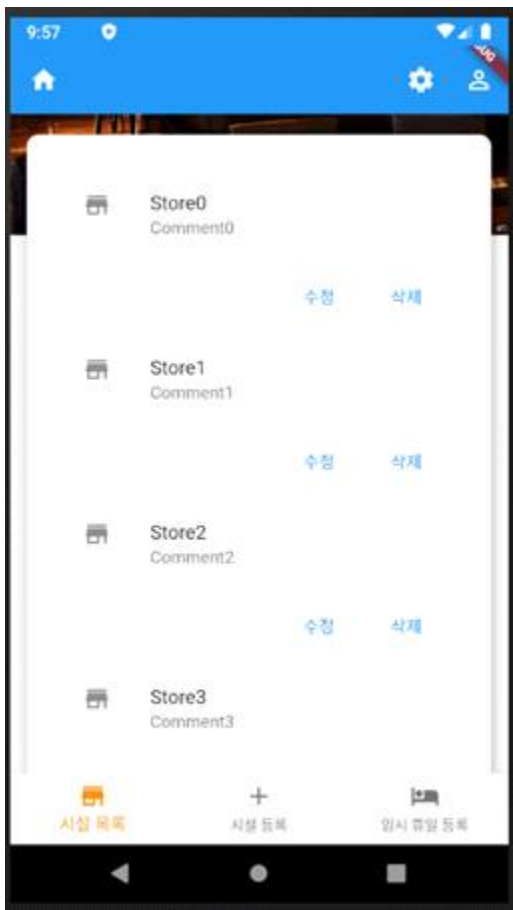
● 김범승 - 관리자용 클라이언트

UI 제작



로그인 이후 메인 화면 구현

초기 구현



최종 구현



Holiday Notifier

```
class FacilitiesListCard extends StatelessWidget {
  final double appBarHeight;

  FacilitiesListCard(this.appBarHeight);

  List dummyFacilities =
    List.generate(4, (i) => {'name': 'Store$i', 'comment': 'Comment$i'});

  @override
  Widget build(BuildContext context) => CenterCard(
    appBarHeight: appBarHeight,
    child: Column(
      children: (dummyFacilities
        .map((facility) => Column(
          children: <Widget>[
            ListTile(
              leading: Icon(Icons.store),
              title: Text(facility['name']),
              subtitle: Text(facility['comment']),
            ),
            ButtonBar(
              children: <Widget>[
                FlatButton(
                  child: const Text('수정'),
                  onPressed: () {},
                ),
                FlatButton(
                  child: const Text('삭제'),
                  onPressed: () {},
                ),
              ],
            ),
          ],
        ))
      ),
    ),
  ).toList());
}
```


Holiday Notifier

알람기능 및 알람 카드

```
import 'package:flutter/material.dart';
import 'package:flutter_local_notifications/flutter_local_notifications.dart';
import 'package:flutter_localizations/flutter_localizations.dart';

class Notification{

  Future launcher() async {
    WidgetsFlutterBinding.ensureInitialized();
    var initAndroidSetting = AndroidInitializationSettings('@mipmap/ic_launcher');
    var initIosSetting = IOSInitializationSettings();
    var initSetting = InitializationSettings(initAndroidSetting, initIosSetting);
    await FlutterLocalNotificationsPlugin().initialize(initSetting);
  }

  Future showNotification() async {
    var android = AndroidNotificationDetails(
      'channelId', 'channelName', 'channelDescription');
    var iOS = IOSNotificationDetails();
    var platform = NotificationDetails(android, iOS);

    await FlutterLocalNotificationsPlugin().show(0, 'title', 'body', platform);
  }

  Future showNotificationSchedule() async {
    var android = AndroidNotificationDetails(
      'channelId', 'channelName', 'channelDescription');
    var iOS = IOSNotificationDetails();
    var platform = NotificationDetails(android, iOS);

    await FlutterLocalNotificationsPlugin().schedule(0, 'title', 'body', DateTime.parse('2020-05-20 16:53:00'), platform);
  }
}
```

Nodejs 임시서버를 이용한 카카오맵

```
var express = require('express');
var app = express();

app.use(express.json());

var MY_KEY = "f0f5e376181343a0b753e08022a9f3c";

app.get('/kaka/lat/long', (req, res) => {
  var lat = req.params.lat;
  var long = req.params.long;
  res.send('<!--CTYPE html-->
<html>
<head><meta charset="utf-8"/><title>Kakao 지도 시각화</title></head>
<body>
<div id="map" style="width:100%;height:400px;"></div>
<script type="text/javascript" src="//dapi.kakao.com/v2/maps/sdk.js?appkey=${MY_KEY}"></script>
<script>
function displayMarker(locPosition, message) {
  var marker = new kakao.maps.Marker({ map: map, position: locPosition });
  var infoContent = message, isMoveable = true;
  var infoWindow = new kakao.maps.InfoWindow({ content : infoContent, removable : isMoveable });
  infoWindow.open(map, marker);
  map.setCenter(locPosition);
}

var container = document.getElementById("map");
var options = {center: new kakao.maps.LatLng(37.5, 127.0), level: 3};
var map = new kakao.maps.Map(container, options);
var locPosition = new kakao.maps.LatLng(37.5, 127.0);
message = "div style="padding:5px">You!</div>;
displayMarker(locPosition, message);
</script>
<script> function helloJams() { console.log("hello"); jams.postMessage("hello"); } </script>
<div style=" position: relative; height: 200px; text-align: center; background-color:#ccc;" >
<div style=" margin: 0; position: absolute; top: 50%; left: 50%; transform: translate(-50%, -50%); ">
<button type="button" onclick="helloJams()">BUTTON/button
</div>
</div>
</body>
</html>";
});

app.listen(3000, () => { console.log("run"); });
```

```
import 'package:flutter/cupertino.dart';
import 'package:flutter/material.dart';
import 'package:flutter_webview_plugin/flutter_webview_plugin.dart';

class Maps extends StatelessWidget{

  @override
  Widget build(BuildContext context){
    return WebViewScaffold(
      appBar: AppBar(
        title: Text("임시 카카오맵"),
      ),
      url: 'http://(IP주소):port/web',
      withJavaScript: true,
      javascriptChannels: Set.from([
        JavaScriptChannel(
          name:"jams",
          onMessageReceived: (JavascriptMessage result){
            print("message ${result.message}");
          }
        )
      ]),
    );
  }
}
```

앱 로고 제작



- 의사소통의 중요성에 대해 배울 수 있었음.
- Flutter 프레임워크와 Flutter Notification 라이브러리를 사용하며 학습하였음.
- Node.js와 웹 뷰 연결에 관해 배웠음.

Holiday Notifier

주요 기능

(1) 위치기반 검색

Facility Entity의 컬럼

```
@JsonSerialize(using = GeometrySerializer.class)
@JsonDeserialize(contentUsing = GeometryDeserializer.class)
private Point coordinates;
```

위치기반 검색 REST API GET요청 메소드

```
@RequestMapping(path = PathString.X_PATH + "{x}" + PathString.Y_PATH + "{y}" +
    PathString.DISTANCE_PATH + "{distanceM}", method = RequestMethod.GET)
public List<Facility> getFacilitiesByCoordinates(@PathVariable("x") double x, @PathVariable("y") double y,
    @PathVariable("distanceM") int side) {
    return facilityService.queryByDistance(x, y, side);
}

@RequestMapping(path = PathString.X_PATH + "{x}" + PathString.Y_PATH + "{y}" +
    PathString.DISTANCE_PATH + "{distanceM}" + PathString.NAME_PATH + "{name}", method = RequestMethod.GET)
public List<Facility> getFacilitiesByCoordinatesAndName(@PathVariable("x") double x, @PathVariable("y") double y,
    @PathVariable("distanceM") int side, @PathVariable("name") String nam) {
    return facilityService.queryByName(x, y, side, nam);
}
```

Facility의 리포지토리 인터페이스의 메소드

```
@Query(value = "CALL DISTANCE(:lon, :lat, :side)", nativeQuery = true)
List<Facility> findAllByCoordinates(@Param("lon") double x, @Param("lat") double y, @Param("side") int side);

@Query(value = "CALL NAMEDISTANCE(:lon, :lat, :side, :nam)", nativeQuery = true)
List<Facility> findAllByCoordinatesAndName(@Param("lon") double x, @Param("lat") double y, @Param("side") int side, @Param("nam") String nam);
```

서버에서 모든 시설을 불러와 피타고라스 방식으로 거리를 재어 보내주는 방식이 아닌 서버 자원을 최소한으로 사용하기 위해 MySQL의 공간 인덱스가 적합하다고 판단, 이를 공부하여 정현석 학우와 온라인 음성 회의를 통해 협의·협력하여 구현. 정현석 학우와 컨트롤러 틀과 프로시저를 작성하고 버그 해결과 의존성을 추가하면서 세부사항 구현.

(2) 이미지 업로드

REST API POST 요청 이미지 추가 컨트롤러 메소드

```
@RequestMapping(method = RequestMethod.POST, consumes = MediaType.MULTIPART_FORM_DATA_VALUE)
public ResponseEntity<?> addFacilityImage(@RequestParam("file") MultipartFile uploadFile,
    @RequestParam("facility_code") int facilityCode) throws Exception {
    if (!userService.isAccessible(facilityCode))
        throw new Exception("Prohibited: Low Grade Role"); // 유저 접근제한
    FacilityImage facilityImage = new FacilityImage();

    try {
        facilityImageService.upload(facilityImage, uploadFile, facilityCode);
    } catch (Exception e) {
        e.printStackTrace();
        return new ResponseEntity<>(e.getMessage(), HttpStatus.BAD_REQUEST);
    }

    URI createdURI = linkTo(FacilityImageController.class).slash(facilityImage.getId()).toUri();
    return ResponseEntity.created(createdURI).body(facilityImage);
}
```

Holiday Notifier

```
public void upload(FacilityImage facilityImage, MultipartFile uploadFile, int facilityCode) throws Exception {  
    //Windows !  
    String path = "C:" + Strings.PathString.FACILITY_IMAGE_FILE_PATH + "/" + facilityCode;  
    //Linux !  
    String path = Strings.PathString.FACILITY_IMAGE_FILE_PATH + "/" + facilityCode;  
    String fileName = "";  
  
    fileName = uploadFile.getOriginalFilename();  
    System.out.println(fileName + " was Uploaded.");  
    if (fileName == null || fileName.isEmpty())  
        throw new Exception("empty file");  
    if (fileName.matches( regex: ".jpg" ) )  
        throw new Exception("wrong mime type");  
    try {  
        facilityImage.setFileName("");  
        facilityImage.setFacilityCode(facilityCode);  
        facilityImageRepository.save(facilityImage);  
        fileName = System.currentTimeMillis() + "_" + facilityImage.getId() + ".jpg";  
        facilityImage.setFileName(fileName);  
        facilityImageRepository.save(facilityImage);  
  
        File directory = new File(path);  
        if (!directory.exists()) {  
            if (directory.mkdir()) {  
                System.out.println("Directory is created!");  
            } else {  
                System.out.println("Failed to create directory!");  
            }  
        }  
  
        System.out.println("UtilFile fileUpload fileName : " + fileName);  
        System.out.println("UtilFile fileUpload uploadPath : " + path);  
  
        File file = new File( pathname: path + "/" + fileName);  
        uploadFile.transferTo(file);  
    } catch (Exception e) {  
        facilityImageRepository.deleteById(facilityImage.getId());  
        throw e;  
    }  
}
```

이미지 파일 유효성 검사

Id를 폴더명으로 생성

파일 업로드

서버의 이미지 저장 경로를 URL로 접근하도록 설정.

```
@Configuration  
public class WebMvcConfig implements WebMvcConfigurer {  
    @Override  
    public void addResourceHandlers(ResourceHandlerRegistry registry) {  
        registry.addResourceHandler( .pathPatterns: Strings.PathString.FACILITIES_IMAGES_FULL_PATH + "/*" )  
            .addResourceLocations("file:" + Strings.PathString.FACILITY_IMAGE_FILE_PATH + "/");  
    }  
}
```

스프링부트의 MultipartFile의 매뉴얼을 읽고 리눅스와 윈도우간의 통합된 파일 IO 기능을 수행시키기 위해 CommonIO 의존성을 추가했음. 프로덕트 환경에선 구글 클라우드 플랫폼의 리눅스 서버를 사용하므로 해당 부분 코드를 작성함.

Holiday Notifier

(3) 푸시 알림

요청이 들어온 객체를 JSON 형태로 변경

```

@RequestMapping(path = Strings.PathString.FACILITY_CODE_PATH + "{facilityCode}", method = RequestMethod.POST)
public @ResponseBody
ResponseEntity<String> send(@PathVariable("facilityCode") int facilityCode, @RequestBody Notification notification) throws Exception {
    if (!userService.isAccessible(facilityCode))
        throw new Exception("Forbidden: low grade role");
    String notifications = notificationService.notification3use(facilityCode, notification.getTitle(), notification.getBody());
    CompletableFuture<String> pushNotification = notificationService.send(notifications);
}
    
```

서비스 부분

```

public String notification3use(int facilityCode, String notificationTitle, String notificationBody) throws Exception {
    LocalDate localDate = LocalDate.now();

    List<Integer> userIds = relationAFService.userIdByFacilityCode(facilityCode);
    if (userIds.isEmpty()) throw new Exception("Not found");
    List<String> tokenList = userService.getDeviceTokensByIds(userIds);
    tokenList.removeIf(Objects::isNull);

    JSONObject body = new JSONObject();
    JSONArray array = new JSONArray();

    for (String s : tokenList) {
        array.put(s);
    }

    body.put("registration_ids", array);

    Map<String, String> notification = new HashMap<>();
    notification.put("title", notificationTitle);
    notification.put("body", notificationBody);

    body.put("notification", notification);

    System.out.println("Notification has been registered: " + body.toString());

    return body.toString();
}
    
```

요청을 REST API 템플릿에 맞춰 작성 후 비동기로 FireBase에 POST 전송.

```

@Async
public CompletableFuture<String> send(String content) {
    RestTemplate restTemplate = new RestTemplate();

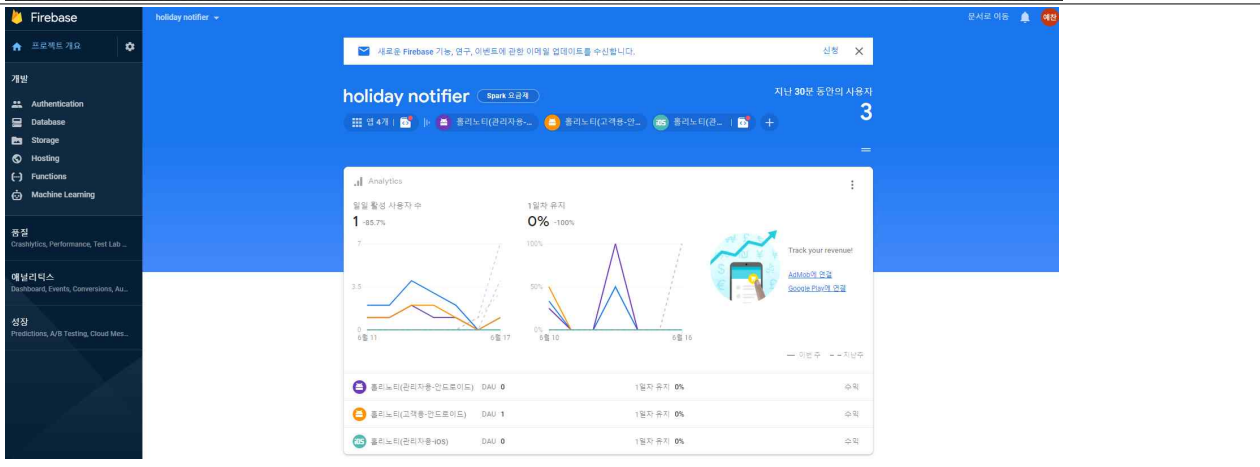
    ArrayList<ClientHttpRequestInterceptor> interceptors = new ArrayList<>();

    interceptors.add(new HeaderRequestInterceptor("Authorization", "key=" + Strings.Api.API_KEY_FCM_SERVER));
    interceptors.add(new HeaderRequestInterceptor("Content-Type", "application/json; charset=utf-8"));
    restTemplate.setInterceptors(interceptors);

    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(new MediaType("application", "json", StandardCharsets.UTF_8));
    HttpEntity<String> entity = new HttpEntity<>(content, headers);

    String firebaseResponse = restTemplate.postForObject(Strings.Api.API_URL_FCM_SERVER, entity, String.class);

    return CompletableFuture.completedFuture(firebaseResponse);
}
    
```



Holiday Notifier

구글의 파이어베이스 서버를 사용하기 위해 해당 프로젝트 생성과 푸시알림 전송 매뉴얼을 학습하고 데이터베이스의 유저 속성에 device_token을 추가함.
이를 통해 사용자 리스트를 해당 device_token 리스트로 변환하고 템플릿에 맞춰 웹서버에서 파이어베이스로 POST 요청을 보내는 것임.

(4) 그 외 작업 내용

서버 구조에 서술된 요청 응답을 위한 각종 Entity, Controller 작성,
Repository 인터페이스 작성,
Multipartfile 업로드를 위한 의존성 추가,
회원가입 유효성 검사, 정규식을 통한 비밀번호 검증,
메소드 참조를 통한 필드 리스트 변환 최적화,
해시 셋을 통한 리스트 중복제거,
Properties에 Database Dialect 공간 인덱스 MySQL 지정,
파일 업로드 용량 제한 설정 등.

Holiday Notifier

● 정현석 – DB 및 서버 보조

작업 내용

DB 구현(테이블 생성 및 속성 수정 또는 추가, Facility 테이블 coordinates 컬럼 SPATIAL 인덱스 생성)

-Change coordinates column to geometry into point on facility table

- Alter coordinates column to have not null, spatial index

- Add coordinates column on facility table

- Create facility_image table
- Add phone_number column on manager, facility table

SPATIAL SPATIAL_COORD(`coordinates`)

(FULLTEXT 인덱스를 추가하였지만, 시설검색 부분에서 사용하지 못한 부분이 아쉬움)

URL 추가

```
public static final String X_PATH = "/x=";
public static final String X_PATH_REGEX = X_PATH + REGEXP;

public static final String Y_PATH = "/y=";
public static final String Y_PATH_REGEX = Y_PATH + REGEXP;

public static final String DISTANCE_PATH = "/distance_m=";
public static final String DISTANCE_PATH_REGEX = DISTANCE_PATH + REGEXP;

public static final String NAME_PATH = "/name=";
public static final String NAME_PATH_REGEX = DISTANCE_PATH + REGEXP;
```

위치기반 검색, 위치기반 시설검색 프로시저

```
DELIMITER //
CREATE PROCEDURE DISTANCE(lon double, lat double, side int)
BEGIN
SET @lon = lon;
SET @lat = lat;
SET @MBR_length = side;
SET @lon_diff = @MBR_length / 2 / ST_DISTANCE_SPHERE(POINT(@lon, @lat), POINT(@lon + IF(@lon < 0, 1, -1), @lat));
SET @lat_diff = @MBR_length / 2 / ST_DISTANCE_SPHERE(POINT(@lon, @lat), POINT(@lon, @lat + IF(@lat < 0, 1, -1)));
SET @diagonal = CONCAT('LINESTRING(', @lon - IF(@lon < 0, 1, -1) * @lon_diff, ',', @lat - IF(@lon < 0, 1, -1) * @lat_diff, ',', @lon + IF(@lon < 0, 1, -1) * @lon_diff, ',', @lat + IF(@lon < 0, 1, -1) * @lat_diff, ');');
SELECT *
FROM facility FORCE INDEX FOR JOIN ("SPATIAL_COORD")
WHERE MBRCONTAINS(ST_LINESTRINGFROMTEXT(@diagonal), coordinates);
END
//
DELIMITER ;
DELIMITER //
```

입력 단위: M

좌표로 이루어진 도형을 감싸는 사각형인 MBR을 만들

시설 테이블에서 SPATIAL 인덱스로 내 위치에서 MBR 안에 포함되는 좌표를 필터링

현재 위도와 경도를 중심으로 받은 반경 거리를 한 변으로 하는 MBR 사각형을 만들기 위해 동, 서쪽으로 떨어진 위도, 남, 북쪽으로 떨어지는 경도의 차이 값으로 대각선을 그리고, 이 대각선을 기반으로 한 MBR을 만들어 이를 각 시설과 비교하여 MBR의 내부의 위도, 경도에 있는 좌표를 MBRCONTAINS에서 필터링함.

Holiday Notifier

```
DELIMITER //
CREATE PROCEDURE NAMEDISTANCE(IN lon double, IN lat double, IN side int, IN nam varchar(255))
BEGIN
SET @lon = lon;
SET @lat = lat;
SET @nam = nam;

SET @A_nam = REPLACE(@nam, ',', '');
SET @R_nam = CONCAT('S', @A_nam, 'S');

SET @RBR_length = side;

SET @lon_diff = @RBR_length / 2 / ST_DISTANCE_SPHERE(POINT(@lon, @lat), POINT(@lon + IF(@lon < 0, 1, -1), @lat));
SET @lat_diff = @RBR_length / 2 / ST_DISTANCE_SPHERE(POINT(@lon, @lat), POINT(@lon, @lat + IF(@lat < 0, 1, -1)));

SET @diagonal = CONCAT('LINESTRING(', @lon - IF(@lon < 0, 1, -1) * @lon_diff, ',', @lat - IF(@lon < 0, 1, -1) * @lat_diff, ',', @lon +
IF(@lon < 0, 1, -1) * @lon_diff, ',', @lat + IF(@lon < 0, 1, -1) * @lat_diff, ')');

SELECT *
FROM facility FORCE INDEX FOR JOIN ('SPATIAL_COORD')
WHERE HERCONTAINS(ST_LINESTRINGFROMTEXT(@diagonal), coordinates) AND replace(name, ',', '') LIKE @R_nam;
END
//
DELIMITER ;
```

**입력받은 시설명
공백 제거**

**데이터들의 공백 제거 후
검색**

위의 위치기반 검색에 추가로 시설명을 받아 이를 공백 제거하고, 시설 테이블의 시설명의 공백 또한 제거하여 검색하여 포함된 문자열을 기반으로 한 검색된 시설명을 반환.

- 위치기반 검색의 프로시저 부분을 구현하기 위해 DB의 SPATIAL 인덱스 사용법 및 위치기반 검색 관련 쿼리문 작성법, 저장 프로시저 사용법 및 호출 부분을 공부하였음.
- 이를 통해 스프링부트의 MVC패턴에 대한 이해 및 URL VARIABLE에 관한 공부가 되었음.

6. 결론

가. 문제점 분석

해당 프로젝트를 진행하면서 클라이언트 담당 팀원의 능력 부족과 의사소통 오류, 시간 부족 문제로 인해 처음 계획했던 몇몇 기능을 포기하게 되었다. 개발 초기에는 어느 정도 역할을 해주었으나, 시간이 갈수록 많은 바를 다하지 못하거나, 착각하여 잘못 구현하는 일이 발생 하였다. 주기적으로 진행 상황을 체크했음에도 일어난 문제로, 다행히 개발 주기를 짧게 나누어 통합 후 점검을 반복하였기 때문에 빠르게 문제를 발견하고 기능 대부분은 구현할 수 있었다. 팀원의 역량에 비해 프로젝트의 규모를 너무 크게 잡은 것도 문제점으로 보인다.

Holiday Notifier

나. 향후 계획

현재 보안이 매우 취약한 상태(SSL 인증서 없음, 통신 시 패킷 암호화 처리 없음 등)이고 이를 보완해야 한다. 그리고 백로그에 수록한 미완성 기능들의 구현과 실제 사용성 테스트 등이 남아있다. 단기간 내에 개선된 UX와 상업적인 서비스 혹은 고객의 정보를 보관할 수 있는 수준의 신뢰성까지는 확보하지 못할 것으로 판단, 장기적으로 보안과 UX 개선을 진행한다.

다. 기대효과

본 시스템을 이용하여 시설은 이용자들에게 정보제공과 더불어 비정기 휴업으로 인한 고객의 불쾌감을 줄이고, 고객은 해당 시설을 직접 방문하거나 전화하지 않고도 휴업 일정을 파악하여 일정, 계획 변경 등을 시간 낭비 없이 진행할 수 있을 것으로 기대한다.

라. 총론

본 프로젝트를 진행하기에 앞서 여러 자료조사를 진행하였고 그에 따라 필요한 내용이 무엇인지 파악한 이후에 프로젝트를 진행하였으나, 팀의 역량 보다 프로젝트의 크기가 비대해지며 기존에 계획하였던 일부 기능이 구현되지 못했고, 보안 문제로 인해 실제 서비스는 보류하게 되었다. 이 프로젝트를 수행하면서 일부 팀원이 기능들을 구현하지 못하거나 잘못 구현하는 아쉬운 점도 있었고, 코로나 감염병 상황으로 인하여 비대면으로 개발 및 회의를 진행하게 되었고, 여러 강의에서 출석 인정 대체 과제를 내주어 많은 과제로 인해 개발이 더뎠다는 문제도 있었다. 다만 이러한 부정적인 주변 상황에도 불구하고, 프로젝트를 진행하며 여러 문제를 개선·보완하다 보니 효율적인 의사소통 방법을 인지 및 학습할 수 있었다.

7. 참고문헌

플러터 설치 및 튜토리얼, 문서

<https://flutter-ko.dev/docs/get-started/install/windows>

다트 설치 및 문서

<https://dart.dev/>

깃

<https://git-scm.com/book/ko/v2/>

플러터에서의 상태관리(스트림, rxdart), BLoC패턴에 관한 설명

<https://software-creator.tistory.com/13>

MVC 패턴에 관한 설명

<https://opentutorials.org/course/697/3828>

『코드로 배우는 스프링 웹 프로젝트(개정판)』, 남가람북스, 구멍가게 코딩단 지음