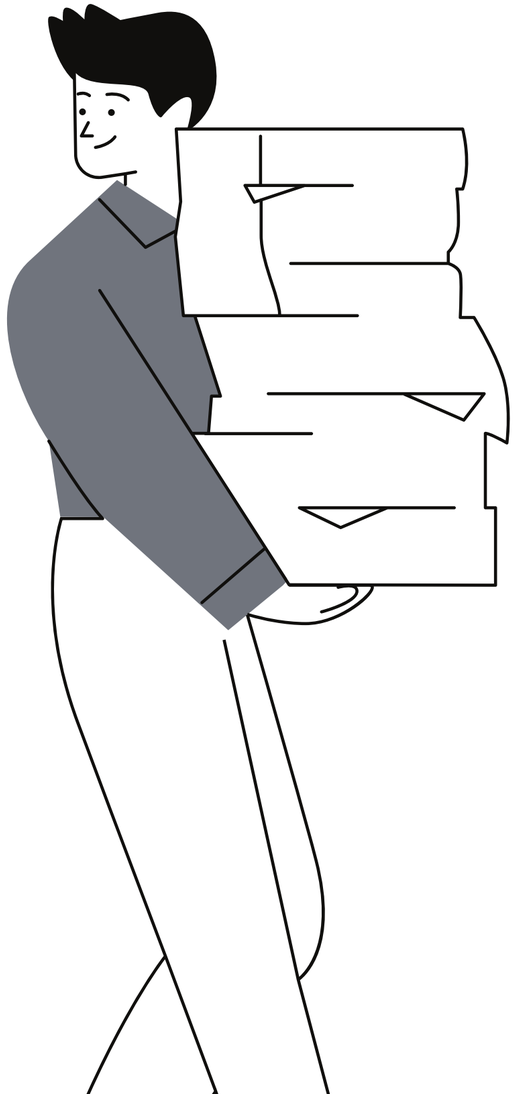
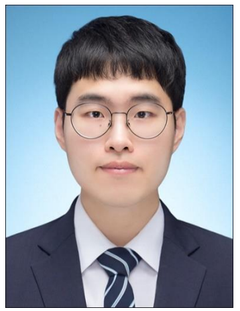
July 4, 2021

Portfolio



July 4, 2021



노력하는 개발자 정준영입니다

정준영 / junyoung Jung

1993.09.18 / 서울특별시

Email : cua0918@naver.com

Github : https://github.com/jjy0918

Blog : https://9327144.tistory.com/

GRADUATION

2012 중산고등학교 졸업
2013 광운대학교 컴퓨터소프트웨어학과 입학
2020 광운대학교 컴퓨터소프트웨어학과 졸업
2021 삼성 청년 SW아카데미

EXPERIENCE

2020 하나금융티아이 입사

AWARDS

2019 졸업작품 전시회 우수상
2021 삼성 청년 소프트웨어 아카데미 성적 우수상
2021 삼성 청년 소프트웨어 아카데미 프로젝트 최우수상

PROJECT

2018 유튜브 뮤직 플레이어
2019 초소형 컴퓨터를 활용한 지능형 신호등
2021 탐방

SKILL

JAVA	90
Spring	90
MySQL	80
C#	95
UNITY	80

PROJECT.1

유튜브 뮤직 플레이어

- 유튜브 뮤직 단점 개선 프로젝트

ABOUT PROJECT

1. 플레이 리스트 개선
2. 음원 정보 확보
3. 비정상적 음원 정상화

유튜브 뮤직의 성장

유튜브 뮤직, 네이버 '바이브' 잡고 '멜론' 흔들러 간다

김임수 기자 | 승인 2020.05.19 15:52

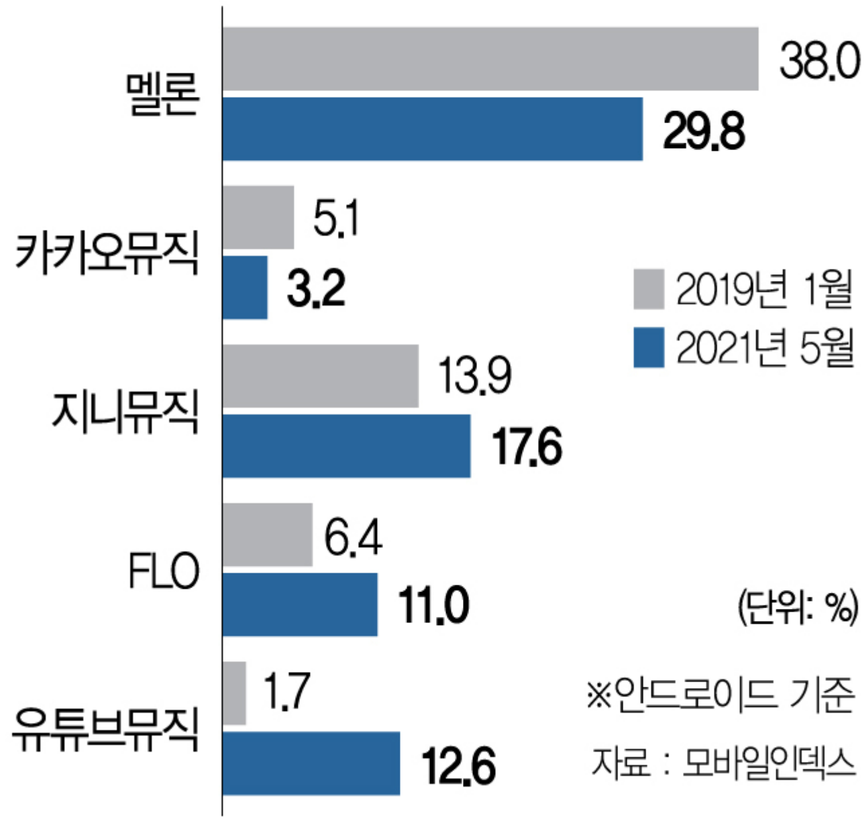
가 가

유튜브 뮤직이 뜬다...음원3사 '초긴장'

신승훈 기자 | 입력 : 2021-05-31 00:05

디지털타임스 | 14면 TOP | 2020.01.20. | 네이버뉴스

진격의 유튜브뮤직... 이용자 1년새 2배 급증



유튜브 뮤직의 단점

1. 플레이 리스트 관리가 힘들다
2. 국내 음원 정보가 부족하다
3. 음원이 정상이 아닐 때가 있다.

July 4, 2021

SKILLS

C#

프로그램 로직 구현

.NET Framework
Windows Form

Windows Form을 이용한
GUI 구현

GitHub

소스 코드 관리 및 형상 관리.
Windows Form과 연계한 웹 연동

PROJECT.1

July 4, 2021

프로그램 구현 개요

01 플레이 리스트 기능

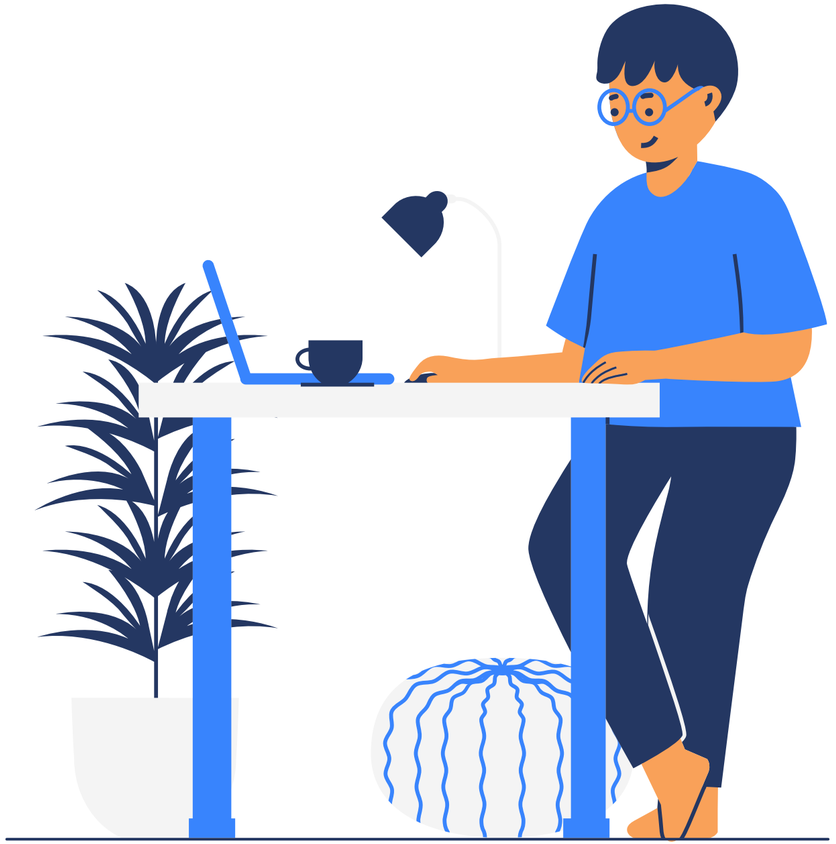
프로그램 실행 시 사용자가 저장했던 플레이 리스트를 정상적으로 불러와야 한다.

02 정확한 음원 정보 검색 기능

사용자가 원하는 정보 검색 시 가수, 앨범 또는 음원을 정확하게 찾을 수 있어야 한다.

03 음원 수정 기능

음원 매핑이 잘못된 경우 사용자가 직접 정확한 음원으로 수정할 수 있어야 한다.



프로그램 구현 내용

01 메인 화면

프로그램 실행 시 사용자가 기존에 저장했던 리스트를 자동으로 불러온다.

프로그램 구현 내용

01 메인 화면

프로그램 실행 시 사용자가 기존에 저장했던 리스트를 자동으로 불러온다.

리스트 정보를 담은 객체를 직렬화 하여 PC에 저장 및 불러오기.

```
public void OpenList()
{

    try
    {
        if(is_first)
        {
            stream = File.Open("MyList.ml", FileMode.OpenOrCreate);
            BinaryFormatter formatter = new BinaryFormatter();
            try
            {
                myList = (MyMusicList)formatter.Deserialize(stream);
                MyMusicList.setObject(myList);
            }
            catch(Exception ea)
            {
                myList = MyMusicList.getObject();
                MessageBox.Show(ea.ToString());
            }

            stream.Close();


            myList.RenewList();
            myList.ArtistList(ArtistLV);
            myList.TitleList(AllMusicLV);
```

```
                NPL = new NowPlayingList("NowPlayingList.npl");

                stream = File.Open(NPL.ListName, FileMode.OpenOrCreate);
                formatter = new BinaryFormatter();
                try
                {
                    NPL = (NowPlayingList)formatter.Deserialize(stream);
                }
                catch(Exception)
                {
                    ;
                }
                NPL.LoadDBtoList(NowPlayingLV);
                stream.Close();




            }
            PrvBT.Enabled = true;
            PlayBT.Enabled = true;
            NextBt.Enabled = true;
            ShuffleBT.Enabled = true;

        }
        catch(Exception ae)
        {

            MessageBox.Show(ae.ToString());
        }
    }
```

프로그램 구현 내용

02 음원 정보 분류

리스트에 저장되어 있는 음원은 모두 가수, 앨범별로 분류되어 있다.

리스트 객체는 프로그램상 하나만 존재하여야 하기 때문에 Singleton 패턴으로 정의하였다.

```
[Serializable]
class MyMusicList
{
    private static MyMusicList thisList;

    List<Artist> MyMusicArtists;

    public List<bool> isTitleChange = new List<bool> { true, true };

    private MyMusicList()
    {
        MyMusicArtists = new List<Artist>();
    }

    public static MyMusicList getObject()
    {
        if (thisList == null)
            thisList = new MyMusicList();
        return thisList;
    }

    public static void setObject(MyMusicList _thisList)
    {

        thisList = _thisList;
    }
```

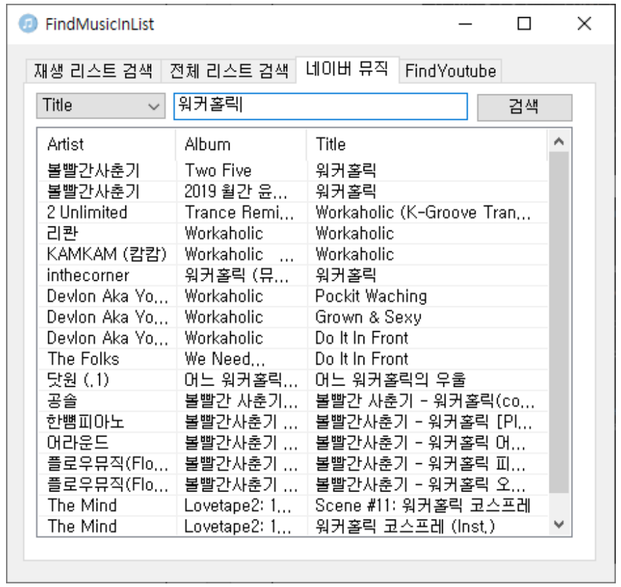
```
    public string getID(string Artist,string Album,string Title)
    {
        Artist a = getArtist(Artist);
        Album ab = a.isContainAlbum(Album);
        return ab.getID(Title);
    }

    public string getNaverNum(string Artist, string Album, string Title)
    {
        Artist a = getArtist(Artist);
        Album ab = a.isContainAlbum(Album);
        return ab.getNaverNum(Title);
    }

    public string getImage(string Artist, string Album)
    {
        Artist a = getArtist(Artist);
        Album ab = a.isContainAlbum(Album);
        return ab.AlbumImage;
    }
```

프로그램 구현 내용

03 음원 정보 검색

유튜브 뮤직의 부정확한 음원 정보 대신,

네이버 뮤직과 멜론의 정확한 음원 정보를 사용하였다.



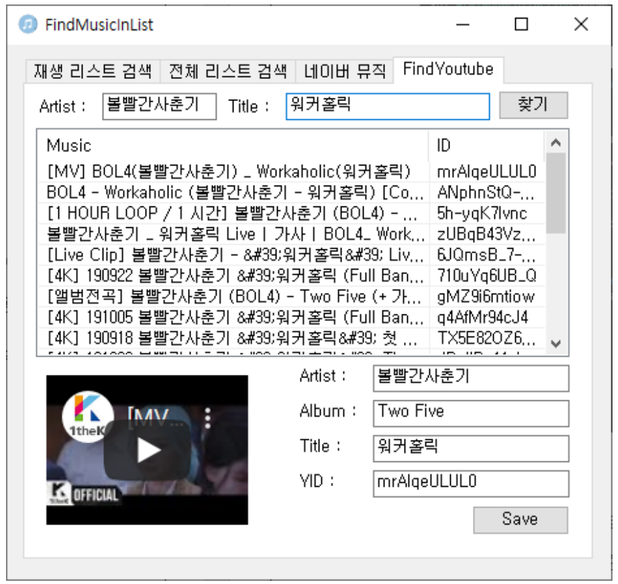

```
foreach (agi.HtmlNode node in nodecol)
{
    try
    {
        inner1 = node.Attributes["href"].Value;

        if(inner1.Contains("?artistId"))
        {
            inner2 = node.Attributes["title"].Value;
            if (inner2.Contains("&amp;"))
                inner2.Replace("&amp;", "");

            artist = inner2;
            NaverMusicSearch.setSearchArtist(inner2,inner1.Substring(inner1.IndexOf("?artist") + 10));
            listViewItem = new ListViewItem(new string[] { artist});
            listViewEx1.Items.Add(listViewItem);

        }
    }
    catch(Exception)
    {
        ;
    }
}
```

프로그램 구현 내용

04 유튜브 뮤직과 음원 매핑

네이버 뮤직으로 찾은 정보를 바탕으로

유튜브 뮤직에서 해당 음원을 검색하여 매핑 및 저장한다.

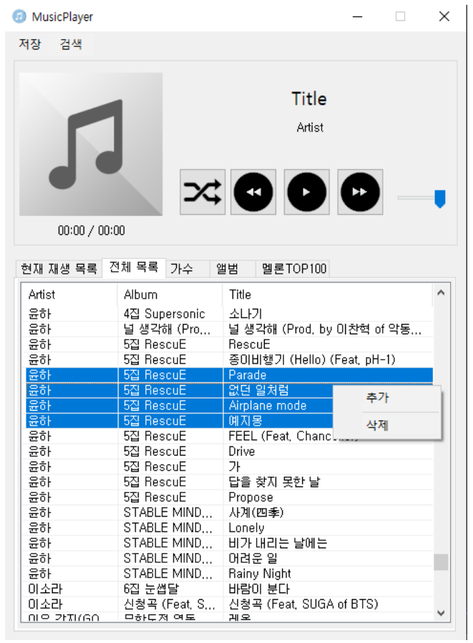


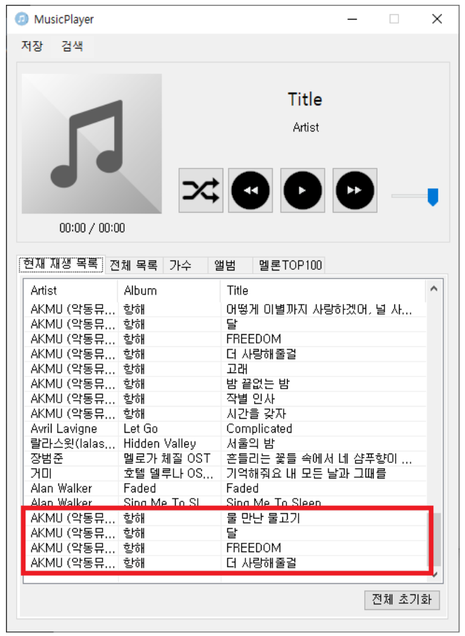

```
var request = youtube.Search.List("snippet");

request.Q = Artist_SearchTB.Text + " " + Title_SearchTB.Text;  // 검색
request.MaxResults = 20;
// Search용 Request 실행
var result = await request.ExecuteAsync();


// Search 결과를 리스트뷰에 담기
foreach (var item in result.Items)
{

    if (item.Id.Kind == "youtube#video")
    {
        try
        {
            string title = item.Snippet.Title;



            YoutubeLV.Items.Add(new ListViewItem(new string[] { title, item.Id.VideoId.ToString() }));
        }

        catch (Exception)
        {
            ;
        }

    }
}

playYoutube(YoutubeLV.Items[0].SubItems[1].Text);
tbYID.Text = YoutubeLV.Items[0].SubItems[1].Text;
```

프로그램 구현 내용

05 플레이 리스트 관리

저장된 리스트에서 재생 목록으로 추가





프로그램 구현 내용

06 음원 재생

재생 목록에 있는 음원 재생 – 웹 페이지와 연동하여 음원 재생

```
private ListPlay()
{
    randomlist = new List<int>();


    myList = MyMusicList.getObject();
    web = new WebBrowser();  //test.html

    path = Application.StartupPath;


    //web.Navigate(path+"₩₩test_.html");

    //web.Navigate("https://output.jsbin.com/qequviz/");
    web.Navigate("https://jjy0918.github.io/yPlay/");

    // web Navigate 대기중
    while (web.ReadyState!=WebBrowserReadyState.Complete)
    {
        Application.DoEvents();
    }


    web.ScriptErrorsSuppressed = true;

    web.ObjectForScripting = this;


    r = new Random();
}


public static ListPlay getObject()
{
    if (My_web == null)
        My_web = new ListPlay();
    return My_web;
}
```

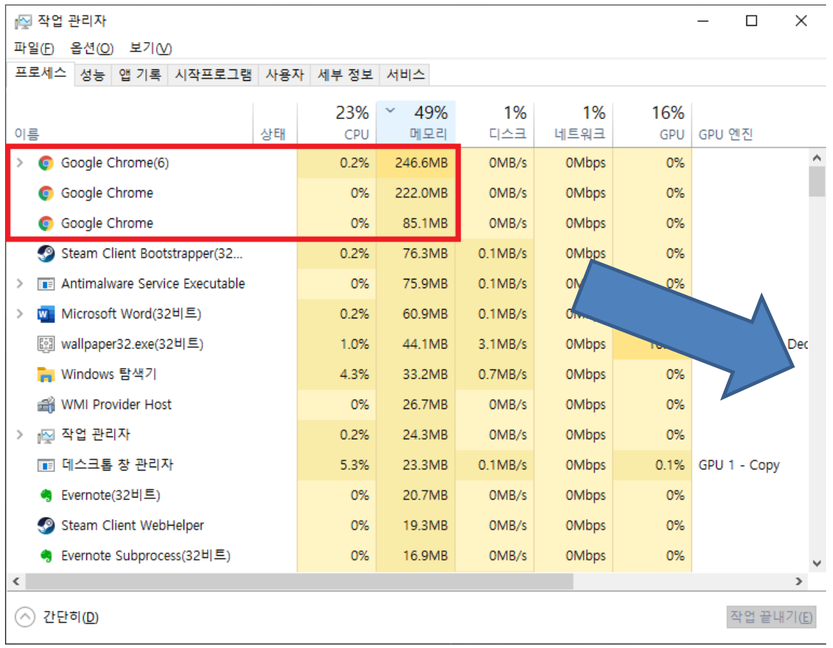
```
public void startVideo()
{
    if (isfirst)
    {
        startMusic(now);
        isfirst = false;
        NextBt.Enabled = true;
        PrvBT.Enabled = true;
    }
    else
        web.Document.InvokeScript("startvideo");

    isStart = true;
    PlayBT.Image = imageList1.Images[1];
    Toolplay.Text = "정지";
}

public void stopVideo()
{
    web.Document.InvokeScript("stopVideo");

    isStart = false;
    PlayBT.Image = imageList1.Images[0];
    Toolplay.Text = "재생";
}
```

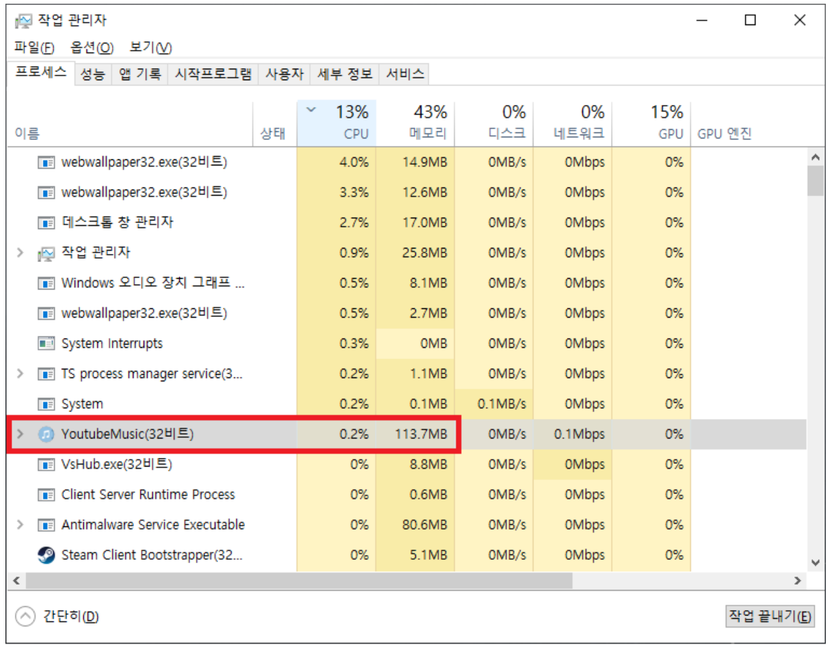
```
function startvideo() {
    var div = document.getElementById("video_slide");
    var iframe = div.getElementsByTagName("iframe")[0].contentWindow;
    iframe.postMessage('{"event":"command","func":"playVideo","args":""}','*');
}

function stopVideo() {
    var div = document.getElementById("video_slide");
    var iframe = div.getElementsByTagName("iframe")[0].contentWindow;
    iframe.postMessage('{"event":"command","func":"pauseVideo","args":""}','*');
}


        function onPlayerStateChange(event) {
            if( event.data == YT.PlayerState.ENDED)
              {
                endMusic();
              }
        }



function endMusic()
{
   window.external.nextMusic();
}
```

프로그램 구현 내용

07 최종 결과

정상 음원 실행 및 메모리 낭비 방지





어려웠던 점 및 해결 방안

01 음원 저장 방법

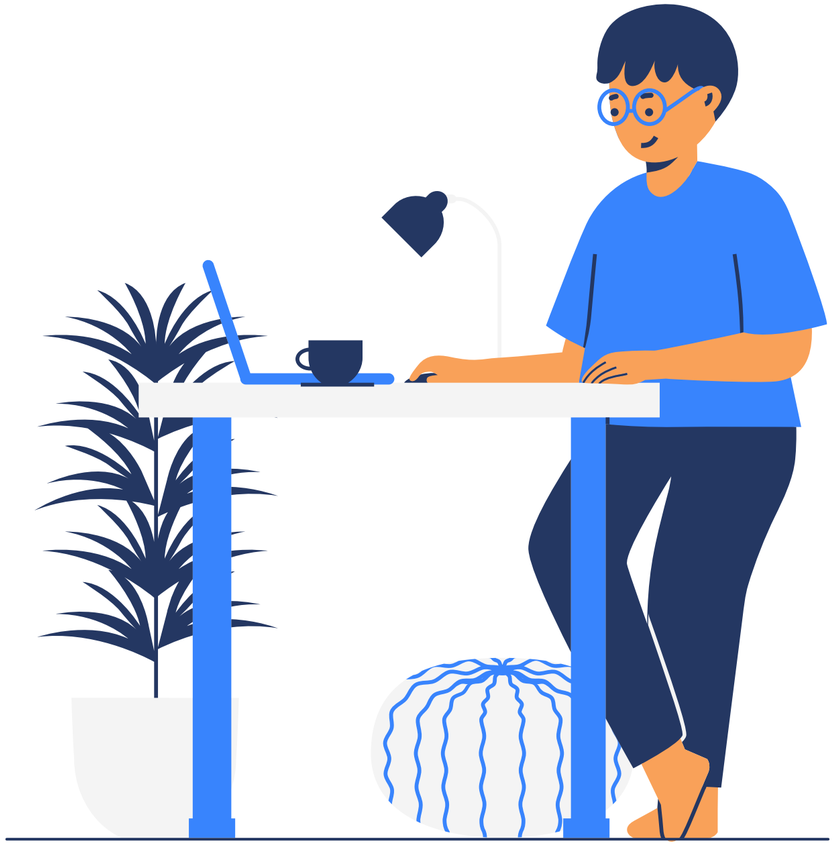
초기에는 유튜브에서 음원을 다운로드 받아 저장하는 방식을 채택하였다.

하지만 용량 문제, 다운로드 시간 등의 문제로 실시간 재생 방식으로 변경하였다.

02 음원 재생 기능

유튜브 API 기능 중 음원 재생 기능이 C#에서는 제공되지 않았다.

JS로는 음원 재생이 가능하여, 깃허브에 웹 페이지를 호스팅하여 연동하였다.



PROJECT.2

초소형 컴퓨터를 활용한 지능형 교통 신호기

- 라즈베리파이를 이용한 차량 인식과 지능형 신호등

ABOUT PROJECT

1. 유니티를 이용한 화면 구현
2. 라즈베리파이를 이용한 차량 인식
3. 대기 차량에 따른 지능형 신호등

PROJECT.2

July 4, 2021

인공 지능 신호등

뉴스 > 경제 > 산업

교통량 연동 '스마트신호등'... 올해 2000곳에 더 설치키로

김호경 기자 입력 2021-05-03 03:00 수정 2021-05-03 03:00

가 가

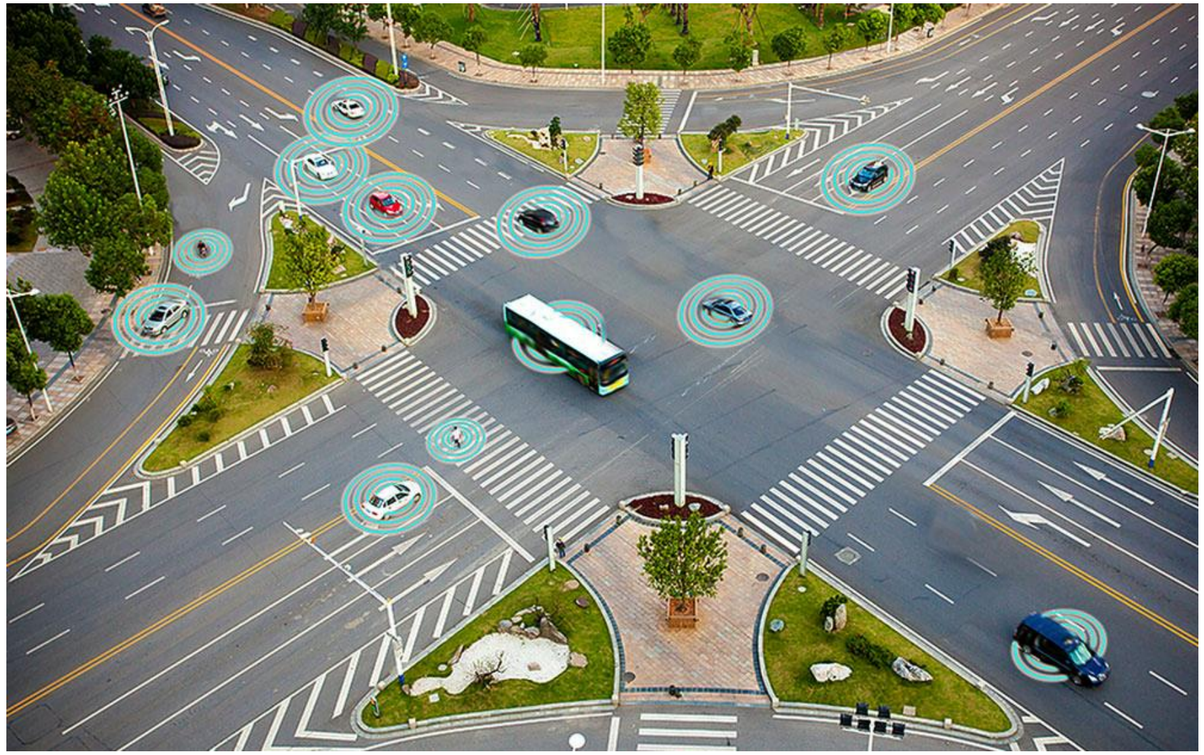
https://www.chosun.com/

교통 빅데이터 읽는 AI 신호등... 암스테르담이 뻥 뚫렸다 - 조선 ...

2019. 5. 25. — 예를 들어 암스테르담이 속한 노르트홀란트주(州)는 2012년 세계 최초로 '지능형 교통 체계(ITS·Intelligent Transportation System)'를 도입했다.

http://www.aitimes.com/

영국, AI 교통신호등 시스템 도입 추진 - AI타임스

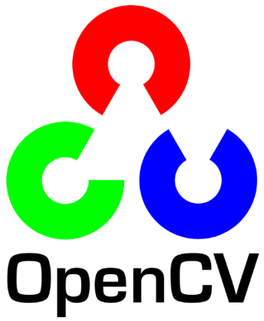


July 4, 2021

SKILLS

Unity

자동차의 움직임 구현



OpenCV

차량 인식

Java

서버 구현

PROJECT.2

July 4, 2021

프로그램 구현 개요

01 디스플레이 – 정준영

경기도 교통 DB센터 자료를 바탕으로

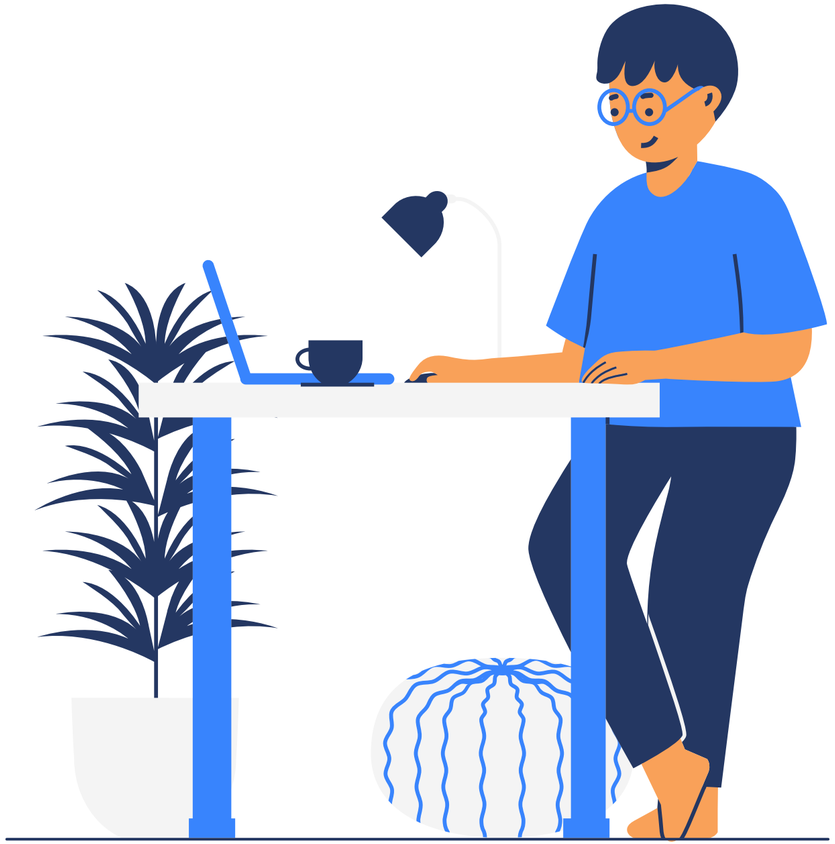
실제 차량과 유사한 움직임 구현

02 카메라

라즈베리파이 카메라를 이용한 차량 인식 구현

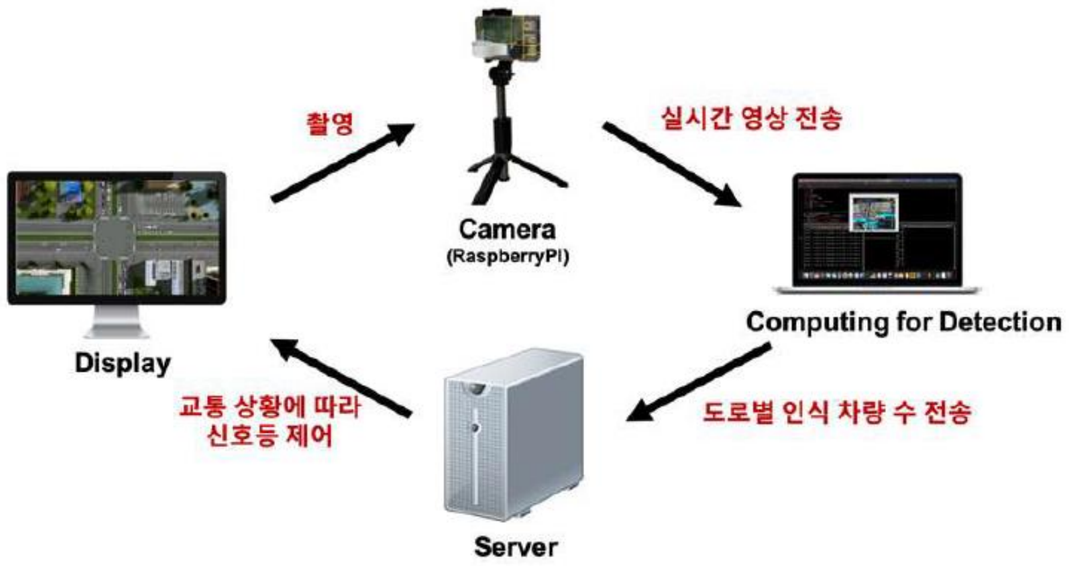
03 서버

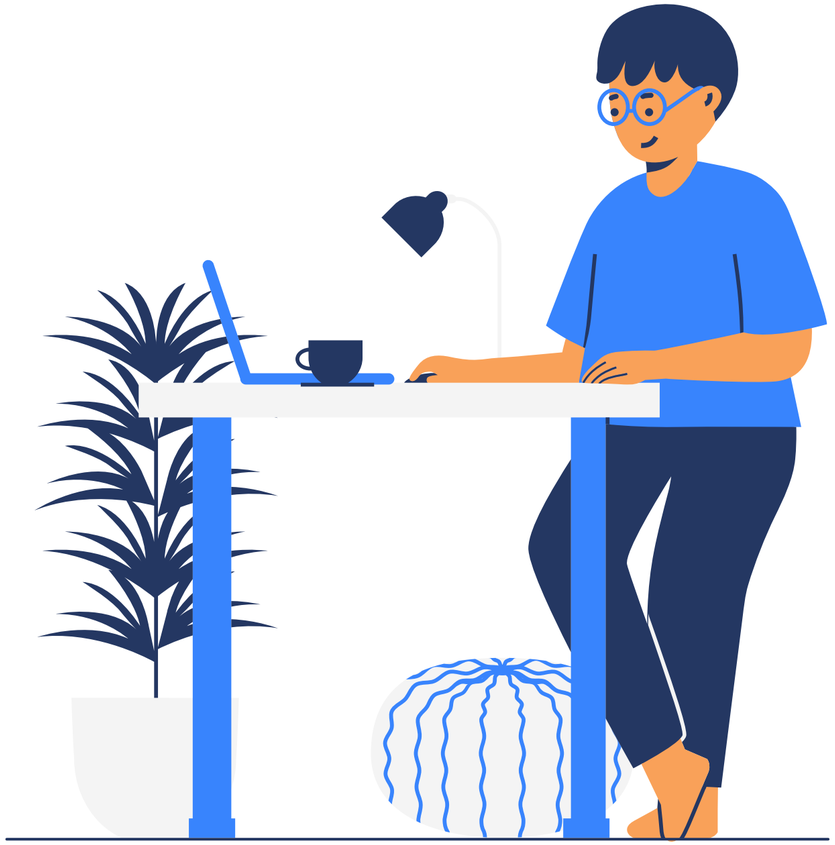
사거리 신호 정책 수립 및 디스플레이와 통신



프로그램 구성도

01 전체 시스템 구성





프로그램 구현 내용

01 시간대 구현

하루를 러시아워(출근), 평상시, 러시아워(퇴근), 야간 네 상황으로 나누어 각 상황별 배경 및 차량의 속도를 구현하였다.

프로그램 구현 내용

01 시간대 구현

하루를 러시아워(출근), 평상시, 러시아워(퇴근), 야간 네 상황으로 나누어 각 상황 별 배경 및 차량의 속도를 구현하였다.

```
timer += Time.deltaTime;
carCreateTimer += Time.deltaTime;
/*
 * speedRush = 0.12f;
 * public float speedNormal = 0.3f;
 * public float speedNight = 1.2f;
 *
 *              | 평상시 -> 퇴근 |          |   퇴근 -> 야간 |          | 야간 -> 출근 |       |   출근 -> 평상시 |
 * ---------------------------------------------------------------------------------------------------------
 * 0            70       80      90        125     140      175       210   230     240     270               290
 *   ->       평상시      |          퇴근          |        야간           |    출근     |    평상시      ->
 */

// 평상시
if (video.vp.time < 70f)
{
    setCarSpeed("평상시");
    carCreatTime = carCreateTimeNormal;

    PeopleCreate.PeopleCreatTime = 0.5f;
}
 // 평상시 -> 퇴근 변경 중 1
 else if(video.vp.time<80f)
 {
    setCarSpeed("평상시");
    if (carCreatTime > carCreateTimeRush)
        carCreatTime -= 0.008f*Time.deltaTime;
    else
        carCreatTime = carCreateTimeRush;
    PeopleCreate.PeopleCreatTime = 0.5f;

}
```

```
// 평상시 -> 퇴근 변경 중 2
else if (video.vp.time < 90f)
{
    setCarSpeed("러시아워-퇴근");
    if (carCreatTime > carCreateTimeRush)
        carCreatTime -= 0.008f * Time.deltaTime;
    else
        carCreatTime = carCreateTimeRush;
    PeopleCreate.PeopleCreatTime = 0.2f;

}
// 러시아워 - 퇴근
else if (video.vp.time < 125)
{
    setCarSpeed("러시아워-퇴근");
    carCreatTime = carCreateTimeRush;
    PeopleCreate.PeopleCreatTime = 0.2f;

    //carCreatTime = 0.12f;

}
//러시아워 퇴근 -> 야간 변경 중1
else if (video.vp.time < 140)
{
    setCarSpeed("러시아워-퇴근");
    if (carCreatTime < carCreateTimeNight)
        carCreatTime += 0.0328f*Time.deltaTime;
    else
        carCreatTime = carCreateTimeNight;
    PeopleCreate.PeopleCreatTime = 0.2f;


}
```

프로그램 구현 내용

02 차량 생성

각 위치 차량 생성 및 속도, 방향 부여

```
if (timer>=carCreatTime)
{
    if(CarStack.Count==0)
    {

        CarRandomList1 = new List<CarRoad> { CarRoad.CAR_HOR_UP_1, CarRoad.CAR_HOR_UP_2, CarRoad.CAR_HOR_DOWN_1, CarRoad.CAR_HOR_DOWN_2,
    CarRoad.CAR_VER_LEFT_1, CarRoad.CAR_VER_LEFT_2, CarRoad.CAR_VER_RIGHT_1, CarRoad.CAR_VER_RIGHT_2 };
        CarRandomList2 = new List<CarRoad> { CarRoad.CAR_HOR_UP_1, CarRoad.CAR_HOR_UP_2, CarRoad.CAR_HOR_DOWN_1, CarRoad.CAR_HOR_DOWN_2 };

        int rvalue = Random.Range(4, 7);
        CarRoad start = CarRandomList1[rvalue];
        CarRandomList1.RemoveAt(rvalue);
        CarStack.Push(start);

        rvalue = Random.Range(4, 6);
        CarRoad last = CarRandomList1[rvalue];
        CarRandomList1.RemoveAt(rvalue);

        for(int i= CarRandomList1.Count-1; i>=0;i--)
        {
            rvalue = Random.Range(0, i);
            CarStack.Push(CarRandomList1[rvalue]);
            CarRandomList1.RemoveAt(rvalue);

        }
```

```
        rvalue = Random.Range(0, 4);
        CarStack.Push(CarRandomList2[rvalue]);
        CarRandomList2.RemoveAt(rvalue);

        rvalue = Random.Range(0, 3);
        CarStack.Push(CarRandomList2[rvalue]);
        CarRandomList2.RemoveAt(rvalue);


    }

    CarCreateRoad(CarStack.Pop());
    timer -= carCreatTime;
}
```

프로그램 구현 내용

03 차량 충돌

차량 충돌에 따른 움직임 구현

```
if (ishitBlock)
{
    if (car_create_th.hor_Down_Traffic_Light != AllObjectS.TRAFFIC_LIGHT_RED)
    {
        // speed = -car_create_th.init_speed;
        if ((n == 1 && car_create_th.hor_Down_Traffic_Light == AllObjectS.TRAFFIC_LIGHT_LEFT))
        {
            ;
        }
        else if (car_create_th.hor_Down_Traffic_Light == AllObjectS.TRAFFIC_LIGHT_RIGHT)
        {

            if (n == 1 && CarDirection == AllObjectS.CAR_DIRECTION_RIGHT)
            {
                isStop = false;
                isRotate = true;
                speed = init_Speed;
                carob.setCarStop(AllObjectS.CAR_HOR_DOWN, car_object, isStop);
                ishitBlock = false;

            }

        }
        else
        {
            speed = init_Speed;
            isStop = false;
            ishitBlock = false;
            carob.setCarStop(AllObjectS.CAR_HOR_DOWN, car_object, isStop);
        }
    }
}
```

```
else
{
    if (colGo != null && !carob.getCarStop(AllObjectS.CAR_HOR_DOWN, colGo))
    {

        speed = init_Speed;
        isStop = false;
        carob.setCarStop(AllObjectS.CAR_HOR_DOWN, car_object, isStop);

    }
}
```

프로그램 구현 내용

04 신호에 따른 차량 반응

신호에 따른 움직임 구현

```
// 정지
if (car_create_th.hor_Down_Traffic_Light == AllObjectS.TRAFFIC_LIGHT_RED)
{
    isStop = true;
    isRotate = false;
    speed = 0;
}

// 좌회전 - 1차선 좌회전, 2차선 정지
else if (car_create_th.hor_Down_Traffic_Light == AllObjectS.TRAFFIC_LIGHT_LEFT)
{
    if (n == 0)
    {
        isStop = false;
        isRotate = true;
        speed = init_Speed;


    }
    else if (n == 1)
    {
        isStop = true;
        isRotate = false;
        speed = 0;
    }


}
```

```
// 직진 및 좌회전 - 1차선 직진 or 좌회전, 2차선 직진
else if (car_create_th.hor_Down_Traffic_Light == AllObjectS.TRAFFIC_LIGHT_LEFT_GREEN)
{
    if (n == 0)
    {
        if(CarDirection==AllObjectS.CAR_DIRECTION_STRAIGHT)
        {

            isStop = false;
            isRotate = false;

        }
        else if (CarDirection == AllObjectS.CAR_DIRECTION_LEFT)
        {


            isStop = false;
            isRotate = true;
        }

    }
    else if (n == 1)
    {
        isStop = false;
        isRotate = false;

    }
    speed = init_Speed;


}
```

프로그램 구현 내용

05 차량 이동

본인의 목적지를 향해 이동하는 차량 움직임 구현

```
if (isRotate)
{
    // 아래
    // 1차선 좌회전

    if (n == 0)
    {
        if (car_object.transform.position.x >= 0.25)
        {
            car_object.transform.eulerAngles = new Vector3(0, 0, 90);

            isRotate = false;

        }
        else
        {
            car_object.transform.eulerAngles = new Vector3(0, 0, angle);

            //angle += (1.95f * Mathf.Abs(speed)) / 6.0f;
            angle += (20.3f * Mathf.Abs(speed * Time.deltaTime));
        }

    }
```

```
car_object.transform.Translate(speed * Time.deltaTime, yspeed * Time.deltaTime, 0);

if (angle >= 90)
{
    car_object.transform.eulerAngles = new Vector3(0, 0, 90);

    isRotate = false;

}
else if (angle <= -90)
{
    car_object.transform.eulerAngles = new Vector3(0, 0, -90);

    isRotate = false;

}


if (x > 12f || y < -6f || y > 6f)
{

    car_create_th.car_hor_down_list[n]--;
    carob.RemoveCar(AllObjectS.CAR_HOR_DOWN, car_object);
    Destroy(this.gameObject);
}
```

프로그램 구현 내용

06 차량 통제

모든 차량의 정보를 가진 객체를 만들고, 중앙에서 차량을 통제한다.

이 객체는 프로그램 상 하나만 필요하기 때문에 Singleton 패턴이 적용되었다.

```
private car_Object()
{


    CarObjects = new List<List<GameObject>>();

    car_speed = new List<List<float>>();
    is_Traffic_block = new List<List<bool>>();
    carTime = new List<List<float>>();
    carStop = new List<List<bool>>();
    carDirection = new List<List<int>>();

    for (int i = 0; i < 4; i++)
    {
        CarObjects.Add(new List<GameObject>());
        car_speed.Add(new List<float>());
        is_Traffic_block.Add(new List<bool>());
        carTime.Add(new List<float>());
        carStop.Add(new List<bool>());
        carDirection.Add(new List<int>());

    }


}

public static car_Object getObject()
{
    if (thisObject == null)
        thisObject = new car_Object();

    return thisObject;
}
```

```
public void setCarSpeed(int n, GameObject go,float speed)
{
    int index = CarObjects[n].IndexOf(go);
    car_speed[n][index]=speed;
}

public void setTrafficBlock(int n, GameObject go, bool tb)
{
    int index = CarObjects[n].IndexOf(go);
    is_Traffic_block[n][index]=tb;
}

public bool getTrafficBlock(int n, GameObject go)
{
    int index = CarObjects[n].IndexOf(go);

    return is_Traffic_block[n][index];
}

public void initTrafficBlock(int n)
{
    for(int i=0;i<is_Traffic_block[n].Count;i++)
    {
        is_Traffic_block[n][i] = false;
    }
}

public int getCarIndex(int n,GameObject go)
{
    int index= CarObjects[n].IndexOf(go);
    return index;
}
```

```
public int getCarDirection(int n, GameObject go)
{
    int index = CarObjects[n].IndexOf(go);
    return carDirection[n][index];
}


public void RemoveCar(int n,GameObject go)
{
    int num;


    num = CarObjects[n].IndexOf(go);
    car_speed[n].RemoveAt(num);
    is_Traffic_block[n].RemoveAt(num);
    carStop[n].RemoveAt(num);
    carTime[n].RemoveAt(num);

    carDirection[n].RemoveAt(num);

    CarObjects[n].RemoveAt(num);


}
```

프로그램 구현 내용

07 서버와 통신

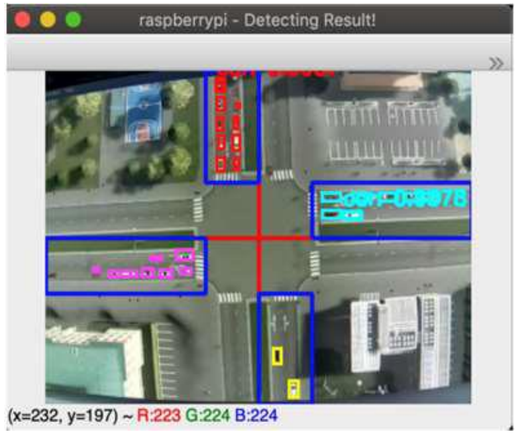
서버에게 차량 정보를 제공하고, 서버로부터 신호등의 내용을 받는다.

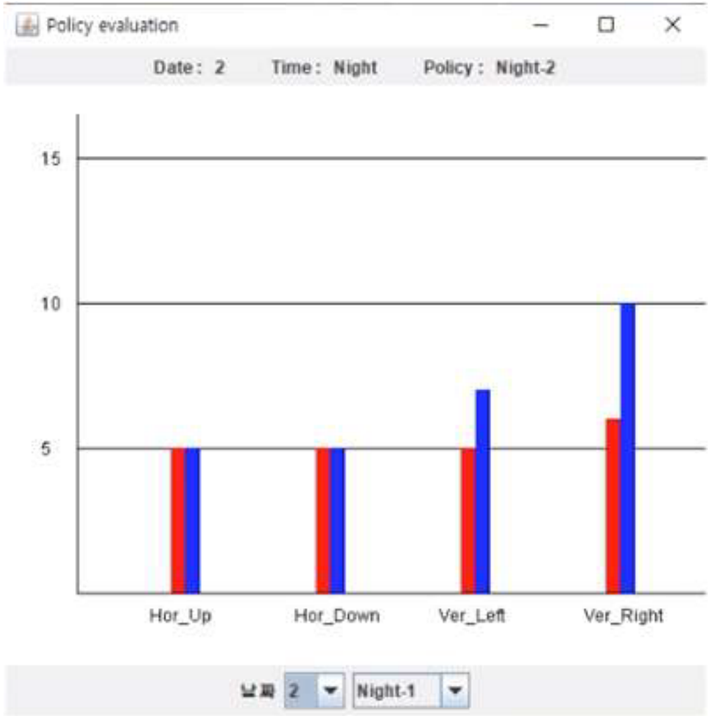
```
public void Run()
{

    try
    {
        localAddr = IPAddress.Parse(ServerIP);
        int portnum = ServerPort;

        IPEndPoint ipep = new IPEndPoint(localAddr, portnum);

        AllObjectS.client = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

        AllObjectS.client.Connect(ipep);

        changeScene = true;

        AllObjectS.ServerConnect = true;


    }
    catch (Exception)
    {
        ConnectFail = true;


    }
}
```

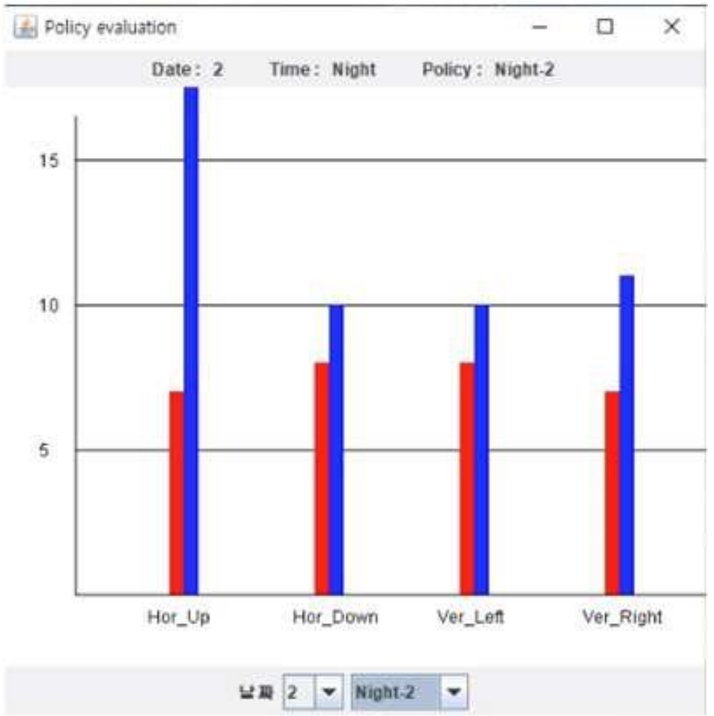
프로그램 구현 내용

08 최종 결과

카메라를 통한 차량 인식 및 신호 변경에 따른 효율성 증가







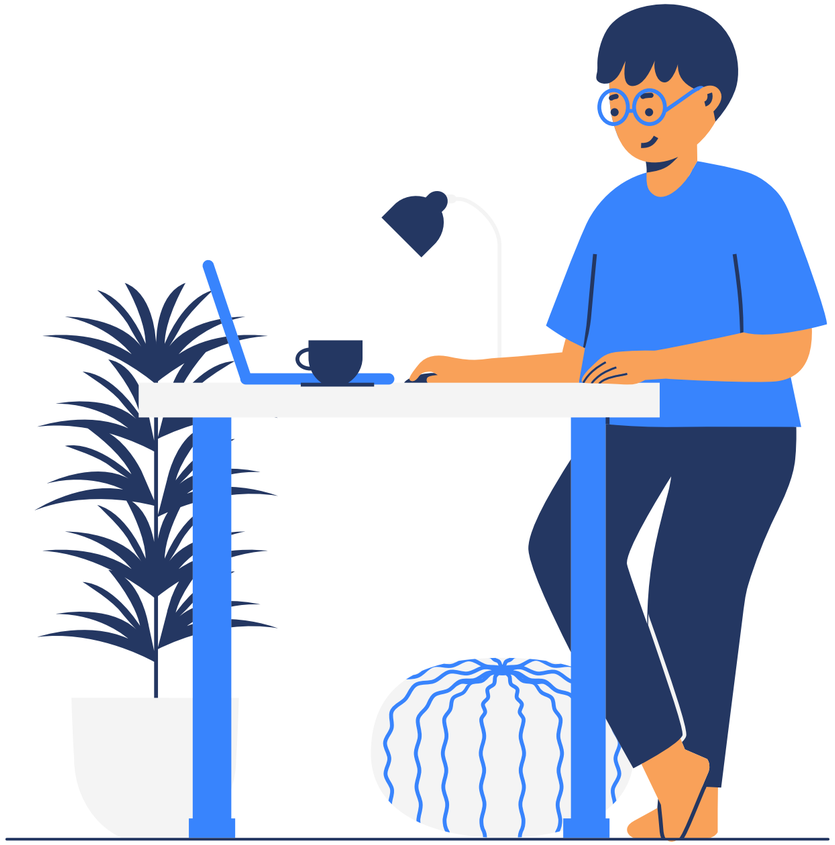
어려웠던 점 및 해결 방안

01 실제 차량들의 움직임 구현

초기 실제 차량의 속도와 움직임을 어떻게 구현해야 하는지 막막했다.
경기도 교통 DB 센터 자료를 바탕으로 차량 속도와 움직임을 분석하여
실제 차량과 유사한 속도와 움직임을 가진 차량들을 구현할 수 있었다.

02 차량의 충돌 구현

차량이 충돌하게 되는 경우 뒷 차량의 움직임과 속도를 어떠한 방식으로
구현해야 하는지 막막했다.
프로그램 상 차량의 모든 정보를 가진 객체를 만들고, 차량이 충돌하게 되면
해당 객체를 통해 충돌 차량 정보를 받아 움직임과 속도를 조절하는 방식으로
문제를 해결하였다.



PROJECT.3

탐방
- 테마별 주거지역 추천 서비스

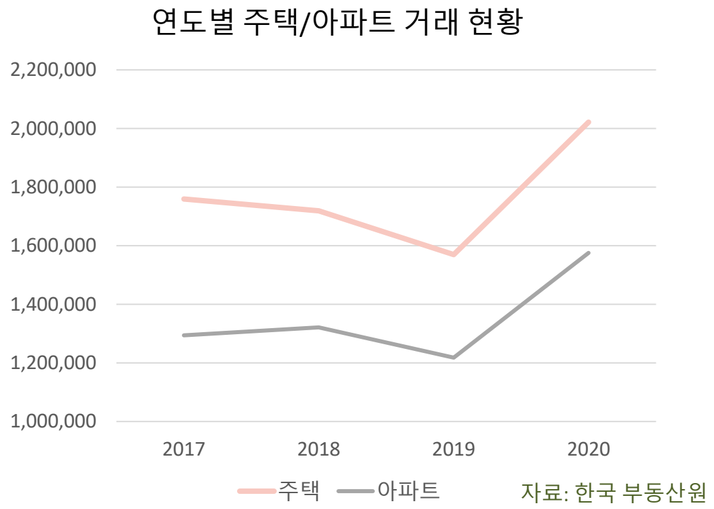
ABOUT PROJECT

1. 주거 지역 선택 서비스
2. 테마별 통계 자료
3. 검색 기능
4. 주거 지역 상세 기능

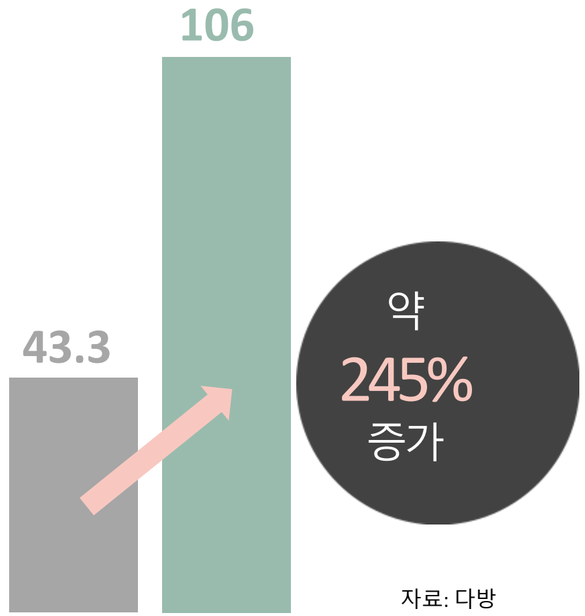
PROJECT.3

July 4, 2021

주택 거래량 증가

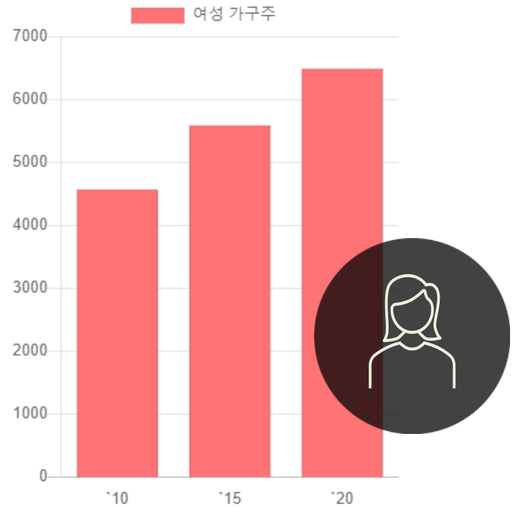


- 주택과 아파트 매매 거래량의 증가
- 2019년도 이후 부동산 시장 과열 가속화



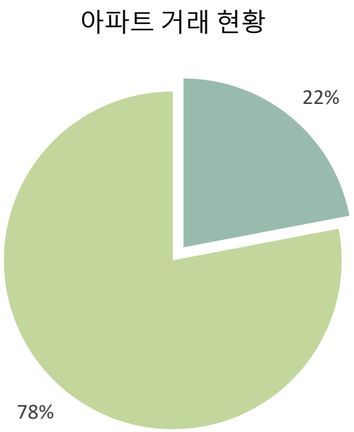
- 부동산 중개 온라인 플랫폼에 대한 수요 증가
- '다방'을 비롯한 부동산 중개 앱 이용자 수 증가

시장 세분화 & 타겟팅



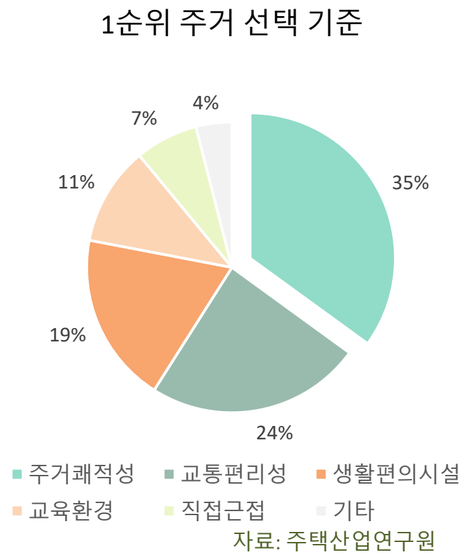
자료: 통계청, 여성가족부

- 여성 1인 가구 수의 증가



자료: 통계청, 한국 부동산원

- 주택 매매 시장 참여 주체로서 고연령층이 상당한 비율을 차지
- 고령화 현상 지속



자료: 주택산업연구원

- 웰에이징 라이프 스타일
- 주거지 선택 기준 순위 변화

PROJECT.3

July 4, 2021

포지셔닝

숲세권

#공원 #지하철 #인구밀도 #대기환경

안전

#CCTV #범죄율 #가로등 #소방서&경찰서

실버라이프

#병원 #경로당 #공원 #시장

July 4, 2021

SKILLS

Vue.js

SPA 화면 구현

Spring boot

REST API 백엔드 서버 구현

MySQL

DB 구축

PROJECT.3

July 4, 2021

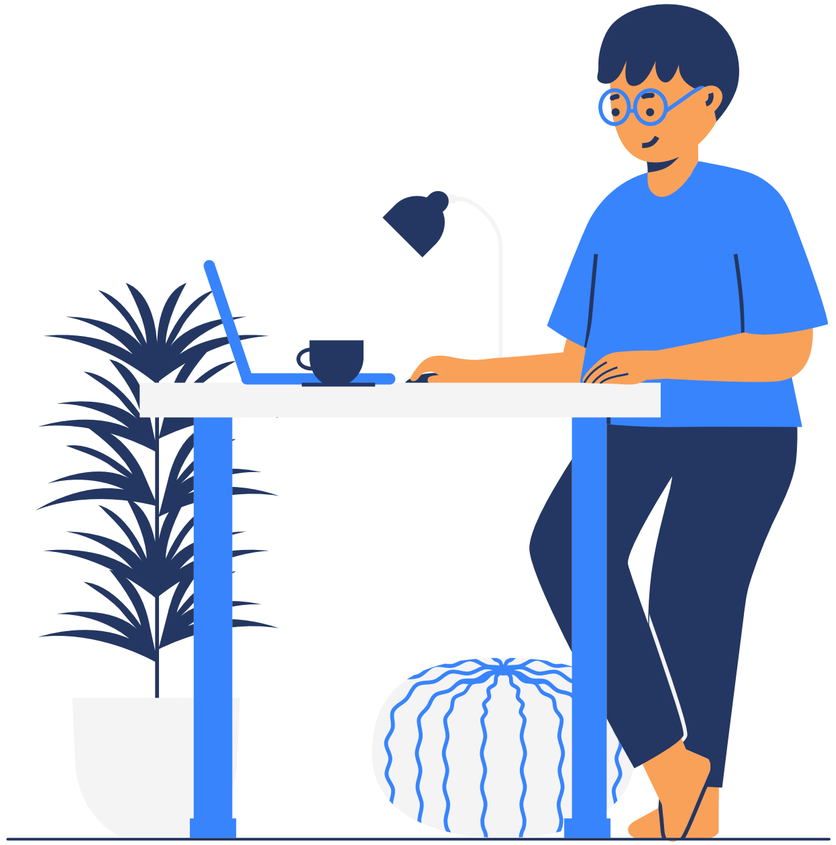
프로그램 구현 개요

01 프론트 엔드

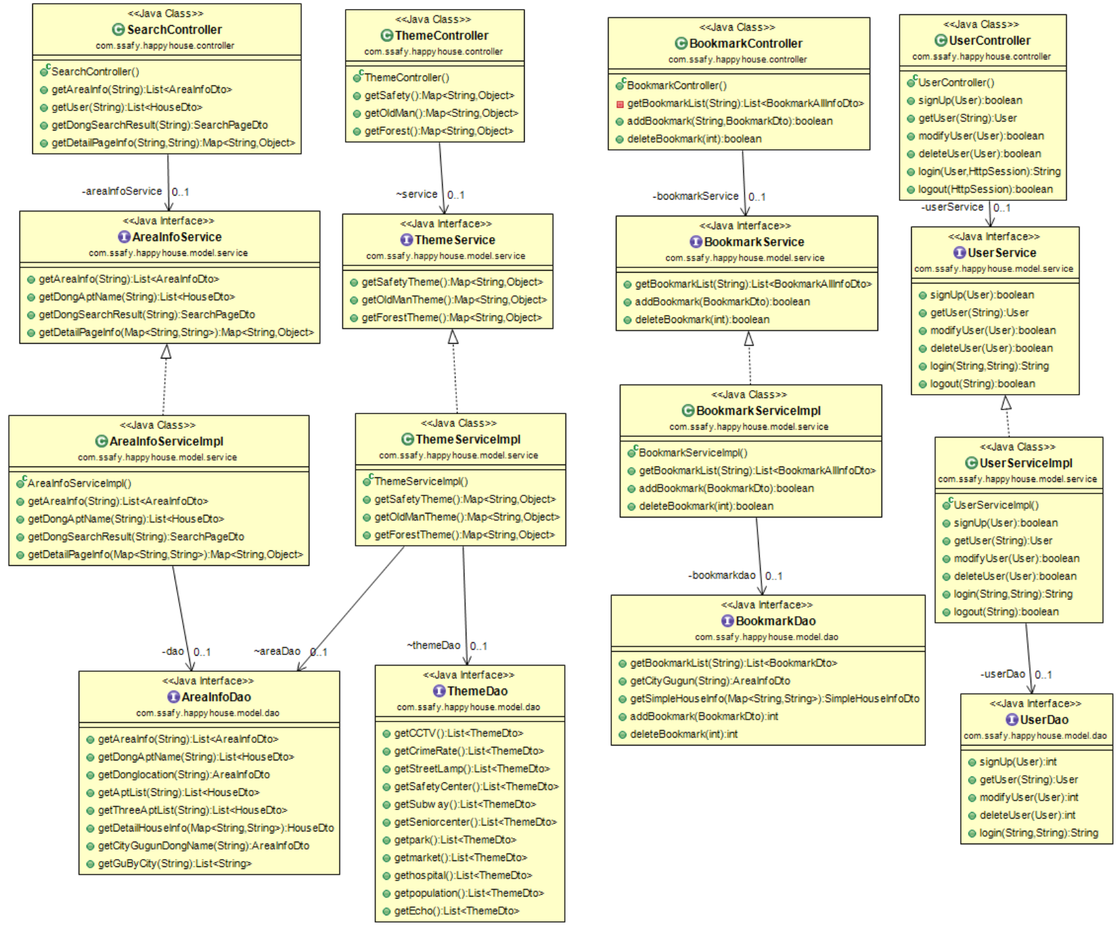
Vue.js를 통한 SPA 구현

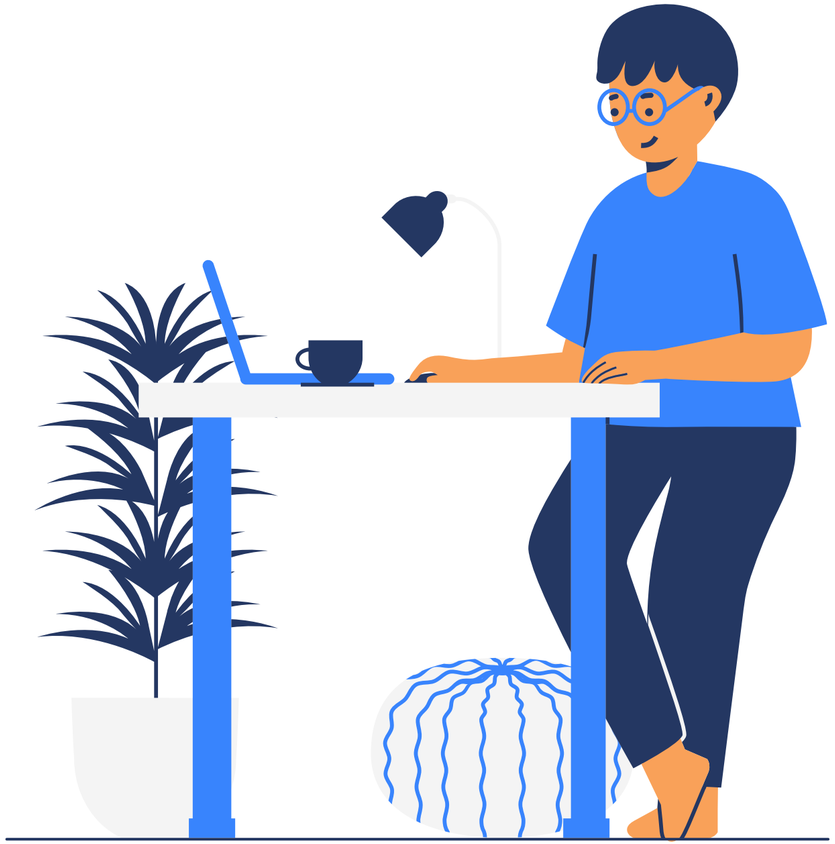
02 백엔드 – 정준영

Spring boot를 통해 REST API 서버 구현



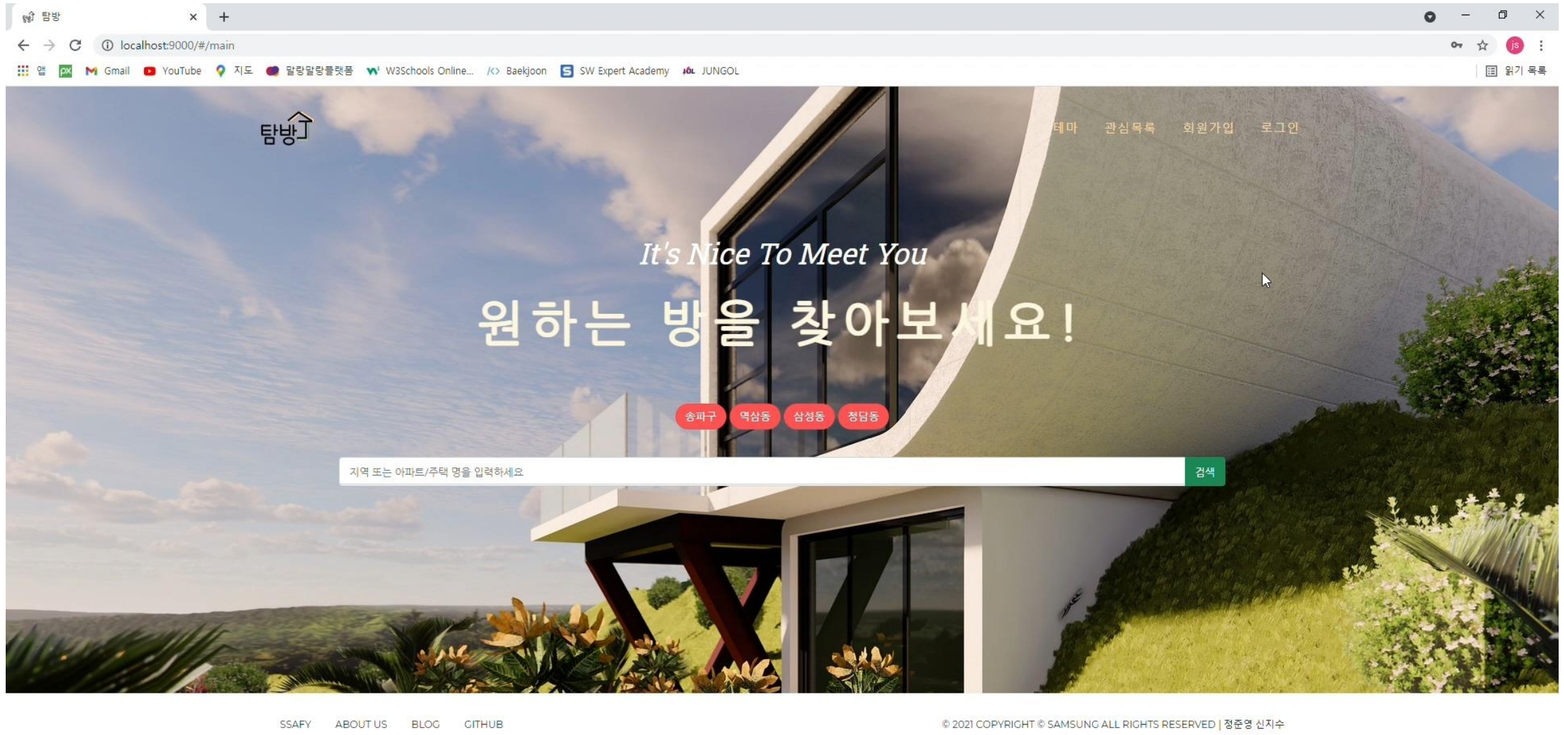
프로그램 클래스 다이어그램





프로그램 구현 내용

01 메인 화면



프로그램 구현 내용

02 회원 가입 및 로그인

프로그램 구현 내용

03 회원 가입 및 로그인

로그인 정보를 세션에 저장하여, 다시 접속해도 로그인 상태가 유지되도록 설정

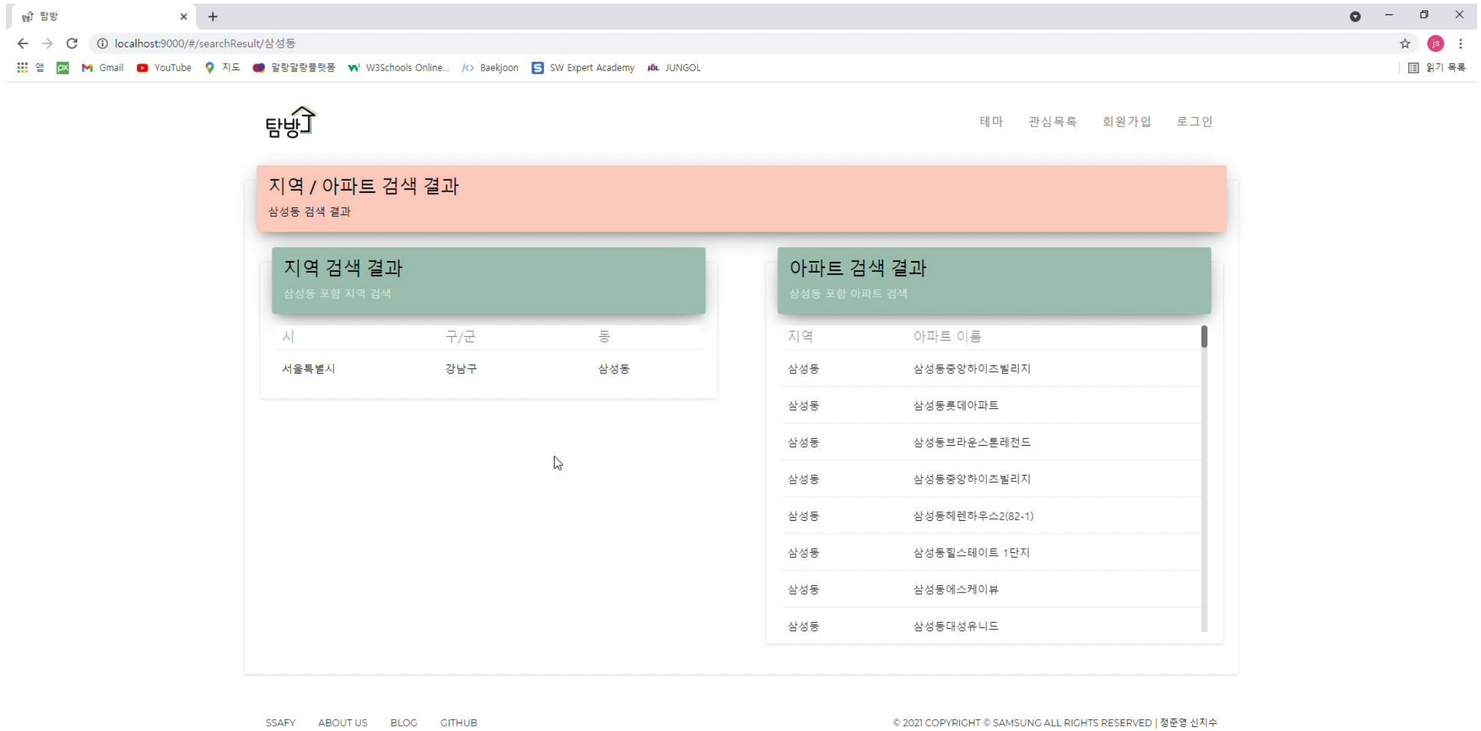
```
// 로그인
@PostMapping("/login")
public String login(@RequestBody User user, HttpSession session) {
    String name = userService.login(user.getId(), user.getPassword());
    if (name != null) {
        session.setAttribute("id", user.getId());
        session.setAttribute("name", name);
        return name;
    } else {
        return "false";
    }
}

// 로그아웃
@GetMapping("/logout")
public boolean logout(HttpSession session) {
    session.invalidate();
    return true;
}
```

```
@Mapper
@Repository
public interface UserDao {

    // 회원 가입
    public int signUp(User user);

    // 회원 정보 불러오기
    public User getUser(String id);

    // 회원 정보 수정
    public int modifyUser(User user);

    // 회원 탈퇴
    public int deleteUser(User user);

    // 로그인
    public String login(String id, String password);

}
```

```
<mapper namespace="com.ssafy.happyhouse.model.dao.UserDao">

   <!-- 회원 가입 -->
    <insert id="signUp" parameterType="User">
        insert into user(id, password, name, birthyear, birthmonth, birthdate, phonenum, email, address)
        values( #{id}, #{password}, #{name}, #{birthyear}, #{birthmonth}, #{birthdate}, #{phonenum}, #{email}, #{address})
    </insert>

    <!-- 회원 정보 불러오기 -->
    <select id="getUser" parameterType="string" resultType="User">
        select *
        from user
        where id = #{id}
    </select>

    <!-- 회원 정보 수정 -->
    <update id="modifyUser" parameterType="User">
        UPDATE user
        SET password=#{password}, phonenum=#{phonenum}, email=#{email}, address=#{address}
        where id=#{id}
    </update>

    <!-- 회원 탈퇴 -->
    <delete id="deleteUser" parameterType="User">
        delete from user
        where id=#{id} and password=#{password}
    </delete>

    <!-- 로그인 -->
    <select id="login" parameterType="string" resultType="String">
        select name
        from user
        where id=#{id} and password=#{password}
    </select>


</mapper>
```

프로그램 구현 내용

03 지역 검색



프로그램 구현 내용

03 지역 검색

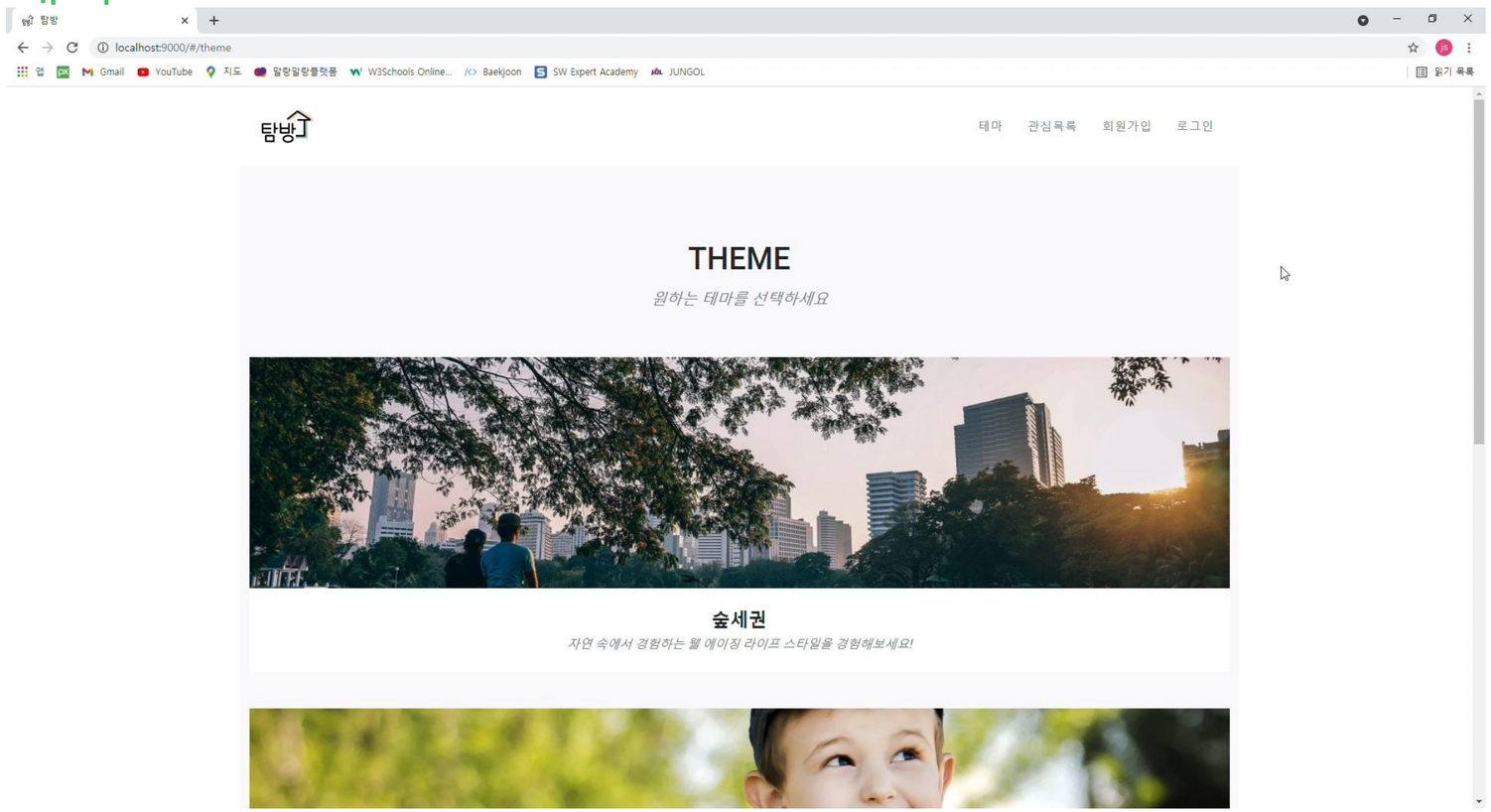
REST 형식으로 지역 및 아파트 상세 정보 검색 구현

```
@CrossOrigin("*")
@RequestMapping("/search")
@RestController
public class SearchController {

    @Autowired
    private AreaInfoService areaInfoService;

    // 지역 이름으로 검색
    @GetMapping("/area/{word}")
    public List<AreaInfoDto> getAreaInfo(@PathVariable String word) {
        return areaInfoService.getAreaInfo(word);
    }

    // 아파트 이름으로 검색
    @GetMapping("/apt/{word}")
    public List<HouseDto> getUser(@PathVariable String word) {
        return areaInfoService.getDongAptName(word);
    }

    // 동 검색에 대한 요청(동 이름이 오면 동 위치 & 아파트 이름/거래 금액/크기/층수/위치)
    @GetMapping("/dong/{word}")
    public SearchPageDto getDongSearchResult(@PathVariable String word) {
        return areaInfoService.getDongSearchResult(word);
    }

    /**
    세부 페이지에 대한 요청(동 이름, 아파트 이름이 넘어오면 HashMap으로 필요한 정보를 리턴)
    HashMap에 담길 정보들: 시, 구군, 동 이름 ⇒AreaInfoDto, 아파트 정보 Dto ⇒HouseDto
                        동에 있는 아파트 리스트 가져오기
     */
    @GetMapping("/detail/{dong}/{aptName}")
    public Map<String, Object> getDetailPageInfo(@PathVariable String dong, @PathVariable String aptName) {
        Map<String, String> paramMap = new HashMap<String, String>();
        paramMap.put("dong", dong);
        paramMap.put("aptName", aptName);
        return areaInfoService.getDetailPageInfo(paramMap);
    }
}
```

```
/**
    세부 페이지에 대한 요청(동 이름, 아파트 이름이 넘어오면 HashMap으로 필요한 정보를 리턴)
    HashMap에 담길 정보들: 시, 구군, 동 이름 ⇒AreaInfoDto, 아파트 정보 Dto ⇒HouseDto
                        동에 있는 아파트 리스트 가져오기
*/
@Override
public Map<String, Object> getDetailPageInfo(Map<String, String> paramMap) {
    Map<String,Object> map = new HashMap<String, Object>();
    // 동 이름으로 시, 구군, 동 이름 가져오기
    AreaInfoDto cityGugunDongName = dao.getCityGugunDongName(paramMap.get("dong"));
    // 동 이름&아파트 이름으로 아파트 상세 정보 검색
    HouseDto detailHouseInfo = dao.getDetailHouseInfo(paramMap);

    // 해당 동에 있는 아파트 리스트 가져오기
    List<HouseDto> houseList = dao.getThreeAptList(paramMap.get("dong"));

    map.put("cityGugunDongName", cityGugunDongName);
    map.put("detailHouseInfo", detailHouseInfo);
    map.put("houseList", houseList);
    return map;
}
```

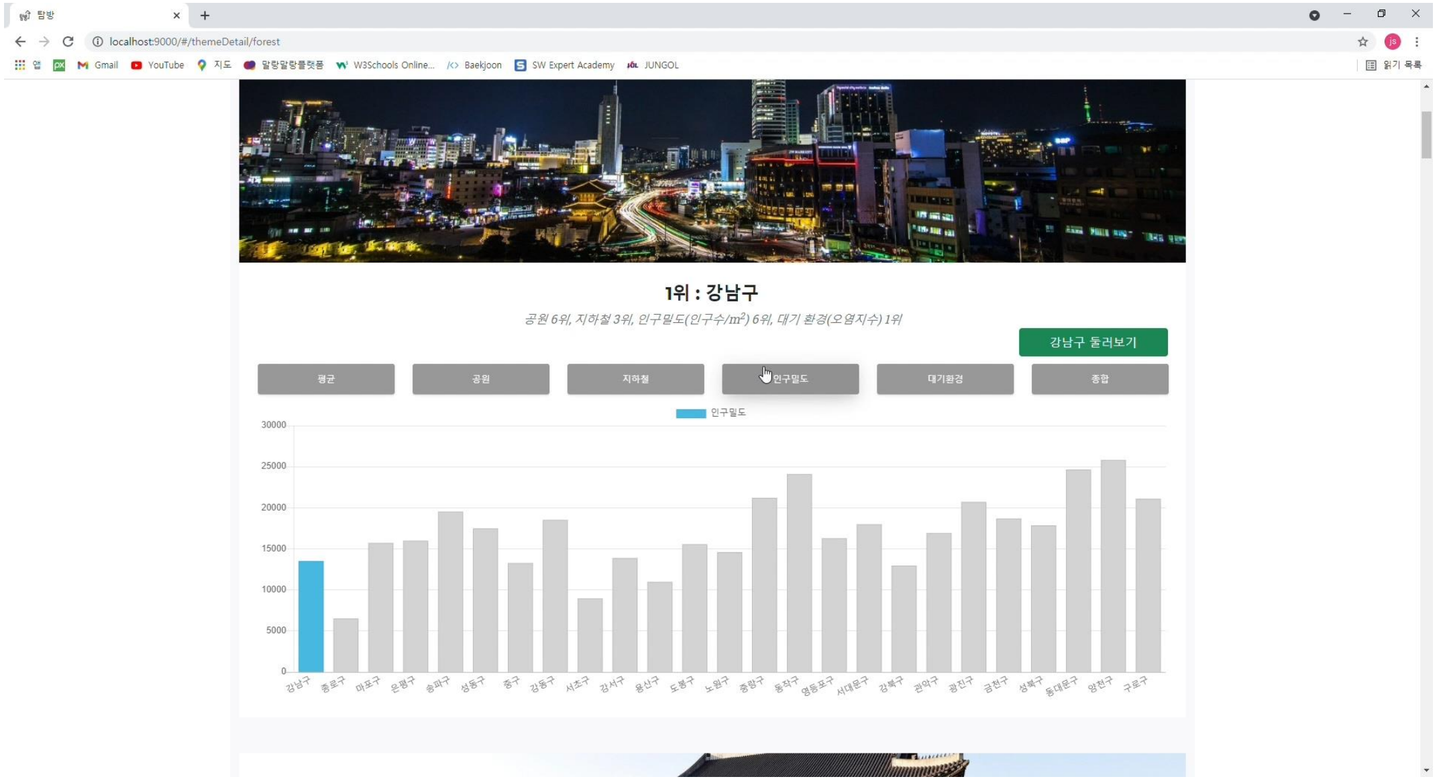
프로그램 구현 내용

04 테마



프로그램 구현 내용

04 테마



프로그램 구현 내용

04 테마

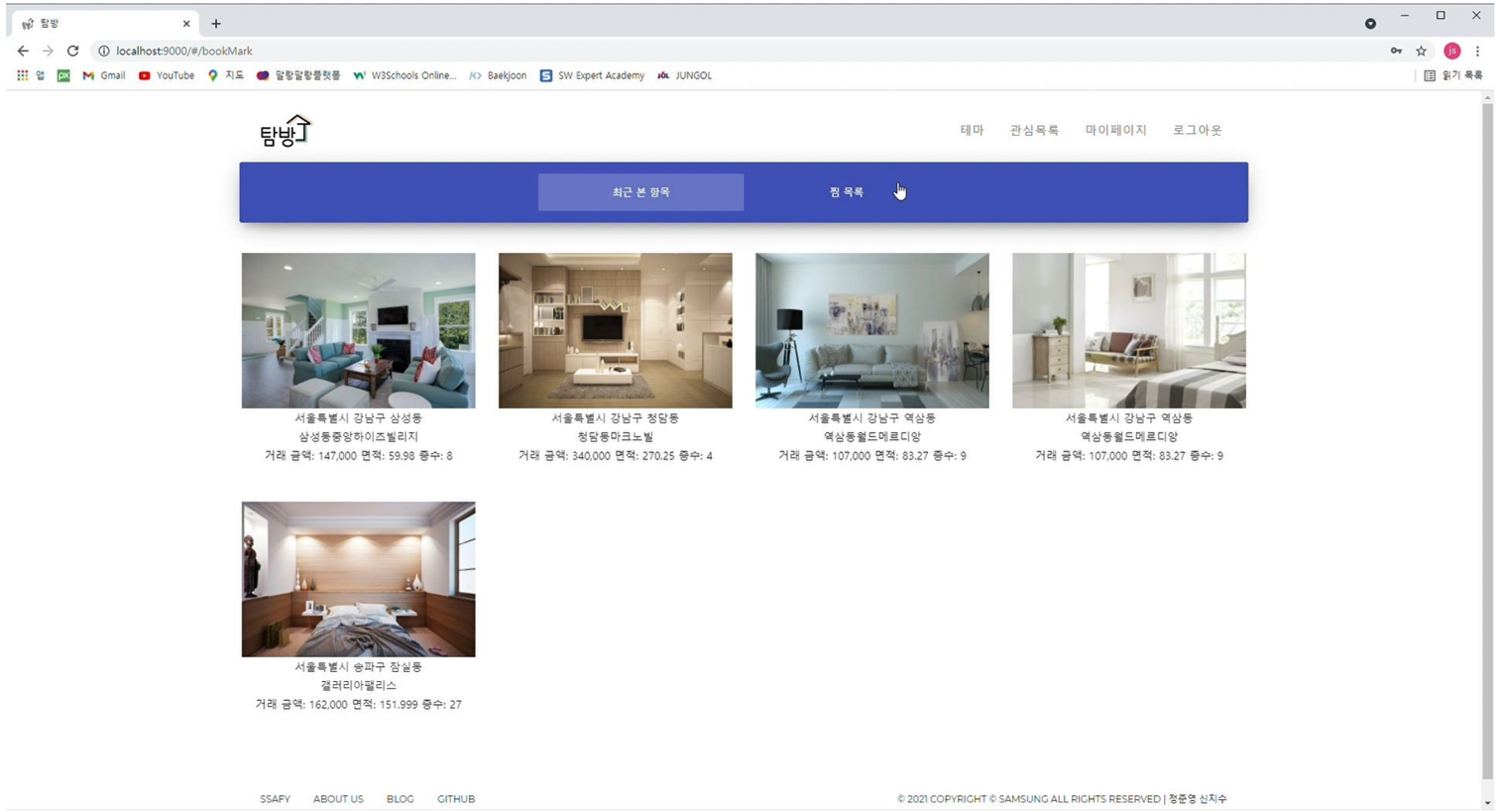
각 테마별 정보를 요청 받으면, DB에서 해당 자료를 가져와 순위 측정 및 전달

```
@CrossOrigin("*")
@RestController
@RequestMapping("/theme")
public class ThemeController {

    @Autowired
    ThemeService service;

    @GetMapping("/safety")
    public Map<String,Object> getSafety(){
        return service.getSafetyTheme();
    }

    @GetMapping("/oldman")
    public Map<String,Object> getOldMan(){
        return service.getOldManTheme();
    }

    @GetMapping("/forest")
    public Map<String,Object> getForest(){
        return service.getForestTheme();
    }
}
```

```
@Autowired
AreaInfoDao areaDao;

@Override
public Map<String, Object> getSafetyTheme() {

    Map<String , Object> result = new HashMap<String, Object>();

    List<String> guNameList = areaDao.getGuByCity("서울특별시");

    List<ThemeDto> cctvList = themeDao.getCCTV();

    List<ThemeDto> crimeRateList = themeDao.getCrimeRate();

    List<ThemeDto> streetlampList = themeDao.getStreetLamp();

    List<ThemeDto> safetyCenterList = themeDao.getSafetyCenter();

    result.put("cctvList", cctvList);
    result.put("crimeRateList", crimeRateList);
    result.put("streetlampList", streetlampList);
    result.put("safetyCenter", safetyCenterList);


    int cctvLength = cctvList.size();
    int crimeRateLength = crimeRateList.size();
    int streetlampLength = streetlampList.size();
    int safetyCenterLength = safetyCenterList.size();
```

프로그램 구현 내용

05 찜목록



프로그램 구현 내용

05 찜목록

찜목록 역시 REST 형태로 구현하였습니다.

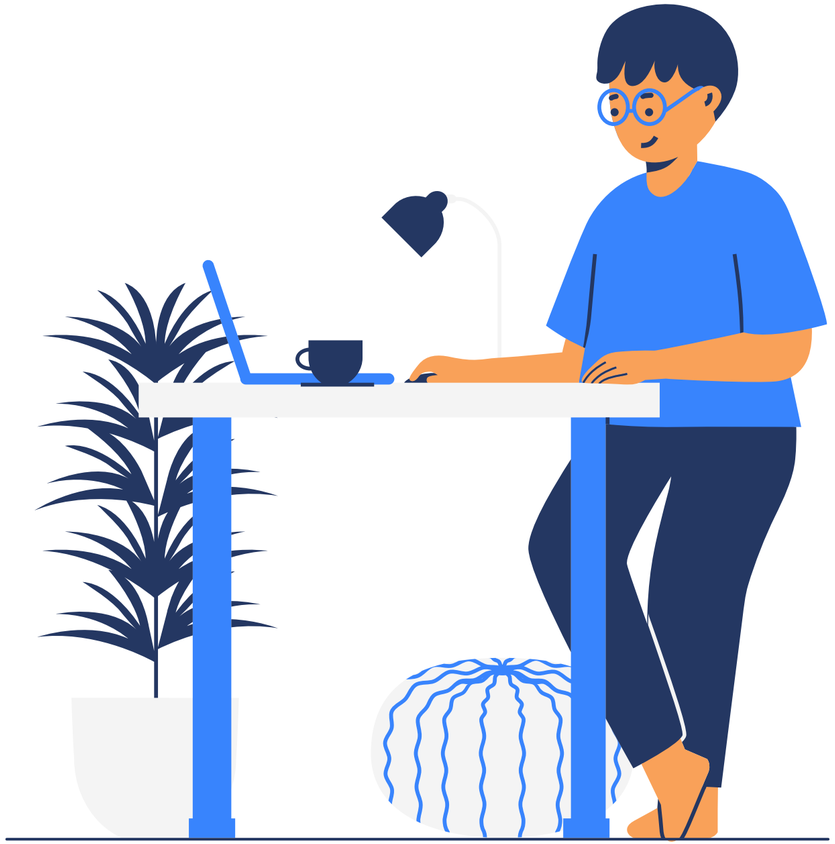
```
@CrossOrigin("*")
@RequestMapping("/bookmark")
@RestController
// 관심 목록 중 찜하기 기능에 관한 처리
public class BookmarkController {

    @Autowired
    private BookmarkService bookmarkService;

    // 찜 목록 전체 불러오기
    @GetMapping("/{id}")
    private List<BookmarkAllInfoDto> getBookmarkList(@PathVariable String id) {
        return bookmarkService.getBookmarkList(id);
    }

    // 찜 목록에 추가
    @PostMapping("/{id}")
    public boolean addBookmark(@PathVariable String id, @RequestBody BookmarkDto bookmarkDto) {
        bookmarkDto.setId(id);
        return bookmarkService.addBookmark(bookmarkDto);
    }

    // 찜 목록에서 삭제
    @DeleteMapping("/{no}")
    public boolean deleteBookmark(@PathVariable int no) {
        return bookmarkService.deleteBookmark(no);
    }

}
```

```
@Autowired
private BookmarkDao bookmarkdao;

// 찜 목록 가져오기
@Override
public List<BookmarkAllInfoDto> getBookmarkList(String id) {
    List<BookmarkAllInfoDto> BookmarkAllInfoDtoList = new ArrayList<BookmarkAllInfoDto>();

    List<BookmarkDto> bookmarkDtolist = bookmarkdao.getBookmarkList(id);
    for (int i = 0, size=bookmarkDtolist.size(); i < size; i++) {
        String dong = bookmarkDtolist.get(i).getDong();
        String aptName = bookmarkDtolist.get(i).getAptName();

        Map<String, String> map = new HashMap<String, String>();
        map.put("dong", dong);
        map.put("aptName", aptName);

        BookmarkAllInfoDto bookmarkAllInfoDto = new BookmarkAllInfoDto(bookmarkDtolist.get(i),
                bookmarkdao.getCityGugun(dong), bookmarkdao.getSimpleHouseInfo(map));
        BookmarkAllInfoDtoList.add(bookmarkAllInfoDto);
    }
    return BookmarkAllInfoDtoList;
}
```

어려웠던 점 및 해결 방안

01 너무 많은 데이터 전달

프론트엔드에서 특정 지역 검색 후 해당 지역의 주거 지역을 요청한다.
초기에는 해당 조건에 맞는 지역을 모두 전달하는 방식을 선택했다.
그러다 보니, 많은 데이터를 DB에 접근하여 가져오고, 그것을 전달하는 과정에서
부하가 발생하고, 해당 데이터를 바탕으로 프론트엔드 에서도 랜더링 하는데
오래 걸리는 문제가 있었다.
이 문제를 해결하기 위하여 백엔드에서 페이징 처리를 적용하였다.



July 4, 2021

Thank you