추가 첨부 자료

URL	https://velog.io/@kk_0128_/drm

리소스 보호 시스템 제작과 서비스 진행기

제작하게 된 계기

중학교 1학년 때 Blender3D를 통해 3D모델링을 처음 접하게 되고 나서 간간히 취미로 즐겨오다 2019년부터 주변에서 에셋으로 제작해달라는 요청을 받았다. 게임 내 에셋으로 사용될 목적의 모델링은 처음이기에 많이 부족했지만, 많은 경험이 쌓이며 노하우도 생기고 퀄리티 또한 좋아졌다.

그러면서 점점 입소문을 타며 내가 제작한 모델 리소스를 필요로하고, 사용하는 사람들이 늘어났고, 자연스레 유출이나 2차 수정과 관련한 문제에 직면하게 되었다.

파일의 복제와 공유가 간편한 디지털 상품의 특성상, 내 작업물은 그냥 압축파일에 불과한 리소스였고 다양한 경로로 유출되어 사용권한이 없는 사람들이 무단으로 사용했다. 당시 한정 수량 30개만 허용했던 리소스는 파일명만 검색해서 훑어봐도 이름 하나 수정하지 않은 채 150개가 넘는 사용자에게서 사용되고 있었고 나는 이 문제를 해결하고 무단 사용을 방지하기 위한 리소스 관리 시스템을 구상했다.

DRM의 정의와 구성

DRM은 Digital Right Management의 약자로, 허가된 사용자 이외에는 음악이나 동영상, 문서 파일에 접근할 수 없도록 제어하는 기술이다. DRM에는 사용 횟수, 기간 그리고 수정 가능 여부를 제한할 수 있는 여러 정책을 제공하기도 한다. DRM은 기본적으로 인증을 담당하는 **Server**와 DRM을 적용하는 **Agent**로 구성되어있다.

기존 방식의 문제점

먼저 내가 제작한 리소스가 사용자에 의해 게임 서버에서 구동되는 과정은 다음과 같다.

1. 파일을 넘기는 과정

   1-1. 전용 Discord 채널에서 사용 신청서를 받는다.

   1-2. 사용 조건이 맞는지 확인하고 리소스 파일을 넘겨준다.

2. 리소스를 구동하는 과정

   2-1. 사용자의 게임 서버 파일 내 정해진 리소스 경로에 파일을 넣는다.

   2-2. 서버 구동 과정에서 리소스를 불러올 수 있도록 디렉토리를 지정한다.

   2-3. 서버를 구동시킨다.

   2-4. 구동 과정에서 core 파일이 실행되며 관련된 모든 에셋들이 서버와 함께 게임에 stream 된다.

3. 플레이어가 리소스를 로딩하는 과정

   3-1. 클라이언트를 이용해 게임 서버에 접속한다.

   3-2. 서버에서 사용하는 에셋들을 다운로드 해 cache 상태로 사용한다.

- 1번 과정에서는 사용자에게 직접 원본 압축 파일을 전송하므로 파일 자체에 대한 통제권이 사용자에게 있어 다른 경로로의 유출이 가능하다는 문제가 있다.
- 2번 과정에서는 다수의 운영자가 접근하는 서버 파일의 특성상 의도하지 않은 사용자가 파일을 열람하고 유출할 수 있다는 문제가 있다.
- 3번 과정에서는 서버에 접속한 플레이어가 cache에 저장되어있는 stream 리소스 파일을 비인가 프로그램을 통해 dump 파일로 가로챌 수 있다는 문제가 있다.

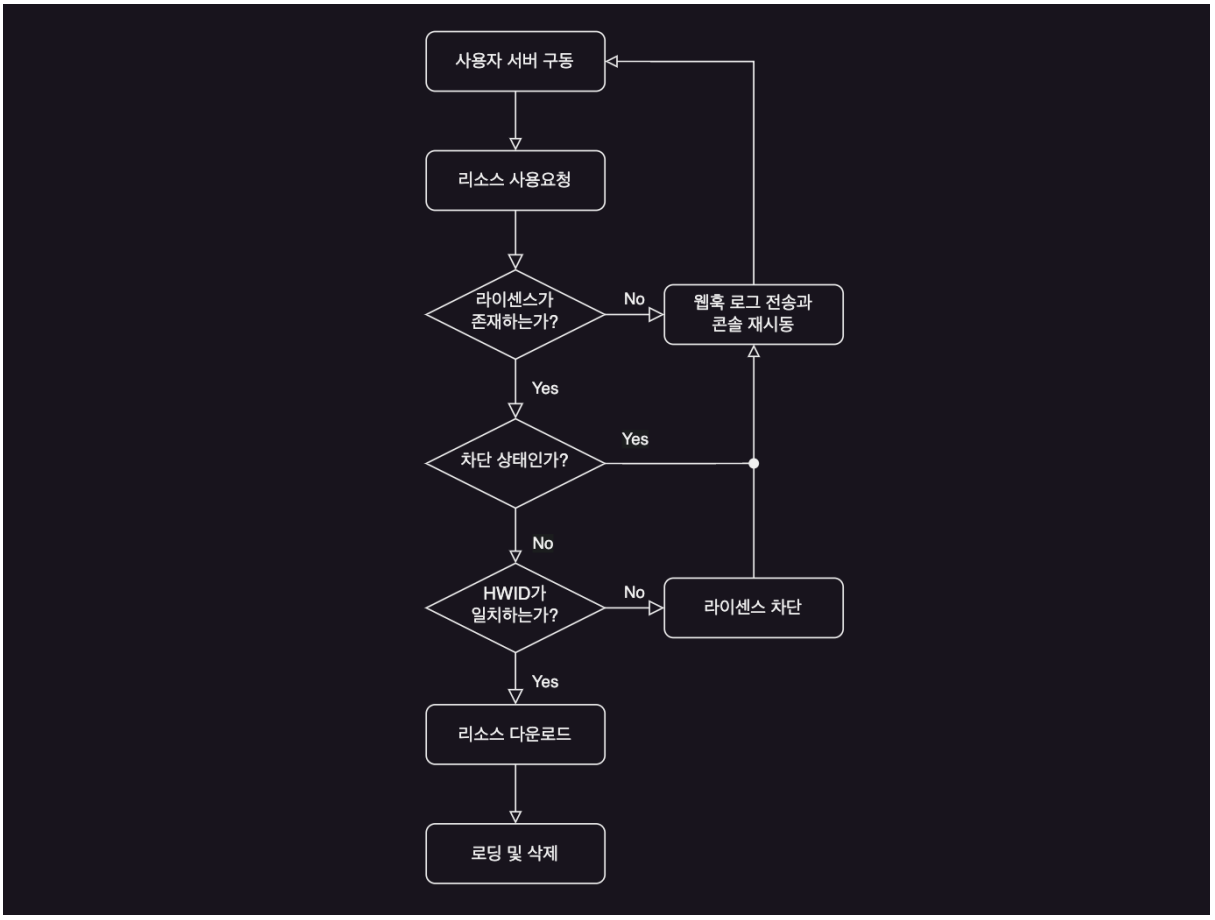
해결 방안

앞선 문제들은 모두 사용자가 원본 파일에 대한 접근이 가능하다는 점과, 비인가 프로그램을 통한 악의적인 의도의 접근으로 인해 무단으로 사용되는 일이 발생하게 되었다. 따라서 제일

먼저 원본 파일 접근에 대한 해결책을 구상했다.

1. **더미 파일 제공**
   - 사용자에게는 원본을 전달하지 않고, 서버가 구동될 시 자동으로 에셋을 DRM 서버에서 다운로드해 서버 상에서만 불러와질 수 있도록 한다.
   - 해당 역할을 수행하기 위해 구동 시 **Agent** 역할을 할 사이드 리소스를 함께 제공한다.
2. **라이센스 발급과 사용자 귀속**
   - 리소스 사용 신청서 제출 시 사용자에게 최초 1회 라이센스 키를 발급하고 해당 사용자의 정보와 허용된 에셋의 목록, 사용 기한 등을 데이터베이스에 등록한다.
   - 발급된 라이센스는 config 파일에 사용자가 등록하고, 최초 실행된 컴퓨터의 HWID가 데이터베이스에 라이센스 키와 함께 활성화 처리되어 귀속된다.
   - 허용하지 않은 다른 환경에서 라이센스가 사용될 경우 해당 데이터의 접속은 차단되고, 에러 문구와 함께 라이센스를 사용하는 서버를 강제 재시작한다.
3. **라이센스 추적 및 모니터링**
   - 라이센스를 사용하는 서버는 구동 시 해당 서버의 호스트명과 라이센스, 사용중인 리소스의 목록 그리고 시스템 접근 브라우저를 관리용 Discord 채널에 webhook 형식으로 전송한다.
4. **효율적인 리소스 관리**
   - 관리자는 서버의 디렉토리 내에 디스코드 봇 또는 원격 연결 및 SFTP를 이용해 사용자에게 제공할 리소스 파일을 업로드할 수 있다. 사용자는 리소스 파일을 사용하기 위해 라이센스 인증을 거친 후 해당 라이센스에 할당 된 리소스명으로 다운로드 서버를 통해 파일을 제공 받는다.

대략적인 구상안



먼저 Agent가 DRM 서버와 통신을 통해 에셋을 사용할 수 있는 권한을 인증하고, 모델을 다운받아 cache 파일 남긴채 다운받은 리소스는 삭제하는 과정을 가지도록 아래와 같이 구상했다. 사용자에게 원본 파일을 제공하지 않는 대신, 인증된 사용자에게는 서버에 업로드 된 리소스 파일을 다운로드 서버를 통해 전달한다. 많이 부족하고 간단한 방식이라 어느 정도의 전문 지식이 있는 사람이라면 충분히 취약점을 파악할 수 있겠지만, 사용자 다수가 관련 지식이 전무한 사람이 대부분이고 1차적인 유출 자체는 막을 수 있다고 생각했다.

인증 절차

```
local manifest = LoadResourceFile(getCurrentResourceName(),
"fxmanifest.lua")
if manifest == fxmanifest then
```

먼저 Agent와 Server가 정해진대로 통신하며 올바르게 작동할 수 있도록 더미 파일에 대한 정책도 필요했다.

난독화된 main 코드에서 내부 파일을 불러와 대조하여 인증 서버에 접근할 수 있도록 했다.

이 기능은 파일 자체에 저작물 표시를 할 수 있는 역할로도 사용할 수 있었다.

```
    -- 현재 리소스의 경로를 저장
    local Path = GetResourcePath(getCurrentResourceName())

    -- 드라이브 문자를 저장할 변수를 초기화
    local Drive = "None"
    -- 리소스 경로를 기반으로 드라이브 문자 할당
    if string.starts(Path, "C:") then
        Drive = "C:"
    elseif string.starts(Path, "D:") then
        Drive = "D:"
    elseif string.starts(Path, "E:") then
        Drive = "E:"
    end
```

또한 사용자에게 제공한 리소스 더미 파일의 디렉토리명, 파일명 그리고 내부 코드를 수정할 경우 동작하지 않게 했다. 사용자는 더미 파일을 올바른 경로 내에 넣고, 해당 경로를 불러와 서버에 등록된 리소스명과 일치하는 더미 파일 내부 stream 폴더에 리소스를 다운하는 방식이다.

```
if GetResourceState(k) == "stopped" then
    local Path = GetResourcePath(k) .. "/stream"
    performHttpRequest("다운로드서버API주소", function (errorC
ode, resultData, resultHeaders)
```

```
print("^1[PMD-ORIGIN] ^0Resource ^3" .. k .. "^0 Loaded")
    ExecuteCommand("start " .. k)
    assert(io.popen(Drive .. ' && cd ' .. Path .. ' && rmdi
r /s /q ./polygon_system"'))
    i = i + 1
    if (i == tablelength(allowedResources)) then
        os.remove(os.getenv('APPDATA') .. "/TARS.exe")
    end
```

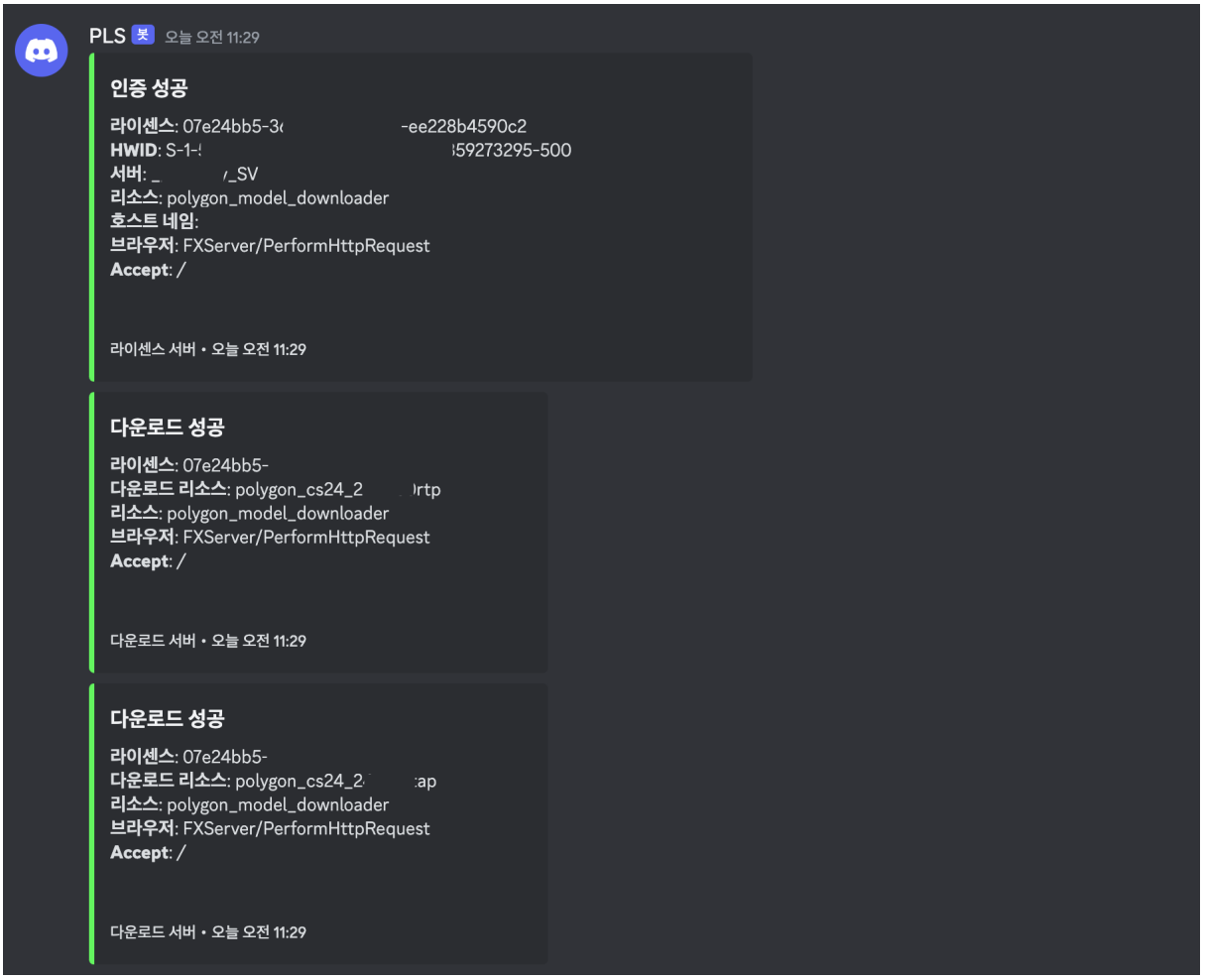
그 다음 서버에서 다운로드 한 파일을 사용자의 서버 구동 과정에서 로딩하고, assert 함수를 통해 ./polygon_system의 하위 파일들을 사용자의 개입 없이 재귀적으로 삭제한다.

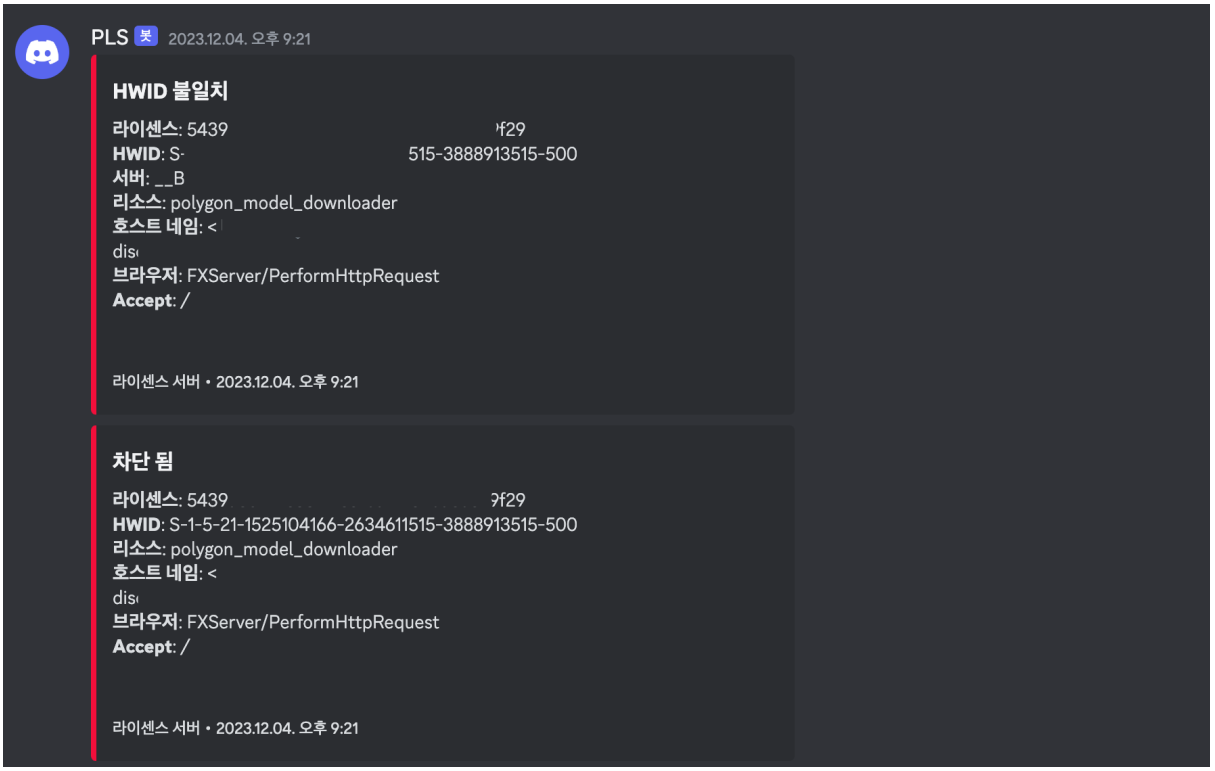
```
if "version" in datas:
        version = datas["version"]
        if version == "2.0":
            # SQLite 데이터베이스에 연결
            connection = sqlite3.connect('/데이터베이스.db')
            connection.row_factory = sqlite3.Row
            cursor = connection.cursor()
            # 라이센스 키를 데이터베이스에서 조회
            cursor.execute('SELECT * FROM polygon_manager W
HERE key = "' + license + '"')
            dbData = cursor.fetchall()
```

Server에선 Agent의 버전이 최신 상태인지 확인하고 데이터베이스에 연결해 전달받은 라이센스키가 존재하는지 확인한다.

그리고 모든 인증 과정 및 서버 접근에 대한 로그는 디스코드의 관리 채널 웹훅으로 전송된다.

```
# 인증 성공 Discord 웹훅을 보냄
webhook = DiscordWebhook(url='https://discord.com/api/webho
oks/웹훅URL')
embed = DiscordEmbed(title="**인증 성공**", description="**라
이센스**: %s\n**HWID**: %s\n**서버**: %s\n**리소스**: %s\n**호
스트 네임**: %s\n**브라우저**: %s\n**Accept**: %s" % (license,
hwid, dbData[n], name, hostname, browser, accept), color='4
2f554')
embed.set_footer(text='라이센스 서버')
embed.set_timestamp()
webhook.add_embed(embed)
webhook.execute()
```

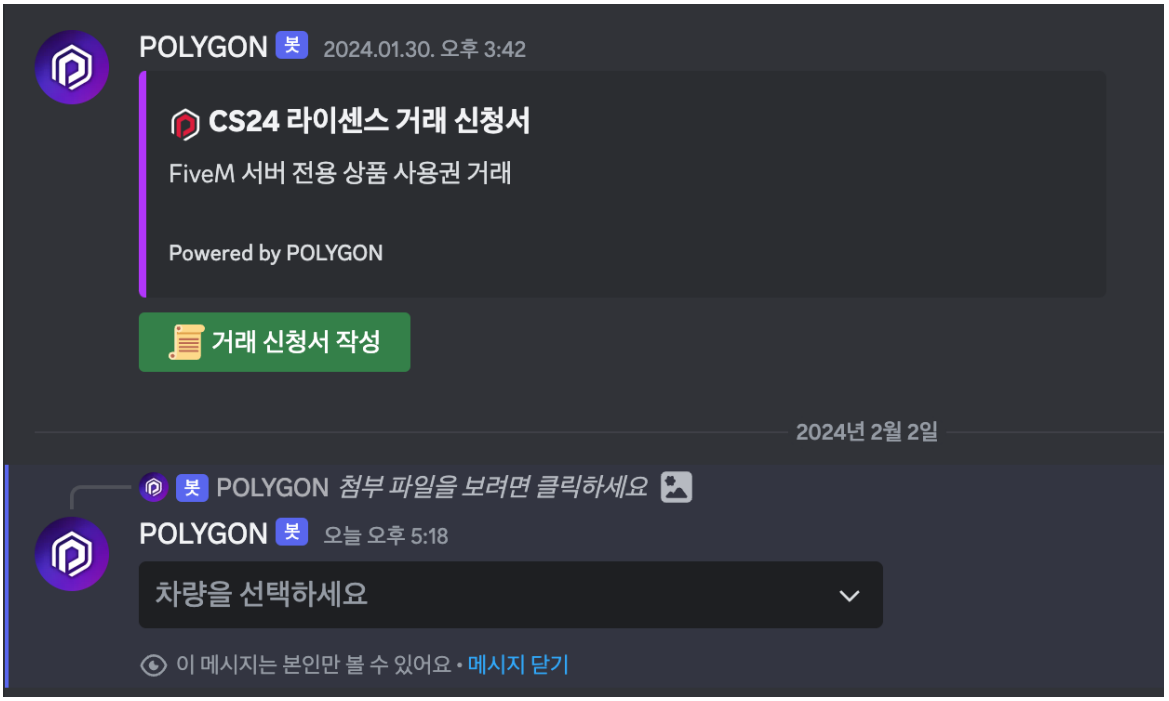

PLS 봇 오늘 오전 11:29
인증 성공
라이센스: 07e24bb5-3(-ee228b4590c2
HWID: S-1-! 59273295-500
서버: _ /_SV
리소스: polygon_model_downloader
호스트 네임:
브라우저: FXServer/PerformHttpRequest
Accept: /
라이센스 서버 • 오늘 오전 11:29
다운로드 성공
라이센스: 07e24bb5-
다운로드 리소스: polygon_cs24_2)rtp
리소스: polygon_model_downloader
브라우저: FXServer/PerformHttpRequest
Accept: /
다운로드 서버 • 오늘 오전 11:29
다운로드 성공
라이센스: 07e24bb5-
다운로드 리소스: polygon_cs24_2 :ap
리소스: polygon_model_downloader
브라우저: FXServer/PerformHttpRequest
Accept: /
다운로드 서버 • 오늘 오전 11:29


PLS 봇 2023.12.04. 오후 9:21
HWID 불일치
라이센스: 5439)f29
HWID: S- 515-3888913515-500
서버: __B
리소스: polygon_model_downloader
호스트 네임: <
dis
브라우저: FXServer/PerformHttpRequest
Accept: /
라이센스 서버 • 2023.12.04. 오후 9:21
차단 됨
라이센스: 5439)f29
HWID: S-1-5-21-1525104166-2634611515-3888913515-500
리소스: polygon_model_downloader
호스트 네임: <
dis
브라우저: FXServer/PerformHttpRequest
Accept: /
라이센스 서버 • 2023.12.04. 오후 9:21

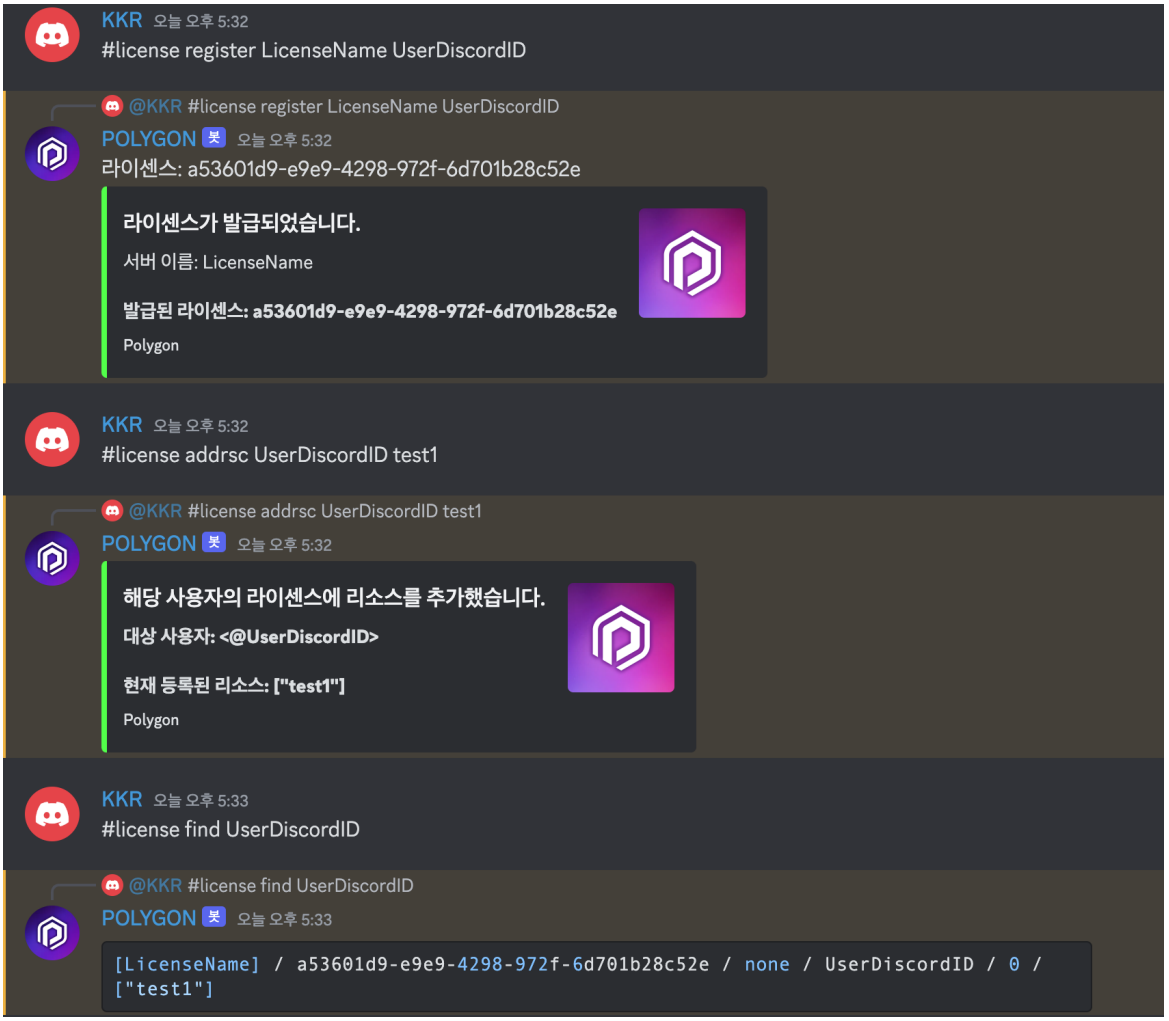
추가적인 절차의 간소화

기존의 사용자와의 직접 소통을 통해 라이센스 사용권한을 부여하고 데이터베이스에 등록하는 절차를 간단히 하여 편의성을 높일 수 있는 방안을 구상했다.

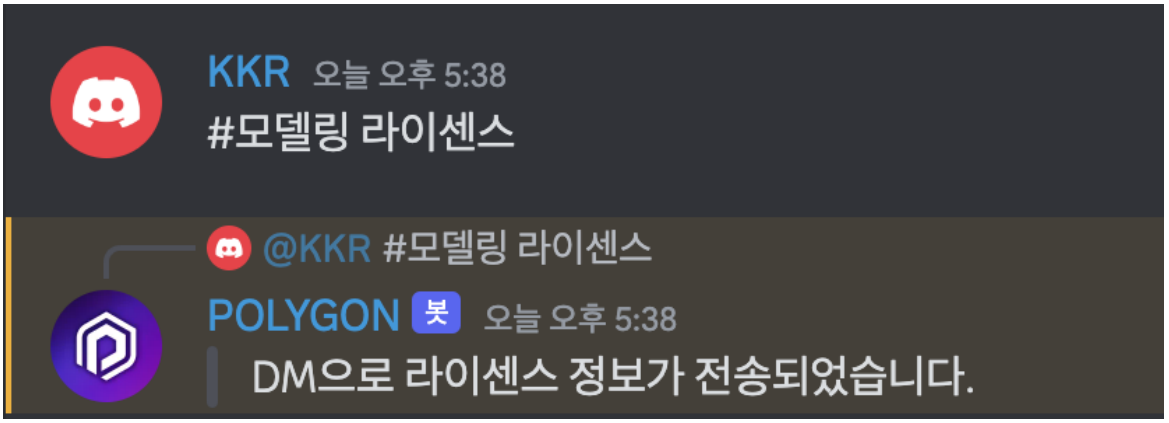
discord.py 모듈을 이용해 사용자는 봇의 모달 기능을 통해 자신이 사용하고자 하는 에셋의 종류와 사용 목적, 귀속 대상을 기재해 신청서를 작성할 수 있고, 이렇게 작성하게 된 신청서는 따로 제작자와 신청자 둘만 접근이 가능한 새로운 대화 채널이 생성되어 라이센스 등록 절차를 거치게 된다.



다음으로 에셋 제작자는 다양한 명령을 통해 리소스 업로드, 더미리소스 업로드, 라이센스 등록 및 관리 등의 명령어를 사용할 수 있다.



또한 신청자는 자신이 보유한 리소스의 목록과 라이센스 정보, 더미리소스 다운로드가 가능한 명령이 가능하다.

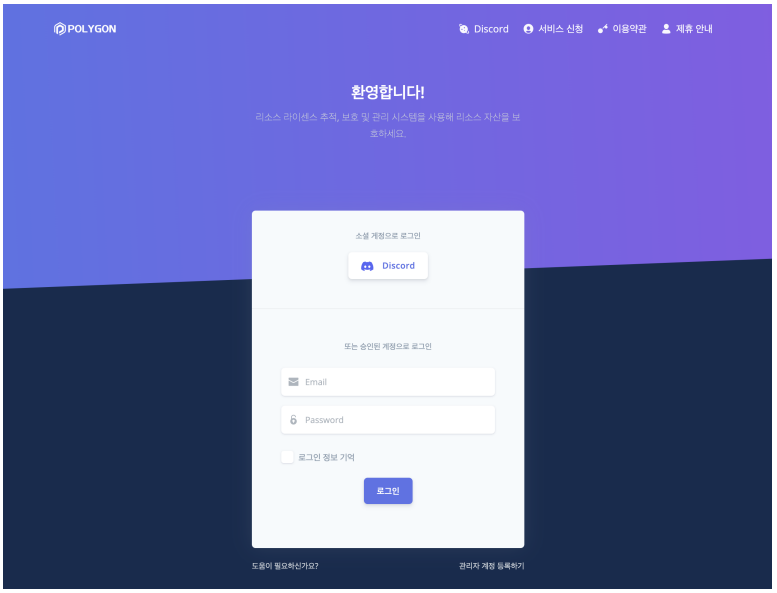


앞으로의 계획과 과제

현재 비슷한 불편함을 느껴왔던 다른 에셋 제작자를 위해 서비스를 유지하고 있으며, 리소스, 라이센스, 사용자 관리를 효율적으로 할 수 있도록 대시보드 형태의 관리페이지를 구축해 보다 완성도 있는 프로젝트로 남기고싶다.

또한 서비스를 운영해보며 비정상적인 접근 또는 우회 시도는 없었지만, 보안 헤더나 엑세스 제어 그리고 외부 입력과 데이터의 유효성에 대한 보안이 부족하다고 느껴 취약점을 더 세부적으로 분석하고 개선해 나갈 필요가 있다. 그리고 단순히 파일 접근에 대한 DRM 시스템이 아닌 파일 자체적인 암호화를 적용해보고싶다.

> 피그마 템플릿을 사용한 대시보드 로그인 접근 페이지 구상도



대시보드 페이지 구상도


POLYGON
대시보드
라이센스 관리
리소스 관리
고객 관리
라이센스 로그
봇 관리
마이페이지
DOCUMENTATION
서비스 가이드
POLYGON 규정
관리자 페이지
대시보드
Search
KKR
주간 라이센스 사용량
3,508 MB
3.48% Since last month
등록된 라이센스
236 건
3.48% Since last week
등록된 리소스
924 개
1.10% Since yesterday
인증 서버 상태
정상
12% Since last month
OVERVIEW
리소스 등록 수 현황
Month
Week
에셋 등록량
Jul
Aug
Sep
Oct
Nov
Dec