**프로젝트 명세서**

**웹 IoT 프로젝트**

**“ PLANTinum, IoT 플랫폼 ”**

**A109팀**

팀장 임진경

팀원 김유민

팀원 문요성

팀원 이지수

팀원 정진아

# 목차

1. 프로젝트 개요 ………………………………………………………………………… 3
2. 소개 …………………………………………………………………………… 3
3. Sub PJT 1. Back-End ……………………………………………………… 3
4. Sub PJT 2. Front-End ……………………………………………………… 3
5. Sub PJT 3. Hardware…………………………………………………………4
6. 프로젝트 목표 ………………………………………………………………………… 5
7. 필수 지식 습득 ………………………………………………………………………… 5
8. 과제 명세 ……………………………………………………………………………… 7
9. Back-End ………………………………………………………………………7

Req. 1. 회원관리

Req. 2. 식물등록

Req. 3. 커뮤니티

Req. 4. 프로필

Req. 5. 관리자

1. Front-End ……………………………………………………………………11

Req. 1. 레이아웃

Req. 2. 메인 화면

Req. 3. 커뮤니티 화면

Req. 4. 반응형 웹

Req. 5. 회원관리

Req. 6. 프로필

Req. 7. 당근 페이지

1. Hardware …………………………………………………………………… 14

Req. 1. Raspberry Pi OS 설치

Req. 2. 환경 세팅

Req. 3. 회로도 작성

Req. 4. 서버 연결

Req. 5. GUI 구성

Req. 6. 클래스 및 함수 목록

Req. 7. 자동 실행

1. 프로젝트 실행 및 배포 …………………………………………………………………18
2. 기술 스택 ……………………………………………………………………………… 21
3. 별첨 ………………………………………………………………………………………22

# 

# 1. 프로젝트 개요

## 소개

PLANTinum이란 Plant(식물)와 Platinum(백금)의 합성어를 의미합니다. 또한, 플랫폼을 통해 계획적(Plan)으로 식물을 관리하고 보살핀다는 의미를 담고 있기도 합니다. 본 프로젝트에서는 반려 식물을 케어해주는 자동화된 하드웨어 시스템과 웹 커뮤니티를 통합한 플랫폼 서비스를 제공하는 『IoT 플랫폼』 구현을 목표로 합니다.

“플랫폼 연동 IoT 서비스” PLANTinum은 사용자가 IoT 기기 화면을 통해 화분 내의 토양 상태를 확인하고, 자동화된 케어 서비스를 제공하도록 하는 몇 가지의 선택 항목을 가집니다. 연동된 웹 플랫폼에서는 식물 상태와 서비스 기록을 확인할 수 있으며 비슷한 다수의 사용자가 경험과 소통을 나눌 수 있는 커뮤니티를 제공합니다.

프로젝트의 진행은 3개의 Sub PJT로 진행되며, 각 Sub PJT의 진행은 아래 표와 다음 각 단계를 설명한 내용을 참고하시기 바랍니다.

|  |  |  |
| --- | --- | --- |
| Sub PJT 1. BE | Sub PJT 2. FE | Sub PJT 3. HW |
| * Django로 백엔드 구성 * MySQL로 DB 구성 * 게시글 관리 | * Window에 기반한 환경 구성 * Vue로 온라인 웹 플랫폼 제작 * 반응형 웹 | * Raspberry Pi로 하드웨어 구성 * API 제작 * BE 서버 연결 |

## Sub PJT 1. Back-End

Sub PJT 1에서는 ERD를 그리며 데이터베이스 구조를 설계하고 Django를 사용하여 서비스의 백엔드 영역을 구축합니다. AWS의 S3를 이용해 이미지 업로드 기능을 구현합니다. 마찬가지로 AWS EC2를 통해 서버를 관리하게 됩니다. 주 기능으로는 회원관리, 식물 페이지, 커뮤니티, 프로필 관리, 관리자 페이지 등이 있습니다. 또한 IoT 기기가 받아오는 센서 값들을 DB에 저장하는 역할을 하기도 합니다.

## Sub PJT 2. Front-End

Sub PJT 2에서는 Vue를 사용하여 서비스의 프론트 영역을 구현합니다. 화면은 메인페이지와 내식물 페이지, 커뮤니티, 프로필 페이지로 나뉘며 각 화면은 반응형으로 웹과 모바일에 차이가 둡니다. 관리자용 페이지가 따로 존재합니다. 구현이 완료되었다면 부가 기능으로 식물 거래를 돕는 당근 페이지를 구현합니다.

## Sub PJT 3. Hardware

Sub PJT 3에서는 라즈베리파이와 LCD 모니터를 사용하여 서비스의 하드웨어를 구현합니다. LCD 화면에 식물의 토양 습도와 온도, 기존 기록 등의 정보를 출력하며 웹과 연동하여 식물의 상태를 기록할 수 있습니다. 토양습도 센서, 온도 센서, 워터 필터 등 다양한 센서와 라즈베리파이를 조합하여 뼈대를 제작한 후, 3D 프린터로 프레임을 출력합니다. 각 센서들이 수집한 데이터들은 정해진 시기에 맞춰 서버로 전송됩니다.

# 2. 프로젝트 목표

1. Django와 Vue를 활용한 IoT 플랫폼 구축
2. 임베디드 KIT를 이용한 IoT 서비스 구현
3. AWS 서버를 통해 배포

# 3. 필수 지식 학습

프로젝트 진행을 위한 자기주도 학습에 실마리가 될 수 있는 필수 학습 대상 키워드와 유용한 참고 자료 링크 목록이 아래 표에 제시되어 있습니다. 아래 목록에 제시된 내용은 기초적인 내용에 해당하는 예이며, 이외 추가로 필요한 항목들을 직접 찾아 학습해 나가시기 바랍니다.

|  |  |  |
| --- | --- | --- |
| 주제 | | 참고 링크 |
| 반응형 SPA | ES6 | JavaScript의 Array 객체에 대한 소개  <https://developer.mozilla.org/ko/docs/Web/JavaScript/Reference/Global_Objects/Array> |
| React | React 공식 문서(React 기본 예시, CRA, JSX, Hook, Context 등)  <https://ko.reactjs.org/docs/hello-world.html> |
| SPA | SPA 개념 설명  <https://linked2ev.github.io/devlog/2018/08/01/WEB-What-is-SPA/> |
| 반응형 웹 | 반응형 웹 디자인 패턴  <https://developers.google.com/web/fundamentals/design-and-ux/responsive/patterns?hl=ko> |
| MariaDB | Maria DB 공식 문서  <https://mariadb.org/documentation/> |
| 임베디드 키오스크 제작 | Raspbian | Raspbian OS 설치 가이드  <https://geeksvoyage.com/raspberry%20pi4/installing-os-for-pi4/> |
| Arduino | 아두이노 기초  <http://blog.naver.com/PostView.nhn?blogId=jamduino&logNo=220812436210> |
| Arduino 연결 | Node.js에서 시리얼 포트를 활용한 아두이노 연결  <https://linked2ev.github.io/devlog/2018/08/01/WEB-What-is-SPA/> |
| Express.js | Express.js의 이해 및 설치  <http://webframeworks.kr/getstarted/express.js> |
| Material-UI | Material UI Theme 사용법  <https://bitly.kr/1LmbXZH> |
| pm2 | 프로세스 관리툴 중 하나인 pm2의 사용법 소개  <https://blog.outsider.ne.kr/1197> |
| crypto | 암호화 모듈 crypto 사용  <https://nodejs.org/api/crypto.html> |
| 임베디드 키오스크 완성 | 인증 | 인증과 권한의 개념  <https://hanee24.github.io/2018/04/21/authentication-authorization/> |
| JWT | JWT 장단점 및 토큰 구성 개념  <http://www.opennaru.com/opennaru-blog/jwt-json-web-token/> |
| Socket | Socket의 정의 및 사용법 이해  <https://en.wikipedia.org/wiki/Network_socket> |
| SSR | SPA에서의 SSR과 CSR  <https://bit.ly/2NK7bfp> |
| Next.js | Next.js 소개  <https://nextjs.org/docs/getting-started> |
| CSS Animation | CSS Animation 소개  <https://developer.mozilla.org/ko/docs/Web/CSS/CSS_Animations> |
| Infinite Scroll | React에서의 Infinite Scroll 구현  <https://reactgo.com/react-infinite-scroll/> |

# 4. 과제 명세

## Sub PJT 1. Back-End

### Req. 1. 회원관리

회원가입 및 로그인 페이지 관리

|  |  |
| --- | --- |
| 제목 | 웹 - 로그인 페이지 |
| 구성도 |  |
| 설명 | * 로그인 * 회원가입 * 아이디/패스워드 찾기 |

Req. 1-1. 회원 가입 기능 구현

|  |  |
| --- | --- |
| 기능 상세 | * 회원가입 API 구현 * ID, e-mail, 패스워드 필수 입력 * 랜덤 닉네임 자동 생성 * DB의 컬럼 조건에 따라 입력 시 형식 체크(정규식) * email 중복 시 가입 불가로 설정 * 패스워드는 암호화하여 통신 및 DB에 저장 |

Req. 1-2. 회원 로그인, 로그아웃 기능 구현

|  |  |
| --- | --- |
| 기능 상세 | * 회원 로그인 API 구현 * ID, e-mail, 패스워드를 필수 입력 항목으로 설정 * 패스워드는 암호화하여 통신 |

Req. 1-2. 회원 비밀번호 찾기 기능 구현

|  |  |
| --- | --- |
| 기능 상세 | * 비밀번호 분실 시 이메일을 통해 찾기 가능 |

### Req. 2. 식물 등록

|  |  |
| --- | --- |
| 기능 상세 | * 나의 식물 등록하기 * 등록 시 랜덤 값 생성-중복없이, 20초 안에 등록, 이후 삭제 * 이름, 심은 날짜 설정 * 토양 수분 체크 * 최근 물주기 날짜 |

Req. 2-1. 등록한 식물 상세 페이지

|  |  |
| --- | --- |
| 기능 상세 | * 이름, 심은 날짜 * 토양 수분 기록 |

Req. 2-1. 실내 정원용 식물 정보 등록

|  |  |
| --- | --- |
| 기능 상세 | * 농촌진흥청 농사로에서 제공하는 공공데이터 사용 * 식물 이름, 물주기 조건(봄, 여름, 가을, 겨울) |

### Req. 3. 커뮤니티

|  |  |
| --- | --- |
| 기능 상세 | * 게시글 제목, 내용, 작성자, 작성 시간(최신 순으로) * 댓글, 대댓글 * 카테고리 정렬 * 지역에 따른 카테고리 분류 * 게시글 검색 * 디폴트 이미지 설정 |

### Req. 4. 프로필

|  |  |
| --- | --- |
| 기능 상세 | * 유저가 작성한 게시글 목록 조회 기능 * 유저가 등록한 식물 조회 기능 * 개인정보(비밀번호) 수정 기능 |

### Req. 5. 관리자

|  |  |
| --- | --- |
| 기능 상세 | * 회원 관리 * 게시글 관리 |

### Req. 6. 추가기능

|  |  |
| --- | --- |
| 기능 상세 | * 즐겨찾기 및 좋아요 * 즐겨찾기 등록한 게시글 조회 |

## 

## Sub PJT 2. Front-End

### Req. 1. 레이아웃

|  |  |
| --- | --- |
| Req. 1-1 | 레이아웃 내용 구성 |
| 기능 명세 | * Header에 로고, 메뉴, 회원관리, 검색 버튼 배치 * Footer에 팀 정보 추가 * 메인 화면의 구성요소 가운데 정렬 |

### Req. 2. 메인 화면- 식물 등록

|  |  |
| --- | --- |
| Req. 2 | 메인화면 |
|  | |

### 

|  |  |
| --- | --- |
| Req. 2-1 | 버튼 및 메뉴에 주요 이벤트 기능 구현 |
| 기능명세 | * 로고 클릭 시 홈화면으로 이동 * login / signup 버튼 클릭 시 해당 페이지로 이동 * search 바를 통해 유저가 등록한 식물 목록 조회 가능 * 물주기, 식물당근, 커뮤니티, 내프로필 등 다른 유저와의 interaction 페이지로 이동 * 등록된 식물을 관리페이지에 보여주며, 등록된 식물이 없을 경우에는 동그라미 안에 + 버튼으로 식물 추가할 수 있도록 구현 * 등록된 식물의 그리드를 클릭함으로써 상세 페이지 접근 |

### Req. 3. 커뮤니티 화면

|  |  |
| --- | --- |
| Req. 3 | 커뮤니티 화면 |
|  | |

|  |  |
| --- | --- |
| Req. 3-1 | 커뮤니티 UI 구현 |
| 기능 명세 | * 모든 유저가 게시글 열람 가능하도록 구현 * 사진 클릭 시 게시글 상세 페이지로 이동 * 인증된 유저에게만 게시글 작성 루트 노출 * 앨범형으로 구성 (default image) * 인증된 유저는 게시글에 댓글 작성 및 수정, 삭제 가능 |

|  |  |
| --- | --- |
| Req. 3-2 | 커뮤니티 추가 |
| 기능 명세 | * 거래를 위한 채팅 서비스 지원 |

### Req. 4. 반응형 웹

|  |  |
| --- | --- |
| Req. 4-1 | 모바일 사이즈일 때 메뉴 사이드 바 노출 |
|  | |
| 설명 | * PC화면에서 메뉴 아이콘 비활성 기능 추가 * 화면 사이즈를 변경해 기능 동작 확인 |

### Req. 5. 회원관리

|  |  |
| --- | --- |
| Req. 5-1 | 회원 가입 UI 구현 |
| 설명 | * 회원 관리 모달 구현 * 모달 안에 회원 관리 UI 구현 |

|  |  |
| --- | --- |
| Req. 5-2 | 회원 로그인 UI 구현 |
| 설명 | * 회원 관리 모달 안에 회원 로그인 UI 구현 |

### Req. 6. 프로필

|  |  |
| --- | --- |
| Req. 6-1 | 유저 정보 변경 UI 구현 |
| 설명 | * 프로필 모달 구현 * 유저 정보 변경 UI 구현 |

|  |  |
| --- | --- |
| Req. 6-2 | 비밀번호 변경 UI 구현 |
| 설명 | * 비밀번호 변경 UI 구현 |

### Req. 7. 일지 페이지

|  |  |
| --- | --- |
| Req. 7-1 | 다이어리 형식의 일지 페이지 구현 |
| 설명 | * 각 게시글에 반응 표시 기능 구현 (ex. 좋아요) |

## Sub PJT 3. Hardware

### Req. 1. 라즈베리파이 OS 설치 및 초기 환경설정

|  |  |
| --- | --- |
| Req. 1-1 | sd카드에 라즈베리파이 os설치 |
| 설명 | * raspberry pi imager 설치 * sd 카드 포맷 후 os설치 |

|  |  |
| --- | --- |
| Req. 1-2 | 모니터를 이용한 초기 설정 |
| 설명 | * 국가, 언어 설정 * 계정 해상도 설정 * WIFI 연결 후 IP확인 * SSH, VNC, SPI, I2C, Serial Port 등 Enable설정 |

|  |  |
| --- | --- |
| Req. 1-3 | 원격제어 연결 |
| 설명 | * VNC Viewer 설치 * IP확인 후 연결 |

### Req. 2. 환경 세팅(라즈베리파이 쉘)

|  |  |
| --- | --- |
| Req. 2-1 | 설치 라이브러리 목록 |
| 설명 | * 라이브러리 설치 전 sudo apt-get update && sudo apt-get upgrade * sudo apt-get install python3.7 * sudo apt-get install vim * vim 설치 후 .vimrc 파일 작성 필요 개인 취향에 맞춰서 작성 * git 설치 (최신 버전으로 받으려면 의존성 패키지 먼저 설치 ) |

### Req. 3. 회로도 작성

|  |  |
| --- | --- |
| Req. 3-1 | 필요 HW 목록 |
| 설명 | * raspberry pi 4 model B * 토양감지센서 * mcp 3008 (디지털 컨버터) x 2 (토양감지센서 1, 수위센서 1) * DHT11(온습도센서) * 네오픽셀 LED, 저항 * lcd monitor 7인치 * 수위센서 * 워터펌프, 모터드라이브 * 브레드보드 |

|  |  |
| --- | --- |
| Req. 3-2 | 회로구성 |
| 설명 | * HW 회로 구성 * 간단한 동작 코드로 각 회로 연결 테스트 |

|  |  |
| --- | --- |
| Req. 3-3 | 기능 관련 |
| 설명 | * date time으로 급수 시기 조절 * 데이터 측정 시간 주기 조절(하루에 두 번?) * DB에는 물 주는 시간만 넘기기 * LCD monitor를 통해서 터치로 조작 가능 |

### Req. 4. 서버 연결

|  |  |
| --- | --- |
| Req. 4-1 | DB 연결 |
| 설명 | * mysql-server && mysql -client 설치 * mysql connector 라이브러리를 이용해서 DB 연결 예정 * db, cur 변수 활용해 db접근 및 쿼리문 전송 |

|  |  |
| --- | --- |
| Req. 4-2 | 데이터 송수신 |
| 설명 | * AWS ubuntu server를 기반하여 MySQL Server 와 통신 * 식물 정보 및 물주기 정보를 받아와 하드웨어에 등록 |

### Req. 5. GUI 구성

|  |  |
| --- | --- |
| Req. 5-1 | 터치 스크린 GUI 연결 |
| 설명 | * 라즈베리파이에 Qtside2 설치 * windows 에서 작성한 python코드 옮겨오기 |

### Req. 6. 클래스 및 함수 목록 및 기능

1. 식물 정보 등록 시 랜덤 코드(unique)를 DB attribute 하나 추가
2. 해당 primary key를 웹페이지에서 확인이 가능해야됨
3. 해당 코드를 화분에 입력 시 식물정보 가져옴

|  |  |
| --- | --- |
| 함수 | 기능 |
| init() | * 각종 센서 및 사용할 DB init * GPIO Pin Number setting |
| plant\_registration() | * 식물 정보 등록(OTP 코드 이용예정) |
| get\_info() | * 파일을 읽어 처음 실행하는 것인지 확인 * 처음 실행시 code로 정보 받기 * 처음 실행이 아닐 시 파일에 등록되어있는 정보 가져오기 |
| Check\_Time() | * 해당 식물의 물 주기에 맞는 시간 측정 |
| control\_watering() | * 서보모터 작동해서 물주기 * 물 주는 작업 끝날때 check\_remaining() 실행 * if( check\_remaining() < 100ml ) watchOut() 실행 |
| control\_nutrition() | * 서보모터 작동해서 영양제 주기 * 물 주는 작업 끝날때 check\_remaining() 실행 * if( check\_remaining() < 100ml ) watchOut() 실행 |
| check\_remaining() | * 수위센서 활용해서 잔량 확인 |
| watchOut() | * lcd화면에 물 충전 알림 or 웹페이지에 알림 |
| control\_light() | * 파라미터로 led 제어 * light(0) => led off * light(1) => led on |
| get\_wheater() | * 실시간 날씨정보 받아오기(보류) |
| get\_sensingData() | * 조도센서, dht11로 온습도,조도 정보 받아오기 |
| send\_sensingData() | * DB에 정보 업데이트 |

### Req. 7. 자동실행

|  |  |
| --- | --- |
| Req. 7 | 전원 공급 시 프로그램 자동 실행 |
| 설명 | * crontab, rc.local 등 맞는 방법 선택 |

### Req. 8. 센서 Pin

|  |  |
| --- | --- |
| 이름 | 필요 Pin |
| LCD monitor | * GND * VCC |
| 모터드라이브  (L9110S) | * GND * VCC * Pin 2개 |
| 워터펌프 | * 모터드라이브에 +, - 핀 연결 |
| DHT11 | * GND * VCC * Data Pin 1 |
| 네오픽셀 LED | * GND * VCC * IN |
| 수위센서(접촉) | * GND * VCC * mcp3008에 Data Pin 1 연결 |
| 토양수분센서 | * GND * VCC * mcp3008에 Data Pin 1 연결 |
| mcp3008(디지털 컨버터) | * GND 2 * VCC 2 * Data Pin 4 |

# 5. 프로젝트 실행 및 배포

## EC2 인스턴스

Amazon Elastic Compute Cloud(Amazon EC2)는 500개가 넘는 인스턴스, 그리고 최신 프로세서, 스토리지, 네트워킹, 운영 체제 및 구매 모델의 옵션과 함께 워크로드의 요구 사항에 가장 잘 부합할 수 있도록 가장 포괄적이고 심층적인 컴퓨팅 플랫폼을 제공합니다.[[1]](#footnote-1) AWS EC2 인스턴스를 통해 리눅스 기반 Ubuntu20.04LTS 서버를 구축하고 MobaXterm의 SSH 통신을 통해 인스턴스에 접속합니다.

## Nginx&uWSGI를 이용한 프로세스 관리

Nginx는 동시접속 처리에 특화된 웹서버 프로그램으로 다수의 사용자가 웹에 접속할 경우에 권장되는 사양입니다. 비동기 처리 방식을 채택하여 사용자들의 요청 간의 버퍼링과 대기 시간을 줄일 수 있다는 장점이 존재합니다. AWS 인스턴스 상에 Nginx를 설치하여 다수의 프로그램을 중계합니다. $ sudo apt-get install nginx 명령어를 통해 Nginx 설치 후 환경설정을 세팅할 수 있습니다.

Nginx는 Node.js 환경에서 Proxy 서버로 활용될 수 있으나 Python을 사용하는 Django는 Nginx를 proxy 서버로 이용할 수 없습니다. 따라서 본 프로젝트에서는 Nginx와 Django를 이어주는 중간 다리 역할로서 uWSGI를 채용합니다.

uWSGI는 python 패키지를 사용할 수 있는 애플리케이션 컨테이너 서버로 애플리케이션 호출을 통해 애플리케이션 서버가 코드와 통신할 수 있도록 합니다. 일반적으로 서버가 접근할 수 있는 python 모듈에서 ‘애플리케이션’이라는 이름의 객체로 제공되곤 합니다. 본 프로젝트의 uWSGI는 실제 물리 서버가 아닌 가상 인스턴스(venv)에서 작동합니다.

|  |
| --- |
| $ cd /etc/nginx/sites-available  $ vi default  #default  …  server {  listen 80 default\_server;  } |

위와 같이 Nginx의 default 파일을 수정하면 localhost의 8000번 포트 대신 Nginx에서 입력한 80번의 포트 번호로도 페이지에 접속할 수 있습니다.

## S3 이미지 등록

Amazon Simple Storage Service(Amazon S3)는 AWS에서 제공하는 객체 스토리지 서비스입니다. 스토리지 내에서 이미지를 포함한 데이터를 저장하고 보호할 수 있습니다. 본 프로젝트에서는 웹서비스와 커뮤니티 내에서 사용되는 다양한 이미지 파일들을 S3에 올림으로써 DB가 무거워지는 것을 방지하여 서비스 성능을 향상시키고자 합니다. 대표적으로 본 프로젝트의 S3 스토리지 내에는 plantinum 웹 서비스에서 기본적으로 제공하는 이미지가 포함되어 있습니다.

## 프로그램 배포

예전으로 거슬러 올라가면 서버를 점검하고 수정할 때마다 서버를 정지시키고 재실행해서 변경사항을 적용하곤 했습니다. 이건 당연히 거대한 서비스상의 결함입니다. 서버가 재실행되는 동안은 사용자들이 접속을 할 수 없고, 처리 중에 있던 서비스가 중간에 중단될 수 있기 때문입니다.

따라서 웹의 배포는 백그라운드 작업으로 진행되며, 지속적인 서비스를 고려한 ‘무중단 배포’를 목표로 합니다. 이를 위해서 우선 SSH 터미널 상에서 소스를 수정해서 저장하고, runserver를 통해 서버를 실행합니다. Ctrl+Z를 눌러 실행되던 프로그램을 일시 정지 시킨 후 shell로 돌아와 $ bg명령어를 작성하면 해당 프로그램이 백그라운드에서 작동하게 됩니다. 추가적으로 $ disown -h 명령어를 작성하면 ssh 연결이 끊어져도 프로세스가 종료되지 않습니다. restart는 프로세스를 무조건 kill한 후에 시작하기 때문에 사용해서는 안 됩니다.

단순한 상황에서는 이것만으로 중단이 사라지지만 이때 몇 가지 구멍에서 끊김 현상이 발생할 수 있습니다.

1. new\_app이 완전히 동작하기 전에 요청을 받을 경우
2. old\_app이 요청 처리 도중에 SIGKILL로 죽을 경우
3. HTTP Keep-Alive 가 사용될 경우

이러한 경우 발생할 수 있는 일부 사태를 방지하기 위해 Redis를 사용합니다.

## Redis 저장소

Redis는 지속성을 보장하는 In-메모리 데이터베이스입니다. 짧은 대기 시간 대비 높은 데이터 처리율을 보이며 멀티 스레드를 사용하여 메모리 관리에 용이합니다. 관계형 데이터베이스인 mySQL과 달리 Redis는 비관계형 데이터베이스이며 데이터에 유효 기간을 두고 있습니다. 일정 기간이 지나면 오래된 데이터를 자동으로 삭제하여 데이터 용량 관리에도 용이합니다. 또한, 데이터가 key-value의 dictionary 형식으로 이루어졌기에 mySQL보다 빠른 속도를 자랑합니다. 이러한 특징들로 인해 채팅 주로 시스템 구현에 이용됩니다.

그 중에서도 Redis의 데이터 유효 기간은 데이터베이스 관리자에 의해 조절 가능합니다. 이를 TTL(Time To Live)라고 부르며 본 프로젝트에서는 OTP 데이터의 관리를 위해 mySQL과 함께 Redis 데이터 베이스를 일부 채용합니다.

OTP 코드가 발급된 채로 서버가 다운될 시, 생성된 OTP는 서버가 복구된 후 IoT 기기가 연결되지 않은 상황에서도 삭제되지 않고 mySQL의 스키마에 남아 새로운 OTP 발급을 저해합니다. 이러한 사태를 해결하기 위해 Redis의 TTL을 사용합니다. Redis는 서버가 다운되어도 TTL에 지정된 시간이 지나면 자동적으로 데이터가 삭제되며 새로운 OTP 발급을 가능케 합니다. 이를 통해 서버가 다운됐을 때, OTP의 재발급이 불가능한 문제를 해결할 수 있습니다.

# 6. 기술 스택

| **PART** | 기술 스택 / HW | 상세 내용 |
| --- | --- | --- |
| 형상관리 | GITLAB | * https://lab.ssafy.com/s07-webmobile3-sub1/S07P12A109.git |
| 이슈관리 | JIRA | * S07P11A109 * S07P12A109 |
| 커뮤니케이션 | Mattermost |  |
| 디자인 | 포토샵/일러스트 |  |
| Figma |  |
| OS | Window | * Windows 10 Enterprise |
| Linux |  |
| DB | MySQL | * MySQL 8.0.29 (64bits) |
| S3 |  |
| Redis |  |
| Server | AWS EC2 ubuntu |  |
| Ngnix |  |
| FE | Vue |  |
| BE | Django |  |
| HW | Raspberry Pi | * Raspberry Pi 4 model B |
| PySide6 |  |
| IDE | Visual Studio Code | * Visual Studio Code 1.68.1 |
| MobaXterm |  |

# \* 별첨

각 명세서에 첨부된 별첨 목록을 첨부합니다.

## 웹 IoT 명세서 1. 반응형 SPA 제작

* Chocolatey 설치
* Node.js 설치
* VS Code 설치
* 프록시 서버 NginX 설치

## 웹 IoT 명세서 2. 웹 서버 및 임베디드 기반 키오스크 제작

* Raspbian OS 설치
* Raspbian OS 환경설정
* Maria DB 설치
* 미들웨어 활용

1. AWS EC2, <https://aws.amazon.com/ko/ec2/features/?trk=ec2_landing>, 2022.08.15 [↑](#footnote-ref-1)