시스템 분석 설계
개인 포토폴리오

20181765 김건호

시스템 분석설계 개인 포토폴리오

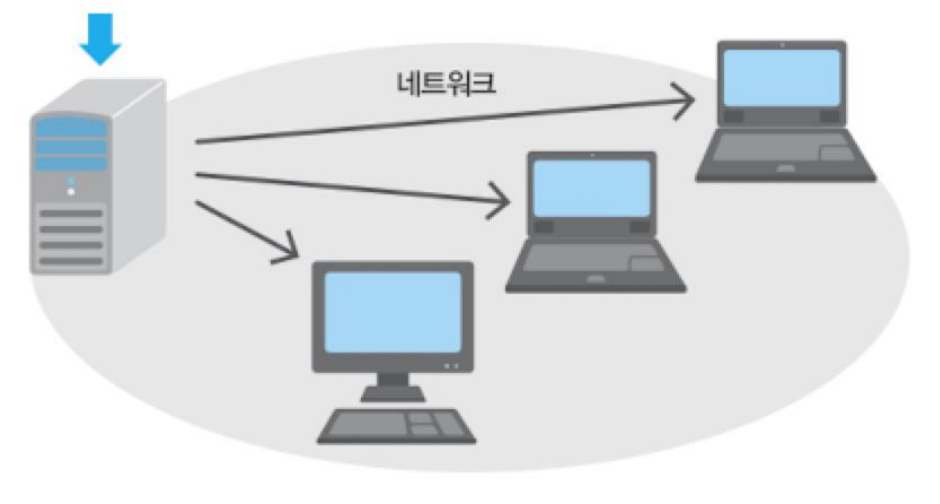
목차

01 Server란? 1-1 서버와 클라이언트

사전적 의미로
서버는 클라이언트에게 네트워크를 통해 정보나 서비스를 제공하는 컴퓨터 시스템으로 컴퓨터 프로그램(server program) 또는 장치(device)를 의미
특히, 서버에서 동작하는 소프트웨어를 서버 소프트웨어(server software)라 한다. 주로 리눅스 등의 운영 체제를 설치한 대형 컴퓨터를 쓰지만, 그렇지 않은 경우도 있다.

컴퓨터에게 서버는 클라이언트에게 서비스를 제공해주는 매개체이다.
Server는 serve + er를 붙인 단어인데, 그 말은 즉, 서버는 클라이언트 에게 여러가지 서비스를 제공하는 것을 의미한다.

예를 들어 아래 사진을 보면 클라이언트에 해당하는 것은 좌측에 있는 것이 서버
네트워크를 타서 우리가 사용하는 홈페이지(웹 브라우저) 등이 클라이언트.



서버와 클라이언트의 예시

01 Server란? 1-2 웹 서버 구동 방식

웹 서버의 구동 방식

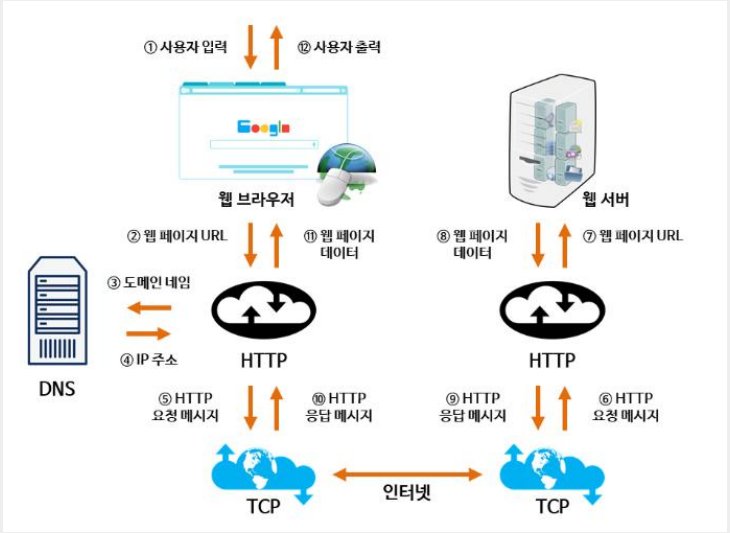
①② 사용자가 웹 브라우저를 통해 찾고 싶은 웹 페이지의 URL 주소를 입력함.
③ 사용자가 입력한 URL 주소 중에서 도메인 네임(domain name) 부분을 DNS 서버에서 검색함.
④ DNS 서버에서 해당 도메인 네임에 해당하는 IP 주소를 찾아 사용자가 입력한 URL 정보와 함께 전달함.

⑤⑥ 웹 페이지 URL 정보와 전달받은 IP 주소는 HTTP 프로토콜을 사용하여 HTTP 요청 메시지를 생성함.
이렇게 생성된 HTTP 요청 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 해당 IP 주소의 컴퓨터로 전송됨.

⑦ 이렇게 도착한 HTTP 요청 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 URL 정보로 변환됨.
⑧ 웹 서버는 도착한 웹 페이지 URL 정보에 해당하는 데이터를 검색함.

⑨⑩ 검색된 웹 페이지 데이터는 또 다시 HTTP 프로토콜을 사용하여 HTTP 응답 메시지를 생성함.
이렇게 생성된 HTTP 응답 메시지는 TCP 프로토콜을 사용하여 인터넷을 거쳐 원래 컴퓨터로 전송됨.

⑪ 도착한 HTTP 응답 메시지는 HTTP 프로토콜을 사용하여 웹 페이지 데이터로 변환됨.
⑫ 변환된 웹 페이지 데이터는 웹 브라우저에 의해 출력되어 사용자가 볼 수 있게 됨.



01 Server란? 1-3 웹 서버와 WAS의 차이점

웹 서버(Web Server)

- HTTP 프로토콜을 기반으로 하여 웹 브라우저의 요청을 서비스 하는 기능을 담당한다.
- 정적인 컨텐츠(.html / .png / .jpg / .css 등)를 제공할 때에는 WAS를 거치지 않고 바로 제공한다.
- 동적인 컨텐츠 요청이 들어왔을때에는 해당 요청을 WAS에 보내고 처리한 결과는 WAS가 전달하게 한다.
- 예를 들면 Apache Server, Nginx, IIS 등이 있다.

(정적인 데이터 처리하는 서버.)

웹 서버의 예시

WAS (Web Application Server)

- Web Server + Web Container = WAS
- DB 조회나 다양한 로직 처리를 요구하는 동적인 컨텐츠를 HTTP 통신을 통해 제공하는 기능을 담당한다.
- 웹 컨테이너, 혹은 서블릿 컨테이너 라고도 불리우며 JSP, Servlet 구동 환경을 제공하는 서버
- 분산 트랜잭션, 보안, 메시징, 스레드 처리 등의 기능을 처리하는 분산 환경에서 사용한다.

(동적인 데이터 처리하는 서버.)

WAS의 예시

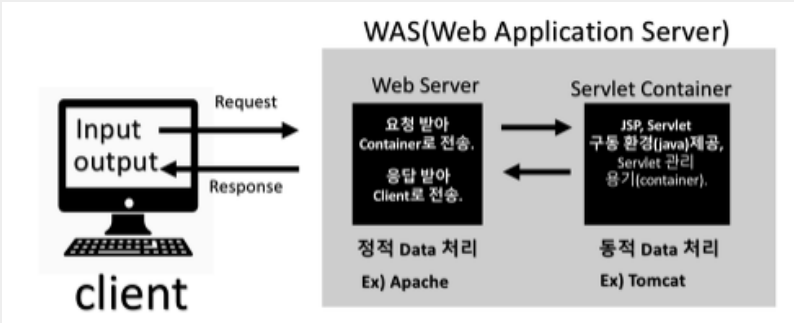
간단하게 말하자면, 웹 서버는 http 요청을 처리하는 것.
WAS는 클라이언트에게 요청을 받아 요청을 처리하고, 다시 클라이언트에게 응답해주는 역할.

01 Server란? 2 Tomcat Server의 구동 방식

톰캣 서버란?

톰캣은 아파치 소프트웨어 재단의 웹 어플리케이션 서버로서, 자바 서블릿을 실행키고 JSP코드가 포함되어 있는 웹 페이지를 만들어준다.
우리가 아는 아파치 톰캣 서버가 일반 웹 서버와 다르다. 라는 사실..
왜냐하면 Apache는 웹 서버를 칭하고, 톰캣은 전 슬라이드에서 나왔지만, WAS이기 때문이다.
아파치와 웹 서버와의 차이의 핵심은 이 컨테이너 기능(웹서버 + 서블릿)이 가능한가 / 불가능한가
서블릿 : 웹 서버 내부에서 동작하는 작은 자바 프로그램이다.

Apache Tomcat이라고 불리는 이유는 아파치의 기능의 일부를 가져와서 사용하기 때문에 그렇게 불리는 것이다.



아파치만 사용하면 정적인 웹페이지만 처리 가능하고,
톰캣만 사용하면 동적인 웹페이지 처리가 가능하지만 아파치에서 필요한 기능을 못가져옴. 또한 여러 사용자가 요청할시에 톰캣에 과부하가 걸림.
아파치와 톰캣을 같이 쓰면 아파치는 정적인 데이터만 처리하고, JSP 처리는 Web Container(톰캣의 일부)로 보내주어 분산처리 할 수 있다.
대부분 Apache의 포트는 80, Tomcat의 포트는 8080으로 처리함.

02 API란? 1-1 API의 기초

API(application programming interface)

주된 의미로는,
1) 컴퓨터나 소프트웨어를 서로 연결한다
2) 프로그램들이 서로 상호작용하는 것을 도와주는 매개체

API의 역할은?
1. API는 서버와 데이터베이스에 대한 출입구 역할
: 데이터베이스에는 정보들이 저장이 된다, 모든 사람들이 이 데이터베이스에 접근하는 것을 방지 하는 차원으로 API는 이를 방지하기 위해 서버와 데이터베이스에 대한 출입구 역할을 하며, 허용된 사람들에게만 접근성을 부여한다.

2. API는 애플리케이션과 기기가 원활하게 통신.
: API는 애플리케이션과 기기가 데이터를 원활히 주고받을 수 있도록 돕는 역할을 함.
애플리케이션 : 우리가 흔히 알고 있는 스마트폰 어플이나 프로그램을 말함.

3. API는 모든 접속을 표준화
API는 모든 접속을 표준화하기 때문에 기계/ 운영체제 등과 상관없이 누구나 동일한 액세스를 얻을 수 있다.
쉽게 말해, API는 범용 플러그처럼 작동한다고 볼 수 있다

02 API란? 1-2 API의 유형, 좋은 점

API(application programming interface)

API유형

1) private API

: private API는 내부 API로, 회사 개발자가 자체 제품과 서비스를 개선하기 위해 내부적으로 발행함. 따라서 제 3자에게 노출되지 않는다.

2) Public API

: public API는 개방형 API로, 모두에게 공개함. 누구나 제한 없이 API를 사용할 수 있는 게 특징.

3) Partner API

:partner API는 기업이 데이터 공유에 동의하는 특정인들만 사용 가능하다. 주로 비즈니스 관계에서 사용된다.

API 사용하면 뭐가 좋을까?

Private API를 이용할 경우, 개발자들이 애플리케이션 코드를 작성하는 방법을 표준화 함으로써, 간소화되고 빠른 프로세스 처리를 가능하게 한다.
또한, 소프트 웨어를 통합하고자 할 때는 개발자들 간의 협업을 용이하게 만들어줄 수 있다.

Public API와 partner API 를 사용하면, 기업은 타사 데이터를 활용하여 브랜드 인지도를 높일 수 있다.
고객 데이터베이스를 확장하여 전환율까지 높일 수 있다.

02 API란? 2 API의 예시, 사용

API 어떻게 사용할까?

API를 무료로 이용할 수 있다고 해도 본인 인증할 수 있는 키가 필요하며, 몇 번까지 무료 초과 이용 시 유료인 API도 있습니다.

ex) **공공데이터포털** : https://www.data.go.kr/ **Kakao Developers** : https://developers.kakao.com/
Naver Developers : https://developers.naver.com/ **OpenWeatherMap** : http://openweathermap.org/

API 사용하기

런던의 현재 날씨를 보여주는 응용 프로그램을 만들기 위해 OpenWeatherMap 이라는 운용 체제의 API를 통해 정보를 제공 받게 됩니다.
var api = "//api.openweathermap.org/data/2.5/weather?q=지역&appid=인증키“

키를 가져와 json를 통하여 사용하게 되면,

```
$(document).ready(function() {
 "use strict";

 var weather, img;
 var weatherApi = "//openweathermap.org/data/2.5/weather?q=London,uk&appid=b6907d289e10d714a6e88b30761fae22";
 //api 값을 가져와서 세팅 해주게 된다.

 $.getJSON(weatherApi).done(function(data) {
  weather = data.weather["0"].main;
  img = "http://openweathermap.org/img/w/" + data.weather["0"].icon + ".png";
  $("#Text").html(weather + "<br><img src='" + img + "'>");
 }).fail(function() {
  console.log("날씨 정보를 불러오지 못했습니다!");
 });
```

이런식으로 사용되게 된다 라는 것을 보여드리고 싶었습니다.

03 GitHub 1-1 GitHub란?

깃허브(GitHub)는

버전관리와 협업을 위한 코드 호스팅 플랫폼이다. 언제나 어디서나 협업 프로젝트를 쉽게 진행 할 수 있도록 돕는 것이다.
내 컴퓨터에 있는 깃의 히스토리(기록)를 가져와서 깃허브 웹사이트(클라우드)에 올릴 수 있고 변경된 히스토리를 확인할 수 있습니다.

깃(Git)
깃(Git)은 분산형 버전 관리 시스템(Version Control System)의 한 종류입니다.
버전 관리는 파일들을 복사, 백업, 저장 등을 해서 관리하는 것을 의미합니다.
이러한 버전 관리는 크게 1) 클라이언트-서버 모델(CVS, SVN 등)과 2) 분산 모델, 두 가지로 나눠서 볼 수 있습니다. 깃(Git)은 분산 모델에 속합니다.

깃(Git)의 장점은 별도로 주고 받는 작업 없이 같은 파일을 여러 명이 동시에, 즉 병렬 개발이 가능합니다.
두 번째 장점으로 작업한 파일에 대한 변경된 정보를 실시간으로 저장해 줍니다. 마지막 장점으로 같은 파일에 대한 각각 다른 버전을 보관할 수 있다는 점입니다.
(저희가 사용하고 있는 교수님 - 조장 - 조원 간의 pr , merge, fork..)

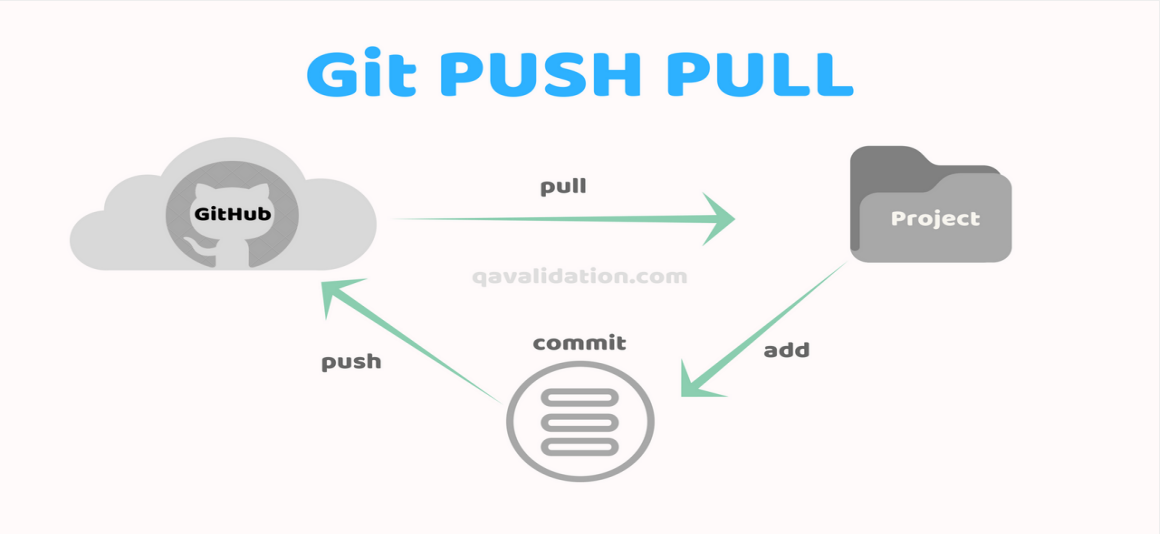
깃(Git)의 기본적인 용어들
Repository : 저장소를 의미합니다. 저장소는 히스토리, 태그, 소스의 Branch에 따라 버전을 저장합니다. 저장소를 통해서 작업자가 변경한 모든 히스토리를 확인할 수 있습니다.
Commit : 현재 변경된 작업 상태를 점검을 마친 뒤 확정하여 저장소에 저장하는 작업을 의미합니다.
Push : 현재 폴더를 그대로 업로드 하는 것이 아니라, 지금까지의 이력/버전(commit)을 push 하는 것입니다. commit한 이력이 repository로 저장됩니다.
Pull : 원격저장소 변경사항(이력)을 그대로 받아옵니다.
Head : 현재 작업중인 Branch를 의미합니다.
Branch : 분기점을 의미합니다. 복사하여 Branch에서 작업을 한 후 완전할 경우 Merge를 합니다.
Merge : Branch의 내용을 현재 Branch로 가져와 합치는 작업을 의미합니다.

03 GitHub 1-2GitHub가 중요한 이유

깃허브의 장점으로는,

팀 프로젝트에서 팀원들과 파일을 공유하는 장소와 공유하는 시간에 구애 받지 않고, 다같이 각자 작업하여,
나온 결과물들을 pull, push, pull request, merge를 통하여 작업물을 합치는 과정을 거칩니다.

그러므로 코드를 일일히 수정할 필요도 없고, 팀원들 간의 코드 공유도 필요없이 가져오고 내보내는 방식으로
작업을 하기 때문에, 편리합니다.

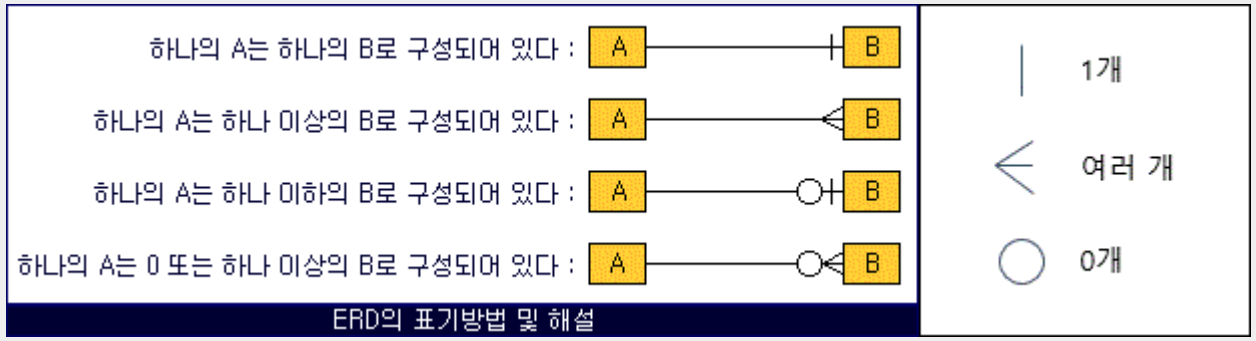


git push, pull의 사진

04 ERD ERD란?

ERD(Entity Relationship Diagram)

시스템의 엔티티들이 무엇이 있는지 어떤 관계가 있는지를 나타내는 다이어그램입니다.
ERD의 규칙으로는 아래 그림과 같습니다. (가져온 그림입니다.)



ERD의 표기방법 및 해설

A는 부모, B는 자식의 관계를 가진 ERD입니다.
~B를 구성하고 있다라는 것은, **' ~B를 포함하고 있다 '** 라고 이해하면 될 것 같습니다.
그리고 부모와 자식의 관계를 알아야 하는데 **A테이블의 기본키를 B테이블이 가지고 있다면**
A테이블이 부모 테이블, B테이블이 자식 테이블을 뜻한다. 라고 생각하면 될 거 같습니다.

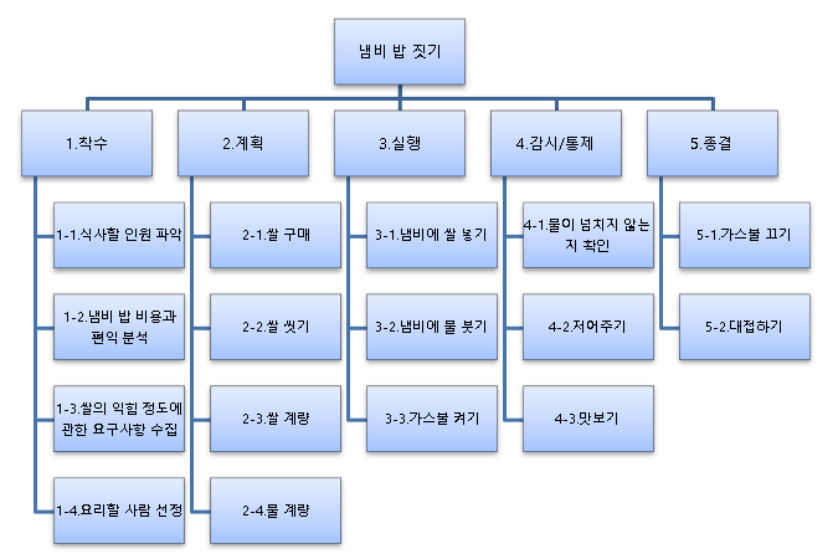
Ex) 예를 들어서 학생ID에 이름 전화번호 이메일을 저장하는 학생 테이블을 만들고,
학생 ID와 과목 ID에 학점을 저장하는 수강내역 테이블을 만들게 된다면,
학생의 ID를 가져와서 저장하는 수강내역 테이블은
하나의 학생의 0~N개의 수강테이블로 만들어지니,
부모 테이블은 학생 테이블, 자식 테이블은 수강 테이블로 분류 되어질 수 있다.

04 작업 분할 구조도

작업 분할 구조도란?

작업 분할 구조도 / WBS(Work Breakdown Structure)

프로젝트 목표를 달성하기 위해 필요한 활동과 업무를 세분화하는 작업이다. 프로젝트 구성 요소들을 계층 구조로 분류하여
프로젝트의 전체 범위를 정의하고, 프로젝트 작업을 관리하기 쉽도록 작게 세분화 하는 것이다.
쉽게 말해서 아래와 같은 사진으로 정리 할 수 있다.

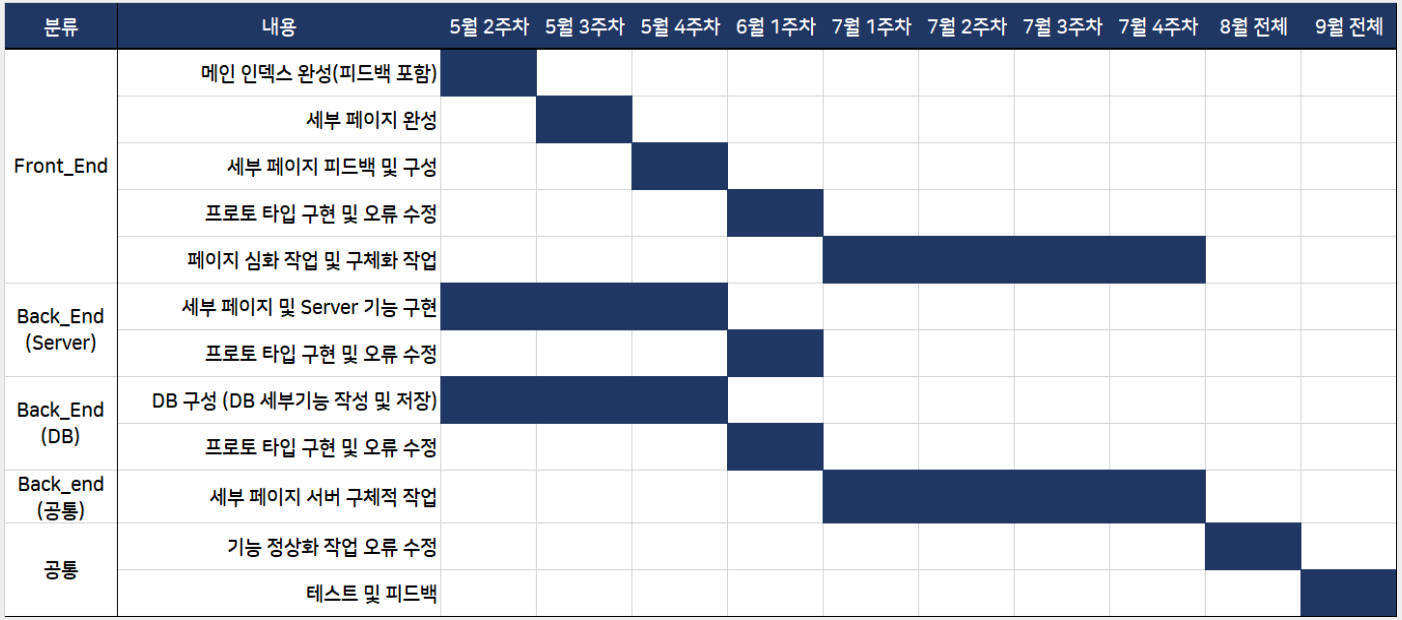


목적은 냄비 밥 짓기로부터 나뉘어지고,
WBS는 단순한 활동의 나열이 아니라 프로젝트 완료시까지 필요한 활동이 가시화되어 있기 때문에
범위에 해당하는 것(ex.물 계량)과 해당하지 않는 것(ex.식기 구매) 등을 쉽게 구분할 수 있어서
명확하게 어느 누구가 어떤 것을 해야 할지 나와있다.

05 Gantt chart Gantt chart란?

간트 차트(Gantt chart)

프로젝트 일정관리를 위한 바(bar)형태의 도구로서, 각 업무별로 일정의 시작과 끝을 그래픽으로 표시하여 전체 일정을 한눈에 볼 수 있다.
예시로 저희 팀에서 만든 간트 차트 사진을 가져 왔습니다.



한 번에 여러가지 프로젝트를 진행할 경우, 프로젝트 별로 시간 리소스 분배를 제대로 하지 않는다면 마감일에 다 끝내지 못할 수 있습니다.
이런 식으로 막대형 그래프로 그려서 프로젝트 기간과 리소스를 예측하는 데 효과적입니다.

etc. 최종정리 정리하면서 느낀점?

프로젝트를 하면서, 하나의 프로젝트를 끝낸다는 게 어렵다는 것을 느꼈고,
복잡하면서도 사람들 간의 협의와, 협력과 서로 간의 피드백을 해줌으로써
단점들을 고쳐 나가며 하는 게 프로그램 또는 웹을 구성하는 모든 프로그래머 분들에게 존경심을 느꼈다.

지금 이렇게 공부한 것도 기본적인 프로그램 기초 문법들이지만,
회사로 나가고, 사회에 뛰어들게 된다면 그때부터는 지금보다 시간이 없고 계속 공부를 하면서
해야 한다는 것을 느꼈다.

앞으로도 더욱 더 공부를 하여, 이번에 나오는 프로토 타입을
2학기에는 수정하여 완벽한 홈페이지로 만들고 싶다는 생각이 들었다.

더 확실하게 공부하여 좋은 프로그래머가 되도록 노력하겠습니다.

E N D O F D O C U M E N T

THANK YOU

20181765 김건호