목차

주요 업무

1. 고객사 등록, 요금 및 사용량 관리
2. 도면 관리 (파일 및 2D 뷰어)

1. 고객사 등록, 요금 및 사용량 관리

계기

리뉴얼된 홈페이지에서의 신규 고객사의 대비와 클라우드(AWS에 빌드되어있는 자사 서버)고객이 늘어남에 따라 고객사의 등록 -> 요금 청구 -> 사용량 집계 과정의 자동화가 필요하게 되었습니다.

진행사항

아래 결과 항목의 프로세스 순서대로 개발을 진행하였습니다. 마케팅을 위해 홈페이지에서 고객의 행동들을 디비에 기록하고, 무료체험 신청 정보를 담당자에게 전달하여 검토할 수 있도록 구현하였습니다. 검토 결과를 토대로 고객사의 DataBase를 생성합니다. 무료체험 이후 유료로 전환하지 않는 경우 일정기간 데이터를 보관한 뒤 파기합니다. 유료전환 이후 고객사의 사용량에 따라 요금을 계산서를 작성합니다.

계산서를 토대로 자동 결제 및 세금계산서를 발행 예정이나 테스트만 진행하고 PG사 심사 대기로 인해 개발이 완료되지 않았습니다.

결과

해당 업무 이후 아래 프로세스대로 고객사를 관리하고 있습니다.

1. 고객의 무료체험 신청
2. 검토 후 고객사 데이터베이스 생성
3. 라이선스 및 사용량 집계
4. 자동 결제 및 세금계산서 발행

2. 도면 관리 (파일 및 2D 뷰어)

계기

고객들의 도면을 웹에서 바로 볼 수 있도록 뷰어기능을 제공하고 있습니다. 기본적으로 도면은 dwg, pdf 파일로 되어 있으며 크기 자체가 크기 때문에 비율이나 dpi설정을 적절하게 계산해서 이미지로 변환 시켜두어야 하는데, 고객들이 도면을 그릴 때 표준을 잘 따르지 않아 기존의 방식으로 생성된 이미지로는 도면을 상세하게 볼 수 없는 문제가 있었습니다. 다양한 형태의 도면을 최소한의 설정으로 보거나, 사용자가 원하는 방법으로 보기 위한 개발이 필요하였습니다.

진행사항

도면을 png 파일로 만들어 뷰어를 통해 화면으로 보여줍니다.

- dwg 파일의 경우 dwg -> pdf -> png
- pdf 파일의 경우 pdf -> png dwg 파일의 경우 AutoCad의 AutoLisp을 활용해 도면 블럭에 결재자 정보등을 자동으로 입력해 주고, 도면에 설정된 사이즈를 토대로 pdf로 변환 시켜줍니다. 도면에 설정이 없다면 저희 솔루션에 사용자가 입력한 사이즈로 변환시키고, 아무것도 없다면 기본값으로 변환 시킵니다. 이 pdf의 크기를 구해 알맞은 비율과 dpi로 png파일을 생성하여 뷰어를 통해 사용자에게 보여줍니다. 이 png 파일은 저희 솔루션의 뷰어를 통해 열 수 있고 회전, 색반전 및 인쇄등을 지원합니다. 또한 png파일이 아닌 크롬 웹에서 지원하는 pdf 웹 뷰어 방식으로도 볼 수 있도록 다양한 설정을 추가하였습니다.

결과

일반적인 도면 작성법을 따르지 않거나 png로 변환된 자사의 뷰어가 아닌 pdf 웹 뷰어 방식으로 여는 등 다양한 고객의 니즈를 더 수용하게 되어 신규 고객들에게 더 많은 어필을 할 수 있게 되었습니다.

기술 스택 도입

1. Redis
2. Jenkins

1. Redis

계기

이중화되어 있는 서버의 각 메모리에 저장되어 있는 세션을 검사하는 과정에서 서버 1에서 생성되고 관리되는 세션이 서버 2로 요청이 가는 경우 세션 정보를 가져오지 못하는 문제점이 발생하였습니다. 해당 서버에 세션이 없는 경우 다른 서버에 세션 검사를 요청하는 로직이 있었으나, 완벽히 동작하지 않는 상태였습니다. 그래서 저는 세션 정보를 DataBase에서 공유하는 방식을 제안하였습니다.

* 로그인 이력 등의 관리를 위해 DataBase에 데이터를 저장하고 있으나 유효성 검사에는 활용되지 않는 상태

진행사항

기존 메모리에서 관리하던 세션 정보를 DataBase에 관리하는 경우 속도 저하가 발생하여 In-memory DataBase인 Redis로 테스트를 진행하였습니다. 로컬에 서버를 올리고 WSL를 활용하여 Redis 설치 및 구현을 진행 하였고, 데이터 베이스, 메모리, Redis 의 속도를 1차적으로 테스트 하였습니다. 테스트 결과가 만족스러워 운영상황을 대비해 Redis Sentinel 구조로 다시 한번 테스트를 진행하여 DataBase 대비 데이터 접근에 300% 만큼의 속도 향상을 도출 하였습니다.

결과

운영환경에 도입을 진행하지는 못했습니다. 로드밸런싱 과정에서 서버가 생성되더라도 2개정도만 운영되는 점, 자사의 AWS에 빌드되어있는 서버의 트래픽은 낮다는 점과 운영 환경에서의 개발 및 비용 문제로 저희 대표님과 상의 후 향후 고객사가 늘어나거나 session 접근 속도 등의 문제가 생기면 재 진행사항하기로 하고 DataBase에서 검사하는 것으로 마무리하였습니다.

2. Jenkins

계기

현 직장에서 구현 -> 테스트 -> 빌드 자동화 처리가 없던 상태였습니다. 때문에 개발이 끝난 후 각 개발자가 테스트 코드를 돌리고 커밋을 진행하였는데, 전체 테스트 코드를 매번 돌리기엔 양이 많아 어느 정도 범위만을 테스트고 커밋하여 새로운 버그 및 기존 버그가 재발하는 등 **고객들에게 솔루션의 신뢰를 떨어트리는 행동이 반복**되었습니다.

빌드는 협업중인 다른 회사와 저희 대표님이 진행하기도 하고 빌드 방식이 획일화 되어있지 않아 구현 -> 테스트 까지의 자동화 고민을 하였습니다. 평소 반복업무를 싫어하여 자동화에 관심이 많았던 저는 구현단계에서는 설계된 DB스키마를 Mybatis, Repository interface, Domain 객체 등으로 변환 시켜주는 앱(아래 링크 참조)을 만들고 젠킨스를 활용하여 테스트하는 방법을 제안하였습니다.

* 참조 Assistant GitHub 링크

진행사항

젠킨스 구현자체는 테스트와 결과 메일 발송으로 간단하였지만, 아래와 같이 개발환경이 획일화 되어있지 않아 하나씩 수정해가며 테스트를 진행하였습니다.

- 설정 파일들이 Resource 위치에 있지 않고 각 분산되어 있는 점
- 메이븐 프로젝트이지만 메이븐을 활용하지 않는 점
    - 일부 라이브러리
    - 프로젝트 설정
    - 빌드 등

결과

젠킨스로 테스트를 자동화한 이후 고객이 수정요청했던 버그가 재발하는 최악의 경우는 발생하지 않았으며 운영서버에도 이전보다 검증된 상태로 반영되고 있습니다. 추후 과제로는 메일 발송 방식이 아닌 저희 솔루션의 업무와 연동하여 할당하는 것과 빌드까지의 자동화가 남아있습니다.