

문성하

01088977064 인증 완료

azqazq195@gmail.com 인증 완료

안녕하세요, 문제점을 파악하고 변화할 수 있는 능력을 지닌 문성하입니다.

학창 시절부터 커뮤니케이션 능력을 개선하기 위해 노력해 왔습니다. 심리학 서적을 읽고 군 복무, 대학교에서의 아르바이트와 동아리 활동을 통해 문제점을 파악하고 변화할 수 있었습니다. 이러한 노력 덕분에 현재는 '어디서든 적응 잘하겠다'는 말을 듣는 정도로 커뮤니케이션 능력이 크게 향상되었습니다.

서버 구조를 재 설계하고 최적화하는 작업을 성공적으로 수행한 경험이 있습니다. 이 과정에서 코드의 품질과 효율성을 높이는데 주력했으며, 여러 처리와 디버깅을 개선하여 프로젝트의 안정성을 높였습니다. 또한, 테스트 자동화를 위한 설계와 필요한 로깅 방식 변경을 통해 유지보수가 용이한 시스템을 구축하였습니다.

향상된 커뮤니케이션 능력을 활용하여 현업에서 발생한 문제점들을 기록하고 개선 방향과 함께 동료들과 공유하며 함께 문제를 해결해 나갔습니다. 이를 통해 기술 스택에 대한 이해를 높이고 적용 가능한 최적의 해결책을 도출할 수 있었습니다.

문제점을 인지하고 더 좋은 방향을 찾아가며 뛰어난 개발자가 되기 위해 노력하고 있습니다. 이러한 능력을 바탕으로 앞으로의 프로젝트에서도 효율적인 문제 해결과 팀워크를 성공적으로 이루어 낼 것입니다.

Github 저장소

<https://github.com/azqazq195>

주요 기술 2개

Kotlin Spring Boot

기술 태그 18개

JWT Android Docker Firebase Google Cloud Platform Java MariaDB MongoDB MySQL Oracle Spring TypeScript flutter AWS ECS Spring Data JPA MyBatis NestJS GitHub Actions

업무 경험

2022-10 ~ 2023-01
(4개월)

주식회사오늘의꽃

백엔드 개발자

NestJS TypeScript Docker MySQL MongoDB Spring Boot Kotlin AWS ECS GitHub Actions

- 서버 재설계 진행 중 회사의 파산으로 인해 프로젝트를 마무리하지 못 했습니다.

- Rest Api 서버 재설계

Kotlin + Spring 활용한 Rest Api 구현

Github Actions을 활용한 테스트 및 배포 자동화

Container 및 ECS 배포

기술 스택

- Language: Kotlin

- Framework: Spring Boot
- Data: JPA / MySQL / Redis / H2
- Test: KoTest
- CI/CD: Git / GitHub Actions
- Infra: Docker / S3 / ECS
- **Rest Api 서버 리팩토링**
기존 서버가 가지는 문제점 파악 및 수정
 1. 불필요한 Dependency 제거
 2. Error Filter 적용 및 Error Response 관리
 3. 디버그 및 사용자 오류 추적 Logging 설계
 4. Test 자동화
 5. User-Role 관리 추가
 - AS-IS: 역할 개념이 아닌 유저 타입(그룹) 개념
 - TO-BE: Role 추가 및 Authentication 추가
 6. DTO validation 추가
 - AS-IS: 입력검사를 비즈니스 로직에서 각각 수행
 - TO-BE: DTO decorator를 통해 수행하도록 책임 분리 및 알맞은 Exception 제공

2021-07 ~ 2022-09
(1년 3개월)

코너스톤테크놀로지(주)

백엔드 개발자

<https://www.csttec.com>

Java Spring MyBatis MySQL

도면관리 PLM DMS PDM 클라우드 솔루션

- **결제 및 세금계산서 연동**
결제 및 세금계산서 발행 연동
결제: 아임포트, 세금계산서: 팜빌
PG사 승인 심사 대기
- **테스트 자동화 구현**
테스트 코드는 작성하나 이를 매번 돌리지 않아 같은 버그 및 새로운 버그가 계속 발생함.
이를 해결하고자 테스트 자동화 처리
 - Jenkins 활용 JUnit 테스트 자동화 및 알림 처리
- **주요 업무**
도면 관리 (파일 및 2D 뷰어)
BOM 관리 (트리 구조 생산 아이템 관리)
프로젝트 관리 (프로젝트 계획 및 실행, 간트 차트)
고객사 요금 및 트래픽 관리

사이드 프로젝트

2022

Assistant

flutter Spring Boot JWT Java Spring Data JPA MariaDB

현직 업무 도움 데스크탑 앱
기능

- 회원가입 및 로그인
- 현직에서 사용중인 sql script 분석
- 분석한 코드를 바탕으로 Domain 객체 Interface, Mybatis 쿼리 자동 생성

기술

- JWT 활용 토큰 방식 로그인.
- Flutter 로 Windows app (msix) 빌드
- Retrofit 활용 RestApi 통신
- Provider 사용으로 비동기식 화면 처리
- Spring Boot, mariadb, jpa를 활용한 서버 구성
 - 팀 구성 — 개인
 - <https://github.com/azqazq195/assistant>

2020

Foret

Java Android Spring Google Cloud Platform Firebase Oracle

스터디 모임 어플
로그인 및 회원가입, 비밀번호 찾기
가입한 그룹 출력, 페이징 처리, 회원 역할에 따라 데이터 노출 및 상태 변경
실시간 채팅, 푸쉬 알림
유저의 로그인 상태 확인, 메세지 읽음 상태, 메세지 삭제 기능
내가 쓴글, 가입한 그룹, 답글, 채팅 등 나의 정보 업데이트 시 푸쉬 알림
게시판, 댓글, 대댓글, 선택 정렬, 좋아요 기능
사진 업로드 및 수정 삭제
검색시 자동완성 기능

- 팀 구성 — 4명
- https://github.com/azqazq195/Project_Foret

활동

2023-08 ~

원티드 프리온보딩 백엔드

https://www.wanted.co.kr/events/pre_ob_open

- 자료구조
- 팀 프로젝트

2020-06 ~ 2020-12

국비교육국비교육

<https://it.ezenac.co.kr/>
과정명

- JAVA 프로그래밍 기초
- 데이터베이스 설계 및 웹 앱/ 연동
- Spring Framework HTML & CSS & 및 JavaScript 를 통한 웹페이지 구축
- JAVA 기반 안드로이드 앱 개발
- JSON 파일을 이용한 데이터 연동

첨부 파일

Foret APP PPT.pdf

목차

코너스톤테크놀러지 (2021-07 ~ 2022-09)

1. 주요 업무

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

2. 기술 스택 도입

1. Jenkins

오늘의 꽃 (2022-09 ~ 2023-01)

1. 주요 업무

1. Rest API 서버 구조로 재설계

코너스톤테크놀러지(주)

2021-07 ~ 2022-09

주요 업무

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

1. 고객사 등록, 요금 및 사용량 관리

문제

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

진행내용

1. 고객의 무료체험 신청

고객의 무료체험을 시작으로 sales 담당자에게 메일로 알람을 발송합니다. 담당자가 고객이 제출한 자료를 검토 후 승인 시 데이터 베이스를 생성하여 자사 솔루션을 사용 가능하도록 구현하였습니다.

2. 라이선스 및 사용량 집계

자사 솔루션은 사용자가 자유롭게 라이선스를 변경할 수 있습니다. 이에 따라 스케줄러를 통해 라이선스 사용량을 매시간 기록합니다. 또한 업로드된 파일의 용량이나 도면 파일을 변환하는 과정에서 생기는 트래픽 등을 집계하여 비용명세서를 작성합니다.

3. 세금계산서 발행

집계된 비용명세서를 토대로 세금계산서를 발행하여 고객사에게 전달합니다. 세금계산서는 팝빌(PopBill) 솔루션을 사용하였습니다. 매월 10일 전달 요금을 발행하도록 구현하였으며 계산서 발행 요청, 결과 등을 저장하여 발행 내역을 관리하고 필요시 재발행할 수 있도록 구현하였습니다.

4. 자동결제 (예정)

위의 순서대로 개발 및 검증이 완료되었다고 판단하여 월별 자동결제를 위한 PG사 심사를 준비하고 있습니다.

2. 도면 관리 (파일 및 2D 뷰어)

문제

고객들의 도면을 별도의 프로그램 없이 웹에서 바로 볼 수 있도록 뷰어기능을 제공하고 있습니다. 다음과 같은 문제가 발생하여 업무를 진행하게 되었습니다.

- 도면 파일(dwg)을 작성할 때의 표준방식을 따르지 않아 도면의 크기나 정보등을 정확히 읽을 수 없는 문제.
- 자사 파일 서버(cloud와 on-premise를 대응하기 위함)와 변환 서버(도면 파일을 변환하기 위한 서버)의 난개발

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 형태에 맞는 최소한의 설정을 해두고 고객이 원하는 설정대로 보는 방법을 추가하고, 표준을 따랐다면 자사 솔루션의 업무와 연동하여 도면파일에도 해당 업무에 관련된 정보를 기록할 수 있도록 확장이 필요하였습니다.

파일 서버와 변환 서버는 예전에 개발 후 방치되어 있는 상태였습니다. 추후 유지보수를 위한 코드 정리 및 서버간의 요청방식 개선이 필요하였습니다.

진행내용

도면을 png 파일로 만들어 이미지 뷰어를 통해 화면으로 보여줍니다.

- dwg 파일의 경우 dwg -> pdf -> png
- pdf 파일의 경우 pdf -> png

도면 파일을 변환하기 위한 정보를 찾거나 입력받아 위의 순서대로 변환하였습니다. 업무에 첨부파일로 연결된 도면 파일은 담당자, 결재자 등의 정보를 도면 파일에 기록할 수 있도록 확장하였습니다.

- AutoCad의 AutoLisp를 활용하여 도면 사이즈를 읽거나 지정하여 정보를 저장합니다.
- 업무의 정보를 도면에 기록합니다.

추가적으로 도면을 작성하고 배포처에 배포할 때 원본파일이 아닌 변환된 pdf 파일을 받을 수 있도록 수정하였습니다.

결과

일반적인 도면 작성법을 따르지 않거나 png로 변환된 자사의 뷰어가 아닌 pdf 웹 뷰어 방식으로 여는 등 고객의 다양한 행동에 대해 대처할 수 있게 되었습니다.

기술 스택 도입 및 검토

1. Jenkins

1. Jenkins

계기

현 직장에서는 **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**을 만들고 젠킨스를 활용하여 테스트하는 방법을 제안하였습니다.

진행사항

젠킨스 구현 자체는 테스트와 결과 메일 발송으로 간단하였지만, 아래와 같이 개발환경이 획일화 되어있지 않아 하나씩 수정해 가며 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 이전 보다 검증된 상태로 반영되고 있습니다. 추후 과제로는 메일 발송 방식이 아닌 저희 솔루션의 업무와 연동하여 할당하는 것과 빌드 자동화가 남아있습니다.

오늘의 꽃

2021-07 ~ 2022-09

주요 업무

1. 서버 구조 재설계 (Refactor)
2. 서버 구조 재설계 (Re-new)

1. 서버 구조 재설계 (Refactor)

문제

입사 당시 기존 모놀리식 서버 구조를 Front-End, Back-End로 분리하며 Rest API로 재설계 작업을 하는 도중이 있었습니다. 재설계 및 Legacy를 제거하는 작업이 있으나 여전히 FrameWork의 LifeCycle을 따르지 않거나 무분별한 Dependency로 인해 재설계 이후에도 문제가 발생할 것을 인지하였으나 급한 작업부터 진행하기로 하였습니다.

진행내용

'오늘의 꽃'에서는 Typescript와 NestJs Framework를 사용하고 있습니다. NestJs Framework를 사용하지만 제공하는 가이드에 다소 벗어난 코드들이 존재하였습니다. 따라서 디버깅이 어려웠고 문제가 발생하면 Dependency를 사용하여 회피해 가는 코드가 많이 있었습니다. 유지보수를 위해 LifeCycle을 파악하고 공식문서를 토대로 수정하였습니다.

1. Error Filter 적용 (RestAdvice in Spring)

- Exception에 대한 처리가 되지 않아 개발자가 직접 명시해둔 몇몇 Exception을 제외한 것들은 메세지나 Reponse 형식이 통일되지 않았습니다.
- 이때문에 Front-end 에서도 에러발생시 이유를 정확히 알지 못하고 자체적으로 에러를 처리하고 있었습니다.
- 에러 양식을 통일 시켜주고 정확한 메세지를 전달하도록 Filter와 Exception 메세지 관리를 설계하였습니다.

2. class-validation, class-transform 활용 (Validation in Spring)

- DTO 입력검사를 Service 로직에서 수행하고 있습니다.
- 이를 DTO에서 Decorator(Annotation in Spring)으로 진행하도록 책임 분리를 하였습니다.

3. 디버깅 용 Logging 방식 변경

- 기존 logging은 각 개발자가 규칙없이 작성한 log, request 요청 정도의 의미 없는 logging이 진행되고 있었습니다.
- 이를 디버깅 가능한 필요한 logging이 되도록 재설계하였습니다.
 - error-filter를 통한 오류 추적.
 - request 수행 시간 및 성공 여부, 실패 이유.
- datadog 및 s3로 관리.

4. Typeorm 활용 (JPA in Spring)

- node 진영의 JPA인 Typeorm 을 적극 활용할 수 있도록 재설계 하였습니다.
- JPA를 지향하는 만큼, Entity Base로 Relation 관리, Query 할 수 있도록 구성.
- Typeorm은 이슈가 많아 JPA 처럼 관리가 되진 않을 것으로 예상됨.

5. Test 자동화를 위한 설계

- Test code가 30% 밖에 되지 않는 점, Local 테스트 환경이 없어 선임개발자가 빌드 전에 통합테스트를 돌리는 점이 문제가 되어 보여 Test code 작성 방식을 통일화 하고 테스트 환경을 설계하였습니다..
- Local 테스트를 위한 DDL, DML
- Local - Preview - Production 빌드 흐름에 맞는 sql script 작성 방법 강구
- 테스트 코드 작성시 중복 코드를 줄일 수 있는 방법 강구

6. 요청자의 정보를 가져올 수 있도록 설계 (SecurityContextHolder.getContext().getAuthentication() in Spring)

- 기존 코드에서 session에 로그인된 유저 정보가 있음에도 해당 유저 정보를 활용하지 않았습니다.
- 이때문에 Front에서 user/me 호출뒤 post 요청시 userId를 직접넣어주어 호출하였습니다.
- NestJs에서는 Spring 처럼 전역적으로 요청자의 정보를 불러올 수 있는 메서드를 지원하지 않아 직접 작성하였습니다.
- Request 시 마다 SecurityContext instance 를 생성하여 유저 정보를 가져옵니다.
- 이로서 User-Role, RequestedUser 등을 활용할 수 있게 되었습니다.

결과

- 디버깅 향상
- 불필요한 Request 정리
- 각 코드들의 책임 분리
- 사용하고 있는 기술 스택 숙지
- 테스트 및 빌드 자동화

- Front, Back 각자 할일만 할 수 있도록 정리

2. 서버 구조 재설계 (Re-new)

문제

1번 항목의 재설계 이후에도 코드 컨벤션의 부재로 인한 러닝커브, 디버깅 문제 등으로 재설계를 진행하게 되었습니다. 이때 NestJs의 전체적인 Refactoring과 새로운 기술스택으로 새로운 서버를 만드는 것 중 새로운 기술스택인 Kotlin + Spring으로 진행하게 되었습니다.

진행내용

[Notion](#)

결과

완료하지 못한채 퇴사.