**세미나 대본**

안녕하십니까 씨스존 연구소 플랫폼팀 신입 연구원 박정훈 입니다.

제가 이번 세미나에서 주제로 선정한 것은 앞서 연구를 진행하고 계신 저희 플랫폼팀 분들의 업무를 한 달 동안 제가 이해한 부분에 대해 정리를 해보았고, 마지막으로 간단하게 학원에서 배운 것과 회사의 프로젝트 구조가 달랐는지에 대해 말씀드리려 합니다.

목차는 다음과 같이 5가지로 나누었고

1,2,3,4,5 (읽기) 이렇게 세미나가 진행되겠습니다.

**<진행 업무>**

현재 저희 연구소 플랫폼팀이 진행중인 업무 : 국토교통부 13가지 과제 중 10과제 – 콜드체인 상태정보 관리 및 실시간 모니터링체계 구축 기술 개발입니다.

진행중인 개발로는 (세가지 읽기)있고 이부분에 대해 자세히 살펴보면 (사진)

이 사진은 융복합 물류산업단 착수회의 자료에서 물류 정보 플랫폼 구축 기술의 개발 기관 및 역할에 대해 나와있는 사진입니다.

과제개요 (타이틀만 읽기),

연구 목표 블록체인 기술을 활용하여 정온물류 모니터링 플랫폼 구축,

연구 개발기관 및 역할 (주관기관 인하대학교) 공동 개발 회사들 중에서 저희 씨스존은 TCLS 서비스 플랫폼 연구를 진행하고 있습니다

* **TCLS (Temperature Controlled Logistics System**) 온도 제어 물류 시스템

제품이나 물품의 운송 및 보관 과정에서 온도를 효과적으로 제어하는 시스템

이를 통해 온도를 지속적으로 모니터링하고 제어하여 제품의 품질을 유지하는 시스템

* **SaaS** (Software as a Service) **서비스형 소프트웨어**

**소프트웨어를 기기에 설치하지 않고 온라인으로 액세스할 수 있는 소프트웨어 라이센스 및 제공 수단을 의미**

콜드체인 TCLS 서비스 플랫폼으로 Valuekeeper Service 가 현재 연구 개발 진행중이며, 이 서비스 에서 담당하는 부분은 의학 / 물류 / 육류 / 식품 이 있습니다. 아래의 항목을 보면 앞서 보았던 관제센터, 통합물류관리 등에 관한 서비스로 toc, fc fd 이렇게 현재 개발중이고 관제센터인 Toc, 식품 물류센터 서비스인 Fc, 식품 배송 서비스인 Fd 등 이렇게 진행중에 있습니다.

(사진)

**<기술스택>**

앞에 개발중인 서비스 환경들을 구축하기위해 어떤 개발도구들을 사용하여 만들었는지 설명 드리기 위해 기술 스택을 정리해 보았습니다.

**백엔드**

**Maven 빌드도구 - 의존성 관리**

**NETTY** 프레임워크 / 서버 개발 (HTTP)

네티는 비동기 이벤트 기반 네트워크 어플리케이션 **프레임워크**로써 유지보수를 고려한 고성능 프로토콜 서버와 클라이언트를 빠르게 개발

**Hibernate ORM 프레임워크**

ORM : 어플리케이션과 데이터베이스 연결 시 SQL언어가 아닌 어플리케이션 개발언어로 데이터베이스를 접근할 수 있게 해주는 툴

프론트엔드

Redux : 리액트 어플리케이션에서 상태를 효과적으로 관리하기 위한 라이브러리

**Actions:** 상태 변경을 나타내는 객체. 애플리케이션에서 특정 동작이 발생할 때 발생

**Store:** 전역 상태를 저장하는 객체로, 애플리케이션의 상태와 불변성을 관리

Context api : 컴포넌트 트리 전체에서 데이터를 전달할 수 있도록 하는 리액트 api.

중첩된 컴포넌트 간에 데이터를 전달하기위해 props 를 사용하지 않아도 된다.

(컴포넌트 트리 -> 중첩된 컴포넌트의 구조를 칭함)

**<프로젝트 구조>**

이제 이러한 개발 도구들을 이용하여 프로젝트를 제작할 때 만든 구조에 대해 설명 드리겠습니다

우선 백앤드 프로젝트구조르 보게되면 vkParent 상위 프로젝트에서 필요한 라이브러리들과 버전을 모두 정의해주고 하위프로젝트를 모듈로 설정하여 전체 버전관리 및 라이브러리 빌드를 한다.

?) vkDb / vkRest 에서 <relativePath> 를 사용하여 부모프로젝트와 자식 모듈 간의 의존성 계를 정의.

?) vkDb / vkRest 에서 <relativePath> 를 사용하여 의존성 주입 ( vkParent의 기능이나 모듈을 사용하기위해)

장점 - 중복 코드를 최소화 하여 유지보수를 높힘 , 프로젝트의 버전관리가 용이해짐

**프론트 (폴더구조)**

Assets : 이미지나 폰트 같이 프로젝트에서 사용되는 자산 폴더

Components : 컴포넌트들이 위치하는 폴더. 화면 UI의 일부를 나타내며 재사용 가능한 빌딩 블록

Hooks: hook 정의하는 폴더. 컴포넌트 로직 재사용을 위한 함수들이 있다

Layout : 페이지에서 공통적으로 사용되는 레이아웃 구성 폴더 (헤더,푸터 같은 공통 레이아웃)

Menu-items : 사이드바, 네비게이션등 메뉴 아이템과 관련된 컴포넌트나 로직

Pages : 각 페이지에 대한 컴포넌트 정의.

Routes: 프로젝트 라우팅 설정과 관련된 파일이 위치. 각 페이지에 대한 경로와 연결된 컴포넌트 정의

Scss : 프로젝트에서 사용되는 SCSS 파일들이 위치. 스타일을 관리하고 구조화하기위한 목적

Store : 상태관리를 위한 폴더 (리덕스 액션, 리듀서 등)

Themems : 프로젝트의 테마와 스타일 관련 파일

Utils: 여러 컴포넌트 또는 모듈에서 사용할 수 있는 유틸리티 함수들의 폴더. 공통적인 기능을 모듈화하고 재사용하기 위한 목적

이렇게 구현된 프로젝트를 배포하는 구조에 대해 설명드리려 합니다.

**<배포구조>**

1. 개발 소스를 git 서버에 푸시한다
2. 젠킨스가 코드 푸시를 감지하고 자동으로 필드 파이프라인을 실행
3. 젠킨스가 프로젝트를 가져와서 메이븐 빌드 도구를 사용하여 소스 코드를 컴파일하고 패키징함
4. 빌드된 어플리케이션은 도커 이미지로 래핑된다. (이미지는 일반적으로 도커파일을 사용하여 정의)
5. 도커 레지스트리에 푸시 – 빌드된 도커 이미지는 도커 레지스트리에 푸시된다
6. 쿠버네티스에서는 빌드한 도커 이미지를 젠킨스를 통해 배포 작업을하고 쿠버네티스에 어플리케이션을 배포한다.

배포가 된 후 동작하는 원리에 대해서 살펴보자면

**동작구조**

1. 클라이언트가 웹브라우저에서 요청을 보냄 (http 요청을 생성하고 해당 요청을 서버로 전송)
2. 쿠버네티스 ingress 컨트롤러가 요청을 확인하여 어떤 서비스로 요청을 라우팅 할지 결정
3. 선택된 서비스는 배포된 pod 중 하나로 클라이언트의 요청을 보냄
4. Pod 내의 어플리케이션이 클라이언트의 요청을 처리하고 응답을 생성
5. Pod 의 응답이 ingress 컨트롤러로 전달되고 클라이언트의 웹 브라우저로 반환

이제 마지막으로 학원에서 배운 것과 회사에서의 업무에 대해 간단한 비교를 하겠습니다

학원에서 사용한 기술 스택으로는 (표) 입니다

스프링 프레임워크를 사용하여 레거시 레거시 프로젝트를 만들었고,

**<학원 / 회사 비교>**

프로젝트 구조

**(학원)**

**(회사)**

**(학원)**

동작 구조

1. 요청이 들어오면 Dispatcher-Servlert이 http 프로토콜로 들어오는 모든 요청을 가장 먼저 받아 컨트롤러에 위임
2. 요청 들어온 매핑값을 HandlerMapping 을 이용해서 Controller 객체 매핑해서 매칭된 메소드 실행

vkRest 에서 컨트롤러 역할 해줌

**(회사)**