서버 모니터링 시스템

개발자: 최경선

1. 프로젝트 설명

Buhl Data Service GmbH의 Web Service팀은 Contact system의 개발을 담당하고 있습니다. Contact system은 고객의 정보를 관리하고 RESTful Web Services와 연동되어 여러 내부 시스템과 고객의 정보를 공유합니다. RESTful Web Services의 담당자는 Web Services와 연결된 내부 시스템의 상태관리 및 Web Services가 동작하는 서버의 상태를 파악하기 위해 모니터링 시스템을 개발하였습니다.
이 서버모니터링 시스템은 적은 사용자를 이유로 인해 PHP 기반으로 하나의 페이지에 구성되었고 직접적으로 Web Services를 호출함으로써 모니터링 시스템에 필요한 데이터를 제공합니다.
시간이 지남에 따라 관리 되어져야 할 고객의 정보가 늘어나고 이로 인해 Web Services 에서는 더 많은 내부 시스템과 연결이 되고 있고, 또한 내부 시스템 운영자 또한 서버모니터링 시스템 사용을 요구하고 있습니다. 이러한 이유로 모니터링 시스템은 개선이 필요한 상황입니다.

해당 프로젝트는 Buhl Data Service GmbH에 근무하며 배운것을 활용해보기 위해 시스템 분석, 설계에서부터 디자인, 개발, 등 프론트엔드, 백엔드, RESTful Web Service 구분하지 않고 진행하였습니다.

2. 구 모니터링 시스템 분석

구 모니터링 시스템의 기능들은 현재 구조적으로 표현되지 않고 하나의 웹페이지에 모든 기능을 표시 하고 있습니다. 시스템을 사용하기를 원하는 사용자에게 기능에 대한 별도의 설명이 필요하였고 복잡한 디자인 구성으로 인해 시스템이 비효율적으로 사용되고 있었습니다.
또한 모니터링 시스템에 보여지는 데이터는 실시간으로 사용자에게 제공 되어져야 하기 때문에 모니터링 시스템에서 여러 번 Web Services를 호출하였고, 이로 인해 Web Services의 서버는 트래픽과 같은 시스템의 부담이 증가하고 있습니다.

3. 프로젝트 목표

1) RESTful Web Services 서버 부담의 경감
모니터링 시스템으로 인한 RESTful Web Services 서버 부담을 줄이기 위해 데이터를 가져오기 위한 새로운 프로세스가 생성 되어져야 합니다.

2) 디자인 개선

서버모니터링 시스템을 통해 사용자는 서버 및 시스템의 문제를 쉽게 파악할 수 있어야 합니다.

3) 사용자 관리

로그인 프로세스를 추가하여 사용자는 관리 되어져야 합니다.

4) 서버 상태 히스토리 제공

RESTful Web Services 담당자는 서버의 이전 상태를 확인할 수 있어야 합니다.

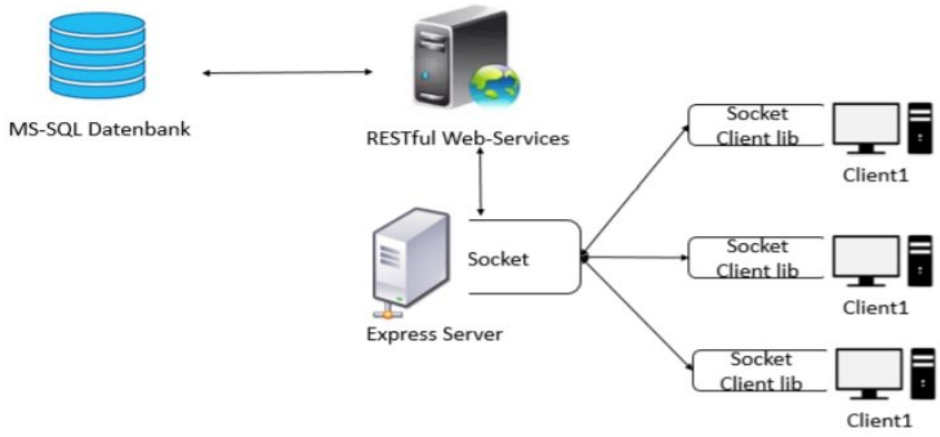
5) RESTful Web Services 발생 메시지 확인

사용자는 RESTful Web Services 에서 발생하는 메시지를 확인 할 수 있어야 합니다.

4.프로젝트 기간

프로젝트는 2019.06.01 부터 2019.08.01까지 2달동안 진행되었습니다.

5.시스템 설계



기존의 시스템은 클라이언트가 직접 RESTful Web Services 를 호출하여 필요한 데이터를 가져오는 방식이였습니다. 새로운 시스템에서는 RESTful Web Services와 통신 및 프론트엔드로 데이터 전달을 위해 Express Node Server가 추가되었습니다.

6. 사용 기술 및 이유

6.1 Frontend

- Angular, Angular Material

프론트엔드 프레임워크로 Angular가 사용되어졌습니다.

사용 이유는 :

1. Express Node 서버와 Javascript, Typescript를 기반으로 개발될 수 있는 이점
2. Angular Material 디자인 컴포넌트를 이용해 사용자에게 효율적인 시스템 디자인을 제공
3. Socket.io 라이브러리로 Node서버와 커뮤니케이션이 가능

- Chart.js

RESTful Web services 서버의 CPU, 메모리 상태는 실시간으로 변하기 때문에 과거의 상태 데이터 또한 표현되어져야 했습니다. 해당 기술을 사용함으로써 Line, Bar형태의 그래픽으로 표현이 가능했습니다.

6.2 Backend

- Socket.io

서버로부터 전달되어지는 데이터를 실시간으로 클라이언트에 전달하기 위해 Socket.io라이브러리가 사용되어 졌습니다. 그밖에 Angular 가 제공하는 데이터 바인딩 기술을 사용할 수 있었지만 Socket.io는 기존 업무에 사용하던 기술로써 복습을 위해 사용하였습니다.

- Notification 라이브러리

RESTful Web services 에서 발생하는 시스템 메시지시 사용자가 바로 인지할 수 있도록 윈도우환경에 표시되어야 했습니다. 이것을 위해 Javascript 의 Notification 라이브러리를 사용하여 기능 구현하였습니다.

6.3 Express Server

6.3.1 Socket.io

Backend와 통신을 위해 Socket.io 라이브러리가 사용되었습니다.

6.3.2 Login

1) Kerberos Server

부서내에서는 Kerberos 유저인증 서비스를 사용중 이였고, 유저가 모니터링 시스템에 접근하게 되면 우선적으로 Kerberos를 통해 유저 정보를 받아옵니다. 로그인 기능 구현을 위해 Kerberos 프로토콜이 사용되었습니다.

2) Express-session

사용자의 기본정보를 관리하기 위해 Express-session패키지가 사용되었습니다.

6.3.3 Request Module

사용자가 모니터링 시스템 접근 권한을 가지고 있는지 확인하기 위해 RESTful Web Services 접근이 필요했고, RESTful Web Services 를 호출할수 있도록 Http request패키지가 사용되었습니다.

6.4 RESTful API Webservices

6.4.1 Message Digest 라이브러리

객체의 무결성 비교를 위해 Message Digest 의 MD5 해시 알고리즘 사용 되었습니다.

6.4.2 Http Request 라이브러리

Express Node서버와 데이터 교환위해 사용되었습니다.

6.4.3 Hibernate

사용자의 기본 정보가 있는 MY-SQL, 그리고 서버 상태, 시스템 정보가 있는 MS-SQL 데이터베이스 서버 연동 및 데이터 가공위해 사용되었습니다.

6.4.4 JavaSysMon 라이브러리

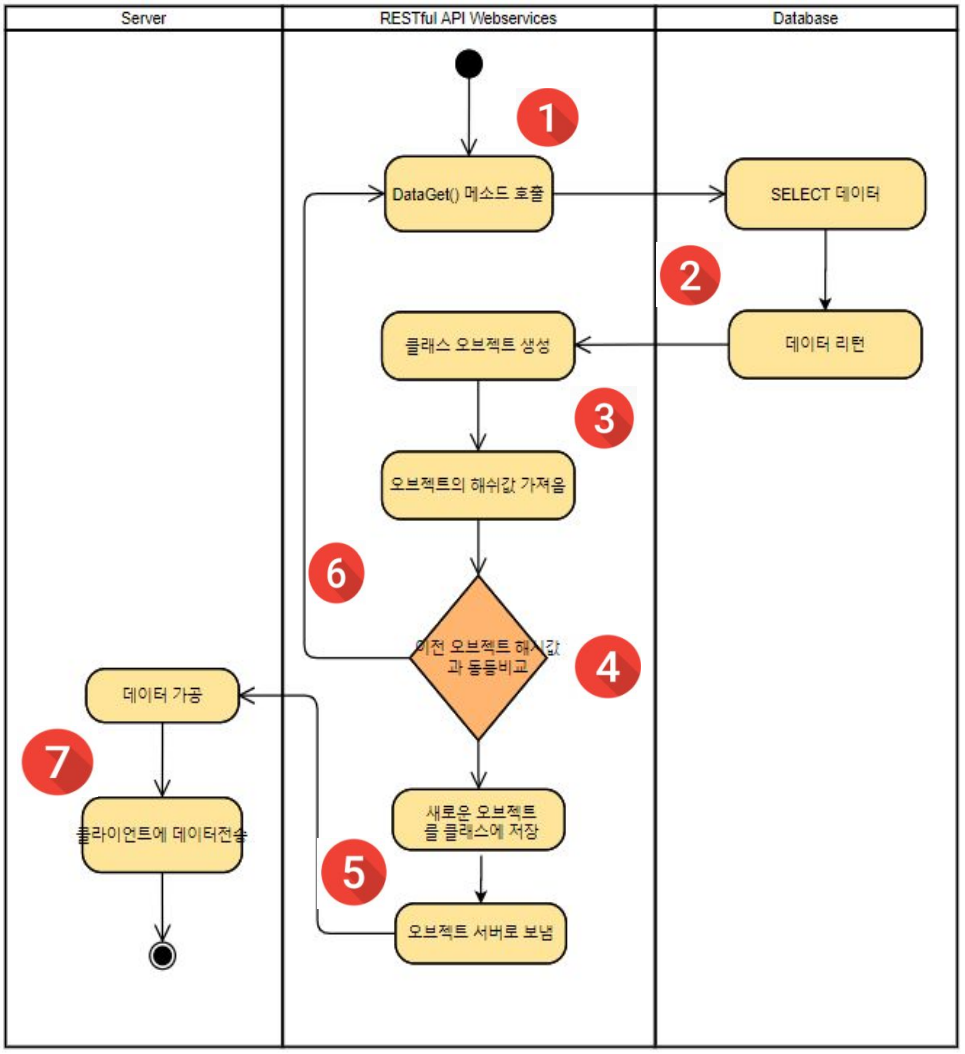
RESTful Webservices 서버의 상태를 서버의 접근 없이 가져오기 위해 JavaSysMon 라이브러리가 사용되었습니다.

7. 프로젝트 결과

7.1 데이터 전송 프로세스 변경

기존의 모니터링 시스템은 서버상태 데이터를 실시간으로 나타내기 위해 SetInterval() 메소드를 통해 RESTful Web services를 여러번 호출하였습니다. 서비스는 요청을 받을때마다 데이터베이스를 접근하고 데이터 가져오기, 데이터 전송과 같은 작업을 반복하여 진행하였습니다. 이러한 프로세스는 서버트래픽 증가 및 비용이 크기때문에 새로운 프로세스 도입이 필요했습니다.

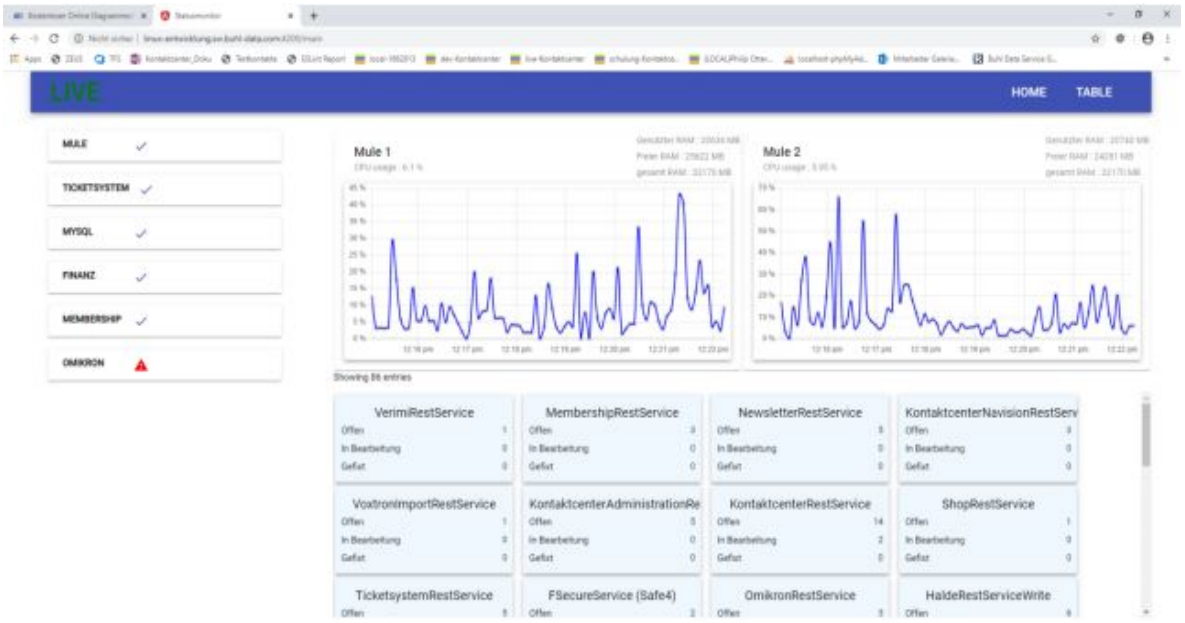
-변경된 프로세스를 액티비티 다이어그램으로 표현:



1. Java의 Cron-Job에 의해 특정 메소드가 호출되어진다.
2. 메소드에서는 시스템에 보여줄 데이터를 가져오기 위해 데이터베이스에 접근한다.
3. 데이터를 가져온 후 데이터클래스의 오브젝트가 생성된다.
4. 생성된 오브젝트의 해시값을 가져와 이전 시스템에 보내진 오브젝트의 해시값과 비교한다.
5. 해시값이 다르면 이전 데이터와 내용이 다르다는 뜻이므로 오브젝트를 클래스 변수 내에 다시 저장 후 서버로 보낸다.
6. 해시값이 다르지않다면 이전 데이터와 내용이 같다는 뜻이므로 1.의 프로세스를 다시 시작한다.
7. 서버에서는 클라이언트에게 데이터를 제공하기 위해 가공 후 클라이언트에게 전송한다.

8. 프로젝트 완성모습

8.1 메인페이지

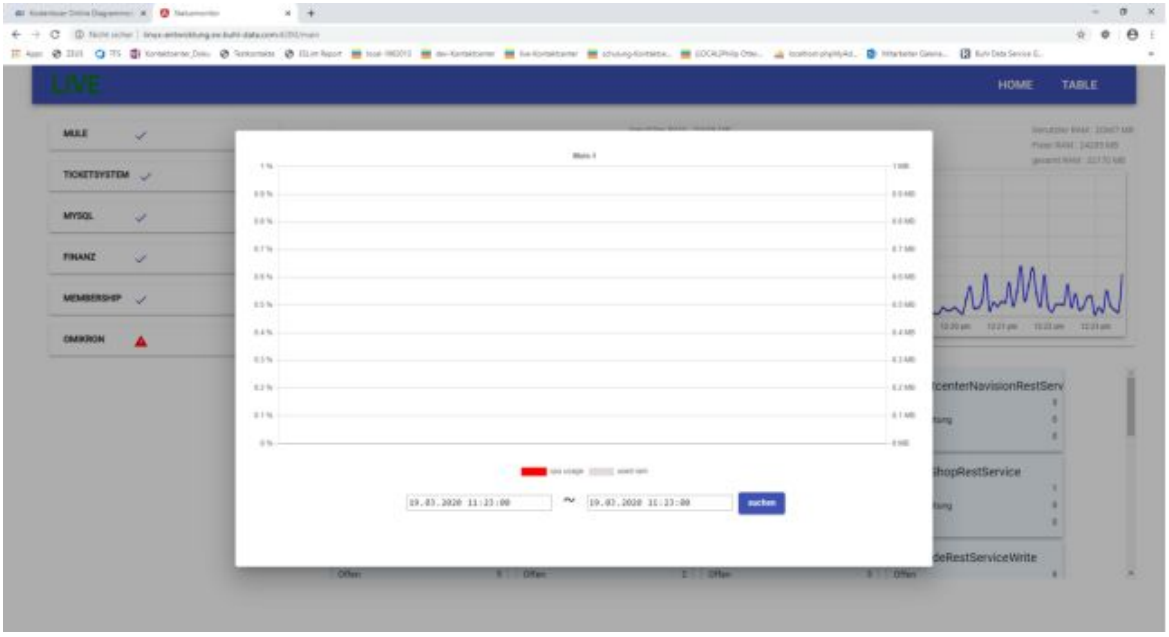


8.1.1 RESTful 서버상태 표현

-RESTful Web Services 서버의 상태는 line 형태로 표현되며, 동시에 RAM의 가용상태또한 표현되었습니다.

8.2 RESTful 과거 서버상태 검색

1) 전체페이지



서버 상태의 표현은Angular material 의 Dialog Component가 사용되었으며, 해당 Component를 클릭시 해당 기능에 접근 가능합니다.

2) 기능설명

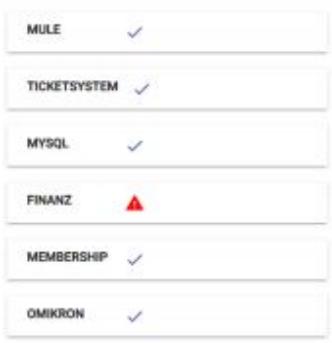
검색할 날짜와 시간 그리고 어떤 데이터가 보여주길 원하는지 CPU, RAM 선택 후 검색 버튼을 누릅니다.

3) 검색 결과 모습

CPU의 상태는 Line으로 표현되고 RAM 가용상태는 Bar 형태로 표현되었습니다.

8.3 RESTful Services와 연결된 서버 상태 표현

RESTful 에서 호출하는 서버는 현재의 상태만 사용자에게 표시합니다.



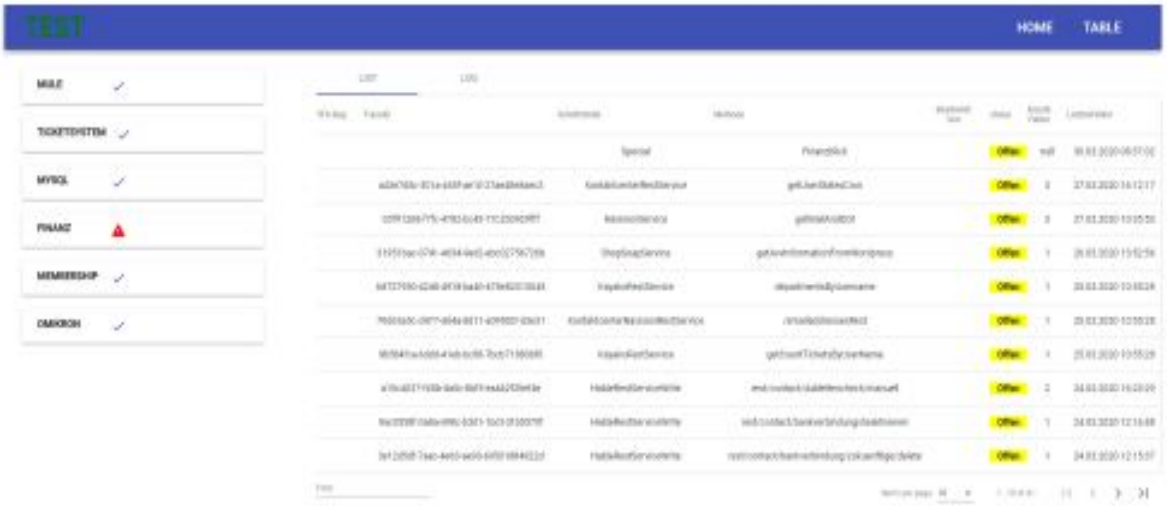
8.4 시스템 발생 메시지 표현

-RESTful Web Services 발생 메시지는 리스트 형태로 한 테이블에 표현됩니다.

-메시지는 RESTful의 Web Services 별로 발생하기 때문에 특정 서비스만 접근할 수 있도록 별도의 인터페이스 생성되어야 했습니다.

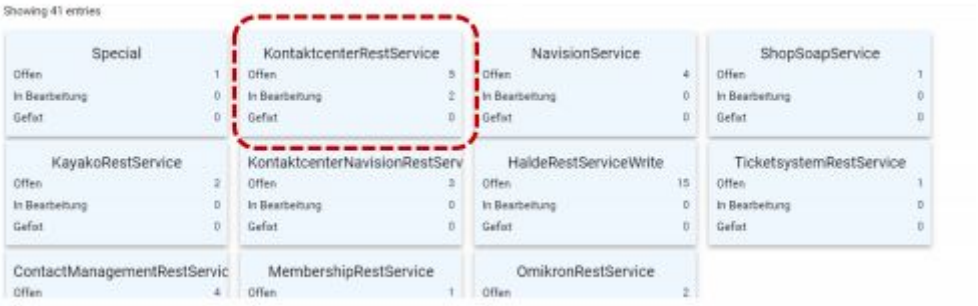
8.4.1 전체 메시지 접근방법: 상단 우측 TABLE 버튼 클릭

8.4.1.1 TABLE 버튼 통해 접근결과



존재하는 모든 서비스 메시지를 보여줍니다.

8.4.2 특정 서비스만 볼수있도록 접근 방법: 하단 서비스 엘리먼트 클릭



8.4.2.1 하단 서비스 엘리먼트 클릭 통해 접근결과

해당 서비스의 메시지만 정렬 후 리스트를 보여줍니다.

9. 저장소 위치

https://github.com/statusmonitor/statusmonitor