DOI : 10.5392/JKCA.2011.11.5.093

SaaS 기반 멀티테넌트 환경을 지원하는 통합전자도서관시스템 구현

Implementation of Integrated Management System for Digital Library Supporting Multi-tenant Environment Based on SaaS

민병원, 오용선
목원대학교 정보통신공학과

Byoung-Won Min(minfam@mokwon.ac.kr), Yong-Sun Oh(ysunoh@mokwon.ac.kr)

요약

현재, 도서관시스템의 소프트웨어 사용 방식은 클라이언트/서버 및 ASP 방식으로 서비스를 제공함으로써 하드웨어 및 소프트웨어 구매비, 설치 및 배포, Customization, Upgrade, 문제점 관리, 라이센스의 고비용 등 소프트웨어 전반에 걸쳐 관리가 힘들고 고비용의 문제점이 있다.

이러한 문제점을 해결하기 위해 SaaS 기반의 전자도서관시스템에서는 멀티테넌트 환경(SaaS 성숙도 레벨 3이상)에서 구현이 가능한 핵심요소들을 개발하였다. 그러므로 초기 투자비용이 거의 없고, 쉽고, 간편하며, 저비용 IT 서비스가 가능한 SaaS 기반의 소프트웨어 온-디멘드 방식의 서비스 모델로 시스템을 구현하였다.

■ 중심어 : | 전자도서관시스템 | 멀티테넌트 | 서비스SW |

Abstract

Currently, the library system using the method of the software on the client / server and ASP by providing services in a manner of hardware and software, and cross-referencing, installation and deployment, Customization, Upgrade, risk management, and software licenses across the high-cost management of difficult and costly There is a problem.

To solve these problems, SaaS based Integrated Management System for Digital Library, multi-tenant environments(SaaS Maturity Level 3 and above) as a key element in the implementation were developed. Therefore, almost no initial investment cost, easy, simple, low-cost IT services available on SaaS based software on-demand service model of how the system is implemented.

■ keyword : | Digital Library System | Multi-Tenant | SaaS |

I. 서 론

요즘 소프트웨어 시장의 주요 트렌드로 클라우드 컴퓨팅, SaaS(Software as a Service), 유틸리티컴퓨팅, SOA, 웹 2.0, RIA 등을 꼽을 수 있다. 이들의 공통점은 서비스에 있으며 서비스는 시장의 주된 성장동력으로 자리매김하고 있다[1]. 이러한 시점에 소프트웨어를 서비스로 발전시킨 SaaS 기술에 대한 관심은 자연스럽다.

* 본 논문은 2010년도 중소기업청 기업부설연구소 지원사업의 일환으로 수행되었음.
접수번호 : #110503-002
접수일자 : 2011년 05월 03일
심사완료일 : 2011년 05월 16일
교신저자 : 오용선, e-mail : ysunoh@mokwon.ac.kr

SaaS 기술은 가트너가 발표한 2009년 10대 전략기술 중에서 첫번째와 두번째를 차지한 가상화와 클라우드 컴퓨팅에 있어서 애플리케이션 가상화 기술에 속하며 클라우드 컴퓨팅의 핵심요소 기술이다[2]. 그리고, SOA, 웹 2.0, RIA들은 소프트웨어를 구현하고 사용, 관리하는 방식의 변화를 설명하는 개념인데 비해서 SaaS는 소프트웨어 유통방식의 변화를 설명하는 개념이다[3]. SaaS는 사용자가 필요한 소프트웨어를 인터넷을 통해 온라인 서비스로 이용할 수 있도록 하는 최신의 소프트웨어 배포 모델로 정의할 수가 있으며[4][5], 또한, 응용 소프트웨어를 인터넷을 통하여 다수의 사용자에게 온라인 서비스로 제공하는 기술로도 정의할 수 있다. 즉, 사용자는 인터넷을 통해 소프트웨어를 사용하고 그에 대한 비용만 지불하는 방식으로 복잡한 소프트웨어 및 하드웨어의 관리라는 부담에서 벗어날 수 있다.

SaaS 비즈니스 모델은 고객의 초기 투자비용이 거의 없고 시스템 관리 필요성도 없어진다. 비용은 매월 또는 매년 서비스 기간에 따라 정해진 금액을 지불하거나 사용량만큼 지불하는 종량제가 있다. SaaS는 패키지화된 애플리케이션을 공급하는 호스티드 애플리케이션 관리 방식과 소프트웨어 및 각종 지원을 인터넷을 통해 다수에게 제공하는 소프트웨어 온-디맨드 방식으로 분류할 수 있다[6].

현재, 도서관시스템의 소프트웨어 사용 방식은 클라이언트/서버 및 ASP 방식으로 서비스를 제공함으로써 HW/SW 구매비, 설치 및 배포, Customization, Upgrade, 문제점 관리, 라이센스의 고비용 등 SW 전반에 걸쳐 관리가 힘들고 고비용의 문제점 발생하고 있다.

본 논문에서 제안한 SaaS 기반 통합전자도서관시스템은 사용자의 커스터마이징을 메타데이터를 활용해서 지원하며 사용자들 또는 사용자 그룹으로 표현되는 테넌트들을 하나의 소프트웨어 인스턴스로 지원 한다는 점에서 차별성을 갖는다. 이는 커스터마이징에 많은 비용이 들고 인스턴스를 개별적으로 띄우기 때문에 규모의 경제를 실현하지 못했던 ASP의 단점을 해결 할 수 있도록 성능을 개선하였다.

또한, 도서관 업무는 표준화 및 모듈화 되어 있으므로 초기 투자비용이 거의 없고, 쉽고, 간편하며, 저비용 IT 서비스가 가능한 SaaS 기반 호스티드 어플리케이션 관리 방식의 서비스 및 소프트웨어 온-디멘드 방식의 서비스 모델로 설계하여 구현하였다.

본 논문은 서론에 이어 제2장에서 시스템의 설계 방법과 구조에 대하여 논하고, 제3장에서 시스템을 구현하는 절차를 제시한다. 이어서 제4장에서는 제안된 시스템의 성능을 분석하여 결과를 제시하였으며, 마지막으로 제5장에서 본 연구의 결과와 향후 연구과제를 제시하였다.

II. 통합전자도서관시스템 설계

2.1 관련 연구

SaaS를 기존의 ASP와 비교해 보면, 사용자의 입장에서는 소프트웨어를 인터넷을 통해 사용하는 점에서는 차별성이 없으나 사용자를 위한 커스터마이징을 ASP처럼 소프트웨어 공급자가 하지 않고 사용자가 직접 할 수 있다는 점에서 차별성을 갖는다. 소프트웨어 공급자의 입장에서는 사용자의 커스터마이징을 메타데이터를 활용해서 지원하며 사용자들 또는 사용자 그룹으로 표현되는 테넌트들을 하나의 소프트웨어 인스턴스로 지원 한다는 점에서 차별성을 갖는다. 이는 커스터마이징에 많은 비용이 들고 인스턴스를 개별적으로 띄우기 때문에 규모의 경제를 실현하지 못했던 ASP의 단점을 해결한다. 정리하면, SaaS 소프트웨어는 다음과 같은 특징을 갖는다[7].

- 네트워크 기반으로 접근하고 관리하는 상업적으로 사용 가능한 소프트웨어
- 각 고객 사이트가 아닌 중앙의 위치에서 활동을 관리하며 고객은 웹을 통해 소프트웨어에 접근
- 애플리케이션 전달은 일반적으로 일대일 모델보다는 일대다 모델에 가까우며, 여기에는 아키텍처, 가격, 파트너링, 관리 특성포함
- 중앙화된 기능 업데이트로 패치와 업그레이드다운로드 필요를 없앰

SaaS를 구현하는 아키텍처는 다양한 테넌트들을 고려한 용이한 환경설정(configurability), 다중 테넌트 지원 정도(multitenant efficiency), 그리고 확장성(scalability)의 특성들을 기준으로 그 성숙도를 다음과 같이 분류할 수 있다[8-12].

- Level 1: Ad Hoc/Custom

전통적인 애플리케이션 제공 서비스인 ASP와 유사한 형태로 각 고객별로 별도의 애플리케이션 인스턴스를 제공하고 커스터마이징하므로 관리가 복잡하고 비용이 큼

- Level 2: Configurable

각 고객별로 별도의 애플리케이션 인스턴스를 제공하고 서비스 내의 설정으로 커스터마이징이 가능 하지만 멀티테넌시를 지원하는 것은 아니며 공급자는 대량의 애플리케이션 인스턴스들을 지원하기 위해서 충분한 하드웨어와 저장공간을 제공해야 함

- Level 3: Configurable, Multi-Tenant-Efficient

모든 고객을 하나의 인스턴스로 지원하고 메타데이터를 통해 커스터마이징하며 멀티테넌시를 제공함. 레벨 2에 비해서 공급자는 효율적인 자원관리가 가능

- Level 4: Scalable, Configurable, Multi-Tenant-Efficient

부하분산 시스템(load-balanced farm)에서 모든 고객을 지원하는 가장 성숙한 모델로 고객의 데이터를 분산관리하며 재구성이 가능한 메타데이터로 각 고객별 커스터마이징을 지원

SaaS의 멀티테넌시란 소프트웨어의 단일 인스턴스가 SaaS 공급자의 서버에서 실행되면서 다중 테넌트들을 지원하는 소프트웨어 아키텍처에 있어서의 원칙을 일컫는다[13-16]. 한편, 클라우드 컴퓨팅에서는 테넌트들간에 하부의 컴퓨팅 자원들을 공동으로 사용하는 것을 클라우드 컴퓨팅의 필수적인 특성으로 보는데 이렇게 논리적으로는 서로 분리되어 있는 테넌트들간의 컴퓨팅자원의 공유를 멀티테넌시라고 일컫기도 한다[17][18].

2.2 통합전자도서관시스템 구조도

본 논문에서 구현하고자 하는 SaaS 기반 Multi-Tenant 환경을 지원하는 통합전자도서관시스템은 [그림 1]과 같이 다중 테넌트 어플리케이션 서비스 계층, SaaS 공통 서비스 계층, SaaS 어플리케이션 생성 환경 계층으로 구성되어 있다[19].

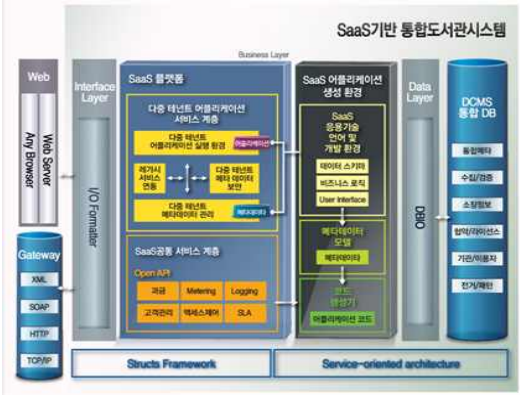


그림 1. 시스템 개념도

다중 테넌트 어플리케이션 서비스 계층의 기능으로는 다중 테넌트 메타데이터 관리기능, 레가시 서비스 연동 기능, 다중 테넌트 메타 데이터 보안 기능, 다중 테넌트 어플리케이션 실행 환경 관리기능 등이 수행되며, 적용기술로써는 메타 데이터 연동 Single-tenant Application 실행 기술, 메타 데이터 연동 Multi-tenant Application 실행 기술, Multi-tenant Load balancing 기술 등이다.

SaaS 공통 서비스 계층의 기능은 Open API로써 고객관리, 과금관리, 액세스관리, 서비스 품질수준(SLA;Service Level Agreement), Metering, Logging 기능 등을 수행하며, 적용기술로써는 SaaS 플랫폼 고객 관리 및 과금 기술, SaaS 서비스 액세스 제어 및 SLA 모니터링 기술, SaaS 서비스 측정(Metering) 기술, SaaS 로깅 기술 등이다.

SaaS 어플리케이션 생성 환경 계층에서는 데이터 스키마관리, 비즈니스 로직 관리, User Interface 등을 수

행하는 SaaS 응용 기술 언어 및 개발 환경관리 기능과 메타데이타 모델관리 기능, 코드 생성기 등으로 구성된다. 적용기술로써는 SaaS Application Description Language 정의, SaaS Metadata Model 정의, SaaS Code Generator, SaaS 응용 개발을 위한 개발 환경, SaaS 플랫폼 연동 시험 환경 기술 등이다.

2.3 서비스 개념도

본 논문에서 구현하고자 하는 통합전자도서관시스템은 [그림 2]와 같이 서비스 라이브러리 컴포넌트, 원내 학술연구정보 통합 레퍼지토리 컴포넌트, 관리용 라이브러리 컴포넌트, 기간시스템 연계 컴포넌트, 정보센터 연계 컴포넌트, 이용자 및 유관기관 컴포넌트 등으로 구성된다.

그림 2. 서비스 개념도

서비스 라이브러리 컴포넌트의 기능으로는 통합검색, 브라우징, 학술저널 및 논문검색, 메타검색, My Library 등의 서비스 구현을 위한 서비스 풀링 기능을 수행한다.

원내 학술연구정보 통합 레퍼지토리 컴포넌트 기능으로는 SOAP/WSDL, XML/MARC, AJAX 기술들을 이용하여 보고서관리, 체크인관리, 기사색인관리, 제본관리, 원문관리, 장서관리, 통합색인관리, 서비DB관리 등을 수행한다.

관리용 라이브러리 컴포넌트의 기능으로는 문서보안, PDF변환, 자연어 색인, 수서, 편목, 연람, 연속 간행물, 기사색인, SDI관리 기능 등을 수행한다.

기간시스템 연계 컴포넌트의 기능으로는 포탈과의 SSO(Single Sign On), Groupware 시스템, 지식관리시스템, 인사관리시스템, CD-NET 시스템 등과의 연계 수행을 인터페이스 기능을 수행한다.

정보센터 연계 컴포넌트에서는 Web DB, 인터넷 서점, 국내외 전자저널, 국내외 학술포탈, 기타 학술포탈 등과의 연계를 위해서 메타검색엔진 기능, NOS(NDSL on SITE) 기능 등을 수행한다.

이용자 및 유관기관 컴포넌트에서는 외부 기관과의 서비스 연계를 위한 프로바이드 기능을 수행한다.

2.4 개발환경

SaaS 플랫폼 기반 통합도서관시스템의 개발 환경은 [그림 3]과 같이 3개의 Layer로 구성되어 있다. 첫 번째 Layer는 SaaS Platform인 Nereus 기반으로 API를 제공한다. 두 번째 Layer는 Spring Framework, AOP, ORM, DAO, MVC, Web, Context 등로 구성된다. Application 구현을 위한 세 번째 Layer는 Web(JSP, JSTL, HTML, CSS, JavaScript, XML), Business Logic(수서, 편목, 열람, 기사색인, 연속간행물 등), iBatis, Oracle로 구성되어 있다.

Spring Framework은 오픈 소스 프레임웍으로서 기업 애플리케이션 개발의 복잡성 문제를 다루기 위해 만들어졌다. Spring Framework의 가장 큰 장점 중 하나는 레이어 아키텍처라는 점이다. J2EE 애플리케이션 개발에 일관된 프레임웍을 제공함과 동시에 원하는 컴포넌트만 선택적으로 사용할 수 있다는 이점이 있다.

iBatis는 비즈니스 로직과 데이터베이스 접근을 분리하여 객체지향을 가능하게 한다. SQL문과 코드를 독립적으로 작성할 수 있어 데이터 처리에 대한 유연성을 제공한다. 또한 트랜잭션 및 캐시 기능을 제공한다.

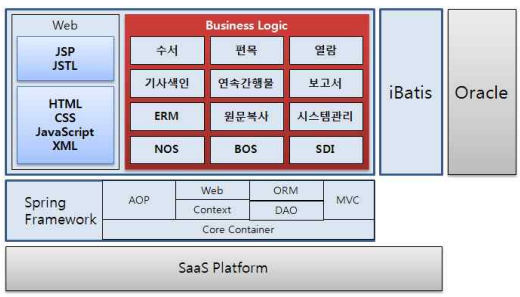


그림 3. 개발 환경 구성도

2.5 기능별 클래스 구조

SaaS 플랫폼 기반 통합도서관시스템의 Application 구현을 위한 클래스는 [그림 4]에서 같이 크게 3개의 구조로 구성이 된다. Controller Class에서는 Web 화면 흐름 정리, Service Class 호출, View(JSP) 처리 등을 수행한다. Service Class에서는 업무로직의 구현, Data 유효성 검사, 트랜잭션 처리, Repository Class 호출 등의 처리한다. Repository Class에서는 데이터베이스를 위한 SQL, Business Logic 등을 수행한다.

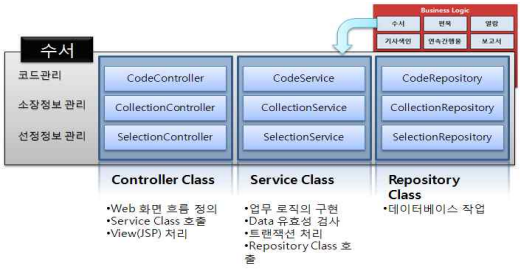


그림 4. 기능별 클래스 구조도

III. 통합전자도서관시스템 구현

3.1 개발 범위

본 논문에서 구현한 시스템 개발 범위는 [그림 5]와 같이 웹기반 도서관 자동화 시스템 분야 12개, 전자정보원 관리 분야 4개, 검색엔진 분야 3개, 언어자원 및 기타분야 3개 등 총 22개 모듈로 구성이 된다.

웹기반 도서관 자동화 시스템 분야는 수서관리, 편목관리, 대출/반납관리, 연간물 시스템, 기사색인시스템, 통합검색시스템, 원문복사서비스시스템, 개인 맞춤정보서비스, 연구보고서관리시스템, 콘텐츠관리시스템 등 통합전자도서관시스템의 주기능을 수행하는 모듈이다.

전자정보원 관리 분야는 NOS(NDSL on Site), BOS(Books on Site), ERM(E-Resource Management), IRS(Institutional Repository solution for dSpace) 등이다.

그림 5. 개발 범위

3.2 서비스 시나리오

SaaS 플랫폼 기반 통합도서관시스템의 서비스 구조는 [그림 6]과 같이 관리자용과 사용자(출판사, Contents Provider, Cross Ref., 타기관 사용자 등)용 서비스로 구분된다.

관리자용 서비스는 수서시스템, 편목시스템, 열람시스템, 기사색인시스템, 연간물시스템, VOD시스템, 원문관리시스템 등 관리자 모듈을 이용하여 데이터베이스 서버에 수서 및 대출정보를 등록, 수정, 삭제 등을 권한을 수행하는 서비스이다.

사용자용 서비스는 유·무선 인터넷을 이용하여 검색서비스, 열람서비스, 원문복사서비스, 최신정보제공서비스, 검색정보관리서비스 등의 기능을 수행하는 서비스이다.

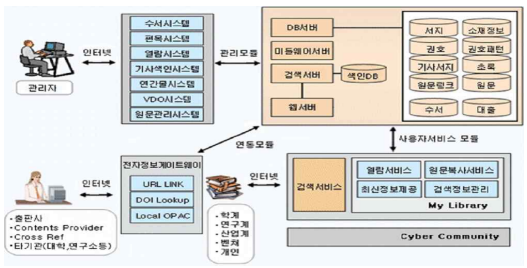


그림 6. 서비스 구조도

3.3 시스템 구현 환경

SaaS 기반 멀티테넌트 환경을 지원하는 전자도서관 시스템의 구현 환경은 [표 1]과 같다. 운영체제로는 Linux5.0을 사용하였고, CPU는 Intel Xeon Dual Core Processor 2.0GHz, 메모리는 2GB ECC DDR SDRAM,

내장 디스크는 146GB/10k rpm SAS Disk * 4 등이다.

표 1. 구현 환경

항 목	세 부 사 양
프로세스	· OS : Linux 5.0 이상 · Type : Intel Xeon Dual Core Processor · Clock Speed : 2.0GHz 이상 · 수량 : 2개 이상(최대 2개까지 확장가능)
메모리	· 2GB ECC DDR SDRAM(최대 8GB Memory이상)이상
내장디스크	· 용량:146GB/10k rpm SAS Disk * 4 이상 · Disk Bay : 6 이상
IO Slot	· 6 Hot-Swap PCI-X 64bit Slots이상
LAN	· Dual 10/100/1000Mbps Ethernet Controller(TP RJ-45)
Graphic-Card	· VIDEO : 8MB SDRAM 이상

3.4 멀티테넌트 환경 구성

멀티테넌트 기술은 단일 인스턴스를 복수의 고객이 공유할 수 있도록 지원하는 가술로서 SaaS 플랫폼에서 멀티태넌시는 개별 사용자(사용자 조직들, 테넌트들)가 자신의 결정에 의하여 설정하여 표현하는 가능이다.

[그림 7]은 본 논문에서 구현된 멀티테넌트 구성도이다. 테넌트들의 그룹을 Nereus Client로 구성하여 서비스를 그룹화 하였으며, 테넌트들의 환경내역을 관리하는 Nerenu Server를 구축하여 구현하였다. 그러므로 컴퓨터의 리소스들을 분할하였으며, 플랫폼과 어플리케이션 코드들을 동시에 접속하는 멀티테넌트를 단일 인스턴스(Single Instance)에서 지원하도록 구성하였다.

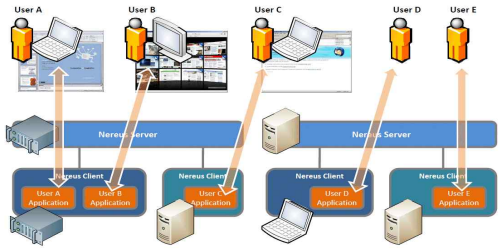


그림 7. 멀티테넌트 구성도

3.5 구현 내용

전술된 시스템의 개발 범위와 서비스 시나리오 개념을 중심으로, 본 논문에서 구현하고자 하는 SaaS 기반 멀티테넌트 환경을 지원하는 전자도서관시스템은 총 22개의 서브시스템과 하부 모듈로 구성된다. 이들 주요 서브시스템의 기능과 구현 내용은 다음과 같다.

1. 관리자용 Application

관리자용 Application은 [그림 8]에서와 같이 핵심 디자인 컨셉은 사용자화(Customise)이다. 각각의 카테고리(메뉴)별로 사용자가 원하는 항목을 편집할 수 있고 화면의 상단으로 편집할 수 있게 디자인하였다. 전체적인 컬러컨셉은 은은하고 화면을 보았을 때 눈의 피로를 덜어줄 수 있는 Green색상을 사용하였다.

새롭게 개편하는 사이트는 디자인적인 요소도 중요하지만 사용성 및 접근성을 고려하여 사용자가 원하는 정보를 쉽게 찾을 수 있도록 하고 태블릿 PC를 포함한 모든 컴퓨터에서 원활하게 접속할 수 있도록 웹표준을 준수하며 개발하는데 초점을 맞추었다. [그림 1]의 ①영역은 업데이트가 자주 일어나는 부분으로 회사의 주요 사업내용을 이미지와 간단한 내용으로 롤링시켜서 고정된 이미지의 사이트가 아닌 변화하고 있는 사이트가 될 수 있도록 하였다. ②영역은 블로그 형식으로 디자인 하여 신규로 구축된 사업이나 진행중인 프로젝트들을 이미지와 간략한 내용을 보여줌으로써 초기화면에서 사용자가 찾을 수 있는 정보의 양을 늘려주도록 하였다. 따라서 태블릿PC 사용자들은 초기화면에서부터 원하는 정보들을 찾을 수 있게 된다. 또한 SNS(소셜네트워크서비스)는 사이트의 홍보에 있어서 빼놓을 수 없는 중요한 요소이기 때문에 트위터나 페이스북 아이콘들을 업데이트되는 개별 항목에 포함하였다. ③영역은 링크소프트 트위터에 올라온 글들을 보여줌으로써 사이트와 사용자들간 효율적인 인터렉션이 이루어지게 할 수 있다.

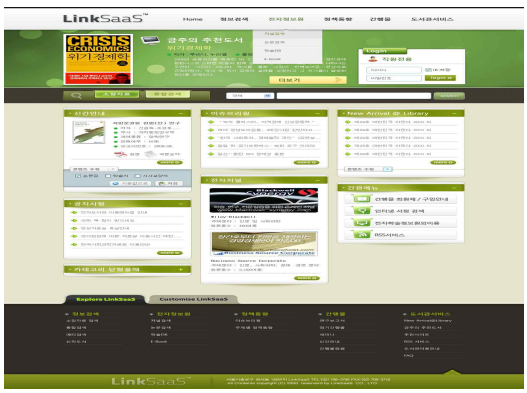


그림 8. 관리자용 Application 화면

2. 사용자용 Application

사용자용 Application은 핵심 디자인 컨셉은 사용자화(Customizing)이다. 각각의 카테고리(메뉴)별로 관리자가 원하는 항목을 편집할 수 있고 화면의 상단으로 편집할 수 있게 디자인하였다. 즉, 관리자(사용자)의 관심 항목이나 업무의 중요도에 따라 항목을 직접 편집할 수 있다.

상단의 메인 메뉴들을 아이콘화하여 관리자가 메뉴의 인식을 보다 쉽게 디자인 했으며 메뉴의 추가 및 삭제 시 보다 유연하게 보여질 수 있도록 좌우 스크롤 형식으로 디자인 되었다.

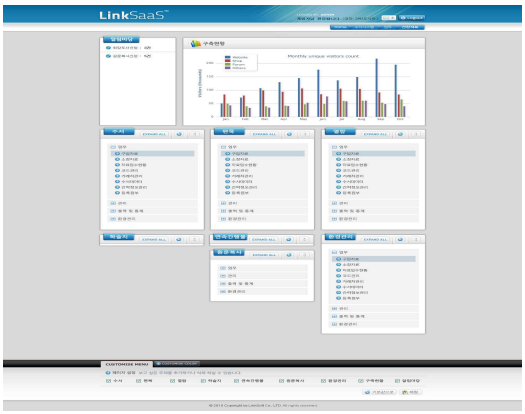


그림 9. 사용자용 Application 화면

테이블의 형식도 기존의 형식적인 테이블 디자인이 아닌, 세련되고 고급스러운 느낌을 전달하도록 디자인하였으며, 디자인 컨셉은 1Depth의 디자인과 통일성을 유지하고 있고, 상세정보보기의 입력폼들은 CSS를 활용하여 보다 세련된 느낌을 주도록 하였다.

IV. 제안 시스템의 성능 분석

제안된 SaaS 기반 멀티테넌트 환경을 지원하는 전자도서관시스템의 성능을 측정하기 위하여, [그림 10]에서와 같이 ①WEB서버, ②DBMS서버, ③미디어서버, ④익스체인지서버, ⑤IPPBX서버, ⑥클라이언트1, ⑦클라이언트2로 구성된 시뮬레이션 환경을 구축하였다.

①번 웹서버로는 IIS 6.0을 사용하였고 제안된 제품의 서버 모듈을 탑재하였다. ②번 DBMS 서버로는 제안된 서버 모듈과 MSSQL 2005를 사용하였다. ③번 미디어서버로는 제안된 서버 모듈과 멀티미디어 서버로 Adobe Media Server 3.1을 사용하였다. ④번 익스체인지서버에는 제안된 서버 모듈과 Red5 Media Server 0.7.0을 설치하였고, ⑤번 IPPBX서버에는 제안된 서버 모듈과 Asterisk 1.4.21 PBX를 설치하여 테스트하였다. ⑥번 클라이언트1에는 제안된 클라이언트 모듈과 웹브라우저 Internet Explorer 6.0과 일반 응용 프로그램

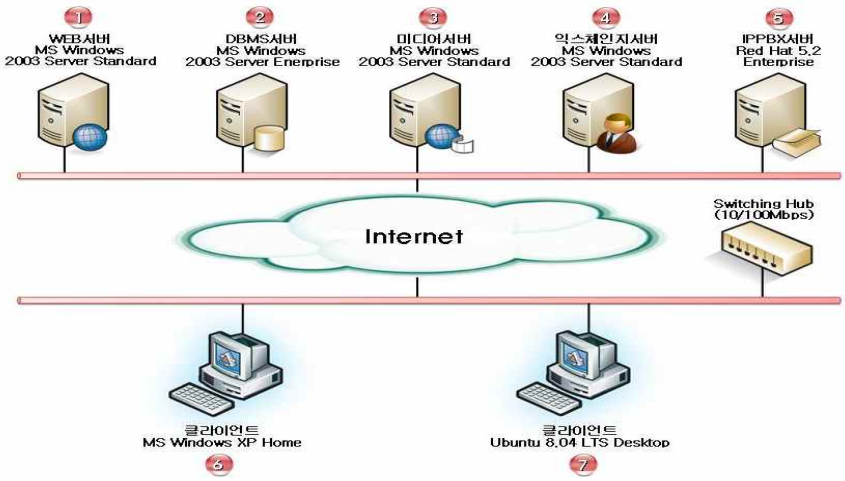


그림 10. 성능분석 환경 구성

인 MS-Office 2007, 한글2007, 바이로봇 Desktop v5.5을 설치하였고, ⑦번 클라이언트2에는 제안된 클라이언트 모듈과 웹 브라우저 Firefox 3.0을 설치하여 클라이언트 모듈 테스트를 실시하였다.

성능 측정 도구로 ①번 서버에 TeamQuest 10.1 Manager를 설치하였고, ⑥번 클라이언트에는 TeamQuest 10.1 View를 설치하였다. 이를 통하여 서버의 자원사용률 측정한다. 또한, ⑥번 클라이언트에는 LoadRunner 8.1을 설치하여 부하 생성 여부를 테스트하였다.

4.1 효율성 분석

1. 분석 시나리오 – 1

동시 사용자 100명이 제안된 전자도서관시스템에 접속하여 데이터의 등록, 수정, 삭제, 조회 기능을 실행하였을 경우, 서버의 자원사용률, 서버의 메모리 사용량 및 서버의 응답시간 등을 측정하였다. 여기서, 자원사용률이란 비유휴 쓰레드 실행에 소비하는 시간의 백분율로 %-processor time으로 평가된다. 메모리 사용량은 컴퓨터에서 실행되고 있는 프로세스에 할당되어 사용된 메모리의 양으로서 'private MBytes'로 측정된다. 또한, 서버의 응답시간은 시스템에서 조회나 요구 등의 명령을 입력한 직후부터 해당 명령의 처리가 완료된 시점까지 소요된 시간으로 '초'를 단위로 하여 측정하였다.

[그림 11]은 이와 같은 분석 시나리오로 서버의 자원사용률(%)과 평균 메모리 사용량을 측정한 결과를 보여주고 있다. 주지하는 바와 같이, 분석 시나리오-1의 조건에서 서버의 최대 자원사용률은 7.81%까지 일시적으로 상승할 수 있으나 해당 기능의 수행을 완료한 후에는 원상태로 복귀하여 안정된 모습을 나타낸다. 한편, 서버의 평균 메모리 사용량은 803MB 정도로 측정되었다. 분석 시나리오-1이 제시하는 플레이 혼잡도에 대하여 제안된 시스템의 CPU 자원과 메모리 자원은 매우 안정된 활용성을 나타내며, 이 조건에서의 데이터 관리는 매우 안정됨을 나타내었다.

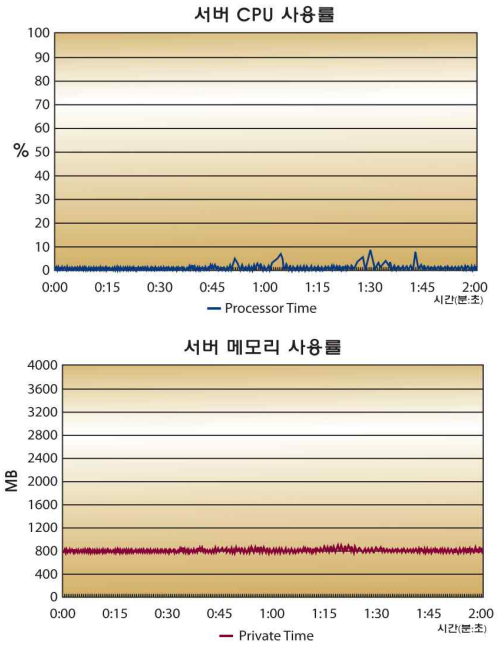


그림 11. 플레이 자원효율성 분석 결과

횟수	1회	2회	3회	4회	5회	평균
시간(초)	55	65	57	67	69	62.9

그림 12. 데이터 변화에 따른 시간효율성 분석 결과

[그림 12]는 분석 시나리오-1에 따른 제안된 시스템의 응답시간을 측정한 결과이다. 서로 다른 데이터의 등록, 수정, 삭제, 조회 명령으로 총 5회 걸쳐 측정을 한 결과, 평균 응답시간은 62.9초로 측정되어 클라이언트 1명당 응답시간은 평균 0.629초이다. 이러한 응답시간은 콘텐츠 실시간 플레이를 보장하는 수준으로 매우 만족스러운 결과로 평가된다.

2. 분석 시나리오 – 2

동시사용자 100명이 제안된 시스템에 접속하여 콘텐츠 '제작' 기능을 실행한 경우, 서버의 자원사용률과 메모리사용량 및 응답시간을 측정하였다.

[그림 13]은 측정 결과를 나타낸다. 서버의 자원사용률은 60.98%까지 일시적으로 증가하지만, 해당 기능 수행을 완료한 후에는 원 상태로 복귀하여 안정되었고, 서버의 평균 메모리사용량은 822MB로 측정되었다. 분석 시나리오-2가 제시하는 콘텐츠 제작 혼잡도에 대하여 제안된 시스템의 CPU 자원과 메모리 자원은 매우 안정된 활용성을 나타내며, 이 조건에서의 콘텐츠 제작 관리는 매우 안정됨을 나타내었다.

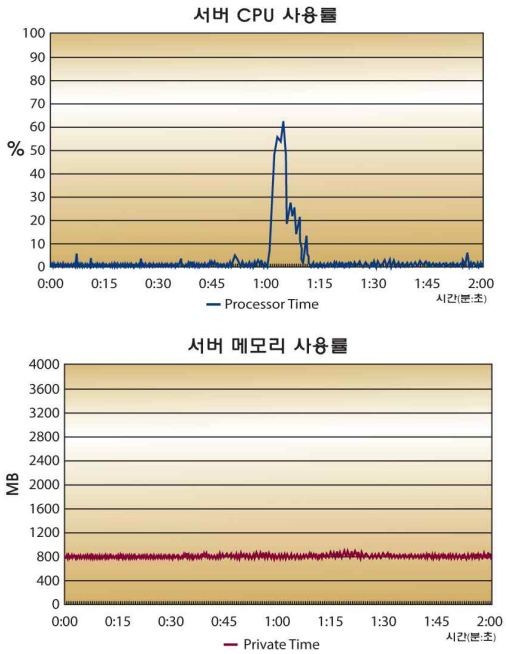


그림 13. 제작 자원효율성 분석 결과

[그림 14]는 시나리오-2에 따른 콘텐츠 제작시 서버의 응답시간을 측정한 것이다. 서로 다른 제작 명령에 의하여 총 5회에 걸쳐 측정한 결과 평균 79.8초로 나타났다. 시나리오-2에 따른 콘텐츠 제작시의 응답시간은 서버의 부하 분산 기능 및 응답속도의 개선을 위한 지능형 버퍼 기능이 우수하게 작용하여 매우 만족스러운 결과를 보여주었다.

횟수	1회	2회	3회	4회	5회	평균
시간(초)	65	99	74	82	79	79.8

그림 14. 제작 시간효율성 분석 결과

3. 분석 시나리오 – 3

동시 사용자 100명이 제안된 시스템에 접속하여 콘텐츠 '암호화' 및 불법복제 차단을 위한 'Ticket 인증' 기능을 수행하였을 경우, 서버의 자원사용률과 메모리 사용량 및 응답시간을 측정하였다.

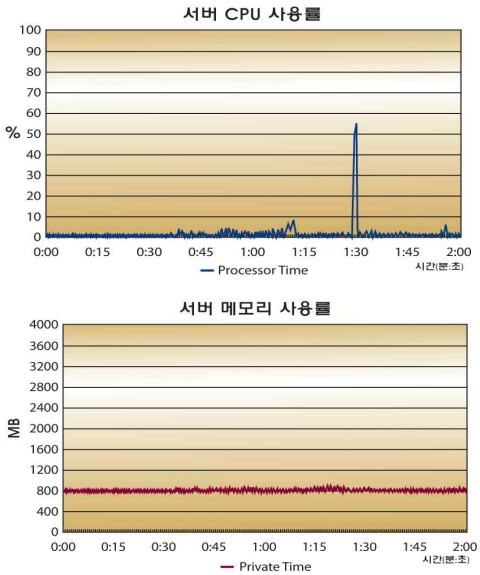
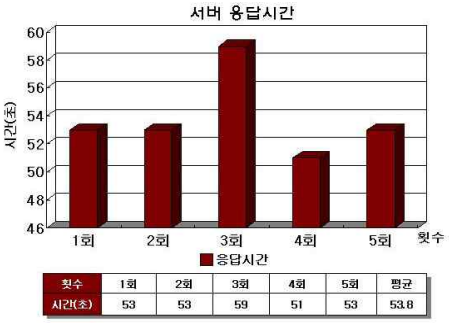


그림 15. 암호화 등 자원효율성 분석 결과

[그림 15]는 시나리오-3의 분석 조건에서 측정된 서버의 자원사용률과 메모리 사용량을 나타낸다. 서버의 CPU 사용률은 54.51%까지 일시적으로 증가하지만, 해당 기능 수행을 완료한 후에는 원 상태로 복귀하여 안정되는 것으로 측정되었고, 서버의 평균 메모리 사용량도 825MB 정도로 안정되었다.

또한 [그림 16]은 시나리오-3에 따르는 콘텐츠 암호화 및 Ticket 인증기능을 수행하여, 서버의 응답시간을 측정한 결과를 나타낸다. 총 5회에 걸친 서로 다른 암호화 및 인증 명령을 부여한 결과 평균 53.8초로 분석되었다. 이러한 측정 결과는 제안된 시스템이 콘텐츠 보안 기능과 콘텐츠 불법복제 차단 기능을 매우 적절하게 수행하고 있는 것을 나타낸다.



횟수	1회	2회	3회	4회	5회	평균
시간(초)	53	53	59	51	53	53.8

그림 16. 암호화 등 시간효율성 분석 결과

4.2 제안된 시스템의 성능향상 분석

본 논문에서 제안한 SaaS 기반 멀티테넌트 환경을 지원하는 전자도서관시스템은 크게 5가지의 기능이 향상되었다.

- **멀티테넌트 지향의 어플리케이션 리소스 공유, 최적화**

프로세스들은 다중 사용자화의 실행을 통한 이익을 위하여 공유된 메모리와 프로세스 공간 안에서 실행되어야 하지만 보여지는 메모리, 프로세스 상태, 메타데이터 설정, 성능의 변동 그리고 테넌트간 잘못된 에러의 발생 처리 등은 테넌트별로 격리되어 보호받아야 한다.

- **멀티테넌트 지향의 데이터 공간의 공유와 격리**

어플리케이션 플랫폼은 데이터 저장소에 의하여 분리된 기술 레이어를 가지고 있으며, 데이터 레이어의 기능은 멀티테넌시를 보장 하도록 구현하였다. 데이터 레벨의 멀티테넌시는 자동적으로 데이터 저장소에 의하여 구현되거나 플랫폼에 의하여 자동적으로 구현되거나 어플리케이션에 의하여 설계되어 구분된다.

- **멀티테넌트 지향의 백엔드 데이터 관리**

데이터베이스는 더 이상 한명의 사용자를 위하여 존재하지 않기 때문에 어느 시점으로 데이터를 복구하여야 할지 정할 수가 없다. 그러므로 본 논문에서 제안된 시스템은 데이터 백업 및 복구, 장애, 롤백, 롤포워드, 진단, 임포트/익스포트 그리고 다른 데이터베이스관리자(DBA) 지향의 백엔드 데이터베이스 프로세스는 반드시 엄격하게 멀티테넌트 지향적으로 구현하였다.

- **일정한 간격을 둔 멀티테넌트 지향의 호스팅**

테넌트의 어플리케이션 데이터 또는 비즈니스 로직은 반드시 다른 테넌트로부터 보호받아야 한다. 그러므로 호스트 제공자는 어플리케이션의 전체 동작을 지원해야하며, 단지 테넌트에게 호스트 DBA나 다른 호스트의 테넌트의 비즈니스 데이터에 접근할 수 있는 기능이 구현된다.

- **멀티테넌트 지향의 보안, 모니터링, 리포팅관리**

테넌트와 연관된 사용자는 테넌트 밖에 있는 리소스들에 대하여 보여지는 것을 막아야 하며, 각 테넌트별 격리된 모니터링과 독립된 관리 인자와 정책을 제공받을 수 있도록 구현된다.

V. 결 론

도서관시스템의 소프트웨어 사용 방식은 클라이언트/서버 및 ASP 방식으로 서비스를 제공함으로써 하드웨어 및 소프트웨어 구매비, 설치 및 배포, Customization, Upgrade, 문제점 관리, 라이센스의 고비용 등 소프트웨어 전반에 걸쳐 관리가 힘들고 고비용의 문제점이 있다.

이러한 문제점을 해결하기 위해 SaaS 기반의 전자도서관시스템에서는 멀티테넌트 환경(SaaS 성숙도 레벨 3이상)에서 구현이 가능한 핵심요소들을 개발하였다.

그러므로 초기 투자비용이 거의 없고, 쉽고, 간편하며, 저비용 IT 서비스가 가능한 SaaS 기반의 소프트웨어 온-디멘드 방식의 서비스 모델로 시스템을 구현하였다.

본 논문에서 제안된 시스템의 특징으로는 멀티테넌트 지향의 어플리케이션 리소스 공유 및 최적화, 멀티테넌트 지향의 데이터 공간의 공유와 격리, 멀티테넌트 지향의 백엔드 데이터 관리, 일정한 간격을 둔 멀티테

넌트 지향의 호스팅 방법으로서 기존 시스템들의 문제점들을 많은 부분 개선하였다.

하지만 JavaEE로 구현되었거나 .NET 어플리케이션 플랫폼으로 구현된 전자도서관시스템의 어플리케이션들은 단순하거나 격리된 태넌시 모델의 배포를 사용하여 SaaS 시나리오의 요구사항을 만족시킬 수 없었다. 향후 이 분야에 대한 연구가 필요하다.

참 고 문 헌

[1] 신현석, "MS 클라우드 컴퓨팅과 애저 서비스 플랫폼 이해," ZDNet Korea, 2008. 12. 12.

[2] 김영만, "웹기반 SaaS 플랫폼," 2008. 12. 17., http://www.software.or.kr/ICSfile/afieldfile/2008/12/18/5.pdf

[3] 디지털타임스, "SaaS와 소프트웨어의 미래," 2006. 5. 4.

[4] http://en.wikipedia.org/wiki/Software_as_a_service

[5] What is SaaS?, http://www.salesforce.com/saas/

[6] 전자신문, "[집중진단-SaaS, Web 2.0시대 연다] (상) SaaS 시대 개막," 2007. 2. 12.

[7] http://ko.wikipedia.org/wiki/SaaS

[8] Frederick Chong and Gianpaolo Carraro, "Architecture Strategies for Catching the Long Tail," Microsoft, 2006(4). (http://msdn.microsoft.com/en-us/library/aa479069.aspx)

[9] http://en.wikipedia.org/wiki/Multitenancy

[10] Yefim V. Natis, "Reference Architecture for Multitenancy: Enterprise Computing," in the Cloud, Gartner, pp.4-8, 2008. 12. 3.

[11] 디지털데일리, "클라우드 컴퓨팅과 비즈니스의 진화," 2009. 5. 21.

[12] http://www.salesforce.com/platform/

[13] "Force.com: Create and Run any Application, On Demand," Salesforce.com

[14] SaaSGrid Quickstart, apprenda, 2008. 4.

[15] Developer Guide, apprenda, 2008. 4.

[16] Software-plus-Services, Microsoft, 2009.

[17] Introducing the Azure Services Platform, David Chappell, Chappell & Associatest, 2008.

[18] Mary-Jo Foley, "Microsoft Takes Off Its xRM

[19] 민병원, 오용선, "SaaS 플랫폼 기반 통합전자도서관시스템 설계", 한국콘텐츠학회 2010 춘계 종합학술대회 논문집, pp.447-449, 2010.

저 자 소 개

민 병 원(Byoung-Won Min) 종신회원

- 2005년 2월 : 중앙대학교 대학원 컴퓨터소프트웨어학과(공학석사)
- 2010년 2월 : 목원대학교 대학원 IT공학과(공학박사)
- 2005년 4월 ~ 2008년 2월 : 영동대학교 컴퓨터공학과 전임강사
- 2008년 3월 ~ 2011년 2월 : 목원대학교 산학협력단 전임강사
- 2011년 3월 ~ 현재 : 목원대학교 공과대학 정보통신공학과 전임강사

<관심분야> : 온톨로지, U-Health, 모바일콘텐츠, 클라우드 컴퓨팅, SaaS, 모바일 클라우드, 스마트 도서관

오 용 선(Yong-Sun Oh) 종신회원

- 1983년 2월 : 연세대학교 공과대학 전자공학과(공학사)
- 1985년 2월 : 연세대학교 대학원 전자공학과(공학석사)
- 1992년 2월 : 연세대학교 대학원 전자공학과(공학박사)
- 2007년 9월 ~ 2008년 8월 : 한국전자통신연구원(ETRI) 초빙연구원
- 1988년 3월 ~ 현재 : 목원대학교 공과대학 정보통신공학과 교수
- 2006년 7월 ~ 현재 : 한국콘텐츠학회 회장

<관심분야> : 디지털통신시스템, 정보공학, 멀티미디어 콘텐츠, 맞춤형 이러닝