creative commons

COMMONS DEED

저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:

저작자표시. 귀하는 원저작자를 표시하여야 합니다.

비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.

변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 이용허락규약(Legal Code)을 이해하기 쉽게 요약한 것입니다.

Disclaimer

dCollection

공학전문석사 학위 연구보고서

오픈 소스를 활용한 Web UI Framework 설계

2021년 2월

서울대학교 공학전문대학원

응용공학과 응용공학전공

김 한 성

오픈 소스를 활용한
Web UI Framework 설계

지도교수 박 근 수

이 프로젝트 리포트를 공학전문석사 학위
연구보고서로 제출함
2021년 2월

서울대학교 공학전문대학원
응용공학과 응용공학전공
김 한 성

김한성의 공학전문석사 학위 연구보고서를 인준함
2021년 2월

위 원 장 _____ 박 진 우 _____ (인)

위 원 _____ 박 근 수 _____ (인)

위 원 _____ 서 은 석 _____ (인)

국문초록

지난 20여 년간 인터넷 인프라의 발전으로 인터넷을 이용한 서비스는 기하급수적으로 증가하였다. 또한 모바일 장치의 등장과 함께 인터넷을 활용한 모바일 환경도 폭발적으로 성장하였다. 모바일 환경의 성장으로 다양한 운영체제와 해상도를 가지는 장치가 보편화 되었으며, 이로 인해 기존 프로그램이나 서비스들은 사용자 편의를 위하여 모바일 환경을 지원하도록 꾸준히 변화하고 있다.

한편, A사의 기간계 업무 시스템은 위와 같은 추세에 맞추어 업무 시스템을 인터넷 환경 기반으로 전환하여 제공하고 있지만, 시스템 사용자는 여전히 특정 운영체제와 개인용 컴퓨터(Personal Computer)를 사용해야 한다.

이에 본 연구에서는 A사의 기간계 업무 시스템을 다양한 장치와 운영체제를 지원할 수 있도록 반응형 웹 기반의 UI 프레임워크를 설계하고 구현하였다. 더불어 UI 프레임워크에 포함되는 라이브러리나 컴포넌트는 오픈 소스 소프트웨어를 사용하여 구매 비용이나 유지보수 비용을 절감하고 유연한 확장성을 확보하였다.

마지막으로 본 연구에서 설계한 UI 프레임워크를 사용하여 실제 운용 중인 업무 화면 5본을 선정하여 구현하였다. 이를 통해 업무 화면의 기능들이 다양한 운영체제와 모바일 장치에서 현재의 시스템과 동일하게 동작하는 것을 검증하였다.

주요어 : 사용자 인터페이스, 반응형 디자인, 웹, 프레임워크, 오픈 소스
학 번 : 2019-21204

목 차

제 1 장 서론 ····· 1
제 1 절 연구동기 ····· 1
제 2 절 문제정의 및 연구내용 ····· 4
제 3 절 연구보고서 구성 ····· 9

제 2 장 연구배경 및 관련연구 ····· 10
제 1 절 연구 배경 ····· 10
제 2 절 관련 연구 ····· 14

제 3 장 As-Is 분석 및 To-Be 기획 ····· 18
제 1 절 요구사항 분석 및 확정 ····· 18
제 2 절 UI 프레임워크 기술 구조 선정 ····· 29
제 3 절 오픈 소스 라이브러리 선정 ····· 33

제 4 장 설계 및 구현 ····· 39
제 1 절 구조설계 ····· 39
제 2 절 구현 ····· 44

제 5 장 결론 ····· 72
제 1 절 요약 및 결론 ····· 72
제 2 절 연구성과 및 기대효과 ····· 73
제 3 절 고찰 ····· 74
제 4 절 향후 계획 ····· 75

참 고 문 헌 ····· 76
Abstract ····· 78

표 목 차

[표 1-1] 본 연구 결과가 만족해야할 조건 네 가지 조건 ····· 7
[표 3-1] 테스트를 진행한 PC 환경 ····· 36
[표 3-2] 테스트 조건 ····· 36
[표 3-3] 본 연구에 사용할 라이브러리의 라이선스 내역 ····· 38
[표 4-1] 컨트롤 종류에 따른 구분 ····· 43
[표 4-2] 컨트롤의 종류 정의 ····· 43
[표 4-3] include_once를 이용한 공통영역 ····· 45
[표 4-4] 표준 통신 함수 정의 ····· 47
[표 4-5] 바인딩을 위한 컨트롤 예시 ····· 54

그 림 목 차

[그림 1-1] 운영체제별 점유율 추이 (2009.01-2020.06) ····· 2
[그림 1-2] 기간계 시스템 UI 예시 ····· 4
[그림 2-1] 브라우저별 점유율 추이 (2009.01-2020.06) ····· 11
[그림 2-2] 반응형 웹의 장치별 레이아웃 변화 ····· 12
[그림 3-1] 일반 화면 ····· 19
[그림 3-2] 그리드 화면 ····· 20
[그림 3-3] 복합 화면 ····· 21
[그림 3-4] 트리 화면 ····· 22
[그림 3-5] 1:N 데이터 관리 화면 ····· 23
[그림 3-6] 멀티셀렉트 리스트 컨트롤의 예시 ····· 23
[그림 3-7] http와 https의 차이 ····· 25
[그림 3-8] 데이터 바인딩의 예시 ····· 26
[그림 3-9] 웹사이트 레이아웃(좌)와 인쇄물(우)이 다른 예시 ····· 27
[그림 3-10] 공통 검색 컨트롤 예시 ····· 28
[그림 3-11] JavaScript library market share trends ····· 30
[그림 3-12] 오픈 소스 소프트웨어의 라이선스 분류 ····· 33
[그림 3-13] Design Template 'concept' ····· 34
[그림 3-14] Design Template에서 제공하는 다양한 컨트롤 ····· 35
[그림 3-15] jsGrid 컨트롤 예제 ····· 35
[그림 3-16] 그리드 컨트롤 테스트 흐름도 ····· 36
[그림 3-17] 테스트 결과 ····· 37
[그림 4-1] 화면의 구성 ····· 39
[그림 4-2] Bootstrap의 Grid System options [25] ····· 40
[그림 4-3] 그리드 시스템 예시 (1) ····· 40
[그림 4-4] 그리드 시스템 예시 (2) ····· 40
[그림 4-5] 다중 클래스 지정 시 그리드 레이아웃의 변화 ····· 41

[그림 4-6] 통신 방식 구조도 ····· 42
[그림 4-7] 디자인 템플릿 ····· 44
[그림 4-8] 수정된 템플릿 ····· 44
[그림 4-9] 로그인 템플릿 수정 ····· 46
[그림 4-10] 로그인 과정 흐름도 ····· 46
[그림 4-11] getHttpRequestJson의 소스코드 ····· 48
[그림 4-12] 공통 모듈의 사용예 ····· 49
[그림 4-13] 데이터 바인딩의 방향 ····· 50
[그림 4-14] 조회 컨트롤 예시 ····· 51
[그림 4-15] 조회 조건 처리 코드 ····· 51
[그림 4-16] getRCond의 결과를 이용하여 데이터 요청 예제 ····· 52
[그림 4-17] JSON 응답 예시 ····· 53
[그림 4-18] 데이터 바인딩 처리 흐름도 ····· 54
[그림 4-19] bindCUDVal 함수 코드 ····· 55
[그림 4-20] CUD를 위한 컨트롤 코드 예시 ····· 56
[그림 4-21] getCUDVal 사용 예시 ····· 56
[그림 4-22] Bootstrap에서 제공하는 팝오버 ····· 57
[그림 4-23] 팝오버 Wrapper Function 코드 ····· 57
[그림 4-24] 팝오버 호출 코드 ····· 58
[그림 4-25] 팝오버 표시 예 ····· 58
[그림 4-26] 화면1 레이아웃 ····· 59
[그림 4-27] 화면1 모바일 레이아웃 ····· 60
[그림 4-28] 화면1의 인쇄 레이아웃 ····· 61
[그림 4-29] 화면2 레이아웃 ····· 62
[그림 4-30] 가로 해상도가 작은 환경에서 그리드 컨트롤 동작 예시 ··· 62
[그림 4-31] 엑셀다운로드 예시 코드 ····· 63
[그림 4-32] 다운로드 된 엑셀 파일 ····· 63
[그림 4-33] 화면3 레이아웃 ····· 64
[그림 4-34] 화면3 모바일 레이아웃 ····· 65

[그림 4-35] 화면3 인쇄 레이아웃 ····· 66
[그림 4-36] 화면4 레이아웃 ····· 67
[그림 4-37] 화면4 트리 데이터의 구조 ····· 67
[그림 4-38] 화면4 모바일 레이아웃 ····· 68
[그림 4-39] 화면5 레이아웃 ····· 69
[그림 4-40] 화면5 모바일 레이아웃 ····· 70
[그림 4-41] 화면5 인쇄 레이아웃 ····· 71

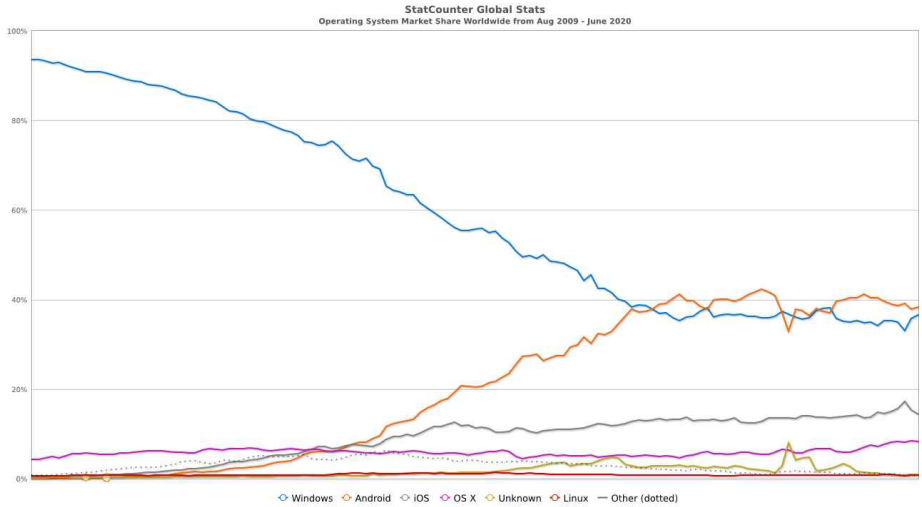
제 1 장 서론

제 1 절 연구동기

지난 20여 년간 인터넷은 엄청난 속도로 발전하였고 현재는 인터넷 없는 사회를 상상하기 어려울 정도로 우리의 삶에 많은 영향을 미치고 있다. 인터넷은 1969년 미국 국방성에서 UCLA와 스탠퍼드 대학교 등에서 보유한 컴퓨터를 네트워크로 연결한 ARPANET을 시초로 인정하고 있으며, 국내에서는 1982년 5월 서울대학교 컴퓨터공학과 중형컴퓨터와 구미의 전자 기술 연구소(현 한국전자통신연구원)의 중형컴퓨터가 1200 bps 전용선을 이용한 TCP/IP 통신을 개시한 SDN(System Development Network)을 구축하여 전 세계에서 두 번째로 인터넷망을 연결하였다 [1]. 이후 국제적으로 '월드 와이드 웹(WWW)'의 확산과 더불어 국내의 초고속 통신 인프라의 구축으로 인터넷을 통한 서비스는 기하급수적으로 늘어나게 되었다.

금융기관 A사[1)]는 금융업무의 원활화를 위하여 설립되었으며, 업무 변화와 사용자의 편의를 위해 지속적으로 전산 업무 시스템을 개선하고 있다. 초고속 통신 인프라와 인터넷의 보급과 발을 맞추어 인터넷을 통하여 시스템에 접속 할 수 있도록 하는 등 네트워크 부분도 계속 발전시켜왔다. 하지만 지속적인 개선 과정에서도 변하지 않는 부분이 있었는데, 이는 시스템 사용자가 이용하는 UI 클라이언트 프로그램은 특정 운영체제 환경에 종속적이라는 점, 그리고 모바일 환경은 지원하지 않고 있다는 것이다.

1) 회사 내규에 따라 비공개 처리함



[그림 1-1] 운영체제별 점유율 추이 (2009.01-2020.06)
(이 그림은 statcounter의 Creative Common License에 의해 제공됨)

2007년 애플사에서 아이폰이 발표된 이후 모바일 기기를 통한 인터넷 사용자는 지난 10여 년간 꾸준히 증가하였다. [그림 1-1]의 statcounter의 통계에 따르면 집계를 시작한 2009년 1월에 94.8%를 점유하던 윈도우즈 운영체제는 2017년 4월 처음으로 구글(Google)사의 안드로이드 (Android) 운영체제에 점유율 1위 자리를 내주었으며, 2020년 6월 현재 안드로이드 38.3%, 윈도우즈 36.55%, iOS 13.3%, OS X 8.35%의 점유율을 기록하고 있다 [2]. 이는 전체 운영체제 시장에서 모바일 운영체제(안드로이드 및 iOS)의 점유율이 51.6%로 PC 운영체제와 동등하거나 상회하는 수준이다.

모바일 환경의 성장은 모바일 애플리케이션(이하 모바일 앱)과 모바일 웹페이지(이하 모바일 웹)를 통한 서비스 증가로 이어지고 있으며, 앞으로도 많은 비즈니스와 서비스가 이러한 추세에 따라 모바일 환경을 지원하는 형태로 구축될 것으로 보인다.

한편 금융기관 A사의 기간 업무 시스템은 서버 구동 소프트웨어부터 개발 프레임워크 및 개발 도구 등 거의 모든 소프트웨어를 상용 제품으로 사용한다. 또한 도입한 상용 소프트웨어의 유지보수를 위하여

매년 유지보수 계약을 체결하고 있으며 이렇게 한 해 유지보수를 위한 비용으로 약 20억 원이 지출되고 있다. 금융권 시스템에서 상용 소프트웨어를 선호하는 배경에는 타 기관에서 이미 많이 적용되어 소프트웨어의 안정성이 검증된 소프트웨어를 선호하는 경향에 따른 것으로 생각된다.

하지만 시스템을 구성하는 다양한 요소 중에서 시스템 사용자(End-User)가 사용하는 UI(User Interface)의 경우 다른 시스템 구성 요소보다 안정성이 강하게 요구되지는 않는 편이다. 오히려 범용성이 좋을수록 사용자의 편의성이 좋아진다. 이러한 관점에서 웹(Web) 기반의 UI는 어떠한 운영체제든 웹브라우저만 구동되면 동작할 수 있어 매우 범용적이다. 그러므로 UI와 관련된 개발 환경과 서비스를 오픈 소스 기반의 웹 환경으로 전환한다면 사용자의 편의성을 개선하면서도 비용 절감 효과도 기대 할 수 있다.

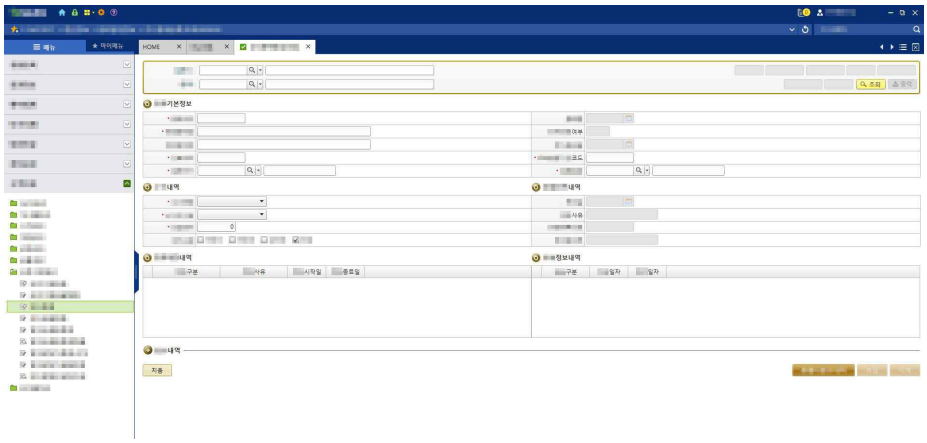
본 연구는 모바일이나 태블릿 등 다양한 장치를 지원하여 사용자의 편의성을 높이고 오픈 소스를 활용하여 향후 유지비용 절감하면서도 특정 장치나 운영체제 또는 특정 브라우저에 종속되지 않는 서비스를 제공할 수 있는 웹 기반 UI와 개발 프레임워크를 설계 및 구현하는 것을 목표로 한다. 또한 실제 운영되고 있는 기간계 시스템의 업무 화면[2]들을 본 연구에서 설계하고 구현한 프레임워크를 이용해서 구현하였을 때 기존과 같이 동일하게 동작 할 수 있는지를 검증하여 이번 연구의 실효성을 확인하고자 한다.

2) 업무 화면 : 기간계 시스템에서 메뉴와 대응되는 업무의 단위프로그램

제 2 절 문제정의 및 연구내용

1. 문제정의

현행 금융기관 A사에서 서비스하는 기간계 시스템의 사용자 UI는 아래 [그림 1-3]과 같다.



[그림 1-2] 기간계 시스템 UI 예시

위 기간계 시스템 UI는 특정 운영체제를 사용해야만 동작할 수 있으며 또한 상용프로그램 또는 상용솔루션을 사용하고 있다. 이 기술구조는 '종속적인 운영체제', '유지보수료 과다', '비유연한 확장성', '모바일 환경 미지원'의 문제점을 수반한다.

· **종속적인 운영체제** : 현재와 같이 모바일 환경이 성장하기 전에 국내외에서 가장 널리 사용되는 개인용 컴퓨터(Personal Computer)의 운영체제는 마이크로소프트 윈도우즈(Microsoft Windows)로 금융기관 A사의 기간계 시스템은 이 운영체제에서 클라이언트 프로그램이 구동되도록 설계되었다. 즉, 금융기관 A사 기간계 시스템의 클라이언트는 태생적으로 운영체제에 대한 종속성을 가지고 있다. 그래서 애플사의 Mac OS나 리눅스 등 다양한 OS 지원할 수 없으며 모바일 환경 사용자 또한 사용할 수 없다. 이는 시스템 사용자에게 특정한 운영체제만을 사용하도록 강제하는 문제점을 야기한다.

· **유지보수료 과다** : 금융기관 A사 시스템의 UI 클라이언트 관련 개발환경 관련 프로그램은 모두 상용제품으로 도입 초기에 구매 비용이 발생하였다. 또한, 매년 유지보수 또는 기술지원을 위하여 도입가의 특정 요율 또는 사용 인원에 비례하여 유지보수료를 지불한다.

금융기관 A사는 In-House IT 방식[3]으로 전산시스템을 운영하고 있으며, 대규모 프로젝트 때 시스템 개발 및 구축 때 외부인원과 함께 개발하는 경우를 제외하고, 거의 모든 유지보수나 추가 기능 개발 및 테스트를 내부 직원이 수행한다. 이러한 기관의 특성으로 개발한 프로그램이 테스트를 거치면서 문제점이 있는 경우 확인하여 즉각적으로 수정된다. 만약 상용 제품의 오류가 발견되는 경우 유지보수를 요청 할 수 있으나, 개발 시한에 맞추지 못하는 경우 이러한 오류를 우회하는 형태로 수정하는 경우가 많다. 그래서 UI 관련 상용 제품의 유지보수는 솔루션 내부 영향이 적은 패치 정도에 그치고 있으며, 경험적으로 실제 수정을 요청하는 경우는 적은 편이다.

만약 테스트 때 발견되지 않은 제품의 오류나 비정상적인 동작이 발생한다고 하더라도 금융시장 시스템 특성상 즉각적인 대응이 필요하기 때문에 상용 소프트웨어 개발사의 기술지원 인력이 도착하기만 기다리는 것이 아니라 내부 담당자들이 자체적인 다른 방법을 강구하여 장애에 대응하기 때문에 실제 UI와 관련된 유지보수 서비스를 자주 이용하지 않고 있다.

· **비유연한 확장성** : 상용 프레임워크를 구매하여 사용하면 강력한 기능을 손쉽게 사용할 수 있다는 장점이 있지만, 반대로 고객사에 맞는 특정 기능을 추가하거나 수정하는 등 커스터마이징(Customizing)이 필요한 경우 상용 프레임워크에 반영하기 어렵다는 단점이 있다.

고객사가 상용 프레임워크 제조사에 범용적인 기능이 아닌 특정 고객에 국한되는 기능을 요청하게 되면 제조사에서는 특정 고객을 위한 소프트웨어를 개발한 후 별도의 버전관리가 필요하기 때문에 관리의

3) 시스템 개발 또는 시스템 유지보수를 사내 IT 담당자가 처리하는 방식

문제 등으로 어려움을 표하는 경우가 많다. 또한, 경험적으로 제조사에서 요구사항에 대한 개발 또는 반영의 의지가 있다고 하더라도 긴급한 오류 수정 건이 아닌 경우에는 실제 프레임워크에 반영되는데 오랜 시간을 소요하는 것이 일반적이다.

· **모바일 환경 미지원** : 현행 기간계 시스템은 PC환경에서 윈도우즈 운영체제에 종속적인 환경을 대상으로 개발되어 스마트폰과 같은 모바일 장치에서는 이용할 수 없다. 또한, 윈도우즈 환경이라도 화면의 크기가 작고 해상도가 낮은 장치에 최적화된 UI를 제공하지 못하고 있다.

[그림 1-1]에서 알 수 있는 것처럼 이미 모바일 환경의 점유율이 일반 개인용 컴퓨터와 비슷하거나 상회하고 있으며 2019년 말부터 창궐한 코로나 19 바이러스로 인한 비대면 업무환경 구축이 속도를 내고 있다. 이는 미래의 사무환경이 PC 환경을 벗어나 다양한 모바일 장치로 확장될 것으로 예측할 수 있고, 미래에 다양한 장치 환경을 지원하지 못한다면 사용자에게 불편을 초래하게 될 것이다.

2. 연구내용

이전 항에서 언급한 네 가지 문제점을 해결하기 위해서는 현재의 기술구조에서 완전히 벗어나는 새로운 기술구조 설계가 필요하다. 본 연구에서는 앞선 절에서 정의한 네 가지 문제점을 해결하면서도 아래 [표 1-2]에 정의된 4가지 조건을 만족하는 반응형 웹 기반의 UI 프레임워크를 설계하려고 한다.

[표 1-1] 본 연구 결과가 만족해야할 조건 네 가지 조건

Cross Platform	특정 Platform(OS)에 비종속적
Cross Browser	다양한 인터넷 브라우저에 대응
Responsive Web Design	반응형 디자인으로 하나의 프로그램으로 크기와 해상도가 다른 여러 장치에서 동일한 기능으로 동작
Open Source	오픈 소스만을 사용하여 비용 절감

· **비종속적인 운영체제** : 웹 기반의 프로그램은 웹브라우저만 구동되는 환경이라면 어디서나 사용할 수 있다. 즉, 스마트폰이나 태블릿과 같은 모바일 장치는 물론 TV나 셋톱박스 등 브라우저가 구동 가능한 환경의 장치는 모두 이용 가능한 환경이다. 본 연구는 표준을 준수하는 웹 기반의 UI를 사용하는 프레임워크를 설계하여 운영체제 종속성이 없고(Cross Platform) 다양한 인터넷 브라우저에서 구동할 수 있는(Cross Browser) 시스템을 구현하고자 한다.

· **유지보수료 절약** : 앞서 언급한 바와 같이 현행 기간계 시스템은 사용자가 시스템을 이용하기 위한 UI를 제공하기 위해 여러 가지 상용 소프트웨어를 사용하고 있다. 이번 연구에서는 향후 시스템 도입 비용 및 유지보수료 절감을 위하여 기능 구현을 위한 라이브러리는 오픈 소스 소프트웨어를 사용하여 구현하는 것을 목표로 한다.

단, 오픈 소스는 라이선스에 따라 제약사항이 상이하므로 향후 라이선스 문제가 발생하지 않는 오픈 소스 소프트웨어만을 사용한다.

· **유연한 확장성** : 오픈 소스의 라이선스마다 차이는 있을 수 있으나 대다수의 오픈 소스 소프트웨어는 소스코드가 공개되어있고 수정이 가능하다. 즉, 필요한 경우 오픈 소스 라이브러리나 컴포넌트를 직접 수정하거나 기능을 추가할 수 있어 유연한 확장성을 갖는다. 이는 시스템 사용자 등 개발이 필요한 다양한 요구에 상용 소프트웨어 대비 신속한 대응이 가능하다는 장점이 있다.

금융기관 A사는 In-House IT 방식으로 시스템을 유지보수 하고 있어 추가적인 기능이 필요한 경우 내부 직원이 오픈 소스에 필요한 기능을 추가하거나 수정 할 수 있기 때문에 상용 소프트웨어를 사용하는 것 대비 시스템 확장에 더 유연하게 대처할 수 있을 것으로 판단된다.

· **다양한 장치 지원** : 기존의 금융기관 A사의 업무시스템 클라이언트가 특정 운영체제만을 지원하여 다양한 장치를 지원할 수 없고 다양한 해상도를 가진 기기에 최적화된 UI를 제공할 수 없었던 문제점을 가지고 있다.

본 연구에서는 반응형 디자인을 이용하여 위 문제를 해결하고자 한다. 반응형 디자인의 업무 화면은 기존 방식의 업무 화면보다 구현 복잡도가 높을 수 있으나 하나의 소스코드로 다양한 장치를 지원할 수 있으므로 사용자의 사용성이나 유지보수 측면에서 훨씬 더 유리하다.

제 3 절 연구보고서 구성

본 연구 보고서의 나머지 부분은 아래와 같은 내용으로, 이를 포함한 총 5장으로 구성되어 있다.

제2장에서는, 웹 기반의 UI 프레임워크가 Cross Platform, Cross Browser, Responsive Web, Open Source 조건을 만족하기 위해 필요한 기술구조의 종류와 특성 등을 분석하고, 이와 관련된 선행 연구를 조사하고 분석한다.

제3장에서는, 현재 사용되고 있는 As-Is 기술구조를 분석하고 본 연구에서 설계하고 구현해야 할 상세한 요구사항을 확정하여 연구의 범위를 정의한다. 또한 요구사항을 Web Based UI로 전환하기 위한 기술구조를 설계하기 위한 오픈 소스 소프트웨어를 탐색 및 선정한다.

제4장에서는, 제3장에서 정의된 요구사항과 기술구조, 사용할 오픈 소스 소프트웨어를 이용하여 구조를 설계한다. 설계된 구조로 UI 프레임워크를 구현하고, 해당 UI 프레임워크를 이용하여 As-Is 시스템에서 사용 중인 5가지 예제 화면을 동일한 기능으로 구현하여 동일하게 동작하는지 검증한다.

제5장에서는, 본 연구의 요약과 결론을 도출하고 연구성과, 기대효과 및 향후 계획에 대하여 기술한다.

제 2 장 연구배경 및 관련연구

제 1 절 연구 배경

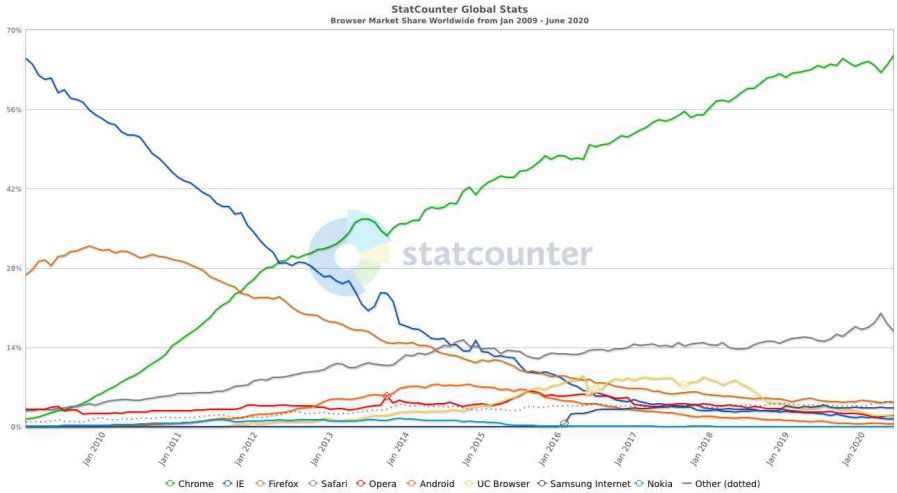
본 연구의 목표는 최종적으로 금융기관 A사에서 현재 운용 중인 기간계 시스템의 클라이언트 UI를 웹 기반의 UI로 전환하여 동일한 기능을 수행 할 수 있는지 확인하고 추가로 앞서 정의한 4가지 요건인 Cross Platform, Cross Browser, Responsive Web Design, Open Source를 만족하도록 설계 및 구현하여 가능한 많은 장치에서 동작시키는 것이다.

이를 위하여 본 절에서는 위 4가지 개념에 대하여 살펴보고, 각각의 기능에 대하여 분석한다. 그리고 그 결과들을 연구 목표에 부합하는 요건을 만족하도록 환경 구축, 설계, 구현에 반영한다.

1. Cross Platform : 크로스 플랫폼은 멀티 플랫폼이라고도 불리며, 두 개 이상의 장치나 운영체제에서 동작할 수 있다는 뜻을 가진다. 많은 장치와 운영체제에서 구동될 수 있는 프로그램일수록 범용성이 증가하지만, 이러한 프로그램들은 통상적으로 가상머신 등 플랫폼 위에서 동작하게 되므로 성능은 감소하는 경향이 있다. 그러므로 프로그램이 사용될 환경과 특성에 따라 구동될 플랫폼이나 언어를 선택해야 한다.

이번 연구는 UI 측면의 개선이며 정적인 화면을 대상으로 구현 할 것이므로 고성능이 필요하지 않다. 그러므로 웹브라우저만 구동되면 실행 가능한 환경인 웹 환경을 기반으로 UI를 구성하여 크로스 플랫폼을 만족하는 시스템을 설계한다.

2. Cross Browser : 웹 환경에서 UI를 렌더링(Rendering)하고 사용자의 행동을 제어하는 웹브라우저는 수많은 제품이 여러 장치에서 사용되고 있으며 [그림 2-1]과 같은 점유율을 보인다 [3].



[그림 2-1] 브라우저별 점유율 추이 (2009.01-2020.06)
(이 그림은 statcounter의 Creative Common License에 의해 제공됨)

웹브라우저는 해석기(Interpreter)의 기능을 포함하고 있으며, 해석기 또한 프로그램이기 때문에 같은 소스코드(HTML, Javascript 등)라도 렌더링 또는 실행 결과가 상이 할 수 있고 특정 소스코드의 경우 일부 브라우저는 지원되지 않는 등 호환성 문제도 발생 할 수 있다. 이러한 호환성 문제를 해결하기 위해서는 웹 표준을 준수하는 코드를 작성하여 브라우저의 오동작이나 오해석을 최대한 줄이는 것이 중요하다.

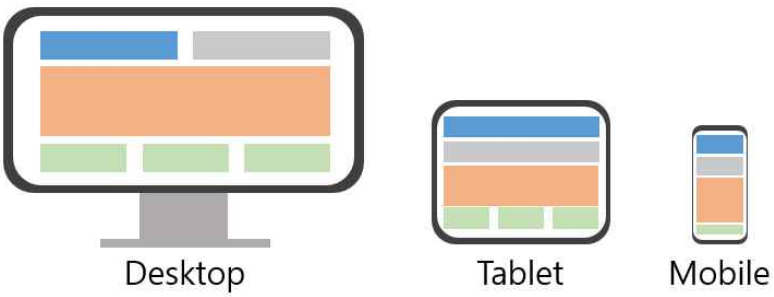
웹 표준을 준수하는 코드라도 Javascript의 경우 웹브라우저에 대한 호환성에 대한 문제를 완벽히 해결할 수 없어 그동안 다양한 오픈 소스 프로젝트가 진행되었다. D3, jQuery, UnderscoreJS, ModernizrJS 등의 라이브러리나 Bootstrap, Foundation 등의 UI 프레임워크들이 이러한 문제를 해결하기 위한 대표적인 프로젝트이다.

위에 나열한 오픈 소스 라이브러리를 사용하게 되면 호환성 문제가 발생할 수 있는 부분에 대하여 라이브러리가 브라우저별로 동작을

최대한 보장(Cross Browser)하게 된다. 그리고 개발자는 구현하고자 하는 논리에만 집중 할 수 있어 개발효율이 향상된다. 본 연구에서 적절한 라이브러리나 프레임워크를 도입하여 웹브라우저별 호환성에 대한 문제를 최소화하면서도 개발 효율을 최대화하고자 한다.

3. Responsive Web Design : 과거 비즈니스 서비스를 위해 제공하는 UI를 개발하는 경우, 주요 목표가 되는 장치들에 맞게끔 UI를 기획하고 구현하여 개발하는 적응형 웹(Adaptive Web)이 일반적이었다.

최근 여러 제조사가 스마트폰과 태블릿, PC, TV 등 다양한 크기의 모바일, 스마트 기기를 쏟아내고 있다. 이러한 다양한 장치를 위한 프로그램을 각각 개발하는 경우 기능의 추가, 수정 등 유지보수 시 동일한 기능을 수행하는 중복 코드들로 인하여 관리 측면에서 많은 문제점을 수반한다. 또한 지원해야 할 장치 화면의 크기나 해상도가 너무 다양하기 때문에 모든 장치에 맞는 UI를 제공하는 것은 개발 및 관리 효율성 면에서 사실상 불가능하다.



[그림 2-2] 반응형 웹의 장치별 레이아웃 변화

다양한 해상도를 갖는 장치를 지원하도록 하기 위하여 반응형 웹 디자인(Responsive Web Design)을 이용 할 수 있다. 반응형 웹은 하나의 소스코드로 장치의 종류에 따라 웹페이지의 레이아웃이 [그림 2-2]와 같이 자동으로 재조정될 수 있도록 작성한 웹페이지다. 본 연구에서는 다양한 장치에 적절하게 대응하여 범용성과 사용자 편의성을 극대화 할 수 있도록 반응형 웹 디자인으로 화면을 설계한다.

4. Open Source : 오픈 소스 소프트웨어는 의미상 소스코드를 공개한 소프트웨어를 뜻한다. 오픈 소스 소프트웨어가 모두 무료로 사용할 수 있는 프리웨어(Free Software)는 아니지만, 대부분의 오픈 소스 소프트웨어는 무료로 사용할 수 있다. 하지만 오픈 소스 소프트웨어가 따르는 라이선스의 종류에 따라 수정이 불가능하거나 오픈 소스를 사용하여 구현한 저작물의 소스코드를 반드시 공개해야 하는 등의 여러 가지 제약사항이 발생 할 수 있다.

널리 사용되는 오픈 소스 소프트웨어 중에서 가장 유의해야 하는 라이선스는 GPL(General Public License) 라이선스일 것이다. GPL 라이선스를 따르는 오픈 소스 소프트웨어는 해당 소프트웨어를 사용한 저작물에 대하여 소스 코드를 반드시 공개해야 한다는 제약사항이 존재하며, 또한 개발된 소프트웨어에 대하여 동일한 GPL 라이선스를 취해야 하는 등 여러 가지 제약이 따른다.

이번 연구에서 설계하고 구현할 대상 프로그램은 추후 실제 업무에 사용할 것을 가정하였기 때문에 GPL 라이선스 또는 그보다 제약사항이 강한 오픈 소스 소프트웨어는 금융기관 A사가 준수해야 할 법률 및 규정으로 인하여 사용할 수 없어 배제한다. 비용 절감의 목적도 달성하기 위해 당연하게도 유료 오픈 소스 소프트웨어도 배제한다.

제 2 절 관련 연구

· **금융권의 오픈 소스 도입** : 금융권에서는 상용 소프트웨어가 안정성이 뛰어나다는 선입견이 있어 상용소프트웨어를 사용하여 시스템 구축 하는 것이 일반적이었다. 디지털데일리 이상일 기자의 기사에서 한국거래소(KRX)의 엑스추어 플러스 시스템을 오픈 소스인 리눅스를 기반으로 구축한 사례를 소개하면서 하나대투증권, K뱅크, 카카오뱅크 등 여러 금융기관에서도 오픈 소스를 도입하려는 움직임에 대하여 언급하였다 [4]. 또한, 2019년 10월 27일 전자신문 함지현 기자의 기사에서 신한은행과 씨티은행의 리눅스 도입을 언급하는 등 금융권에서 오픈 소스의 도입이 가속화 되고 있음을 언급하였다 [5].

이처럼 최근 금융권에서도 오픈 소스 소프트웨어에 대한 인식이 변하고 있으며 오픈 소스 소프트웨어를 활용한 시스템 구축이 활발하게 검토되고 있다.

· **웹 기반 금융업무 증가** : 한국은행 금융정보화추진협의회 전자금융총람에서 2013년 말 기준 27개 증권사가 범용 웹브라우저를 통한 트레이딩 시스템을 제공하고 있으며 등록 고객수 1,809만명, 연중 실이용고객수 279만명, 연중 이용건수는 9억 건을 기록한 것을 알 수 있다 [6]. 또한, 2019년 11월 25일 아이뉴스24 최은정 기자의 기사에서 KB국민카드 차세대 시스템에 금융 계정계 전체를 웹 표준 단말로 적용한 사례를 소개하였다 [7].

2017년 12월 12일 지디넷코리아 김우용 기자는 기사에서 금융 단말도 HTML5 기반 웹 표준으로 모두 가능해졌으며 멀티OS, 멀티브라우저, 멀티 디바이스에 대응하는 프로그램을 사용해야 사용자를 만족시킬 수 있다는 인스웨이브시스템즈 이상희 이사의 발언을 언급하였다 [8].

이를 통해 웹 환경으로 금융 서비스 제공하는 것이 다양한 환경의 사용자를 만족시키는 데 도움이 된다는 것을 알 수 있다.

· **오픈 소스 라이선스** : 오픈 소스 소프트웨어 통합지원센터는 오픈 소스 라이선스를 FOSS 라이선스 유무, 특허, 반환, 공유, 상용 등에 따라 분류하였다 [9]. 또한, 한국저작권위원회는 복제, 배포, 수정의 권한허용, 배포 시 라이선스 사본첨부, 저작권고지, 수정 내용 고지 등 라이선스의 주요 제약사항별로 비교하였다 [10].

오픈 소스 라이선스 중 MIT 라이선스는 저작권 표시와 허가조항을 소프트웨어의 모든 복제본 혹은 중요한 부분에 포함해야한다는 내용을 제외하고는 강제적인 제약사항이 없어 사용에 자유롭다.

· **반응형 웹 디자인** : World Wide WebConsortium (W3C) 표준위원회는 2008년 7월 Mobile Web Best Practices 1.0 Basic Guidelines에서 웹 애플리케이션 사용자에게 장치에 관계없이 동일한 정보와 서비스를 제공하기 위한 "One Web" 원칙을 권장하였다 [11].

Ethan Marcotte는 2010년 5월 반응형 웹 디자인(Responsive Web Design)의 용어와 개념을 제시하였다 [12]. 반응형 웹 디자인은 W3C의 CSS3에서 미디어 유형(Media type)과 미디어 기능(Media feature)로 구성되는 미디어 쿼리(Media query)를 사용하여 뷰포트의 특정 해상도 임계 값에 따라서 페이지에 구성되는 이미지나 구성 요소의 레이아웃을 변경하는 방법을 제시하였다.

Mark Otto와 Jacob Thornton은 블루프린트(Blueprint)라는 프론트엔드(Front-end) 프레임워크(Framework)를 개발하였고 2011년 8월 부트스트랩(Bootstrap)으로 이름을 바꾸어 오픈 소스로 공개하였다 [13]. 부트스트랩은 12단의 그리드 시스템을 기반으로 손쉽게 반응형 웹 사이트나 웹 프로그램을 만드는 방법을 제시하였다.

Befekadu Mezgebu Temere는 반응형 웹 디자인 프레임워크인 부트스트랩과 파운데이션(Foundation)를 비교하여 부트스트랩이 반응형 프레임워크로 더 성능이 좋음을 주장하였다 [14].

· **JavaScript** : Netscape와 Sun Microsystems는 기본적인 하이퍼텍스트 문서를 조금 더 동적으로 표시하기 위하여 JavaScript를 개발하였다 [15]. JavaScript를 통해 웹 페이지를 조작하여 동적으로 변하는 페이지를 만들 수 있게 되면서 널리 사용되었고, 다양한 기능이 추가되며 현재에 이르렀다.

· **Ajax** : Jesse James Garrett은 2005년 2월 A New Approach to Web Applications에서 비동기 통신 방식인 Ajax라는 개념을 소개하였다 [16]. Ajax는 브라우저와 웹서버가 통신할 때 웹페이지가 모두 갱신되어 불필요한 대역폭이 낭비되는 기존의 단점을 극복하기 위해, 화면에 필요한 데이터만 웹서버에 요청하고 응답 데이터를 브라우저에서 처리하여 화면을 변경하는 방식을 제시하였다.

· **jQuery** : John Resig은 2006년 뉴욕시 바캠프(Barcamp NYC)에서 제이쿼리(jQuery)를 소개하였다 [17]. 제이쿼리는 기존에 널리 사용되는 자바스크립트(JavaScript)보다 직관적으로 웹페이지를 조작할 수 있도록 단순한 문법을 제시하였다. 또한, 여러 종류의 브라우저에서 자바스크립트가 가지고 있던 호환성 문제를 해결할 수 있도록 고안되었다.

Jingjing Li와 Chunlin Peng은 jQuery-based Ajax General Interactive Architecture에서 제이쿼리는 브라우저 호환성이 매우 뛰어나다는 것을 언급하였다 [18].

· **jsGrid** : Artem Tabalin은 2014년 12월 제이쿼리를 기반으로 표 형식의 데이터를 손쉽게 다룰 수 있는 제이에스그리드(jsGrid)를 개발하여 공개하였다 [19]. 제이에스그리드는 데이터의 입력, 수정, 삭제뿐만 아니라 필터링, 페이징, 정렬 등의 기능을 함께 제공하여 표 형태의 데이터를 손쉽게 다룰 수 있도록 제시하였다.

· **패스워드 관리 강화** : Khowfa, W와 Silasai, O는 패스워드를 더욱 안전하게 관리하기 위하여 패스워드에 해시(Hash)를 사용하기 전에 임의문자(Salt)를 추가하여 해시하는 경우 패스워드가 더 안전함을 주장하였다 [20]. 패스워드를 해시하기 전에 임의문자를 추가하면 공격자가 암호를 해독하는데 더 많은 시간을 소비하기 때문이다.

제 3 장 As-Is 분석 및 To-Be 기획

제 1 절 요구사항 분석 및 확정

본 연구의 달성 목표(To-Be)를 정의하기 위하여 현행 금융기관 A사의 클라이언트 UI(As-Is)를 분석하여 요구사항을 도출한다. 그리고 도출된 요구사항을 달성하기 위한 설계와 이를 기반으로 한 구현을 통해 최종적으로 연구의 목표가 달성되었는지 검증한다. 요구사항은 '업무'의 영역과 '공통'의 영역으로 분리하여 도출한다.

· **업무 영역** : 업무 영역은 시스템 사용자가 업무를 위한 화면에 접근하여 데이터의 입력, 조회, 수정, 삭제 등 사용자의 행동을 제어하여 업무와 관련한 제반 사항을 처리하는 영역이다. 해당 영역에서는 사용자의 편의를 위하여 텍스트, 라디오, 체크박스, 리스트(그리드), 트리, 차트 등의 컨트롤들을 업무에 맞게 조합하여 사용한다.

· **공통 영역** : 공통 영역은 사용자가 실제 업무를 처리하기 위한 업무 화면 외에 로그인/로그아웃, 접근 권한, 메뉴, 서버와의 통신 기능, 디자인 규칙 등을 포함한다. 즉, 업무 영역에서 사용자가 업무 수행을 위해 필요한 기능이나 처리를 제외한 모든 나머지 클라이언트 기능을 공통의 영역으로 간주한다.

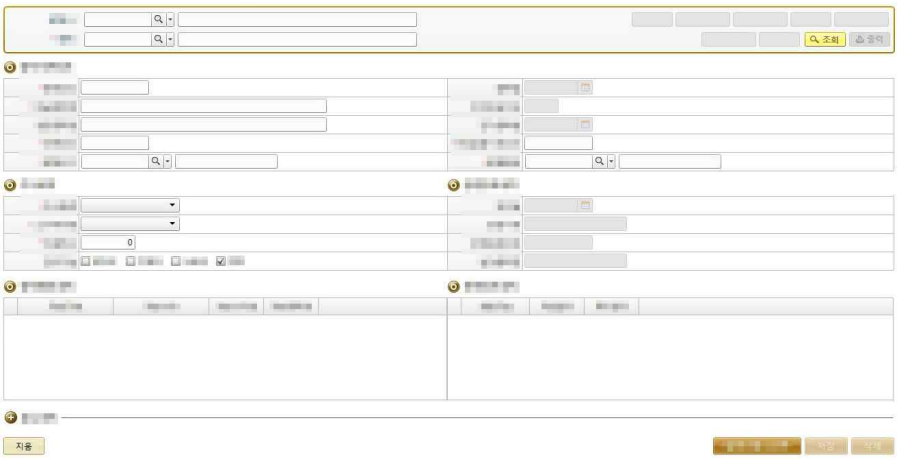
다음의 항에서 위 두 가지 영역 관점으로 금융기관 A사 기간계 클라이언트 UI를 분석하여 요구사항을 도출할 것이다. 단, 본 연구가 한정된 시간에 진행되어야 하므로 이번 연구에서는 자주 활용되는 형태의 업무 화면 5본을 예제로 선정하고 동일한 기능을 수행할 수 있도록 설계 및 구현하는 것을 목표로 요구사항을 도출하였다.

(1) 업무 화면 분석 및 예제 화면 선정

업무 화면은 주로 사용자가 데이터베이스 내에 있는 데이터 조작을 위하여 화면을 사용하게 되며 주로 CRUD라고 불리는 입력(Create), 조회(Read), 수정(Update), 삭제(Delete)의 4가지 처리를 한다.

기간계 시스템 화면에서 사용자가 데이터를 직관적이고 효율적으로 관리할 수 있도록 데이터의 속성에 맞는 컨트롤을 사용한다. 컨트롤은 UI Framework에서 지원하는 컨트롤이거나 직접 개발한 컨트롤이다. 웹 기반의 UI Framework로 전환하기 위해서는 현행 시스템에서 제공하는 컨트롤들을 동일한 기능으로 제공할 수 있어야 한다. 본 연구에서는 활용 빈도와 형태에 따라 5개의 예제 업무 화면을 선정하여 구현한다.

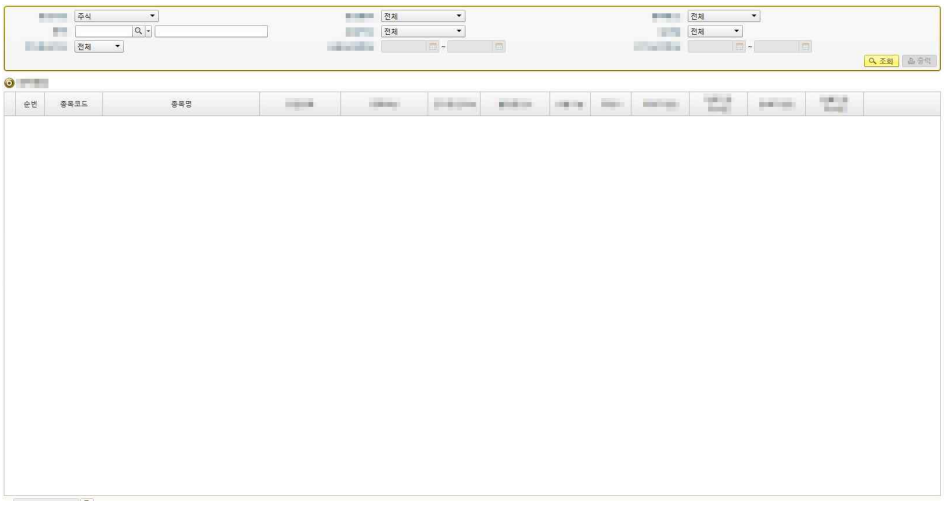
· **일반 화면 (화면 1)**



[그림 3-1] 일반 화면

위 [그림 3-1]과 같은 일반 화면은 데이터베이스에서 1개의 행이 갖는 속성들을 관리하는 화면이다. 텍스트 컨트롤, 콤보박스 컨트롤, 체크박스 컨트롤, 라디오 컨트롤 등 가장 자주 사용되는 일반 컨트롤들로 구성되어있다. 다양한 일반 컨트롤이 정상적으로 동작하는지 구현하고 검증하기 위하여 본 화면을 구현 대상 화면으로 선정하였다.

· 그리드 화면 (화면 2)

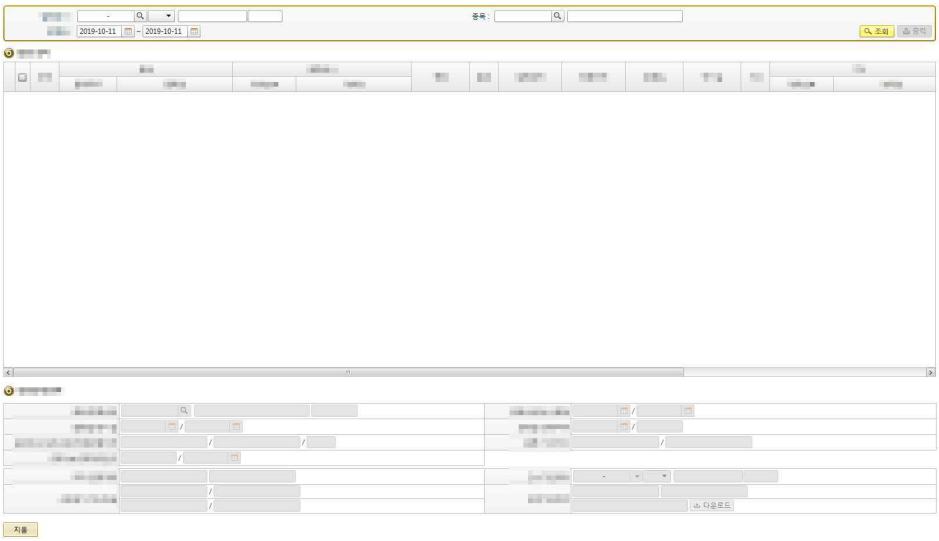


[그림 3-2] 그리드 화면

그리드 컨트롤은 [그림 3-2]와 같이 다수의 행과 열의 데이터를 조회하고 조작하기 위한 컨트롤이다. 그리드의 개념은 표와 비슷하지만, 사용자가 특정 열을 기준으로 정렬 하거나, 한 번에 표시되는 행의 개수를 제한하는 페이징(Paging), 그리드 컨트롤에서 직접 데이터를 입력하거나 수정하는 등 직관적이고 다양한 기능을 제공하기 때문에 단순히 표의 형태로 데이터를 표시하는 것보다 훨씬 효율적이다.

기간계 시스템과 같은 금융 환경에서는 사용자가 업무 화면에서 대량의 데이터를 다루는 경우가 많기 때문에 그리드 컨트롤을 사용하는 화면들의 사용 빈도가 매우 높고 중요하다. 그러므로 그리드를 사용하는 위 화면을 구현 대상 화면으로 선정하였다.

· 복합 화면 (화면 4)



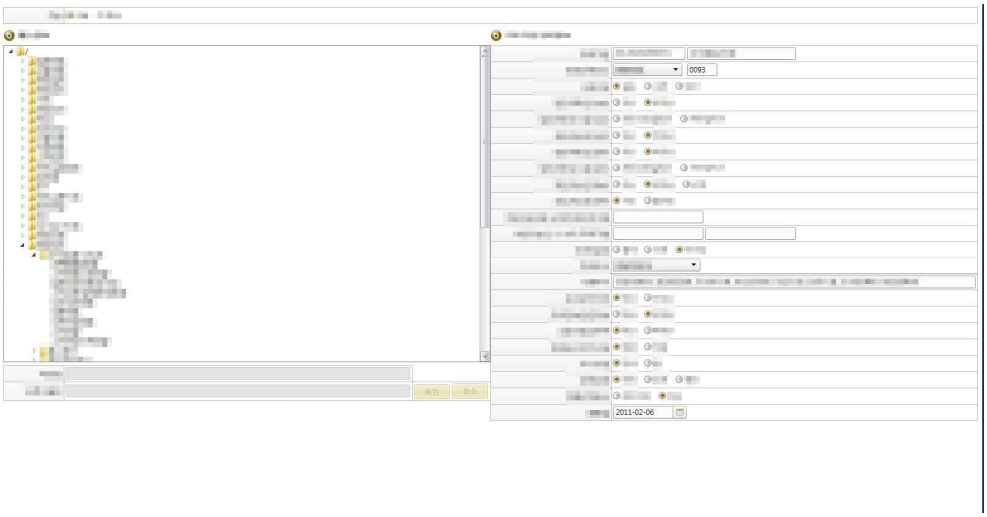
[그림 3-3] 복합 화면

복합 화면은 일반적인 컨트롤들이 그리드와 연동되는 구조를 갖는 화면이다. [그림 3-3]의 경우 상단 그리드에 조회되는 데이터와 하단 일반 컨트롤 영역에서 사용되는 데이터의 원천은 동일하지만, 복합적으로 UI 구성한 화면이다. 이처럼 복합적인 화면을 구성하게 되는 이유는 그리드 컨트롤에서 구현하기 어려운 UI의 제어가 필요한 경우가 발생하기 때문이다.

예를 들어 사용자가 선택하는 값에 따라서 입력받을 데이터의 형태나 개수 또는 종류가 달라지는 경우에 그리드 컨트롤만으로는 구현하기 어렵거나 비효율적이다. 즉, 그리드의 한 행에 대하여 상세한 데이터와 UI의 제어가 필요한 경우 그리드와 일반 컨트롤을 함께 사용하며 연동시키는 것으로 문제를 해결할 수 있다.

이처럼 일반컨트롤과 그리드의 연동 및 제어도 기간계 시스템의 업무화면에서 매우 보편적으로 사용되고 있기 때문에 이번 연구의 구현 대상 화면으로 선정하였다.

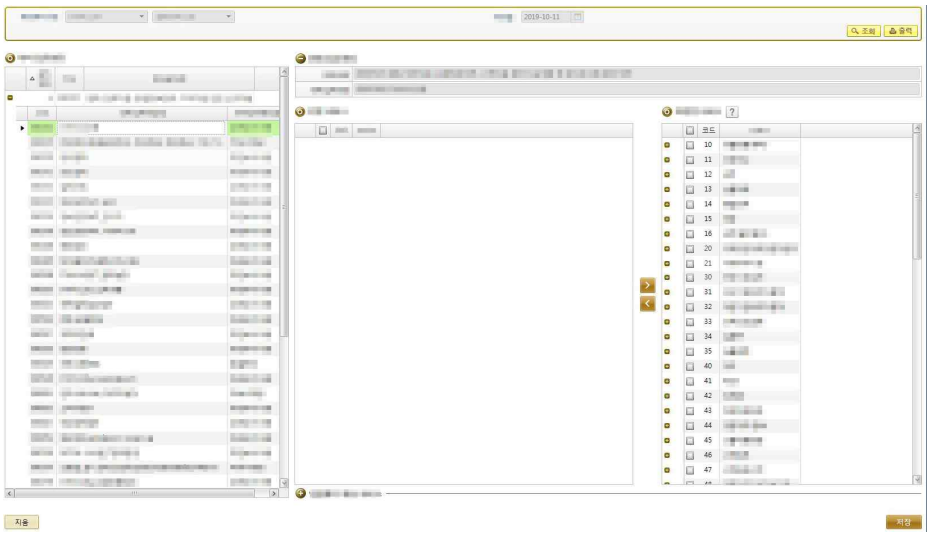
· 트리 화면 (화면 4)



[그림 3-4] 트리 화면

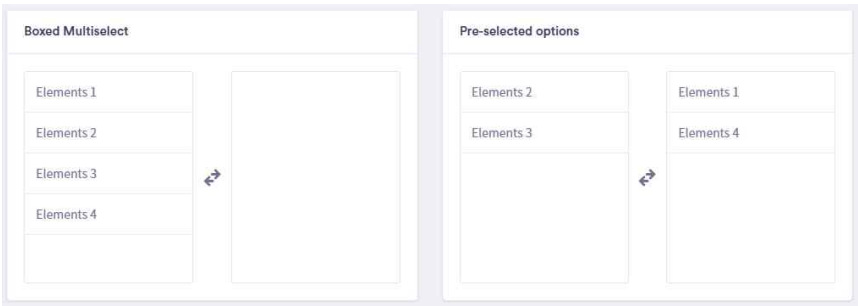
트리 컨트롤은 [그림 3-4]의 좌측과 같이 부모/자식 관계의 여러 노드 데이터를 직관적으로 표현하고 효율적으로 관리하는 데 주로 사용한다. 기간계 시스템에서 트리 컨트롤을 여러 화면에서 빈번하게 사용하고 있지는 않지만, 트리 컨트롤을 사용하고 있는 본 화면 자체의 사용 빈도가 높고, 해당 데이터를 공통 영역에서 사용하고 있으므로 구현 대상 화면으로 선정하였다.

· 리스트 화면 (화면 5)



[그림 3-5] 1:N 데이터 관리 화면

한 행의 데이터가 여러 개의 하위 데이터를 가지는 1:N 관계의 데이터도 업무적으로 자주 발생한다. [그림 3-5]는 좌측 그리드의 한 행과 우측 그리드의 다수의 행의 관계를 설정하는 화면으로 현행 화면에서는 여러 개의 그리드 컨트롤을 이용하고 있다.



[그림 3-6] 멀티셀렉트 리스트 컨트롤의 예시

그리드로 구현한 [그림 3-5]의 화면을 [그림 3-6]의 멀티셀렉트 리스트 컨트롤을 사용하는 것으로 개선하면 적은 수의 컨트롤로 종속 관계를 설정 및 해제할 수 있어 효율적이다. 그러므로 본 화면은 멀티셀렉트 리스트 컨트롤을 사용하는 방법으로 구현한다.

(2) 공통 영역 분석 및 요구사항 확정

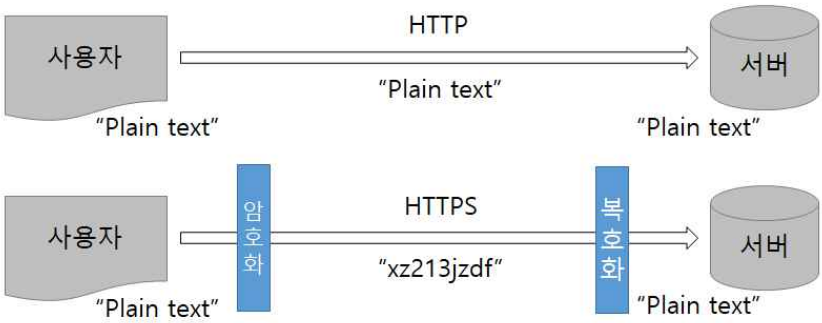
금융기관 A사 기간계 시스템 UI 클라이언트에서 공통 영역은 업무 화면에서 업무 처리를 위한 것을 제외한 거의 모든 부분이기 때문에 상당히 광범위하다. 본 연구에서는 앞서 선정한 5가지 화면을 구현하는데 필요한 기능을 공통 기능의 범위로 정하여 다음과 같이 검토하고 요구사항을 도출하였다.

· **로그인/로그아웃** : 기간계 시스템에서 사용자가 로그인하기 위해서는 여러 가지 행정적 신청과 관련 서류 제출을 통해 최종적인 회원가입 신청이 승인되어야만 시스템에 로그인하여 접근할 수 있다. 본 연구에서는 로그인과 로그아웃 기능을 구현하되, 회원 관리는 대상 범위에서 제외하고 로그인에 필요한 정보는 임의로 작성한다.

· **UI 디자인** : UI 디자인은 사용자가 화면을 통해 업무를 처리하게 되는 경우 당연하게 필요하다. 디자인은 단순히 화면을 미려하게 표시하는 것뿐만 아니라, 화면에 배치되는 컨트롤들도 표준화하는 것도 포함한다. 즉, 여러 업무 화면을 각기 다른 개발자가 구현하더라도 시각적인 면에서 통일감을 줄 수 있도록 해야 한다. 본 연구에서도 각 화면이 표준화된 컨트롤을 사용할 수 있도록 디자인 템플릿을 사용하여 가이드를 마련한다.

· **접근 제어** : 현재 기간계 시스템은 사용자가 내부직원인지, 외부사용자인지에 따라서 메뉴의 구성이나 화면의 레이아웃이 달라지는 등 사용자의 접근을 제어하고 있다. 이러한 접근 제어를 하려면 사용자 정보와 접근 제어와 관련된 정보를 관리해야 하는데, 본 연구의 범위에서 사용자의 관리는 법률 및 규정에 의해 구현 대상으로 포함 할 수 없다. 다만 실제 업무에 적용할 것을 가정하여 앞서 업무 영역에서 구현 대상으로 선정한 접근 제어와 관련된 화면을 구현하여 접근 제어의 기반을 마련한다.

· **구간통신암호화** : 구간통신 암호화는 사용자가 프로그램을 이용해 서버와 통신 할 때 송수신하는 데이터를 [그림 3-7]과 같이 암호화하는 것을 말한다. 최근 로그인 등 회원 관리를 필요로 하는 웹 사이트에서 구간통신 암호화는 권장을 넘어 필수적인 요소로 자리 잡고 있다.

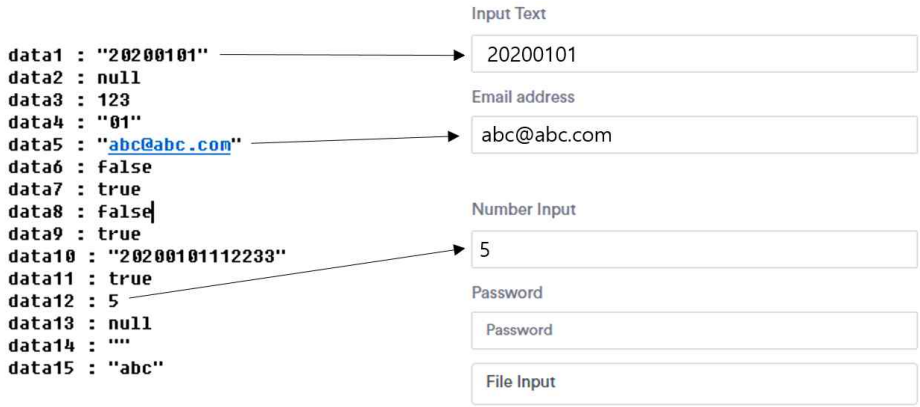


[그림 3-7] http와 https의 차이

현행 기간계 시스템에서는 구간통신 암호화를 위하여 N사의 상용 소프트웨어를 이용하고 있으나, 웹 기반에서 동작하려면 https (HyperText Transfer Protocol over Secure Socket Layer)를 사용하는 것이 적절할 것이다. 본 연구에서는 개발 환경의 제약으로 인해 http를 이용하여 구현할 예정이지만, 추후 구현된 프로그램을 https 환경에서 동작시키면 동일한 결과를 얻을 수 있으므로 구간통신암호화는 https가 적용된 것으로 가정한다.

· **서버통신지원** : 기술구조에 따라 다를 수 있지만 일반적으로 클라이언트가 서버와 통신을 할 때 서버의 주소나 IP(Internet Protocol), 포트(Port) 등 공통적인 정보를 이용하여 서버와 통신한다. 만약 서버와 통신에 필요한 소스 코드를 개발자들이 각 업무에서 개별적으로 구현하게 되면 매우 비효율적일 뿐만 아니라 서버의 변경이나 통신에 필요한 추가적인 프로세스가 필요한 경우 모든 업무 프로그램을 수정해야 하는 문제점이 발생하므로 본 연구에서는 프레임워크에서 서버와 통신에 필요한 래퍼 펑션(Wrapper Function)을 제공한다.

· **데이터 바인딩** : 사용자가 데이터를 관리할 때 서버의 데이터를 화면의 컨트롤에 표시하여 서로 연결된 것처럼 동작하게 하는 경우가 많다. 이때 서버의 데이터를 화면의 컨트롤에 표시될 수 있도록 연결하는 것을 바인딩(Binding)이라고 한다.

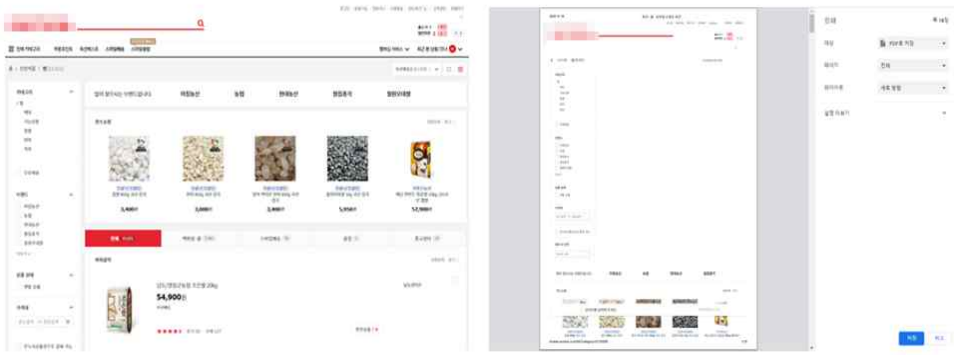


[그림 3-8] 데이터 바인딩의 예시

공통 업무에서 바인딩과 관련한 별도의 지원이 없다면, 업무 화면에서는 서버에서 받은 데이터를 직접 컨트롤과 연결해주어야 한다. 만약 서버에서 조회한 데이터가 많은 항목으로 이루어진 경우 개발자는 모든 컨트롤과 데이터를 연결하는 코드를 개발해야 한다. 이는 개발자가 바인딩 작업을 반복적으로 수행해야만 하기 때문에 비효율적이다.

본 연구에서는 UI 프레임워크에서 서버에서 조회한 데이터를 화면에 배치된 컨트롤 유형에 맞게 자동으로 바인딩하거나, 컨트롤에 표시된 데이터를 서버에 전달하기 위해 수집하는 기능을 설계하고 구현한다. 이를 통해 업무 화면에서는 공통의 바인딩 지원 기능을 이용하여 효율적인 개발과 유지보수를 할 수 있다.

· **인쇄(프린팅)** : 사용자는 필요에 의해 화면에 표시된 데이터를 프린터를 이용하여 종이 출력물로 인쇄를 하는 경우가 종종 발생한다. 웹 환경의 경우 브라우저에 내장된 인쇄 기능을 사용하여 인쇄를 할 수 있지만 일반적으로는 [그림 3-9]와 같이 사용자가 원하지 않는 레이아웃으로 출력되거나 내용이 잘리는 등 여러가지 문제가 발생하기도 한다. 이는 출력하고자 하는 화면이 용지의 크기와 맞지 않거나 웹 페이지에서 인쇄를 위한 설계가 되어있지 않기 때문이다.

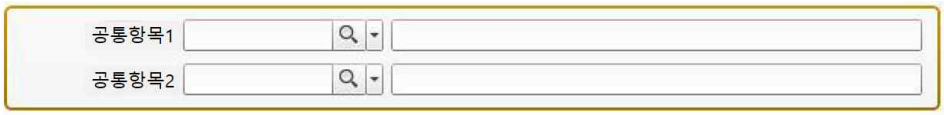


[그림 3-9] 웹사이트 레이아웃(좌)와 인쇄물(우)이 다른 예시

경험적으로 위와 같은 문제를 해결하기 위해서 인쇄할 용지에 맞게 인쇄용 화면이나 레이아웃을 추가로 개발하는 경우가 많다. 그러나 이러한 형태는 업무 화면에 변화가 생기는 경우 업무 화면과 인쇄용 화면을 모두 수정해야 하고, 업무용 화면의 데이터를 인쇄용 화면에 추가로 바인딩 해야 하는 등 중복적인 작업을 유발하는 문제점이 있다.

본 연구에서는 업무 화면을 반응형으로 동작하도록 구현하는 것을 목표로 하였는데, 이는 인쇄 시 사용되는 용지의 크기에도 동일하게 적용된다. 즉, 인쇄할 용지를 하나의 장치 화면으로 인식하여 레이아웃이 변화한다. 그러므로 기본적인 브라우저의 인쇄 기능을 이용하여도 화면의 내용이 잘리거나 레이아웃이 틀어지지 않게끔 인쇄가 가능하다. 추가로, 인쇄를 원하지 않는 컨트롤이나 화면의 요소는 인쇄되지 않게끔 개발자가 선택적으로 정의할 수 있는 방법도 제시한다.

· **공통 검색 컨트롤** : 금융기관 A사 업무에서 특정 정보의 경우 거의 모든 업무에서 공통으로 함께 사용해야하는 공통의 데이터가 존재한다. 공통으로 이용하는 데이터를 각 업무 화면에서 제각기 검색 컨트롤로 구현해서 사용하게 되면 앞서 '서버통신지원' 항목에서도 언급한 것과 같이 중복적인 개발의 문제와 공통 데이터 관리의 변화에 따른 수정 등으로 다양한 문제점이 발생하기 때문에 여러 업무에서 동일하게 사용할 수 있도록 공통의 영역에서 기능을 제공한다.



[그림 3-10] 공통 검색 컨트롤 예시

본 연구에서도 현행과 같이 여러 업무에서 공용으로 사용하는 공통 컨트롤을 구현하고, 업무별 화면에서 쉽게 사용할 수 있도록 제공하는 방법을 제시하여 각 업무 화면에서 현재와 같이 편리하게 공통 검색 컨트롤을 사용할 수 있도록 한다.

제 2 절 UI 프레임워크 기술 구조 선정

앞서 정의한 업무 영역 및 공통 영역의 요구사항을 효율적으로 달성하기 위해 UI 프레임워크의 설계가 필요하다. UI 프레임워크는 업무 화면과 공통 기능을 제공하기 위해 사용할 라이브러리와 컴포넌트 그리고 직접 구현한 기능이 포함된다. 본 절에서는 웹 기반의 UI 프레임워크 기술 구조와 환경을 클라이언트와 서버로 나누어 정의한다.

(1) 클라이언트 환경

· **HTML과 CSS** : 웹 환경에서 UI는 HTML(HyperText Markup Language)과 CSS(Cascading Style Sheets)를 브라우저가 해석하고 렌더링하여 화면에 표시한다. 본 연구에서 웹 표준을 준수하는 HTML과 CSS를 사용하여 화면을 구성한다.

· **JavaScript와 jQuery** : 웹 페이지가 HTML과 CSS로만 구성되는 경우 동적인 웹페이지를 구현하는 데 한계가 있어 JavaScript를 사용한다 [15]. 하지만 연구 배경에서 언급한 것처럼 JavaScript는 브라우저별 호환성 문제가 있기 때문에 개발자들이 여러 브라우저를 지원하는 웹 사이트 개발하는 데에 많은 불편을 겪었다. 호환성 문제를 해결하기 위해 여러 가지 라이브러리가 개발되었는데, 그중 현재까지 가장 많은 시장 점유율을 가지는 라이브러리가 jQuery이다 [17]. 본 연구에서는 jQuery 라이브러리를 사용하여 브라우저별 호환성을 해결하고 개발 편의성을 높인다.

· **Bootstrap** : 반응형 디자인의 웹 페이지를 구현하려면 CSS에 해상도 별로 적용할 레이아웃 스타일을 정의해야 한다. 하지만 CSS도 JavaScript와 마찬가지로 브라우저별 호환성 문제가 존재하기 때문에 직접 CSS를 최적화시키는 것은 비효율적이고 어려운 일이다.

이는 반응형 웹 라이브러리를 사용하여 호환성 문제를 해결할 수 있다. 현재 오픈 소스 소프트웨어로 공개된 반응형 디자인을 지원하는 대표적인 웹 라이브러리 또는 프레임워크로는 부트스트랩(Bootstrap), 파운데이션(Foundation), 아이오닉(Ionic) 등이 있다.

	2019 1 Aug	2019 1 Sep	2019 1 Oct	2019 1 Nov	2019 1 Dec	2020 1 Jan	2020 1 Feb	2020 1 Mar	2020 1 Apr	2020 1 May	2020 1 Jun	2020 1 Jul	2020 1 Aug	2020 19 Aug
jQuery	97.4%	97.5%	97.5%	97.5%	97.3%	97.4%	97.4%	97.4%	97.5%	97.5%	97.5%	97.5%	97.5%	97.5%
Bootstrap	25.4%	25.7%	25.9%	26.0%	26.2%	26.4%	26.7%	26.9%	26.9%	27.0%	27.0%	27.0%	27.0%	27.0%
Modernizr	14.5%	14.4%	14.3%	14.2%	14.1%	14.1%	14.0%	13.9%	13.8%	13.6%	13.5%	13.5%	13.4%	13.3%
Underscore	4.0%	4.0%	4.1%	4.1%	4.2%	4.2%	4.3%	4.4%	4.4%	4.5%	4.5%	4.6%	4.7%	4.7%
Popper	1.9%	2.0%	2.2%	2.3%	2.4%	2.6%	2.8%	2.9%	3.1%	3.2%	3.3%	3.4%	3.6%	3.6%
MooTools	2.7%	2.6%	2.6%	2.6%	2.5%	2.4%	2.3%	2.3%	2.2%	2.2%	2.1%	2.0%	2.0%	2.0%
Moment.js	2.0%	2.0%	1.8%	1.8%	1.8%	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%	1.9%
ASP.NET Ajax	2.0%	2.0%	1.9%	1.9%	1.9%	1.9%	1.9%	1.8%	1.8%	1.8%	1.8%	1.7%	1.7%	1.7%
Backbone	1.1%	1.0%	1.0%	1.0%	1.0%	1.0%	1.0%	0.9%	0.9%	0.9%	0.9%	0.9%	0.9%	0.9%
RequireJS					0.9%	0.9%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%	0.8%

[그림 3-11] JavaScript library market share trends
(이 그림은 W3Techs의 호의에 의해 제공됨)

[그림 3-11]은 W3Techs의 JavaScript 라이브러리의 점유율을 나타내는 자료로, 반응형 웹 라이브러리 중에서는 Bootstrap이 가장 많은 점유율을 차지하고 있는 것을 알 수 있다 [13] [21].

Bootstrap은 반응형 디자인의 페이지 레이아웃을 지원하는 것뿐만 아니라, 화면에서 표시될 입력 폼(Form), 버튼, 탭, 리스트 등 디자인 된 각종 컨트롤들을 포함하고 있다. 이는 앞서 기술한 통일감 있는 화면 개발을 위해 표준화된 UI 디자인을 가져야 한다는 사항을 만족하고 있어 Bootstrap을 UI 프레임워크에 포함하였다.

· **Ajax** : 서버의 데이터에 따라 웹 페이지가 변하거나 컨트롤의 값이 변하게 하려면 크게 3가지 방법이 존재한다. 첫 번째는 페이지를 완전히 새로 요청(refresh)하는 것이고, 두 번째는 Ajax를 사용하는 방법이 있으며 세 번째로 WebSocket을 사용하는 방법이 있다.

첫 번째 방법은 페이지 자체를 다시 요청하여 서버의 데이터에 따라 HTML을 새로 응답하는 방식이며 가장 고전적인 방법이다. 이 방법은

페이지를 새로 표시하기 위해 모든 요소를 갱신하게 되어 브라우저에 깜빡임이 발생하고, 갱신해야 할 데이터를 포함한 모든 HTML 코드를 새로 전송해야하기 때문에 서버에 불필요한 부하를 발생 시킨다.

이 문제를 해결하기 위해 두 번째 방식인 Ajax(Asynchronous JavaScript and XML) 통신의 개념이 등장하였다 [16]. Ajax는 새로운 데이터를 수신하기 위해 페이지 전체를 다시 요청하는 것이 아니라 JavaScript로 데이터만을 수신할 수 있는 주소를 호출한다. 그리고 수신된 데이터를 특정 컨트롤에 바인딩하거나 특정 요소를 JavaScript를 이용하여 변화시키는 방법으로 현재 가장 널리 쓰이는 방법이다. 그러나 클라이언트가 요청하지 않으면 서버에서 클라이언트로 데이터를 보낼 수 없는 http 방식의 한계가 존재하는 것이 한계이다.

세 번째 방법인 WebSocket 방식은 Ajax의 단점을 보완하기 위하여 등장하였다. Ajax의 경우 클라이언트(브라우저) 중심으로 동작하기 때문에 서버가 클라이언트에 이벤트나 데이터를 전달하기 위해서는 클라이언트에서 계속 변화사항이 있는지 체크(Polling)하는 방법밖에 존재하지 않았다. 이는 사용자가 늘어날수록 Polling으로 인한 서버의 부담이 급증하는 문제를 발생시켰으며 이를 해결하기 위해 WebSocket 방식 통신이 고안되었다. 이는 마치 TCP 통신을 하는 것처럼 연결을 유지한 채로 데이터를 송수신 할 수 있어 실시간(Real-Time) 데이터를 다루는 웹 페이지에서 사용되고 있다.

위 세 가지 방법 중 WebSocket 방식이 가장 진보된 방식이긴 하나 구형 브라우저들에 대한 호환성 문제를 가지고 있어 범용성이 떨어지고, 본 연구에서 구현하고자 할 기간계 시스템의 업무 화면들은 대부분 데이터의 실시간 갱신을 요구하는 성격의 화면이 아니기 때문에 WebSocket 대신 가장 보편화한 Ajax 방식을 이용하여 구현한다.

(2) 서버 환경

본 연구는 금융기관 A사의 기간계 시스템 클라이언트를 웹 기반으로 전환하는 것에 대한 연구이지만 클라이언트가 서버와 통신하며 동작하게끔 UI 프레임워크를 제공하는 것을 포함하므로 통신 테스트를 위해 웹 서버와 데이터베이스를 이용해야 한다. 서버 환경은 운용중인 환경과 동일한 환경이 가장 이상적이지만, 보안과 비용 등의 제약으로 동일한 환경을 사용할 수 없어 본 연구에서는 PHP와 MySQL을 이용하였다.

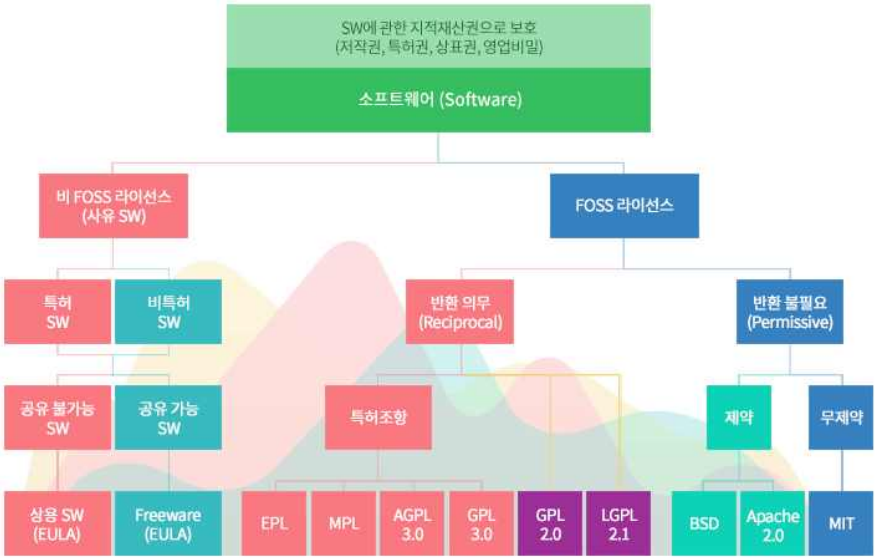
· PHP : 대표적인 서버 사이드 스크립트 언어 중 하나로 분류되는 PHP는 동적인 웹페이지를 생성하기 위해 설계된 언어이다 [22]. 하지만 PHP는 서버 사이드 스크립트이므로 동적으로 페이지를 변경하려면 앞서 페이지를 새로 요청해야 하는 문제점을 가지고 있다. 그러므로 PHP를 이용한 페이지 렌더링은 가능한 배제 하고 서버에서 데이터베이스에 접근하기 위한 목적으로만 사용한다. 본 연구에서 사용한 PHP의 버전은 5.2이다.

· MySQL : 데이터베이스는 다수의 데이터를 여러 사용자가 효율적이고 편리하게 이용할 수 있도록 통합하여 관리되는 데이터의 집합이다. 데이터베이스는 관계형, 객체형, 문서형 등 여러 가지 종류가 존재하는데 그중 가장 널리 사용되는 데이터베이스가 관계형 데이터베이스이다. 관계형 데이터베이스는 오라클(Oracle), PostgreSQL, MYSQL, MSSQL 등 다양한 제품이 있으나 이 중 가장 많이 쓰이는 오픈 소스 기반의 데이터베이스가 MySQL이다.

본 연구에서는 업무 화면에서 입력, 수정, 삭제, 조회에 사용하는 데이터를 MySQL을 이용하여 관리하고 PHP를 이용해 접근하여 사용한다. 본 연구에서 사용한 MySQL의 버전은 5.1이다.

제 3 절 오픈 소스 라이브러리 선정

앞선 절에서 UI 프레임워크와 클라이언트 및 서버에서 사용할 기술 구조를 정의하였다. 본 절에서는 정의된 UI 프레임워크를 기반으로 업무 화면을 구현할 때 사용할 오픈 소스 라이브러리를 탐색하고 확정한다.



[그림 3-12] 오픈 소스 소프트웨어의 라이선스 분류
(이 그림은 한국저작권위원회의 호의에 의해 제공됨)

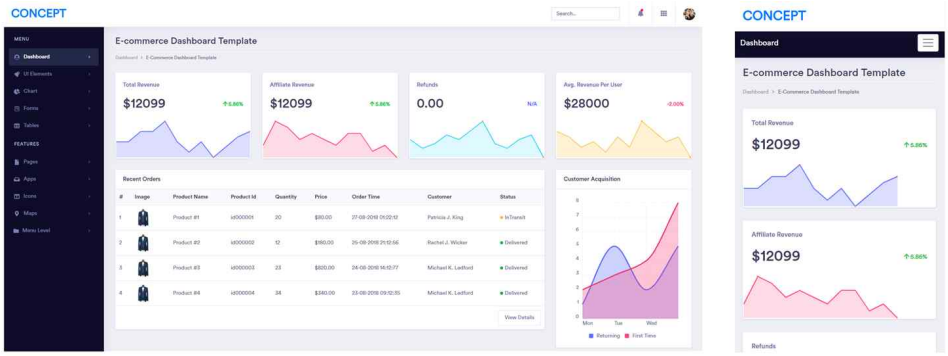
오픈 소스를 탐색하기에 앞서 오픈 소스의 라이선스에 대하여 먼저 검토하여야 한다. 정보통신산업진흥원 공개SW라이선스에서는 오픈 소스 라이선스를 [그림 3-12]와 같이 분류하였으며, 한국저작권위원회에서는 라이선스에 따라 이용, 배포, 수정, 2차 저작물 공개 의무 등의 여러 가지 제약조건에 대하여 비교하였다 [9] [10].

금융기관 A사는 기관 특성상 업무에 사용되는 소스 코드를 공개할 수 없고, 상업적 이용을 하고 있기 때문에 이에 적합한 오픈 소스 소프트웨어를 사용해야만 차후 문제의 소지가 적다. 본 연구에서는 MIT 라이선스나 그보다 제약이 더 낮은 오픈 소스 라이브러리만을 사용하여 오픈 소스 소프트웨어 라이선스로 인한 문제를 피하였다.

(1) 기능별 오픈 소스 라이브러리 선정

본 장의 1절에서 5가지 예제 화면과 예제 화면들을 구현하기 위한 공통 기능을 본 연구의 목표 요구사항으로 정의하였다. 이 요구사항들을 설계하고 구현할 때 사용할 오픈 소스 소프트웨어 또는 라이브러리를 아래와 같이 탐색하고 선정하였다.

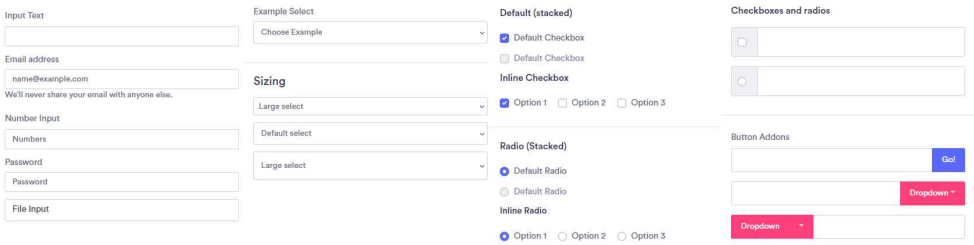
· **Design Template** : 로그인 페이지나 로그인 이후 사용하는 페이지는 공통적인 UI 디자인이 필요하다. 실제 시스템에 적용할 때는 회사의 컨셉이 반영된 UI를 직접 디자인을 해야 할 것이지만 본 연구에서는 미리 개발된 오픈 소스 디자인 템플릿을 사용하였다. 앞서 UI 라이브러리로 Bootstrap을 선정하였기 때문에 Bootstrap을 기반으로 개발된 Colorlib의 'concept' 템플릿을 이용한다 [23].



[그림 3-13] Design Template 'concept'

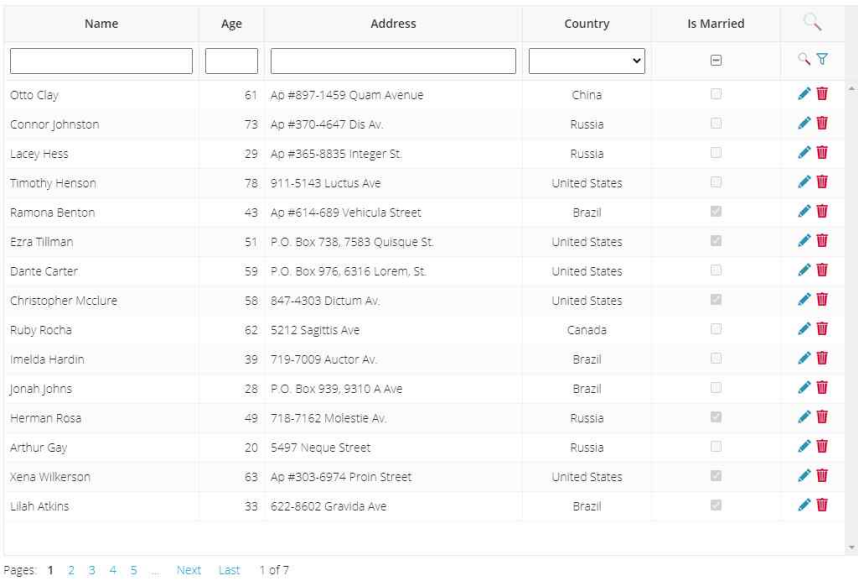
[그림 3-13]과 같이 해당 템플릿은 Bootstrap을 이용하여 반응형 디자인을 만족하도록 설계되었으며 장치의 해상도에 따라 동적으로 변화한다. 또한 Bootstrap의 컨트롤을 이용한 표준화된 컨트롤 디자인을 제공하고 있어 시각적으로도 통일된 느낌의 화면을 구현할 수 있다.

· **일반 컨트롤** : 일반 텍스트 입력, 드롭다운, 체크박스, 라디오 컨트롤 등 기본적인 입력 컨트롤을 사용하게 된다. 이 컨트롤 들은 [그림 3-14]와 같이 디자인 템플릿에서 제공하는 컨트롤들을 사용하여 모든 화면의 UI가 통일감 있도록 구성한다.



[그림 3-14] Design Template에서 제공하는 다양한 컨트롤

· **그리드 컨트롤** : 여러 행의 데이터를 사용자가 효율적으로 관리할 수 있도록 하는 그리드 컨트롤은 jQuery 기반의 jsGrid를 사용하였다. jsGrid는 데이터 필터링, 데이터 관리, 페이징, 정렬 등 다양한 기능을 내장하고 있으며 데이터베이스와의 연동도 지원하여 편리하다.



Name	Age	Address	Country	Is Married
Otto Clay	61	Ap #897-1459 Quam Avenue	China	
Connor Johnston	73	Ap #370-4647 Dis Av.	Russia	
Lacey Hess	29	Ap #365-8835 Integer St.	Russia	
Timothy Henson	78	911-5143 Luctus Ave	United States	
Ramona Benton	43	Ap #614-689 Vehicula Street	Brazil	✓
Ezra Tillman	51	P.O. Box 738, 7583 Quisque St.	United States	✓
Dante Carter	59	P.O. Box 976, 6316 Lorem, St.	United States	
Christopher Mcclure	58	847-4303 Dictum Av.	United States	✓
Ruby Rocha	62	5212 Sagittis Ave	Canada	
Imelda Hardin	39	719-7009 Auctor Av.	Brazil	
Jonah Johns	28	P.O. Box 939, 9310 A Ave	Brazil	
Herman Rosa	49	718-7162 Molestie Av.	Russia	✓
Arthur Gay	20	5497 Neque Street	Russia	
Xena Wilkerson	63	Ap #303-6974 Proin Street	United States	✓
Lilah Atkins	33	622-8602 Gravida Ave	Brazil	✓

[그림 3-15] jsGrid 컨트롤 예제

그리드 컨트롤은 사용자의 편의성도 중요하지만, 가장 중요한 것은 대량의 데이터를 처리할 때 가볍고 빨라야 한다. 이러한 조건을 만족하는지 확인하고자 다음과 같은 내용으로 테스트를 진행하였다.

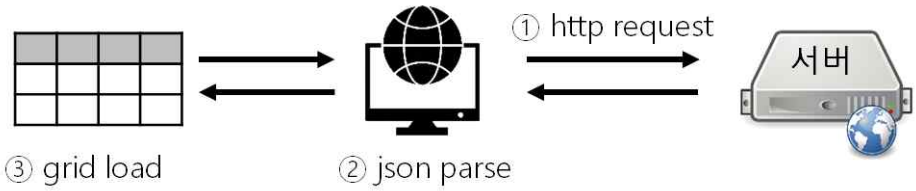
[표 3-1] 테스트를 진행한 PC 환경

테스트 PC 환경
CPU : Intel Core i5-8250U RAM : 8.00 GB O/S : 윈도우즈 10 Pro x64 브라우저 : Chrome 84.0.4147.135

[표 3-2] 테스트 조건

테스트 조건
- 데이터는 JSON 형식을 사용, 각 Row별 필드 수는 13개 - 그리드는 페이징을 사용하고 페이지당 15개의 Row를 표시

[표 3-1]의 테스트 환경과 [표 3-2]의 테스트 조건에서 [그림 3-16]과 같은 흐름으로 1천 건, 1만 건, 10만 건 데이터를 요청하여 화면의 그리드 컨트롤에 바인딩을 완료할 때까지 소요된 시간을 구간별로 각 5회씩 측정하여 [그림 3-17]과 같은 결과를 얻을 수 있었다.



[그림 3-16] 그리드 컨트롤 테스트 흐름도

Data : 1천건 (287,028 byte)						
항목	1회차	2회차	3회차	4회차	5회차	평균
http request	0.117	0.139	0.116	0.134	0.156	**0.132**
json parse	0.002	0.002	0.002	0.001	0.001	**0.002**
grid load	0.023	0.023	0.024	0.023	0.024	**0.023**

Data : 1만건 (2,870,028 byte)						
항목	1회차	2회차	3회차	4회차	5회차	평균
http request	0.677	0.503	0.509	0.730	0.369	**0.558**
json parse	0.021	0.024	0.019	0.019	0.019	**0.020**
grid load	0.031	0.033	0.035	0.034	0.034	**0.033**

Data : 10만건 (28,700,028 byte)						
항목	1회차	2회차	3회차	4회차	5회차	평균
http request	4.562	4.105	3.569	3.940	3.144	**3.864**
json parse	0.203	0.201	0.209	0.203	0.206	**0.204**
grid load	0.024	0.025	0.030	0.028	0.030	**0.027**

[그림 3-17] 테스트 결과

테스트 결과에서 조회되는 데이터의 증가로 http request 후 응답 소요 시간은 비교적 일정하게 증가하였다. 다만 http request 후 응답 데이터를 수신할 때 통신 처리, 웹 서버 처리, 데이터베이스 조회 및 서버 측 데이터 변환과 네트워크 송신 등 여러 단계를 거치고 있기 때문에 데이터 건수의 증가율과 소요 시간은 동일한 비율로 증가하지는 않았다. JSON parse 소요 시간은 데이터 건수의 증가율과 동일한 비율로 일정하게 증가하였다. 반면 grid load 소요 시간은 비교적 일정한 결과를 얻을 수 있었다. 이는 데이터의 증가에도 소요시간이 거의 동일하므로 대량의 데이터를 다루는 데 적합하다. 추가로 그리드 컨트롤의 페이지 이동 시간도 측정하였는데, 이 역시 grid load 소요 시간과 비슷한 0.02초에서 0.03초로 일정하여 데이터의 증가와 무관하게 일정한 성능을 내고 있음을 확인하였다.

위와 같은 테스트 결과는 jsGrid가 대량의 데이터를 처리하는 성능이 양호하여 업무에 사용하는데 적합하다고 판단하였다.

· **트리 컨트롤** : 트리 컨트롤은 jQuery 기반의 Simple Tree를 사용한다 [24]. 오픈 소스 기반의 트리 컨트롤은 매우 다양하지만 이전에 사용하기로 한 디자인 템플릿과 디자인적으로 이질감이 없어 해당 오픈 소스를 본 연구에 사용하기로 한다.

· **리스트 컨트롤** : 리스트 컨트롤은 Design Template에서 함께 제공하는 Multiselect 오픈 소스 라이브러리 컨트롤을 사용한다.

· **기타 컨트롤** : 앞서 분석한 요구사항에 정의되어있지는 않지만 앞서 선정한 디자인 템플릿에서 함께 제공하는 달력(FullCalendar)과 차트(C3Chart) 라이브러리도 본 연구에 포함한다.

(2) 오픈 소스 라이브러리 라이선스 확인

마지막으로 라이브러리들이 표시한 라이선스를 확인하여 차후 문제나 분쟁의 소지가 있는지 확인하였다. [표 3-3]과 같이 모두 MIT 라이선스 또는 그보다 제약이 덜한 라이선스를 사용하고 있어 문제가 없음을 확인하였다.

[표 3-3] 본 연구에 사용할 라이브러리의 라이선스 내역

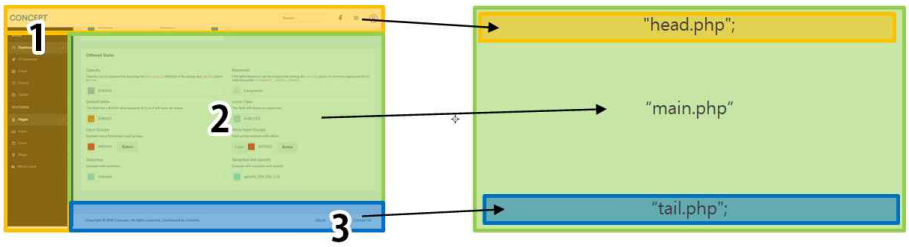
라이브러리	라이선스
Bootstrap	MIT
jQuery	MIT
Colorlib concept Template	MIT
jsGrid	MIT
Simple Tree	MIT
Multi-select	WTFPL
FullCalendar	MIT
C3Chart	MIT

제 4 장 설계 및 구현

제 1 절 구조설계

제3장에서 정의한 UI 프레임워크 기술구조를 이용하여 어떻게 화면을 구성하고 서버와 어떻게 통신 할지 구조적인 설계가 필요하다. 본 절에서는 화면 구성과 반응형 레이아웃 구성, 서버와의 통신 그리고 데이터 바인딩 부분으로 나누어 설계한다.

· **화면 구성** : 화면은 크게 공통의 영역과 업무의 영역으로 나뉜다.



[그림 4-1] 화면의 구성

공통 영역은 [그림 4-1]의 1번과 3번 영역과 같이 모든 업무 화면이 공통으로 표시해야 할 영역이다. 업무 영역은 [그림 4-1]의 2번과 같이 사용자가 선택한 메뉴에 맞는 업무 화면이 표시되는 업무영역이다.

업무 화면을 구현할 때 [그림 4-1]의 1, 3번 는 공통 영역은 업무 화면에서 미리 개발된 head.php와 tail.php 파일을 포함(include)하는 고전적인 방법을 사용하였다. 이는 차후 서버 사이드 스크립트 언어나 환경에 따라 변경될 수 있다. 중요한 것은 업무별 개발자가 [그림 4-1]의 2번의 업무영역의 개발에만 집중할 수 있도록 설계하여 공통 영역의 변화가 업무 영역에 영향이 없도록 하는 것이다.

· **반응형 레이아웃** : 업무 화면을 반응형으로 구현하기 위하여 앞서 선정한 Bootstrap의 그리드 시스템(Grid System)을 활용한다.

	Extra small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	Extra large ≥1200px
Max container width	None (auto)	540px	720px	960px	1140px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-
# of columns	12				
Gutter width	30px (15px on each side of a column)				
Nestable	Yes				
Column ordering	Yes				

[그림 4-2] Bootstrap의 Grid System options [25]

Bootstrap의 그리드 시스템은 [그림 4-2]처럼 화면을 가로 해상도에 따라 5가지(Extra small, Small, Medium, Large, Extra large)로 분류하고 장치의 해상도에 따라 각 영역이 어떻게 위치할지 결정한다.

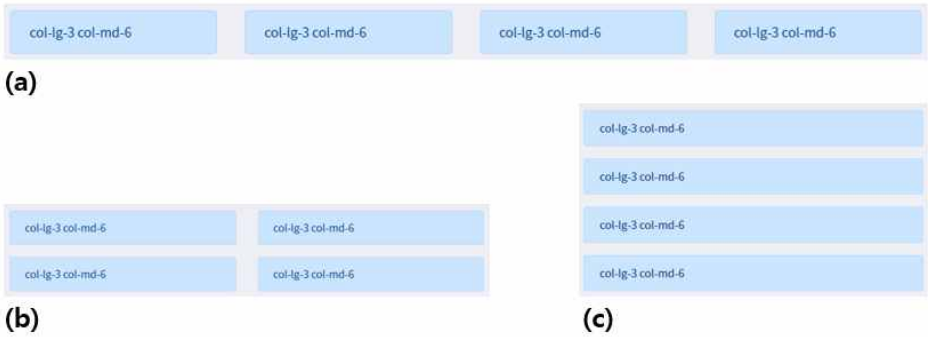
그리드 시스템에서 한 줄(Row)은 가상의 12개의 열(Col)으로 나누고 있으며 각 열마다 장치의 해상도에 따라 얼마만큼 영역을 차지하게 될지 HTML 태그(Tag)의 클래스(Class)를 지정하는 방식으로 동작한다.

[그림 4-3] 그리드 시스템 예시 (1)

[그림 4-4] 그리드 시스템 예시 (2)

예를 들어 [그림 4-3]과 같이 col-sm-8과 col-sm-4로 지정하면 장치의 디스플레이 해상도가 576px 이상인 경우 col-sm-8은 12개의 열 중 8칸을, col-sm-4는 4칸을 차지하도록 설정된다. 만약 장치의 디스플레이 해상도가 576px 미만이 되면 [그림 4-4]와 같이 두 줄로 표시가 되고 12개의 열의 가로 폭을 갖게 된다.

그리드 시스템에서 하나의 열에 클래스를 다중으로 지정하면 여러 가지 해상도에 대응되는 의도된 레이아웃을 구성할 수 있다.



[그림 4-5] 다중 클래스 지정 시 그리드 레이아웃의 변화

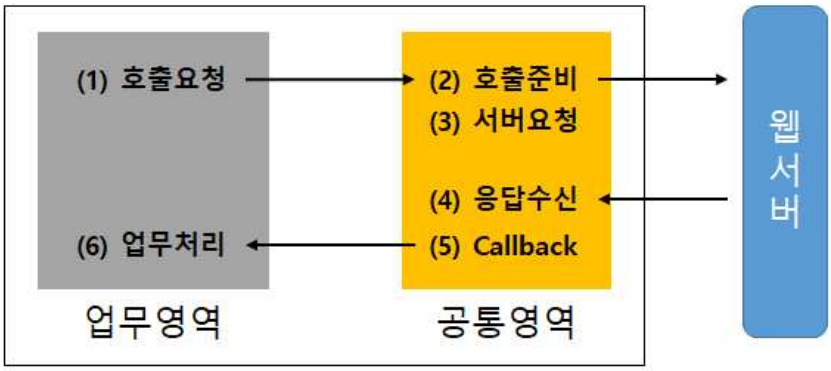
[그림 4-5]와 같이 열의 클래스를 col-lg-3 col-md-6으로 지정하게 되면 가로 해상도가 960px 이상인 경우(a)와 720px 이상인 경우(b), 그리고 그 이하인 경우(c)로 레이아웃이 변경된다.

이처럼 Bootstrap의 그리드 시스템을 활용하여 업무별로 화면을 설계하면 장치의 해상도에 따라서 레이아웃이 의도에 맞게 변화하는 반응형 디자인 화면을 구현할 수 있다.

· **서버와의 통신** : JavaScript의 Ajax를 사용하여 서버에 접근할 때는 기본적으로 GET, POST와 같은 요청 방법(Request Method), 그리고 호출할 페이지 주소(URL), 전달할 변수(Parameter) 그리고 비동기 호출 여부를 지정하여 사용한다.

업무 화면별로 서버와 통신하기 위한 Ajax 통신을 제각각 구현하여 사용하게 되면, 통신 방식의 차이에 따른 충돌이나 오류가 발생할 수 있고 서버와 통신하기 위한 변화사항이 발생하게 되면 모든 화면을 수정해야하는 등의 문제점을 야기한다.

그러므로 UI 프레임워크에서 서버와 통신에 필요한 Wrapper Function을 구현하여 제공하는 것이 마땅하다. [그림 4-6]과 같이 업무영역에서는 (1) UI 프레임워크에 Ajax 통신 시 사용할 요청 방법,



[그림 4-6] 통신 방식 구조도

페이지 주소, 변수, 비동기 호출 여부와 호출이 완료되었을 때 수행 할 Callback Function의 이름을 받아 호출을 요청하고 (2) UI 프레임워크는 요청한 형태와 정보에 맞게끔 처리한 후 (3) 실제 서버에 요청하고, (4) 요청에 대한 응답이 수신되었을 때 (5) 업무영역에서 지정한 Callback Function에 수신한 데이터를 전달하여 (6) 업무영역에서 업무에 필요한 처리를 하는 형태로 설계하였다.

이처럼 업무영역에서 서버와 통신을 요청하는 구조로 설계하면 공통영역에서 유효성 검증이나 추가적인 암호화 작업 등 공통적인 사항을 업무 영역에 영향이 없이 변경할 수 있고, 다중 시스템 구성 시 테스트 환경과 실제 환경을 나눌 때도 유용하게 사용할 수 있다.

· **데이터 바인딩 :** 데이터 바인딩(Data Binding)은 화면에 보이는 컨트롤에 원하는 값이 나오도록 하거나 컨트롤의 값을 이용하기 위하여 변수로 전환하는 것을 말한다. 만약 프레임워크에서 아무런 지원을 하지 않는다면 업무화면 개발자는 서버에서 조회한 모든 데이터를 화면에 배치한 모든 컨트롤에 대하여 JavaScript나 jQuery를 이용하여 직접 바인딩 처리를 해야 한다.

금융기관 A사에서 처리하는 금융 관련 업무에서는 다양한 관점에서 데이터를 조회하거나 추출하여 확인 및 분석하는 일이 많다. 그래서 조회 조건 컨트롤이 다수인 화면과 증권의 종목정보 등 다수의 원천

데이터를 관리하는 화면의 비중이 높다. 이러한 화면들을 별도의 지원 없이 각 업무에서 직접 구현하면 데이터 바인딩에 필요한 코드가 반복적으로 발생하게 된다.

예를 들어 5개의 조회조건과 20개의 데이터 항목을 가지는 업무 화면에서 조회, 입력, 수정 기능이 존재한다고 가정하면 조회용 조건 항목 5개와 20개의 데이터 항목이 바인딩 처리 대상이 된다. 그렇다면 조회 5번, 입력, 수정, 삭제 각각 20번의 바인딩이 필요하여 65번의 데이터 바인딩 소스 코드를 구현해야 한다.

이러한 문제는 바인딩 규칙을 정의하고 규칙대로 정의된 컨트롤을 UI 프레임워크에서 자동으로 바인딩 할 수 있도록 설계 및 지원하여 해결할 수 있다. 화면에 배치되는 컨트롤을 기능에 따라 (1) 조회 컨트롤과 (2) 입력/수정/삭제 컨트롤로 분류하고 [표 4-1]과 같이 분류에 따라 클래스 명을 지정하여 지정된 클래스 명에 따라 UI 프레임워크가 데이터 바인딩을 할 수 있도록 설계하였다.

[표 4-1] 컨트롤 종류에 따른 구분

	(1) 조회조건용	(2) 입력/수정/삭제용
클래스명	dataRCond	dataCUDVal

추가로 UI 프레임워크가 컨트롤의 종류에 따라 서로 다른 바인딩 처리를 하기 위하여 텍스트, 셀렉트, 그리드 등 컨트롤의 종류를 나타내는 구분자를 Data Attribute를 이용하여 지정하도록 정의하였다.

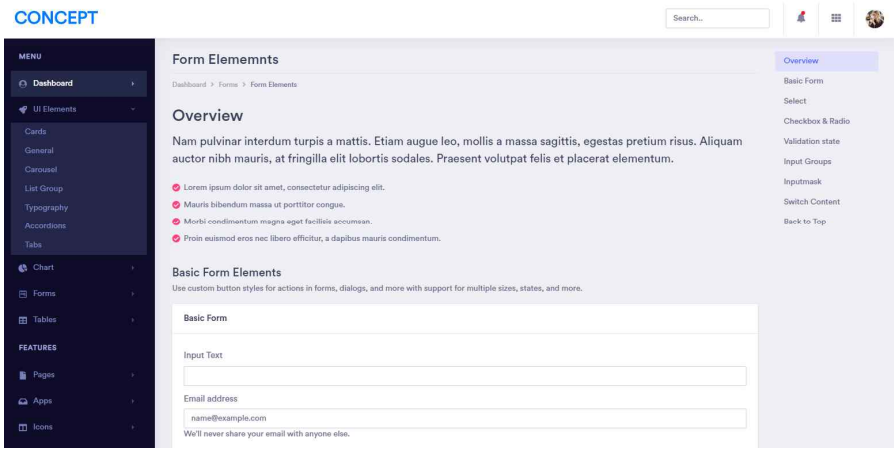
[표 4-2] 컨트롤의 종류 정의

Data Attribute 명	컨트롤 종류
data-valtype	input-text, input-select, input-checkbox, input-radio, jsgrid, mul-sel

제 2 절 구현

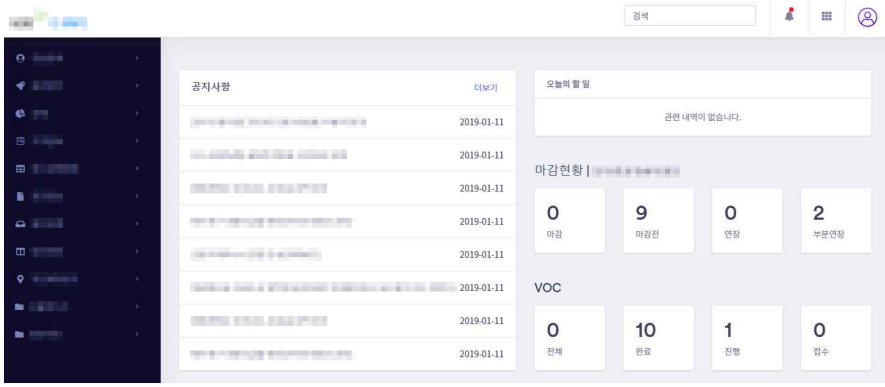
본 절에서는 앞서 설계한 구조를 바탕으로 금융기관 A사 기간계 시스템의 UI를 웹 기반으로 구현하는 내용을 기술한다.

(1) 디자인 템플릿 수정 및 구조화



[그림 4-7] 디자인 템플릿

앞서 선정한 디자인 템플릿을 브라우저에 표시하면 [그림 4-7]과 같은 화면이 표시된다. 이 템플릿의 소스 코드를 수정하고 구조화하여 업무화면 영역과 공통의 영역으로 나누어 표시되도록 하였다.



[그림 4-8] 수정된 템플릿

[그림 4-8]은 [그림 4-7]의 기본 디자인 템플릿을 수정하여 변경한 것으로 상단 영역과 메뉴 영역 등 모든 화면에서 공통으로 사용할 부분에 해당한다.

그리고 [그림 4-1]에서 설계한 바와 같이 개별 업무화면 개발자는 공통영역 코드를 구현하지 않고 업무 화면영역 구현에만 집중 할 수 있도록 [표 4-3]의 소스 코드와 같이 구현하였다.

[표 4-3] include_once를 이용한 공통영역

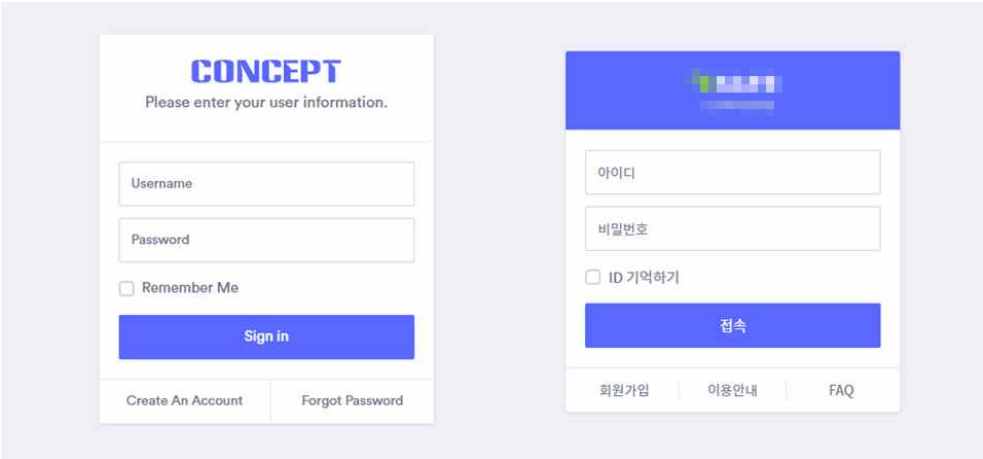
```
<?php
        include_once "head.php";
?>
<!-- 업무별 화면 소스코드 시작 -->

                    ..... (중략) .....

<!-- 업무별 화면 소스코드 끝 -->
<?php
        include_once "tail.php";
?>
```

업무 화면을 구현하는 소스코드는 처음과 끝에 공통기능으로 작성된 head.php와 tail.php 파일을 포함하여 사용하면 된다. 이 파일을 불러오는 것만으로도 기본적인 공통 영역이 모두 표시되기 때문에 나머지 업무 영역에서 다루고 표시할 내용만 구현하면 된다.

이번 연구에서는 서버 환경의 서버 스크립트 언어로 PHP를 사용하였기 때문에 PHP의 include_once를 이용하여 구현하였지만, 필요에 따라서 클래스나 함수를 통해 제공하는 등 여러 가지 방법이 존재한다. 이는 실제 적용될 서버 스크립트 언어에 따라서 적절히 변경하여 적용할 수 있다.

(2) 로그인/로그아웃

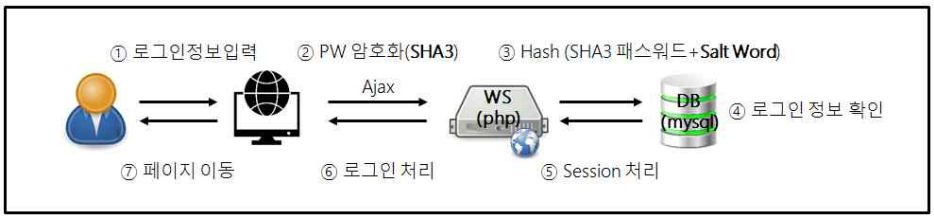
로그인과 로그아웃 기능을 구현하기 위하여 먼저 디자인 템플릿에서 함께 제공하는 로그인 화면 템플릿을 수정하였다. [그림 4-9]의 왼쪽이 템플릿에서 제공하는 디자인이며, 우측이 수정한 디자인이다.



[그림 4-9] 로그인 템플릿 수정

로그인 프로세스는 본 연구의 직접적인 범위는 아니지만, UI 프레임워크가 각 업무화면의 접근 권한이 있는지 판단하여 접근 제어를 구현하기 위해 구현하였다.

[그림 4-10]과 같이 통상적으로 사용하는 로그인 처리와 유사한 과정으로 구현하였지만, 보안성 향상을 위하여 서버에서 사용자별로 패스워드에 임의의 Salt Word를 추가하는 기법을 사용하였다 [20].



[그림 4-10] 로그인 과정 흐름도

(3) 서버 통신 지원 함수

앞서 업무화면에서 필요한 데이터를 송수신하기 위하여 UI 프레임워크는 Ajax를 이용하여 서버와 통신하는 기능의 함수(Wrapper function)을 제공하기로 하였다. 이러한 공통 함수는 금융환경에서 통계 목적이나 보안, 분쟁 등의 문제로 사용자가 시스템의 UI를 통해 이루어지는 다양한 행위들을 기록하는 경우가 많기 때문에 서버와의 통신을 표준화된 방식으로 제공하는 것은 매우 유용하다.

[표 4-4] 표준 통신 함수 정의

```
getHttpRequestJson(PATH, CALLBACK, METHOD, PARAM)

PATH : 서버스크립트파일경로
CALLBACK : 콜백 함수 이름
METHOD : GET 또는 POST
PARAM : 서버로 송신할 파라미터
```

[표 4-4] 내용과 같이 정의된 함수를 업무화면에서 호출하여 통신을 요청(getHttpRequestJson)하면 UI 프레임워크는 통신이 정상적으로 완료되는 경우 그 결과를 콜백 함수에 인자로 돌려주도록 구현하였다. 만약 서버와의 통신 자체에 문제가 발생하는 경우에는 콜백 함수를 호출하지 않고 UI 프레임워크 자체에서 오류를 발생시켜 오류를 사용자에게 알리도록 구현하였고, 그 외 서버 스크립트에서 발생하는 오류의 경우는 업무에서 처리할 수 있도록 콜백 함수를 호출시키도록 구현하였다.

```
function getHttpRequestJson(URL, ID, ParamType, Json) {
  // UI 프레임워크에서 필요한 기능을 추가로 구현
  // ex) 통계성 데이터 기록, 암호화 처리, 접근권한 확인 등

  $.ajax({
    type: ParamType,
    url : URL,
    data: Json,
    dataType : "json",
    success : function(resultdata, status, xhr) {

      // 콜백 함수 호출
      var fn = window[ID];

      if (typeof fn === "function") {
        fn(resultdata);
      }
    },
    error: function(jqXHR, textStatus, errorThrown) {
      alert('서버 요청 중 오류가 발생하였습니다.('+ID+')');
      console.log(jqXHR.responseText);
    }
  });
}
```

[그림 4-11] getHttpRequestJson의 소스코드

(4) 공통 검색 모듈

금융기관 A사 기간계 환경에서 다수의 화면에서 자주 사용하는 공통 성격의 데이터가 존재한다. 이는 회사나 종목코드와 같은 것들이다. 금융기관 A사의 업무 분장 데이터는 관리부서가 존재하여 업무의 영역이라고 볼 수 있으나, 다수의 화면에서 공통으로 사용하기 때문에 공통 영역에서 모듈 형태로 제공하고 있다. 본 연구에서도 현행 금융기관 A사 업무 환경과 같이 공통 영역에서 모듈이 형태로 제공하여 업무 화면에서 손쉽게 사용할 수 있도록 하였다.

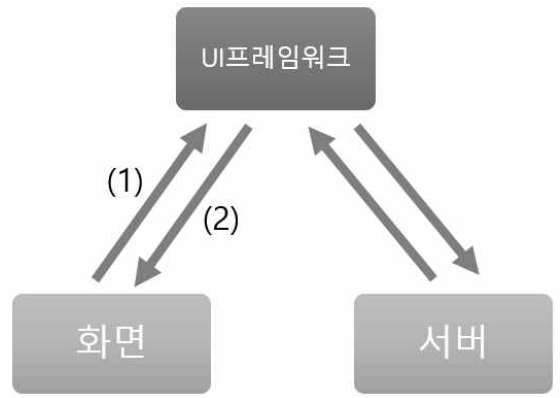
[그림 4-12] 공통 모듈의 사용예

[그림 4-12]는 공통 모듈을 이용하는 업무화면의 예시로 PHP의 include_once를 이용하여 업무 화면 소스 코드 상단에 구현된 공통 모듈 서버 스크립트를 포함하도록 선언하고, 원하는 위치에 공통 모듈 코드를 생성시키는 함수를 호출하여 사용하도록 하였다.

또한, 업무 화면에 배치된 컨트롤의 ID와 공통 모듈의 컨트롤 ID가 중복되면 화면이 비정상적으로 동작할 수 있어 업무 개발자가 공통 모듈 내부 컨트롤의 ID를 지정하여 사용할 수 있도록 설계하여 의도하지 않은 오류를 발생시키지 않도록 구현하였다.

(5) 데이터 바인딩 자동화

본 절에서는 앞서 설계한 바와 같이 데이터와 컨트롤 간 바인딩과 초기화를 지원하는 공통 기능을 구현한다. 데이터 바인딩은 [그림 4-13]의 (1)과 같이 화면의 컨트롤의 값을 서버로 전송하기 위하여 데이터를 변수 처리하는 것과 [그림 4-13]의 (2)와 같이 서버의 데이터를 화면의 컨트롤에 바인딩 처리 하는 것의 두 가지 방향이 있다.



[그림 4-13] 데이터 바인딩의 방향

금융기관 A사 기간계 환경에서 화면의 컨트롤은 조회를 위한 컨트롤의 영역과 데이터를 관리하기 위한 영역이 서로 구분되어있다. 즉, 조회를 위한 조건의 값에 해당하는 컨트롤과 입력, 수정, 삭제를 위한 값을 갖는 컨트롤을 구분하여 사용한다.

본 연구에서 구현하는 UI 프레임워크의 데이터 바인딩 지원 기능도 영역을 구분하여 동작할 수 있도록 설계한다. 조회 조건을 변수화 하는 '조회 조건 데이터 처리 지원', 서버에서 조회한 데이터를 컨트롤에 연결하는 '데이터 바인딩 처리 지원', 화면의 값을 서버로 전송하기 위한 '데이터 전송 처리 지원'으로 구분하여 다음과 같이 구현하였다.

· 조회 조건 데이터 처리 지원

사용자가 서버에서 데이터를 조회하는 경우 조회 조건에 해당하는 컨트롤의 값을 읽어서 서버로 조회 요청하면 조건 값에 해당하는 데이터를 수신하게 된다. 이때 UI 프레임워크는 조회 조건에 해당하는 컨트롤의 값만 처리할 수 있도록 클래스를 정의하였다.

```
<input id="tempText" type="text" class="form-control dataCUDVal"
       data-valtype="input-text" data-valcrud="CUD">
```

[그림 4-14] 조회 컨트롤 예시

[그림 4-14]의 예시에서 조회조건에 해당하는 컨트롤의 class에는 'dataRCond'를 지정하고, data-valtype에는 컨트롤의 종류를 알 수 있도록 'input-select'로 지정하였다.

```
function getRCond() {
  var dataArr = {};

  // dataRCond 클래스 순환
  $('.dataRCond').each(function(i, obj) {
    if($("#"+obj.id).data("valtype") == "input-checkbox")
    {
      // 체크박스이면 true/false를 1/0으로 변환
      if($("#"+obj.id).prop("checked")) {
        dataArr[obj.id] = "1";
      }
      else
      {
        dataArr[obj.id] = "0";
      }
    }
    else if($("#"+obj.id).data("valtype") == "jsgrid")
    {
      dataArr[obj.id] = $("#"+obj.id).jsGrid("option","data");
    }
    else if($("#"+obj.id).data("valtype") == "input-radio")
    {
      // radio면 안에 name으로 찾는다.
      dataArr[obj.id] = $('input:radio[name='+key+']:checked').val();
    }
    else if($("#"+obj.id).data("valtype") == "mulsel")
    {
      dataArr[obj.id] = $("#"+obj.id).val().toString();
    }
    else
    {
      dataArr[obj.id] = $("#"+obj.id).val();
    }
  });

  return dataArr;
}
```

[그림 4-15] 조회 조건 처리 코드

개별 업무 화면에서 UI 프레임워크에 구현된 getRCond 함수를 호출하면 [그림 4-15]와 같이 dataRCond 클래스가 선언되어있는 컨트롤들을 모두 순환하면서 data-valtype에 따라 처리하여 데이터 배열로 돌려준다. 업무에서는 UI 프레임워크가 수집한 조회 조건에 해당하는 데이터 배열을 이용하여 [그림 4-16]과 같이 서버에 데이터 조회 요청을 할 수 있다.

```
var dataArr = getRCond();

// GET방식 json조회는 array로 그대로 호출하여 $_GET으로 받음
getHttpRequestJson("./php/sample01/getSample01Data.php",
                   "getSample01DataCb",
                   "GET",
                   dataArr);
```

[그림 4-16] getRCond의 결과를 이용하여 데이터 요청 예제

이처럼 UI 프레임워크가 제공하는 함수를 사용하면 업무 화면을 구현할 때 손쉽게 필요한 데이터를 바인딩 할 수 있고 이를 서버에 전달할 수 있어 효율적이다.

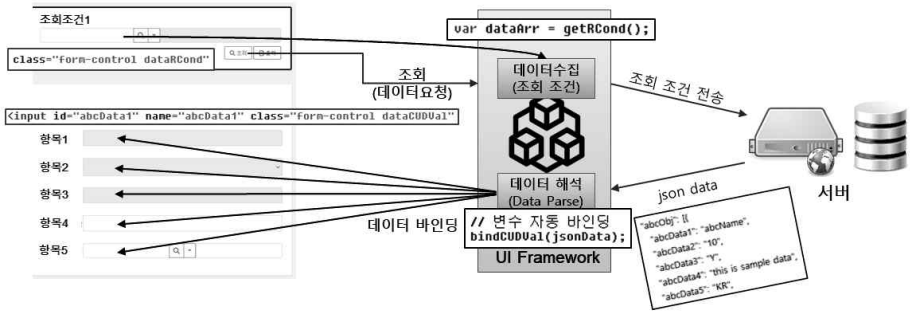
· **데이터 바인딩 처리 지원**

앞서 정의한 바와 같이 클라이언트의 데이터 요청에 대하여 JSON 형태의 데이터로 응답(Response)하도록 설계하였다. JSON 형태의 데이터는 [그림 4-17]과 같이 key와 value의 쌍의 형태로 구성된다.


```
{
  "result": true,
  "abcObj": [{
    "abcData1": "abcName",
    "abcData2": "10",
    "abcData3": "Y",
    "abcData4": "this is sample data",
    "abcData5": "KR",
    "abcData6": null,
    "abcData7": false,
    "abcData8": 150,
    "abcData9": null,
    "abcData10": "20200101235959"
  }],
  "gridObj": {
    "grid1": [{
      "gridData1" : "1",
      "gridData2" : "name1",
      "gridData3" : "100",
      "gridData4" : "Y",
      "gridData5" : "N",
      "gridData6" : "20200102235959"
    }, {
      "gridData1" : "2",
      "gridData2" : "name2",
      "gridData3" : "200",
      "gridData4" : "N",
      "gridData5" : "N",
      "gridData6" : "20200103235959"
    }]
  }
}
```


[그림 4-17] JSON 응답 예시

UI 프레임워크는 서버에서 받은 JSON 형태의 데이터를 [그림 4-18]와 같은 흐름으로 바인딩하게 된다. 먼저 JSON 데이터를 해석(Parse)한 후 [표 4-5]의 코드와 같이 각 컨트롤에 지정된 클래스와 data-attribute를 이용하여 바인딩 처리할 컨트롤을 탐색한다. 만약 JSON 데이터의 Key 값과 컨트롤의 ID가 맞는 경우 JSON 데이터의 Value 값을 컨트롤에 바인딩 처리를 한다.



[그림 4-18] 데이터 바인딩 처리 흐름도

[표 4-5] 바인딩을 위한 컨트롤 예시

```
<input id="isin" type="text" required="" data-parsley-type="" placeholder=""
class="form-control dataCUDVal" data-valtype="input-text"
data-valcrud="CRU">
```

화면에 표시될 데이터는 조회 조건과 1:1의 관계를 갖는 일반적인 데이터와 1:N의 관계의 다중 데이터가 복합적으로 구성될 수 있다. 서버의 응답은 [그림 4-17]의 예시와 같이 1:1의 관계를 갖는 일반 컨트롤의 데이터는 abcObj에, 1:N의 관계를 갖는 그리드와 멀티셀렉트 컨트롤의 데이터는 gridObj와 mulselObj 등으로 구분하여 응답하도록 설계하였다.

업무 화면 개발자는 콜백 함수를 통해 받은 JSON형태의 데이터를 UI 프레임워크에서 제공하는 bindCUDVal 함수를 통해 바인딩 요청하면 UI 프레임워크는 [그림 4-19]와 같이 데이터의 종류와 컨트롤에 따라 바인딩 처리를 수행한다.

```
if(data.abcObj[0] != null && data.abcObj[0] != "") {
  Object.keys(data.abcObj[0]).forEach(function(key) {
    if(typeof $("#"+key)[0] != "undefined")
    {
      //log($("#"+key)[0].type);
      var objType = $("#"+key).data("ksdvaltype");
      if(objType == "input-checkbox")
      {
        $("#"+key).prop("checked", data.abcObj[0][key]);
      }
      else if(objType == "input-radio")
      {
        $('input:radio[name='+key+']:input[value=' + data.abcObj[0][key] + ']').prop("checked", true);
      }
      else
      {
        $("#"+key).val(data.abcObj[0][key]);
      }
    }
  }));
}
```

(a)

```
if(data.gridObj != null && data.gridObj != "") {
  // gridObj 바인딩
  Object.keys(data.gridObj).forEach(function(key) {
    if(typeof $("#"+key)[0] != "undefined") {
      var objType = $("#"+key).data("ksdvaltype");
      if(objType == "jsgrid") {
        $("#"+key).jsGrid("option", "data", data.gridObj[key]);
      }
    }
  }));
}
```

(b)

```
if(data.mulselObj != null && data.mulselObj != "") {
  // mulselObj 바인딩
  Object.keys(data.mulselObj).forEach(function(key) {
    if(typeof $("#"+key)[0] != "undefined") {
      var objType = $("#"+key).data("ksdvaltype");
      if(objType == "mulsel") {
        var selectedArr = [];

        // option 추가
        for(var i = 0; i < data.mulselObj[key].length; i++){
          $("#"+key).multiSelect('addOption', {
            value: data.mulselObj[key][i].inputSeq,
              text: data.mulselObj[key][i].inputSeq+' '+data.mulselObj[key][i].serviceNm }});

          if(data.mulselObj[key][i].selectedYn == "Y")
          {
            log(data.mulselObj[key][i].inputSeq + ' 선택처리');
            selectedArr.push(''+data.mulselObj[key][i].inputSeq);
          }
        }

        // 선택처리
        $("#"+key).multiSelect('select', selectedArr);
      }
    }
  }));
}
```

(c)

[그림 4-19] bindCUDVal 함수 코드

[그림 4-19]의 (a)는 1:1 관계의 abcObj 데이터를 처리하는 코드이고, (b)는 1:N 관계를 갖는 gridObj 처리 코드이며 (c)는 멀티셀렉트 데이터를 처리하는 mulselObj 처리 코드이다.

위와 같이 데이터의 성격과 종류에 따라 처리 프로세스를 구분하면 추후 새로운 컨트롤에 대한 지원이 추가되어야 할 때 JSON 데이터 구조와 UI 프레임워크 코드를 추가하더라도 기존 업무 화면에 영향 없이 유연하게 대응할 수 있다.

· 데이터 전송 처리 지원

업무 화면에 존재하는 컨트롤에 표시된 값들은 서버의 값이 조회되어 바인딩 되었거나 사용자가 직접 입력하여 바인딩 된 값이다. 이 값들은 서버로 전송되어 입력(Create), 수정(Update) 또는 삭제(Delete)와 같은 데이터 처리를 위해 사용된다. 이 때도 UI 프레임워크가 데이터 처리를 지원하도록 구현하였다.

```
<input id="korSecnNm" type="text" required="" placeholder=""
       class="form-control dataCUDVal" data-valtype="input-text" data-valcrud="CUD">
<input type="checkbox" id="rightsTpcd01" name="rightsTpcd01"
       class="custom-control-input dataCUDVal" data-valtype="input-checkbox" data-valcrud="CUD">
<select class="form-control dataCUDVal" id="isinType"
        data-valtype="input-select" data-valcrud="CUD">
```

[그림 4-20] CUD를 위한 컨트롤 코드 예시

[그림 4-20]과 같이 전송 처리할 컨트롤의 클래스에 dataCUDVal를 선언하고, 컨트롤의 종류를 구분하는 data-valtype과 각 컨트롤이 입력, 수정, 삭제 처리에서 사용되는 컨트롤인지 구분 할 수 있도록 data-valcrud를 선언한다. data-valcrud의 경우 입력 시에만 사용되는 컨트롤이라면 “C”로, 입력과 수정에 사용되는 컨트롤이라면 “CU”와 같이 사용한다.

```
var dataArr = getCUDVal("CU");

var jsonDataArr = JSON.stringify(dataArr);

getHttpRequestJson("./php/sample01/saveSample01Data.php", "saveCb", "POST", jsonDataArr);
```

[그림 4-21] getCUDVal 사용 예시

업무 화면 개발자는 데이터 전송 처리 시 UI 프레임워크의 getCUDVal 함수를 사용하여 처리의 대상이 되는 컨트롤의 값만 손쉽게 받을 수 있다. [그림 4-21]의 예시에서 입력(C)과 수정(U)에만 사용되는 컨트롤만 처리하도록 변수로 "CU"를 사용하였고, UI 프레임워크는 사용자가 요청한 입력과 수정에만 사용될 컨트롤의 값만 결과로 넘겨주어 서버에 전송할 데이터를 효과적으로 수집 할 수 있다.

(6) 팝오버(Popover)

금융 환경에서 사용자가 데이터 처리를 요청하기 전에 컨트롤들의 값이 유효한 값인지 검증하는 것은 매우 중요하다. 만약 비정상적인 값이나 잘못된 값이 전송되는 경우 작게는 단위 업무처리 오류부터 크게는 연계된 업무나 외부 기관에 영향을 미쳐 금융 업무가 정상적으로 개시될 수 없는 상황도 발생하기 때문이다. 이와 같은 문제를 최소화하려면 화면에서 사용자가 입력한 값을 검증하면서 어떤 컨트롤의 값이 비정상적인지 정확하게 인지 할 수 있도록 시각화해야 한다.

[그림 4-22] Bootstrap에서 제공하는 팝오버

UI 프레임워크에서 사용하는 Bootstrap 라이브러리에는 [그림 4-22]와 같은 팝오버(Popover) 컴포넌트를 제공한다. UI 프레임워크는 팝오버 컴포넌트를 활용하여 지정된 컨트롤에 일정 시간만큼 팝오버 메시지를 보여줄 수 있도록 [그림 4-23]과 같은 함수를 구현하였다.

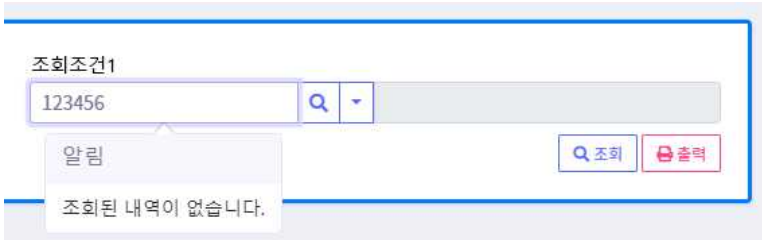
```
function showPopover(id, delay, title, text, placeNm) {
  var objId = $("#"+id);
  objId.popover({
    placement : placeNm,
    trigger: 'manual',
    title: title,
    content: text
  });

  objId.popover('show');
  objId.on('shown.bs.popover', function() {
      setTimeout(function() {
          objId.popover('hide');
      }, delay);
  });
}
```

[그림 4-23] 팝오버 Wrapper Function 코드

사용자의 입력 값 검증과 같이 업무 화면에서 사용자에게 특정한 위치에 메시지를 시각적으로 보여주려면 [그림 4-24]와 같이 컨트롤의 id와 표시될 시간(ms), 제목, 내용과 팝오버가 표시될 위치를 지정하여 호출하면 된다. 팝오버는 [그림 4-25]와 같이 표시되어 사용자가 화면의 어느 컨트롤의 값을 확인해야 하는지 직관적으로 알 수 있다.

```
showPopover(id, 1500, "알림", "조회된 내역이 없습니다.", "bottom");
```

[그림 4-24] 팝오버 호출 코드



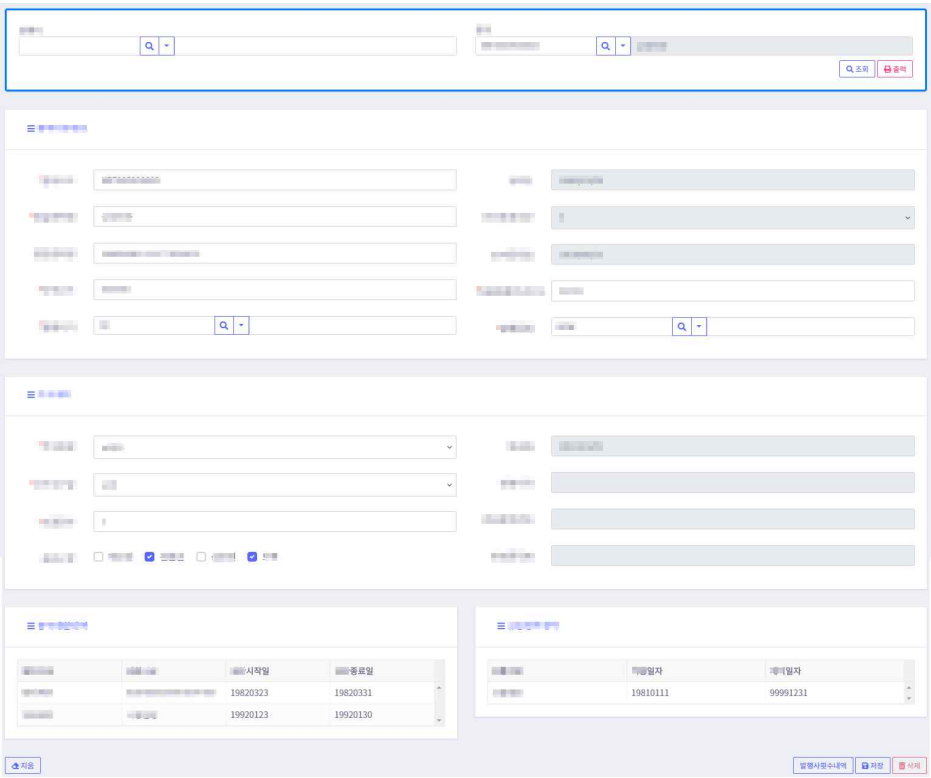
[그림 4-25] 팝오버 표시 예

(7) 업무 화면

현행 금융기관 A사 기간계 업무 화면이 본 연구에서 구현한 UI 프레임워크를 이용하여 동일하게 구현되고 기능적으로 같게 동작할 수 있는지 확인하기 위하여 앞선 제3장 1절에서 선정한 5본의 업무 예제 화면을 다음과 같이 구현하고 동작을 확인한다.

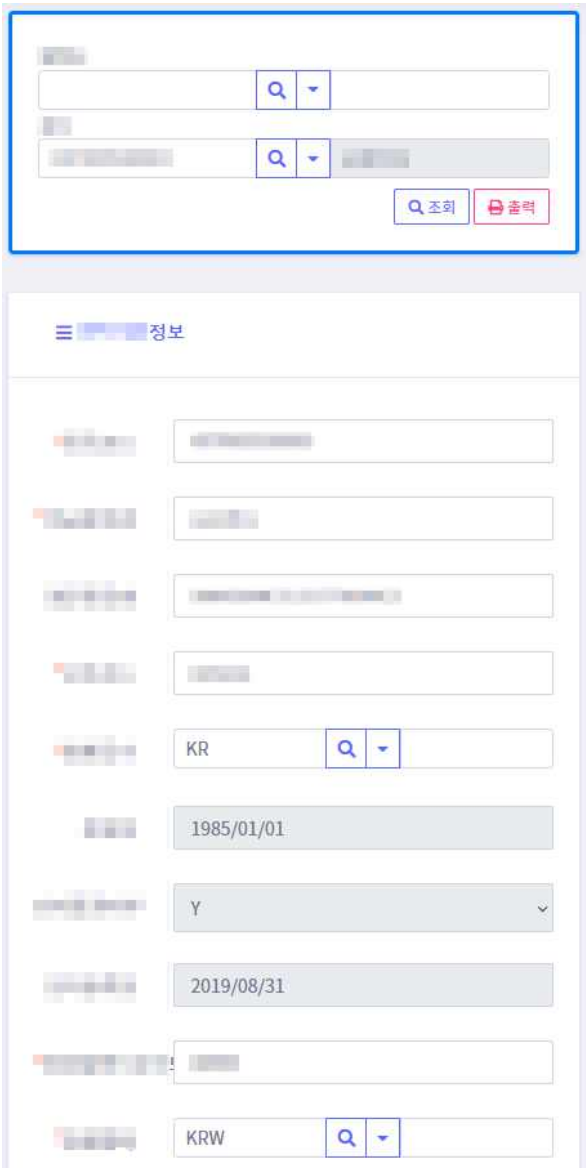
· **화면1**

본 화면은 일반 컨트롤이 주로 배치되며 상단 조회 영역에 공통 컨트롤이 존재하고 하단에 텍스트, 체크, 콤보 등이 함께 존재하는 화면이다. 이 화면을 UI 프레임워크 기반에서 [그림 4-26]과 같이 구현하였다.



[그림 4-26] 화면1 레이아웃

화면이 반응형 화면으로 동작할 수 있도록 Bootstrap의 그리드 시스템을 사용하여 화면의 해상도가 작아지는 경우 좌우로 배치된 컨트롤이 [그림 4-27]과 같이 컨트롤이 상하로 자동으로 변경되도록 구현하였다.



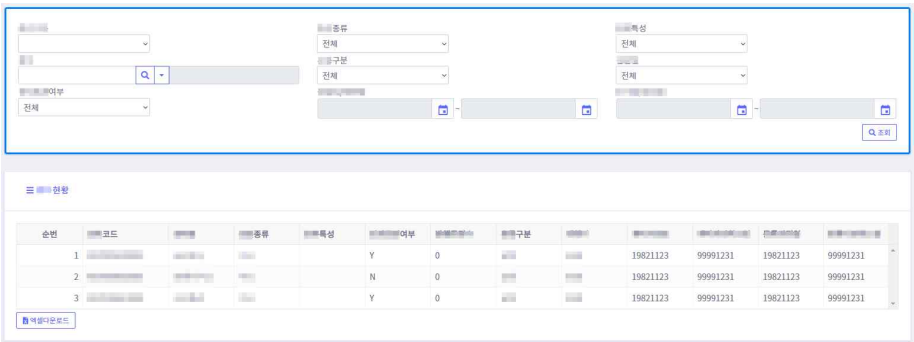
[그림 4-27] 화면1 모바일 레이아웃

인쇄(출력) 시 화면의 공통 영역 등 불필요한 컨트롤이 제외되고 인쇄할 수 있도록 인쇄에 제외할 컨트롤에 d-print-none 클래스 속성을 지정하여 구현하였고 [그림 4-28]와 같이 인쇄된다.

[그림 4-28] 화면1의 인쇄 레이아웃

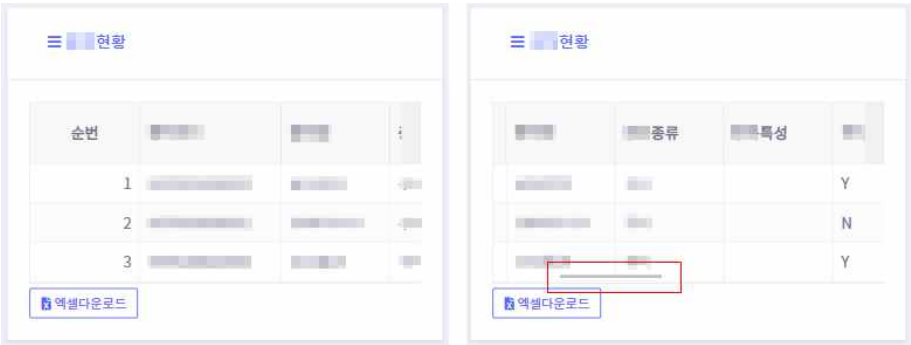
· 화면2

본 화면은 조회 기능만 사용하는 업무 화면으로 사용자가 상단의 조회영역에서 조건을 설정하면 서버에서 조회 조건에 부합하는 데이터를 하단 그리드 컨트롤에 표시한다. 이 화면을 UI 프레임워크 기반으로 구현하면 [그림 4-29]과 같다.



[그림 4-29] 화면2 레이아웃

본 화면에서 하단의 그리드 컨트롤은 가로로 많은 열(Column)을 표현하게 된다. UI 프레임워크에서 포함한 그리드 컨트롤(jsGrid)은 모바일 등 가로 해상도가 작은 환경에서는 [그림 4-30]의 우측 그림과 같이 그리드 자체에 스크롤 기능이 활성화된다. 사용자는 마우스나 스와이핑으로 그리드를 좌우로 움직이며 데이터를 확인 할 수 있다.



[그림 4-30] 가로 해상도가 작은 환경에서 그리드 컨트롤 동작 예시

또한 경험적으로 금융 환경에서 그리드 컨트롤을 사용하는 화면의 경우 인쇄보다는 그리드 컨트롤의 데이터를 직접 다운로드 하여 활용하는 경우가 많다. 이는 인쇄용지 폭의 한계로 데이터를 인쇄용지에 인쇄하여 활용하기보다 마이크로소프트사의 엑셀 등과 같은 외부 소프트웨어를 사용하는 것이 활용의 폭이 넓기 때문이다. 이러한 맥락에서 UI 프레임워크는 그리드 컨트롤의 데이터를 쉽게 다운로드 할 수 있도록 지원하기 위해 Wrapper Function을 구현하였다.

```
<button type="button" class="btn btn-outline-primary btn-xs mt-1"
        onClick="javascript:excelDown('gridId', '파일명');">
  <i class="fas fa-file-excel"></i> 엑셀다운로드
</button>
```

[그림 4-31] 엑셀다운로드 예시 코드

본 화면에서 [그림 4-31]의 예시 코드와 같이 '엑셀다운로드' 버튼을 클릭할 때 excelDown 함수를 그리드 컨트롤의 id와 파일명을 인수로 호출시키면 UI 프레임워크에 구현된 excelDown 함수는 인수로 받은 그리드 ID를 이용하여 컨트롤을 찾고, 해당 그리드(jsGrid)의 헤더 부분과 바디 부분을 찾아 엑셀 형태로 변환하여 다운로드 시킨다. 다운로드 된 파일의 내용은 [그림 4-32]과 같다.

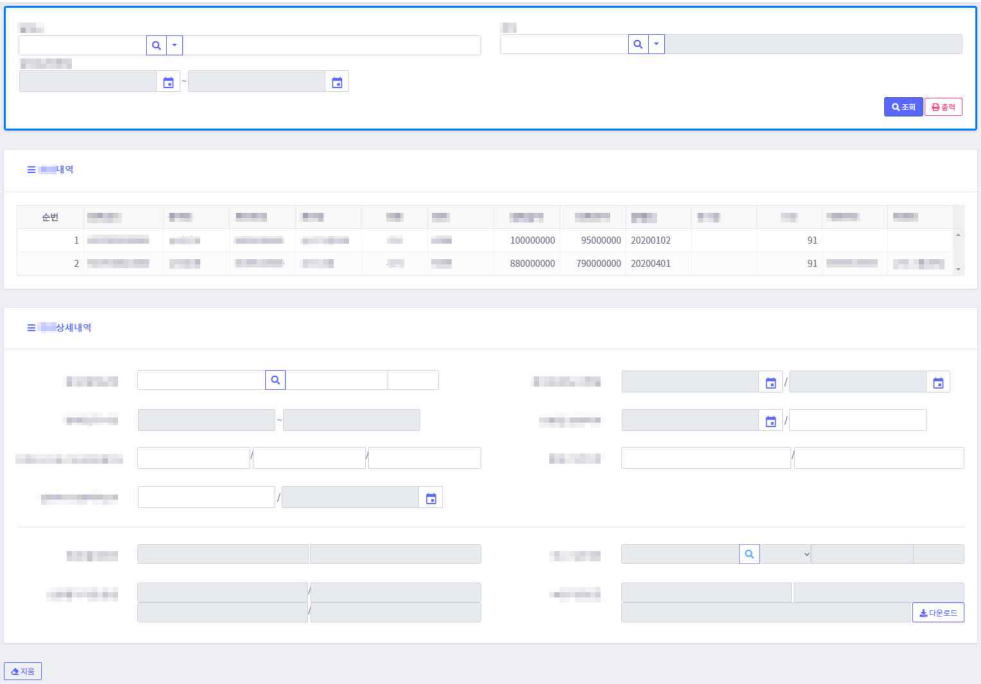
	A	B	C	D	E	F	G	H	I	J
1	순번	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9
2	1	data 1-1	data 1-2	data 1-3	data 1-4	data 1-5	data 1-6	data 1-7	data 1-8	data 1-9
3	2	data 2-1	data 2-2	data 2-3	data 2-4	data 2-5	data 2-6	data 2-7	data 2-8	data 2-9
4	3	data 3-1	data 3-2	data 3-3	data 3-4	data 3-5	data 3-6	data 3-7	data 3-8	data 3-9

[그림 4-32] 다운로드 된 엑셀 파일

이번 연구에서 구현 대상이 되는 예제 화면 중 본 화면에만 엑셀 다운로드 기능을 구현하였으나 UI 프레임워크에서 공통으로 지원하는 것이므로 필요시 어느 화면에서나 사용할 수 있다.

· 화면3

본 화면은 그리드 컨트롤과 일반 컨트롤이 복합적으로 구성되고 연계하는 화면으로 UI 프레임워크를 이용하여 [그림 4-33]과 같이 구현하였다.

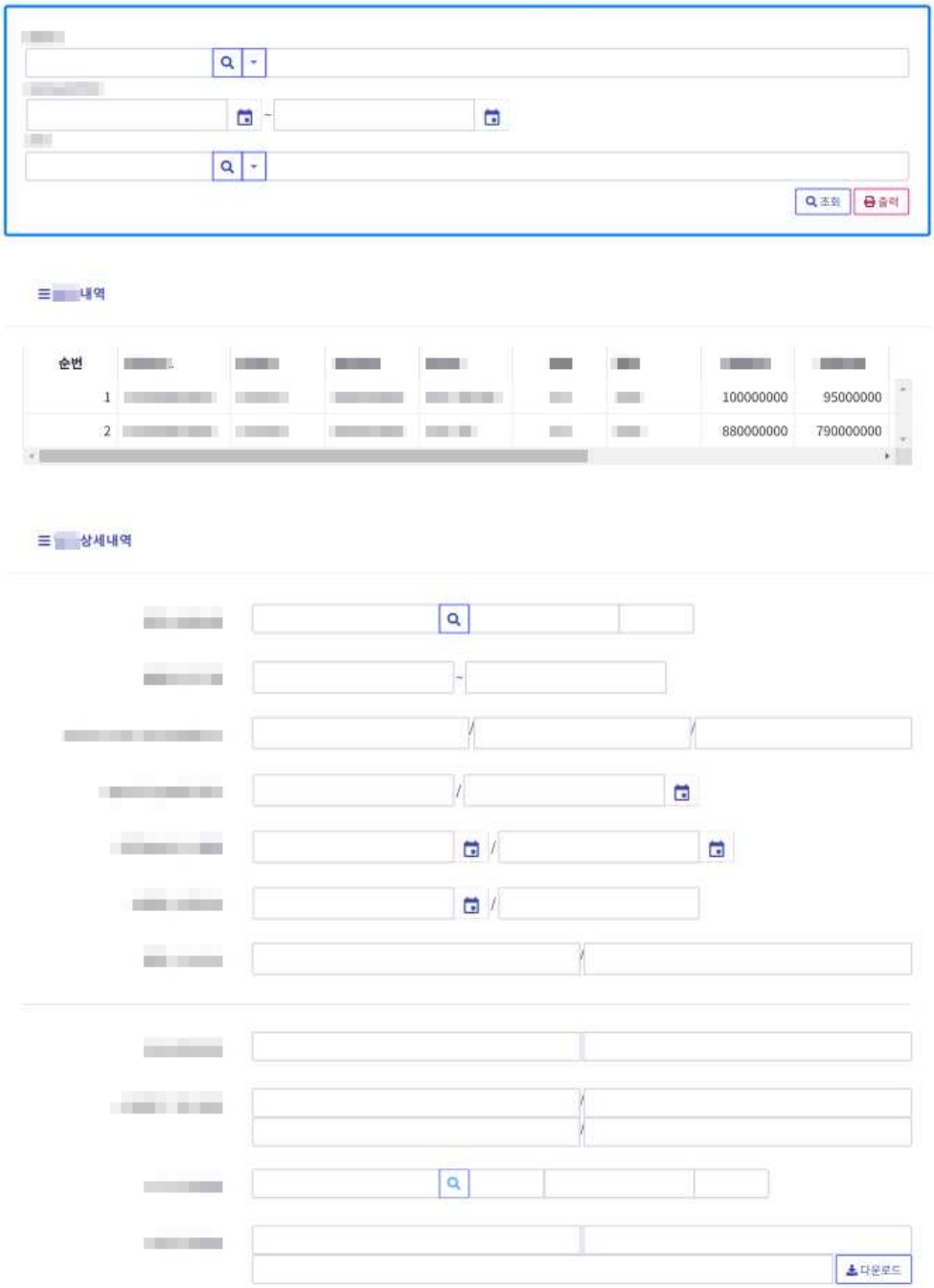


[그림 4-33] 화면3 레이아웃

상단 조회 영역에서 그리드에 조회할 데이터의 조건을 설정하여 조회하면 중간 그리드 컨트롤에 다수의 행을 조회하여 표시하고, 표시된 그리드 컨트롤의 행을 클릭하면 해당 행과 관련된 데이터를 다시 조회하여 일반 컨트롤들로 구성된 하단 상세내역에 표시하는 복합적인 제어를 하도록 구현하였다.

반응형 화면으로 구성하였으며 모바일 환경에서는 [그림 4-34]과 같이 화면 레이아웃이 변화한다. 인쇄(출력) 기능도 앞서 구현한 화면들과 같은 방식으로 반응형 설계에 기반으로 동작하여 [그림 4-35]과 같이 표시된다.

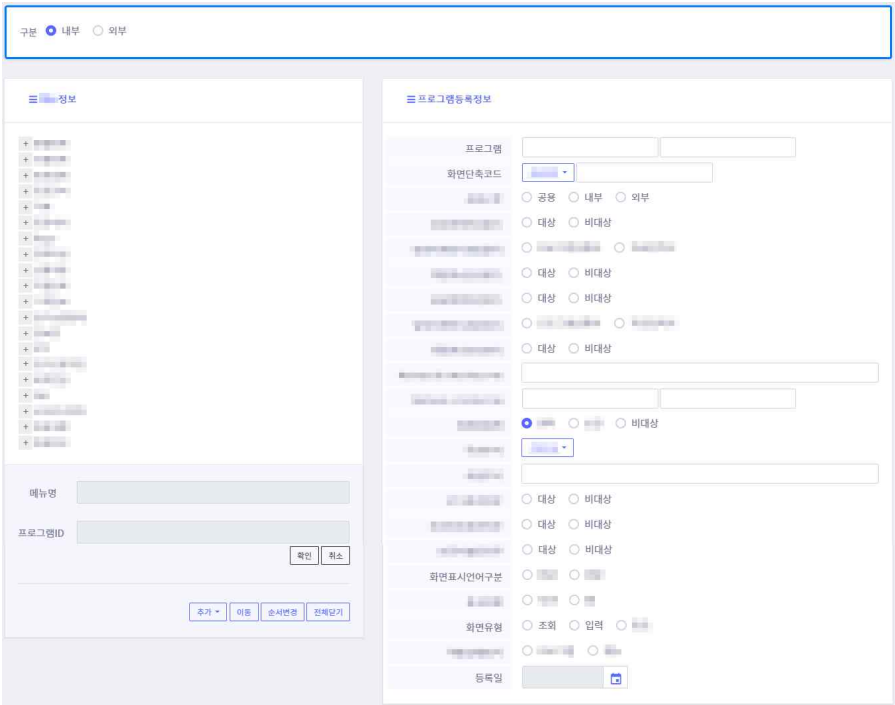
[그림 4-34] 화면3 모바일 레이아웃



[그림 4-35] 화면3 인쇄 레이아웃

· **화면4**

현행 화면과 동일하게 트리 컨트롤을 좌측에 표시하고, 선택된 트리 노드의 상세 정보가 우측에 배치되는 구조를 갖는 화면으로 구현하였다. UI 프레임워크를 이용하여 구현한 화면은 [그림 4-36]과 같다.



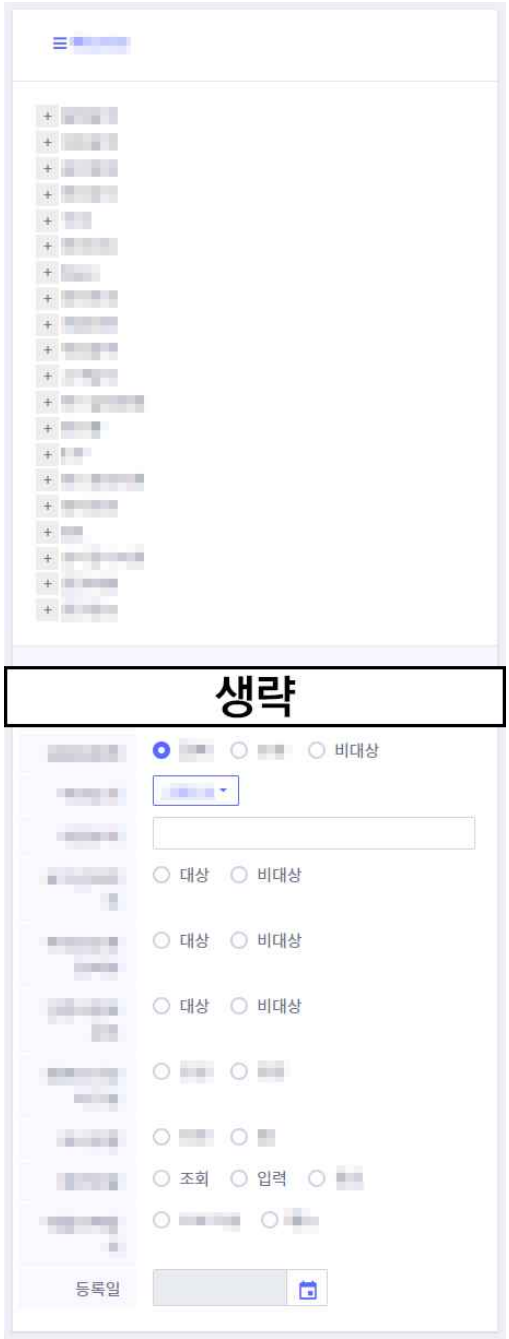
[그림 4-36] 화면4 레이아웃

트리 데이터는 [그림 4-37]과 같이 각 노드에 고유번호를 부여하고 상위 노드가 존재하는 경우 상위 노드의 번호를 이용하여 관계를 설정하였다. 이는 현행 금융기관 A사에서 노드의 상·하위 연결을 하는 구조와 동일하게 설계한 것으로 차후 실제 적용에 용이하도록 하였다.

```
▶0: {Seq: 1, parent Seq: null, Nm: "", roleSelectionTpcd: "2"}
▶1: {Seq: 2, parent Seq: 1, Nm: "", roleSelectionTpcd: "1"}
▶2: {Seq: 3, parent Seq: 1, Nm: "", roleSelectionTpcd: "1"}
▶3: {Seq: 4, parent Seq: null, Nm: "", roleSelectionTpcd: "2"}
▶4: {Seq: 5, parent Seq: 4, Nm: "", roleSelectionTpcd: "1"}
▶5: {Seq: 6, parent Seq: 4, Nm: "", roleSelectionTpcd: "1"}
▶6: {Seq: 7, parent Seq: null, Nm: "", roleSelectionTpcd: "2"}
▶7: {Seq: 8, parent Seq: 7, Nm: "", roleSelectionTpcd: "1"}
▶8: {Seq: 9, parent Seq: 7, Nm: "", roleSelectionTpcd: "1"}
```

[그림 4-37] 화면4 트리 데이터의 구조

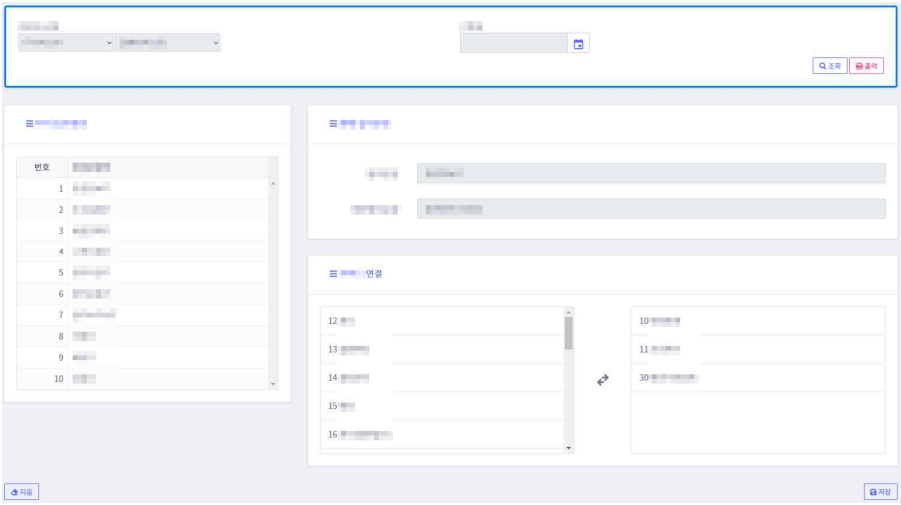
모바일 환경에서 반응형으로 동작하는 화면은 [그림 4-38]와 같다.



[그림 4-38] 화면4 모바일 레이아웃

· 화면5

본 화면은 1:N의 관계를 가지는 데이터를 관리하기 위한 화면으로 현행 그리드만으로 구성된 화면을 개선하여 그리드와 멀티셀렉트 컨트롤을 조합하여 구현하였다. 구현된 화면은 [그림 4-39]과 같다.



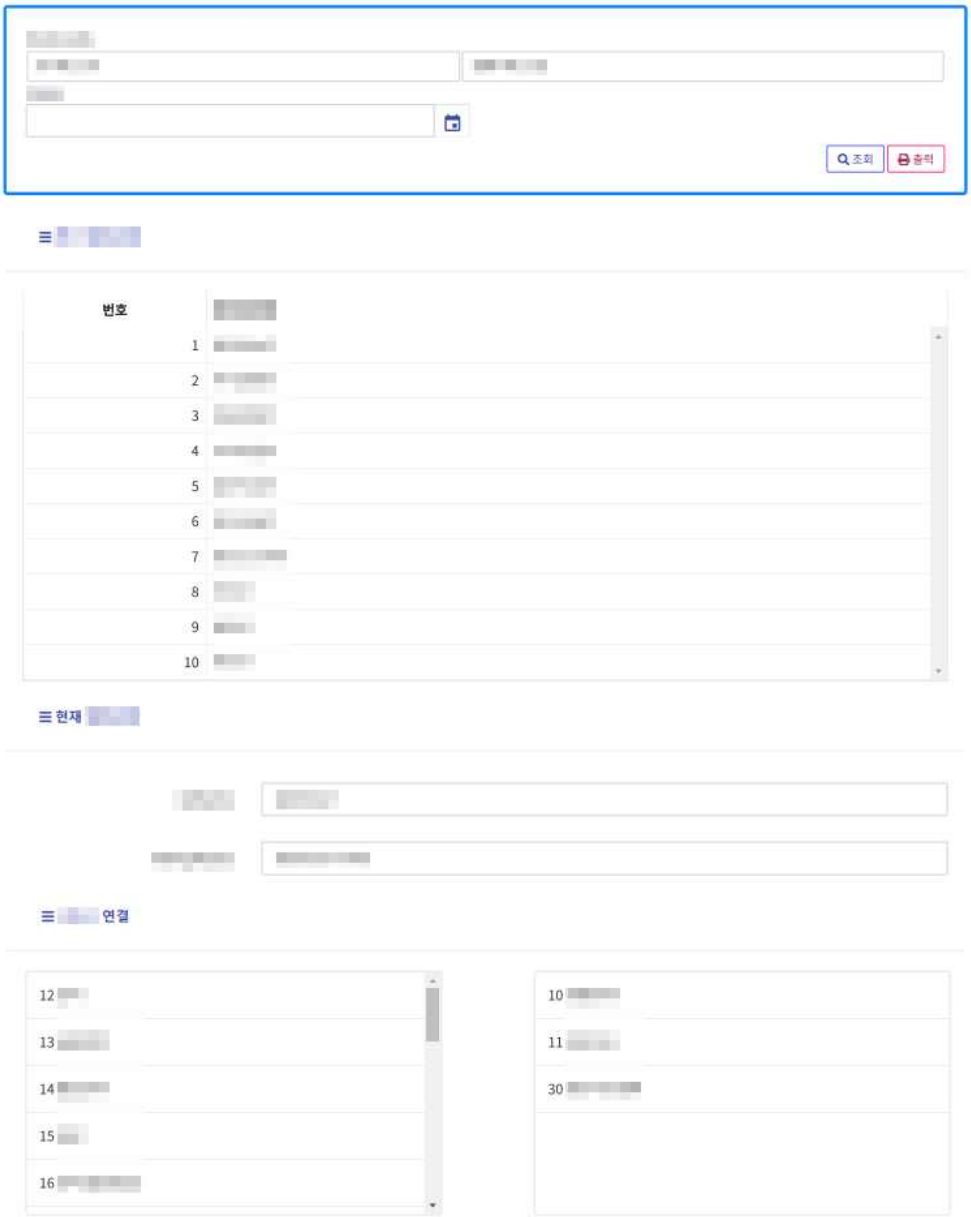
[그림 4-39] 화면5 레이아웃

좌측 그리드를 클릭하면 우측 하단 멀티셀렉트에 하위 항목들이 표시된다. 그리드에 선택한 행과 연결되지 않은 하위 항목은 좌측에, 연결된 하위 항목은 우측에 표시된다. 하위 항목들은 클릭을 통해 연결을 해제하거나 다시 연결할 수 있다.

현행 화면의 경우 하위 항목을 연결하는데 그리드 컨트롤을 이용하였지만 본 연구에서는 멀티셀렉트 컨트롤을 사용하여 구현하였다. 멀티셀렉트 컨트롤은 화면에서는 두 개의 리스트로 표시되지만, 내부적으로는 1개의 컨트롤을 사용하는 것이므로 구현 및 유지보수 편의성이 증가할 것이다.

모바일 환경에서 변화되는 레이아웃은 [그림 4-40]과 같고, 인쇄(출력) 기능도 앞서 구현한 화면들과 같은 방식으로 반응형 디자인에 기반하여 동작하며 [그림 4-41]과 같이 표시된다.

[그림 4-40] 화면5 모바일 레이아웃



[그림 4-41] 화면5 인쇄 레이아웃

제 5 장 결론

제 1 절 요약 및 결론

본 연구는 특정 운영체제에서 상용 소프트웨어를 기반으로 운영 중인 금융기관 A사 기간계 시스템을 여러 가지 운영체제와 다양한 화면의 해상도를 가지는 장치를 지원하고, 사용자 편의성과 범용성을 극대화하며, 상용 소프트웨어의 유지보수료를 절감하고자 오픈 소스 소프트웨어를 사용한 웹 기반으로 전환 할 수 있는지 설계하고 구현하여 검증하고자 하였다.

여러 운영체제의 사용자를 지원하기 위하여 웹을 기반으로 기술 구조를 설계하였고, 다양한 해상도의 장치를 지원하기 위해 반응형으로 동작하는 라이브러리를 활용하였다. 또한 비용 절감 목표를 달성하기 위해 오픈 소스를 라이브러리를 이용하였다.

연구의 범위를 확정하기 위해 현행 시스템을 분석하여 '공통 영역'과 '업무 영역'으로 나누고, 자주 사용되는 형태의 업무 화면 5본을 '업무 영역'의 요구사항으로 정의하였다. 또한 선정된 업무 화면들을 구현하기 위해 필요한 공통 기능들을 '공통 영역'의 요구사항으로 정의하였다.

요구사항들을 구현하기에 앞서 효율적으로 업무 시스템을 구현하기 위하여 요구사항에 맞는 UI 프레임워크를 설계하였다. UI 프레임워크는 표준 HTML과 CSS를 기본으로 JavaScript와 jQuery를 이용하여 브라우저의 호환성 문제를 해결하였다. 또 업무 화면이 반응형으로 동작할 수 있도록 Bootstrap의 Grid System을 활용하였다. 그리고 업무 화면의 특성에 따라 jsGrid, Simple Tree, Multi-select, FullCalendar, C3Chart와 같은 오픈 소스 라이브러리를 UI 프레임워크에 추가하였다.

마지막으로 UI 프레임워크를 이용하여 5본의 업무 화면을 설계 및 구현하였으며 다양한 운영체제와 장치에서 정상적으로 동작하는지 검증하여 본 연구의 목표를 달성하였음을 확인하였다.

제 2 절 연구성과 및 기대효과

본 연구에서 오픈 소스를 이용한 웹 기반 UI 프레임워크를 설계 및 구현하였고 해당 프레임워크를 기반으로 구현한 업무 화면들이 현행 업무 화면들과 기능적으로 동일하게 동작하는 것을 확인하였다.

본 연구의 결과는 앞서 목표한 바와 같이 Cross Platform, Cross Browser, Responsive Web Design의 요건을 부합하도록 구현하였으므로 다양한 장치와 운영체제, 브라우저를 이용하는 사용자가 시스템을 이용할 수 있어 범용성과 사용자의 편의성이 기존 시스템 기술구조 대비 비약적으로 증대되는 효과가 있다.

또한 UI 프레임워크에 포함된 오픈 소스 소프트웨어들은 복제, 배포, 수정이 자유롭고 오픈 소스 소프트웨어를 사용하여 제작한 소프트웨어의 소스코드 공개 의무, 동일한 라이선스로 재배포 의무가 없는 오픈 소스 소프트웨어만을 사용하였으므로 차후 실제 시스템에 활용될 때 별도의 비용이 발생하지 않아 시스템 유지보수 예산 절감 효과를 기대할 수 있을 것이다.

더불어 금융기관 A사의 기간업무 시스템이 아닌 타 시스템 또는 신규 시스템 구축 시에 본 연구에서 제시하는 웹 기반 UI 프레임워크를 사용하여 구축한다면 범용성과 사용자 편의성이 높은 시스템을 만들 수 있으면서도 시스템 구축에 필요한 예산이나 비용을 절감 할 수 있을 것으로 기대된다.

제 3 절 고찰

본 연구에서 이용한 웹 기반 기술은 여러 운영체제와 장치에서 사용할 수 있어 범용성이 기존대비 탁월하다는 장점이 있다. 또한 오픈 소스 소프트웨어는 확장이 용이하고 비용절감이 가능하다는 장점을 가진다.

그러나 이러한 장점들은 보안 측면에서 Trade-off가 발생한다. 다양한 장치를 지원하는 만큼 여러 장치에서 각각 발생할 수 있는 취약점에 대비해야 할 수도 있으며, 소스코드가 공개되어 있는 오픈 소스의 소프트웨어의 특성상 보안 취악점에 따른 위협에 노출될 수 있다는 단점이 있다.

그러므로 본 연구에서 설계하고 구현된 결과를 차후 실제 적용하기 전에 보안 취약점 분석을 통해 문제점을 분석하고 보완해야하며, 필요한 경우 추가적인 개발을 해야 할 가능성도 존재한다.

금융기관 A사는 In-House IT 방식으로 시스템을 유지보수 하고 있는데, 오픈 소스 소프트웨어를 도입하게 되면서 상용 소프트웨어를 사용하던 기존의 업무 환경 대비 유지보수를 담당하는 내부 인력이 필요하게 되고, 이는 내부 인력에게 추가적인 업무가 발생하기 때문에 인력 충원이 필요할 수 있다.

하지만 위와 같은 단점에도 불구하고 다양화되는 사용자 환경을 지원하기 위해서는 이러한 크로스 플랫폼 UI는 피할 수 없을 것이기 때문에 웹 기반으로의 전환이 불가피 할 것으로 생각된다. 또한, 금융 인프라를 제공하는 해외 회사들이 IT인력을 계속 보강하면서 IT회사처럼 변화하는 추세로 볼 때, 상용 소프트웨어나 외부 인력에 의한 시스템 유지보수보다는 내부의 시스템 유지보수 전문가를 계속 양성하는 것이 올바른 방향일 것이다.

제 4 절 향후 계획

본 연구에서는 시간 및 자원적 한계로 인하여 5본의 업무 화면과 해당 화면들의 구현에 필요한 공통 기능에 한하여 연구하였다. 이를 실제 업무 환경에서 적용하기 위해서는 본 연구를 바탕으로 시스템 전반에 걸친 검토와 UI 프레임워크 확장이 필요하다.

첫째, 실제 업무 환경에서 사용되는 모든 업무 화면에 대하여 본 연구에서 설계한 웹 기반 UI 프레임워크에 대하여 전수 검토가 필요하다. 당연하게도 본 연구에서 구현한 UI 프레임워크는 한정적인 부분에 대한 선행적 연구이므로, 실제 사용 중인 모든 업무 화면을 지원할 수 없다. 그러므로 향후 실제 적용을 위해서는 업무에 사용되는 모든 화면을 대상으로 적용에 필요한 기능을 취합하고 분석하여 UI 프레임워크 확장을 위한 추가적인 오픈 소스를 도입을 검토하고 필요에 따라 직접 구현해야 한다.

둘째, 공통 기능에 대한 추가적인 분석이 필요하다. 업무 화면과 마찬가지로 현재 운영 중인 시스템의 모든 공통 기능이 본 연구에서 설계한 UI 프레임워크가 지원하는 것은 아니므로 실제 운영 중인 시스템 공통 기능에 대한 모든 기능 리스트 취합과 웹 기반 전환 가능 여부 등의 검토가 선행되어야 할 것이다.

셋째, 법률 또는 규정 등 행정적인 검토가 필요하다. 금융기관 A사는 자본시장과 금융투자업에 관한 법률, 전자금융감독규정 등 여러 가지 법률과 규정을 준수해야 한다. UI 프레임워크를 실제 업무에 적용할 때 금융기관 A사가 준수해야 할 다양한 법률과 규정에 위배되는 사항이 없는지 등 다각도의 검토와 보완이 필요할 것으로 생각된다.

위 세 가지 사항에 대하여 향후 추가적인 연구와 검토를 진행할 예정이다.

참 고 문 헌

[1] 한세희 (2012). 우리나라 최초 인터넷 연결<1982년 5월>. 전자신문.

[2] statcounter. Operating System Market Share Worldwide Jan 2009 to June 2020. https://gs.statcounter.com/os-market-share#monthly-200901-202006.

[3] statcounter. Browser Market Share Worldwide Jan 2009 to June 2020. https://gs.statcounter.com/browser-market-share#monthly-200901-202006.

[4] 이상일 (2016). '오픈소스' 받아들이는 금융권, 대세인가? 디지털데일리.

[5] 함지현 (2019). 신한銀, 리눅스 전환 박차...금융시장에 부는 리눅스 물결. 전자신문.

[6] 장병화 (2015). 전자금융총람. 한국은행.

[7] 최은정 (2019). 인스웨이브, KB국민카드 차세대시스템에 웹표준 금융계정계 적용. 아이뉴스24.

[8] 최우용 (2017). 금융업무 단말, 웹표준으로 다 된다. 지디넷코리아.

[9] 정보통신산업진흥원, 공개SW 라이선스, https://www.oss.kr/oss_license

[10] 한국저작권위원회, 라이선스비교표, https://www.olis.or.kr/license/compareGuide.do

[11] World Wide WebConsortium. Mobile Web Best Practices 1.0 Basic Guidelines. http://www.w3.org/TR/2008/REC-mobile-bp-20080729/.

[12] Marcotte, E. (2017). Responsive web design: A book apart n 4. Editions Eyrolles.

[13] Otto, M., & Thornton, J. (2011). Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web. Bootstrap.

[14] Temere, B. (2017). Responsive web application using Bootstrap and Foundation: Comparing Bootstrap and Foundation Frontend Fr

ameworks.
[15] Keith, J. (2005). A Brief History of JavaScript. DOM Scripting: W eb Design with JavaScript and the Document Object Model, 3-10.
[16] Garrett, J. J. (2005). Ajax: A new approach to web applications.
[17] Resig, J. (2006). jQuery: New Wave Javascript.
[18] Li, J., & Peng, C. (2012, June). jQuery-based Ajax general intera ctive architecture. In 2012 IEEE International Conference on Com puter Science and Automation Engineering (pp. 304-306). IEEE.
[19] Tabalin A. (2014). Lightweight Grid jQuery Plugin. http://js-grid. com/.
[20] Khowfa, W., & Silasai, O. (2019). The Efficiency of using Salt Agai nst Password Attacking. JOURNAL OF SOUTHERN TECHNOLOG Y, 12(1), 217-227. Retrieved from https://so04.tci-thaijo.org/index.ph p/journal_sct/article/view/103246
[21] Statistics, U. Market share trends for JavaScript libraries. W3Tec hs-Web Technology Survey. https://w3techs.com/technologies/hist ory_overview/javascript_library.
[22] Manual, P. H. P. (2009). A History of PHP.
[23] Colorlib. Concept - Bootstrap 4 Admin Dashboard Template. http s://colorlib.com/polygon/concept/index.html
[24] jqueryscript. Semantic Tree View Plugin With jQuery - Simple Tree. https://www.jqueryscript.net/other/semantic-tree-view.html
[25] bootstrap. Grid system. https://getbootstrap.com/docs/4.5/layout/grid/

Abstract

Web UI Framework Design Using Open Source

Hanseung Kim
Department of Engineering Practice
Graduate School of Engineering Practice
Seoul National University

Internet services have increased exponentially over the past 20 years due to the development of Internet infrastructure. In addition, with the advent of mobile devices, the mobile environment using the Internet has exploded. With the growth of the mobile environment, devices with various operating systems and resolutions have become common, and for this reason, existing programs and services are steadily changing to support the mobile environment for user convenience.

On the other hand, the core business system of the company A is provided by converting the business system based on the Internet environment in accordance with the above trend, but system users still have to use a specific operating system and personal computer.

Therefore, in this study, a responsive web-based UI framework was designed to support various devices and operating systems for A company's core business system. In addition, the library or

component that should be included in the UI framework uses open source software to reduce purchase cost and maintenance costs, and to make customization easier.

Finally, using the UI framework designed in this study, 5 work screens in actual operation were selected and implemented. Through this, it was verified that the functions of the work screen operate in the same manner as the current system in various operating systems and mobile devices.

keywords : User Interface, Responsive Design, Web, Framework, Open Source
Student Number : 2019-21204