CHAPTER

2

현대 웹 시스템 구조 및 아키텍처

본격적인 API 개발에 들어가기에 앞서, 웹 시스템들이 시스템 구조적으로 어떻게 발전하게 되었는가와 현대 웹 시스템의 구조와 아키텍처는 어떻게 형성되어 있는가에 대해서 간략하게 알아보도록 하자. 이러한 역사와 배경 지식을 이해하게 되면 왜 API 백엔드 개발이 필요한지, 그리고 백엔드 개발자들은 어떠한 역할을 하며 어떠한 기술이나 능력이 요구되는지에 대해서 더 잘 이해할 수 있게 된다.

또한 현대의 개발팀의 구조, 즉 어떠한 직군들이 있고, 어떠한 역할들이 있는지에 대해서도 알아보도록 하겠다. 그럼으로써 독자들이 실제 개발자로서 실무를 하게 될 때 팀에서 어떠한 역할을 담당하게 될지 더 이해할 수 있고, 또한 앞으로 발전해 나아가는 목표를 정하는 데 도움이 될 것이다.

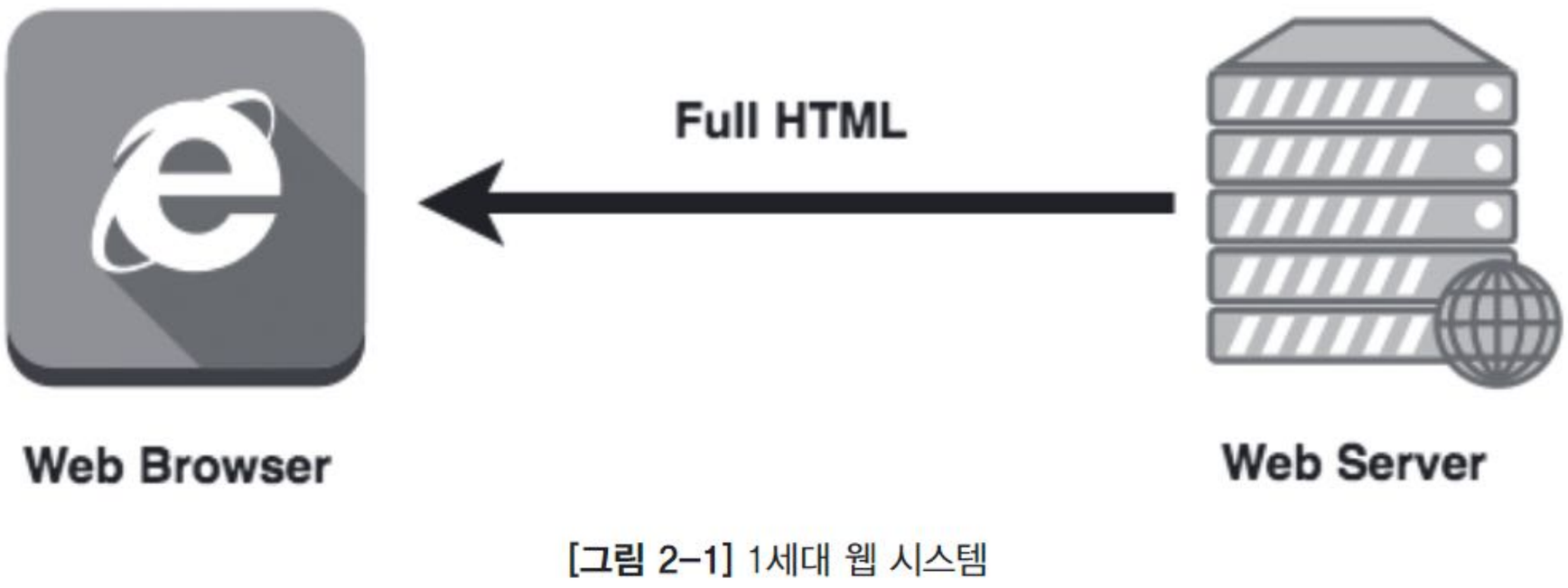
- 웹 시스템들의 발전 역사
- 현대의 웹 시스템 들의 구조 및 아키텍처
- 현대의 개발팀의 구조

웹 시스템들의 발전 역사

초기의 웹 시스템

1989년에 팀 버너스 리(Tim Berners-Lee)가 월드와이드웹(WWW, World Wide Web)을 발명하고 난 후로 많은 서비스들이 웹을 통해 제공되었다. 특히 2000년대 닷컴붐(Dotcom Boom) 때에 엄청나게 많은 웹 서비스 회사들이 생기면서 웹 시스템들과 웹 관련 기술들이 엄청난 발전을 하였다.

웹이 발명되고 난 후 초기 1세대 웹 시스템들은 지금의 웹 시스템들보다 훨씬 단순한 형태였다. 당시에는 지금처럼 다양하고 복잡한 서비스를 웹을 통해서 제공하는 것이 아니라 정말 단순하게 문서(text)를 웹 브라우저를 통해 보는 수준이었다.



[그림 2-1] 1세대 웹 시스템

그러므로 웹 서버는 단순히 웹 브라우저가 요청하는 해당 페이지를 보내 주는 정도의 기능만 했다. 예를 들어, /news.html 페이지를 웹 브라우저가 요청하면 해당

HTML 페이지를 보내 준다. 그러면 웹 브라우저는 웹 서버에서 전달받은 HTML 파일을 렌더링하여 사용자에게 보여 준다. 간단하다는 장점은 있으나 굉장히 정적인(static) 페이지만 보여 줄 수밖에 없는 구조였다.

[그림 2-2] 초기의 애플(Apple) 사이트

자바스크립트의 역할이 커지기 시작함

단순했던 웹 서비스들이 조금씩 복잡해지기 시작하면서 사용자 인터랙션(user interaction)이 중요해지기 시작한다. 즉 정적인 데이터나 문서만을 보여 주지 않고 웹 페이지에서 사용자와 동적인 상호작용이 중요해지기 시작한 것이다. 이러한 부분을 구현하기 위해서 자바스크립트의 역할이 커지기 시작한다. 자바스크립트는 웹 브라우저에서 실행이 가능한 프로그래밍 언어다. 주로 웹상에서 동적인 기능을 제공하기 위해 사용된다. 많은 웹사이트들이 단순히 HTML을 통해 정적인 문서를 보여 주는 것을 넘어서 자바스크립트를 통해 동적인 기능들을 제공하기 시작했다.

[그림 2-3] 자바스크립트의 역할 증가

웹 서버가 HTML뿐만 아니라 동적인 기능을 구현하는 자바스크립트 코드까지 같이 웹 브라우저에 전송하면 웹 브라우저는 서버에서 전달받은 자바스크립트 코드를 실행해서 동적인 기능들이 사용자에게 제공되도록 한다. 그러므로 전체 페이지를 로드(load)하지 않아도 사용자의 인풋(input)들을 동적으로 처리하고 새로운 데이터들을 제공할 수 있게 된 것이다. 초기에는 이렇게 자바스크립트를 통해서 사용자와 동적인 상호작용 기능을 구현하는 기술이 AJAX라는 이름으로 알려졌었다.

점점 웹 서비스가 정적에서 동적으로 변화되어 가고 HTML보다 자바스크립트의 역할이 점점 더 커져 갔지만, 아직도 현재 통용되는 API 개념이 널리 사용되고 있지는 않았다. 여전히 동일한 서버에서 HTML, 자바스크립트, 그리고 데이터가 전부 웹 브라우저 등의 클라이언트로 전송되고 있었다. 그리고 데이터는 XML의 구조로 전송되는 것이 일반적인 시대였다.

XML은 데이터를 전송하기 위한 markup 언어다. 구조는 HTML과 비슷하나 데이터를 표현하기 위해 사용된다.

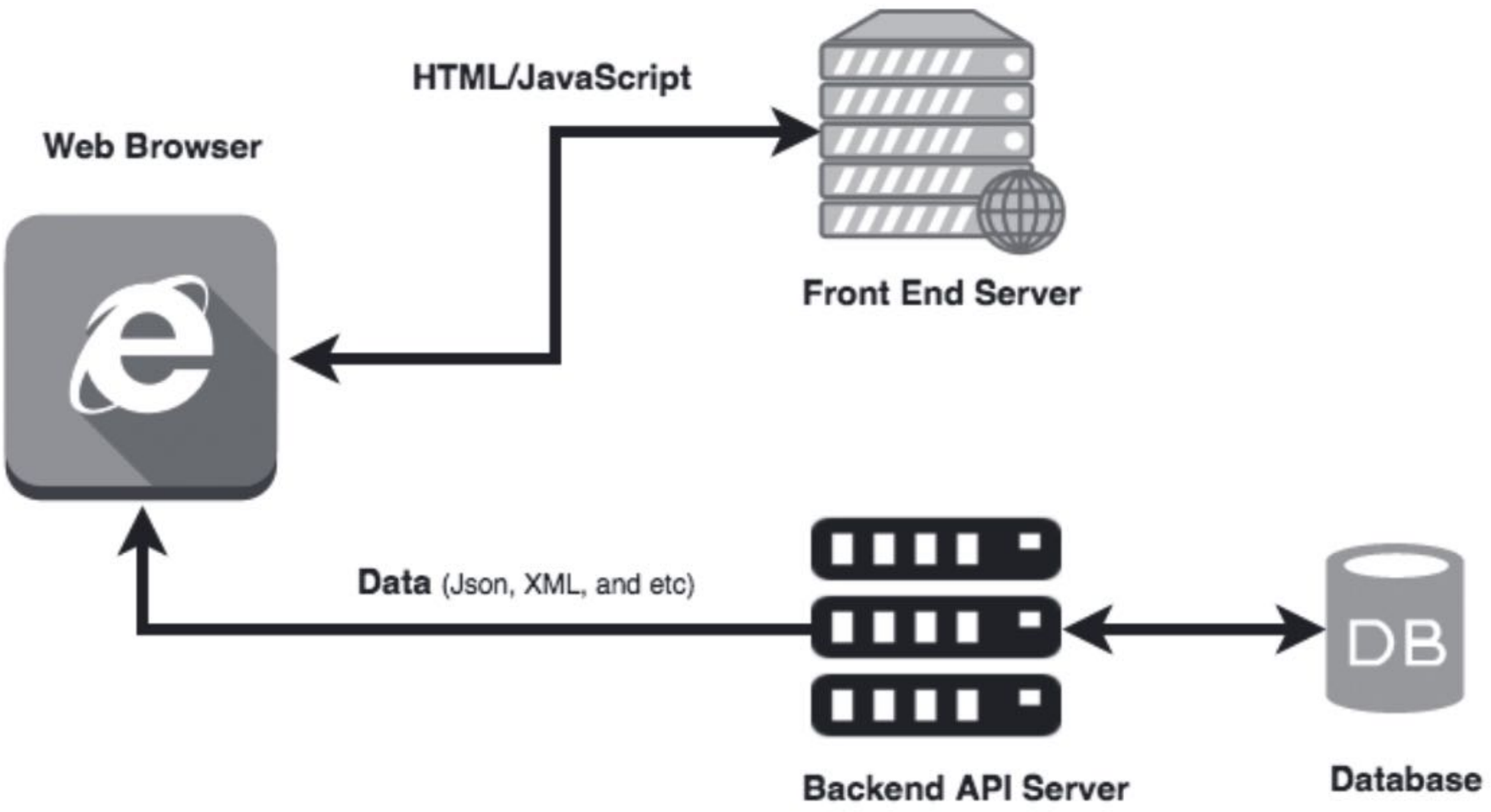
구별되기 시작하는 프론트엔드와 백엔드

동적인 서비스가 중요시되고 많은 웹 서비스들이 발전하면서 자바스크립트의 역할이 훨씬 커지게 되면서 급기야는 HTML도 자바스크립트 코드로 직접 생성하게 되는

방식이 사용되기 시작했다. 그 전에는 자바스크립트가 HTML 파일에 속한 일부분에 지나지 않았다면 이제는 자바스크립트가 주가 되어서 HTML 생성부터 사이트(프론트엔드(frontend))에 관한 모든 부분을 구현하게 되었다. 그러므로 전에는 부분적으로만 동적이었던 웹사이트나 서비스들이 이제는 전체적으로 동적이 된 것이다.

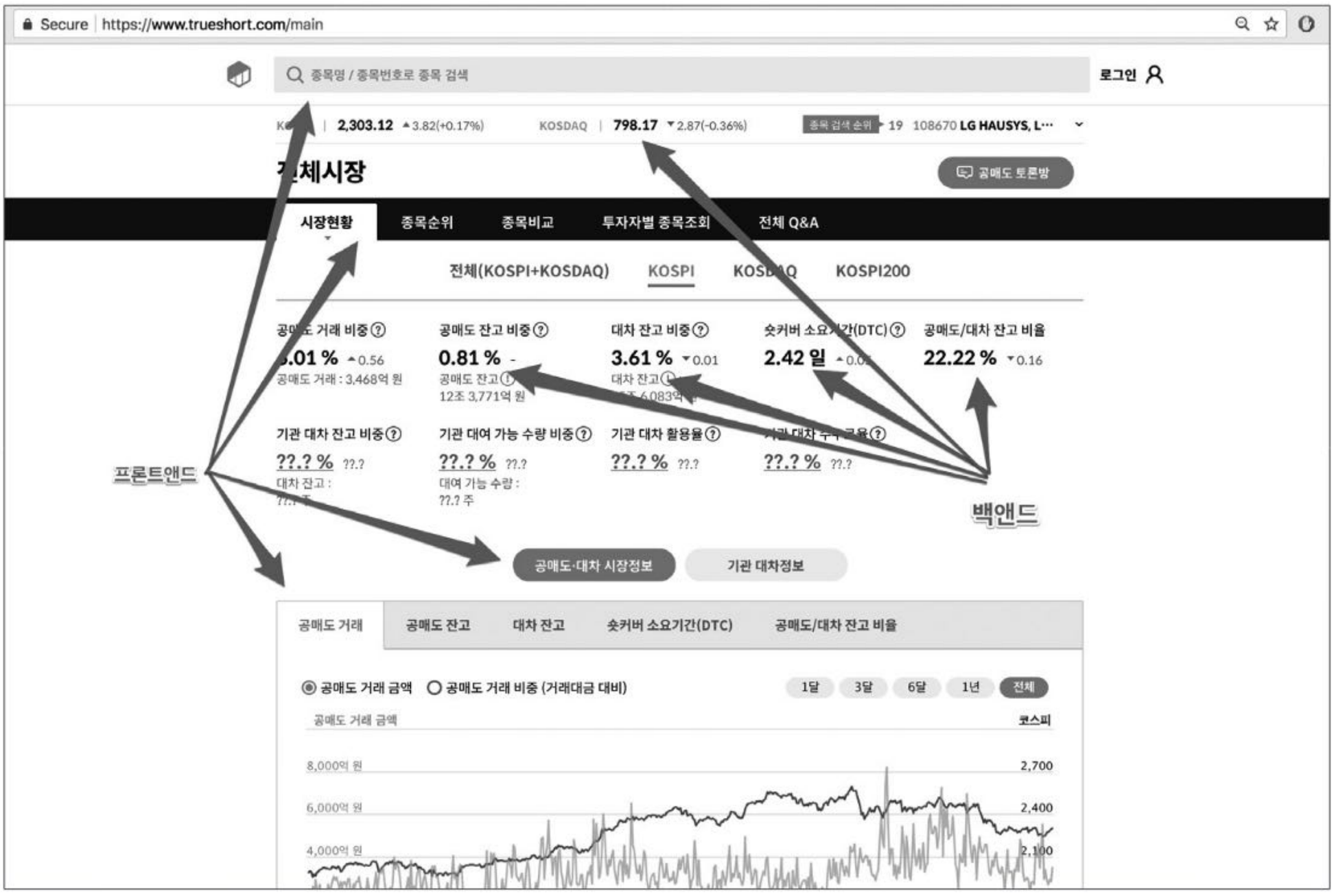
자바스크립트가 주가 되면서 SPA(Single Page Application) 방식의 프론트엔드 개발이 인기를 얻게 되었다. SPA는 이름 그대로 단일 페이지로 모든 웹사이트/서비스의 기능을 구현하는 것이다. 페이지는 아주 단순하게 기본 HTML 구조에 메인 자바스크립트 파일이 포함되어 있는데, 이 메인 자바스크립트가 모든 페이지 및 기능들을 동적으로 구현하게 되는 것이다.

이렇게 단일 페이지의 자바스크립트를 통해 모든 페이지를 동적으로 구현하게 되다 보니 자연스럽게 웹 브라우저가 필요한 서버와의 통신은 데이터 전송이나 생성 및 수정에 관한 것이 대부분이 된다. 사이트의 페이지를 렌더링(rendering)하는 데 필요한 자바스크립트 코드는 최초의 통신에서 모두 한 번에 받으므로 그다음부터는 서버와 데이터만 주고받으면 된다.



[그림 2-4] 프론트엔드와 백엔드 분리 구조

그렇다 보니 자연스럽게 프론트엔드 서버와 백엔드 서버가 나뉘게 된다. 프론트엔드 서버는 페이지 렌더링에 필요한 HTML과 자바스크립트 파일을 전송하는 역할을 담당하고, 백엔드 서버는 페이지에서 필요한 데이터 생성 및 전송을 담당하는 역할을 하게 된다.



[그림 2-5] 트루쇼트 사이트

저자가 공동 창업한 공매도 플랫폼인 트루쇼트(True short) 사이트(그림 2-5)를 예를 들어 보자. 트루쇼트 사이트의 메뉴, 검색 창, 버튼, 전체적인 페이지 구조, 차트 등은 전부 프론트엔드 서버에서 받은 HTML과 자바스크립트 코드를 통해서 렌더링 된 것이다. 이러한 메뉴, 버튼, 링크 등은 실시간으로 바뀌는 것이 아니므로 첫 접속 때 프론트엔드 서버에게 받은 HTML과 자바스크립트 코드를 실행함으로써 프론트엔드 서버와 지속적인 통신 없이도 페이지가 구현 가능하다.

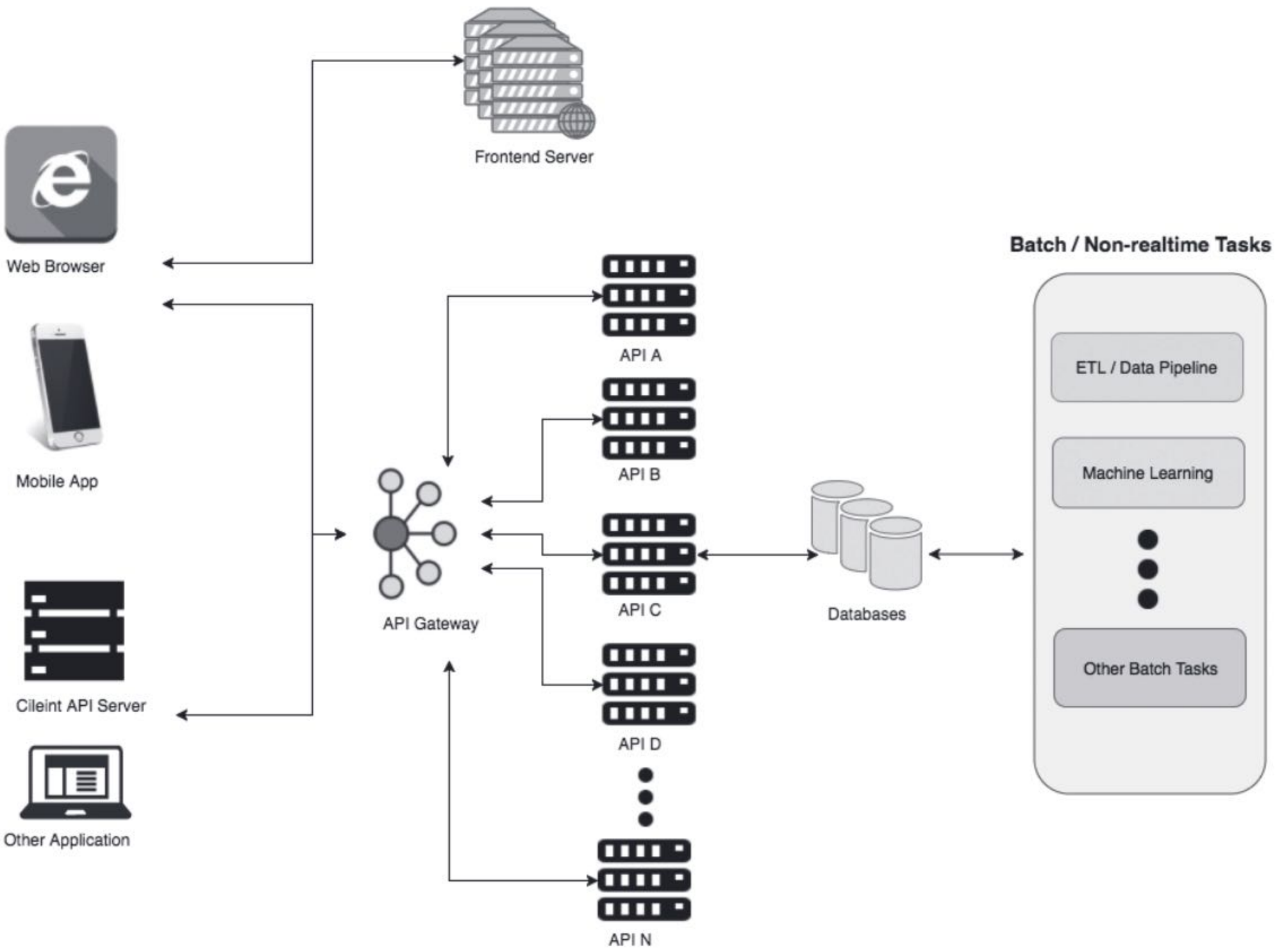
그에 반해, 실제 데이터들, 예를 들어 "공매도 잔고 비중" 수치, 코스피와 코스닥 값

들 등 실제 데이터들은 매번 업데이트되므로 지속적으로 백엔드 서버와의 통신을 통해 데이터들을 업데이트해서 보여 주게 된다.

이렇게 시스템적으로 프론트엔드 서버와 백엔드 서버의 기능이 명확히 구분되면서 그에 따라 프론트엔드 개발자와 프론트엔드 서버 개발자의 역할과 요구되는 기술들 또한 구분이 된다. 프론트엔드 개발자는 주로 시스템의 UI(User Interface)와 UX(User Experience) 부분을 구현하는 역할을 담당한다. 웹 시스템의 경우 최근에 프론트엔드 개발자들이 주로 사용하는 기술은 HTML과 CSS 그리고 자바스크립트다. 또한 ReactJS, AngularJS, Webpack, npm, 그리고 그 외 여러 자바스크립트 프레임워크 및 라이브러리가 사용된다.

반면 백엔드 API 개발자의 역할은 프론트엔드 시스템(혹은 다른 클라이언트 시스템)과 데이터를 실시간으로 주고받을 수 있는 기능을 구현하는 역할을 담당한다. 특히 많은 수의 동시 요청을 장애 없이 실시간으로, 그리고 최대한 빠른 속도로 처리할 수 있는 시스템을 구현하는 것이 백엔드 개발자의 중요한 역할이 된다. 그러므로 백엔드 개발에 주로 사용되는 언어도 안정적이고 확장성이 높으며 실행 속도도 높은 시스템을 구현하기에 유리한 언어가 주로 사용된다. 대표적인 언어들로는 Java, Scala, Rust, Python 등이 있다. 처리 용량이나 속도가 크게 중요하지 않은 백엔드 시스템을 구현할 때는 개발의 속도와 편리함이 장점인 언어들이 사용되기도 한다. 그러한 언어들은 Ruby, PHP 등이 있다. 최근에는 NodeJS라는 자바스크립트 엔진이 인기를 얻으면서 자바스크립트로 구현된 백엔드 시스템도 늘어나는 추세다. 자바스크립트를 사용해서 백엔드 시스템을 구현하면 가장 큰 장점 중 하나는 자바스크립트를 사용해서 프론트엔드와 백엔드 둘 다 구현할 수 있다는 점이다.

현대 웹 시스템들의 구조 및 아키텍처



[그림 2-6] 현대 웹 시스템 아키텍처

현대에 들어와서는 웹 시스템의 규모가 더욱더 커지고 무엇보다 처리해야 하는 동시 요청 수와 데이터의 규모가 기하학적으로 증가하면서 웹 시스템들의 구조 또한 더욱 방대해지고 복잡해지게 된다. API 시스템들이 처리해야 하는 동시 요청 수가 기하급수적으로 늘어나고, 또한 API 시스템들이 너무 방대해지고 복잡해지는 문제를 해결하기 위해서 MSA(Micro Service Architecture) 같은 새로운 아키텍처 개념이 발전됨으로써 API 서버들이 훨씬 더 세분화되며 규모가 커지게 된다.

또한 분석해야 하는 데이터의 양이 엄청나게 늘어나면서 ETL(Extract, Transfer, Load) 혹은 Data Pipeline 시스템이 발전하게 되며, Hadoop 등 대용량 분석 프레임워크 등의 발달로 이른바 "Big Data" 분석 시스템이 많은 회사들의 백엔드 시스템

에 도입된다. 게다가 최근에는 ML(Machine Learning)과 AI 기술의 발달로 많은 회사들이 ML과 AI 시스템도 활용하게 됨으로써 백엔드 시스템의 영역은 계속해서 방대해져 가는 추세다.

그러므로 현대의 백엔드 개발자의 영역은 일반적인 백엔드 API 시스템 개발부터 Data Pipeline 시스템, Machine Learning 시스템, Big Data 분석 시스템 등 비실시간(혹은 준실시간) 대규모 데이터 수집 및 분석 시스템, 그리고 Machine Learning 시스템까지 넓어지고 있다.

백엔드 개발자로 입문하게 되면 주로 API 시스템 구현부터 시작하게 된다. 그러면서 경력과 경험이 높아질수록 대용량 데이터 수집 및 분석 시스템 구현의 역할을 담당하게 되는 것이 일반적이다(물론 예외의 경우는 얼마든지 있을 수 있다). 그 이유는 일반적으로 API보다는 데이터 수집 및 분석 시스템 구현이 난이도가 더 높기 때문이다. 여러분도 백엔드 개발에 입문했으므로 API 개발부터 시작해서 차근차근 경력과 실력을 쌓아 나가면 네이버, 카카오, 구글(Google), 페이스북(Facebook), 아마존(Amazon), 그리고 넷플릭스(Netflix) 시스템 정도의 대규모 데이터 분석 시스템을 구현할 정도로 발전할 수 있을 것이다.