# 시스템 설계 기초

코스 코드: CB2001101

# 시스템이란 무엇인가요?


- 컴퓨팅 시스템
- 실제 시스템

# 시스템 디자인이란 무엇인가요?

- 시스템 설계는 특정 요구사항을 충족하는 시스템의 **아키텍처, 인터페이스 및 데이터를** 정의하는 프로세스입니다. 시스템 설계는 일관되고 효율적인 시스템을 통해 비즈니스 또는 조직의 요구 사항을 충족합니다. 시스템 구축 및 엔지니어링에 대한 체계적인 접근 방식이 필요합니다. 좋은 시스템 설계를 위해서는 인프라부터 데이터 및 데이터 저장 방식에 이르기까지 모든 것을 고려해야 합니다.

# 시스템 설계가 중요한 이유는 무엇인가요?

시스템 설계는 비즈니스 요구사항을 충족하는 솔루션을 정의하는 데 도움이 됩니다. 이는 시스템을 구축할 때 가장 먼저 내릴 수 있는 결정 중 하나입니다. 이러한 결정은 나중에 수정하기가 매우 어렵기 때문에 높은 수준에서 생각하는 것이 필수적인 경우가 많습니다. 또한 시스템이 발

전함에 따라 아키텍처 변경 사항을 더 쉽게 추론하고 관리할 수 있습니다.

# 이 시스템 설계에 접근하는 단계

1. **요구 사항을 이해합니다:** 설계 프로세스를 시작하기 전에 시스템의 요구사항과 제약 조건을 이해하는 것이 중요합니다. 여기에는 문제 공간, 성능 요구 사항, 확장성 요구 사항 및 보안 문제에 대한 정보 수집이 포함됩니다.
2. **주요 구성 요소를 파악합니다:** 시스템의 주요 구성 요소와 이들이 서로 어떻게 상호 작용하는지 파악합니다. 여기에는 서로 다른 구성 요소 간의 관계와 이들이 시스템의 전체 기능에 어떻게 기여하는지 파악하는 것이 포함됩니다.
3. **적절한 기술을 선택합니다:** 요구 사항과 구성 요소에 따라 시스템을 구현할 적절한 기술을 선택합니다. 여기에는 하드웨어 및 소프트웨어 플랫폼, 데이터베이스, 프로그래밍 언어 및 도구 선택이 포함될 수 있습니다.
4. **인터페이스를 정의합니다:** API, 프로토콜, 데이터 형식 등 시스템의 다양한 구성 요소 간의 인터페이스를 정의합니다.
5. **데이터 모델을 설계합니다:** 데이터베이스의 스키마, 데이터 파일의 구조, 구성 요소 간의 데이터 흐름을 포함하여 시스템의 데이터 모델을 설계합니다.

6. **확장성과 성능을 고려합니다:** 로드 밸런싱, 캐싱, 데이터베이스 최적화 등의 요소를 포함하여 설계의 확장성 및 성능에 미치는 영향을 고려하세요.
7. **설계를 테스트하고 검증하세요:** 실제 데이터로 시스템을 테스트하여 설계를 검증하고 사용 사례를 검토하고 발생하는 문제를 해결하기 위해 필요에 따라 변경합니다.
8. **시스템을 배포하고 유지 관리합니다:** 마지막으로 시스템을 배포하고 버그 수정, 구성 요소 업데이트, 필요에 따른 새 기능 추가 등 시간이 지남에 따라 유지 관리합니다.

# 기능 요구 사항

- 이는 최종 사용자가 시스템이 제공해야 하는 기본 기능으로 구체적으로 요구하는 요구 사항입니다. 이러한 모든 기능은 계약의 일부로 시스템에 반드시 통합되어야 합니다.
- 이는 시스템에 제공될 입력, 수행되는 작업 및 예상되는 출력의 형태로 표현되거나 명시됩니다. 비기능적 요구 사항과 달리 최종 제품에서 직접 확인할 수 있는 사용자가 명시한 요구 사항입니다.

# 비기능 요구 사항

- 이는 프로젝트 계약에 따라 시스템이 충족해야 하는 품질 제약 조건입니다. 이러한 요소가 구현되는 우선순위나 정도는 프로젝트마다 다릅니다. 비동작 요구 사항이라고도 합니다. 다음과 같은 문제를 다룹니다:
- 휴대성
- 보안
- 유지 관리 가능성
- 신뢰성
- 확장성
- 성능
- 재사용 가능성

- 유연성

**예시:**

- 각 요청은 최소한의 지연 시간으로 처리되어야 하나요?
- 시스템 가치가 높아야 합니다.

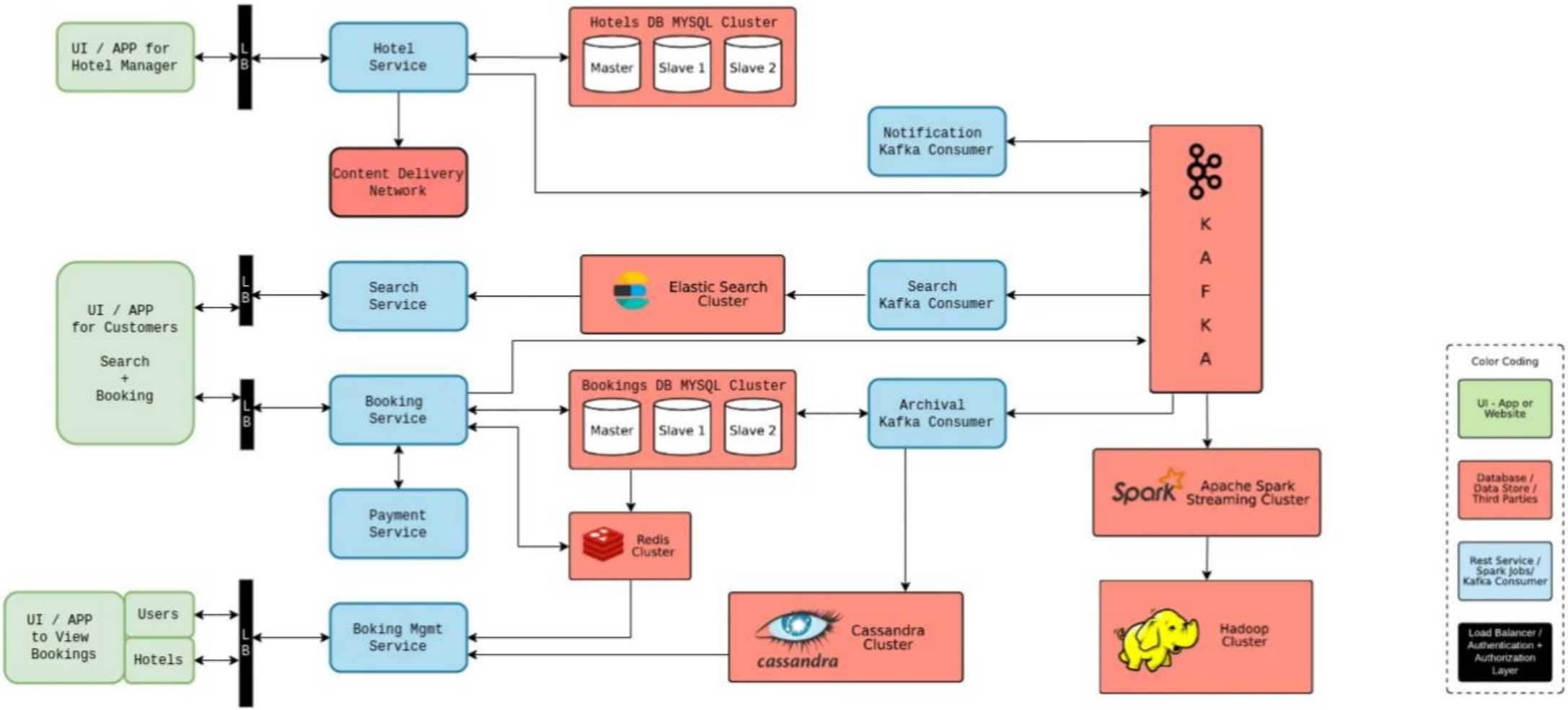
# 차이점

| Functional Requirements | Non Functional Requirements |
| --- | --- |
| A functional requirement defines a system or its component. | A non-functional requirement defines the quality attribute of a software system. |
| It specifies “What should the software system do?” | It places constraints on “How should the software system fulfill the functional requirements?” |
| Functional requirement is specified by User. | Non-functional requirement is specified by technical peoples e.g. Architect, Technical leaders and software developers. |
| It is mandatory. | It is not mandatory. |
| It is captured in use case. | It is captured as a quality attribute. |
| Defined at a component level. | Applied to a system as a whole. |

| | |
|---|---|
| Helps you verify the functionality of the software. | Helps you to verify the performance of the software. |
| Functional Testing like System, Integration, End to End, API testing, etc are done. | Non-Functional Testing like Performance, Stress, Usability, Security testing, etc are done. |
| Usually easy to define. | Usually more difficult to define. |
| Example<br>1) Authentication of user whenever he/she logs into the system.<br>2) System shutdown in case of a cyber attack.<br>3) A Verification email is sent to user whenever he/she registers for the first time on some software system. | Example<br>1) Emails should be sent with a latency of no greater than 12 hours from such an activity.<br>2) The processing of each request should be done within 10 seconds<br>3) The site should load in 3 seconds when the number of simultaneous users are > 10000 |

# 예시: 에어비앤비 시스템 아키텍처

# Airbnb/Booking.com System Design




https://interviewnoodle.com/hotel-booking-system-design-airbnb-1283bbef71ac

# 기능 요구 사항

## 1. **주택 소유자**

- 플랫폼에 호텔 등록 가능
- 호텔에서 객실 유형 추가/업데이트/삭제하기
- 지정된 객실 유형의 객실 추가/업데이트/삭제
- 객실 유형의 가격 및 재고를 일 단위로 정의합니다.

## 2. **사용자/고객**

- 도시, 체크인 및 체크아웃 날짜별로 예약 가능한 호텔을 검색할 수 있습니다.
- 호텔을 선택하고 이용 가능한 모든 호텔 유형과 가격을 확인할 수 있습니다.

- 원하는 객실 유형을 선택하고 예약을 진행할 수 있습니다.
- 예약이 완료되면 예약 세부 정보에 대한 알림을 받습니다.
  완료

# 비기능 요구 사항

- 호텔 관리자/주택 소유주 및 예약과 관련된 시스템 처리 작업 흐름은 매우 일관성이 있어야 합니다.
- 고객에게 호텔을 보여주는 Discovery 플랫폼은 가용성이 높아야 합니다.
- 시스템의 지연 시간이 짧아야 합니다.
- 증가하는 호텔 수를 처리할 수 있는 확장성이 뛰어난 시스템이어야 합니다. 신규 유입 고객 수
- 시스템은 특정 날짜에 두 명의 고객이 같은 객실을 예약할 수 없도록 동시 요청을 처리할 수 있어야 합니다.

# 나머지 API

1.호텔 등록
**포스트/ 호텔/ 등록**
2.호텔의 객실 유형 추가
**POST /hotel/{hotel_id}/객실 유형**
3.호텔 객실 추가
**POST /hotel/{hotel_id}/객실 유형/{객실 유형_id}/객실**
4.인근 호텔 목록 반환
**GET /hotels/location/{location_id}**
5.호텔이 세부 정보를 반환하면
**GET /hotel/{hotel_id}**
6.호텔 객실 예약
**POST /예약**
7.사용자에 대한 예약을 반환합니다.
**GET /user/{user_id}/bookings**
8.호텔 예약을 반환합니다.
**GET /hotel/{hotel_id}/bookings**
9.호텔 체크인
**PUT /예약/{예약_id}/체크인**
10.호텔 체크아웃

APIs 1,2,3 will be part of the **Hotel Management Service.**
APIs 4,5 will be part of the **Discover Platform.**
APIs 6,9,10 will be part of the **Booking Service.**
APIs 7,8 will be part of the **Booking History Service.**

# PUT /예약/{예약_id}/체크아웃

# API

- API는 애플리케이션 소프트웨어를 구축하고 통합하기 위한 정의와 프로토콜의 집합입니다. 정보 제공자와 정보 사용자 간의 계약이라고도 하며, 소비자에게 필요한 콘텐츠(호출)와 생산자에게 필요한 콘텐츠(응답)를 설정합니다.
- 즉, 컴퓨터나 시스템과 상호 작용하여 정보를 검색하거나 기능을 수행하려는 경우 API를 사용하면 원하는 내용을 해당 시스템에 전달하여 시스템이 요청을 이해하고 수행할 수 있도록 도와줍니다.
- API는 사용자 또는 클라이언트와 그들이 원하는 리소스 또는 웹 서비스 사이의 중재자라고 생각할 수 있습니다. **또한 조직이 리소스와 정보를 공유하면서 보안, 제어 및 인증을 유지하여 누가 무엇에 액세스할 수 있는지 결정할 수 있는 방법이기**

**도 합니다.**

- API의 또 다른 장점은 다음과 같은 세부 사항을 알 필요가 없다는 것입니다.
  **캐싱** - **리소스가 검색되는** 방법 또는 **리소스의 출처**.

# Restful API

## REST란 무엇인가요?

- REST(Representational State Transfer)는 API의 작동 방식에 조건을 부과하는 소프트웨어 아키텍처입니다. REST는 처음에 인터넷과 같은 복잡한 네트워크에서 통신을 관리하기 위한 지침으로 만들어졌습니다. REST 기반 아키텍처를 사용하면 대규모로 고성능의 안정적인 통신을 지원할 수 있습니다. 쉽게 구현하고 수정할 수 있어 모든 API 시스템에 가시성과 플랫폼 간 이식성을 제공할 수 있습니다.
- API 개발자는 여러 가지 아키텍처를 사용하여 API를 설계할 수 있습니다. REST 아키텍처 스타일을 따르는 API를 REST API라고 합니다. REST 아키텍

처를 구현하는 웹 서비스를 RESTful 웹 서비스라고 합니다. RESTful API라는 용어는 일반적으로 RESTful 웹 API를 가리킵니다. 그러나 REST API와 RESTful API라는 용어를 혼용하여 사용할 수 있습니다.

# 계속

다음은 REST 아키텍처 스타일의 몇 가지 원칙입니다:

**통일된 인터페이스**

- 통일된 인터페이스는 모든 RESTful 웹서비스 설계의 기본입니다. 이는 서버가 표준 형식으로 정보를 전송한다는 것을 나타냅니다. 형식이 지정된 리소스를 REST의 표현이라고 합니다. 이 형식은 서버 애플리케이션에서 리소스의 내부 표현과 다를 수 있습니다. 예를 들어 서버는 데이터를 텍스트로 저장하지만 HTML 표현 형식으로 전송할 수 있습니다.

**무국적자**

- REST 아키텍처에서 상태 비저장은 서버가 모든 클라이언트 요청을 이전의 모든 요청과 독립적으로 완료하는 통신 방식을 의미합니다. 클라이언트는 어떤 순서로든 리소스를 요청할 수 있으며, 모든 요청은 상태가 없거나 다른 요청과 격리되어 있습니다. 이 REST API 설계 제약 조건은 서버가 매번 요청을 완전히 이해하고 처리할 수 있음을 의미합니다.

**계층화된 시스템**

- 계층화된 시스템 아키텍처에서 클라이언트는 클라이언트와 서버 사이의 다른 인증된 중개자에 연결할 수 있으며 서버로부터 계속 응답을 받을 수 있습니다. 서버는 다른 서버에 요청을 전달할 수도 있습니다. 보안, 애플리케이션, 비즈니스 로직 등 여러 계층이 함께 작동하여 클라이언트 요청을 처리하는 여러 서버에서 실행되도록 RESTful 웹 서비스를 설계할 수 있습니다. 이러한 계층은 클라이언트에게 보이지 않습니다.

**캐시 가능성**

- RESTful 웹 서비스는 서버 응답 시간을 개선하기 위해 일부 응답을 클라이언트 또는 중개자에 저장하는 프로세스인 캐싱을 지원합니다. 예를 들어 모든 페이지에 공통 헤더 및 푸터 이미지가 있는 웹사이트를 방문한다고 가정해 보겠습니다. 새 웹사이트 페이지를 방문할 때마다 서버는 동일한 이미지를 다시 전송해야 합니다. 이를 방지하기 위해 클라이언트는 첫 번째 응답 후에 이러한 이미지를 캐시하거나 저장한 다음 캐시에서 직접 이미지를 사용합니다. RESTful 웹 서비스는 캐시 가능 또는 캐시 불가능으로 정의되는 API 응답을 사용하여 캐싱을 제어합니다.

**온디맨드 코드**

- REST 아키텍처 스타일에서는 서버가 소프트웨어 프로그래밍 코드를 클라이언트로 전송하여 클라이언트 기능을 일시적으로 확장하거나 사용자 지정할 수 있습니다. 예를 들어 웹사이트에서 등록 양식을 작성할 때 잘못된 전화번호와 같은 실수가 발견되면 브라우저는 즉시 이를 강조 표시합니다. 이는 서버에서 전송한 코드 때문에 가능한 일입니다.

# 계속...

- RESTful API의 장점은 무엇인가요?

RESTful API에는 다음과 같은 이점이 있습니

다:

**확장성**

- REST API를 구현하는 시스템은 클라이언트-서버 상호 작용을 최적화하기 때문에 효율적으로 확장할 수 있습니다. 상태 비저장성은 서버가 과거 클라이언트 요청 정보를 보유할 필요가 없으므로 서버 부하를 제거합니다. 잘 관리된 캐싱은 일부 클라이언트-서버 상호 작용을 부분적으로 또는 완전히 제거합니다. 이러한 모든 기능은 성능을 저하시키는 통신 병목 현상을 일으키지 않으면서 확장성을 지원합니다.

**유연성**

- RESTful 웹 서비스는 완전한 클라이언트-서버 분리를 지원합니다. 다양한 서버 구성 요소를 단순화하고 분리하여 각 부분이 독립적으로 발전할 수 있도록 합니다. 서버 애플리케이션의 플랫폼 또

는 기술 변경이 클라이언트 애플리케이션에 영향을 미치지 않습니다. 애플리케이션 기능을 계층화할 수 있어 유연성이 더욱 향상됩니다. 예를 들어 개발자는 애플리케이션 로직을 다시 작성하지 않고도 데이터베이스 계층을 변경할 수 있습니다.

**독립성**

- REST API는 사용되는 기술과 무관합니다. API 설계에 영향을 주지 않고 다양한 프로그래밍 언어로 클라이언트 및 서버 애플리케이션을 모두 작성할 수 있습니다. 또한 통신에 영향을 주지 않고 어느 쪽의 기반 기술을 변경할 수도 있습니다.

# RESTful API 클라이언트 요청에는 무엇이 포함되나요?

- *GET*
- 클라이언트는 GET을 사용하여 서버의 지정된 URL에 있는 리소스에 액세스합니다. 클라이언트는 GET 요청을 캐시하고 RESTful API 요청에 매개변수를 전송하여 서버가 데이터를 전송하기 전에 필터링하도록 지시할 수 있습니다.
- *POST*
- 클라이언트는 POST를 사용하여 서버로 데이터를 전송합니다. 요청에는 데이터 표현이 포함됩니다. 동일한 POST 요청을 여러 번 전송하면 동일한 리소스가 여러 번 생성되는 부작용이 있습니다.
- *PUT*
- 클라이언트는 PUT을 사용하여 서버의 기존 리소스를 업데이트합니다. POST와 달리 RESTful 웹 서비스에서 동일한 PUT 요청을 여러 번 수행해도 동일한 결과가 나타납니다.
- *삭제*
- 클라이언트는 DELETE 요청을 사용하여 리소스를 제거합니다. DELETE 요청은 서버 상

태를 변경할 수 있습니다. 그러나 사용자에게 적절한 인증이 없으면 요청은 실패합니다.

# 수업 배정