**캡스톤디자인 상세 제안서**

|  |  |
| --- | --- |
| **프로젝트명** | **예스24 라이브홀 티켓 예매 애플리케이션** |
| **학번** | **201945058** |
| **이름** | **이 수** |

2023.08.30

**인하공업전문대학**

**컴퓨터시스템과**

**목 차**

1. 프로젝트 개요............................................................................

2. 기존 유사 시스템 고찰 .........................................................

3. 프로젝트 구현에 필요한 기술 고찰.....................................

4. 프로젝트 내용............................................................................

5. 구현 및 개발 툴....................................................................

6. 일정계획..................................................................................

7. 산출물종류 및 제출계획.....................................................

8. H/W 및 S/W 요건..............................................................

9. 기타..........................................................................................

1**. 프로젝트 개요**

제가 예스24 라이브홀 티켓 예매 애플리케이션을 개발 프로젝트로 시작하게 된 이유는 백엔드 개발자로서의 전문성을 극대화하고, 실무에서 요구되는 다양한 능력을 키우기 위해서 입니다. 개발자의 역할은 단순히 코드를 작성하는 능력을 넘어서, 데이터의 일관성을 유지해야 하고, 성능을 최적화해야 하며, 동시성 이슈 해결과 같은 복잡한 문제들을 효과적으로 해결할 수 있는 능력이 필요하다고 판단하였습니다. 특히 대규모 트래픽을 처리하는 서비스의 경우에는 이러한 능력이 더욱 필요하며, 다양한 취업 공고와 기술 블로그를 통해서도 이러한 내용을 확인할 수 있었습니다. 실제로 대용량 트래픽이 발생하는 서비스 환경에서는 백엔드 개발자가 맡아야 할 역할은 성능 유지와 동시성 관리를 보장하는 것이라고 생각합니다. 그렇기 때문에 이 프로젝트에서는 대용량 트래픽 처리, 접속자 대기열 시스템, 선착순 이벤트 시스템, 그리고 동시성 이슈에 대한 복잡한 백엔드 로직에 대해 초점을 맞출 계획입니다.

첫째로, 실제 예스24 라이브홀 홈페이지를 크롤링하여 공연 정보를 등록하는 기능을 구현할 예정입니다. 이를 위해 자바 기반의 스프링 부트와 JPA를 활용하여 백엔드 로직을 작성할 것이며, 크롤링된 데이터는 AWS RDS에 안정적으로 저장될 것입니다. 또한 데이터의 일관성을 유지하고 성능을 최적화하는 작업도 병행할 예정입니다.

둘째로, 대용량 트래픽을 효율적으로 관리하기 위한 접속자 대기열 시스템을 구현할 예정입니다. 이는 특히 티켓 예매가 시작되는 순간에 발생하는 급격한 트래픽 증가를 안정적으로 처리하기 위한 메커니즘이며, 이를 위해 카프카를 활용하여 메시지 큐를 구성할 예정입니다.

셋째로, 선착순 이벤트나 특별 프로모션을 위한 로직을 구현할 것입니다. 이러한 이벤트에서는 동시성 관리가 매우 중요한데, 이를 위해 카프카와 Redis를 활용해 분산 잠금과 순차적인 요청 처리를 체계적으로 구현할 계획입니다.

넷째로, 메시지와 알림 기능은 카프카를 사용하여 비동기 메시징을 구현할 예정입니다. 이를 통해 서비스의 확장성과 처리 효율성을 높일 수 있습니다.

마지막으로, 서버의 배포 및 운영 측면에서도 상당한 노력을 기울일 예정입니다. AWS ECS를 이용해 도커 컨테이너 기반의 배포와 관리를 진행하며, 이를 통해 애플리케이션의 확장성과 유연성을 더욱 높이고자 합니다. 이 모든 배포 및 관리 과정은 쿠버네티스를 활용하여 자동화될 것이며, 이를 통해 오류가 발생했을 경우 빠르고 효율적으로 대응할 수 있도록 할 것입니다. 이러한 방법들을 통해, 실무에서 요구되는 다양한 능력과 지식을 체계적으로 쌓아 나갈 예정입니다.

이러한 복잡한 기술 스택과 아키텍처를 실제로 적용해보고 구현하는 과정은 백엔드 개발자로서의 전반적인 실무 능력을 체계적으로 향상시킬 수 있는 중요한 경험이 될 것입니다. 이 프로젝트를 통해 각 기술의 실제 적용 사례를 깊게 이해하고, 실무에서 자주 발생하는 문제를 직접 해결해보면서 얻는 지식은 가치있는 자산이 될 것입니다. 따라서 이 프로젝트는 단순히 하나의 작업이나 과제를 넘어, 백엔드 개발자의 전반적인 역량을 집중적으로 향상시키는 중요한 기회로 작용할 것입니다. 이로 인해 프로젝트의 필요성과 그 가치는 더욱 명확하고 분명해지며, 이를 통해 시장에서 요구하는 실력을 갖추는 데 큰 도움이 될 것입니다.

**2. 기존 유사 시스템 고찰**

|  |  |  |  |
| --- | --- | --- | --- |
| 기능 | 인터파크 티켓 | 티켓링크 | 예스24 티켓 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**3. 프로젝트 구현에 필요한 기술 고찰**

**1) 필요한 기술 스택**

- 백엔드: Spring Cloud, Spring Security, Spring Boot, Spring Data JPA, AWS RDS, AWS S3, AWS ECS, Kafka, Redis 등

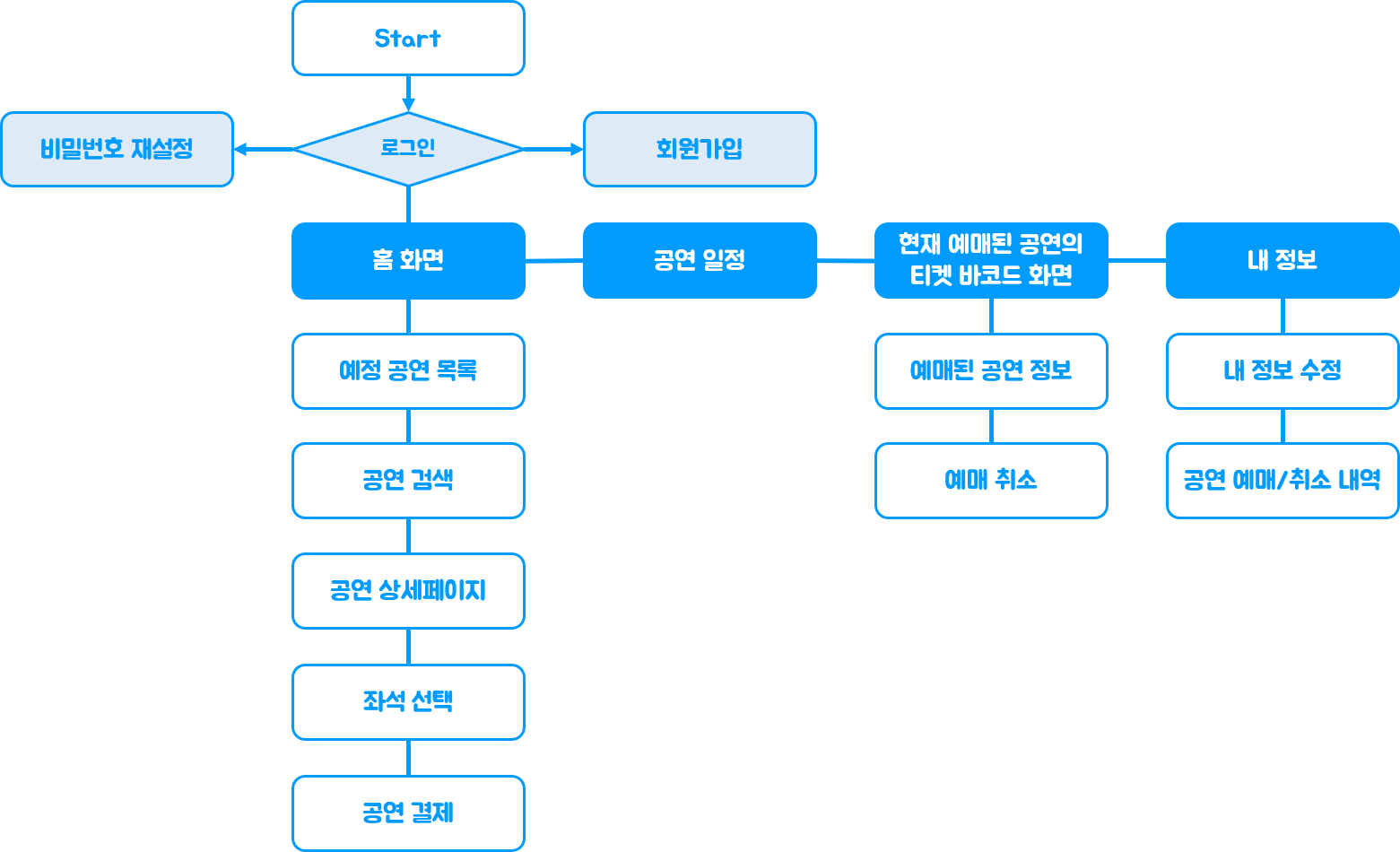
- 프론트엔드: React Native, Java Script,Type Script 등

- 서버 & 배포: AWS ECS, Docker, Kubernetes 등

**4. 프로젝트의 내용**

**1 ) S/W개발 일반 요구사항**

**(1) 애플리케이션 흐름도**



**(2) S/W 관련 사용자 요구사항**

**⓵ 계정 관리**

1.1 로그인

- 사용자는 이메일과 비밀번호를 사용하여 로그인할 수 있다.

- 로그인 후에는 자동으로 로그인 상태를 유지한다.

1.2 회원 가입

- 휴대폰 인증을 통해 신원을 확인할 수 있다.

- 필수 정보와 선택 정보를 입력하여 회원 가입을 완료할 수 있다.

1.3 비밀번호 재설정

- 이메일 인증을 통해 비밀번호를 재설정할 수 있다.

**⓶ 프로필 관리**

- 프로필 사진, 닉네임, 전화번호 등을 추가하거나 수정할 수 있다.

**⓷ 공연 정보**

3.1 공연 목록

- 현재 진행 중인, 예정된, 종료된 공연을 카테고리별로 볼 수 있다.

3.2 공연 상세 정보

- 공연의 상세 정보를 확인할 수 있다.

3.3 공연 검색

키워드를 사용하여 공연을 검색할 수 있다.

**⓸ 티켓 예매**

4.1 공연 선택

- 원하는 공연을 선택하여 상세 정보를 볼 수 있다.

4.2 좌석 선택

- 좌석 배치도를 통해 원하는 좌석을 선택할 수 있다.

- 선택 중인 좌석은 일시적으로 자신만 선택할 수 있다.

- 선택한 좌석은 일정 시간 내에 결제를 완료해야 한다.

4.3 결제

- 결제 프로세스를 거쳐 티켓을 구매할 수 있다.

**⓹ 알림 및 푸시 메시지**

5.1 공연 알림

- 원하는 공연에 대한 알림을 확인할 수 있다.

5.2 티켓 예매 상태 변경 알림

- 예매 상태가 변경되면 알림을 받을 수 있다.

**⓺ 성능 및 안정성**

6.1 대기열 시스템

- 서버에 과부하가 발생할 경우, 자동으로 대기열에 들어가서 순차적으로 서비스를 이용할 수 있다.

6.2 선착순 이벤트

- 선착순 이벤트가 있을 경우, 빠르고 정확하게 참여할 수 있다.

6.3 동시성

- 여러 사용자가 동시에 같은 데이터에 접근해도 문제가 발생하지 않는다.

**⓻ 모니터링 및 로깅**

7.1 시스템 모니터링

- 서비스의 상태는 실시간으로 모니터링되며, 이상 시 알림을 받을 수 있다.

7.2 로깅

- 문제가 발생한 경우, 로그를 통해 원인을 파악할 수 있다.

**2) 세부 요구사항**

**(1) 개발 요구사항**

**⓵ 로그인**

- 기술 스택 : Spring Boot, Spring Security, JWT (JSON Web Token), Spring Data JPA, AWS RDS 등

- 기능 : 사용자가 입력한 이메일, 비밀번호를 검증하고, JWT 토큰을 발급하여 로그인 상태를 유지한다.

**⓶ 회원 가입**

**2.1 휴대폰 번호 인증 코드 발송 API**

- 기술 스택 : Spring Boot, Spring Cloud, Kafka, Redis 등

- 기능 : 사용자가 휴대폰 번호를 입력하면, Kafka를 통해 SMS 발송 서비스에 메시지를 보낸다. 그리고 휴대폰 번호와 인증 코드는 Redis에 일시적으로 저장한다.

**2.2 인증 코드 검증 API**

- 기술 스택 : Spring Boot, Spring Cloud, Redis 등

- 기능 : 사용자가 입력한 인증 코드와 Redis에 저장된 인증 코드를 비교하여 검증한다.

**2.3 회원 정보 입력 및 저장 API**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS, AWS S3 등

- 기능 : 사용자는 이메일, 비밀번호, 이름, 닉네임, 휴대폰 번호, 생년월일, 성별 등을 입력한다. 그리고 선택적으로 프로필 사진도 AWS S3에 업로드할 수 있다.

**2.4 회원 가입 완료**

- 기술 스택 : Spring Boot, AWS ECS, AWS RDS 등

- 기능 : 모든 필요한 정보와 인증이 완료되면, AWS RDS에 사용자 정보를 저장한다.

**⓷ 비밀번호 재설정**

**3.1 인증번호 이메일 전송 API**

- 기술 스택 : Spring Boot, Spring Cloud, Kafka, Redis 등

- 기능 : 사용자가 이메일을 입력하면, Kafka를 통해 이메일 발송 서비스에 메시지를 보낸다. 또한 이메일과 인증 코드는 Redis에 일시적으로 저장한다.

**3.2 인증번호 검증 API**

- 기술 스택 : Spring Boot, Spring Cloud, Redis 등

- 기능 : 사용자가 입력한 인증 코드와 Redis에 저장된 인증 코드를 비교하여 검증한다.

**3.3 비밀번호 재설정 API**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS, Spring Security 등

- 기능 : 인증이 완료된 후 사용자는 새로운 비밀번호를 설정한다. 그리고 사용자의 새 비밀번호는 Spring Security를 통해 암호화되어 AWS RDS에 저장된다.

**⓸ 프로필 정보 수정**

- 기술 스택 : Spring Boot, AWS S3, Spring Data JPA, AWS RDS 등

- 기능 : 사용자는 프로필 사진, 닉네임, 전화번호 등을 추가하거나 수정할 수 있다.

**⓹ 공연 정보 조회**

**5.1 공연 목록 조회**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS 등

- 기능 : 현재 진행 중인, 예정된, 종료된 공연을 카테고리별로 분류한다.

**5.2 공연 상세 정보 조회**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS, AWS S3 등

- 기능 : 공연 제목, 장소, 시간, 가격, 공연 포스터 사진 등 상세 정보를 보여준다.

**5.3 공연 검색**

- 기술 스택: Spring Boot, Spring Data JPA, AWS RDS

- 기능 : 공연명 등의 키워드를 사용하여 검색이 가능하다.

**⓺ 티켓 예매**

**6.1 공연 선택**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS 등

- 기능 : 사용자는 리스트에서 공연을 선택하여 상세 정보 페이지로 이동한다.

**6.2 좌석 선택**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS, Redis 등

- 기능 : 좌석 배치도를 이용하여 사용자는 좌석을 선택할 수 있다. 해당 좌석 정보는 Redis에 일시적으로 저장된다. 또한 Redis에 저장된 좌석은 다른 사용자에게는 "선택 중"으로 표시되며 선택할 수 없다. 그리고 결제가 완료되면 Redis에서 해당 좌석 정보를 삭제하고, RDS에 최종적으로 저장한다.

**6.3 결제**

- 기술 스택 : Spring Boot, Spring Data JPA, AWS RDS 등

- 기능 : 결제를 진행한다. 그리고 결제 전, 사용자에게 최종 확인 메시지를 보여준다.

**6.4 타임아웃 처리**

- 기술 스택 : Spring Boot, Redis 등

- 기능 : 사용자가 좌석을 선택하고 일정 시간 내에 결제를 완료하지 않으면, Redis에서 해당 좌석 정보를 자동으로 삭제한다.

**⓻ 알림 및 푸시 메시지**

**7.1 공연 알림 설정**

- 기술 스택 : Spring Boot, Spring Cloud, Kafka 등

- 기능 : 사용자는 원하는 공연에 대한 알림을 설정할 수 있다. 또한 설정한 알림은 애플리케이션 내 또는 이메일로 받을 수 있다.

**7.2 티켓 예매 상태 변경 알림**

- 기술 스택 : Spring Boot, Kafka 등

- 기능 : 예매 완료, 예매 취소 등의 상태 변경 시, 사용자에게 알림이 전송된다.

**⓼ 성능 및 동시성 이슈**

**8.1 대용량 트래픽 처리**

- 기술 스택 : Spring Boot, Spring Cloud, Kafka, Load Balancer (Nginx) 등

- 기능 : 트래픽이 급증할 경우, Load Balancer를 사용해 여러 서버에 트래픽을 분산한다. 또한 비동기 메시지 큐 (Kafka)를 활용하여 서비스 간의 통신을 비동기로 처리한다.

8.2 접속자 대기열 시스템

- 기술 스택 : Redis, Spring Boot 등

- 기능 : 과도한 트래픽이 발생할 경우, 대기열 시스템을 도입하여 접속자를 일정 수준으로 제한한다. 또한 대기 중인 접속자는 Redis에 저장되며, 순차적으로 서비스를 이용할 수 있게 된다.

8.3 선착순 이벤트 시스템

- 기술 스택 : Spring Boot, Redis, Kafka 등

- 기능 : 선착순 이벤트가 진행될 경우, Kafka와 Redis를 활용하여 빠르고 안정적인 처리를 보장한다.

**8.4 동시성 이슈 처리**

- 기술 스택 : Spring Boot, Spring Data JPA (Optimistic Locking), Redis (for caching)

- 기능 : 여러 사용자가 동시에 같은 데이터에 접근할 경우, Optimistic Locking을 통해 동시성 이슈를 해결한다. 또한 자주 사용되는 데이터는 Redis를 활용해 캐싱하여 성능을 향상시킨다.

**⓽ 모니터링 및 로깅**

**9.1 시스템 모니터링**

- 기술 스택 : Grafana, Prometheus

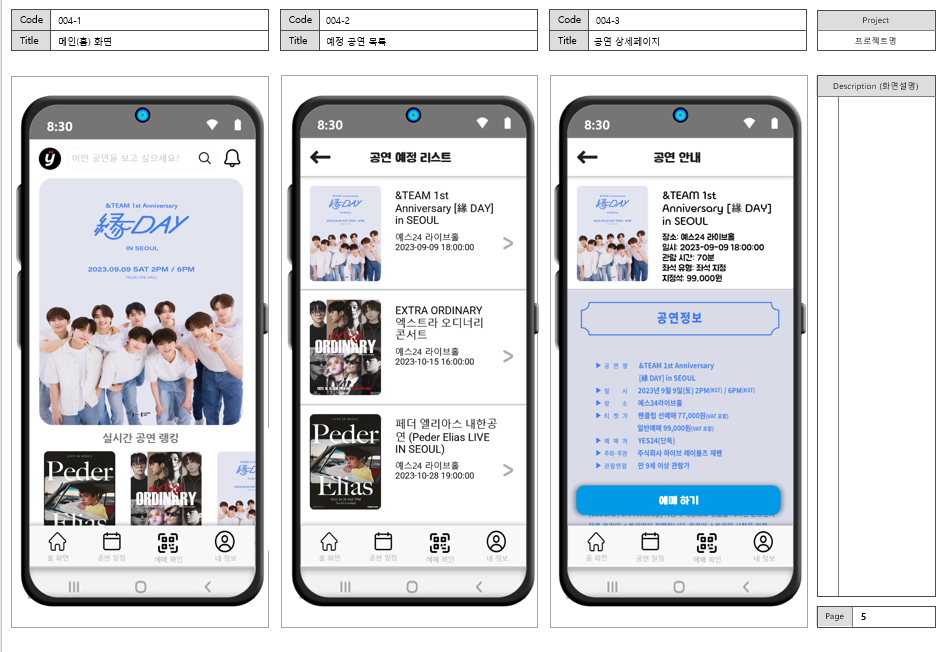
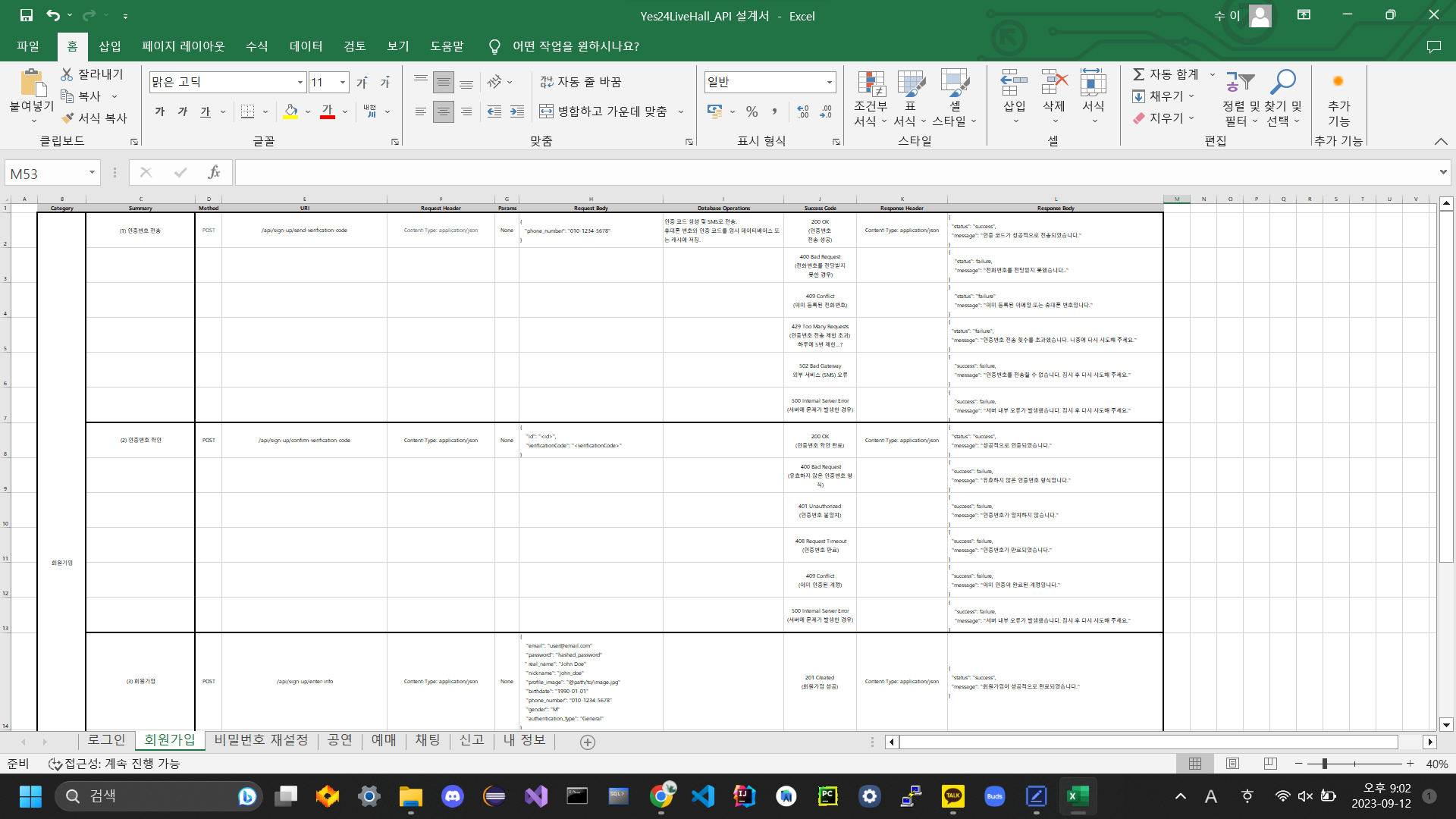
- 기능 : 서비스의 성능 지표, 에러율, 트래픽 등을 실시간으로 모니터링한다.

**9.2 로깅**

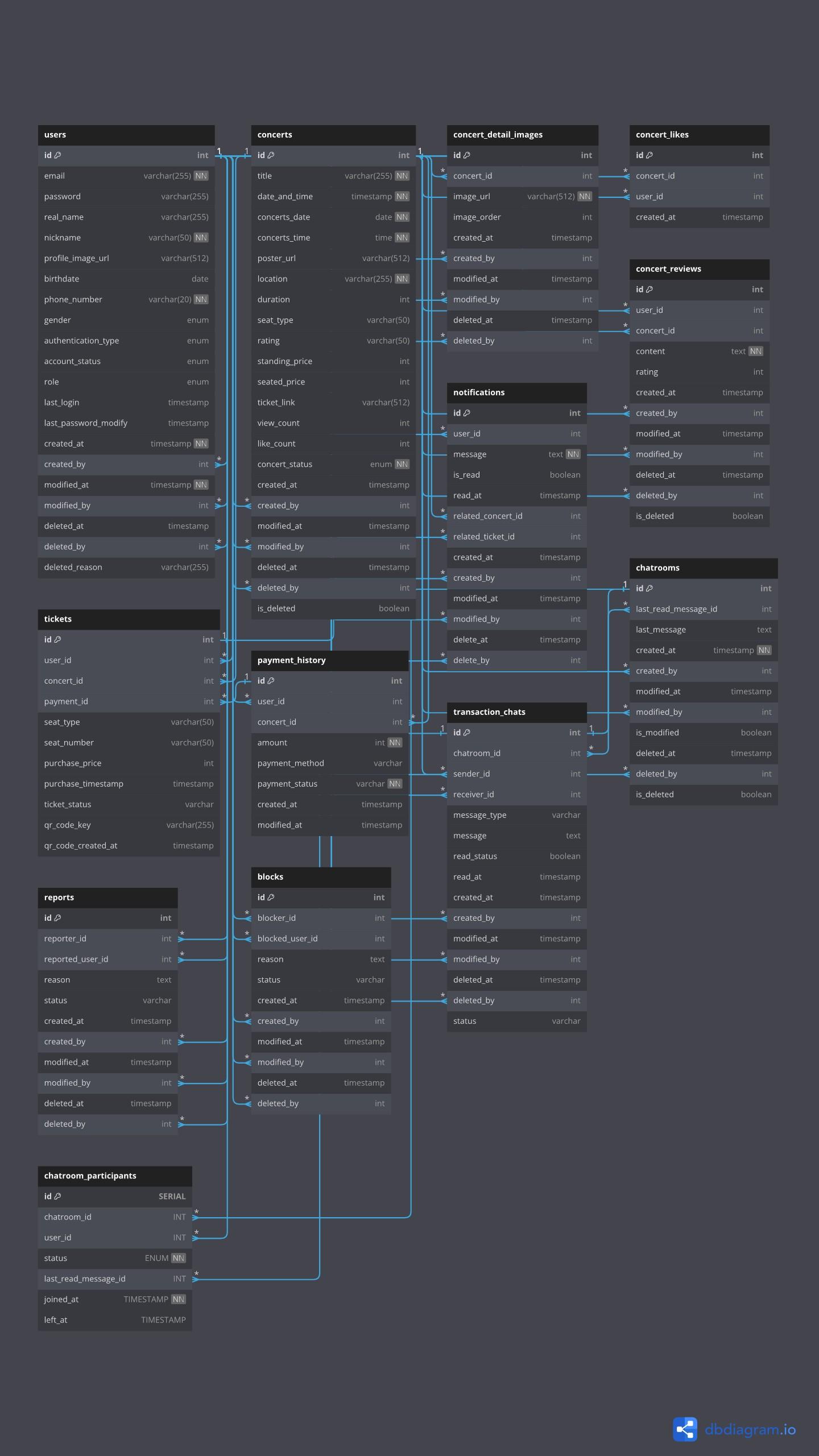
- 기술 스택 : ELK Stack (Elasticsearch, Logstash, Kibana) or AWS CloudWatch

- 기능 : 시스템의 로그를 수집, 분석, 시각화한다. 또한 에러 발생 시 알림을 받을 수 있도록 설정한다.

**(2) 개발 세부 내용**

** **

**[ 앱 디자인 설계 ] [ api 설계 ]**

****

**[ ERD 설계 ]**

**5. 구현 및 개발 툴**

- IntelliJ IDEA 2022.3.2 (Ultimate Edition)

- Visual Studio Code 1.81.1

**6. 일정 계획**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 시작일 | 완료일 | 9월 | | | | 10월 | | | | 11월 | | | | 12월 | | | 비고 |
| 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 |
| 계획,  제안서 제출 | 08/30 | 09/12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 분석 / 설계 | 08/30 | 09/12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 구현 | 09/14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 문서화 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**7. 산출물종류 및 제출계획**

◦ 보고계획

|  |  |  |
| --- | --- | --- |
| **보고명** | **보고내용** | **보고방법** |
| 제안서제출 | - 프로젝트 제안서 제출 | - 제안서 제출 |
| 중간보고서 | - 중간결과 | - 중간보고서 |
| 결과 발표 | - 완성된 프로젝트 결과 시연 | - 프로젝트 시연 |
| 기말보고서 | - 최종결과 | - 최종결과보고서 |

**8. H/W 및 S/W 공급**

- 시스템 내역

**□ 개발/운영용 장비**

|  |  |  |  |
| --- | --- | --- | --- |
| 구 분 | 규 격 | 수량 | 비고 |
|  |  |  |  |

**□ S/W**

|  |  |  |  |
| --- | --- | --- | --- |
| 구 분 | 규 격 | 수량 | 비고 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

9. 기타사항