Software Design Specification

bitCO2e

by Team1
김영민, 박경린, 성주용, 송태현, 이찬

Instructor: 이은석
Document Data: 26 May, 2024
Faculty: SungKyunKwan University

Document History

Date	Version	Description	Writer
21 May, 2024	1.1	서론, 소개	김영민, 박경린, 이찬
23 May, 2024	1.2	시스템 아키텍처-Overall 시스템 아키텍처-Frontend 개발계획	김영민, 박경린, 이찬
24 May, 2024	1.3	시스템 아키텍처-Backend, 프로토콜 디자인, 데이터베이스 디자인, 테스팅 계획	송태현, 성주용
26 May, 2024	1.4	테스팅 계획 내용 추가	송태현
26 May, 2024	1.5	1.4 Document Structure 내용 추가, Contents 목록 최신화	송태현
26 May, 2024	1.6	문서 포맷팅	성주용

Contents

List of Figures

List of Tables

1. 서론

1.1. Readership

본 문서는 다음의 독자를 대상으로한다. 먼저, 본 시스템의 개발자들이다. 개발자는 Front-end 팀과 Back-end 팀으로 구성되며 그린 알고리즘을 서칭, 개발하는 개발자 또한 포함된다. 다음으로는 추후 프로그램을 유지보수 할 개발자들이다. 금학기 이후 BitCO2e 시스템을 수정 보완하고자 하는 개발자는 본 문서를 통해 세부 디자인 사항을 확인할 수 있다. 그리고 오픈 소스로 공개되어 있는 'BitCO2e' 시스템을 수정하여 자신의 시스템에 적용하고자 하는 개발자도 독자에 포함된다. 마지막으로 소프트웨어 공학 개론 수업의 교수, 조교, 학생들이 대상 독자이다.

1.2. Scope

본 문서는 'BitCO2e' 시스템의 디자인 사항을 기술한다. 전반적인 시스템 아키텍처와 세부적인 Front-end, Back-end 구조를 다이어그램을 사용해 설명한다. 또한 본 문서는 데이터베이스 구조, 클라이언트-서버 프로토콜 디자인과 테스트 계획, 개발도구와 개발에 사용된 라이브러리 등에 대한 내용을 포함한다.

1.3. Objective

본 디자인 명세서의 목적은 BitCO2e 시스템의 기술적 디자인에 대한 세부적인 정보 제공에 있다. 이 문서는 Front-end, Back-End, Database 측면에서 설계를 정의한다. 디자인 명세서의 설계 사항은 요구사항 명세서의 내용을 바탕으로 작성되었으며, 개발자는 본 문서를 바탕으로 개발을 진행하고, 추후 유지보수를 위해 본 문서가 활용될 수 있다.

1.4. Document Structure

본 디자인 명세서의 구조는 아래와 같다.

1) 서론: 본 문서의 목적, 예상 독자 및 문서의 구조에 대해 설명한다.
2) 소개: 본 문서를 작성하는데 사용된 도구와 다이어그램들, 프로젝트 범위 그리고 참고자료에 대해 설명한다.
3) 시스템 아키텍처 - Overall: 시스템의 전체적인 구조를 서술한다.
4) 시스템 아키텍처 - Frontend: Frontend 시스템의 구조를 서술한다.
5) 시스템 아키텍처 - Backend: Backend 시스템의 구조를 서술한다.
6) 프로토콜 디자인 - 클라이언트와 서버의 통신 프로토콜 디자인을 서술한다.
7) 데이터베이스 디자인 - 시스템의 데이터베이스 디자인을 서술한다.
8) 테스팅 계획: 테스트 종류, 테스트 시나리오를 포함한 테스트 계획을 서술한다.
9) 개발 계획: 시스템의 구현을 위한 개발환경 및 기술 스택, 디자인과 시스템 실행 제약사항을 서술한다.

2. 소개

2.1. Objectives

이 파트에서는 디자인 명세서에 사용된 다이어그램, 프로젝트 범위, 참고자료에 대해 기술한다. Front-end, Back-end 아키텍처를 표현하기 위해 사용된 다이어그램의 종류와 다이어그램 작성을 위해 사용된 도구, 각 다이어그램의 유형과 목적을 설명한다. 또한 'BitCO2e' 시스템의 범위와 문서 작성을 위해 참고한 자료를 기록한다.

2.2. Applied Diagrams

본 디자인 명세서에서는 여러 다이어그램을 사용하여 시스템 디자인을 표현한다. BitCO2e 전체 시스템에 대한 디자인을 나타내기 위해 process diagram과 use case diagram이 사용된다. 프론트엔드 아키텍처 표현을 위해서는 sequence diagram과 state diagram을 사용하며 백엔드 아키텍처 표현을 위해서는 ER diagram을 사용한다.

2.2.1. Used Tools

본 디자인 명세서에서 사용된 다이어그램은 마이크로소프트 파워포인트를 이용해 작성되었다. 파워포인트는 다이어그램 작성에 매우 유용한 도구로, 여러 도형, 텍스트 상자, 아이콘 등을 활용해 직관적이고 이해하기 쉬운 시각 자료를 제작할 수 있다.

2.2.2. Diagram Types

1) Process Diagram

프로세스 다이어그램은 시스템이나 비즈니스 프로세스의 흐름을 시각적으로 표현한 다이어그램이다. 이는 프로세스의 단계, 의사결정 지점, 데이터 흐름 등을 나타낸다. 일반적으로 다음 단계를 나타내는 화살표와 각 프로세스의 단계를 나타내는 도형을 이용한 순서도로 만들어진다.

2) Use Case Diagram

Use Case Diagram은 시스템과 사용자 간의 상호작용을 모델링하는 다이어그램이다. 시스템과 상호작용하는 actor와 actor의 use case를 바탕으로 작성된다. actor는 시스템 외부에서 시스템과 상호작용하는 주체로, 사용자나 또 다른 시스템을 의미한다. use case는 시스템이 제공하는 기능을 기술한 것이다.

3) Sequence Diagram

Sequence Diagram은 객체 간의 상호작용을 순서에 따라 표현하는 다이어그램이다. 객체가 메시지를 주고받는 순서를 나타낸다. 객체는 시스템 안에서 상호작용하는 객체를 의미하며 메시지는 객체 간에 주고받는 request, interaction 등을 의미한다.

4) State Diagram

State Diagram은 객체가 가질 수 있는 상태와 상태 간의 전이를 모델링 하는 다이어그램이다. 각 오브젝트의 라이프 사이클을 시각적으로 표현한다. state는 객체가 가질 수 있는 상태를 나타내며, 전이는 한 상태에서 다른 상태로 state가 변화하는 것을 의미한다. 이러한 상태 전이를 유발하는 것을 이벤트라고 한다.

5) ER Diagram

ER Diagram은 데이터베이스 설계를 위한 모델링 도구이다. 데이터베이스의 엔티티와 엔티티 간의 관계를 시각적으로 표현한다. 엔티티는 데이터베이스에 저장되는 객체를 나타내며 각 엔티티 간의 연관성을 관계, 엔티티의 세부사항을 속성이라고 한다.

2.3. Project Scope

이 명세서에서 소개하는 ‘BitCO2e’ 시스템은 사용자가 입력한 Java 소스코드를 리팩토링하고, 이를 통해 절감한 탄소배출량을 분석, 시각화하여 사용자에게 제공한다. 또한 로그인한 회원에 한해 포인트(bit)를 제공하며 회원은 bit를 소모하여 자신의 비영리적 광고를 웹사이트에 게시할 수 있다.

따라서, 본 프로젝트는 회원 관리 시스템(회원가입, 로그인, 회원정보 관리 등), Java 소스 코드 리팩토링 시스템, 광고 처리 시스템을 포함한다. 본 프로젝트에서는 사용자의 소스 코드 리팩토링의 대상 언어를 Java에 한정하며, 이외의 프로그래밍 언어는 이 시스템의 범위에 포함하지 않는다.

2.4. References

Green Algorithms, https://www.green-algorithms.org/

https://support.microsoft.com/ko-kr/topic/visio%EC%9D%98-%ED%94%84%EB%A1%9C%EC%84%B8%EC%8A%A4-%EB%8B%A4%EC%9D%B4%EC%96%B4%EA%B7%B8%EB%9E%A8-f064cd25-d7d5-47b8-87e1-ecb3c39cc165 (Visio의 프로세스 다이어그램. ms docs)

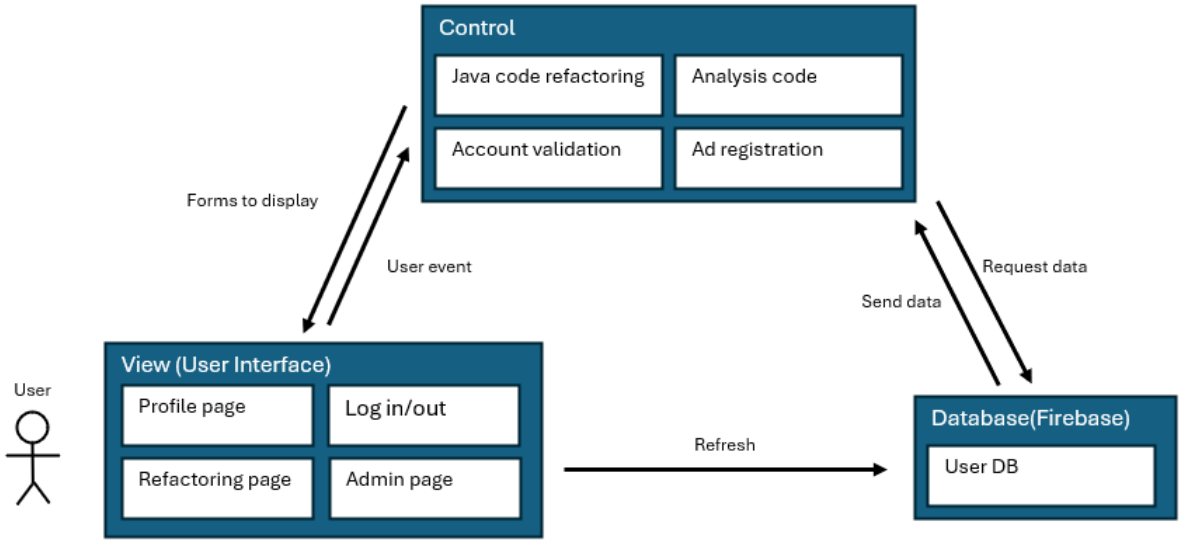
3. 시스템 아키텍처 - Overall

3.1. Objectives

시스템 아키텍처 파트는 시스템 구성에 대한 전반적인 개요를 제공한다. 전체적인 시스템 구성을 Process diagram과 Usecase diagram으로 설명한다.

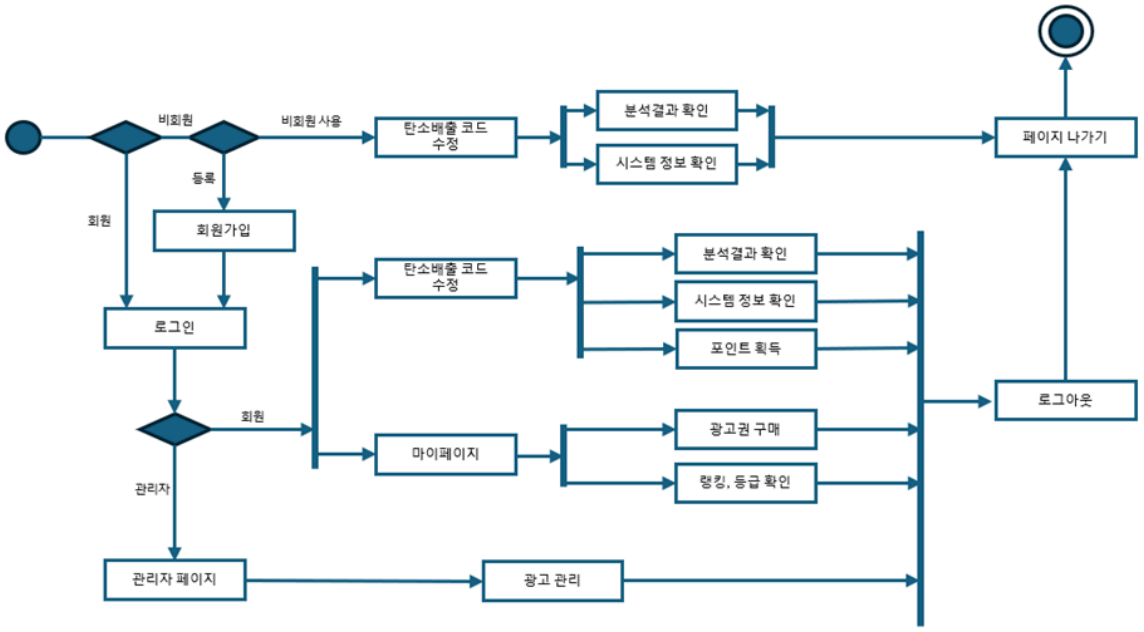
3.2. System Organization

BitCO2e 서비스는 클라이언트-서버 모델을 사용해 설계되었다. 사용자는 BitCO2e 서비스의 프론트엔드인 웹페이지와 상호작용한다. 프론트엔드 웹사이트는 백엔드 서버와 HTTP 통신을 통해 API를 호출하여 상호작용한다. 백엔드 어플리케이션은 프론트엔드로부터의 요청을 컨트롤러로 배포하고, Firebase 데이터베이스에서 필요한 정보를 획득하여 처리한 후 처리 결과를 프론트 엔드로 내보낸다. 사용자는 포인트(bit) 획득, 랭킹 관리를 위해 계정을 생성해야 한다. 회원가입 절차에서는 사용자가 입력한 계정명이 데이터베이스에 저장되어있는 기존 계정명과 중복되지 않는 계정이어야 한다. 소스 코드 리팩토링 기능은 회원이 아닌 사용자도 사용할 수 있다.



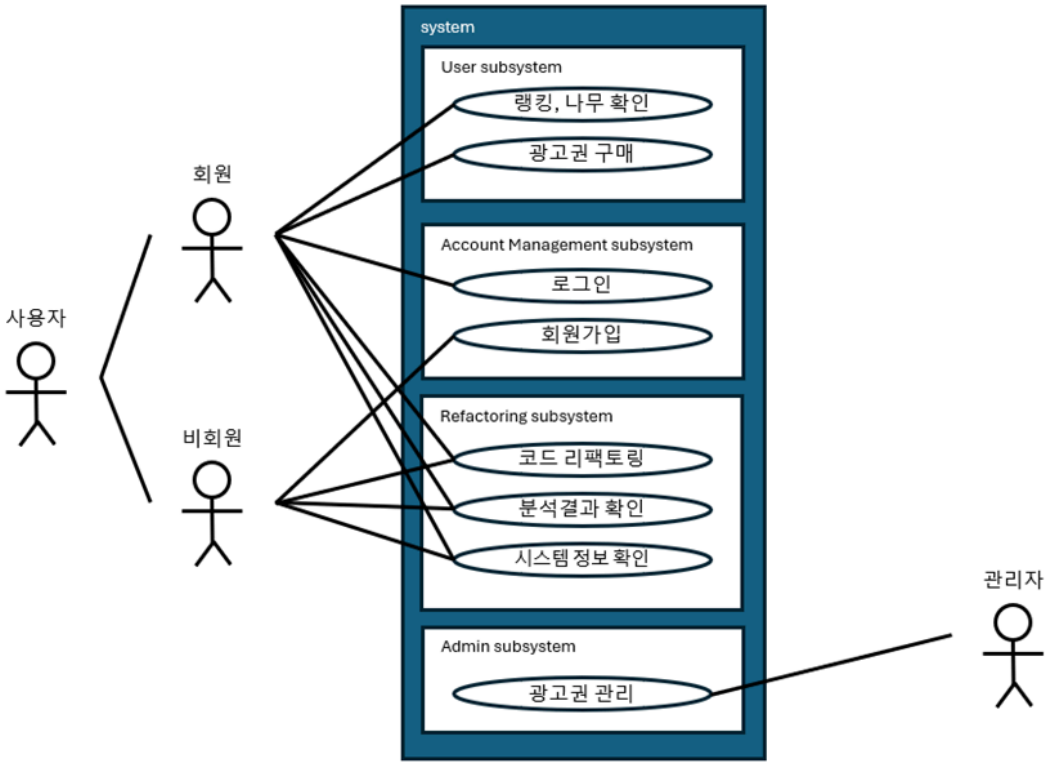
[Figure 1] Overall System Architecture

3.2.1. Process Diagram (System)



[Figure 2] Process diagram - Overall

3.2.2. Use Case Diagram



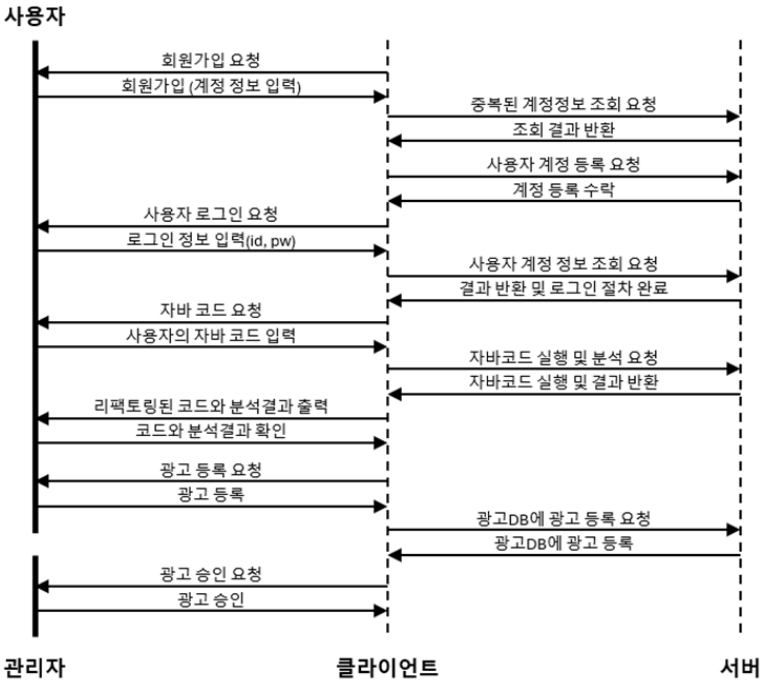
[Figure 3] Use case diagram - Overall

4. 시스템 아키텍처 - Frontend

4.1. Objectives

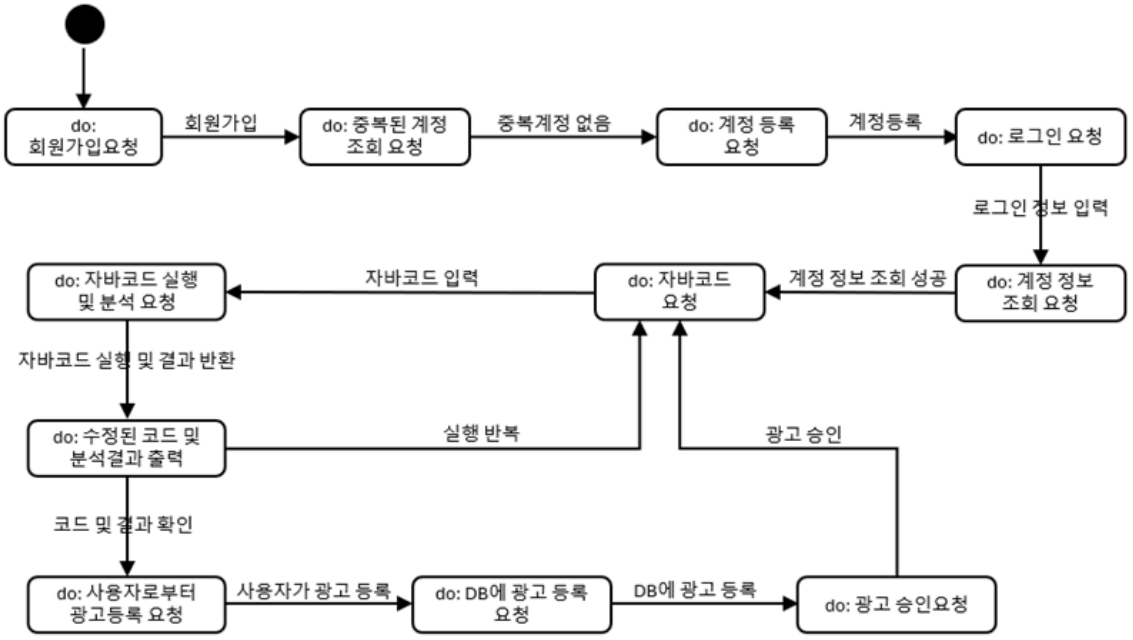
이 파트에서는 프론트엔드 시스템의 구조와 컴포넌트별 속성 및 기능, 그리고 각 구성요소의 연결관계 등을 sequence diagram, state diagram 등을 통해 나타낸다.

4.2. Sequence Diagram



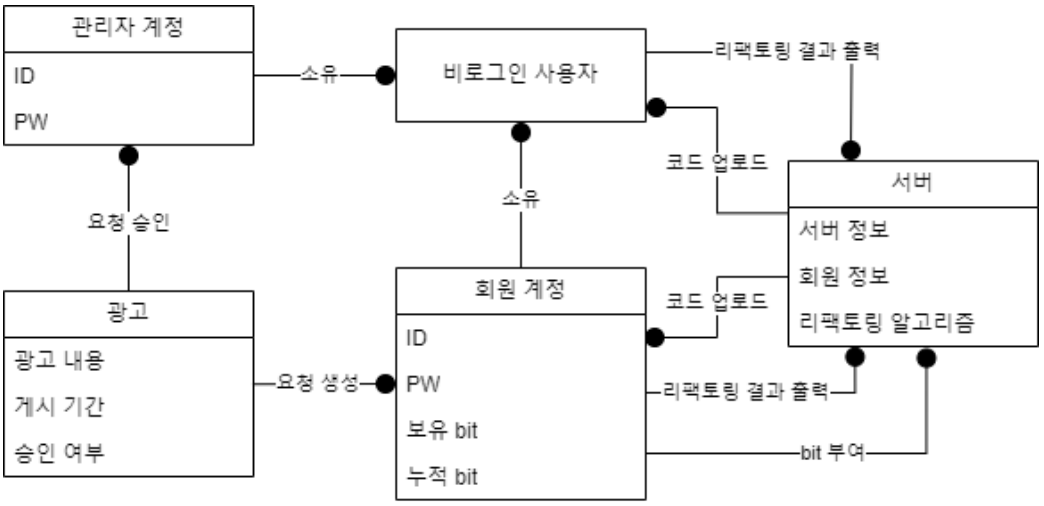
[Figure 4] Sequence diagram - Frontend

4.3. State Diagram(Client-side)



[Figure 5] State diagram - Frontend

4.4. Attribute of Classes



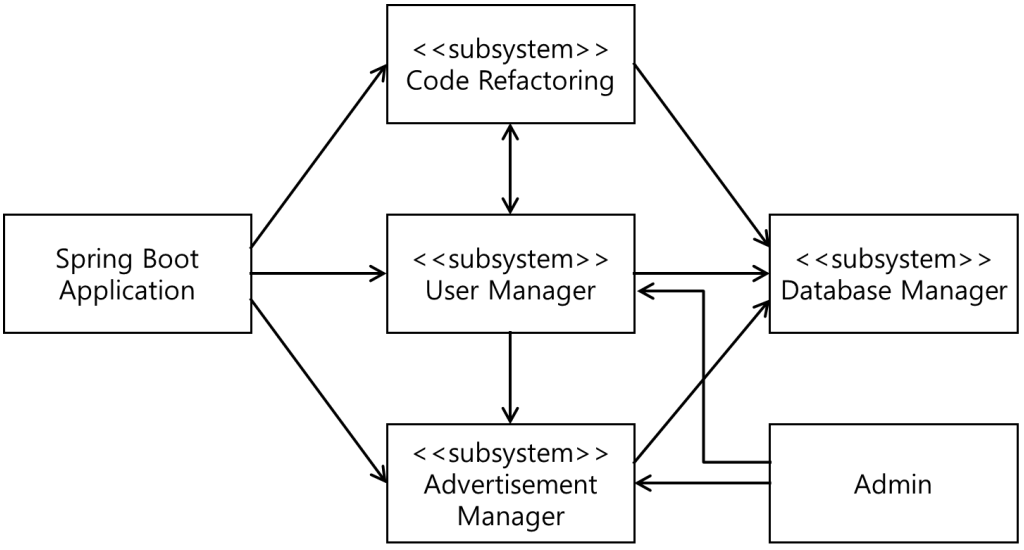
[Figure 6] Attribute of classes - Frontend

5. 시스템 아키텍처 - Backend

5.1. Objectives

백엔드 시스템의 구조, 속성, 기능 그리고 하위 컴포넌트 간의 관계를 설명하고, class diagram과 sequence diagram으로 나타낸다.

5.2. Backend Architecture



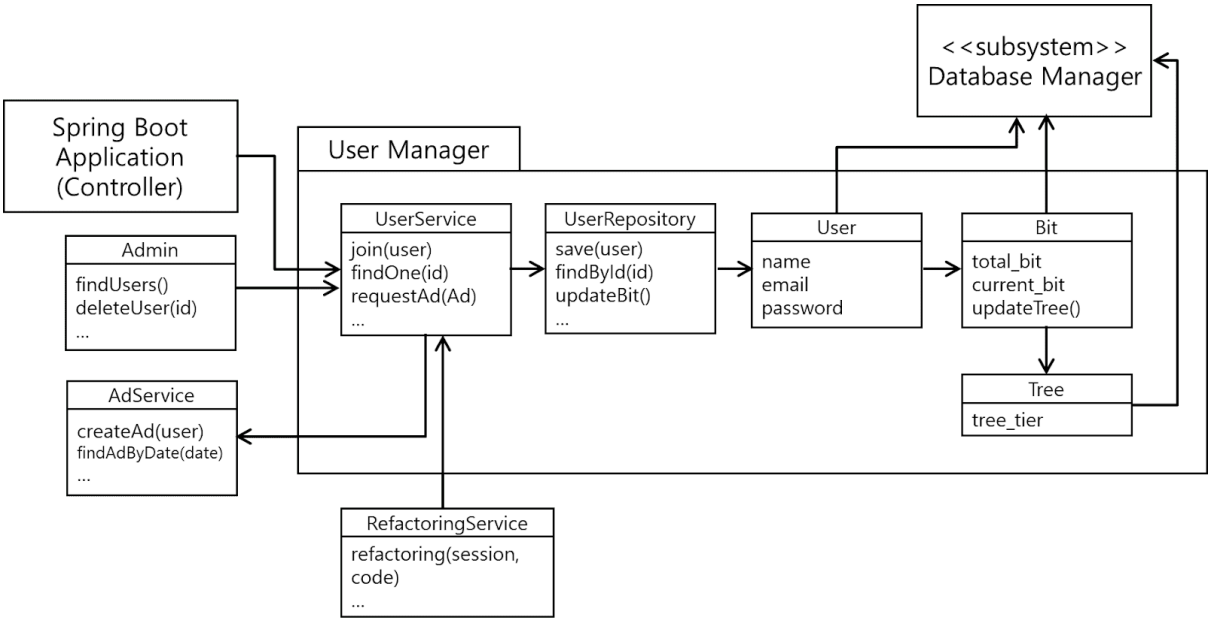
[Figure 7] Backend architecture

Spring Boot를 이용하여 백엔드를 구성한다. 다이어그램에서 나타낸 Spring Boot Application은 Frontend로 부터 받은 요청을 처리하는 Controller를 포함한 Spring Application 전반을 뜻한다. Frontend에서 요청이 들어오면 Controller에서는 하위 컴포넌트들을 활용하여 요청을 수행한다. 수행 중 발생하는 데이터 생성과 변동은 Database Manager로 보내져 처리된다. Database Manager는 Firebase를 database로 사용하고 database를 관리하는 sub-system이다.

5.3. Subcomponents:

5.3.1. User Manager

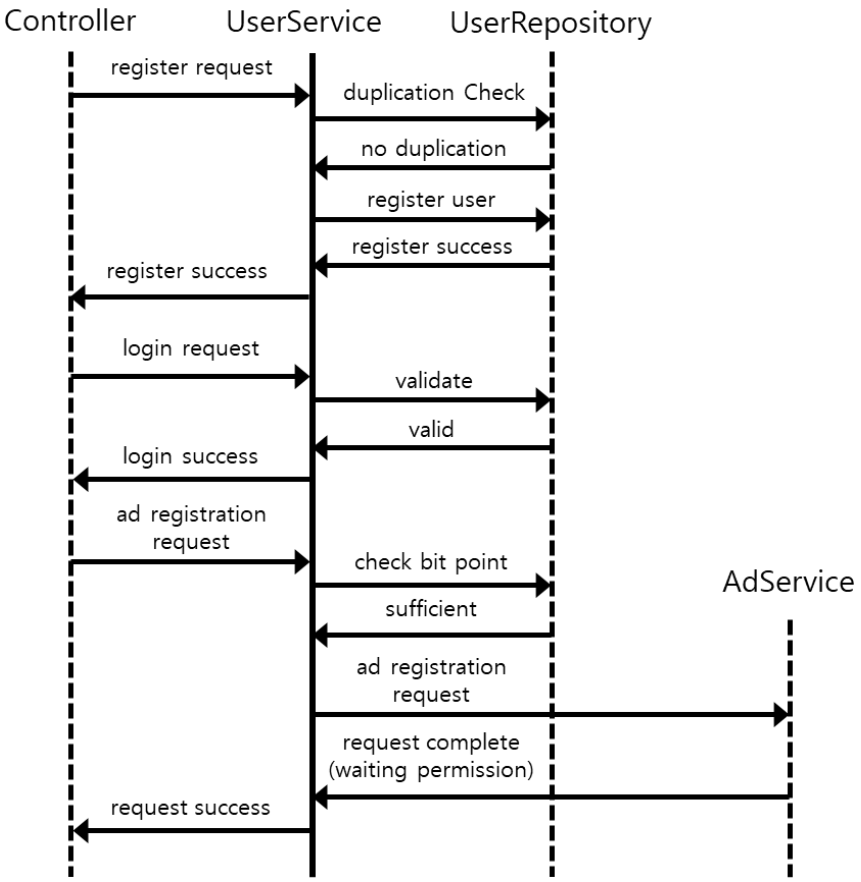
User의 데이터를 관리하고 관련 요청을 수행하는 sub-system이다. Register와 login같은 user 데이터를 직접적으로 생성하거나 조회하는 요청부터, 광고 등록 요청이나 코드 리팩토링 요청과 같은 user 데이터에 의존하는 요청까지 user 데이터와 관련된 모든 요청에 직간접적으로 적절한 역할을 수행한다.



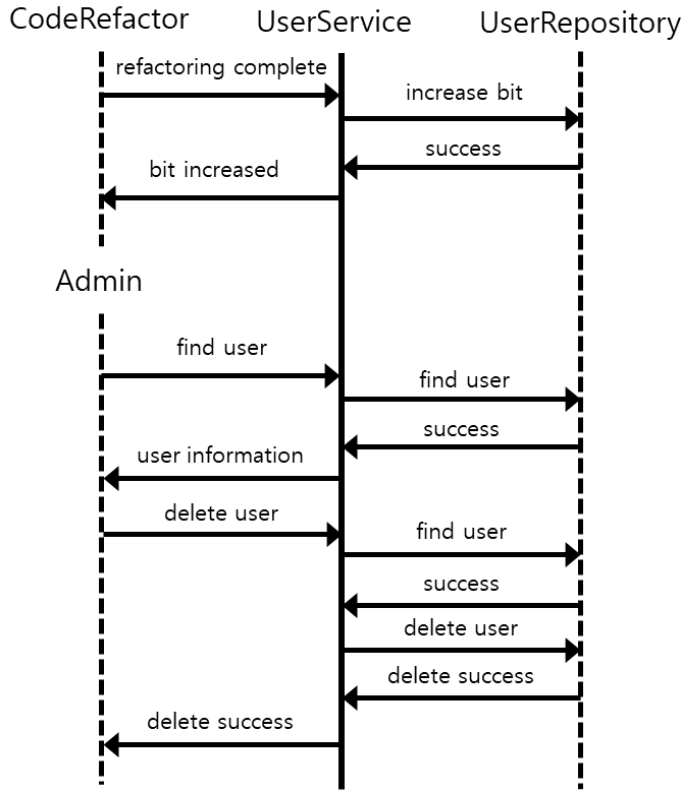
[Figure 8] Class diagram - Backend: User manager

UserService: 다른 외부 컴포넌트와 직접 연결되어 다른 system과의 중재자 역할을 한다.

UserRepository: User 데이터와 직접 상호작용하여 관리하는 컴포넌트이다.



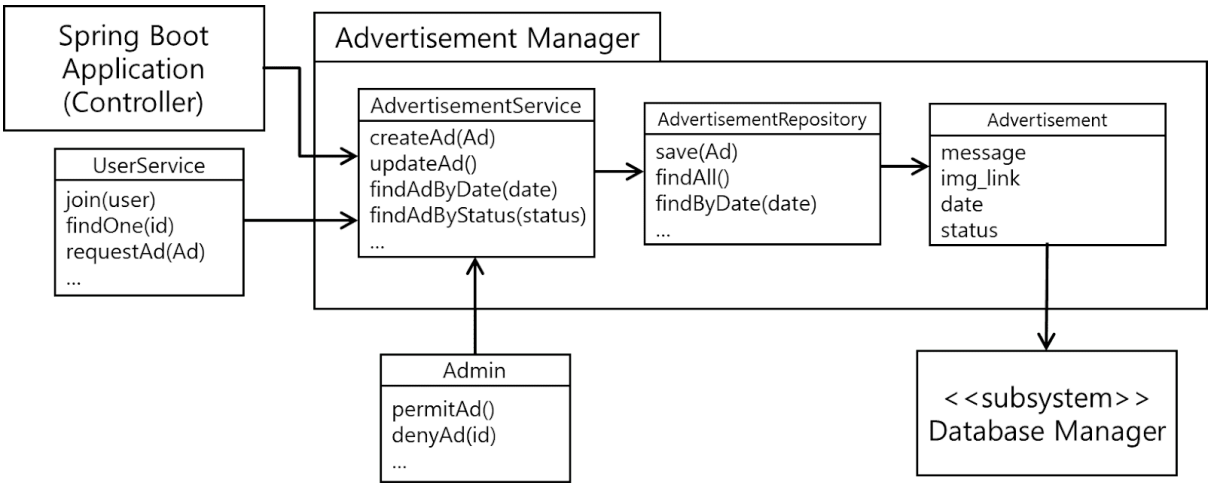
[Figure 9] Sequence diagram - User manager 1



[Figure 10] Sequence diagram - User manager 2

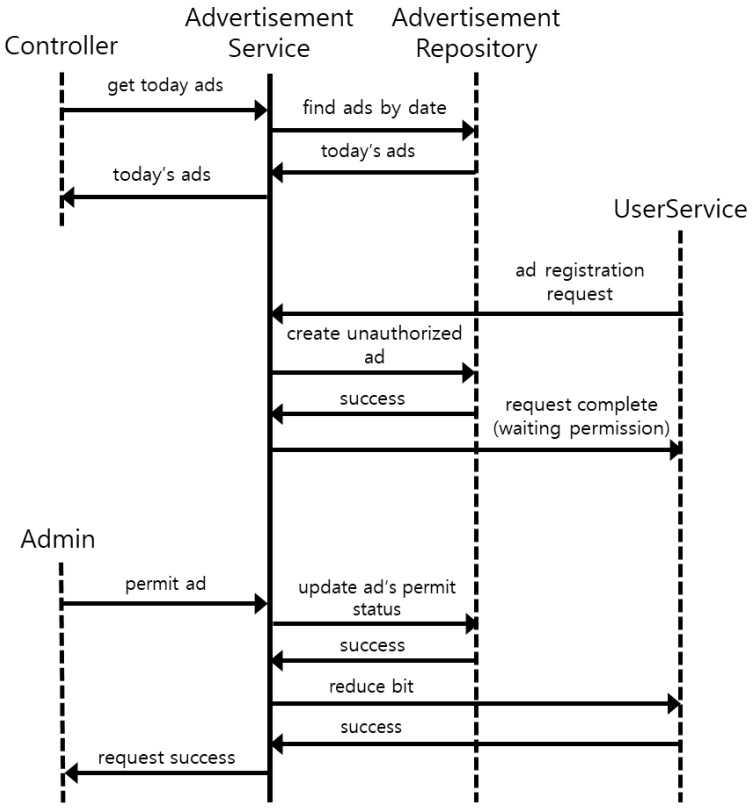
5.3.2. Advertisement Manager

광고 데이터를 관리하고 관련 요청을 수행하는 sub-system이다.



[Figure 11] Class diagram - Backend: Advertisement manager

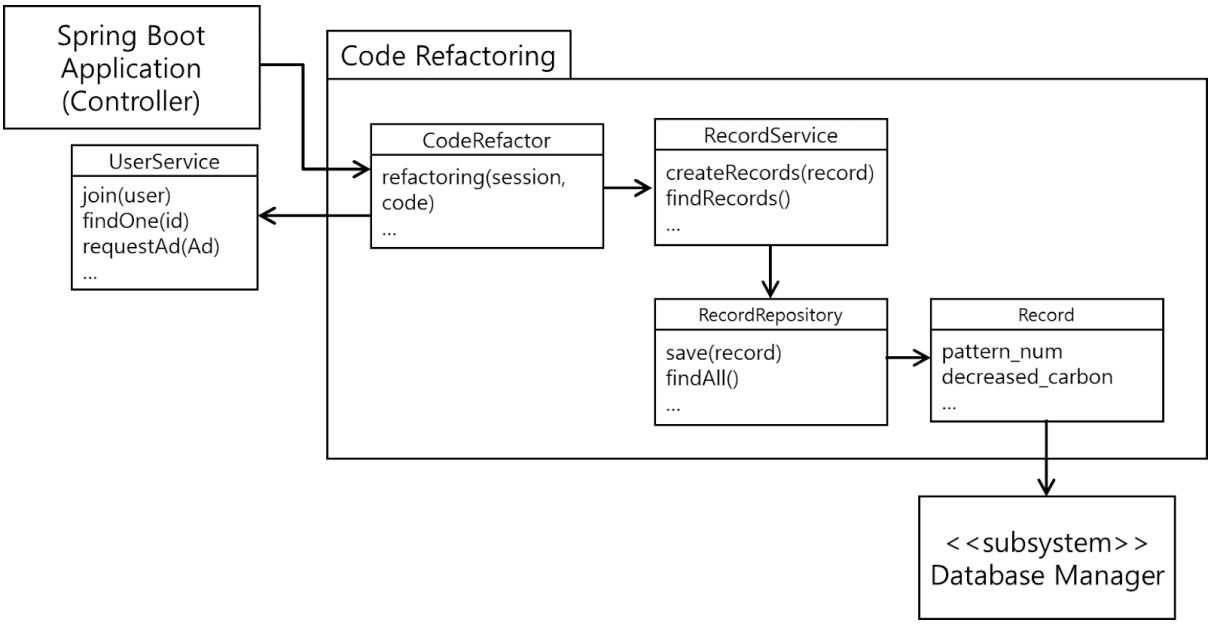
AdvertisementService: 다른 외부 컴포넌트와 직접 연결되어 다른 system과의 중재자 역할을 한다.
AdvertisementRepository: Advertisement 데이터와 직접 상호작용하여 관리하는 컴포넌트이다.



[Figure 12] Sequence diagram - Backend: Advertisement manager

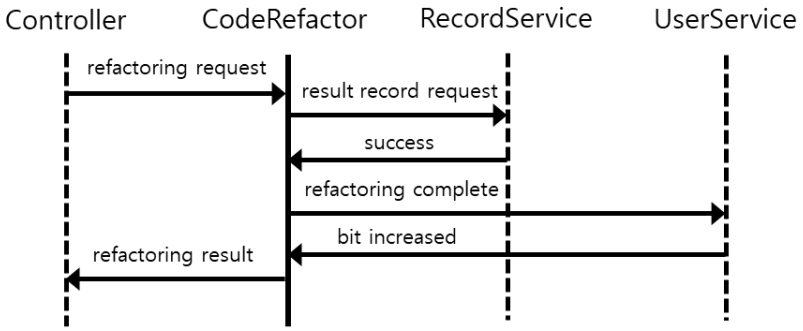
5.3.3. Code Refactoring

리팩토링 요청을 수행하고 리팩토링 기록, Record 데이터를 관리하는 sub-system이다



[Figure 13] Class diagram - Backend: Code refactoring

CodeRefactor: 리팩토링을 직접 수행하는 컴포넌트이다.
RecordService: Record 데이터를 이용하는 다른 컴포넌트와 Repository 및 Record데이터 간의 중재자 역할을 한다.
RecordRepository: Record 데이터와 직접 상호작용하여 관리하는 컴포넌트이다.



[Figure 14] Sequence diagram - Backend: Code refactoring

6. 프로토콜 디자인

6.1. Objectives

이 파트에서는 백엔드 서버와 프론트엔드 서버간의 통신시 사용되는 프로토콜에 대해 기술한다. 특히, 프로토콜 뿐 아니라, 데이터 교환 형식도 몇가지 소개한다. 추가적으로 통신 보안의 프로토콜도 함께 소개한다.

6.2. HTTP

HTTP(Hypertext Transfer Protocol)은 웹에서 가장 널리 사용되는 통신 프로토콜이다. HTTP는 네트워크 장치 간에 정보를 전송하도록 설계된 애플리케이션

계층 프로토콜이며 네트워크 프로토콜 스택의 다른 계층 위에서 실행된다. 일반적으로는 클라이언트 서버에서 백엔드 서버로 요청을 보내고, 이에 맞는 응답을 백엔드 서버가 반환하는 구조이다.

6.3. RESTful API

RESTful API(Representational State Transfer API)는 HTTP를 기반으로 하는 아키텍쳐 스타일이다. HTTP method(GET, POST, PUT, DELETE 등)을 사용하여 행동을 정의하고, URI로 리소스를 식별한다. URI에는 명사만 사용하는 컨벤션이 존재한다.

6.4. JSON

JSON(JavaScript Object Notation)는 데이터 교환 형식으로, 널리 사용되는 텍스트 기반의 경량 데이터 교환 포맷이다. 특히 위의 RESTful API의 요청/응답 형식으로 자주 사용된다.

6.5. Form-data

Form-data는 주로 웹 폼에서 파일 업로드와 같은 멀티파트 데이터를 서버로 전송할때 사용되는 데이터 교환 형식이다. 멀티파트 형식은 각각의 파트가 경계 문자열로 구분된다. 각 파트는 헤더와 본문으로 구성되고, 파일 업로드의 경우 파일의 메타데이터와 파일 내용이 포함된다. bitCO2e에서는 유저가 광고 신청시, 웹 어플리케이션에 업로드 할 이미지 파일을 서버에 Form-data 형식을 활용하여 전송한다.

6.6. SSL/TLS

HTTPS는 HTTP를 암호화 프로토콜을 추가적으로 사용하여 통신을 암호화한다. 유명 암호화 프로토콜에는 ssl과 tls가 있다. 이들은 클라이언트와 서버 간에 전송되는 데이터를 암호화하여 도청, 데이터 변조, 위조 등의 공격을 방지한다. 특히 최근 웹브라우저는 HTTP 통신을 허용하지 않기 때문에, bitCO2e 배포시, 인증서를 발급하여 HTTPS 프로토콜을 구성할 예정이다.

6.7. API

6.7.1. Signup

Request	Detail	
URI	/signup	
Method	POST	
Request Body	username	User unique name
	email	User email
	password	User password
Response	Status code	Description
Success	201	User successfully created
Failed	400	Missing required fields

	401	Username is conflicted
Failed	500	Internal server error

[Table 1] Signup API

6.7.2. Login

Request	Detail	
URI	/login	
Method	POST	
Request Body	username	User unique name
	password	User password
Response	Status code	Description
Success	200	Login is successful
Failed	400	Missing required fields
	401	Invalid username or password
Failed	500	Internal server error

[Table 2] Login API

6.7.3. Refactoring

Request	Detail	
URI	/refactoring	
Method	POST	
Request Body	code	Code text
Response	Status code	Description
Success	200	Code refactored successfully
Failed	400	Invalid code is provided
Failed	500	Internal server error

[Table 3] Refactoring API

6.7.4. Get user’s bit info

Request	Detail	
URI	/bit	
Method	GET	
Request Query String	username	specific username
Response	Status code	Description
Success	200	User bit is successfully returned
Failed	400	Invalid username is provided
	401	Authentication required
	404	User not found
Failed	500	Internal server error

[Table 4] Get bit info API

6.7.5. Buy advertisement

Request	Detail	
URI	/advertisement	
Method	POST	
Request Body	username	specific username
	current_bit	current bit of user
	used_bit	used bit for advertisement
	message	advertisement message
	image	advertisement image
Response	Status code	Description
Success	200	Advertisement application is successfully created
Failed	400	Missing required fields
	401	Authentication required
	413	Image file size exceeds the limit
Failed	500	Internal server error

[Table 5] Buy advertisement API

6.7.6. Review advertisement

Request	Detail	
URI	/advertisement	
Method	POST	
Request Query String	Advertisement id	Specific advertisement id
Response	Status code	Description
Success	200	Application reviewed successfully
Failed	400	Missing required query parameter
	403	User does not have permission
	404	Application not found
Failed	500	Internal server error

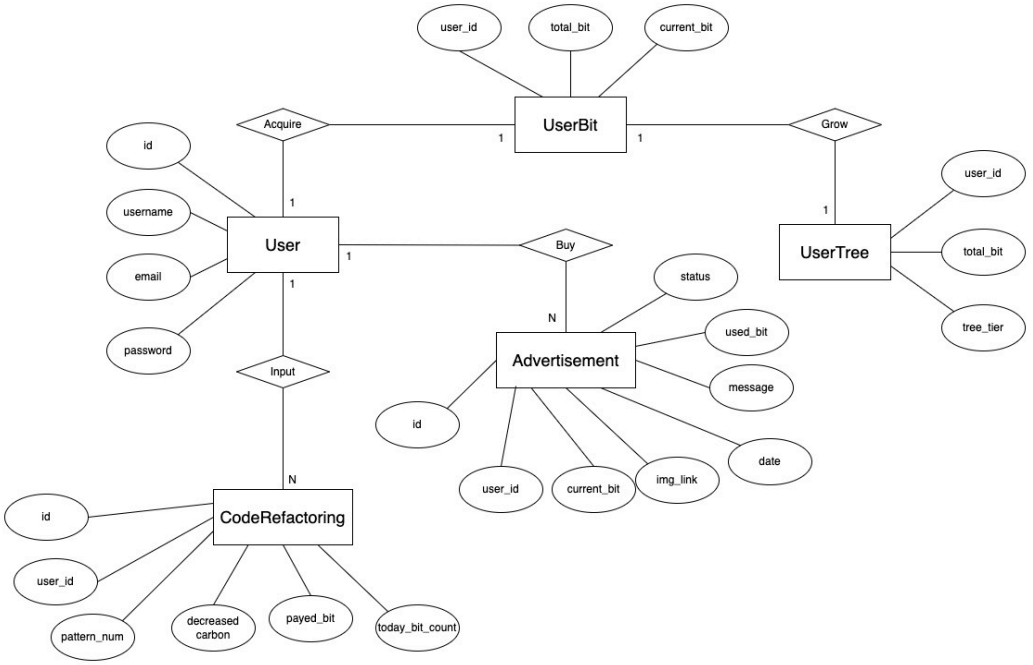
[Table 6] Review advertisement API

7. 데이터베이스 디자인

7.1. Objectives

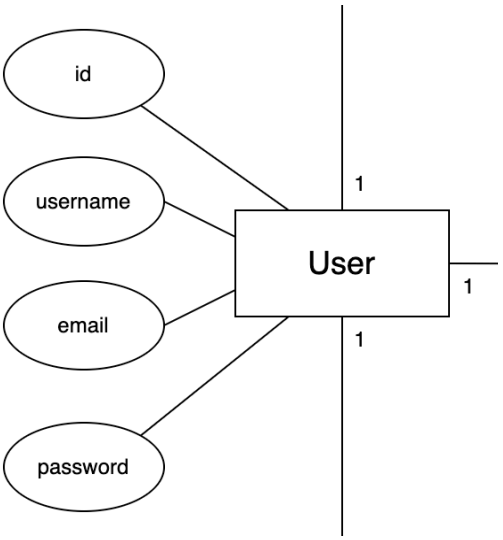
이 파트에서는 시스템 데이터 구조가 어떻게 데이터베이스로 구성되고 표현되는지에 대해 설명한다. ERD를 통해 전체적인 entity와 그들간의 관계를 식별하고, 관계형 Schema 및 DDL을 제시함으로써 bitCO2e의 데이터베이스를 소개한다.

7.2. ER Diagram



[Figure 15] ER Diagram

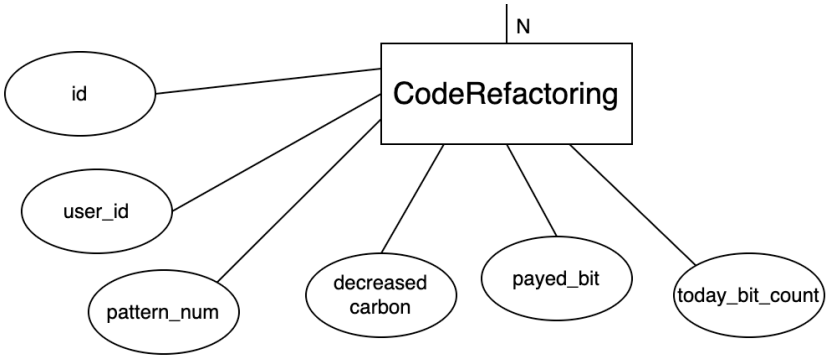
7.2.1. User



[Figure 16] User Entity

User Entity는 bitCO2e에 회원가입한 사용자에 해당된다. Id, username, email, password를 attribute로 가지고, id는 primary key이다. Id는 auto_increment의 특징을 가진다.

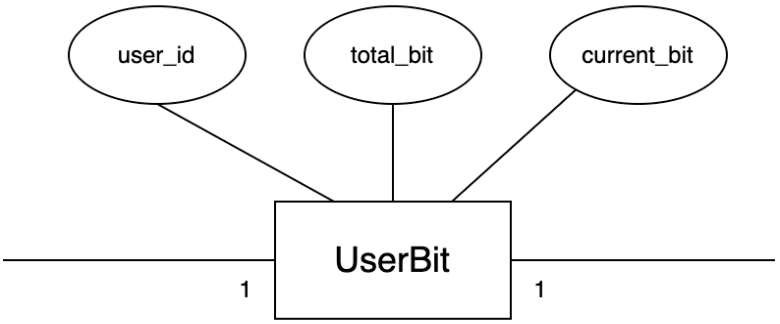
7.2.2. CodeRefactoring



[Figure 17] CodeRefactoring Entity

CodeRefactoring은 사용자가 입력한 코드를 리펙토링 하였을때 리펙토링 결과를 저장한다. Id, user_id, pattern_num, decreased_carbon, payed_bit, today_bit_count를 attribute로 가진다. Id가 primary key이며 auto_increment의 특징을 가진다. User_id는 User 테이블의 id를 참조하여 foreign key이다. User과 CodeRefactoring은 유저가 여러번 코드 리펙토링 작업을 할 수 있기 때문에 1:N의 관계이다.

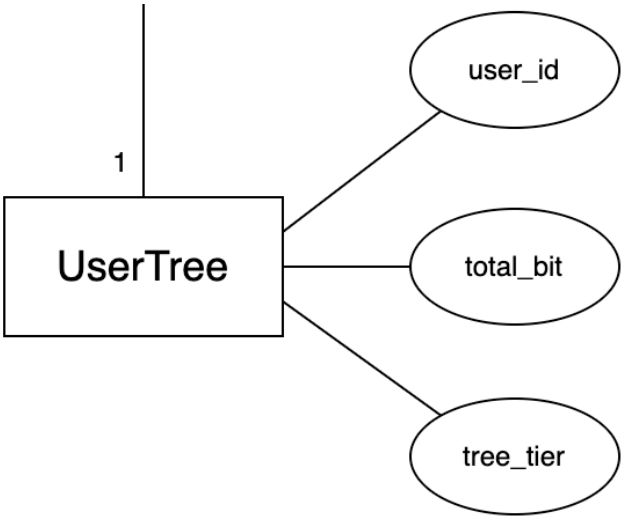
7.2.3. UserBit



[Figure 18] UserBit Entity

UserBit은 사용자가 코드 리펙토링으로 인해 지급받은 Bit 포인트를 저장한다. User_id, total_bit, current_bit를 attribute로 가지며, user_id는 User 테이블의 id를 참조하여 foreign key이자 primary key이다. User와 UserBit는 1:1의 관계이다.

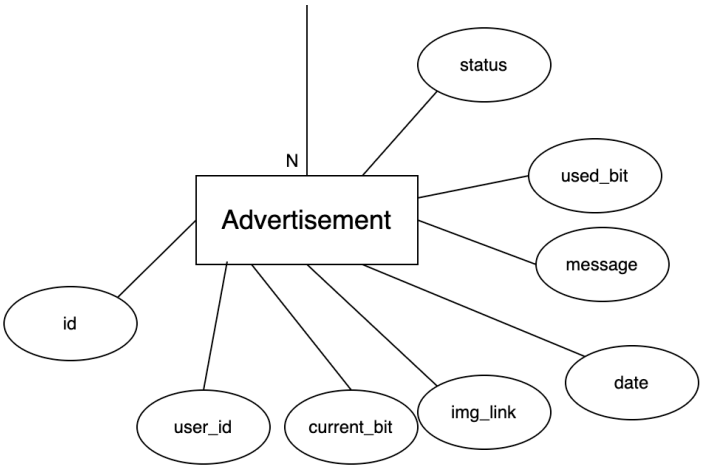
7.2.4. UserTree



[Figure 19] UserTree Entity

UserTree는 사용자의 누적 bit포인트를 바탕으로 기록되는 사용자 가상 나무를 저장하는 Entity이다. User_id, total_bit, tree_tier를 attribute로 가지며, user_id는 User 테이블의 id를 참조하여 foreign key이자 primary key이다. UserBit와 UserTree는 1:1의 관계이다.

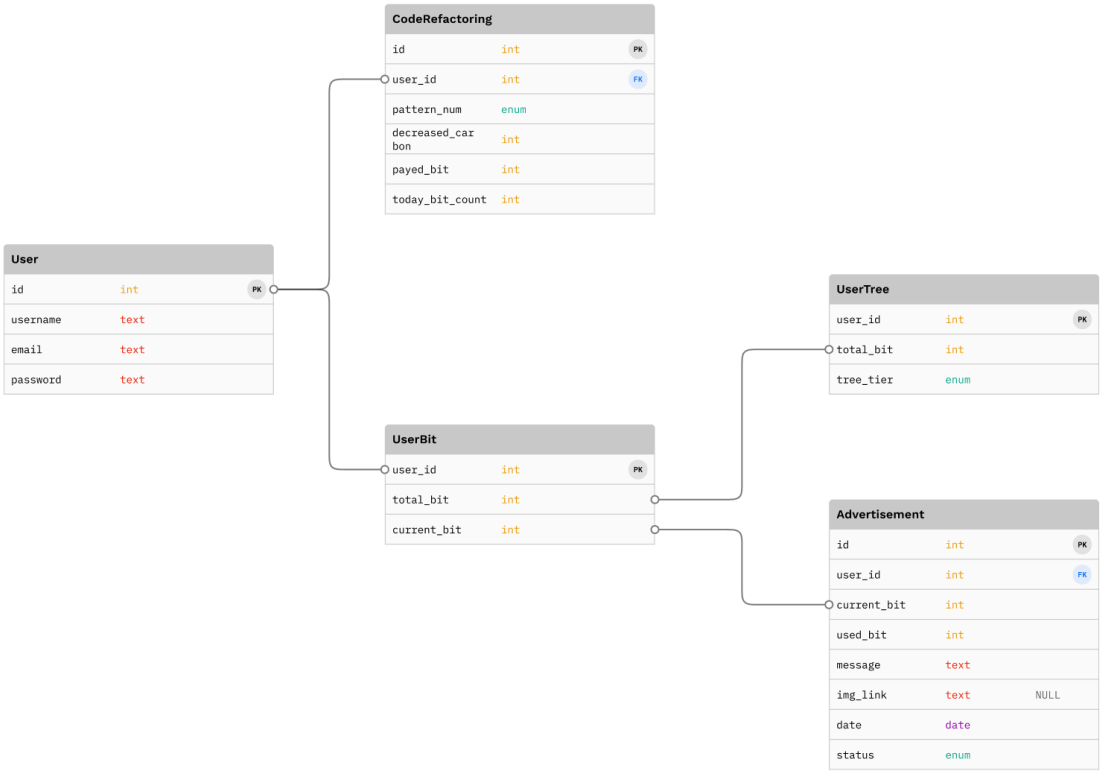
7.2.5. Advertisement



[Figure 20] Advertisement Entity

Advertisement는 사용자의 현재 bit포인트를 사용하여 등록할 수 있는 광고 데이터를 저장한다. Id, user_id, current_bit, used_bit, message, img_link, status, date를 attribute로 가진다. Id는 primary key이며 auto_increment의 특징을 가진다. User_id는 User 테이블의 id를 참조하여 foreign key이다. 유저는 여러번 광고를 구매할 수 있기때문에, User과 Advertisement는 1:N의 관계이다.

7.3. Relational Schema



[Figure 21] Relational Schema

7.4. SQL Data Definition Language

7.4.1. User

```
CREATE TABLE User (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL,
    email VARCHAR(255) NOT NULL,
    password VARCHAR(255) NOT NULL
);
```

7.4.2. CodeRefactoring

```
CREATE TABLE CodeRefactoring (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    pattern_num ENUM('1', '2', '3'),
    decreased_carbon INT,
    payed_bit INT,
    today_bit_count INT,
    FOREIGN KEY (user_id) REFERENCES User(id)
);
```

7.4.3. UserBit

```
CREATE TABLE UserBit (
    user_id INT PRIMARY KEY,
    total_bit INT,
    current_bit INT,
    FOREIGN KEY (user_id) REFERENCES User(id)
);
```

7.4.4. UserTree

```
CREATE TABLE UserTree (
    user_id INT PRIMARY KEY,
    total_bit INT,
    tree_tier ENUM('seed', 'sprout', 'sapling', 'young tree', 'mature tree'),
    FOREIGN KEY (user_id) REFERENCES User(id),
    FOREIGN KEY (total_bit) REFERENCES UserBit(total_bit)
);
```

7.4.5. Advertisement

```
CREATE TABLE Advertisement (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    current_bit INT,
    used_bit INT,
    message VARCHAR(255),
```

```
    img_link VARCHAR(255),
    date DATE,
    status ENUM('applied', 'approved', 'rejected'),
    FOREIGN KEY (user_id) REFERENCES User(id),
    FOREIGN KEY (current_bit) REFERENCES UserBit(current_bit)
);
```

8. 테스팅 계획

8.1. Objectives

본 시스템에 대한 단계별 테스트 계획에 대해 설명한다. 테스트 종류, 환경, 계획, 시나리오를 계획한다.

8.2. Testing Policy

8.2.1. Test Strategy

8.2.1.1. Development Testing

개발 단계에서 수행되고 시스템의 품질을 개선하고 결함을 조기에 발견하여 개발 과정의 효율성을 향상시키기 위한 과정이다. 개발자가 코드를 작성하고 변경할 때마다 지속적으로 수행되어야 한다. Unit testing, integration testing, functional testing, regression testing을 포함한다.
Backend에서는 JUnit, Spring Test, Spring MVC Test 를 이용하여 모든 testing을 수행할 수 있도록 한다. Frontend에서는 Jest나 React Testing Library를 이용하여 testing을 진행한다. 그리고 Cypress를 이용하여 end-to-end testing을 할 수 있도록 한다.

8.2.1.2. Release Testing

시스템이 배포되기 전에 수행되어야 하는 테스트 과정이다. 최초 배포 시뿐만 아니라 새로운 버전의 시스템을 배포하기 전에 항상 수행되어야 한다. 시스템의 모든 기능이 하자 없이 정상적으로 작동하는지 확인하여야 하며. 소프트웨어의 품질을 검증하는 것을 목표로 한다.
본 시스템은 우선 알파버전으로 배포하여 내부 테스팅을 진행하고, 이후 User Testing을 위한 베타버전으로 배포할 예정이다.

8.2.1.3. User Testing

본 시스템을 실제 사용하는 사용자에게 제공하여 시스템의 사용자 경험(UX)와 기능을 평가하는 과정이다. 함께 수업을 수강하는 학우들과 개발팀원의 지인들을 모아 테스트 유저 그룹을 만들고 베타버전 시스템을 배포하여 피드백 받을 수 있도록 한다. 이때 시나리오를 함께 전달하여 원활한 테스팅이 이루어질 수 있도록 한다.

8.2.2. Test Environment

development testing 에서는 개발자가 각자의 개발 환경에서 테스트를 진행하는 것으로 한다. release testing 에서는 명세서에서 설정한 최소한의 요구사항을 만족하는 환경에서 시스템이 작동하는 지 확인해야 한다. 이를 위해 virtual machine을 활용하여 가상의 작동환경을 구축하여 테스트를 수행한다. 그 최소한의 요구사항은 아래와 같다

(1) 1GHz 이상의 프로세서 (Core i3 또는 그 이상)
(2) 1GB(32 비트) 또는 2GB(64비트) 이상의 RAM
(3) 16GB(32비트) 또는 32GB(64비트) 이상의 하드 드라이브 공간
(4) DirectX 9 이상(WDDM 1.0 드라이버 포함) 호환 그래픽 카드
(5) 800x600 이상의 디스플레이
(6) 인터넷 연결
(7) 마우스, 키보드 등의 적절한 HID 연결

위 요구사항들은 Windows 10 운영체제에서 Chrome 브라우저를 실행하여 원활한 웹서핑을 하기 위한 최소한의 조건들이다.

8.2.3. Test Schedule

- Release Testing: 2024-05-25 ~ 2024-05-31
- User Testing: 2024-06-01 ~ 2024-06-07

8.2.4. Test Scenarios

테스트 시나리오는 실제 시스템 운용 단계에서 일어날 수 있는 상황들을 미리 가정하여 효과적인 검증이 이루어질 수 있도록 한다. 기본적인 테스트 시나리오를 아래에 기술하였으며 경우에 따라 시나리오 수정, 추가가 가능하다.

8.2.4.1. 비회원 Refactoring 요청 시나리오

Description

비회원 유저가 로그인을 하지 않은 상태에서 Refactoring 요청을 하는 경우를 검증한다.

전제조건

1. 사용자는 웹페이지에 로그인 되어 있지 않아야 한다.

시나리오 단계

1. 사용자가 refactoring 페이지에 접속한다.
2. 코드 입력 란에 최적화를 원하는 코드를 입력한다.
3. 코드 최적화 결과를 사용자에게 제공한다.

8.2.4.2. 회원 Refactoring 요청 시나리오

Description

회원 유저가 로그인을 한 상태에서 refactoring 요청을 하는 경우를 검증한다.

전제조건

1. 사용자는 웹사이트에 회원가입이 되어 있어야 한다.
2. 사용자는 웹사이트에 로그인 되어 있어야 한다.

시나리오 단계

1. 사용자가 웹사이트에 로그인한다
2. 사용자가 refactoring 페이지에 접속한다.
3. 코드 입력 란에 최적화를 원하는 코드를 입력한다
4. 코드 최적화 결과를 유저에게 제공하고 사용자가 새로 획득한 bit 포인트를 확인할 수 있다.

8.2.4.3. 회원 Dashboard 조회 시나리오

Description

회원 유저가 로그인을 한 상태에서 자신의 활동정보를 조회하는 경우를 검증한다.

전제조건

1. 사용자는 웹사이트에 회원가입이 되어 있어야 한다.
2. 사용자는 웹사이트에 로그인 되어 있어야 한다.

시나리오 단계

1. 사용자가 웹사이트에 로그인한다
2. 사용자가 dashboard 페이지에 접속한다.
3. 사용자의 refactoring 요청 내역, 보유 bit 포인트, tree를 제공한다.

8.2.4.4. 광고 등록 요청 시나리오

Description

회원 유저가 로그인을 한 상태에서 자신의 bit 포인트를 소모하여 광고등록 요청을 보내는 경우를 검증한다.

전제조건

1. 사용자는 웹사이트에 회원가입이 되어 있어야 한다.
2. 사용자는 웹사이트에 로그인 되어 있어야 한다.

시나리오 단계

1. 사용자가 웹사이트에 로그인한다.
2. 사용자가 dashboard 페이지에 접속한다.
3. 사용자가 광고등록 버튼을 클릭하여 광고등록 페이지로 이동한다.
4. 광고에 필요한 정보들을 기입하고 등록 요청을 보낸다
5. 광고 등록 요청이 성공했음을 유저에게 알린다.

8.2.4.5. 광고 허용 시나리오

Description

관리자가 일반 사용자들이 신청한 광고 등록 요청을 확인하고 허용하는 경우를 검증한다.

전제조건

1. 사용자가 관리자 계정으로 웹사이트에 로그인 되어 있어야 한다.

시나리오 단계

1. 사용자가 웹사이트에 관리자 계정으로 로그인한다.
2. 사용자는 관리자 페이지에 접속한다.
3. 사용자는 일반 사용자들로 부터 접수된 광고등록 요청들을 확인할 수 있다.
4. 사용자는 원하는 광고등록 요청을 허용한다.

5. 해당 광고를 등록요청한 사용자는 자신의 dashboard 페이지에서 허용사실을 확인할 수 있다.

9. 개발 계획

9.1. Objectives

개발 계획 파트에서는 BitCO2e 시스템의 개발환경 및 기술 스택, 디자인과 시스템 실행에 관한 제약사항을 기술한다.

9.2. Frontend Environment and Tools

9.2.1. HTML

HTML은 HyperText Markup Language의 약자로, 웹 페이지의 구조를 정의하는 마크업 언어이다.문서의 내용 이외의 문서 구조, 서식 등을 포함한다. 또한 HTML 요소는 텍스트, 이미지, 링크 등의 웹페이지 내용을 구조화 하며 브라우저는 이를 해석하여 화면에 표시한다.

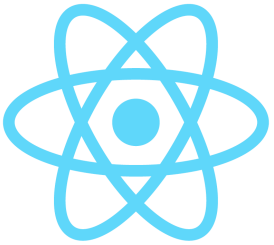
9.2.2. CSS

CSS는 Cascading Style Sheets의 약자로, HTML로 작성된 웹페이지의 스타일을 지정하여 웹페이지를 꾸미는 언어이다. CSS는 선택자와 속성, 속성값을 활용해 요소의 외형을 정의한다. CSS는 여러 스타일 시트의 규칙을 병합하고 우선 순위를 설정하는 “cascading” 방식으로 동작한다.

9.2.3. Javascript

Javascript는 객체 기반의 스크립트 프로그래밍 언어이다. 주로 웹 브라우저 내에서 사용되며 웹 페이지에 동적인 기능을 추가하기 위해 사용된다. JavaScript는 브라우저에서 실행되어 사용자 인터페이스의 상호작용을 가능하게 한다. 웹 페이지가 로드된 이후에도 코드를 실행할 수 있어 동적 콘텐츠 생성에도 사용된다.

9.2.4. React



리액트는 사용자 인터페이스를 구축하기 위한 JavaScript 라이브러리이다. 리액트는 컴포넌트를 기반으로 한 아키텍처를 사용해 재사용 가능한 UI 구성 요소를 만들고 관리한다. 또한 리액트는 단방향 데이터 흐름과 가상 DOM을 사용해 성능을 최적화 한다. 싱글 페이지 어플리케이션과 모바일 어플리케이션 개발에 주로 사용된다.

9.3. Backend Environment and Tools

9.3.1. Spring

스프링은 자바 플랫폼을 위한 오픈 소스 어플리케이션 프레임워크다. 스프링은 dependency injection을 통해 어플리케이션의 구성요소를 관리하며, 다양한 모듈을 활용해 데이터 접근, 트랜잭션 관리, 웹 어플리케이션 개발 등을 지원한다. 동적인 웹 사이트 개발을 위한 여러 서비스도 제공한다.

9.3.2. Firebase

파이어베이스는 구글에서 제공하는 클라우드 기반 어플리케이션 개발 플랫폼으로, 실시간 데이터 베이스, 인증, 호스팅, 클라우드 메시징 등의 다양한 백엔드 서비스를 제공한다. BitCO2e 서비스에서는 파이어베이스에서 제공하는 데이터베이스 서비스인 firebase realtime database와 firebase firestore를 활용해 어플리케이션 요구사항을 충족시킨다. 회원 정보 등의 일반 데이터 관리를 위해서는 firebase realtime database가 사용되며 광고 이미지 저장 및 관리를 위해서는 firebase firestore를 활용한다.

9.4. Constraints

‘BitCO2e’ 서비스의 디자인과 개발, 사용에 있어 요구되는 세부적인 제약사항은 아래와 같다.

- Refactoring 대상 언어는 Java로 한다.
- 사용자 정보 보호를 위해 로그인 정보 등 개인 정보는 서버에서 암호화하여 저장하고 처리한다.
- 사용자 편의적이고 직관적인 UI를 지향한다.

- 사용자 수 증가, 지원 언어 증가, 기능 증가 등 추후 시스템의 확장 가능성을 염두에 둔다.
- 시스템 유지보수에 용이하도록 일관성과 가독성을 유지한다.
- 악의적 사용자로부터 비정상적인 시스템 이용을 고려한다.
- BitCO2e 시스템은 web application으로 web browser가 동작할 수 있는 환경에서만 실행이 가능하다.
- 사용자의 입력 코드에 대해서 단 하나의 refactoring 결과 코드를 제공해야한다.
- 사용자의 광고권 구매로 업로드 되는 광고는 상업적 목적이 없는 개인적인 광고만을 허용한다.
- 사용자가 입력한 자바코드를 리팩토링하고 사용자에게 리팩토링 코드와 분석결과를 제공하는데 5초 이상의 시간이 소요되어서는 안된다.
- 시스템 자원 낭비를 최소화 하도록 소스 코드를 최적화한다.
- 개발은 윈도우10 이상에서 이루어져야 하며 윈도우 10, 11 환경을 타겟으로 한다.

9.5. Assumptions and Dependencies

BitCO2e 시스템은 웹 기반으로 제공되는 서비스이며 사용자가 적절한 입출력 장치와 통신 기능을 가진 기기를 통해 서비스를 이용할 것으로 가정한다. 따라서, 시스템이 제공하는 서비스 외 네트워크 연결 및 기타 주변 장치의 문제에 대해서는 고려하지 않는다. 기준이 되는 Windows 10 환경과 Chrome 브라우저의 최소 시스템 요구사항은 다음과 같다.

- 1GHz 이상의 프로세서 (Core i3 또는 그 이상)
- 1GB(32비트) 또는 2GB(64비트) 이상의 RAM
- 16GB(32비트) 또는 32GB(64비트) 이상의 하드 드라이브 공간
- DirectX 9 이상(WDDM 1.0 드라이버 포함) 호환 그래픽 카드
- 800x600 이상의 디스플레이

시스템은 상기한 요구사항들을 만족하고 인터넷에 연결되어 있으며 마우스, 키보드 등의 적절한 HID를 갖춘 사용자의 접근을 가정하고, 그러한 경우에 올바른 작동을 보장한다.