



# GREEN CODERS

JAVA 코드 그린이 플랫폼

Team 7

김유빈, 나건우, 성봉진, 이재빈, 정한샘, 최장섭

SECTION 01

# 시스템 구성

# What is “Green Coders”



Best Refactoring Pattern

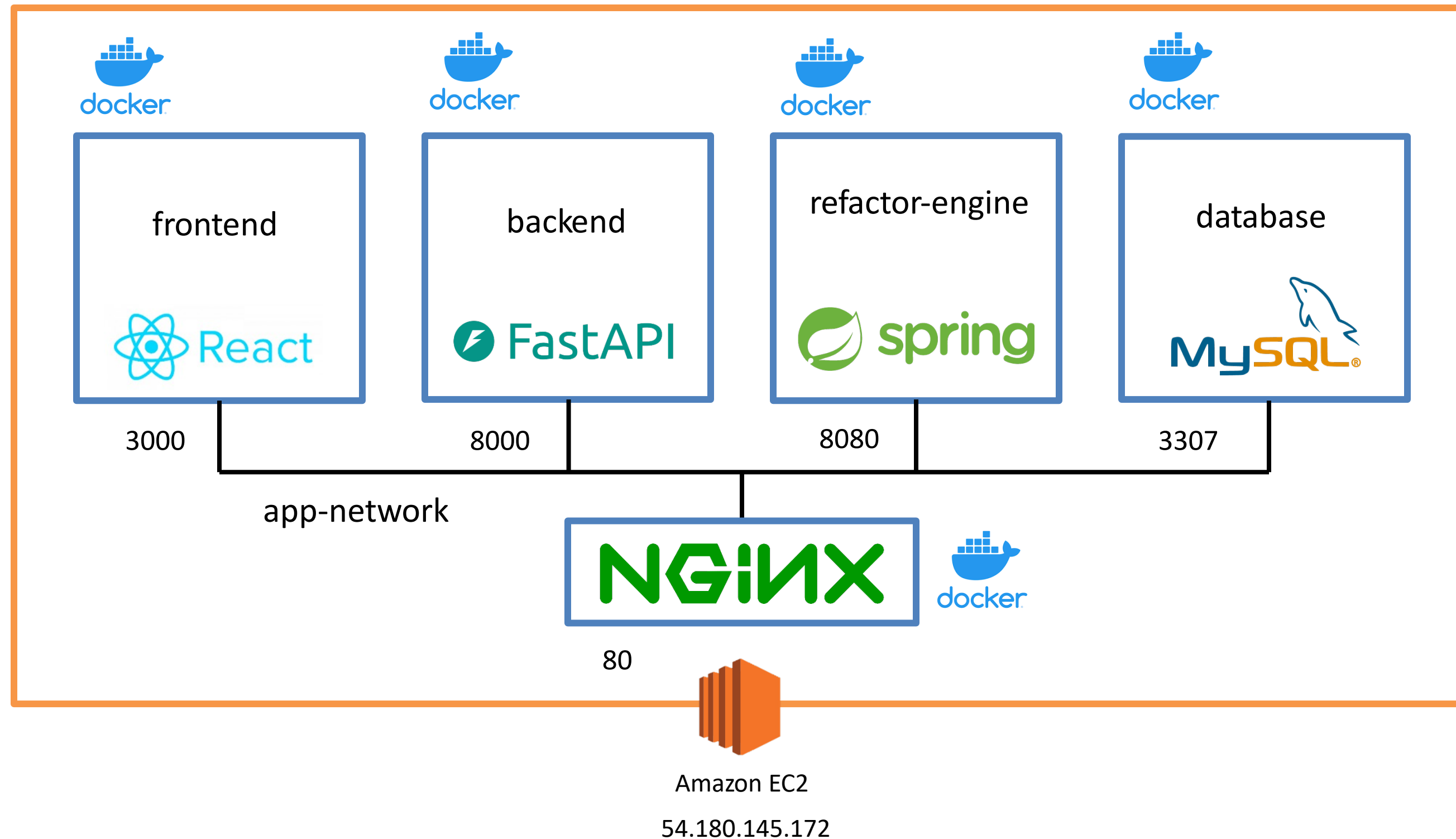


Dashboard



환경 기여 인지

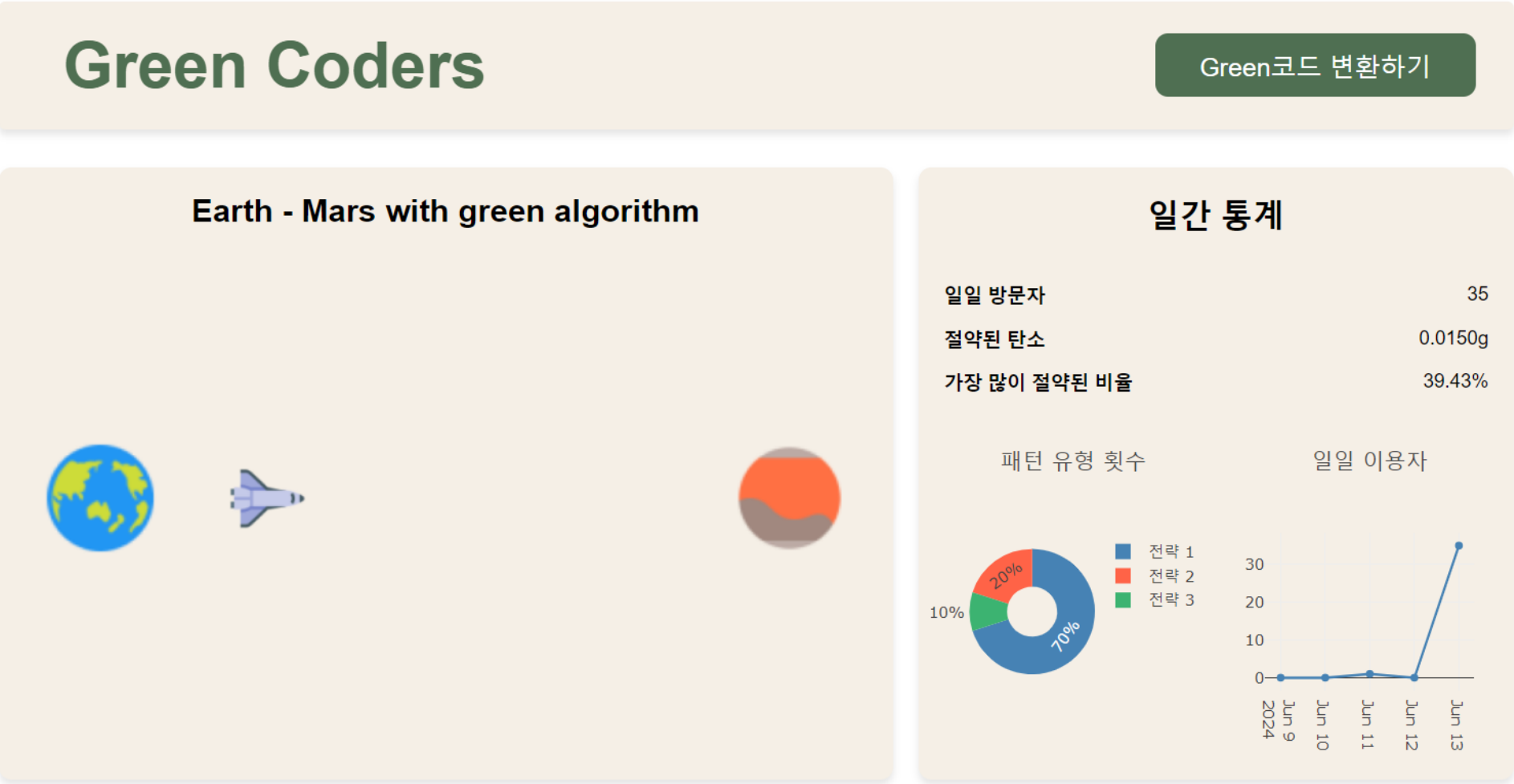
# System Architecture



SECTION 02

# 인터페이스

About



# User Interface



## 패턴 별 코드 개별 제공

1. 효과적인 코드 패턴 공유
2. 직관적인 이해

# User Interface

## About Green Coders

### What is Green Coders?

Green Coders에서 그린화 패턴을 적용하고 싶은 코드를 입력하면 탄소배출량과 함께 리팩토링한 코드를 제공한다. 이를 통해 사용자는 더 적은 탄소를 배출하는 친환경적인 코드를 얻을 수 있다.

### Why should we use Green coders?

최근 전산 과학 분야에서 탄소 배출량이 기하급수적으로 증가하면서, 탄소 배출을 줄이는 알고리즘 사용의 중요성이 높아지고 있다. 코드 실행 시 발생하는 탄소 배출량을 측정함으로써, 개발자들이 소프트웨어의 탄소 발자국을 인식하고, 환경 친화적인 개발을 장려할 수 있도록 돕는다.

### Carbon Emission Formula

$$C = E \times CI$$

$$E = t \times (nc \times Pc \times uc + nm \times Pm) \times PUE \times 0.001$$

$$C = t \times (nc \times Pc \times uc + nm \times Pm) \times PUE \times CI \times 0.001$$

Close

## 신고 사유

부적절한 내용이 포함되어 있습니다.

신고하기

닫기

## 구체적인 글은 모달 활용

1. 중요 정보, 기능 분리 제공
2. 시각화 효과에 방해 X



## SECTION 03

# API

# API Details

Clear RESTful API

**BuggyCode**

**FixedCode**

**FixStrategy**

**Report**

# API Details

## Clear RESTful API

**BuggyCode**

**FixedCode**

**FixStrategy**

**Report**

생성

**POST** /buggyCodes

**POST** /fixedCodes

**POST** /fixStrategy

**POST** /reports

# API Details

## Clear RESTful API

**BuggyCode**

**FixedCode**

**FixStrategy**

**Report**

생성

**POST** /buggyCodes

**POST** /fixedCodes

**POST** /fixStrategy

**POST** /reports

조회

**GET** /buggyCodes

**GET** /fixedCodes

**GET** /fixStrategy

**GET** /reports

**GET** /buggyCodes/{id}

**GET** /fixedCodes/{id}

**GET** /fixStrategy/{id}

**GET** /reports/{id}

# API Details

## Clear RESTful API

**BuggyCode**

**FixedCode**

**FixStrategy**

**Report**

생성

**POST** /buggyCodes

**POST** /fixedCodes

**POST** /fixStrategy

**POST** /reports

조회

**GET** /buggyCodes

**GET** /fixedCodes

**GET** /fixStrategy

**GET** /reports

검색

**GET** /buggyCodes/{id}

**GET** /fixedCodes/{id}

**GET** /fixStrategy/{id}

**GET** /reports/{id}

**GET** /fixedCodes/strategy/{id}

**GET** /fixedCodes/buggyCode/{id}

**GET** /reports/fixedCode  
/{id}

# API Details

## Clear RESTful API

### BuggyCode

### FixedCode

### FixStrategy

### Report

생성

POST /buggyCodes

POST /fixedCodes

POST /fixStrategy

POST /reports

조회

GET /buggyCodes

GET /fixedCodes

GET /fixStrategy

GET /reports

검색

GET /buggyCodes/{id}

GET /fixedCodes/{id}

GET /fixStrategy/{id}

GET /reports/{id}

삭제

GET /fixedCodes/strategy/{id}

GET /fixedCodes/buggyCode/{id}

DELETE /fixedCodes/{id}

GET /reports/fixedCode/{id}

DELETE /reports/{id}

DELETE /reports

/fixedCode/{id}

# API Details

## More Stable Response

```
{  
  result: HttpStatus==2XX 인 경우, "success",  
    그외의 경우, "fail"을 저장합니다.  
  
  onSuccess: result==success 인 경우, API가 반환한 데이터를 저장합니다.  
  onError: result==fail 인 경우, 예외의 타입과 메시지를 저장합니다.  
}
```

# API Details

**API Docs is release!**

**More details in...**

**[Green-Coders Main API Docs](#)**



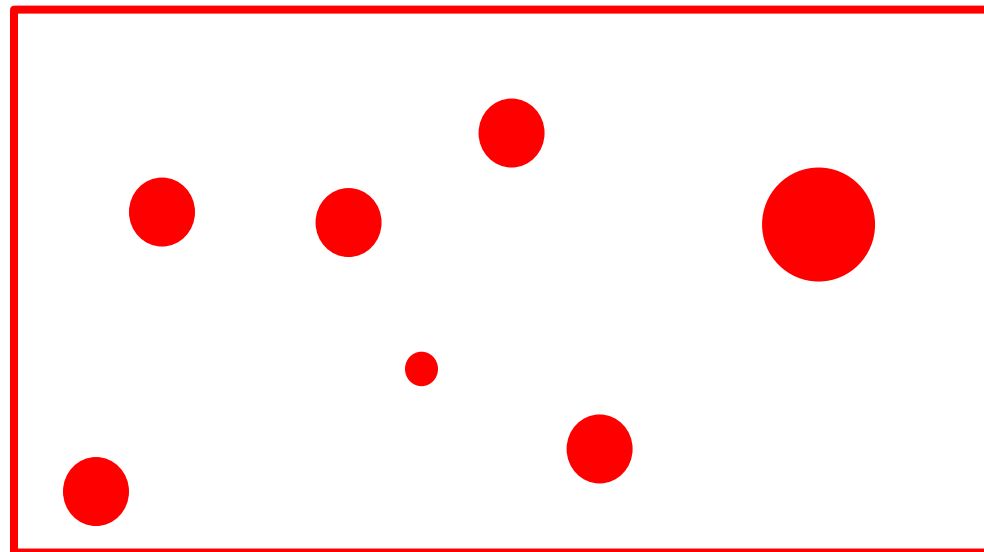
SECTION 04

# 그린화 리팩터링

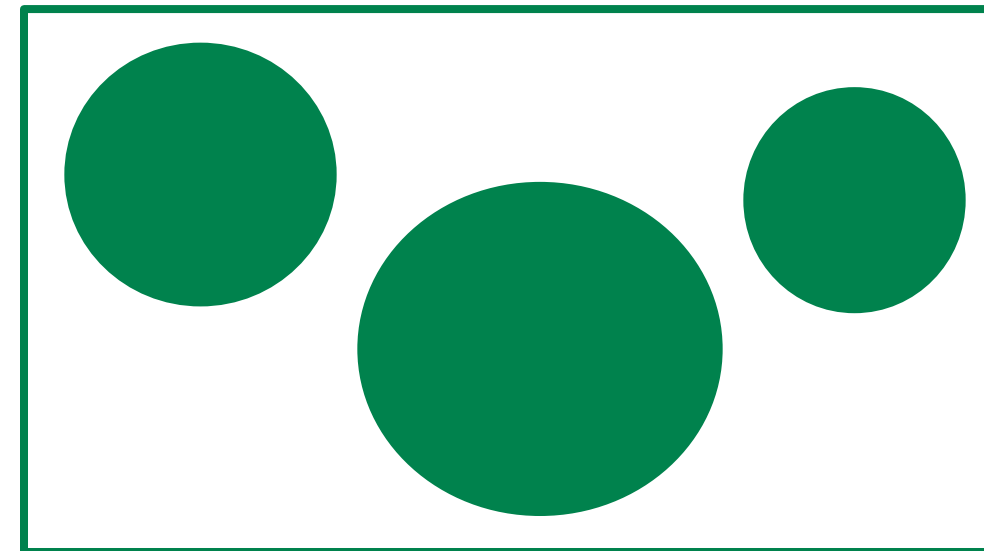
# Refactoring Details

The thing is **coverage** in real world!

Many strategies but, Small coverage



Less Strategy but, Large coverage



# Refactoring Details

**Refactoring 1** (baseline)  
duplicated *.size()*  
in iteration statement

```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<arr.size(); i++) {  
            System.out.println("Hello");  
        }  
    }  
}
```

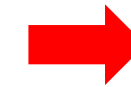


```
public class Fixed {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        int arrSize = arr.size();  
        for(int i=0; i<arrSize; i++) {  
            System.out.println("Hello");  
        }  
    }  
}
```

# Refactoring Details

**Refactoring 1** (baseline)  
duplicated `.size()`  
in iteration statement

```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<arr.size(); i++) {  
            System.out.println("Hello");  
        }  
    }  
}
```



```
public class Fixed {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        int arrSize = arr.size();  
        for(int i=0; i<arrSize; i++) {  
            System.out.println("Hello");  
        }  
    }  
}
```

```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<10; i++) {  
            for (int j=0; j<arr.size(); ++j) {  
                System.out.println("Nested hello");  
            }  
            System.out.println("Hello");  
        }  
    }  
}
```



Cannot refactor! 😞  
go to infinite loop

# Refactoring Details

Supports all cases of nested For-statement!

## Refactoring 1 (Green-Coders)

duplicated *.size()*  
in iteration statement

```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<10; i++) {  
            for (int j=0; j<arr.size(); ++i) {  
                System.out.println("Nested hello");  
            }  
            System.out.println("Hello");  
        }  
    }  
}
```



```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<10; i++) {  
            int arrSize1 = arr.size();  
            for (int j=0; j<arrSize1; ++i) {  
                System.out.println("Nested hello");  
            }  
            System.out.println("Hello");  
        }  
    }  
}
```

# Refactoring Details

Supports all cases of nested For-statement!

Refactoring 1 (baseline)  
duplicated *.size()*  
in iteration statement

```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<10; i++) {  
            for (int j=0; j<arr.size(); ++j) {  
                System.out.println("Nested hello");  
            }  
            System.out.println("Hello");  
        }  
    }  
}
```



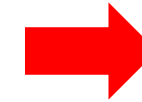
```
public class Buggy {  
    public static void main(String[] args) {  
        ArrayList<String> arr = new ArrayList<>();  
        arr.add("a");  
        arr.add("b");  
        arr.add("c");  
  
        for(int i=0; i<10; i++) {  
            int arrSize1 = arr.size();  
            for (int j=0; j<arrSize1; ++j) {  
                System.out.println("Nested hello");  
            }  
            System.out.println("Hello");  
        }  
    }  
}
```

New variable name is added **special signature**  
**It prevents duplicated variable name!**

# Refactoring Details

## Refactoring 2 (baseline) obsolete nested If statement

```
public class Buggy {  
    public static void main(String[] args) {  
        boolean cond1 = true;  
        boolean cond2 = false;  
        boolean cond3 = true;  
  
        if(cond1) {  
            if(cond2) {  
                if(cond3) {  
                    System.out.println("Hello");  
                }  
            }  
        }  
    }  
}
```



```
public class Buggy {  
    public static void main(String[] args) {  
        boolean cond1 = true;  
        boolean cond2 = false;  
        boolean cond3 = true;  
  
        if(cond1 && cond2 && cond3) {  
            System.out.println("Hello");  
        }  
    }  
}
```

# Refactoring Details

## Refactoring 2 (baseline) obsolete nested If statement

*in real world...*

```
public class Buggy {  
    public static void main(String[] args) {  
        boolean cond1 = true;  
        boolean cond2 = false;  
        boolean cond3 = true;  
        boolean cond4 = true;  
        boolean cond5 = true;  
  
        if(cond1) {  
            if(cond2) {  
                if(cond3) {  
                    System.out.println("Hello");  
                    if (cond5) {  
                        System.out.println("こんにちは");  
                    }  
                    System.out.println("你好");  
                }  
            }  
        }  
  
        if (cond4) {  
            if (cond5) {  
                System.out.println("안녕하세요");  
            }  
        }  
    }  
}
```

IF-statements are nested

**very complex!!!**





# Refactoring Details

## Refactoring 2 (Green Coders) obsolete nested If statement

*in real world...*

```
public class Buggy {  
    public static void main(String[] args) {  
        boolean cond1 = true;  
        boolean cond2 = false;  
        boolean cond3 = true;  
        boolean cond4 = true;  
        boolean cond5 = true;  
  
        if(cond1) {  
            if(cond2) {  
                if(cond3) {  
                    System.out.println("Hello");  
                    if (cond5) {  
                        System.out.println("こんにちは");  
                    }  
                    System.out.println("你好");  
                }  
            }  
        }  
  
        if (cond4) {  
            if (cond5) {  
                System.out.println("안녕하세요");  
            }  
        }  
    }  
}
```

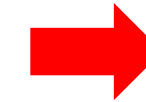
Green-Coders can refactor it! 😊

```
public class Buggy {  
    public static void main(String[] args) {  
        boolean cond1 = true;  
        boolean cond2 = false;  
        boolean cond3 = true;  
        boolean cond4 = true;  
        boolean cond5 = true;  
  
        if(cond1) {  
            if(cond2 && cond3) {  
                System.out.println("Hello");  
                if (cond5) {  
                    System.out.println("こんにちは");  
                }  
                System.out.println("你好");  
            }  
  
            if (cond4 && cond5) {  
                System.out.println("안녕하세요");  
            }  
        }  
    }  
}
```

# Refactoring Details

**Refactoring 3** (baseline)  
duplicated object creation  
in iteration body

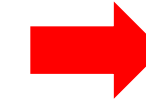
```
public class Buggy {  
    public static void main(String[] args) {  
        for(int i=0; i<10; i++) {  
            Greeting hello = new Greeting();  
            System.out.println(hello.message);  
        }  
    }  
}
```



```
public class Fixed {  
    public static void main(String[] args) {  
        Greeting hello = new Greeting();  
        for(int i=0; i<10; i++) {  
            System.out.println(hello.message);  
        }  
    }  
}
```

# Refactoring Details

```
public class Buggy {  
    public static void main(String[] args) {  
        for(int i=0; i<10; i++) {  
            Greeting hello = new Greeting();  
            System.out.println(hello.message);  
        }  
    }  
}
```



```
public class Fixed {  
    public static void main(String[] args) {  
        Greeting hello = new Greeting();  
        for(int i=0; i<10; i++) {  
            System.out.println(hello.message);  
        }  
    }  
}
```

**Refactoring 3** (baseline)  
duplicated object creation  
in iteration body

```
public class Buggy {  
    public static void main(String[] args) {  
        for(int i=0; i<10; i++) {  
            for (int j=0; j<10; ++j) {  
                for (int k=0; k<10; ++k) {  
                    Greeting hello = new Greeting();  
                    System.out.println(hello.message);  
                }  
            }  
        }  
    }  
}
```

How about nested loop?

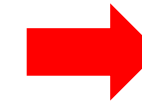


Cannot refactor! 😞

# Refactoring Details

**Refactoring 3** (baseline)  
duplicated object creation  
in iteration body

```
public class Buggy {  
    public static void main(String[] args) {  
        for(int i=0; i<10; i++) {  
            for (int j=0; j<10; ++j) {  
                for (int k=0; k<10; ++k) {  
                    Greeting hello = new Greeting();  
                    System.out.println(hello.message);  
                }  
            }  
        }  
    }  
}
```



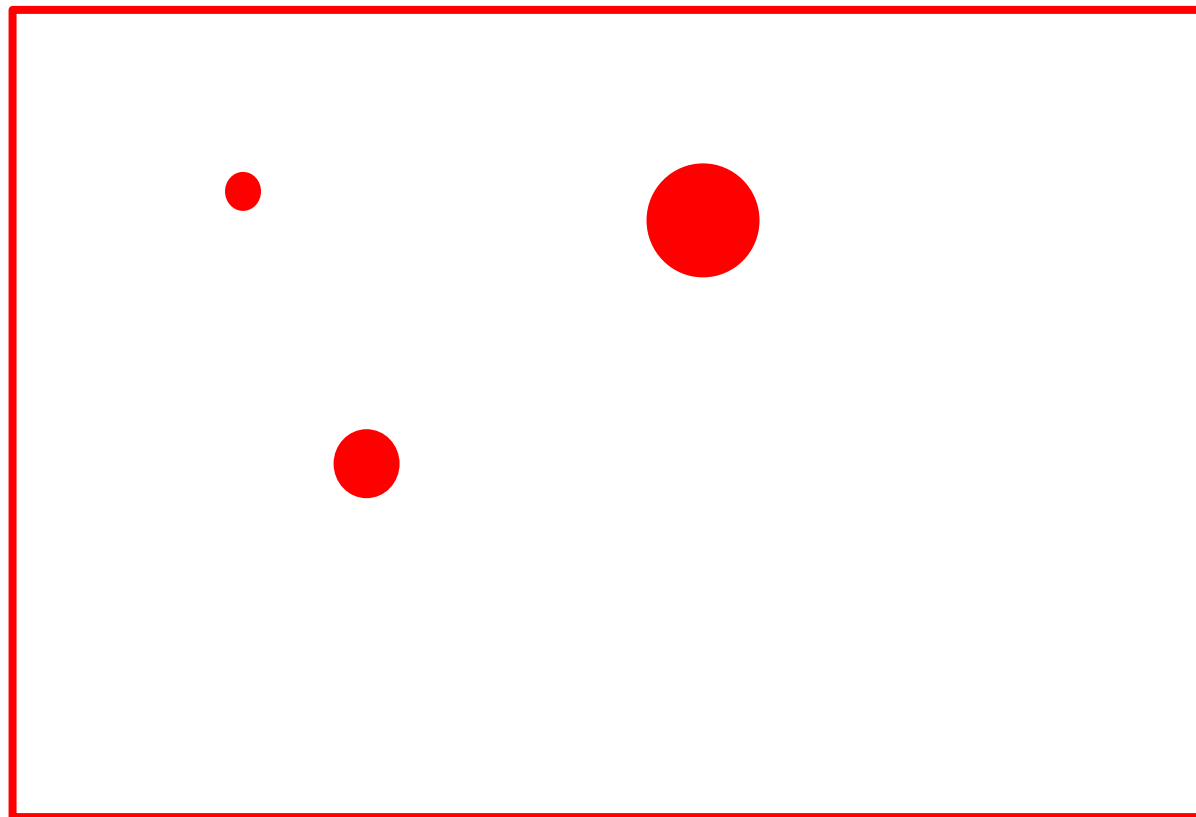
**Supports all cases of nested For-statement!**

```
public class Fixed {  
    public static void main(String[] args) {  
        Greeting hello = new Greeting();  
        for(int i=0; i<10; i++) {  
            for (int j=0; j<10; ++j) {  
                for (int k=0; k<10; ++k) {  
                    System.out.println(hello.message);  
                }  
            }  
        }  
    }  
}
```

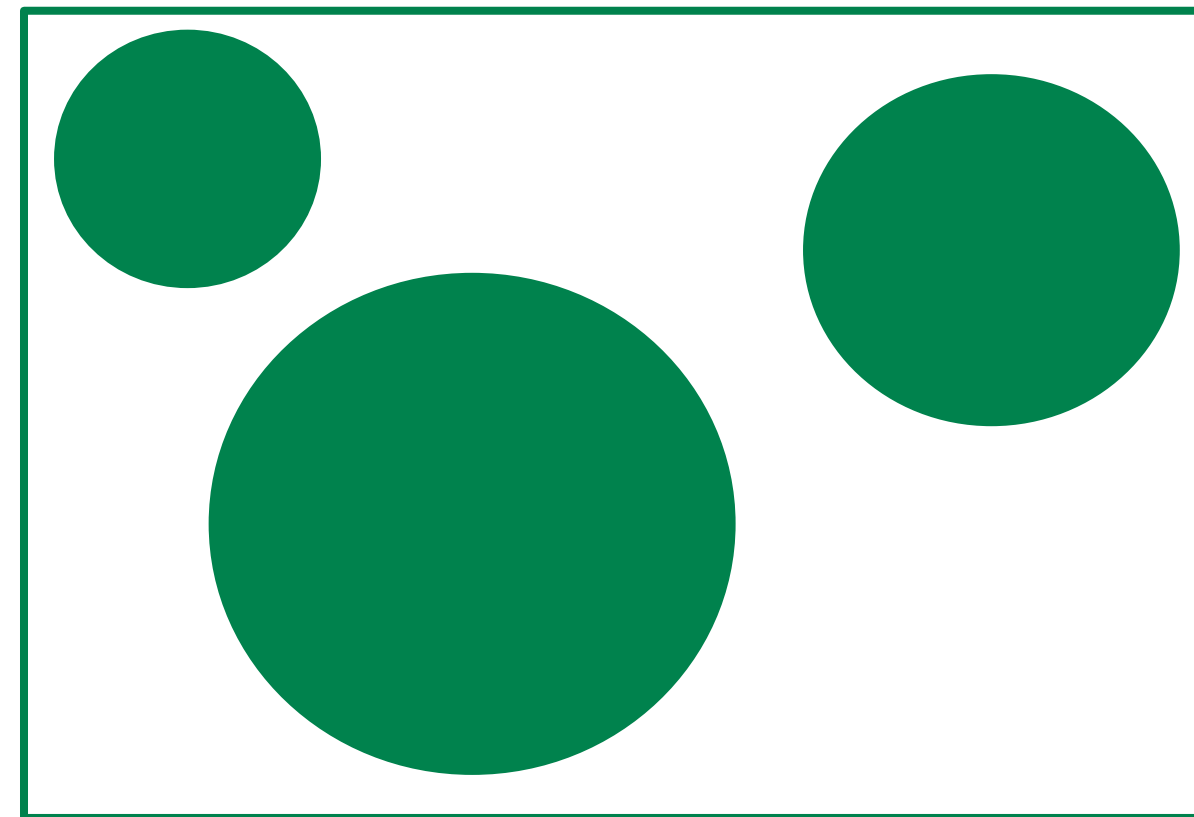
# Refactoring Details

## Green-Coders has Amazing Coverage!

baseline



Green-Coders



SECTION 05

# 프로젝트 진행

# Project Schedule

Week	6	7	8	9	10	11	12	13	14	15	16		
BE	제안서 작성	요구사항 스펙 작성		API 스펙 작성	단위테스트 작성	초기 단위 테스트	DB 연결	API 개발	종합테스트	배포	발표	시연 및 코드리뷰	
FE					UI/UX 템플릿 개발		UI/UX 구현			시각자료 병합			발표자료
Design				UI/UX 템플릿 디자인	UI/UX 고도화		시각자료 고도화						
Meeting	제안서 발표	요구사항 스펙		API 스펙	API 스펙 작성	inc1 코드리뷰 & 단위테스트		UI/UX 리뷰	종합테스트 및 최종 코드리뷰		최종발표 및 시연	시연 자료	

# Project Tool



## SE7 Project

**Deploy Update (0612)**

<http://54.180.145.172/>  
<http://54.180.145.172:8000/docs>

**Github Rule**

- main 브랜치는 배포용 브랜치입니다! 직접 push는 지양해주세요.
- fork 레포지토리나 각자의 브랜치에서 작업하고 main 브랜치에 pull request하는 방식으로 진행하겠습니다.
- 작업을 시작하기 전에 항상 main 브랜치에서 pull부터 받아주세요!!
- pull request하는 기준은 **정상 실행 여부**입니다. 모듈이 완성되지 않아도 괜찮습니다. 다만 빌드 과정에서 오류가 있으면 안됩니다!

- DB 구성
  - MySQL 서버 실행

```
sudo /etc/init.d/mysql restart
```
  - 서버 접속

```
sudo mysql
```
  - se라는 DB user 생성하고 비밀번호(310036)

```
show databases; // 현재 database 확인
use mysql; // mysql DB로 이동
show tables; // 현재 table 확인
select user, host from user; // 현재 생성된 user 확인

create user se@localhost identified by '310036';
// root라는 이름의 user는 localhost에 접근가능하게 되고 비밀번호는 310036
// DB_USERNAME = se, DB_PASSWORD = 310036
```

```
grant all privileges on *.* to se@localhost;|
// se의 localhost 접근에 대한 모든 권한 부여
```

- 새로운 DB인 green\_coders\_db 생성

```
show databases; //현재 생성된 database 확인
create database green_coders_db; // green_coders_db이라는 새로운 database 생성
use green_coders_db; // green_coders_db으로 database 변경
```

### 프로젝트 실행(로컬)

먼저 FastAPI 백엔드를 실행하고, 그 다음 React 프론트엔드를 실행.

localhost:3000 에서 페이지 확인가능.

- DB(3306)

```
sudo /etc/init.d/mysql restart
```
- 백엔드(8000)

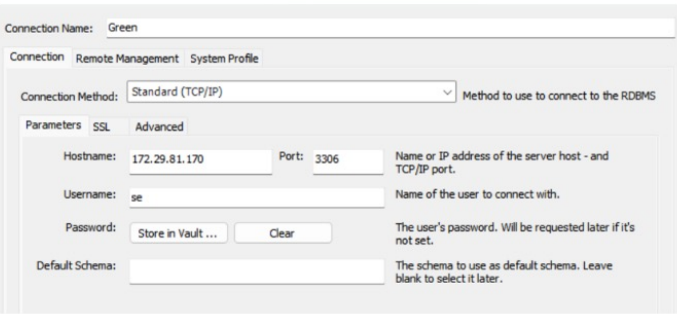
```
uvicorn app.main:app --host 0.0.0.0 --port 8000
```
- 리팩토링(8080)

```
chmod +x gradlew
./gradlew build
java -jar build/libs/refactor-engine-1.2.1.jar
```

- 프론트엔드(3000)

```
npm start
```

### Mysql Workbench 연결



- Hostname: 본인의 mysql이 실행되고 있는 ipv4
  - `ip addr | grep "inet"`

```
(base) root@Ubin108:~# ip addr | grep "inet"
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
inet 172.29.81.170/20 brd 172.29.95.255 scope global eth0
inet6 fe80::215:5dff:fe87:a967/64 scope link
```

- 여기서 아래 172.29.81.170을 입력해주었습니다.
- 각자 컴퓨터마다 해당 addr은 다르므로 직접 확인해주세요.



# Project Tool



🔗 0 Open   ✓ 35 Closed		Author ▼	Label ▼	Projects ▼	Milestones ▼	Reviews ▼	Assignee ▼	Sort ▼
🔗 Initial Service Version								💬 1
#36 by jaebin-code was merged 11 minutes ago								
🔗 remove line number in buggy/fixed part data								
#35 by ChoiJangSeop was merged 28 minutes ago								
🔗 complete Calculator page & Admin page								💬 1
#34 by saem020402 was merged 11 hours ago								
🔗 solve refactoring2 infinite-loop error								
#33 by ChoiJangSeop was merged yesterday								
🔗 add new column in fixedCode/buggyCode table & modify refactor-engine response & solve some error in refactoring 2								
#32 by ChoiJangSeop was merged yesterday								
🔗 resolve some bugs in service layer & add "nRefactoring" attribute in response								
#31 by ChoiJangSeop was merged 2 days ago								
🔗 Full system deploy (Backend, Frontend, Refactoring-engine, Database)								
#30 by Ubin108 was merged 2 days ago								
🔗 BE, FE, Refactor-engine connect complete								
#29 by Ubin108 was merged 2 days ago								
❗❗ connected with buggycode db								💬 1
#28 by saem020402 was closed 2 days ago								
🔗 [중요] 백엔드, 프론트엔드, 리팩토링 엔진 연결 완료								💬 1
#27 by jaebin-code was merged 2 days ago								

saem020402 commented 12 hours ago • edited ▼

Collaborator ⋮

calculator 페이지에서 리팩토링, 전략별 리팩토링, fixed\_code 에 저장까지 완료했습니다.  
admin 페이지에서 테이블 불러오기와 row 삭제도 정상동작합니다.

INSERT INTO fix\_strategy (fix\_strategy\_id, description) VALUES (1, 'value\_for\_other\_columns');  
INSERT INTO fix\_strategy (fix\_strategy\_id, description) VALUES (2, 'value\_for\_other\_columns');  
INSERT INTO fix\_strategy (fix\_strategy\_id, description) VALUES (3, 'value\_for\_other\_columns');

입력하시고 실행하시면 정상작동 확인할 수 있습니다.  
이제 best code, result 페이지만 재빈님이 구현하시면 개발 완료일 것 같습니다!

(저는 proxy 에러가 발생해서 "proxy": "<http://127.0.0.1:8000>" 로 했어요.)

1

complete Calculator page & Admin page

✓ 87717b2

jaebin-code commented 11 hours ago

Collaborator ⋮

고생 많으셨어요! 이제 이어서 마무리하겠습니다

saem020402 commented last week

Collaborator ⋮

admin page에서 report table 생성했습니다. 데이터 연동은 아직이고 임의로 넣어놓았습니다.  
about page는 디자인과 문구 조금 수정하였습니다.  
에러 예외처리 했습니다. 오류날시 모달 띄워 경고보여주도록 했습니다.

이제 백엔드 연동할게요!!

1

adit report page, about page & exception handling

✓ 1943466

Ubin108 commented last week

Collaborator ⋮

네 좋습니다!!

그리고 혹시 좌측 상단에 있는 조그만 글씨의 green coder는 없애고  
현재 헤더에 있는 큰 글씨의 Green Coders를 클릭할 수 있게 하는건 어떨까요?  
기존 화면 디자인이랑은 조금 다르지만 더 직관적인 디자인일 것 같습니다.

1

# Project Tool



## Github



ChoiJangSeop commented 2 days ago • edited ▾

Collaborator ...

**BuggyCode와 FixedCode에 세부 구현 로직을 추가했습니다.**

추가한 기능은 아래와 같습니다

### 1. 날짜 기반으로 BuggyCode와 FixedCode 조회

- Request URL:

```
GET /api/buggyCodes/date/{조회할 날짜}
GET /api/fixedCodes/date/{조회할 날짜}
```

- Response: 기존의 리스트 형식의 데이터 응답과 동일합니다.

### 2. FixedCode 감소율 기준 랭킹 조회

- Request URL :

```
GET /api/fixedCodes/ranking/{전략번호}/{랭킹수}
입력된 전략 번호에 해당하는 리팩터링 된 코드 중 상위 랭킹수 만큼의 코드를 리스트 형식으로 응답합니다.
```

- Response: 기존의 리스트 형식의 데이터 응답과 동일합니다.
- 존재하는 데이터보다, 더 많은 양의 랭킹 수를 입력하면, 전체 데이터가 정렬되어 리턴됩니다.

해당 기능을 프론트에서 다음과 같이 사용할 수 있습니다.

### case1. 5일간의 사용자 수 조회

```
GET /api/buggyCodes/date/{오늘날짜}
GET /api/buggyCodes/date/{오늘날짜-1}
GET /api/buggyCodes/date/{오늘날짜-2}
GET /api/buggyCodes/date/{오늘날짜-3}
GET /api/buggyCodes/date/{오늘날짜-4}
```

각 응답의 **nlitems** 필드를 모두 더하면 5일간의 사용자 수 입니다.



jaebin-code commented yesterday

Collaborator ...

백엔드와 프론트엔드, 리팩토링 엔진을 연결 완료했습니다.  
수정사항이 많으니 꼭 읽어주세요.

1.

우선 엔진 버전이 refactor-engine-1.0.0.jar 에서 refactor-engine-1.0.1.jar로 바뀌었으니 꼭 확인 부탁드립니다.  
장섭님과 엔진과 프론트엔드의 통신을 위해 cors 관련하여 수정사항이 들어있습니다.

2.

엔진에 코드를 넣을 때, 파일 형식은 json으로 하되, json.stringify 사용시에 엔진에서 처리하는 로직과 달라서 엔진 사용시에 사용 금지 부탁드립니다. 현재 Calculator 파일에서 해당 내용 확인 가능합니다. (장섭님과의 회의 내용입니다.)

한샘님께서 코드 제출 직후 페이지와 코드 제출에 각 데이터베이스에 정보를 보내주는 로직을 구현해주시면 될 것 같습니다.

감사합니다.



Ubin108 commented 2 days ago • edited ▾

Collaborator ...

전체 시스템에 대한 배포를 완료했습니다.  
배포는 AWS EC2 인스턴스에서 진행되었습니다.  
각각의 시스템은 별도의 컨테이너로 이루어져 있고 Port 번호를 통해 서로 통신합니다.

모든 컨테이너는 2024spring\_41class\_team7\_app-network 도커 네트워크에 속해있도록 하여 서로 유기적인 통신이 이루어질 수 있습니다. 이때 동일한 IP 주소 아래에 여러 컨테이너가 배포되기 때문에 nginx를 통해서 라우팅 설정을 해주었습니다.

```
/ : Frontend (3000)
/api/ : Backend (8000)
/refactor/ : refactor-engine (8080)
```

refactor-engine은 JAVA 코드의 refactoring을 다루기 때문에 자바스프링환경에서 개발되었고, 기존 Backend 모듈들은 Python 기반의 FastAPI로 개발되었기 때문에 두 모듈은 분리된 컨테이너 상에서 작동되도록 하였습니다.  
Frontend는 API 호출 시에 라우팅 path를 통해 필요한 컨테이너를 호출하게 됩니다.

DB의 경우에는 개발과정에서 삭제 후 재실행됨에 따라 정보가 손실되지 않도록 하기 위해 docker compose 에 포함시키지 않고 별도로 이미지를 만들고 컨테이너로 실행했습니다. DB는 3307 포트를 사용했으며 Backend와 연결되도록 했습니다.

deploy 버전의 경우에는 nginx 라우팅 설정을 위해 local 버전과는 호출 path가 다를 수 있음에 유의해야 합니다.  
main 브랜치에는 local 버전으로 올라가며 deploy 버전은 주석 처리하여 표시했습니다.



1



Google Sheet

소공개Team 7

파일수정보기삽입서식데이터도구확장프로그램도움말

메뉴

100%123기본값10BBI

J15

	A	B	C	D	
1	Refactor API	옆으로 스크롤 하면 해당 API에 관한 데이터베이스 정보가정보가 다이어그램으로 나타나있습니다		특이사항) 잘못된 요청으로 인한 4XX, 5XX 에러는 발생하지 않음. 단, "status=fail"로 출력되므로, 이	
2		url		body	
3			request	text/plain (user input code)	
4	특정 리팩터링 전략에 대한 리팩터링 코드 요청	POST /refactoring/{refactoringId}	response	{ "refactorId": 1, "buggyCodePart": "buggy part", "fixedCodePart": "fixed part" }	1 : for statemer 2. if statementr - refactoring 사 - refactorID가 4
5			request	text/plain (user input code)	
6	모든 리팩터링 전략을 통한 리팩터링	POST /refactoring/all	response	{ "fixedCodeText": "refactoring code text" }	
7				user input code 유의사항! : 반드시 Buggy 클래스 내부의 main 함수에 구현된 코드여야만 합니다.	
8	Main API				
9	reponse 형식 설명	{ "status": "success or fail", "onSuccess": {}, "onError": { "exception": "", "message": "" } }		모든 API에 대한 응답은 이와 같은 형식으로 구성됩니다. status는 요청에 대한 응답이 성공 했는지 실패 했는지, 표시합니다. 성공(success)했을 경우, "onSuccess" 바디에 응답의 결과가 표시됩니다. 실패(fail) 했을 경우, "onFail"에 실패와 관련된 정보가 표시됩니다. "exception" 필드에는 예외 타입을, "message" 필드에는 예외에 대한 정보가 표시됩니다.	
10					
11	FixedCodes				
12			request	{ "buggyPart": "buggy part", "fixedPart": "fixed part", "reducedAmount": 0.0, "reducedRate": 0.0, "buggyCodeId": 0, "fixStrategyId": 0 }	
13	FixedCode 생성	POST /fixedCodes	response	{ "status": "success", "onSuccess": { "fixedCodeId": 0 }, "onError": { "exception": "" } }	

+≡Team 7ProposalReq\_SpecDesign\_SpecAPI#2#3#4#5#6#7#8#9#10#13

**p.s. How about Open API?**



감사합니다