GREEN CODERS

JAVA 코드 그린화 플랫폼

Team 7

김유빈, 나건우, 성봉진, 이재빈, 정한샘, 최장섭

SECTION 01

시스템 구성

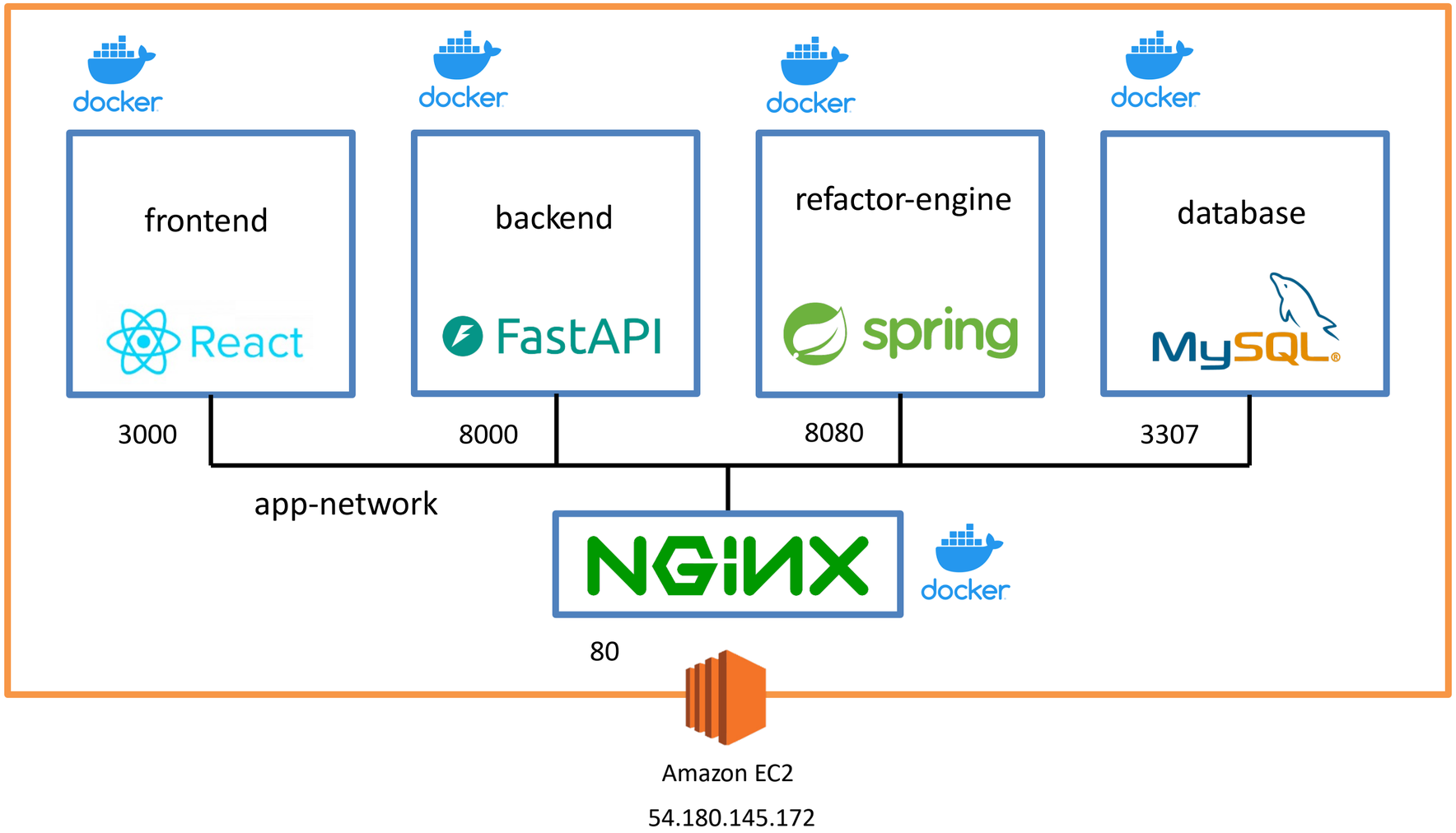
What is “Green Coders”

Best Refactoring Pattern

Dashboard

환경 기여 인지

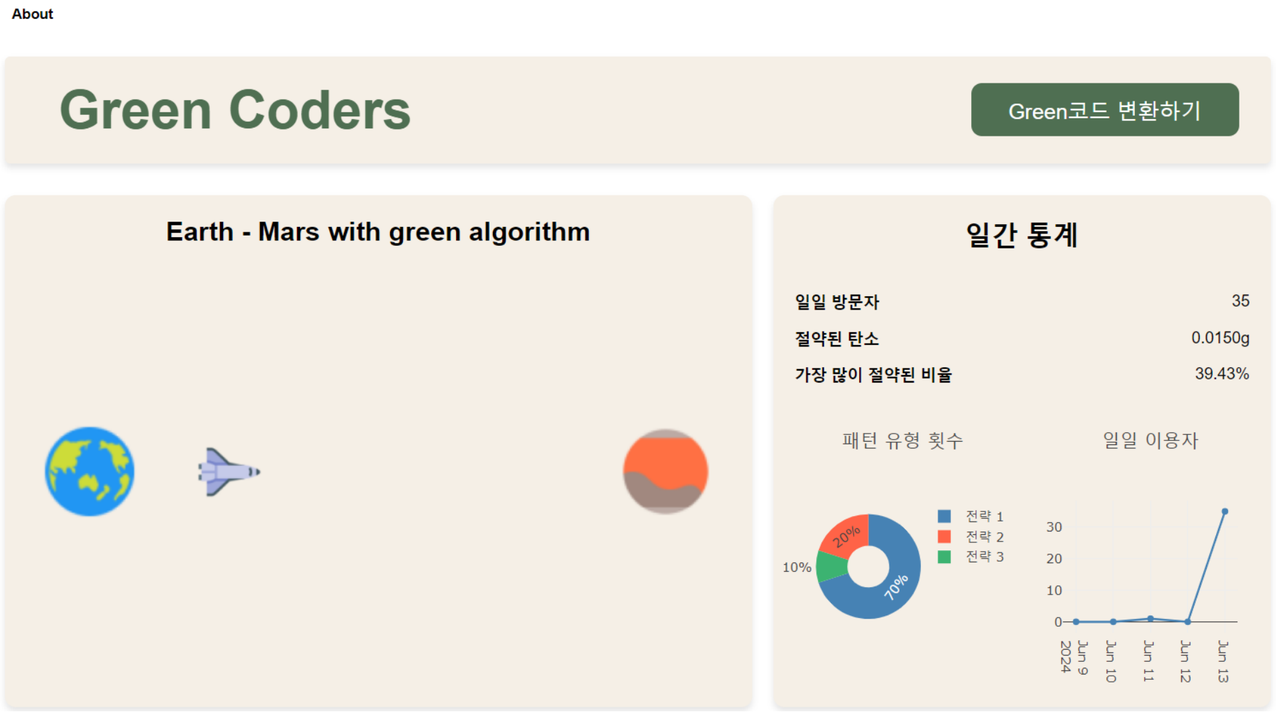
System Architecture



SECTION 02

인터페이스

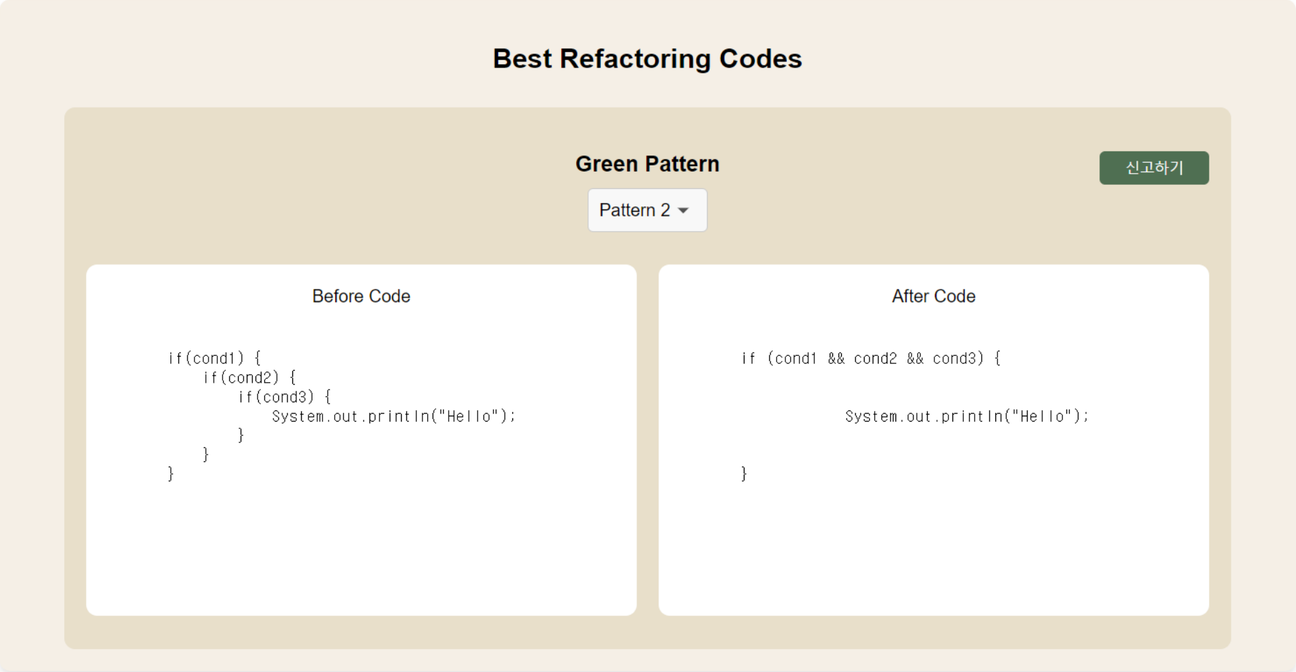
User Interface



시각화 전략

1. 접근성 향상
2. 관심 증대

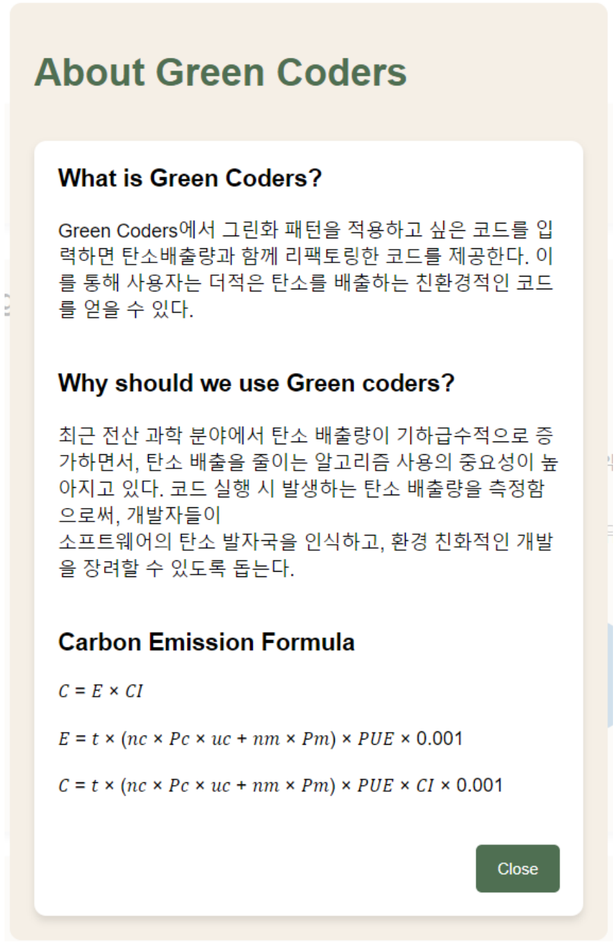
User Interface

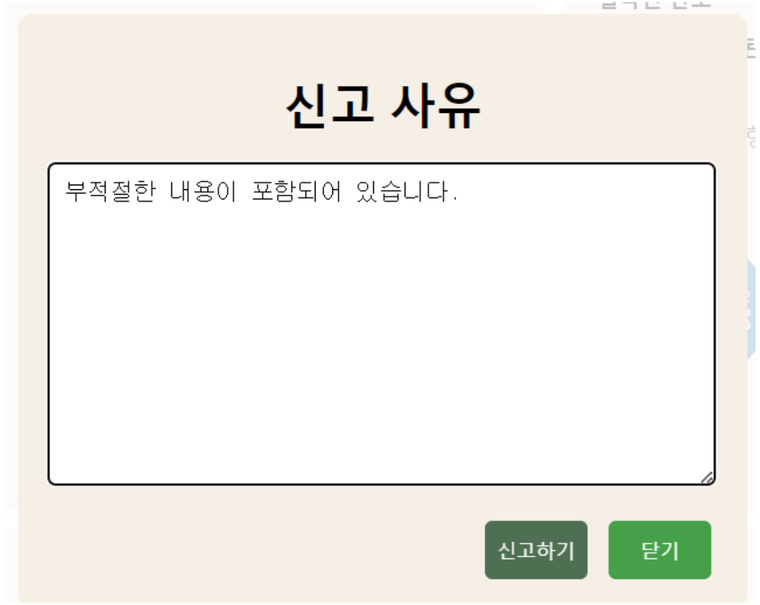


패턴 별 코드 개별 제공

1. 효과적인 코드 패턴 공유
2. 직관적인 이해

User Interface





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 /reports/fixedCode/{id}
		GET /fixedCodes/buggyCode/{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 /reports/fixedCode/{id}
		GET /fixedCodes/buggyCode/{id}		
삭제		DELETE /fixedCodes/{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 메세지를 저장합니다.

}

API Details

API Docs is release!

More details in...

Green-Coders Main API Docs

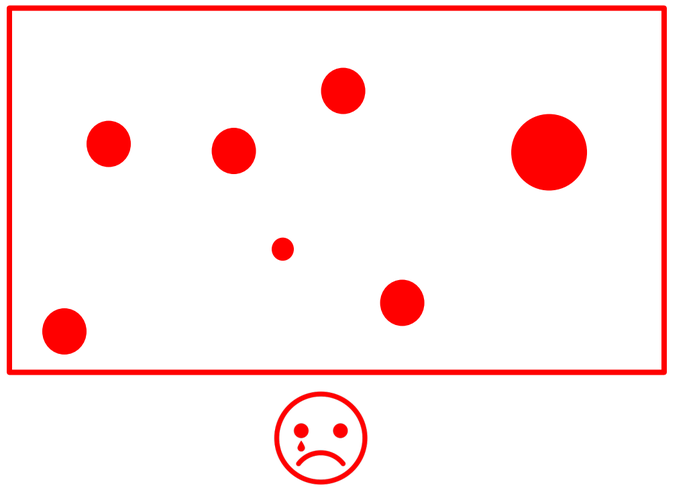
SECTION 04

그린화 리팩터링

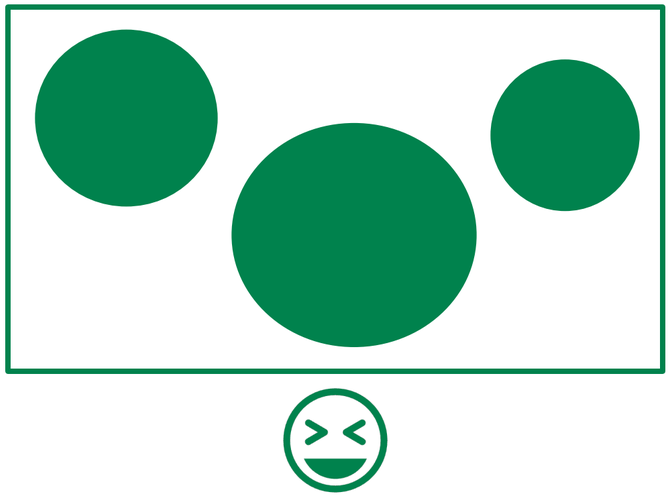
Refactoring Details

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

```java
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");
        }
    }
}
```

```java
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

```java
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");
        }
    }
}
```

```java
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");
        }
    }
}
```

```java
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");
        }
    }
}
```

Cannot refactor! 

go to infinite loop

Refactoring Details

Refactoring 1 (Green-Coders)

duplicated *.size()*

in iteration statement

Supports all cases of nested For-statement!

```java
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");
        }
    }
}
```

```java
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

Refactoring 1 (baseline)

duplicated *.size()*

in iteration statement

Supports all cases of nested For-statement!

```java
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");
        }
    }
}
```

```java
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");
        }
    }
}
```

New variable name is added **special signature**

It prevents duplicated variable name!

Refactoring Details

Refactoring 2 (baseline)

obsolete nested If statement

```java
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");
                }
            }
        }
    }
}
```

```java
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...

```java
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

```java
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);
        }
    }
}
```

```java
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

```java
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);
        }
    }
}
```

```java
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

```java
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

```java
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);
                }
            }
        }
    }
}
```

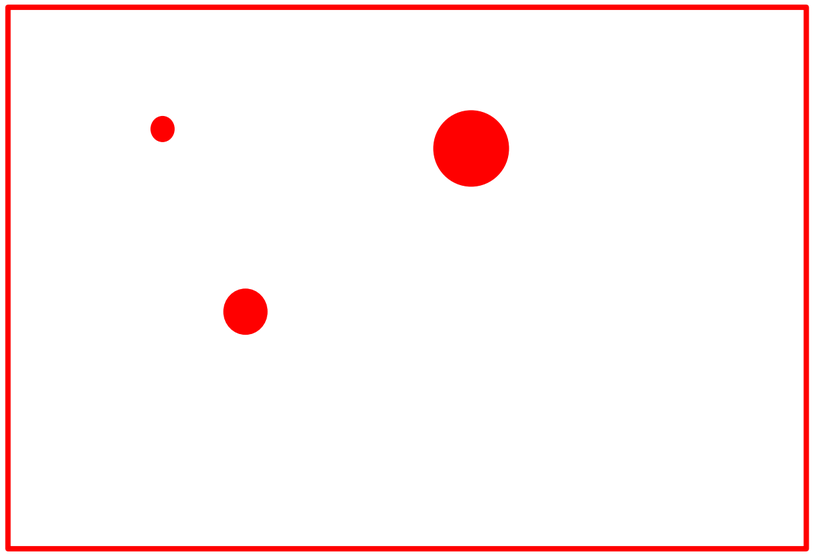
Supports all cases of nested For-statement!

```java
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);
                }
            }
        }
    }
}
```

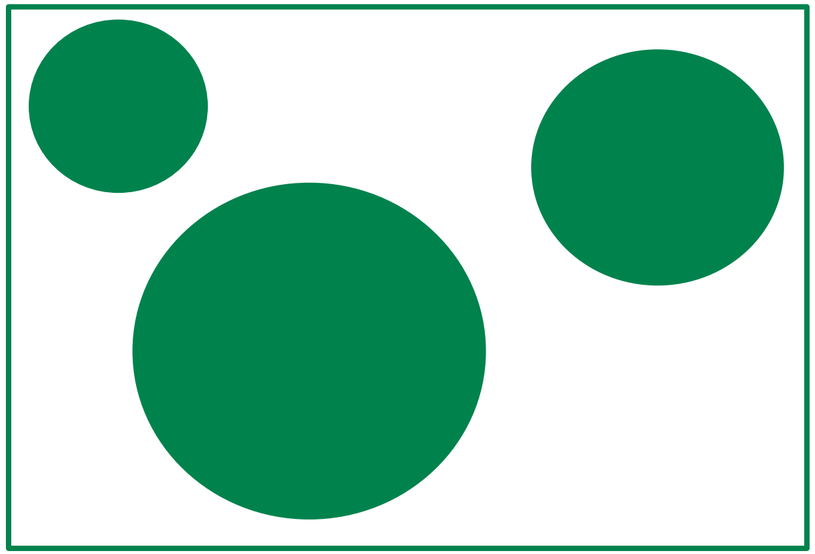
Refactoring Details

Green-Coders has Amazing Coverage!

baseline



Green-Coders



SECTION 05

프로젝트 진행

Project Schedule

<table>
<tr><td>Week</td><td>6</td><td colspan="2">7</td><td>9</td><td>10</td><td colspan="2">11</td><td>12</td><td colspan="2">13</td><td>14</td><td>15</td><td>16</td></tr>
<tr><td>BE</td><td rowspan="3">제안서 작성</td><td colspan="3" rowspan="3">요구사항 스펙 작성</td><td rowspan="2">API 스펙 작성</td><td>단위테스트 작성</td><td rowspan="2">초기 단위 테스트</td><td>DB 연결</td><td colspan="2">API 개발</td><td rowspan="2">종합테스트</td><td>배포</td><td rowspan="3">발표</td><td rowspan="3">시연 및 코드리뷰</td></tr>
<tr><td>FE</td><td>UI/UX 템플릿 개발</td><td colspan="2" rowspan="2">UI/UX 구현</td><td rowspan="2">시각자료 병합</td><td rowspan="2">발표자료</td></tr>
<tr><td>Design</td><td>UI/UX 템플릿 디자인</td><td colspan="2">UI/UX 고도화</td><td colspan="2">시각자료 고도화</td></tr>
<tr><td>Meeting</td><td>제안서 발표</td><td>요구사항 스펙</td><td></td><td>API 스펙</td><td>API 스펙 작성</td><td colspan="2">inc1 코드리뷰 & 단위테스트</td><td>UI/UX 리뷰</td><td colspan="2">종합테스트 및 최종 코드리뷰</td><td>최종발표 및 시연</td><td>시연 자료</td><td></td></tr>
</table>

Project Tool

Notion

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)

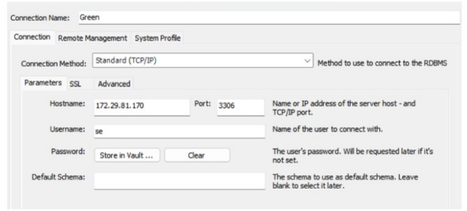
  `uvicorn app.main:app --host 0.0.0.0 --port 8000`
- 리팩토링(8080)

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

- 프론트엔드(3000)

  `npm start`

Mysql Workbench 연결



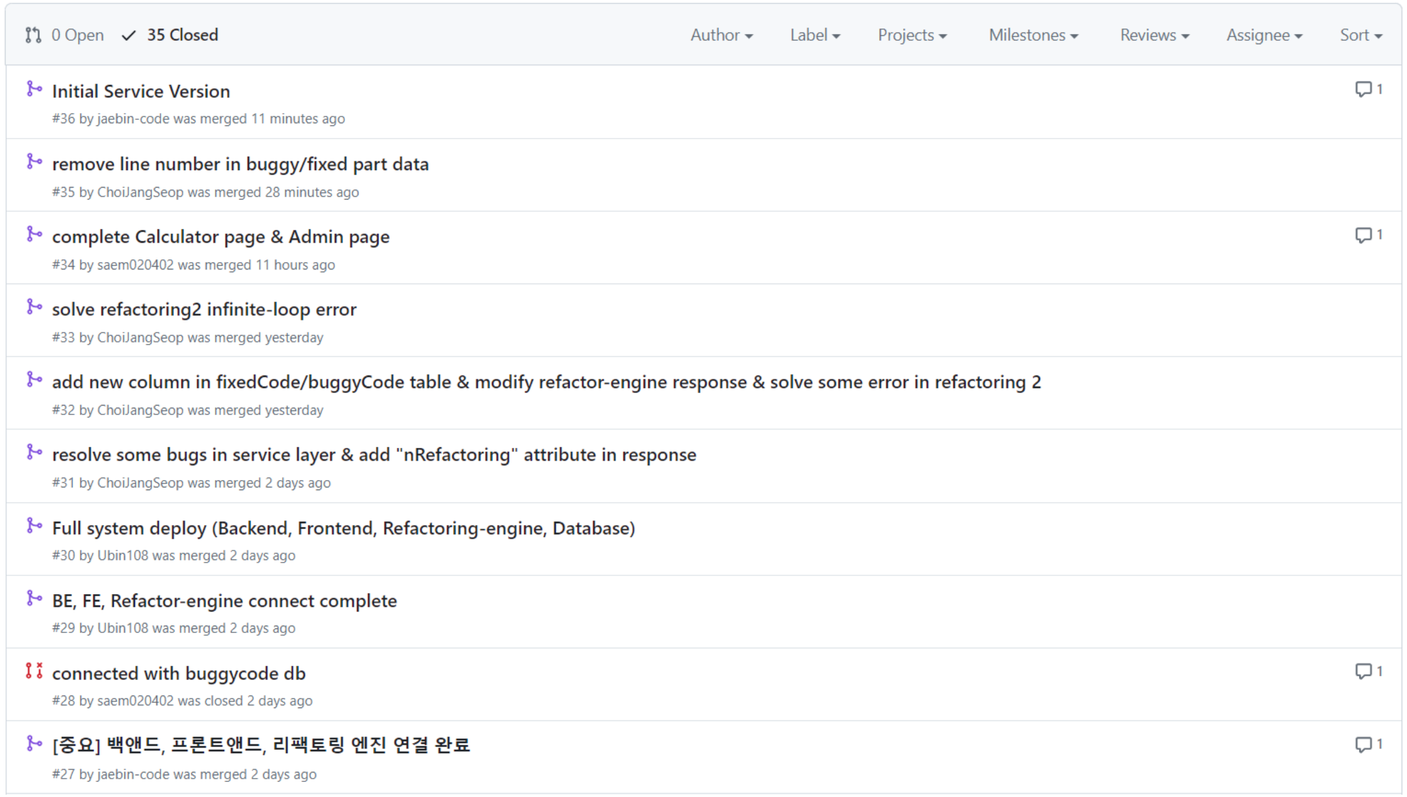
- Hostname: 본인의 mysql이 실행되고 있는 ipv4
  - `ip addr | grep "inet"`

```
(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

Github



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 프런트에서 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 **nItems** 필드를 모두 더하면 5일간의 사용자 수 입니다.

jaebin-code commented yesterday — Collaborator

백앤드와 프론트앤드, 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.stringfy 사용시에 엔진에서 처리하는 로직과 달라서 엔진 사용시에 사용 금지 부탁드립니다. 현재 Calculator 파일에서 해당 내용 확인 가능합니다. (장섭님과의 회의 내용입니다.)

한샘님께서는 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.

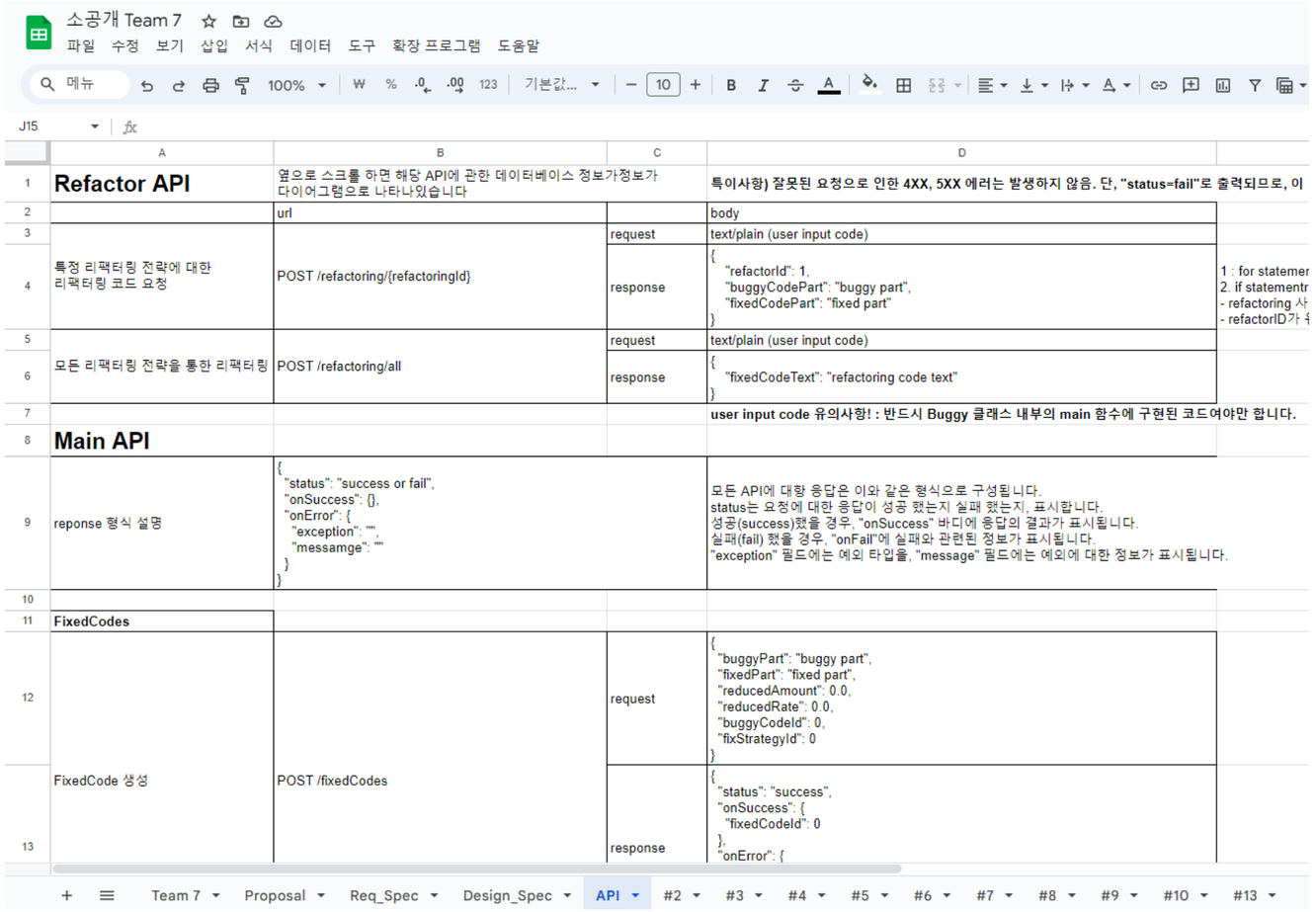
DB의 경우에는 개발과정에서 삭제 후 재실행됨에 따라 정보가 손실되지 않도록 하기 위해 docker compose 에 포함시키지 않고 별도로 이미지를 만들고 컨테이너로 실행했습니다. DB는 3307 포트를 사용했으며 Backend와 연결되도록 했습니다.

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

Project Tool

Google Sheet



소공개 Team 7

파일 수정 보기 삽입 서식 데이터 도구 확장 프로그램 도움말

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	특정 리팩터링 전략에 대한 리팩터링 코드 요청	POST /refactoring/{refactoringId}	request	text/plain (user input code)
4			response	{ "refactorId": 1, "buggyCodePart": "buggy part", "fixedCodePart": "fixed part" }
5	모든 리팩터링 전략을 통한 리팩터링	POST /refactoring/all	request	text/plain (user input code)
6			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": "", "messamge": "" } }		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	FixedCode 생성	POST /fixedCodes	request	{ "buggyPart": "buggy part", "fixedPart": "fixed part", "reducedAmount": 0.0, "reducedRate": 0.0, "buggyCodeId": 0, "fixStrategyId": 0 }
13			response	{ "status": "success", "onSuccess": { "fixedCodeId": 0 }, "onError": {

Team 7 | Proposal | Req_Spec | Design_Spec | API | #2 | #3 | #4 | #5 | #6 | #7 | #8 | #9 | #10 | #13

p.s. How about Open API?

감사합니다