최재호

포트폴리오(Sonar Service)

제목	Sonar service
작성일	2021-03-30
작성자	최재호

목차

CONTENTS

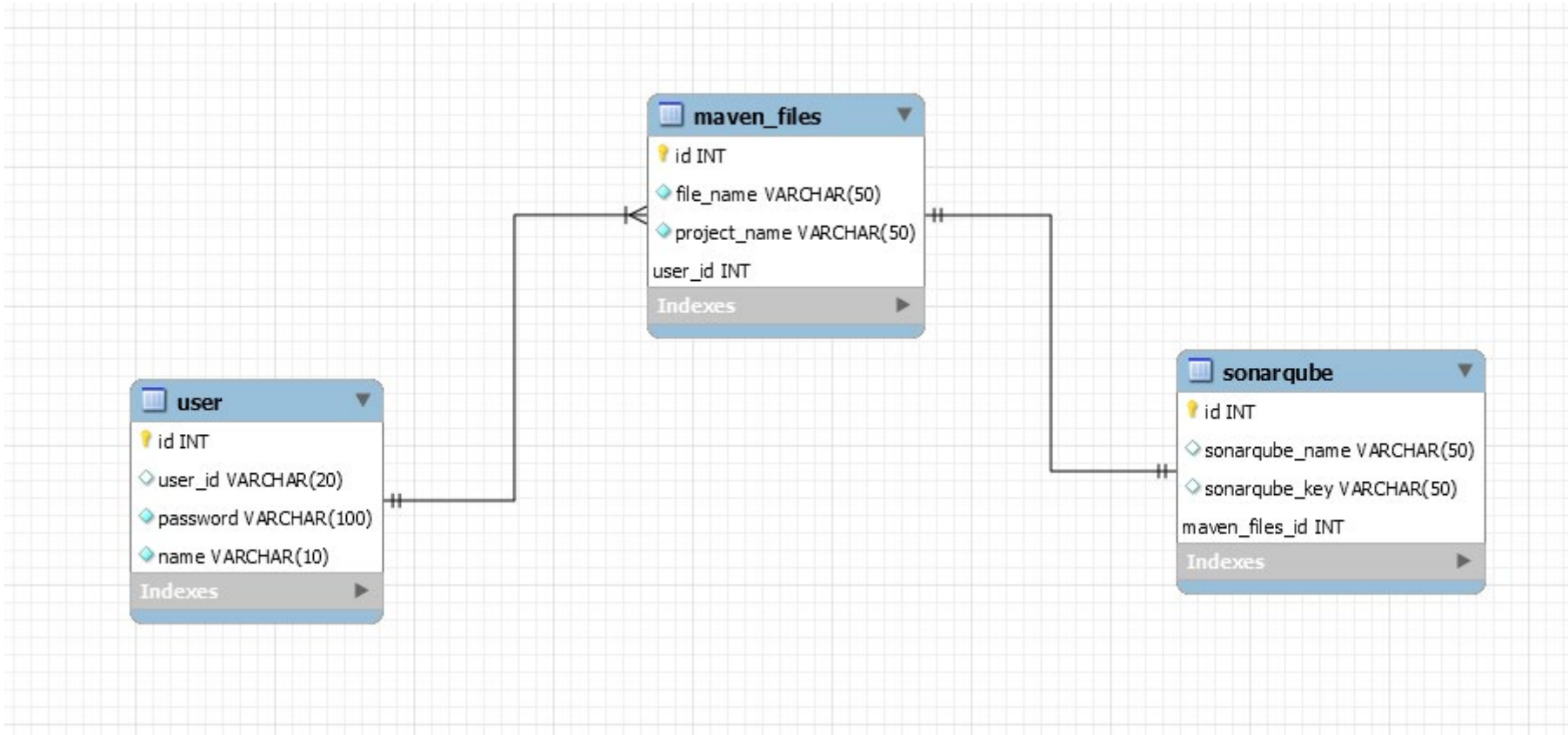
Sonar Service 개요

개요

구분	설명
개발 환경	Spring boot, logback, log4j2, maven, mybatis, mysql, thymeleaf, bootstrap, jquery
개발 인원	1명
요약	Sonar Service는 오픈소스인 Sonarqube를 연동하여 Maven프로젝트에 대한 품질을 표현주는 프로젝트 입니다. Dashboard에서 프로젝트에 대한 버그, 보안취약점, 코드악취, 커버리지, LOC 등 소나큐브 분석 내역을 확인할 수 있습니다.
기능	• 회원가입, 로그인 • Maven 파일 업로드 • 소나큐브 분석 • 소나큐브 대시보드 확인

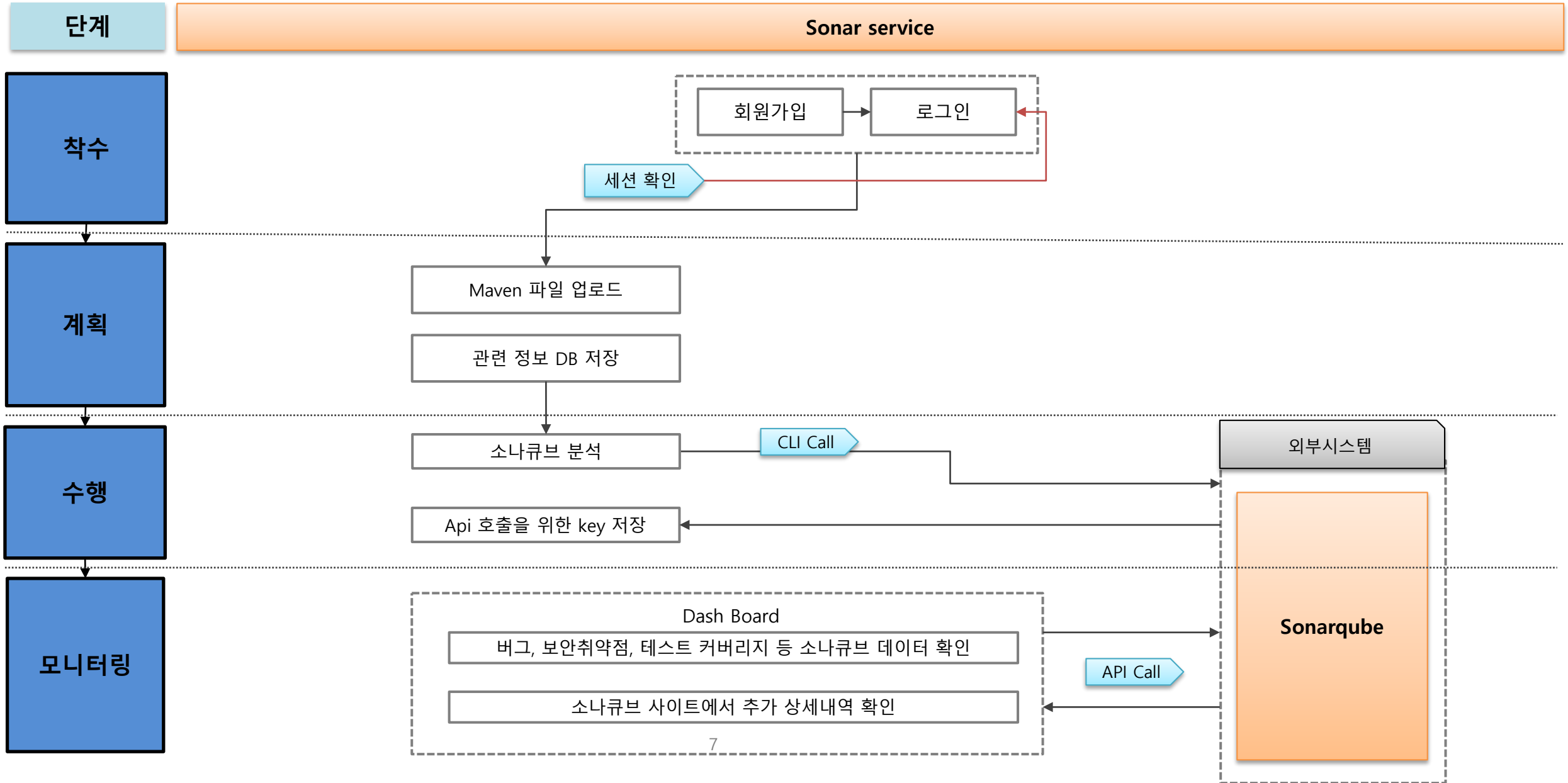
Sonar Service ERD

ERD


maven_files
id INT
file_name VARCHAR(50)
project_name VARCHAR(50)
user_id INT
Indexes
user
id INT
user_id VARCHAR(20)
password VARCHAR(100)
name VARCHAR(10)
Indexes
sonarqube
id INT
sonarqube_name VARCHAR(50)
sonarqube_key VARCHAR(50)
maven_files_id INT
Indexes

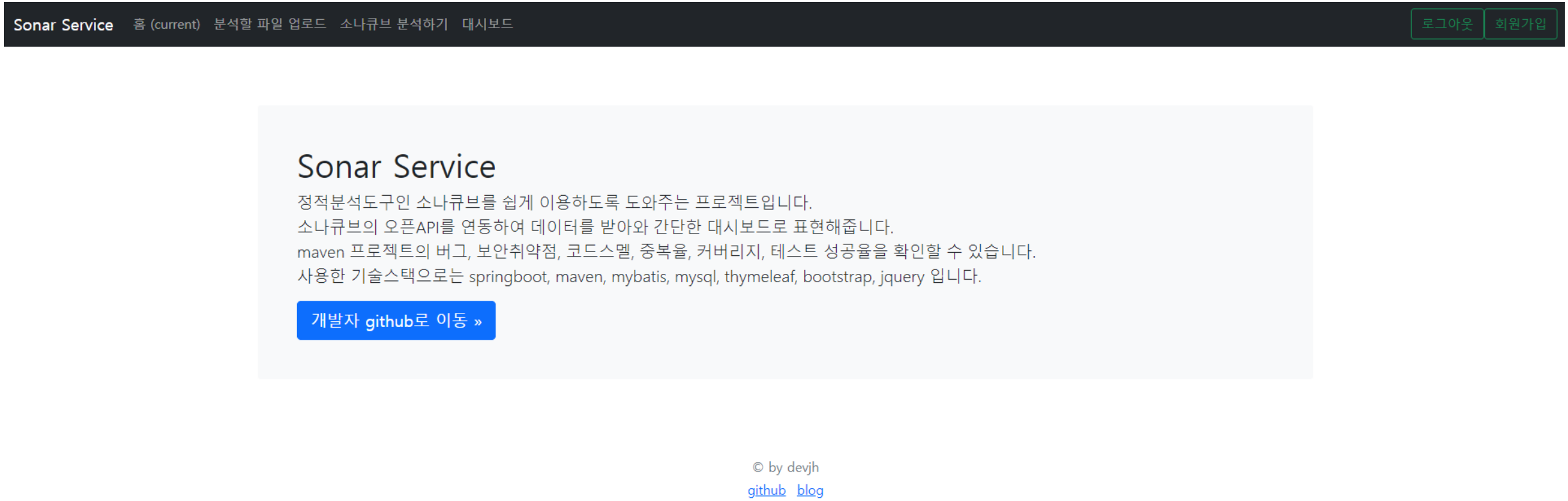
SonarService Process

프로세스

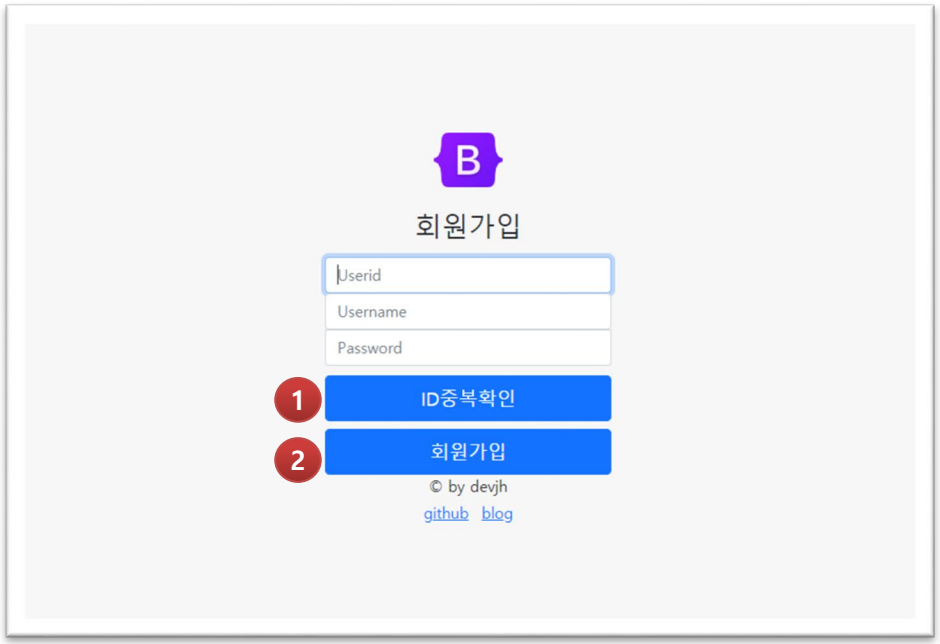

단계
Sonar service
착수
계획
수행
모니터링
회원가입
로그인
세션 확인
Maven 파일 업로드
관련 정보 DB 저장
소나큐브 분석
CLI Call
외부시스템
Api 호출을 위한 key 저장
Sonarqube
Dash Board
버그, 보안취약점, 테스트 커버리지 등 소나큐브 데이터 확인
API Call
소나큐브 사이트에서 추가 상세내역 확인

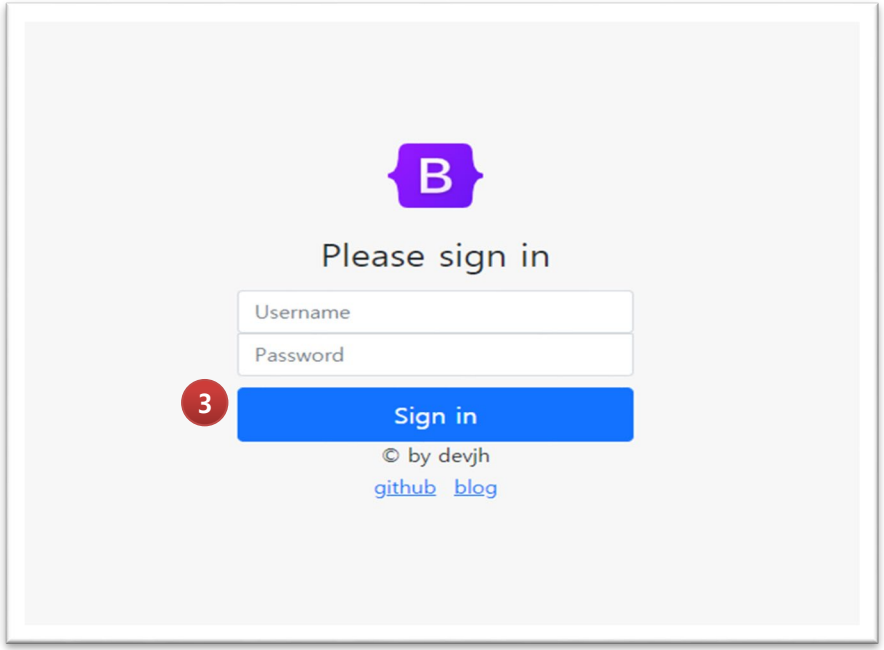
Sonar Service 화면 및 소스코드

화면1. 홈 화면


Sonar Service
홈 (current)
분석할 파일 업로드
소나큐브 분석하기
대시보드
로그아웃
회원가입
Sonar Service
정적분석도구인 소나큐브를 쉽게 이용하도록 도와주는 프로젝트입니다.
소나큐브의 오픈API를 연동하여 데이터를 받아와 간단한 대시보드로 표현해줍니다.
maven 프로젝트의 버그, 보안취약점, 코드스멜, 중복율, 커버리지, 테스트 성공율을 확인할 수 있습니다.
사용한 기술스택으로는 springboot, maven, mybatis, mysql, thymeleaf, bootstrap, jquery 입니다.
개발자 github로 이동 »
© by devjh
github
blog

화면1. 회원가입, 로그인






```xml
<select id="checkIdDuplicate" resultType="int">
    SELECT COUNT(*)
    FROM USER
    WHERE user_id = #{userId}
</select>
```

1

```
/**
 * 회원가입 메서드
 * @param userDto
 */
public void insertMember(UserDto userDto) {
    userDto.setPassword(passwordEncoder.encode(userDto.getPassword()));
    int insertCnt = userDao.insertMember(userDto);
    if (insertCnt != 1) {
        throw new RuntimeException("필수정보를 모두 입력해주세요.");
    }
}
```

2

```
/**
 * db에 복호화된 password와 유저가 입력한 password를 복호화한 값이 같은지 비교
 * 로그인 성공시 20분의 유지시간을 가진 세션 부여
 * @param userId 사용자가 입력한 아이디
 * @param password 사용자가 입력한 비밀번호
 */
public UserDto userLogin(String userId, String password, HttpSession httpSession) {
    String encodedPassword = userDao.getEncodedPassword(userId);
    boolean matches = passwordEncoder.matches(password, encodedPassword);
    if (!matches) {
        throw new RuntimeException("등록되지 않은 유저입니다.");
    }
    UserDto userDto = userDao.userLogin(userId);
    httpSession.setAttribute( s: "UserInfo", userDto);
    httpSession.setMaxInactiveInterval(20*60);
    return userDto;
}
```

2

구분	상세설명
1	DB에 중복된 아이디가 있는지 확인 합니다.
2	비밀번호를 암호화하여 유저정보를 user table에 저장합니다.
3	사용자가 입력한 아이디를 통해 암호화된 비밀번호를 가져오고, 입력한 비밀번호를 암호화 했을때 일치하는지 확인, 일치한다면 세션에 추가하였습니다. (로그아웃시 invalidate로 세션 헤제)

화면1. 회원가입, 로그인

```java
@Configuration
@AllArgsConstructor
public class WebMvcConfiguration implements WebMvcConfigurer {

    final InterceptorHandler interceptorHandler;

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        List<String> excludeList = new ArrayList<>();
        excludeList.add("/");
        excludeList.add("/users/**");
        excludeList.add("/api/users/**");
        excludeList.add("/css/**");
        excludeList.add("/js/**");

        registry.addInterceptor(interceptorHandler)
                .addPathPatterns("/**")
                .excludePathPatterns(excludeList);
    }
```

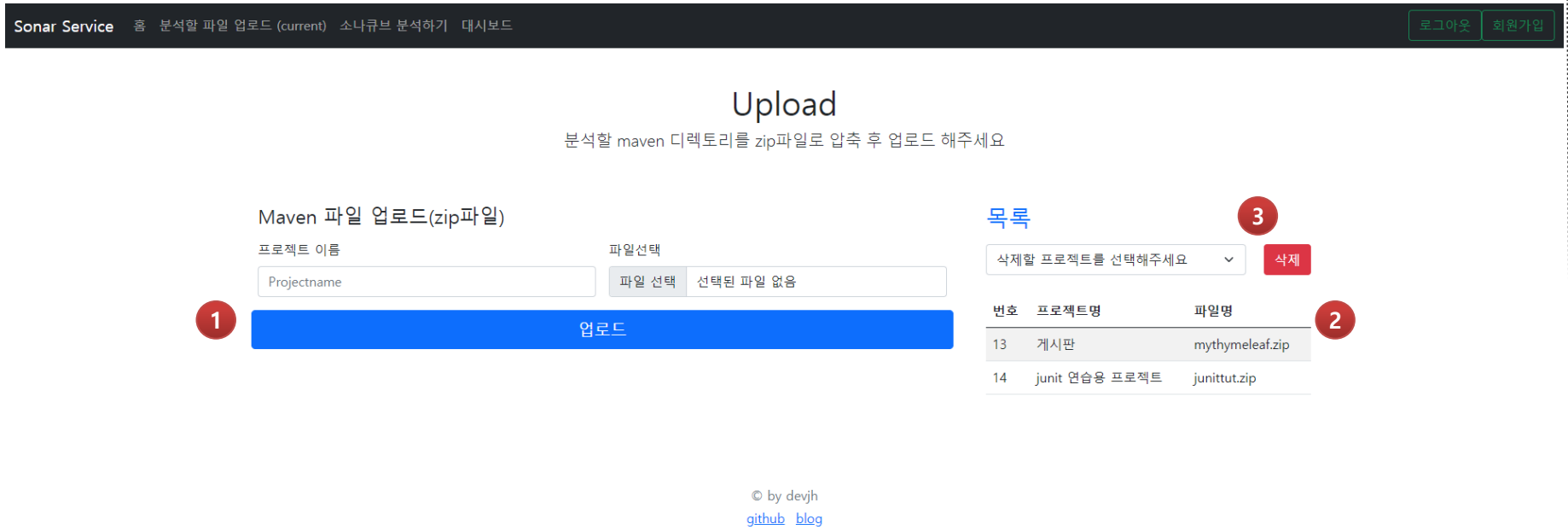
1

```java
@Component
public class InterceptorHandler implements HandlerInterceptor {

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        HttpSession session = request.getSession();
        Object data = session.getAttribute( s: "UserInfo");
        if (data == null) {
            response.sendRedirect( s: "/users/login");
            return false;
        }
        return true;
    }
}
```

2

구분	상세설명
1	인터셉터에서 설정입니다. Login logout, ID중복체크 등의 api와 정적 리소스들을 제외한 나머지 요청은 컨트롤러에 접근하기 전에 인터셉터를 거치도록 설정하였습니다.
2	세션에서 유저정보를 가져와 유저정보가 있다면 컨트롤러로, 없다면 로그인 페이지로 리다이렉트 시켰습니다.

화면2. Maven 파일 업로드



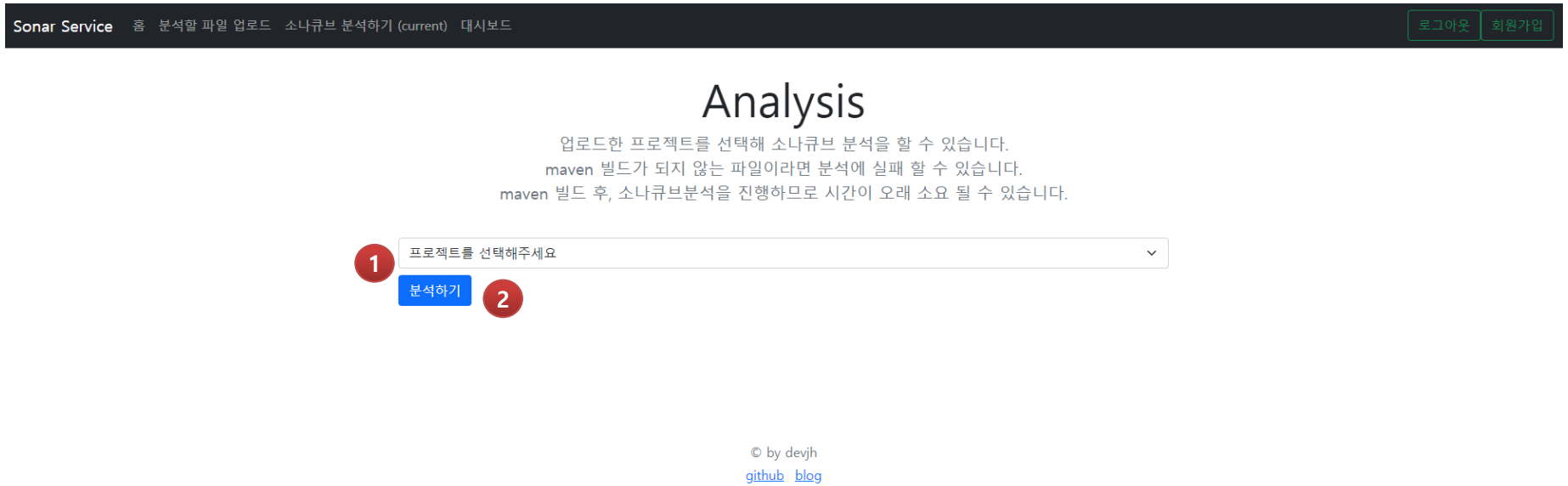
구분	상세설명
1	프로젝트명을 입력하고 zip파일을 업로드하면 유저ID이름의 폴더를 생성, 해당 zip파일 저장, DB에 file정보 저장합니다.
2	업로드버튼을 누른 누르면 화면은 redirect(새로고침시 이전요청이 다시 서버에 가는것을 방지하기 위한 POST REDIRECT GET)되며 db에 추가된 목록이 table에 표시됩니다.
3	삭제할 프로젝트를 선택, 삭제할 수 있습니다.

화면2. Maven 파일 업로드

```java
public void fileUpload(MultipartFile multipartFile, String projectName, HttpSession httpSession) {
    UserDto userInfo = (UserDto) httpSession.getAttribute( s: "UserInfo");
    String fileName = multipartFile.getOriginalFilename();
    String fileExtension = this.getFileExtension(fileName);

    if(!fileExtension.equals(".zip")) {
        throw new FileException("zip파일을 업로드 해주세요");
    }

    boolean isDuplicatedProjectName = this.checkProjectNameDuplicated(projectName, userInfo.getId());
    if (isDuplicatedProjectName) {
        throw new FileException("중복된 프로젝트명입니다. 프로젝트명을 변경해주세요");
    }

    boolean isDuplicatedFileName = this.checkFileNameDuplicated(fileName,userInfo.getId());
    if (isDuplicatedFileName) {
        throw new FileException("중복된 파일입니다. 목록을 확인해주세요");
    }
    String downloadDir = rootLocation;
    downloadDir = downloadDir + "/" + userInfo.getUserId();
    File makeFolder = new File(downloadDir);
    if(!makeFolder.exists()) {
        makeFolder.mkdir();
        log.info("{} 폴더생성", downloadDir);
    }
    log.info(downloadDir);
    Path copyOfLocation = Paths.get( first: downloadDir + File.separator +StringUtils.cleanPath(multipartFile.getOriginalFilename()));
    try {
        Files.copy(multipartFile.getInputStream(), copyOfLocation, StandardCopyOption.REPLACE_EXISTING);
    } catch (IOException e) {
        log.error(e.getMessage());
        throw new FileException("파일업로드 실패");
    }
    FileDto fileDto = FileDto.builder().userId(userInfo.getId()).fileName(multipartFile.getOriginalFilename()).projectName(projectName).build();
    this.insertFile(fileDto);
}
```

구분	상세설명
1	MultipartFile 라이브러리를 사용하여 파일업로드를 하였습니다.
2	예외는 RuntimeException을 상속받은 FileException 클래스를 만들어 ControllerAdvice에서 redirect하도록 처리하였습니다.
3	FileUpload 이후에는 사용자정보와 파일정보를 DB에 저장하였습니다.

화면3. 소나큐브 분석



구분	상세설명
1	Session을 통해 유저정보를 확인, 유저가 업로드 한 프로젝트명을 가져와 콤보박스로 표시합니다.
2	분석을 진행할 시 정적 분석을 하고, 소나큐브 Table에 api 연동을 위한 key를 저장합니다.

화면3. 소나큐브 분석

```java
/**
 * 유저정보와 fileId를 통해 DB에서 파일정보를 select, 해당 zip파일의 압축을풀고 소나큐브분석 메서드
 * @param fileId fileUid
 * @return
 * @throws Throwable
 */
@Transactional
public boolean analysis(int fileId, HttpSession httpSession) throws Throwable {
    FileDto fileDto = fileDao.getByFileId(fileId);
    String pureFileName = fileDto.getFileName().replace( target: ".zip", replacement: "");

    UserDto userInfo = (UserDto) httpSession.getAttribute( s: "UserInfo");
    unzipService.decompress(fileDto.getFileName(), userInfo.getUserId());
    mavenInstall(pureFileName,userInfo.getUserId());

    // 분석후, 소나큐브에 테이블에 insert
    // 이전에 insert 한경우 skip
    if (sonarqubeDao.getCnt(fileDto.getId()) == 0) {
        SonarqubeDto sonarqubeDto = SonarqubeDto
                .builder()
                .sonarqubeName(pureFileName)
                .sonarqubeKey(pureFileName + ".key")
                .mavenFileId(fileDto.getId())
                .build();
        sonarqubeDao.insertSonarqube(sonarqubeDto);
    }

    return true;
}
```

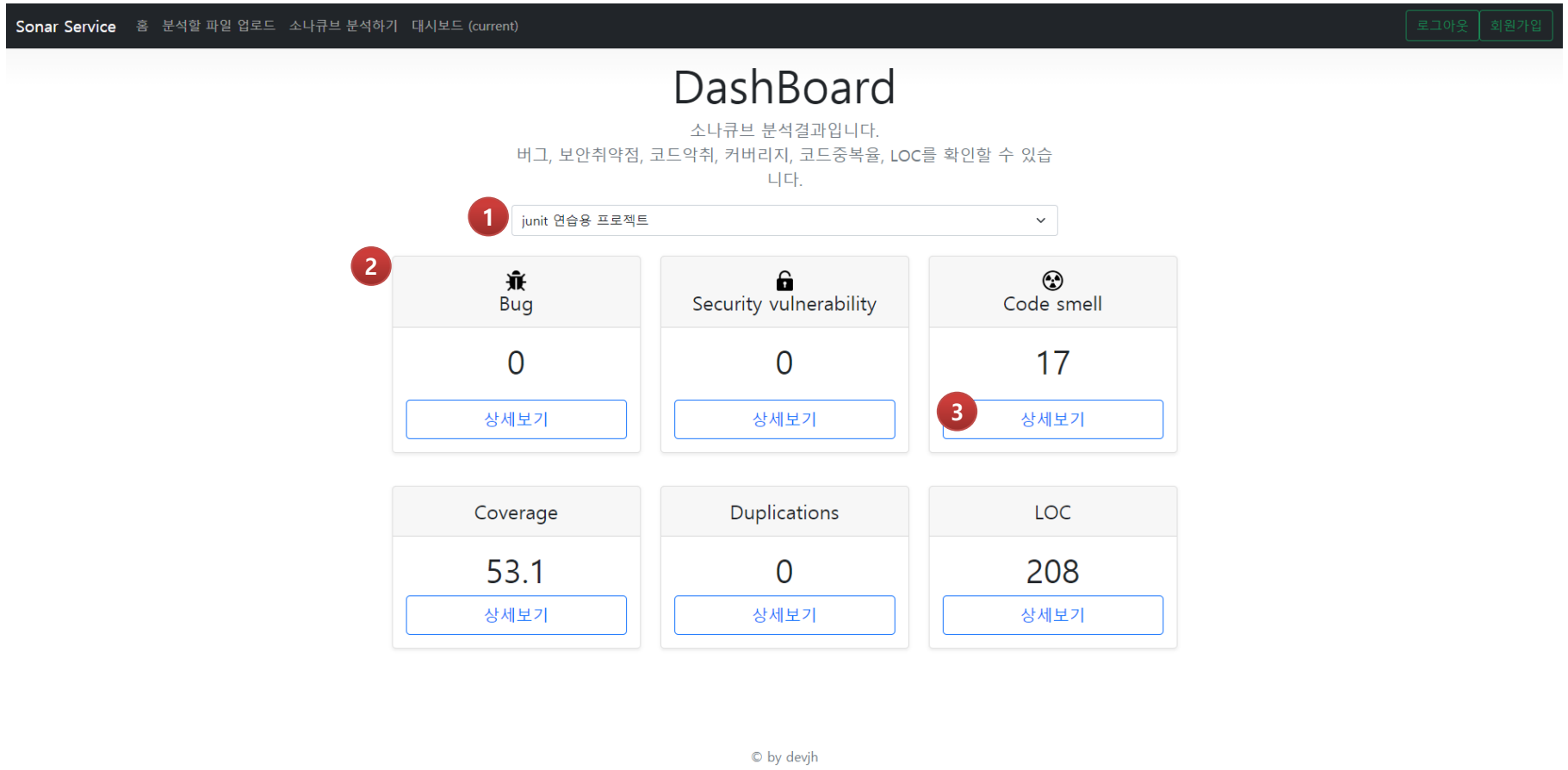
구분	상세설명
1	Session을 통해 유저정보를 확인, 유저가 업로드한 파일을 이용해 압축을 풀고 분석합니다.
2	분석이 끝나면 소나큐브 key와 소나큐브 프로젝트이름을 DB에 저장합니다.(소나큐브 API 호출시 필요)

화면3. 소나큐브 분석

```java
private void mavenInstall(String pureFileName, String userId) throws IOException, InterruptedException {
    List<String> command = new ArrayList<>();
1   command.add("cmd");
    command.add("/c");
    command.add("mvn clean install -f " + rootDir + "/" + userId + "/" + pureFileName +
            " sonar:sonar -Dsonar.login=" + sonarToken +
            " -Dsonar.projectKey=" + pureFileName + ".key"+
            " -Dsonar.host.url=" + sonarUrl + " -Dsonar.projectName=" + pureFileName);
    for (String comm : command) {
        log.info("{}",comm);
    }
    ProcessBuilder processBuilder = new ProcessBuilder(command);
    processBuilder.redirectErrorStream(true);
    Process process = processBuilder.start();
    StringBuilder stringBuilder = new StringBuilder();
2   if (process != null) {

        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(process.getInputStream()));
        String line = null;
        while ((line = bufferedReader.readLine()) != null) {
            stringBuilder.append(line + "\n");
            log.info("line {}",line);
        }
    }
    process.waitFor();

3   if (stringBuilder.toString().contains("BUILD FAILURE")) {
        throw new RuntimeException("분석 실패! 소나큐브서버오류 또는 메이븐빌드 오류입니다.");
    }
}
```

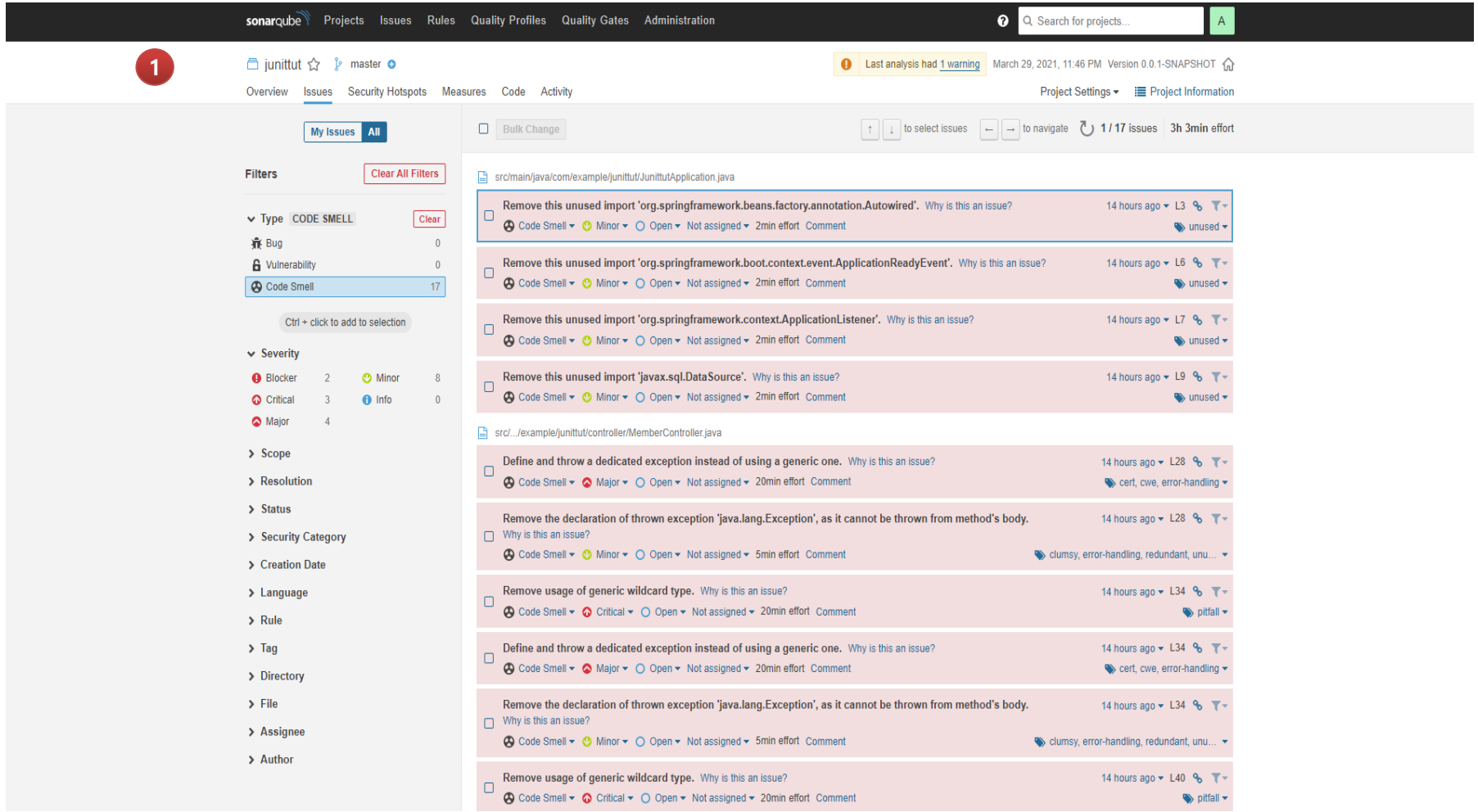
구분	상세설명
1	프로세스를 만들어 cmd환경에서 cli명령어로 소나큐브 분석을 시작합니다.
2	프로세스가 존재한다면 프로세스의 스트림을 읽어와 로그로 남기며 StringBuilder에 문자열을 저장합니다.(힙 메모리 낭비 방지)
3	StringBuilder를 String으로 변환하고 진행한 프로세스의 실패여부를 파악하여 사용자에게 전달합니다.

화면4. 대시보드



구분	상세설명
1	Session을 통해 유저정보를 확인, 유저가 분석했던 프로젝트명을 콤보박스로 보여줍니다.
2	소나큐브 api를 이용하여 데이터를 받아옵니다.(RestTemplate과 JsonParser를 이용하여 데이터 교환, 파싱)
3	상세보기를 클릭시 소나큐브 서버의 상세페이지로 이동합니다.

화면4. 대시보드



구분	상세설명
1	소나큐브 상세보기 버튼을 클릭시 이동하는 페이지로 어떤 종류의 버그가 어디에서 발생했는지 상세하게 확인 가능합니다.

화면4. 대시보드

```java
public SonarqubeMeasure measure(int sonarqubeId) {
    SonarqubeDto sonarqubeDto = sonarqubeDao.getById(sonarqubeId);
    String sonarqubeKey = sonarqubeDto.getSonarqubeKey();
    String hostUrl = sonarUrl;
    String apiUrl = "/api/measures/component";
    String metricKeys = "bugs,vulnerabilities,code_smells,coverage,ncloc,duplicated_lines_density";

    UriComponents builder = UriComponentsBuilder.fromHttpUrl(hostUrl+apiUrl)
            .queryParam( name: "component",sonarqubeKey)
            .queryParam( name: "metricKeys",metricKeys)
            .build();

    RestTemplate restTemplate = new RestTemplate();
    HttpHeaders headers = new HttpHeaders();
    headers.setContentType(new MediaType( type: "application", subtype: "json", Charset.forName("UTF-8")));
    restTemplate.getInterceptors().add(new BasicAuthenticationInterceptor(sonarUsername,sonarPassword));
    ResponseEntity<String> exchange = null;
    try {
        exchange = restTemplate.exchange(builder.toUriString(), HttpMethod.GET, new HttpEntity<String>(headers), String.class);
    } catch (RestClientException e){
        log.error(e.getMessage());
        throw new RuntimeException("소나큐브 서버 에러입니다.");
    }

    String body = exchange.getBody();
    SonarqubeMeasure sonarqubeMeasure = this.getMetric(body);
    return sonarqubeMeasure;

}
```

구분	상세설명
1	Application.properties파일에 정의된 소나큐브 정보를 가져온 후 RestTemplate을 이용하여 소나큐브 api 연계, 버그, 보안취약점, 코드스멜, loc, 중복률, 커버리지를 받아옵니다.

화면4. Maven 대시보드

```
public SonarqubeMeasure getMetric(String jsonData) {
    JsonParser jsonParser = new JsonParser();
    JsonArray JsonArray = jsonParser.parse(jsonData)
            .getAsJsonObject().get("component")
            .getAsJsonObject().get("measures")
            .getAsJsonArray();

    SonarqubeMeasure sonarqubeMeasure = new SonarqubeMeasure();
    for (JsonElement jsonElement : JsonArray) {
        String metric = jsonElement.getAsJsonObject().get("metric").getAsString();
        String value = jsonElement.getAsJsonObject().get("value").getAsString();
        switch (metric) {
            case "bugs":
                sonarqubeMeasure.setBugs(Integer.parseInt(value));
                break;
            case "duplicated_lines_density":
                sonarqubeMeasure.setDuplicated(Double.parseDouble(value));
                break;
            case "code_smells":
                sonarqubeMeasure.setCodeSmell(Integer.parseInt(value));
                break;
            case "ncloc":
                sonarqubeMeasure.setLoc(Integer.parseInt(value));
                break;
            case "coverage":
                sonarqubeMeasure.setCoverage(Double.parseDouble(value));
                break;
            case "vulnerabilities":
                sonarqubeMeasure.setVulnerability(Integer.parseInt(value));
                break;
            default:
                break;
        }
    }
    log.info("{}",sonarqubeMeasure);
    return sonarqubeMeasure;
}
```

구분	상세설명
1	소나큐브 api 연계 후 받은 Json문자열을 JsonParser를 이용하여 파싱, 객체로 변환하여 리턴 합니다.

Sonar Service 소스코드

소스코드

소스코드는 github에서 관련된 내용은 blog에서 확인 가능합니다.

구분	URL
github	http://github.com/jaeho310/sonarservice
blog	https://frozenpond.tistory.com