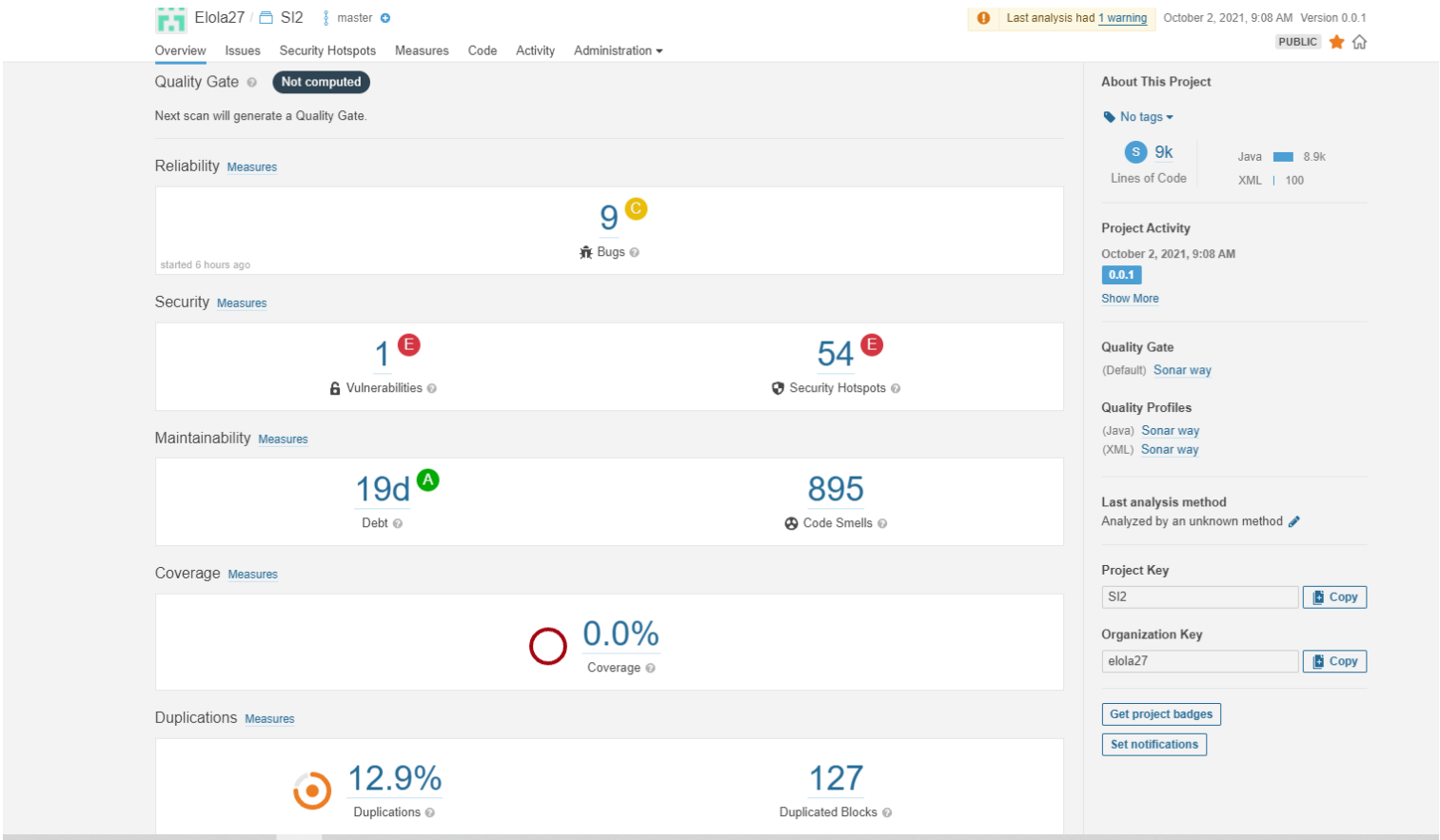
SONAR LANAREN DOKUMENTAZIOA

PDF honetan aurki daiteke SonarLint tresna erabiliz Bets21 proiektuan eginiko lana dokumentatuko da. Horretarako, hainbat atal ikusiko dira: lehen analisia egin osteko datuak, egindako aldaketak, eta azkenik bukaerako emaitza.

LEHEN ANALISIA

Lehendabiziko analisia egin ostean eta sonarcloud.io webgunera jaurtitzean emaitzak, honako datuak lortu ditugula ikusi dezakegu (beheko argazkia ikusi):



- ➢ Reliability atalean, kodean aurkitzen diren bug motako arazoak daude.
- ➢ Security atalean, bi motatako arazoak aurki daitezke: programaren segurtasuna hausteko modukoa eta zulotxoak.
- ➢ Maintainability atalean, kode basura kopurua eta mantenu guztia egiteko denbora zehazten da.

Datu horiek ikusita, ikusi daiteke hobekuntza handia dagoela egitea oraindik, bereziki kode basura asko dagoela ikusten da eta hori gutxitzen saiatuko gara.

EGINDAKO ALDAKETAK

Atal honetan saiatuko gara mota bakoitzeko arazoak bilatzen eta horiei soluzioak ematen. Sailkapena honela egin daiteke arazoena:

- ➢ **Ohar mota:** hainbat mota daude honen barnean:
  - Code smell: Kode basura deritzona da, funtzionamenduan arazorik ez du sortzen baina mantenuan.
  - Vulnerability: Erasoetarako irekitako puntua, larria izan daiteke funtzionamendurako.
  - Bug: Kode akatsa, funtzionamendua gera dezake.
- ➢ **Ohar zorroztasuna:** Beste hainbeste eta gehiago mota daude:
  - Blocker: Berehala zuzentzea komeni, aplikazioaren portaera alda dezake.
  - Critical: Ez-usteko jarrera aplikazioarena edo segurtasun akatsa. Errebisatzea komeni.
  - Major: Komenigarria zuzentzea, kalitate defektua normalean.
  - Minor: Kalitate defekturen bat, baina xehetasun txikikoa.
  - Info: Proposatzen diren hobekuntzak.

Beraz, goazen mota bakoitzetik bat/batzuk hartzea eta horiek zuzentzea:

- ➢ Major + Smell Code:
  - Add the @Override annotation above this method signature
    Arazo hau bi tokitan agertzen zen: ErrepikatzaileakGUI.java fitxategian 88 eta 133 lerroetan eta MugimenduakIkusiGUI.java fitxategian 154 lerroan. Azkenean, defektuzko bi metodo gainidaztean ahaztuta zegoen @Override etiketa ezartzea eta hori jartzea zen kontua.
  - This block of commented-out lines of code should be removed
    Komentaturik aurkitzen ziren kode zatiak borratzeko eskaera zen, azkenean mantenua zailagoa egitea bihurtzea baitzeukaten. Fitxategi konkretu bat esatea zaila da; izan ere, toki askotan aurkitu baitugu hau.

- ➢ Major + Bug:
  - Remove or correct this “removeAll” call
    ApustuaEginGUI.java interfazean 208 lerroan removeAll() funtzioa erabiltzen genuen zerrendako elementu guztiak ezabatzeko, baina clear() funtzioa erabiltzea gomendatzen du SonarLint-ek.
  - Identical expressions should not be used on both sides of a binary operator
    DataAccess.java fitxategian 613 lerroan objektu berdinaren elementuen artean == egiten zen eta SonarLint-ek dio beti false izango dela kondizio hori, beraz if baldintza hori kendu dugu bat ere zentzurik ez zuelako bertan jarraitzeak.
- ➢ Minor + Smell Code:
  - Remove this unnecessary cast to ...
  - Remove this empty statement
  - Remove this unnecessary call

- ….
  Gehienbat honako arazo hauek dira beharrezkoak ez diren cast-ak kentzeko eskaera, ArrayList-ak sortzerako unean new ArrayList<>() deklarazioa jartzea, aldagaiak berrizendatzea patroi bat jarraitzeko: asko eta asko dira horrela aurki daitezkeen “arazoak” eta denak eguneratzeko denbora asko behar denez, saiatu gara batzuk kentzen

➢ Blocker + Vulnerability:

- Disable access to external entities in XML parsing
  Hau da arazo larriena aurkitu diguna gure Bets21 proiektu guztian, antza denez arazoa dator soilik barneko entitateei bakarrik uzten diela erabiltzen XML fitxategia, eta hori dokumentu motaren definizioan ezartzen da ea barneko entitateak ala kanpokoak erabil dezaketen.

  SonarLint-ek soluzio bat ematen du nola konpondu arazo hori, eta jarraian utziko dizuegu zuek ikusi dezazuen soluzioa; baina, gure jakinduria alorre horretan oso mugatua denez, eta ez dugunez oso ongi ulertzen fitxategi horien funtzionamendua erabaki dugu lehen zegoen bezala uztea:

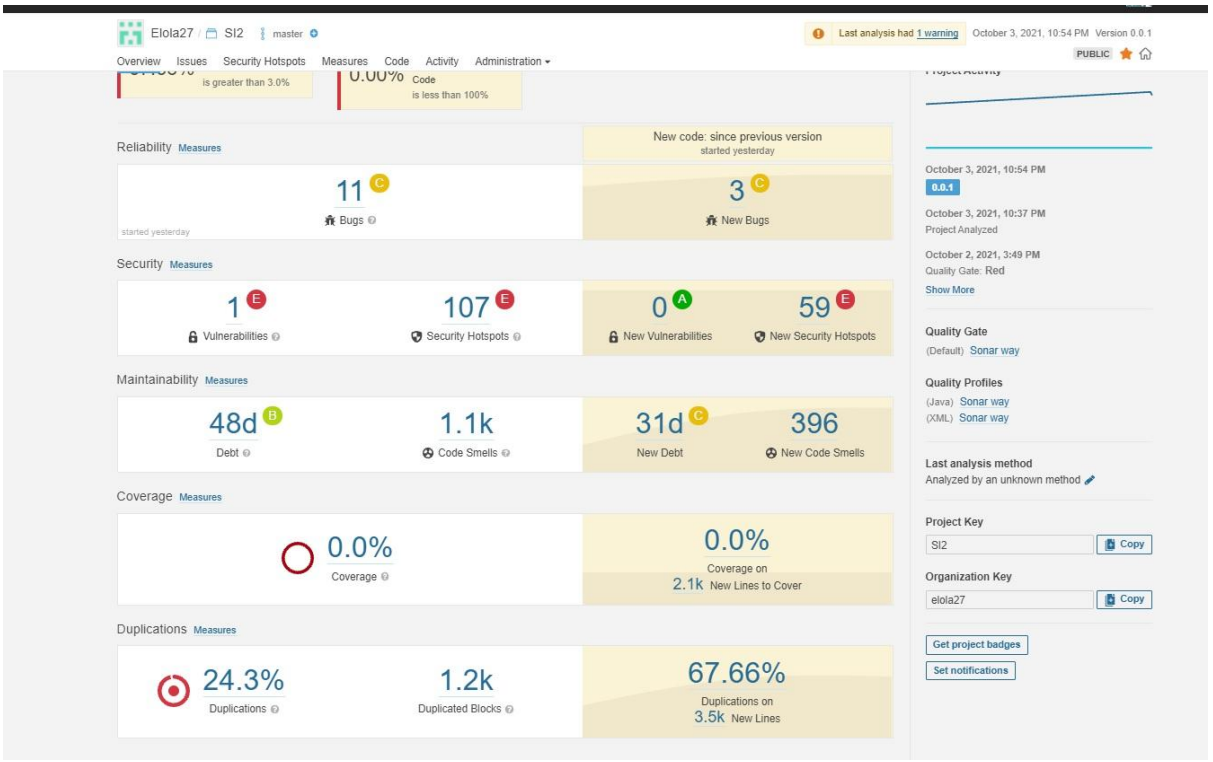
```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
// to be compliant, completely disable DOCTYPE declaration:
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
// or completely disable external entities declarations:
factory.setFeature("http://xml.org/sax/features/external-general-entities", false);
factory.setFeature("http://xml.org/sax/features/external-parameter-entities", false);
// or prohibit the use of all protocols by external entities:
factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_DTD, "");
factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "");


SAXParserFactory factory = SAXParserFactory.newInstance();
// to be compliant, completely disable DOCTYPE declaration:
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
// or completely disable external entities declarations:
factory.setFeature("http://xml.org/sax/features/external-general-entities", false);
factory.setFeature("http://xml.org/sax/features/external-parameter-entities", false);
// or prohibit the use of all protocols by external entities:
SAXParser parser = factory.newSAXParser(); // Noncompliant
parser.setProperty(XMLConstants.ACCESS_EXTERNAL_DTD, "");
parser.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "");

XMLInputFactory factory = XMLInputFactory.newInstance();
// to be compliant, completely disable DOCTYPE declaration:
factory.setProperty(XMLInputFactory.SUPPORT_DTD, false);
// or completely disable external entities declarations:
factory.setProperty(XMLInputFactory.IS_SUPPORTING_EXTERNAL_ENTITIES, Boolean.FALSE);
// or prohibit the use of all protocols by external entities:
factory.setProperty(XMLConstants.ACCESS_EXTERNAL_DTD, "");
factory.setProperty(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "");

TransformerFactory factory = javax.xml.transform.TransformerFactory.newInstance();
// to be compliant, prohibit the use of all protocols by external entities:
factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_DTD, "");
factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_STYLESHEET, "");

SchemaFactory factory = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
// to be compliant, completely disable DOCTYPE declaration:
factory.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
// or prohibit the use of all protocols by external entities:
```

```
factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_DTD, "");
factory.setAttribute(XMLConstants.ACCESS_EXTERNAL_SCHEMA, "");
```

Beste hainbat arazo konpontzen joan gara denboran zehar, baina bere momentuan ez ziren dokumentatuak izan eta orduan ezin dugu zehazki azaldu zein motakakoak izan diren zehazki.

AZKEN ANALISIA

Behin aldaketa guzti horiek egiten jardun ostean, azken analisi bat egin eta honako emaitzak lortu ditugu azken emaitzak:



Proiektu honetan lanean jardun garen bi ikaskideok ez dugu ulertzen nola gerta daitekeen akatsak zuzentzen jardun eta gero kalitatea okerragoa izatea lehen aldian igotakoa baino ere. Egia da proba kasuen fitxategiak ere sartu direla, baina hala ere ez dugu uste horiek horrelako kalitate txartze bat sortzeko aukera ematen dutenik.