# Miten sovellusta on testattu?

Sovellukselle on kirjoitettu testejä kolmella eri tavalla:

1. Java-luokkien yksikkötestit on toteutettu seuraavasti:  
   1. JUnit-testeillä, jotka on ajetaan tavallisina testitiedostoina
   2. JUnit-testeillä, jotka on ajetaan MockitoJUnitRunner-luokalla, ja joissa testattavan luokan riippuvuuksista on toteutettu Mockiton avulla tynkä-luokat. Tämä siksi, että riippuvuudet toimisivat ennustettavalla tavalla, ja testissä voidaan keskittyä vain testattavan luokan toimintaan sillä oletuksella, että riippuvuuksille on omat testit, jotka testaavat, että riippuvuus toimii oikealla tavalla, kun sen metodeja kutsutaan.
   3. JUnit-testeillä, jotka on ajetaan SpringJUnit4ClassRunner- luokalla käyttäen samaa ohjelman käynnistyskonfiguraatiota kuin ohjelmaa normaalisti käynnistettäessä. Siis testejä ajettaessa Spring käynnistetään ajamalla Launcher-luokan l aunch-metodi. Kun tällainen testi ajetaan, käyttäytyy testi- metodi kuin ohjelma olisi käynnistetty normaalisti, ja Spring olisi alustanut kaikista luokista “beanit” oliosäiliöönsä.
   4. JUnit-testejä, jotka on ajettu SpringJUnit4ClassRunner- luokalla, mutta käynnistyskonfiguraatio on määritelty itse erikseen. Tällaisia testejä on käytännössä vain yksi erikoistesti, jolla testataan, että WebSocket-yhteyden kautta lähetetyt viestit päätyvät kontrollerin metodille, ja vastaavasti kontrolleri- metodi kykenee lähettämään vastauksen WebSocket-yhteyden kautta.
2. Integraatiotestit on toteutettu JUnit-testeinä, jotka ajetaan käyttäen WebDriverRunner-luokkaa. Integraatiotesteissä sovellusta käytetään Selenium WebDriverin avulla, eli erilaisia sivun nappeja painellaan ja annetaan erilaisia syötteitä, ja testit vertaavat WebDriverin sivulla näkemää sisältöä odotettuun sisältöön varmistaakseen, että sovellus toimii oikealla tavalla.
3. Lisäksi JavaScriptillä luoduille metodeille on tehty yksikkötestejä Jasminella.

Pelkkien testien lisäksi olemme suorittaneet jatkuvasti tutkivaa testausta siten, että olemme avanneet yhteen selaimeen(esim. Google Chrome) kirjautuneen käyttäjän, ja toiseen erilaiseen selaimeen(esim. Mozilla Firefox) kirjautumattoman käyttäjän, jotta olemme voineet simuloida erilaisia asiakkaan ja hoitajan välisiä kanssakäymisiä, ja tällä tavalla tutkia ohjelman toimintaa.

# Miten testit ajetaan?

Sekä Javan-luokkien, että JavaScriptillä tehtyjen metodien yksikkötestit ajetaan käyttäen Maven-surefire-pluginia. Käytetty komentorivin komento on “mvn test”. Java-luokkien yksikkötestit löytyvät “src/main/test/java/sotechat”-kansion alta. JavaScript-metodien Jasmine-testit puolestaan löytyvät “src/main/test/webapp”-kansion alta.

Kun komento “mvn test” ajetaan, luo Jacoco-maven-plugin ajetuista Java-luokkien yksikkötesteistä raportin rivikattavuudesta. Raportti löytyy polusta “target/site/jacoco”, ja raporttia voi tarkastella web-selaimessa avaamalla “jacoco”-kansiosta löytyvän “index.html”-tiedoston.

Yksikkötestejä on myös mahdollista mutaatiotestata pitest-maven-pluginilla komennolla “mvn org.pitest:pitest-maven:mutationCoverage”. Mutaatiotestauksesta luodaan raportti “target/pit-reports”-kansioon. “Pit-reports”-kansion alakansiosta löytyy ylläolevan “jacoco”-raportin mukainen “index.html”-tiedosto, jonka voi avata selaimessa.

Hyväksymätestit eli tässä tapauksessa integraatiotestit löytyvät “src/main/test/java/integrationTests”-kansion alta. Hyväksymätestit ajetaan Maven-failsafe-pluginin avulla, ja käytetty komento on “mvn integration-test”.

# Checkstyle

Ohjelman funktionaalisuuden lisäksi olemme testanneet ohjelmakoodin ylläpidettävyyttä maven-checkstyle-pluginin avulla. Tämä plugin ajetaan komennolla “mvn checkstyle:checkstyle”, ja se luo “target/site”-kansioon “checkstyle.html”-nimisen tiedoston, jonka voi avata selaimessa, ja josta näkee kuinka hyvin ohjelmakoodi noudattaa “checkstyle.xml”-tiedostoon määriteltyjä koodin ulkoasua koskevia sääntöjä. “Checkstyle.xml”-tiedosto löytyy polusta “src/main/resources”.

Olemme määritelleet Checkstylen siten, että esimerkiksi seuraavanlaisessa koodissa Checkstyle huomauttaisi siitä, että ohjelmakoodi alkaa samalta riviltä kuin metodin nimi.

Ei siis näin..

public Conversation() {this.messagesOfConversation = new ArrayList<>();this.participantsOfConversation = new ArrayList<>();

}

..vaan näin:

public Conversation() {

this.messagesOfConversation = new ArrayList<>();

this.participantsOfConversation = new ArrayList<>();

}