# Testiautomaatiokysymykset kehitysprojekteille

Näiden kysymysten avulla on tarkoitus muodostaa malli testiautomaation kehittämiselle ko. projektissa ja auttaa testaussuunnitelman laatimisessa, sisältäen testausstrategian. Kysymysten avulla saadaan käsitys siitä, kuinka laaja testiautomaatio tarvitaan, miten se toteutetaan ja kuka pystyttää, kehittää ja ylläpitää sitä. Testiautomaation käyttöönoton paras ajankohta on uuden projektin alussa, mutta ei kuitenkaan ole mahdotonta ottaa sitä myös käyttöön projektin jo edettyä.

**Kysymykset**

**1. Suunnitellaan testiautomaation toteutus ja tarpeellisuus liiketoiminnan näkökulmasta**

**1.1 Miksi testiautomaatio on tärkeä toteuttaa ko. projektissa?**

*Onko sovelluksessa paljon liiketoiminnalle kriittisiä osa-alueita? Onko regresisiotestaukselle suuri tarve? → tällöin automaatio on kustannustehokkaampi ratkaisu*

**1.2 Mitä haluamme testiautomaatiolla saavuttaa?**

*Merkittäviä hyötyjä automaatiosta ovat: regressiotestauksen helpottuminen, palautteen lisääntyminen, luotettavuuden kasvu, ajan vapautuminen merkityksellisempään testaamiseen, yleinen laadun parantuminen, kustannussäästöt sekä ketterissä ohjelmistokehityksissä nopeammat julkaisusyklit*

**1.3 Miten testiautomaatio tukee manuaalista testausta?**

*Kaikkea ei voi ja pidä automatisoida. Parhaimmillaan nämä kulkevat rinnakkain oikeassa suhteessa. Tutkivaa testausta ei esim. voi automatisoida ja se on silti tärkeä osa testausta. Manuaalista testausta tarvitaan uusien virheiden löytämiseen ja uusien toiminnallisuuksien testaamiseen.*

**1.4 Mikä on projektin odotettu elinkaari?**

*Mitä pidempi odotettu elinkaari sen tärkeämpää on automatisoida testejä.*

**1.5 Kuinka iso ja kompleksinen on suunniteltu sovellus?**

*Testiautomaation laajuuden tärkeys korostuu isoissa projekteissa. Pienissä projekteissa, varsinkin jos toimintaympäristö dynaaminen, ei testiautomaatiota välttämättä tarvita, ainakaan laajassa määrin.*

**1.6 Projektin aikataulu, budjetti ja resurssointi?**

*Testiautomaatio maksaa itsensä takaisin testisuorituskertojen lukumäärän kasvaessa, alussa kustannukset isoimmat: testiympäristön pystytys, testien suunnittelu jne. Meillä käytetyt testaustyökalut ovat pääosin ilmaisia. Testien ylläpito vie myös kustannuksia.*

**2. Suunnitellaan teknisellä tasolla yhdessä tiimin kanssa mitä ja miten automatisoidaan**

**2.1 Millä tasolla testausta tehdään (API, GUI)?**

*Esim. Mikä sovelluksessa on tärkeää, onko UI isossa roolissa? kaupallisissa sovelluksissa UI:n rooli kasvaa vrt. sisäiset sovellukset.  Kuinka paljon aikaa/resursseja käytettävissä? API-testit nopeampia tuottaa.*

**2.2 Mitä ohjelmointikieltä/tekniikkaa käytetään testiautomaatiossa?**

*Tavoitteena yhtenäistää testikoodi ja kehityksen lähdekoodi, jos mahdollista. Ensisijainen työkalu testiautomaatioon on Playwright. Työkalu on käytettävissä ohjelmointikielillä: Node.js (JavaScript) -> parhaiten tuettu, Python, .NET (C#), sekä Java. Toinen vaihtoehto on Robot Framework + browser-kirjasto. Käyttökokemuksia eri kielistä:* [Playwright käyttökokemuksia - Digitalisaatioratkaisut - Apaja (elisa.fi)](https://apaja.elisa.fi/pages/viewpage.action?pageId=1557341842)

**2.3 Mikä mahdollinen CI/CD-työkalu on käytössä projektissa?**

*Varmistetaan sen toimivuus testiautomaation kanssa. Oletuksena automaattien julkaisujärjestelmä tarvitsee rinnalleen aina automaattiset testit.  Sovitaan staattisten koodianalyysityökalujen (sonarQube) käytöstä.  Sovitaan mihin repoon ja haaraan automaatiokoodit tallennetaan (oletuksena samaan repoon kuin muu koodi).*

**2.4 Onko olemassa/tarkoitus luoda erillistä testiympäristöä missä suoritetaan testiautomaatio?**

*Ideaalitilanne olisi, että testiautomaatio toimisi omassa ”steriilissä” ympäristössä*

**2.5 Tarvitseeko testidataa generoida erikseen? Kuka tekee?**

*Luodaanko testidataa erikseen joka ajossa, tarvitaanko skripti?*

**2.6 Millä selaimella ja päätelaitteilla (mobiili, desktop) testiautomaatiota tehdään?**

*Lähtökohtaisesti valitaan yksi verkkoselain ja se viimeisin julkaistu versio, jolla suoritetaan testiautomaatio (Useilla selaimilla testiautomaation kehittäminen on hidasta, johtuen selainperäisistä ominaisuuksista)*

**2.7 Onko aikaisempia projekteja, joita käyttää hyödyksi testiautomaation kehittämisessä?**

*Testiautomaation kehittäminen nopeutuu huimasti, kun voi käyttää hyödyksi aikaisempia testiautomaatioviritelmiä.*

**2.8 Onko tarve automatisoida myös ei-toiminnallisia asioita, kuten suorituskykyä tai kuormitusta?**

*Vähintäänkin staattinen koodianalyysi on hyvä ottaa mukaan julkaisuputkeen tarkistamaan koodin laadun.*

**2.9 Tarviiko testiautomaatioon tehdä jotain mock-toteutuksia?**

*Esim. onko sovelluksessa käytössä jotain 3.osapuolen järjestelmiä, joihin tarvitaan mock-toteutusta jäljittelemään oikeaa toimintaa, esim. sähköposti, tekstiviestit, korttimaksujärjestelmä.*

**3.Suunnitellaan toteutus: miten edetään, jotta saadaan testiautomaatio tukemaan ohjelmistokehitystä valitulla teknisellä toteutuksella**

**3.1. Onko tiimiläisillä halua työskennellä yli perinteisen rajojen**?

a.  Jos vastaus kyllä, sovitaan roolit ja vastuut

*Päävastuu testaussuunnitelmasta ja koordinoimisesta kuuluu aina testausasiantuntijalle. Kehittäjät voivat kuitenkin osallistua automaatiotestien kehittämiseen ja ylläpitoon sovitulla tavalla.*

b. Jos vastaus ei, pidetään tiimi kuitenkin ajan tasalla automaatiotekemisestä

*Esim. Tehdään testaussuunnitelma ja annetaan se tiimille katselmoitavaksi. Katselmoidaan tiimin kesken kaikki tehdyt testiautomaatiotestit.*

**3.2 Onko kehityksessä tarkoitus hyödyntää ketteriä menetelmiä (agile, devops)?**

*Ketterässä kehityksessä korostuu automatisoitujen reggressiotestien tärkeys, kun tuotetaan uusia toiminnallisuuksia nopealla syklillä. Kehitysprosessin pitää tukea testiautomaatiota (BDD, ATDD). Testiautomaatio suoritetaan mahdollisuuksien mukaan docker-kontissa.*

**3.3 Mitä asioita/toiminnallisuuksia automatisoidaan? Mitkä ovat sovelluksen tärkeimmät/kriittisimmät osa-alueet?**

*Lähtökohtaisesti aloitetaan automaatio aina tärkeimmistä käyttötapauksista. Laajat yksikkötestit vaikuttavat siihen, että testiautomaation ei tarvitse testata erikseen esim. jokaista syötevariaatiota.  Otetaanko mukaan vain onnistuneet tapaukset vai testataanko myös ns. laittomat syötteiden arvot.*

**3.4 Miten käsitellään/raportoidaan testiautomaation virheet testiajoissa?**

*Estääkö virheet CI/CD-putken läpimenon? Ajojen raportit sähköpostiin, teamsiin omalle kanavalle?  Minne bugihavainnot kirjataan ja miten määritellään kiireellisyys? Hyvä käytäntö myös olisi, että virheistä tulee automaattisesti kuvakaappaukset ja testien ajosta olisi olemassa videonauhoitus.*

**3.5 Miten ja mihin testaussuunnitelma ja muut testausdokumentit dokumentoidaan?**

*Testaussuunnitelma pitää olla projektitiimiläisten saatavilla Apajassa projektin sivujen yhteydessä. Testiautomaation lähdekoodin readme:hen kirjataan myös mahdollisimman kattavasti tietoa testiautomaatioratkaisusta, vähintäänkin setup ja millä automaatiokoodia pystyy kehittämään lokaalisti omalla koneella.*

**3.6 Onko syytä rakentaa ensin PoC?**

*Uusien asioiden äärellä, on hyvä aloittaa tekeminen PoC:n rakentamisella, jotta voidaan todentaa kyseisen ratkaisun toimivuus.*

**3.7 Miten testien ylläpito saadaan toimivaksi?**

*Testit on hyvä lähtökohtaisesti toteuttaa niin, että ylläpito on helppoa: globaalit muuttujat/keywordit samassa paikassa, ei koodiduplikaatteja, Page-oliot käytössä, stabiiilit elementtien lokaattorit . Yksi hyvä paikantamisstrategia on ottaa HTML-elementeille käyttöön test-id-attribuutin arvo, jota käytetään testiautomaatiossa. Testeistä on hyvä tehdä mahdollisimman luettavia, tarkoituksenmukaisia ja hyvin uudelleenkäytettäviä.*

**3.8 Tarvitaanko testeihin ns. abstraktiotasoa, jossa testit ovat avattu selkokielellä?**

*Onko olemassa tarve, että testejä haluaisi tarkastella henkilö, joka ei omaa teknistä taitoa ymmärtää niitä, esim. asiakas, liiketoiminnan puoli?*

**3.9 Missä vaiheessa testejä kehitetään ja suoritetaan?**

*Jos sovelluksen ulkoasu ja olemus hakevat alussa vielä paikkaansa, on testejä turha alkaa vielä toteuttamaan näihin ominaisuuksiin.*

**4. Käytetyt mittarit liittyen testiautomaation kehittämiseen ja tuloksiin**

**4.1 Kuinka monta automatisoitua testiä on olemassa?**

**4.2 Mikä on testien läpimenoaika?**

*Ideaalitilanne on, että testien palautesykli pysyy kohtuullisena. End-to-End testit ovat tyypillisesti melko hitaita, testien rinnakkaisajolla voi säästää aikaa.*

**4.2 Mikä on automaatiotestikattavuus?**

**4.3 Tuotantovirheiden määrä?**

*Olisiko testiautomaatiolla voitu välttää joitakin virheitä?*

**5. Yleiset haasteet liittyen testiautomaatioon**

**5.1 Ovatko testit vakaita?**

*Testien epävakaus on yleinen ongelma. Tämä johtuu suurelta osin verkkoviiveistä ja asynkronisista web-pyynnöistä. Testit voivat toimia eri tavalla riippuen missä ympäristössä ne ajetaan.  Tämän vuoksi olisi tärkeää saada testit nopeasti toimimaan, sillä alustalla mihin ne halutaan. Muut tärkeät seikat huomioida, on datan alustaminen ympäristöstä ennen automaatioajoa, ns. alustettu selainistunto jokaiseen testisuiteen. Satunnaiset virheet voi myös kiertää ajamalla testi uudelleen, jolloin tilastollisesti voidaan luottaa testin luotettavuuteen. Testien on myös hyvä olla riippumattomia toisistaan, jolloin yhdestä virheestä ei synny dominoefektiä.*