Ohjelmistokehityksen teknologioita - *S*eminaarityö

Playwright työkaluna järjestelmätestauksessa

1 Testaus

Santtu Lehtokari

3.5.2022

Sisältö

Tiivistelmä

Tässä työssä tutustutaan järjestelmätestaukseen Playwright -työkalun avulla. Tutustumisen kohteina ovat itse työkalun asennus ja käyttö projektin yhteydessä, testien määrittäminen sekä niiden ajaminen ja testitulosten raportointi.

Motivaattorina työn suorittamiseen on tutustuminen järjestelmätestaukseen aiheena ja erityisesti tähän Playwright -työkaluun osana sitä. Tässä työssä järjestelmätestit kohdennetaan ohjelmistoprojekti 2 -kurssin “Casino” front-endin testaamiseen, joka on toteutettu reactilla. Järjestelmätesteillä voidaan todeta frontin tiettyjen käyttötapausten toimivuus riippumattomasti, sillä järjestelmätestaushan on mustalaatikkotestausta, joka ei ole riippuvaista järjestelmän entuudestaan tuntemisesta.

Lopputuloksena työssä syntyy dokumentaatiota playwrightin käyttöönotosta projektissa, testitapaukset playwright yhteensopivana koodina ja viimeisenä tietysti itse testiajojen tulokset.

1 Johdanto

Tarkoituksena on tutustua järjestelmätestaukseen ja Playwright -työkaluun jolla sitä tässä työssä tehdään. Selvitettäviä asioita ovat Playwrightin käyttöönotto, käyttäminen, testien määritys ja testien ajaminen sekä tulosten analysointi. Tavoitteena on oppia käyttämään työkalua osana projektia, rakentaa testejä, ajaa testit ja saada onnistuneesti todettua jonkin asian toimivuus. Testien kohteena on ohjelmistoprojekti 2 -kurssin Casinon front-end.

Playwright on e2e eli end-to-end järjestelmätestausta varten tehty työkalu, joka on Microsoftin tekemä. Se tukee kaikkia nykyaikaisia moderneja selaimia, käyttöjärjestelmiä ja sen API-toimii monilla eri ohjelmointikielillä. Playwrightiä voi ajaa itsenäisenä testiajurina tai kirjastona esim Jestin tai Mochan kanssa. Työn lähteenä käytetään ainoastaan Playwrightin omaa dokumentaatiokirjastoa, ei muita tutoriaaleja taikka videoita. Työvaiheet alla.

1. Playwrightiin tutustuminen, käyttöönotto ja käyttö “Casino” -projektin kanssa

2. Testien määritys

3. Testien ajaminen

4. Testien lopputulokset ja analysointi

2 Playwright projektissa

2.1 Tutustuminen ja alkuvaihe

Ennen varsinaista tekemistä tutustuin Playwrightin dokumentaatioon ja sen eri käyttötapoihin. Yritin esimerkiksi asentaa Playwrightin toimimaan Jestin kanssa yhdessä meidän react fronttiprojektin yhteyteen. Tälläinen asennus kuitenkin tuotti ongelmia ja aikatauluhaasteiden vuoksi playwright-jest setti jäi unholaan ja tässä työssä keskitytään ainoastaan käyttämään pelkästään playwrightin omaa testauskirjastoa (playwright-test).

Playwright-jest asennus itsessään on ongelmallinen react-frontti työn kanssa mikäli projektin alustuksessa on käytetty CRA (Create-React-App) -työkalua. Tämän kyseinen työkalu ja sen tekemä paketti ylikirjoittavat konfiguraatiot joita tarvitsee esimerksi playwright-jestiä käytettäessä ja tekee käyttöönotosta monimutkaista sekä aikaavievää.

Alkuvaiheen tutkimuksissa käytin turhaa aikaa tuohon playwright-jest kombinaatioon ja niiden konfigurointeihin kunnes ymmärsin että sitä ei kannata tuon CRA:n kanssa yrittää edes säätää. Tein mm. jest-config.js, playwright-jest.config tiedostoja joiden avulla pyrin vaikuttamaan siihen miten jest ja playwright ajaa noista testejä, mutta ongelmaksi muodostui jopa projektin käynnistys ja buildaus tuon CRA:n takia. Sen takia tässä työssä keskitytään nyt playwrightin omaan testiajuriin pelkästään.

2.2 Käyttöönotto ja asennus

Playwrightiä voi käyttää itsenäisenä testiajurina tai jonkun toisen testausajurin yhteydessä. Tässä työssä asennan playwrightin mukaan projektiimme itsenäisenä testiajurina ja käytämme kirjastona (playwright-test).

Luon myös Playwrightin testitapauksille oman “e2e” nimisen kansion projektimme juureen, jonne rakennamme varsinaiset testit.

```
yarn add -D playwright playwright-test
```

2.3 Käyttäminen ja komennot

Playwrightin ajaminen on hyvin yksinkertaista ilman toista testausajuria. Komentojen ajamiseen käytetään NPX-komentoa, joka on tarkoitettu node pakettien ajamista varten.

Kun playwright on asennettu sitä ajetaan alla olevalla komennolla. Ajuri osaa etsiä itse playwright testit projektin kansiosta ja alkaa suorittamaan niitä.

3 Testit

3.1 Asennuksen jälkeinen demo

Testataan nopeasti pienellä testillä että playwright toimii asennuksen jälkeen projektissa ja suorittaa testin/testejä. Sitä varten teen todella yksinkertaisen testin ja suoritan ajokomennon npx playwright test.

```
const { test, expect } = require('@playwright/test');

test('Holdem renders', async ({ page }) => {
  await page.goto('http://localhost:3000/holdem');

  await expect(page).toHaveTitle(/React App/);

});
```

```
PS C:\nodejs\projects\casino_front\casino_frontend> npx playwright test

Running 1 test using 1 worker

[WebServer] (node:24088) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE
tupMiddlewares' option.
(Use `node --trace-deprecation ...` to show where the warning was created)
[WebServer] (node:24088) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWAR
setupMiddlewares' option.
  ✓  src\e2e\App.test.js:3:1 › Holdem renders (9s)


  1 passed (13s)
PS C:\nodejs\projects\casino_front\casino_frontend>
```

Homma pelittää projektissa, siirrytään varsinaisten testien pariin.

3.2 Testitapauksista

Testitapauksina käytetään kasinolle rekisteröitymistä, kirjautumista ja pokeripöytään liittymistä. Koska seminaarityön deadline on viikkoa ennen kasinon viimeisten täskien toteutumista ei tähän työhöön mennessä kyetä vielä tekemään laajempaa testausta playwrightilla. Varsinkin pokeripuolen testaaminen olisi hankalaa ja monimutkaista kun osa toiminnoista valmistuu vasta loppuviikosta ja seminaarityön pitää olla palautettuna viikkoa ennen hyvissä ajoin ennen projektin Sprint 4 demoa.

3.3 Testitapaustaulukko

Testitapaus- ja kuvaus	Syötteet	Odotetut tulokset
R_T1 Rekisteröityminen palveluun oikeilla tiedoilla käyttäjäksi	Käyttäjätunnus: Playwright Salasana: Playwright Salasana uudelleen: Playwright	Rekisteröinti onnistuu, koska annetut tiedot täyttävät vaatimukset
R_T2 Rekisteröityminen palveluun puutteellisella salasanalla	Käyttäjätunnus: Playwright Salasana: 123 Salasana uudelleen: 123	Rekisteröityminen ei onnistu, koska salasana ei täytä asetettuja pituusvaatimuksia
R_T3 Rekisteröityminen palveluun tunnuksella, joka on jo varattu	Käyttäjätunnus: Playwright Salasana: Playwright Salasana uudelleen: Playwright	Rekisteröityminen ei onnistu, koska järjestelmä ilmoittaa käyttäjätunnuksen olevan varattu
R_T4 Rekisteröityminen palveluun ilman määritettyä tunnusta	Käyttäjätunnus: Salasana: Playwright Salasana uudelleen: Playwright	Rekisteröityminen ei onnistu, koska tarvittavia tietoja ei ole annettu
K_T1 Kirjautuminen palveluun oikeilla tiedoilla	Käyttäjätunnus: Playwright Salasana: Playwright Salasana uudelleen: Playwright	Kirjautuminen onnistuu, sessio luodaan. Järjestelmä kuittaa sisäänkirjautumisen.

K_T2 Kirjautuminen palveluun virheellisellä salasanalla	Käyttäjätunnus: Playwright Salasana: abc Salasana uudelleen: abc	Kirjautuminen epäonnistuu, järjestelmä ilmoittaa sisäänkirjautumisen epäonnistumisesta virheilmoituksella
K_T3 Kirjautuminen palveluun virheellisellä käyttäjätunnuksella	Käyttäjätunnus: Wright Salasana: abc1 Salasana uudelleen: abc1	Kirjautuminen epäonnistuu, järjestelmä ilmoittaa sisäänkirjautumisen epäonnistumisesta virheilmoituksella
~~**K_T4 Palvelusta uloskirjautuminen**~~		~~Kirjautuminen ulos onnistuu ja järjestelmä ilmoittaa uloskirjautuksesta.~~

3.4 Testien määritys Playwrightin tyyliin

Koodasin testitapaustaulukossa määritetyt testit playwrightin dokumentaation avulla ja lisäsin ne e2e kansioon. En nähnyt järkeväksi kopioida tähän kaikkia koodeja, joten testitapausten koodit löytyvät linkin takaa.

https://github.com/Agile-Applet/Front-End/tree/playwright-testing/src/e2e

Haastavaa testien määrityksestä teki react ja meillä käytössä oleva material-ui kirjasto. Onneksi kuitenkin assertointi onnistuu luokkien avulla ja myös käyttöliittymän elementeillä on data-testid attribuutit, joista voi ottaa kiinni elementtien löytämiseksi.

4 Testaus Playwrightilla

4.1 Testauksen puitteista

Testaus suoritetaan ajamalla npx playwright test komento, joka käynnistää paikallisen buildin frontista porttiin 3000 ja ajaa tehdyt playwright testit itsenäisinä selainprosesseina.

Back-end on testauksen aikana pyörimässä localhostissa ja ei ole riippuvainen tästä frontin kokonaisuudesta. Ts. tällä frontin osuudella ei ole mitään tekemistä bäkin pyörityksen

kanssa ja bäcki voisi yhtä hyvin olla nyt internetissä jota testataan, vaikka frontti menee localhostina. Back-end on viimeisin ajantasaisin versio testien aikana.

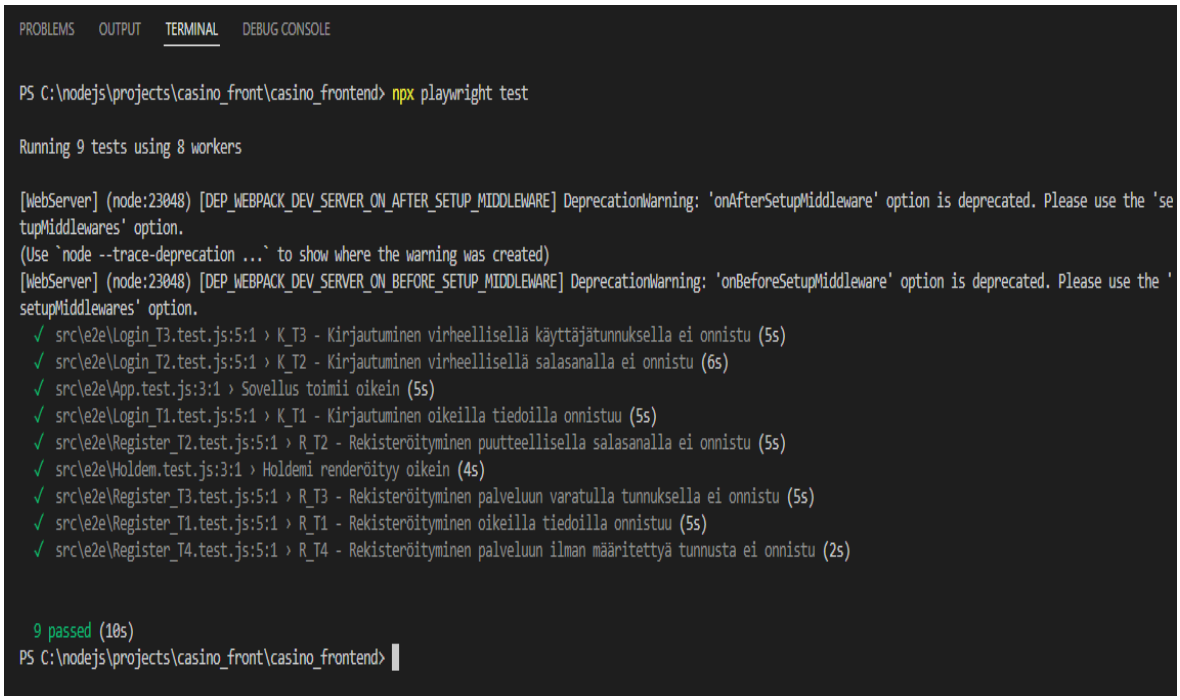
4.2 Testien ajaminen

Testit ajetaan komennolla “npx playwright test”, jonka jälkeen odotellaan tuloksia. Tästä materiaalia videolla.

4.3 Tulokset ja analysointi

Kaikki testitapausluettelossa olleet testit menivät määritysten mukaisesti läpi. Huomioitavaa oli se, että tein useamman ajokierroksen ennen tätä viimeistä testauskierrosta.

Alussa rakensin kaikki testit samoihin tiedostoihin (Esim. Login.test.js, Register.test.js) mutta myöhemmin pilkoin ne testitapauksen mukaan erilleen, jotta playwright osaa ajaa niitä monisäikeisenä. Tällä tavoin testien suoritusaikaa saatiin hieman pienennetyä kun kaikki testitapaukset eivät sijaitse samassa tiedostossa.



5 Yhteenveto

Vaikka käytin työkalun opetteluun ja itse työhön suositellun työajan verran (-20h) aikaa tuntuu silti siltä että jäi vielä paljon opittavaa. Tämä todella oli pintaraapaisu tälläisestä e2e testauksesta ja työkalusta.

Yhteenvetona voisi sanoa, että mikäli järjestelmätestausta aikoo soveltaa johonkin omaan projektiin on hyvä perehtyä valittuun työkaluun rauhassa jo ennen testitapausten miettimistä. Lisäksi käyttöliittymäsuunnittelussa tulisi jo huomioida testausta varten tarvittavat attribuutit tai luokat, jotka auttavat testitapausten kirjoittamisessa koodiksi. Itsellä oli hieman pulmaa rakentaa suunnittelemani testitapaukset koodiksi juurikin sen takia, että kaikilla elementeillä ei ollut järkevää css-luokkaa tai attribuuttia johon tarrata kiinni testien koodissa, joten jouduin käyttämään enemmän aikaa itse testien kirjoittamiseen vaikka logiikka oli sinänsä selvä.

Johtopäätöksenä voin todeta että tälläinen testaustyökalu on oikein käytettynä erittäin hyödyllinen ja tervetullut, kunhan ohjelmistokehityksen eri vaiheissa huomioidaan se että testausta tullaan tälläisellä työkalulla tekemään. Itse laatimani testit menivät läpi onnistuneesti ja testit saatiin lopulta suoritettua alun haasteista riippumatta.

Itselleni tämä oli erittäin mielenkiintoinen ja hyvä oppimiskokemus. Tämä oli ensimmäinen kerta kun käytin vastaavaa työkalua ja tein sitä varten konkreettiset testitapaukset koodimuotoisiksi. Pidän tätä osaamista hyödyllisenä jatkoa ajatellen. e2e automaattitestauksella voi vähentää erittäin paljon manuaalista työtä, joten tulen varmasti jatkossakin käyttämään Playwrightia tai vastaavaa työkalua osana ohjelmistotestausta.

Järjestelmätestausta itsessään voisi vielä opiskella lisää ja tietysti tuota työkalua. Playwrightissa on ihan älyttömästi mahdollisia tapoja tehdä noita testejä, joten sen hyvään haltuunottoon saa varmastikin kulumaan aikaa. Tätä työtä voisi jatkaa meidän holdemin testaamisella ja ottamalla mukaan monipuolisempia toiminnallisuuksia playwrightista.

Työssäni en käyttänyt muita lähteitä kuin Playwrightin oma nettidokumentaatio. Kaiken kaikkiaan tämä oli opettavainen ja hyödyllinen kokemus.

Github (menee suoraan testeihin): https://github.com/Agile-Applet/Front-End/tree/playwright-testing/src/e2e

Video: https://haagahelia-my.sharepoint.com/:v:/g/personal/bgp050_myy_haaga-helia_fi/EZ75lZCcrP9KrUBoa6ub9FUBSXoT0ZkYXeW1Dw43Psn5hQ?e=OOUvp0

Lähdeluettelo

Playwright. (2022). Playwright: Docs. URL: https://playwright.dev/docs/intro