TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Tatjana Levizi 206198IABB

Microsoft Dynamics 365 Business Central automaattestid Connector App'i jaoks

Bakalaureuseõpe lõputöö

Juhendaja: Rivo Lemmik
PhD

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Tatjana Levizi

19.02.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua ja käivitada Business Central'i automaattestid Connector App'i liidese jaoks, mis seob omavahel D365 Business Central'i majandustarkvara ja Alldevice'i hooldustarkvara omavahel.

Enne testide kirjutamist uuriti Microsofti materjali ning Connector App'i lõputööd, mis kirjutas TalTechi lõpetaja Rainis Mäemees - ning pandi kirja eesmärgid ja skoop.

Bakalaureusetöö tulemusena valmis automaattestid Connector App'i liidese jaoks, tänu millele saab vea eelnevalt tuvastada ja vältida rakenduste ebakõlasid Microsofti uue versiooniga.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 48 leheküljel, 5 peatükki, 35 joonist, 2 tabelit.

Abstract

Microsoft Dynamics 365 Business Central automated tests for the Connector App

The purpose of this bachelor thesis is to create and run Business Central automatic tests for the Connector App interface, which connects D365 Business Central business software and Alldevice maintenance software together.

Before writing the tests, Microsoft's material and Connector App thesis, written by TalTech graduate Rainis Mäemees, were studied - and the goals and scope were written down.

As a result of the bachelor's work, automatic tests were completed for the Connector App interface, thanks to which the error can be detected in advance and the inconsistencies of the applications with the new Microsoft version can be avoided.

The thesis is written in Estonian and contains text on 48 page, 5 chapter, 35 figure, 2 table.

Lühendite ja mõistete sõnastik

Business Central	Majandustarkvara.
Alldevice	Hooldustarkvara.
AppSource	Veebipood, mis sisaldab ärirakendusi ja teenuseid.
D365	*Dynamics 365*.
Connector App	Liides, mis seob Microsoft D365 Business Central'i ja Alldevice hooldustarkvara omavahel.
ERP	Ettevõtte ressursside planeerimise tarkvara.
SaaS	Tarkvara teenusena ehk pilvetarkvara.
OnPrem	Kasutatakse haldamiseks ettevõte füüsilisi servereid ja IT-infrastruktuuri.
NAV	Majandustarkvara.
ATDD	Vastuvõtu testipõhine arendus.
Codeunit	Koodiüksus, mis sisaldab enda sees meetodeid.
API	*Application Programming Interface*, rakendustarkvara liides
AL	Programeerimiskeel.
Codeunit	Koodiüksus.
JSON Token	On avatud standard (RFC 7519) JSON-vormingul põhinevate juurdepääsulubade loomiseks.
JSON Path	JSON Path on JSON-i päringukeel, mis sarnaneb XML-i XPathiga.
JSON Key	On JSON-i andmestruktuur krüptograafiliste võtmete

	esitamiseks.
Test Toolkit	Rakendus kirjutavate testide automatiseerimiseks ja käivitamiseks.
Code Coverage	Vahend, mis sisaldab andmeid automaatse testi täitmise ja rakenduse koodi testiga hõlmatud ulatuse kohta.
Postman	Postman on API-platvorm API-de loomiseks ja kasutamiseks.

Sisukord

Jooniste Loetelu

Tabelite Loetelu

1 Sissejuhatus

Automaattestid on vaja selle jaoks, et neid käivitada pärast igat muudatust ja kiiresti saada tagasisidet. D365 Business Central'i majandustarkvara jaoks on vaja ka kirjutada teste, vaatamata sellele, et Microsofti poolt on kirjutatud oma testid enne rakenduse AppSource'i panemist. Lisaks Microsoft soovitab kõigile kirjutada oma rakenduse jaoks oma automaatteste.

Bakalaureusetöö eesmärgiks kirjutada automaatteste D365 Business Central'i ja Alldevice hooldustarkvara Connector App liidese jaoks. Lisaks analüüsida testide kirjutamist ning teha üldistatud järeldused, mis kehtivad erinevate liideste (Connector App'i) puhul sarnaselt, et töö tulemus oleks laiendatav ühelt projektilt, sarnaste vajadustega valdkonnale.

Töö teoreetilises osas tutvustatakse Business Central'i tarkvara, automaatestide. Samuti autor uurib liidest, mis on loodud D365 Business Central'i majandustarkvara ja Alldevice hooldustarkvara vahel.

Töö praktilises osas kirjutatakse automaatteste Connector App'i jaoks, mille looja on Rainis Mäemees. Automaattestide kirjutamiseks jälgib töö autor Microsoft'i poolt loodud parimaid tavasid, nii koodi stiili kui ka struktuuri osas.

Bakalaureusetöö teema valik tuleneb autori huvist D365 Business Central'i tarkvara arendamisest ja testimisest. Uued teadmised saab autor kasutada tänase töökohal, mis on seotud D365 Business Central'i tarkvara arendamise ja testimisega.

2 Probleemi püstitus ja projekti eesmärk

Käesolevas peatükis kirjutab lahti töö autor metoodikast, tööriistadest, probleemi püstitusest ja eesmärgist, mis on seotud antud tööga.

2.1 Taust

2.1.1 Microsoft Dynamics 365 Business Central

Microsoft Dynamics 365 Business Central on majandustarkvara. Teiste sõnadega see on vananenud Microsoft Dynamics NAV-ist uus versioon, kus on uus funktsionaalsus, kujundus ja võimalused. Nende erinevuses seisneb sellest, et NAV'i saab alla laadida ainult lokaalsesse serverisse, kuid Business Central'il on kaks võimalust: pilveteenus, lokaalne server.

Business Central kasutades saavad ettevõtted optimeerida oma äriprotsesse, mis aitavad oma ettevõtet juhendada: finantsid, tootmine, müük, tarnimine, projektijuhtimine, teenused ja palju muud. See on valmis rakendus, kuhu saavad lisada oma funktsionaalsust ERP arendajad, selleks arendajad kasutavad Visual Studio Code, sest Business Central on ehitatud AL koodi keeles [1].

Business Central'is saab paigaldada keskkonda kahte moodi: kas pilveteenusena (*Saas*), kus saab kasutada Business Central'i interneti kaudu ning andmed säilitatakse Microsofti pilves; teine võimalus on installida Business Central'i privaatsesse lokaalsesse serverisse (*OnPrem*) ning juurde on vaja installida SQL andmebaasi, kuhu säilitatakse andmed [2].

Business Central'i keskkondade põhierinevuseks saab tuua, et pilve ERP-tarkvara eesmärk on pakkuda töötajatele paindlikkust ja väiksemaid organisatsioonilisi kulusid, kuid kohapealne ERP-tarkvara eesmärk on pakkuda organisatsioonile täielikku kontrolli, sealhulgas turvalisust [3].

Pärast keskkonna valimist, kus hakatakse tegema arendust, on vaja veel valida keskkonda, kas *Sandbox* või *Production*. Nende valimine sõltub sellest, mida on vaja ettevõttele, aga tavaliselt ettevõtted loovad ühe *Production* ja üht või rohkem *Sandbox* keskkonda. Selleks, et arendused

ja testimised ei katkesta kliendi tööd on mõeldud erinevaid keskkonnaid. *Production* keskkonda kasutatakse siis, kui on valmis lahendus ja ettevõtte või klient saab kasutada oma tavatöös. *Sandbox* keskkonda kasutatakse siis, kui on vaja midagi juurde arendada ja pärast testida nii automaatestidega, kui ka käsitsi [3].

Tabelis 1. on välja toodud automaattestide kirjutamiseks Business Central'i keskkonna valikute plussid ja miinused. Võrreldakse oma vahel pilveteenus ja lokaalne server.

Tabel 1.Automaattestide kirjutamiseks Business Central'i keskkonna valikute plussid ja miinused.

	Pilveteenus	**Lokaalses serveris majutamine**
Sandbox	Jah	Jah
Production	Ei	Jah

Kui valik jäi *Sandbox*'i keskkonna peal, siis saab valida, kas käivitada keskkonda, mis on juurutatud Dynamics 365 Business Centrali teenusena, või konteineri põhjal, mida hostitakse Azure'i virtuaalmasina või lokaalselt. Mõlemad valikud pakuvad AL arendustööriistu [4].

Kui seadistatakse konteineri Microsoft Azure'i abiga või lokaalselt, siis mõlemad pakuvad sama kogemust ja kasutavad Docker'it konteineripõhise rakenduse infrastruktuuri pakkumiseks. Erinevus on nende vahel järgmine:

- Azure'is hostimisel on Docker installitud ja konfigureeritud automaatselt. Azure'i hostimine nõuab aga Azure'i litsentsi tellimist ja iga konteineri Sandbox'i eest võivad kehtida lisatasud [5].
- Kohalik hostimine nõuab, et arvutis oleks installitud Windows 10, Windows Server 2016 või Windows Server 2019 ning enne konteineri Sandbox'i seadistamist peab installima ja konfigureerima Docker'i [5].

2.1.2 Automaattestid

ISTQB® testimise automatiseerimise definitsioon on "tarkvara kasutamine testide läbiviimiseks või hooldamiseks". Võib ka öelda: "Testi automatiseerimine on käsitsi tehtud testide täitmine masinate poolt." Seega hõlmab kontseptsioon kõiki tarkvara kvaliteedi

testimise tegevusi arendusprotsessis, sealhulgas erinevaid arendusetappe ja testimise tasemeid, aga ka projektiga seotud arendajate, testijate, analüütikute ja kasutajate vastavaid tegevusi [7].

Sellest lähtuvalt ei ole testimise automatiseerimine mitte ainult testide komplekti täitmine, vaid kogu igat tüüpi testtarkvara loomise ja juurutamise protsess. Teisisõnu, kõik tööd, mida on vaja automatiseeritud testide kavandamiseks, läbiviimiseks, hindamiseks ja nende kohta aruandluseks [7].

Asjakohane testtarkvara sisaldab [7]:

- Tarkvara
- Dokumentatsioon
- Katsejuhumid
- Testikeskkond

Automaattestide eesmärgid [7]:

1. Parandada testimise tõhusust ja seega vähendada testimise üldkulusid.
2. Testi täitmise aja lühenemine, lühemad testitsüklid ja sellest tulenevalt võimalus testi täitmise sagedust suurendada.
3. Kvaliteedi säilitamine või parandamine.

Üldised automaattestid võib näha Java, Python, C# keeldes. Business Central ei ole erijuhtum ja vaatamata sellele, et Microsofti poolt on tehtud oma automaattestid, on ikka vaja testida oma arendust. Tabelis 2. Business Central'i automaattestimise kohta on toodud plussid ja miinused [8].

Tabel 2. Business Central'i automaattestimise plussid ja miinused.

Plussid	**Miinused**
Sõitke testimisega ülesvoolu ja säästke kulusid.	Kulud on liiga kõrged ja muudavad ettevõtteid konkurentsivõimetuks.
Testimise automatiseerimine vabastab aega	Ettevõtte ei ole harjunud seda nii tegema ning

igapäevaseks äritegevuseks.	ettevõte müügi- ja juhtimismeeskonnad ei näe sellest kasu ega ole "veendav".
Testimise automatiseerimise tõttu jätkake erinevate projektidega tegelemist.	Testimise automatiseerimiseks on liiga palju erinevaid projekte.
	Microsoft'il on testid automatiseeritud ja Dynamics 365 Business Central pole siiski vigadeta.
	Ettevõtete igapäevane äri ei jäta ruumi uue distsipliini lisamiseks.

Microsoft'il on enda rakenduste turuplats (*AppSource*), kuhu ei nõuda rohkem Microsoft automaatteste, sest Microsoft'i enda pool on loodud automaatteste, mis käivad läbi publitseerinud rakendust turuplatsi. Siiski rangelt soovitatakse kasutada automatiseeritud teste. Lisaks tulevikus hakkavad kliendid seda üha enam nõudma, nähes, et konkurendid harjutavad ja pakuvad automatiseeritud testimist. Viimane annab konkurendile eelise, kuna tal on rohkem aega funktsioonide lisamiseks ja sagedamini väljalaskmiseks [8].

2.1.3 Connector App

Connector App on rakendus, mis on tehtud Business Central'ile, mis sünkroniseerib andmeid Alldevice'i ja Business Central'i vahel ja pärib andmeid Alldevice'ist [9].

Selle rakenduse tehti, sest Business Central'i ja Alldevice'i vahel ei olnud liidest ning ettevõtete poolt on suurenes nõudlus, kuna nad kasutavad seda oma tava töös. Pärast rakendamise loomist, kasutajad võivad rohkem mitte käsitsi andmeid sisestama mõlemasse tarkvarasse, vaid andmete sünkroniseerimine toimub mõlema tarkvara vahel automaatselt. Lisaks sellele on tehtud selline võimalus, et saab vaadata hooldustöö käskusid Business Central'is ja luua kaubažurnaali ridasid hooldustöödes kasutatud varuosadest, mis aitab laovarude seisu hoida aktuaalsena ning parandab varude tarne ja nõudluse õigeaegset planeerimist [9].

2.2 Probleemipüstitus

Autori töö käigus, mis on seotud Business Central'i arendamisega, on pannud tähelepanu sellele, et rakenduse jaoks testide kirjutamine ei ole arendaja tava töös, sest Microsofti poolt on tehtud oma automaattestid, kuigi testide katvus ei ole 100%. Seetõttu Microsoft rangelt soovitab kirjutada oma teste vaatamata sellele, et arendus ei lähe *AppSource*'i.

Autori töökogemusega arenduse testimine toimub järgmise protsessiga:

1. Konsultant räägib kliendiga ja kirjeldab töö lahti arendajale, mis on vaja teha.
2. Arendaja kirjutab koodi ning selle käigul testib käsitsi. Kui kõik toimub suunab konsultandile tagasi testimisele, mis toimub käsitsi.
3. Kui arendus töötab, paneb konsultant kliendi keskkonda testimisele; kui mitte - suunab tagasi arendajale koos tagasisidega.

Selle protsessi tutvumisega on arusaadav, et ei ole ühtegi sõnat automaattestimisest. Seega tänapäeval see on suur probleem Business Central'i arendamisel. See probleem kokku puudub Connector App-iga. Tänu autori poolt loodud teste liidese jaoks võiks ette tuvastada erinevaid vigu ja parandada neid nt. uue versiooni kompileerimise jooksul või enne publitseerimist.

2.3 Lähtetingimused

Eeltingimusel peaks ettevõte kasutama Business Central'i ja Connector App'i, mida saab Business Central'ile keskkonnas seadistada, või kui planeeritakse need tarkvarad kasutada lähiajal.

2.4 Eesmärgid

Eesmärgid mida antud automaattestid peaks täitma:

- Testid on kirjutatud 100% katvusega.
- Vigade tuvastamine testide käivitamise ajal.
- Käsitsi testimise vähendamine.

2.5 Metoodika

Töö autor analüüsib antud töö raames automaattestide dokumentatsiooni ning lisaks loeb selle teemaga seotud raamatuid. Samuti autor uurib automaattesti dokumentatsiooni, mis on tehtud Microsoft'i poolt. Lisaks sellele uurib toodud näited, mis on saadaval Internetis või raamatustes, kuna automaatteste koodid ei ole avalikud ning on vähe näiteid, sest nagu üles mainitud ei ole automaatteste kirjutamine ettevõttes üldises töös.

3 Lahenduste analüüs

3.1 Parimate tavade analüüs

Et *AppSource*'i lisada rakendusi peavad olema täidetud nõuded, mis on seatud Microsoft'i poolt [10]. Üks nende nõuetest on testida oma rakendust enne publitseerimist. Kuigi Connector App ei ole publitseerinud Microsoft'i turuplatsi, autor ikka jälgib neid nõuded, kuna tulevikus liides võiks olla pannud *AppSource*'i.

Parimad tavad mida enne automaattestide kirjutamist tuleks jälgida, et testid oleksid tõhusad:

- On vaja saada aru, miks tuleks teste kavandada ja välja töötada enne nende kirjutamist ja teostamist. Täielik plaan kirjeldab eri tüüpi teste, mida tuleb läbi viia, näiteks jõudlus-, rakendus- ja turvatestid; tingimused, mille alusel need peavad olema täidetud; ja kriteeriumid, mis muudavad need edukaks [8].

- Uurida, kuidas olemasoleva funktsiooni jaoks testiplaani seadistada. Selles plaanis peavad olema ette kirjutatud missugused peavad olema nii positiivseid, kui ka negatiivseid teste. Teiseks on vaja testid erastada, missugused on automaattestid, missugused - mitte. Kui on automaattestid siis tuleks täpsustada, kas äriloogikat testitakse ilma kasutajaliidesele juurdepääsuta ja need, mis nõuavad kasutajaliidese funktsioonide testimist (UI testid). Ja viimaseks list peab sisaldama testide prioritiseerimist [8].

- Pärast plaani kirjutamist on vaja uurida erinevaid testide ja testide jaoks andmete seadistamiste põhimustreid [8].

Lisaks sellele on olemas parimad tavad, mida on vaja jälgida automaattestide kirjutamise ajal:

- Testkoodi tuleks hoida testitavast koodist eraldi. Nii saab testitud koodi tootmisse avaldada ilma testkoodi avaldamata [11].

- Automaattestid ei tohiks nõuda kasutaja sekkumist [11].

- Testid peaksid jätma süsteemi samasse üldtuntud olekusse, milles need käivitati, et saaks testi uuesti käivitada või muid teste suvalises järjekorras ja alustada alati samast olekust [11].

- Kasutada testides rangelt väärtusi ainult siis, kui neid tõesti vajate. Kõigi muude andmete puhul kaaluge juhuslike andmete kasutamist [11].

ISAIV raames töö autor on loonud Postman'is API teste, kus on samuti omad parimad tavad. Need on järgmised:

- Reaalsete andmete kasutamine – mida täpsemalt testiandmed kajastavad tingimusi, millega API tootmises kokku puutub, seda põhjalikum ja täpsem on testimisprotsess. Parim viis testiandmete realistlikkuse tagamiseks on alustada allikast – äriprotseduuridest, mille toetamiseks API on loodud [16].

- Positiivsed ja negatiivsed testid – see on vaja, et testid on käinud läbi erinevate viisil, nii õnnestuse kui ka ebaõnnestuse korral, et tuvastada viga [16].

- Kasutage andmeid dünaamiliste väidete esitamiseks - saab sisendandmete komplekti hõlpsasti lisada uusi testistsenaariume, ilma et oleks vaja funktsionaalset testimist ise muuta. Ja kui tarnepoliitika muutub, peavad muutuma ainult testi andmed ja väited – kõik muu jääb samaks [16].

- Jälgige API vastuseid – kui API-s tehakse mitu muudatust ja regressioonitesti käigus avastatakse uus viga, võib olla tohutu ülesanne täpselt kindlaks teha, milline muudatus vea põhjustas. Salvestatud API päringute ja vastuste teegiga tutvumine muudab uue probleemi ilmnemise hetke tuvastamise ja selle parandamise palju lihtsamaks [16].

- Andmepõhised funktsionaalsed testid jõudluse ja turvalisuse tagamiseks taaskasutamine - andmepõhise funktsionaalse testi taaskasutamine toob jõudluse ja turbe hindamise protsessidesse terve annuse tegelikkust ning suurepärased tööriistad, nagu ReadyAPI, muudavad selle ülemineku lihtsaks [16].

Võrreldes oma vahel Business Central'i ja API jaoks loodud automaatteste, töö autor võib öelda, et on olemas sarnasused nagu positiivsete ja negatiivsete testide loomine. Erinevuseks võib tuua see, et Business Central'il testkood tuleks hoida testivast koodist eraldi, kuid API korral testid kirjutatakse samal kohal, kus tehakse päringud. Kokkuvõtteks, saab öelda, et

automaattestide puhul on olemas üldised parimad tavad, kuigi igal tehnoloogial on olemas oma spetsiifika ja enne automaattestide kirjutamise on vaja tutvuda esialgselt selle tehnoloogia automattestide loomise dokumentatsiooniga.

3.2 Automaattestide ülesehituse Business Central'is analüüs

3.2.1 Testide Codeunit ja testide meetodeid

Business Central'i pilveteenuste rakenduste arendamisel peab andma igale objektile unikaalse numbri ja nime. Lisaks tuleb jälgida veel, et objektide numbrid ei kattuks teiste rakenduste objekti numbritega, muidu lähevad rakendused konflikti ja kood ei kompileeru [12].

Business Central'is kirjutatakse teste *Codeunit* objektis. *Codeunit* on AL-koodi konteiner, mida saab kasutada paljudes rakendusobjektides. Tavaliselt rakendatakse äriloogikat *Codeunit'*is ja ning kutsutakse välja teistes objektides, mis peab seda konkreetset loogikat täitma. Näiteks *Page* objektil on nupp, mis moodustab uue kliendi, *Codeunit*'is kirjeldatud äriloogika põhjal [13].

```
Codeunit 50113 CreateCustomer
{
  TableNo = Customer;
  trigger OnRun();
  begin
   CheckSize();
  end;
  procedure CheckSize(var Cust: Record Customer)
  begin
    if not Cust.HasShoeSize() then
      Cust.ShoeSize := 42;
  end;
}
```

Joonis 1. Codeunit näidis

Selleks, et *Codeunit* oleks testipõhine, siis omaduseks on vaja panna *SubType = Test.*

```
Codeunit 60000 MyFirstTestCodeunit
{
  SubType = Test;
}
```

Joonis 2. SubType Codeunit'i jaoks näidis

Testipõhine *Codeunit* võiks enda sees omada 3 erinevat tüüpi meetodeid:

1. *Test* meetod - testivad äriloogikat rakenduses, kus iga meetod hõlmab tehingut. Nende meetoditele on vaja deklareerida atribuuti *[Test]* [14].

```
[Test]
procedure MyFirstTestFunction()
begin
end;
```

Joonis 3. Test meetodi näidis

2. *Handle*r meetod - meetodeid kasutatakse testide automatiseerimiseks kasutajaga suhtlemise kaudu. See võib testida sõnumikasti sisu, käsitleda kinnitusdialoogi abiga jne. Nende meetoditele on vaja deklareerida atribuuti *[Handler]* [14].

```
[Handler]
procedure MyFirstHandlerFunction()
begin
end;
```

Joonis 4. Handler meetodi näidis

3. *Norma*l meetod - on tavalised meetodid testkoodi struktureerimiseks ja samade disainipõhimõtete kasutamiseks, mida kasutatakse juba rakenduse teistes koodüksustes. Nende meetoditele ei ole vaja deklareerida atribuuti [14].

```
procedure MyFirstFunction()
begin
end;
```

Joonis 5. Normal meetodi näidis

3.2.2 Testide jooksja ja testide eraldamine

Testide jooksjat on vaja selle jaoks, et käivitada teste, mis on salvestatud mitmes *Codeunit'is,* ning kontrollida nende täitmist ja koguda, kinnitada tulemusi. Testide araldamist on vaja selle jaoks, et käivitada teste isoleeritult, nii et iga testi või testkodediühiku käivitamine toimub sama andmeside andmete abil, keerates pärast testi või testi lõpetamist kõik kirjutamise tehinguid tagasi. See muudab iga testi käitamise korratavaks ja deterministlikuks [8].

Testide jooksja seadistamiseks on vaja kirjutada *SubType = TestRunner* ja testide eraldamiseks saab kirjutada 3 erinevat tüüpi [14]:

1. *Disabled* - see on vaikeväärtus ja ei võta andmebaasi muudatusi tagasi. Testid pole üksteisest isoleeritud.

2. *Function* - pärast iga testi Codenit'is saage andmebaasi kõik muudatused tagasi.

3. *Codeunit* - pärast iga testi käivitamist saage kõik andmebaasi muudatused tagasi.

```
codeunit 50000 MyTestRunnerCodeunit
{
   SubType = TestRunner;
   TestIsolation = Codeunit;
}
```

Joonis 6. Testide jooksja ja testide eraldamise näidis

3.2.3 Testilehed

Testilehed jäljendavad tegelikke lehti, kuid ei esinda kliendi arvutis ühtegi kasutajaliidest. Testilehed võimaldavad testida lehel olevat koodi, kasutades AL-i kasutaja interaktsiooni lehe simuleerimiseks [14].

```
codeunit 60003 MyFourthTestCodeunit
{
  Subtype = Test;
  [Test]
  procedure MyFirstTestPageTestFunction()
  var
    PaymentTerms: TestPage "Payment Terms";
  begin
    PaymentTerms.OpenView();
    PaymentTerms.Last();
    PaymentTerms.Code.AssertEquals('LUC');
    PaymentTerms.Close();
  end;
}
```

Joonis 7. Testilehe näidis

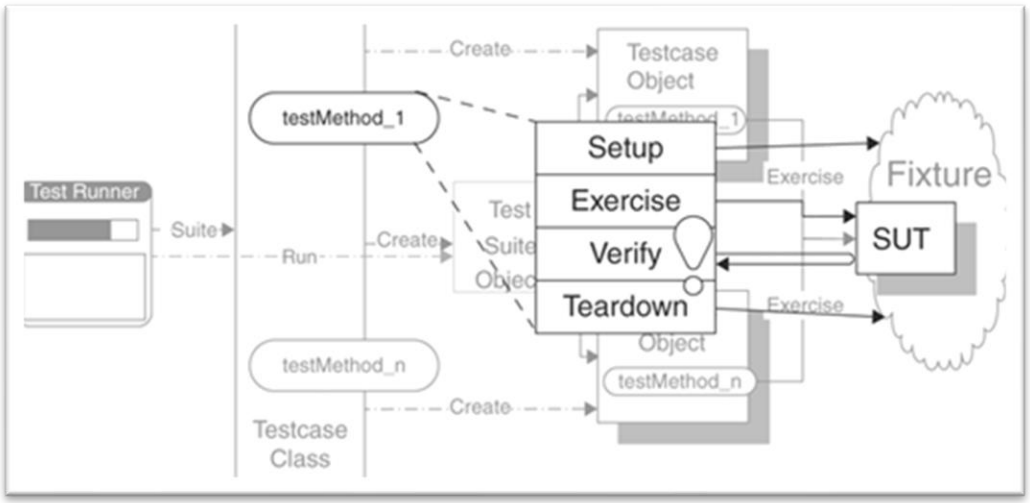
3.3 Testide plaan

Enne plaani kirjutamist, autor hakkas uurima Connector App'i koodi. Analüüsi põhjal, autor sai aru sellest, et on vaja kirjutada umbes 11 teste. 11 testi kirjutamist on plaanis realiseerida aprillikuu jooksul. Kõik testid on automaatsed ning nende eest vastutab autor. Testide kirjutamisel võib autor ka kasutada abimeetodeid realiseerimiseks.

3.4 Testide põhimustreid

3.4.1 Neljafaasiline testimine

Neljafaasiline testimine näeb välja järgmiselt [15]:



Joonis 8. Neljafaasilise testimise joonis.

Põhiosad [15]:

1. Seadistamine *(Setup)* – seadistatakse testseadme, mis on vajalik selleks, et *SUT* näitaks oodatud käitumist ja kõike, mida peab tegeliku tulemuse jälgimiseks paika panema. Ehk määrab *SUT*-i oleku enne testimist, mis on testi oluline sisend.

2. Harjutus *(Exercise)* - suhtleme *SUT*-ga. Ehk see on koht, kus testitavat tarkvara tegelikult käivitatakse. Testi lugedes peaks nägema, millist tarkvara käivitatakse.

3. Kinnitamine *(Verify)* - tehakse kõik, mis on vajalik, et teha kindlaks, kas oodatud tulemus on saavutatud. Ehk see on koht, kus täpsustatakse oodatavat tulemust.

4. Lammutamine *(Teardown)* – lõhutakse katseseadme, et viia „maailm“ tagasi sellesse olekusse, milles selle on leidnud. Ehk on seotud „majapidamisega“.

Seda neljafaasilise põhimustrit kasutas Microsoft C/SIDE testimise algusaastatel. Nagu järgmise testfunktsiooni näidel, mis on võetud *Codeunit*'ist SCM Inventory Misc. III (137295):

```
[Test]
[HandlerFunctions('SalesInvoiceStatisticsPageHandler,
```

```
    CreditMemoConfirmHandlerYes')]
[Scope('OnPrem')]
procedure PstdSalesInvStatisticsWithSalesPrice()
var
  SalesLine: Record "Sales Line";
  PostedSalesInvoice: TestPage "Posted Sales Invoice";
  DocumentNo: Code[20];
begin
  // Verify Amount on Posted Sales Invoice Statistics after
  // posting Sales Order.
  // Setup: Create Sales Order, define Sales Price on
  // Customer,.
  Initialize;
  CreateSalesOrderWithSalesPriceOnCustomer(SalesLine, WorkDate);
  LibraryVariableStorage.Enqueue(SalesLine."Line Amount");
      // Enqueue for SalesInvoiceStatisticsPageHandler.
  // Exercise: Post Sales Order.
  DocumentNo := PostSalesDocument(SalesLine, true);
      // TRUE for Invoice.
  // verify: Verify Amount on Posted Sales Invoice Statistics.
  // Verification done in SalesInvoiceStatisticsPageHandler
  PostedSalesInvoice.OpenView;
  PostedSalesInvoice.Filter.SetFilter("No.", DocumentNo);
  PostedSalesInvoice.Statistics.Invoke;
end;
```

Joonis 9. Neljafaasilise testimustri näidis.

3.4.2 Vastuvõtu testipõhine arendus

Vastuvõtu testipõhine arendus on teatud ka nagu ATDD (*Acceptance Test-Driven Development*) ja Microsoft ise kasutab seda põhimustrit enda testide kirjutamises. ATDD käsitleb testi seadistamise, harjutuse ja kontrollimise etappe. Kuid see tutvustab terviklikumat struktuuri, mis on ka kasutajale lähemal, kuna teste kirjeldatakse kasutaja vaatenurgast. Muster on määratletud järgmiste nn siltidega [8]:

- *Feature* - määrab, millist funktsiooni test või testjuhtumite kogum testib. Märgitud lila värviga.
- *Scenario* - määrab ühe testi jaoks testitava stsenaariumi. Märgitud kulla värviga.
- *Given* - määrab, milliseid andmeseadeid on vaja; testjuhtumil võib olla mitu *Given* silti, kui andmete seadistamine on keerulisem. Märgitud punase värviga.

- *When* - määratleb testitava toimingu; igal testjuhtumil peaks olema ainult üks *When* silt. Märgitud rohelise värviga.

- *Then* - määratleb toimingu tulemuse või täpsemalt tulemuse kontrollimise; kui kehtib mitu tulemust, on vaja mitut *Then* silti. Märgitud sinise värviga.

Järgmine testi näide, mis on võetud *Codeunit*'ist ERM Bank Reconciliation (134141), kuvab ATDD kujundusmustripõhise testi.

```
[Test]
[Scope('OnPrem')]
procedure VerifyDimSetIDOfCustLedgerEntryAfterPosting
  BankAccReconLine()
var
  BankAccReconciliation: Record "Bank Acc. Reconciliation";
  BankAccReconciliationLine: Record "Bank Acc. Reconciliation Line";
  StatementAmount: Decimal;
  CustomerNo: Code[20];
  CustLedgerEntryNo: Integer;
  DimSetID: Integer;
begin
  // [FEATURE] [Customer]
  // [SCENARIO 169462] "Dimension set ID" of Cust. Ledger
  //                   Entry should be equal "Dimension Set ID" of
  //                   Bank Acc. Reconcilation Line after posting
  Initialize;
  // [GIVEN] Posted sales invoice for a customer
  CreateAndPostSalesInvoice(CustomerNo, CustLedgerEntryNo,
    StatementAmount);
  // [GIVEN] Default dimension for the customer
  CreateDefaultDimension(CustomerNo, DATABASE::Customer);
  // [GIVEN] Bank Acc. Reconcilation Line with
  //         "Dimension Set ID" = "X" and
  //         "Account No." = the customer
  CreateApplyBankAccReconcilationLine(
    BankAccReconciliation, BankAccReconciliationLine,
    BankAccReconciliationLine."Account Type"::Customer,
    CustomerNo, StatementAmount, LibraryERM.CreateBankAccountNo);
  DimSetID :=
    ApplyBankAccReconcilationLine(
      BankAccReconciliationLine,
      CustLedgerEntryNo,
      BankAccReconciliationLine."Account Type"::Customer, '');
  // [WHEN] Post Bank Acc. Reconcilation Line
  LibraryERM.PostBankAccReconciliation(BankAccReconciliation);
```

```
  // [THEN] "Cust. Ledger Entry"."Dimension Set ID" = "X"
  VerifyCustLedgerEntry(
    CustomerNo, BankAccReconciliation."Statement No.", DimSetID);
end;
```

Joonis 10. Vastuvõtu testipõhise arenduse testimustri näidis.

3.5 Testide andmete seadistamiste põhimustreid

1. Eelehitatud kinnitusteks - need on testi andmed, mis luuakse enne mis tahes testi käivitamist. Dynamics 365 Business Central'i kontekstis võib see olla ettevalmistatud andmebaas, näiteks CRONUS, Microsofti pakutav demo ettevõte [8].

2. Jagatud kinnitusvahend või laiska seadistus - see puudutab testide rühma jagatud andmete seadistamist. Dynamics 365 Business Central'i kontekstis puudutab see üldisi põhiandmeid, lisaandmeid ja seadistusandmeid, nagu kliendi- ja valuutaandmed ning ümardamise täpsust, mis kõik on vajalikud testide rühma käitamiseks [8].

3. Värske kinnitus või värske seadistus - see hõlmab andmeid, mis on eriti vajalikud ühe testi jaoks, nagu tühi koht, konkreetne müügihind või postitatav dokument [8].

3.6 Lahenduse visioon

Selles peatükis töö autor kirjeldab bakalaurusetööks loodud automaattestide nõuded, nende kirjutamiseks platvormi ja tehnoloogiate valik ning missugune on testide ülesehitus ja skoop.

3.6.1 Nõuded lahendusele

Nõuded Connector App'i jaoks automaattestide rakendus peab täitma on järgmised:

- Automaattestide koostatud plaanide jälgimine.
- Platvormi seadistamine automaattestide kirjutamiseks ning testimiseks.
- Automaattestid peavad vähendama käsitsi testimist.

3.6.2 Teostuse platvormi valik

Connector App'i automaatteste arendab töö autor Business Central'i SaaS versioonile, mida hakatakse käivitama lokaalselt konteineri põhjal Docker'i abiga. Automaattestide kirjutamise programmeerimise keeleks on AL. Arendusvahendina kasutatakse Visual Studio Code.

3.6.3 Tehnoloogiate valik

Autor hakkab kasutama Business Central'is loodud Microsoft'i poolt *Test ToolKit*, mis võimaldab kontrollida automaatseid teste ja neid käivitada, ning *Code Coverage*, mille abil saab jälgida koodi katvust testidega. Tulemused kogutakse ja kuvatakse testriista lehel. Lisaks, see on sisseehitatud Docker konteineris, mis autor hakkab kasutama oma töös [14].

3.6.4 Automaattestide ülesehitus

Automaatteste hakatakse kirjutama eraldi rakendusena. Kõik testid autor hakkab kirjutama *Codeunit'is* kasutades vajalikke atribuute. Igal *Codeunit'il* pannakse unikaalse ID ja nimi. Meetodidel kasutatakse nimed, mis on seotud funktsionaalsuse testimisega.

Automaattestide kirjutamiseks autor hakkab kasutama töö raames järgmiseid mustreid:

- Testi täitmine - mida lühem on automaatne test, seda rohkem seda kasutatakse.
- Andmete seadistamine - testjuhtumite kavandamisel on kohe selge, milliseid andmeid ja millises etapis vaja läheb; see kiirendab testide kodeerimist.

3.6.5 Lahenduse skoop

Funktsionaalsed nõuded:

- Automaattestid peavad tuvastama uue versiooniga vigu enne rakenduse publitseerimist.
- Business Central'is võiks näha kõik olemasolevaid teste, mis on kirjutatud Connector App'i jaoks.
- Teste saab käivitada Business Central'i kaudu.
- Teste saab käivitada Docker'i kaudu.
- Teste saab käivitada rakenduse kaudu, kuhu nad on kirjutatud.
- Teste saab käivitada PowerShell kaudu.

Tulevikuplaanideks, kui Connector App läheb AppSource'i rakenduste poodi, siis pannakse ka seotud rakendusega automaattestide rakendus. Ning Connector App'i omanikuga edaspidise koostöö.

4 Lahenduse realiseerimine

Selles peatükis on toodud automaattestide koodid ja nende kirjeldus.

4.1 Automaattestide kood ja kirjeldus

Igal automaattestil on olemas vähemalt üks negatiivne ja üks positiivne test.

4.1.1 API-ga seotud testid

ConnectorApp'il on olemas Codeunit "CPC Alldevice API Management" ja leht "Alldevice API seadistus", mis vastutavad API tegevuste eest. Lisaks kui Business Central'is on ebakorrektselt määratud API seadistsed, antakse kohe viga ja lõpetatakse tegevus.

Funktsioon *GetRequestJsonWithAuth* vastutab API kaudu autentimise eest. Positiivse test läbib, kui saadud Json pole tühi.

```
[Test]
0 references | Run Test | Debug Test
procedure TestGetRequestJsonWithAuthentication()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    JsonObjectText: Text;
begin
    APIManagement.GetRequestJsonWithAuth().WriteTo(JsonObjectText);

    Assert.IsTrue(JsonObjectText <> '', 'Is empty');
end;
```

Joonis 11. API autentimise positiivne test.

Testid on tehtud varuosade, kategooriate ja tootjate jaoks sarnaselt. Näidisena autor võtab varuosade sünkroniserimist.. Negatiivse testi puhul avatakse leht "Alldevice API seadistus", kus määratakse juhuslikult genereeritud tekst pikkusega 100 väljale "Alldevice API kasutajanimi". Pärast proovitakse käivitada funktsiooni "Sünkroniseeri varuosad", aga antakse viga, sest autentimine on valesti seadistatud.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontGetRequestJsonWithAuthenticationSpares()
var
    AlldeviceAPISetup: TestPage "CPC Alldevice API Setup";
    APIManagement: Codeunit "CPC Alldevice API Management";
begin
    AlldeviceAPISetup.OpenEdit();
    AlldeviceAPISetup."CPC Alldevice API Username".SetValue(Random(100));
    asserterror AlldeviceAPISetup.GetSpares.Invoke();

    Assert.IsTrue(GetLastErrorText() <> '', 'Error is raised');
end;
```

Joonis 12. API autentimise negatiivne test.

Funktsioon *GetJsonToken* vastutab selle eest, et saada Json *token*'i võti järgi. Positiivne test läbib, kui saadud Json *token* pole tühi ja ise määratud funktsioonis Json võti, praegusel näitel *'auth'*, on õige, ehk eksisteerib.

```
[Test]
0 references | Run Test | Debug Test
procedure TestGetJsonToken()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    JsonTokenText: Text;
begin
    APIManagement.GetJsonToken(APIManagement.GetRequestJsonWithAuth(), 'auth').WriteTo(JsonTokenText);

    Assert.IsTrue(JsonTokenText <> '', 'Is empty');
end;
```

Joonis 13. Json token saamise positiivne test.

Negatiivse testi puhul genereeritakse juhuslikult tekst pikkusega 4 Json võti jaoks ja pannakse see funktsiooni. Test läbib, kui funktsioon annab viga valesti määratud Json võti tõttu, sest ei saa leida Json *token*'it selle järgi.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontGetJsonToken()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    Error001: Label 'Could not find a token with key %1', Locked = true;
    JsonTokenText: Text;
begin
    JsonTokenText := Random.AlphabeticText(4);
    asserterror APIManagement.GetJsonToken(APIManagement.GetRequestJsonWithAuth(), JsonTokenText);

    Assert.IsTrue(GetLastErrorText() = StrSubstNo(Error001, JsonTokenText) , 'Errors are not equal');
end;
```

Joonis 14. Json token saamise negatiivne test.

Sama põhimõttega on tehtud testid funktsioonile *SelectJsonPath*, mis vastutab selle eest, et saada Json *token* tema tee järgi.

Funktsioon *GetList* vastutab selle eest, et saada nimekirja vastavalt URL-i liide. Testid on tehtud varuosade, kategooriate ja tootjate jaoks sarnaselt. Näidisena autor võtab varuosade nimikirja saamist. Positiivse testi puhul esialgselt võetakse andmed tabelist „Alldevice API Seaditsus“. Teiseks, proovitakse saada nimikirja API kaudu. API vastust saadakse selle nimikirja põhjal, ja test läbib kui API vastus on *„success“*, ehk nimikiri on kättes.

```
[Test]
0 references | Run Test | Debug Test
procedure TestGetListSpares()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    AlldeviceAPISetup: Record "CPC Alldevice API Setup";
    ResponseText: Text;
    Response: HttpResponseMessage;
    Content: HttpContent;
begin
    AlldeviceAPISetup.Get();
    ResponseText := APIManagement.GetList(AlldeviceAPISetup."CPC Get Spares List Prefix");
    Response.Content.ReadAs(ResponseText);

    Assert.IsTrue(Response.IsSuccessStatusCode, 'Status code is failure');
end;
```

Joonis 15. Varuosade nimikirja saamise positiivne test.

Negatiivse testi puhul avatakse leht „Alldevice API Seadistus“, kus määratakse juhuslikult genereeritud tekst pikkusega 15 välja „Varuosade kogumi ressursi API URL-i liide“ jaoks. Proovitakse käivitada funktsiooni „Sünkroniseeri varuosad“, kuid antakse viga valesti määratud URL-i liide tõttu.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontGetListSpares()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    AlldeviceAPISetup: TestPage "CPC Alldevice API Setup";
    Prefix: Text;
    ResponseText: Text;
    Response: HttpResponseMessage;
    Content: HttpContent;
begin
    AlldeviceAPISetup.OpenEdit();
    AlldeviceAPISetup."CPC Get Spares List Prefix".SetValue(Random.AlphabeticText(15));
    asserterror AlldeviceAPISetup.GetSpares.Invoke();

    Assert.IsTrue(GetLastErrorText() <> '', 'Error raised');
end;
```

Joonis 16. Varuosade nimikirja saamise negatiivne test.

Funktsioon *PostRequest* vastutab andmete uuendamise/lisamise Alldevice'is API kaudu eest. Testid on tehtud varuosade, kategooriate ja tootjate jaoks sarnaselt. Näidisena autor võtab varuosade andmete uuendamist/lisamist Alddevice'is API kaudu. Esialgselt saadakse andmed tabelist „Alldevice API Seadistus". Teiseks, lisatakse uus kaup abifunktsiooniga *CreateItemSpare*. Järgmiseks luuakse Json objekt autentimise ja uue kauba andmetega ja pannakse funktsiooni *PostRequest*. Test läbib, kui API vastus on „success", ehk andmed on uuendatud/lisatud.

```
[Test]
0 references | Run Test | Debug Test
procedure TestPostRequestSpares()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    AlldeviceAPISetup: Record "CPC Alldevice API Setup";
    JsonObject: JsonObject;
    JsonObjectText: Text;
    Item: Record Item;
    Response: HttpResponseMessage;
    Content: HttpContent;
begin
    AlldeviceAPISetup.Get();

    Item := CreateItemSpare();

    JsonObject := APIManagement.GetRequestJsonWithAuth();
    JsonObject.Add('name', Item.Description);
    JsonObject.Add('code', Item."No.");
    JsonObject.Add('product_id', Item."CPC Alldevice Product Id");

    APIManagement.PostRequest(AlldeviceAPISetup."CPC Update or Add Spare Prefix", JsonObject).WriteTo(JsonObjectText);
    Response.Content.ReadAs(JsonObjectText);

    Assert.IsTrue(Response.IsSuccessStatusCode, 'Status code is failure');
end;
```

Joonis 17. Andmete uuendamise/lisamise API kaudu positiivne test.

Negatiivse testi puhul on protsessid jäänud samaks, kuid URL-i liide asemel genereeritakse juhuslikult tekst pikkusega 15, mis pannakse funktsiooni *PostRequest*. Valesti määratud URL-i liide tõttu antakse viga ja lõpetatakse tegevus.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontPostRequest()
var
    APIManagement: Codeunit "CPC Alldevice API Management";
    AlldeviceAPISetup: Record "CPC Alldevice API Setup";
    JsonObject: JsonObject;
    JsonObjectText: Text;
    Prefix: Text;
begin
    AlldeviceAPISetup.Get();
    Prefix := Random.AlphabeticText(15);
    asserterror APIManagement.PostRequest(Prefix, JsonObject).WriteTo(JsonObjectText);

    Assert.IsTrue(GetLastErrorText() <> '', 'Error doesn''t raise');
end;
```

Joonis 18. Andmete uuendamise/lisamise API kaudu negatiivne test.

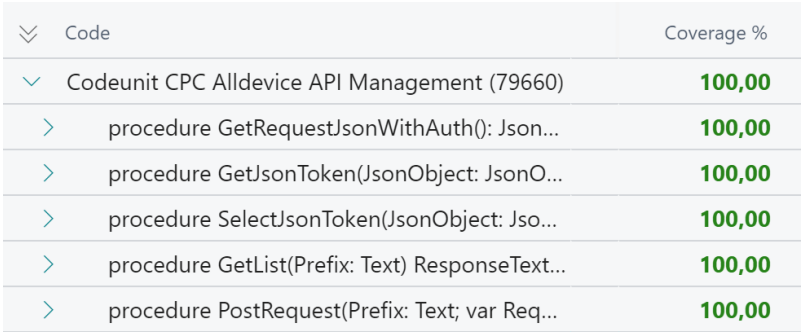
Lehe „Alldevice API Seaditsus“ avamisel, kui ei saada andmeid tabelist „Alldevice API Seadistus“, siis luuakse uued andmed. Selle testimiseks on kirjutatud test, kus esialgselt kustutakse maha andmed tabelist ja võrreldakse oma vahel kaks numbrit: enne lehe avamist ja pärast. Test läbib, kui enne avamist ridade arv on väiksem kui pärast avamist ridade arv.

```
[Test]
0 references | Run Test | Debug Test
procedure TestAlldeviceAPISetupOpenPage()
var
    AlldeviceAPISetupPage: TestPage "CPC Alldevice API Setup";
    AlldeviceAPISetupTable: Record "CPC Alldevice API Setup";
    RowCountBeforOpenPage: Integer;
    RowCountAfterOpenPage: Integer;
begin
    AlldeviceAPISetupTable.DeleteAll();
    RowCountBeforOpenPage := AlldeviceAPISetupTable.Count;
    AlldeviceAPISetupPage.OpenEdit();
    RowCountAfterOpenPage := AlldeviceAPISetupTable.Count;

    Assert.IsTrue(RowCountBeforOpenPage < RowCountAfterOpenPage, 'Row count doesn''t change');
end;
```

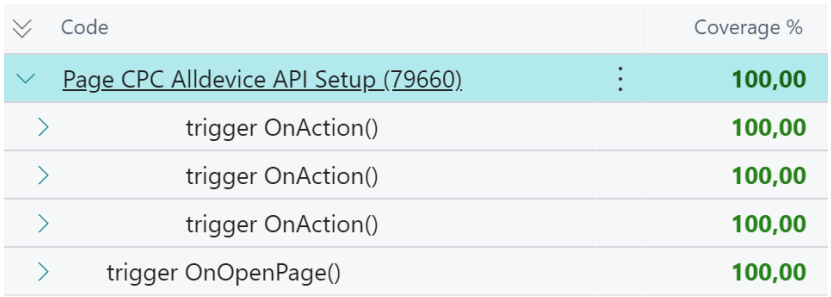
Joonis 19. „Alldevice API Setup“ lehe avamise test.

Kokkuvõteks koodi katvus on 100%.



Code	Coverage %
Codeunit CPC Alldevice API Management (79660)	100,00
procedure GetRequestJsonWithAuth(): Json...	100,00
procedure GetJsonToken(JsonObject: JsonO...	100,00
procedure SelectJsonToken(JsonObject: Jso...	100,00
procedure GetList(Prefix: Text) ResponseText...	100,00
procedure PostRequest(Prefix: Text; var Req...	100,00

Joonis 20. Codeunit “CPC Alldevice API Management” koodi katvus.



Code	Coverage %
Page CPC Alldevice API Setup (79660)	100,00
trigger OnAction()	100,00
trigger OnAction()	100,00
trigger OnAction()	100,00
trigger OnOpenPage()	100,00

Joonis 21. Lehe “Alldevice API Seadistus” koodi katvus.

4.1.2 API logidega seotud testid

Iga sünkroniseerimisega lisatakse uus rida tabelisse „CPC Alldevice API Logs“, kus saab vaadata API kohta info. Selle funktsionaalsuse testimiseks võrdleb omavahel autor kahte numbrid: enne sünkroniseerimist ridade arvu tabelis ja pärast. Testid on tehtud varuosade, kategooriate ja tootjate jaoks sarnaselt. Näidisena autor võtab varuosade sünkroniseerimist ja nende tabeli lisamist. Test läbib, kui pärast sünkroniseerimist ridade arv on suurem, kui enne sünkroniseerimist ridade arv.

```
[Test]
[HandlerFunctions('ErrorMessagesHandler')]
0 references | Run Test | Debug Test
procedure TestCreateAPILogSpares()
var
    NumberBeforeSync: Integer;
    NumberAfterSync: Integer;
    ApiLogs: Record "CPC Alldevice API Logs";
    ApiSetup: TestPage "CPC Alldevice API Setup";
begin
    NumberBeforeSync := ApiLogs.Count;
    ApiSetup.OpenEdit();
    ApiSetup.GetSpares.Invoke();
    NumberAfterSync := ApiLogs.Count;

    Assert.IsTrue(NumberAfterSync > NumberBeforeSync, 'No rows inserted');
end;
```

Joonis 22. Ridade lisamise lehele „Alldevice API Logid“ positiivne test.

Negatiivse testi puhul andmed ei lisata, kui lehel “Alldevice API Seadistus” on andmed määratud valesti, ehk peab andma seotud API-ga viga.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontCreateAPILog()
var
    NumberBeforeSync: Integer;
    NumberAfterSync: Integer;
    ApiLogs: Record "CPC Alldevice API Logs";
    ApiSetup: TestPage "CPC Alldevice API Setup";
begin
    NumberBeforeSync := ApiLogs.Count;
    ApiSetup.OpenEdit();
    ApiSetup."CPC Alldevice API Username".SetValue(Random.AlphabeticText(100));
    asserterror ApiSetup.GetSpares.Invoke();
    NumberAfterSync := ApiLogs.Count;

    Assert.IsTrue(NumberAfterSync = NumberBeforeSync, 'Rows inserted');
end;
```

Joonis 23. Ridade lisamise lehele „Alldevice API Logid“ negatiivne test.

Lehel „Alldevice API Logid“ on olemas kaks nuppu, millega saab vaadata saadetud ja vastuvõetud JSON andmetüüpe. Testid on tehtud varuosade, kategooriate ja tootjate jaoks sarnaselt nii vastuvõetud, kui ka saadetud JSON andmetüüpe vaatamiseks. Näidisena autor võtab varuosade vastuvõetud JSON andmetüüpe vaatamist.

Esialgselt avatakse leht „Alldevice API Seadistus“, kus käivitatakse varuosade sünkroniseerimine. Pannakse kinni ja avatakse leht „Alldevice API Logid“, kus peavad olema sünkroniseerimise kohta andmed. Leitakse viimane ja vaadetakse, kas sellel real on olemas väärtus väljal „CPC Response JSON Data“. Kui jah, siis käivitatakse protseduur, mille abil saab vaadata vastuvõetud JSON andmetüüpe ja omistatakse väärtusele *IsRun* TÕENE. Test läbib, kui muutujal *IsRun* on TÕENE väärtus.

```
[Test]
[HandlerFunctions('ErrorMessagesHandler')]
0 references | Run Test | Debug Test
procedure TestOpenResponseJsonSpares()
var
    ApiLogs: Record "CPC Alldevice API Logs";
    ApiSetup: TestPage "CPC Alldevice API Setup";
    IsRun: Boolean;
begin
    IsRun := false;

    ApiSetup.OpenEdit();
    ApiSetup.GetSpares.Invoke();
    ApiSetup.Close();

    ApiLogs.FindLast();
    ApiLogs.CalcFields("CPC Response JSON Data");
    if ApiLogs."CPC Response JSON Data".HasValue then begin
        ApiLogs.OpenResponseJson();
        IsRun := true;
    end;

    Assert.IsTrue(IsRun, 'Open response json doesn''t run');
end;
```

Joonis 23. Vastuvõetud JSON andmetüüpe vaatamise positiive test.

Lisaks on tehtud eraldi test, mis kontrollib, kas see vastuvõetud JSON ei ole tühi, ehk tema tekst ei ole – „{}“. Nagu eelmisel testil, protseduur on jäänud samaks, lisaks et kontrollida teksti pikkust, on vaja esialgselt BLOB tüüpi konverteerida TEXT tüübiks. Test läbib, kui teksti pikkus on rohkem kui 2.

```
[Test]
[HandlerFunctions('ErrorMessagesHandler')]
0 references | Run Test | Debug Test
procedure TestOpenResponseJsonSparesNotBlank()
var
    ApiLogs: Record "CPC Alldevice API Logs";
    ApiSetup: TestPage "CPC Alldevice API Setup";
    InStream: InStream;
    TextToTest: Text;
begin
    ApiSetup.OpenEdit();
    ApiSetup.GetSpares.Invoke();
    ApiSetup.Close();

    ApiLogs.FindLast();
    ApiLogs.CalcFields("CPC Response JSON Data");
    ApiLogs."CPC Response JSON Data".CreateInStream(InStream);
    InStream.ReadText(TextToTest);

    Assert.IsTrue(StrLen(TextToTest) > 2, 'Response json text is empty');
end;
```

Joonis 24. Vastuvõetud JSON tühja kontrollimise test.

Negatiivse teksti puhul vastuvõetud JSON andmetüüpe vaatlemiseks pole võimalik, kui ei ole ühtigi rida tabelis „Alldevice API Logid“. Esialgselt kustutatakse kõik read tabelist ja proovitakse käivitada funktsiooni, millega saab vaadata vastuvõetud JSON andmetüüpe. Test läbib, kui antud viimane viga ei ole tühi.

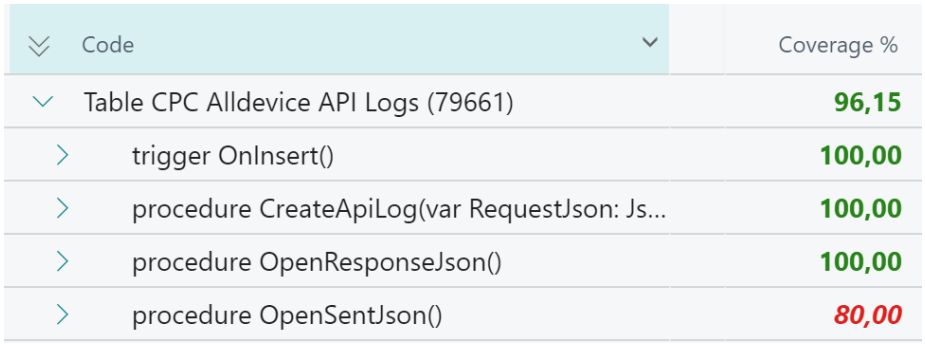
```
[Test]
0 references | Run Test | Debug Test
procedure TestDontOpenResponseJson()
var
    ApiLogsPage: TestPage "CPC Alldevice API Logs";
    ApiLogsRec: Record "CPC Alldevice API Logs";
    ApiSetup: TestPage "CPC Alldevice API Setup";
begin
    ApiLogsRec.DeleteAll();
    ApiLogsPage.OpenEdit();
    ApiLogsPage.GoToRecord(ApiLogsRec);
    asserterror ApiLogsPage.ViewResponseJson.Invoke();

    Assert.IsTrue(GetLastErrorText() <> '', 'Response json exsists');
end;
```

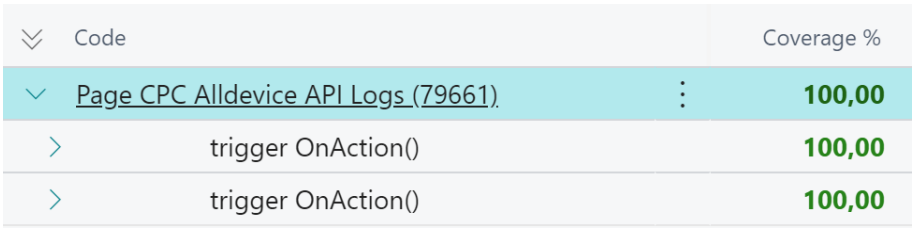
Joonis 25. Vastuvõetud JSON andmetüüpe vaatamise negatiivne test.

Kokkuvõteks katvus on 98,1%.

- 80% funktsioonil *OpenSentJson*, sest autor märkas Rainise Mäemees koodis, et ta kasutas saadetud JSON andmetüüpi vaatamise asemel, vastuvõetud JSON andmetüüpi vaatamist. Ehk kaks korda tulevad sarnased andmetüübid. Ja seetõttu ei läbinud saadetud JSON andmetüüpi vaatamise negatiivne test



Joonis 26. „Alldevice API Logid“ tabeli koodi katvus.



Joonis 27. „Alldevice API Logid“ lehe koodi katvus.

4.1.3 Sünkroniseerimistega seotud testid

Rainis Mäemees on lisatud sünkroniseerimistega seotud testid oma rakenduses, kus ta testis, kas loodud uus varuosa/kategooria/tootja Business Central’is sünkroniseerib Alldevice’i. Töö autor siis testis, kas Alldevice’ist sünkroniseeritud varuosad, kategooriad, või tootjad lisatakse Business Central’i.

Testid on tehtud varuosade, kategooriate ja tootjate jaoks sarnaselt. Näidisena autor võtab varuosade sünkroniseerimist Business Central’i.

Positiivse testi puhul võrreldakse oma vahel kaks numbrit: enne kauba tabeli ridade arvu sünkroniseerimist ja pärast. Test läbib kui pärast sünkroniseerimist ridade arv tabelis on rohkem, kui enne.

```
[Test]
[HandlerFunctions('ErrorMessagesHandler')]
0 references | Run Test | Debug Test
procedure TestAddSparesFromAlldevice()
var
    SyncSpares: Codeunit "CPC Alldevice Spares";
    Item: Record Item;
    NumberBeforeFunc: Integer;
    NumberAfterFunc: Integer;
begin
    NumberBeforeFunc := Item.Count;
    SyncSpares.Run();
    NumberAfterFunc := Item.Count;

    Assert.IsTrue(NumberAfterFunc > NumberBeforeFunc, 'Spares not inserted');
end;
```

Joonis 28. Varuosade sünkroniseerimise Business Central'i positiivne test.

Negatiivse testi puhul lehel „Alldevice API Seadistus" genereeritakse juhsulikult tekst pikkusega 15 väljale „Varuosade kogumi ressursi API URL-i liide". Pärast proovitakse sünkroniseerida varuosasid. URL-i liide valesti määratud tõttu sünkroniseerimine ebaõnnestub, ehk siis read ei lisanud Business Central'i tabelisse. Test läbib, kui ridade arv enne ja pärast sünkroniseerimist on võrdsed.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontAddSparesFromAlldevice()
var
    AlldeviceAPISetup: TestPage "CPC Alldevice API Setup";
    Item: Record Item;
    NumberBeforeFunc: Integer;
    NumberAfterFunc: Integer;
begin
    NumberBeforeFunc := Item.Count;
    AlldeviceAPISetup.OpenEdit();
    AlldeviceAPISetup."CPC Get Spares List Prefix".SetValue(Random.AlphabeticText(15));
    asserterror AlldeviceAPISetup.GetSpares.Invoke();
    NumberAfterFunc := Item.Count;

    Assert.IsTrue(NumberAfterFunc = NumberBeforeFunc, 'Spares inserted');
end;
```

Joonis 29. Varuosade sünkroniseerimise Business Central'i negatiivne test.

4.1.4 Hooldustöödega seotud testid

Hooldustööde vaatamiseks Business Central'is on Rainis Mäemees loonud lehe „Alldevice hooldustööd", kuskohast on võimalik vaadata Alldevice'is loodud hooldustöid ja luua hooldustöödes kasutatud varuosadest kaubažurnaali ridasid [9]. Kui kasutaja avab lehe, siis lisatakse read, ja selleks kasutas Rainis Mäemees *temporary* tabelit, mis tähendab, et hooldustööde andmete pärimisel Alldevice'ist ei kirjutata kirjeid andmetabelisse, vaid hoitakse mälus, nii kaua, kuni leht on aktiivne [9].

Selle funktsionaalsuse testimiseks töö autor on loonud teste, mis kontrollib enne lisamist ridade arv tabelis ja pärast. Test läbib, kui pärast lehe avamist ridade arv on suurem, kui enne.

```
[Test]
0 references | Run Test | Debug Test
procedure TestGetTasksRecord()
var
    AlldeviceTasks: Codeunit "CPC Alldevice Tasks";
    AlldeviceTasksTabel: Record "CPC Alldevice Service Tasks" temporary;
    NumberRowsBeforeOpenPage: Integer;
    NumberRowsAfterOpenPage: Integer;
begin
    NumberRowsBeforeOpenPage := AlldeviceTasksTabel.Count;
    AlldeviceTasks.GetTasksRecord(AlldeviceTasksTabel);
    NumberRowsAfterOpenPage := AlldeviceTasksTabel.Count;

    Assert.IsTrue(NumberRowsAfterOpenPage > NumberRowsBeforeOpenPage, 'Rows not inserted');
end;
```

Joonis 30. Hooldustööde lisamise tabelisse positiivne test.

Negatiivse testi puhul esialgselt lehel „Alldevice API Seadistus" lisatakse väljale „Hooldustööde kogumi ressursi API URL-i liide" juhuslikult genereeritud tekst pikkusega 15. Pärast proovitakse saada hooldustööde andmeid, kuid antakse viga, et URL'i liide ei ole korrektne, ehk andmeid ei saa pärida. See tähendab, et andmed ei pea ilmuma lehel, ning ridade arv enne ja pärast käivitamist peavad olema võrdsed.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontGetTasksRecord()
var
    AlldeviceTasks: Codeunit "CPC Alldevice Tasks";
    AlldeviceAPISetupPage: TestPage "CPC Alldevice API Setup";
    AlldeviceTasksTabel: Record "CPC Alldevice Service Tasks" temporary;
    AlldeviceTasksPage: TestPage "CPC Alldevice Tasks List";
    NumberRowsBeforeOpenPage: Integer;
    NumberRowsAfterOpenPage: Integer;
begin
    AlldeviceAPISetupPage.OpenEdit();
    AlldeviceAPISetupPage."CPC Get Service Task List".SetValue(Random.AlphabeticText(15));
    NumberRowsBeforeOpenPage := AlldeviceTasksTabel.Count;
    asserterror AlldeviceTasks.GetTasksRecord(AlldeviceTasksTabel);
    NumberRowsAfterOpenPage := AlldeviceTasksTabel.Count;

    Assert.IsTrue(NumberRowsBeforeOpenPage = NumberRowsAfterOpenPage, 'Rows inserted');
end;
```

Joonis 31. Hooldustööde lisamise tabelisse negatiivne test.

Hooldustööde lehel, saab luua kaubažurnaali ridu varuosade laost mahakandmiseks. Selleks on vaja vajutada nupule „Moodusta kaubažurnaali read“. Nupule vajutamisel esiteks päritakse Alldevice’ist kasutatud varuosasid funktsiooniga *GetUsedSparesList*, mille abil lisatakse need kasutatud varuosad tabelisse „CPC Alld. Service Tasks Lines“. Pärast need read lähevad juba kaubažurnaali.

Positiivse testi puhul ridade arv enne kasutatud varuosade saamist peab olema vähem, kui pärast tegevuse käivitamist. Esialgselt saadakse hooldustööde andmeid Alldevice’ist. Kuna suure andme tõttu, autor võtab juhuslikult genereeritud sammuga kuni 10 andmed hooldustööde tabelist ja kontrollitakse, kas sellise valitud rea „Hooldustöö Id“ järgi on olemas kasutatud varuosad. Kui need on olemas, siis kasutatud varuosad peavad lisanduma tabelisse „CPC Alld. Service Tasks Lines“, mis on vaja kaubažurnaali ridade genereerimiseks.

```
[Test]
0 references | Run Test | Debug Test
procedure TestGetUsedSparesList()
var
    AlldeviceTasks: Codeunit "CPC Alldevice Tasks";
    AlldeviceTasksTabel: Record "CPC Alldevice Service Tasks" temporary;
    AlldeviceTasksLineTabel: Record "CPC Alld. Service Task Lines" temporary;
    NumberRowsBeforeGetUsedSparesList: Integer;
    NumberRowsAfterGetUsedSparesList: Integer;
begin
    NumberRowsBeforeGetUsedSparesList := AlldeviceTasksLineTabel.Count;
    AlldeviceTasks.GetTasksRecord(AlldeviceTasksTabel);
    if AlldeviceTasksTabel.FindSet() then
        repeat
            AlldeviceTasks.GetUsedSparesList(AlldeviceTasksLineTabel, AlldeviceTasksTabel."CPC Task Id");
            if AlldeviceTasksLineTabel.Count <> 0 then
                NumberRowsAfterGetUsedSparesList := AlldeviceTasksLineTabel.Count;
        until (AlldeviceTasksTabel.Next(Random.DecimalInRange(10, 0)) = 0) or (AlldeviceTasksLineTabel.Count <> 0);

    Assert.IsTrue(NumberRowsAfterGetUsedSparesList > NumberRowsBeforeGetUsedSparesList, 'Rows not inserted');
end;
```

Joonis 32. Kasutatud varuosade andmete pärimise positiivne test.

Lisaks sellele testile on tehtud kontroll, kas lisatud andmed tabeli „CPC Alld Service Tasks Lines“ ei ole tühjad. Näidisena autor võtab kontrolli väljale „CPC Id“.

```
[Test]
0 references | Run Test | Debug Test
procedure TestGetUsedSparesListCheckFieldCPCId()
var
    AlldeviceTasks: Codeunit "CPC Alldevice Tasks";
    AlldeviceTasksTabel: Record "CPC Alldevice Service Tasks" temporary;
    AlldeviceTasksLineTabel: Record "CPC Alld. Service Task Lines" temporary;
    NoEmpty: Boolean;
begin
    AlldeviceTasksLineTabel.DeleteAll();
    AlldeviceTasks.GetTasksRecord(AlldeviceTasksTabel);
    if AlldeviceTasksTabel.FindSet() then
        repeat
            AlldeviceTasks.GetUsedSparesList(AlldeviceTasksLineTabel, AlldeviceTasksTabel."CPC Task Id");
            if AlldeviceTasksLineTabel.Count <> 0 then begin
                AlldeviceTasksTabel.FindFirst();
                NoEmpty := AlldeviceTasksLineTabel."CPC Id" <> 0;
            end;
        until (AlldeviceTasksTabel.Next(Random.DecimalInRange(10, 0)) = 0) or (AlldeviceTasksLineTabel.Count <> 0);

    Assert.IsTrue(NoEmpty, 'Field "CPC Id" Is Empty');
end;
```

Joonis 33. „CPC Alld. Service Tasks Lines“ tühjade väljade test.

Negatiivse testi puhul esialgselt vavatakse leht „Alldevice API Seadistus“. Väljale „Hooldustöödes kasutatud varuosade kogumi ressurssi API URL-i liide“ genereeritakse juhuslikult tekst pikkusega 15. Pärast seda proovitakse saada funktsiooniga *GetUsedSparesList* kasutatud varuosad. Vale URL-i liide tõttu antakse viga ja ei lisata read, seetõttu enne funktsiooni käivitamist ridade arv tabelis „CPC Alld. Service Tasks Line“ peavad olema võrdsed pärast funktsiooni käivitamist ridade arvuga.

```
[Test]
0 references | Run Test | Debug Test
procedure TestDontGetUsedSparesList()
var
    AlldeviceTasks: Codeunit "CPC Alldevice Tasks";
    AlldeviceAPISetupPage: TestPage "CPC Alldevice API Setup";
    AlldeviceTasksLinesTabel: Record "CPC Alld. Service Task Lines";
    NumberRowsBeforeFunc: Integer;
    NumberRowsAfterFunc: Integer;
begin
    AlldeviceAPISetupPage.OpenEdit();
    AlldeviceAPISetupPage."CPC Get Used Spares List Pref.".SetValue(Random.AlphabeticText(15));
    NumberRowsBeforeFunc := AlldeviceTasksLinesTabel.Count;
    asserterror AlldeviceTasks.GetUsedSparesList(AlldeviceTasksLinesTabel, Random.DecimalInRange(10, 0));
    NumberRowsAfterFunc := AlldeviceTasksLinesTabel.Count;

    Assert.IsTrue(NumberRowsAfterFunc = NumberRowsBeforeFunc, 'Rows inserted');
end;
```

Joonis 34. Kasutatud varuosade andmete pärimise negatiivne test.

Kokkuvõteks koodi katvus on 95,74%.

- Funktsioonil *GetTaskRecord* katvus on 92%, sest kohal, kus omistatakse tühja väärtust seadme nimele kood ei pääse, sest Alldevice'is seadme nimi määramine hooldustöö moodustamise ajal on kohustuslik ja see ei saa olla tühi. Ehk kood ei pääse selle koodireale mitte ühtegi kord.

Code	Coverage %
Codeunit CPC Alldevice Tasks (79669)	95,74
local procedure GetListSpares(Prefix: Text; T...	100,00
procedure GetTasksRecord(var AlldeviceServ...	92,00
procedure GetUsedSparesList(var ServiceTas...	100,00

Joonis 35. Codeunit "CPC Alldevice Tasks" koodi katvus.

5 Kokkuvõte

Bakalaurusetööks autor kirjutas automaatteste ConnectorApp'ile - mis sünkroniseerib andmeid Alldevice'i ja Business Central'i vahel, ning pärib andmeid Alldevice'st [9].

Kui tulevad uuendused kas Microsoft'i või ConnectorApp'i poolt võivad tekkida vead ning tänu automaatestidele neid võib tuvastada enne uuenduse publitseerimist kas Microsoft'i veebipoodi või kliendi serverile.

Töö teoreetilises osas loodi automaattestide kirjutamise jaoks skoop ja funktsionaalsed nõuded. Põhirõhk oli pandud 100% katvusega automaattestide kirjutamisele.

Enne automattestide kirjutamist uuriti erinevad dokumentatisoonid, raamatud. Lisaks sellele vaadeti Internetis avalikud näited, mis on seotud bakalaurusetöö teemaga.

Tulemuseks saadeti:

- 98,68% katvusega automaattestid.
- Saab tuvastada vigu pärast Microsoft'i või ConnectorApp'i uuendust.
- Käsitsi testimist vähenemine.

Loodud bakalaurusetöö järelduseks võib tuua, et lugedes tööd võib lugeja endale teha selgeks, kuidas luua automaatteste erinevate projektide jaoks. Lisaks saab lugeja tutvuda Business Central'iga, selle keskkonna valiku plusse ja miinuse automaattestide kirjutamiseks ning mis üldse on automaattestid ja milleks need on vaja.

Loodud automaattestid on tulevikus plaanis lisada Microsoft'i rakenduste veebipoodi (*AppSource*) ja teha koostööd ConnectorApp'i autoriga.

Kasutatud kirjandus

[1] Microsoft Learn, “Welcome to Dynamics 365 Business Central”, 28.02.2023. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/. [Kasutatud: 06.03.2023].

[2] Microsoft Learn, “Deployment of Dynamics 365 Business Central”, 04.08.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/deployment/deployment. [Kasutatud: 06.03.2023].

[3] Microsoft Learn , “Production and Sandbox Environments”, 04.03.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/administration/environment-types. [Kasutatud: 06.03.2023].

[4] Microsoft Learn, “Sandbox Environments for Dynamics 365 Business Central Development”, 14.06.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-sandbox-overview. [Kasutatud: 06.03.2023].

[5] Microsoft Learn, “Get started with the Container Sandbox Development Environment”, 15.02.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-get-started-container-sandbox. [Kasutatud: 06.03.2023].

[6] Microsoft Learn, “Testing the Application Overview”, 16.08.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-testing-application. [Kasutatud: 06.03.2023].

[7] Manfred Baumgartner, Thomas Steirer, Marc-Florian Wendland, Stefan Gwihs, Julian Hartner, Richard Seidl, “Test Automation Fundamentals”, September 2022. [Võrgumaterjal]. Available: https://learning.oreilly.com/library/view/test-automation-fundamentals/9781681989839/. [Kasutatud:07.03.2023].

[8] Luc van Vugt, “Automated Testing in Microsoft Dynamics 365 Business Central - Second Edition”, December 2021. [Võrgumaterjal]. Available: https://learning.oreilly.com/library/view/automated-testing-in/9781801816427/. [Kasutatud: 08.03.2023].

[9] Rainis Mäemees, “D365 Business Central’i liidestamine Alldevice’i hooldustarkvaraga”, 04.01.2023. [Võrgumaterjal]. Available: https://digikogu.taltech.ee/et/Item/2a05402b-7d1f-4a77-b455-afc7d171136f. [Kasutatud: 05.03.2022].

[10] Microsoft Learn, “Technical Validation”, 16.11.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-checklist-submission. [Kasutatud: 21.03.2023].

[11] Microsoft Learn, “Implement test automation techniques in Business Central”. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/training/modules/test-automation. [Kasutatud: 21.03.2023].

[12] Microsoft Learn, “Object Ranges in Business Central”, 14.06.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-object-ranges. [Kasutatud: 21.03.2023].

[13] Microsoft Learn, “Codeunit Object”, 15.02.2022. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-codeunit-object. [Kasutatud: 21.03.2023].

[14] Microsoft Learn, “ Introduction to test automation in Business Central”. [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/training/modules/introduction-test-automation/. [Kasutatud: 21.03.2023].

[15] Gerard Meszaros, “ xUnit Test Patterns: Refactoring Test Code”, May 2007. [Võrgumaterjal]. Available: https://learning.oreilly.com/library/view/xunit-test-patterns/9780131495050/. [Kasutatud: 11.04.2023].

[16] SoapUI, “5 Best Practices for Data Driven API Testing”. [Võrgumaterjal]. Available: https://www.soapui.org/learn/api/5-best-practices-for-data-driven-api-testing/. [Kasutatud: 11.05.2023].

Lisa 1 - Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Tatjana Levizi

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Microsoft Dynamics 365 Business Central automaattestid Connector App'i jaoks", mille juhendaja on Rivo Lemmik
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

02.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Automaattestide lähtekood

https://github.com/tatjanalevizi/ConnectorAppAutomatedTests