TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Henrik Hansson 178887IABB
Kaari Kaasik 164000IABB
Sken Selge 185067IABB

BUSINESS CENTRALI JA SCORO API LIIDESTAMINE NING APPSOURCE'I PUBLITSEERIMINE

Meeskondlik lõputöö

Juhendajad: Rivo Lemmik (Colmbus Eesti AS)
Taavi Sadam (Colmbus Eesti AS)
Viljam Puusep (Tallinna Tehnikaülikool)
Karl-Erik Karu (Tallinna Tehnikaülikool)

Tallinn 2021

Autorideklaratsioon

Kinnitame, et oleme 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: Henrik Hansson, Kaari Kaasik, Sken Selge
18.05.2021

Annotatsioon

Lõputöö eesmärgiks oli viia lõpule eelneva meeskonnaprojekti aine raames alustatud projekt selleks, et arendatud lahendus oleks valmis ettevõtte poolt AppSource'i publitseerimiseks. Lõputöö raames oli tarvis viia lõpule arendused, kood korrekteerida, vead parandada ning valmistada ette dokumendid Microsofti nõuete järgi AppSource'i, äritarkvara liidestuste pood, jaoks.

Lõputöö tulemusena valmis terviklik lahendus andmete sünkroniseerimiseks Scoro ja Business Centrali tarkvarade vahel. Business Centrali kasutaja saab üle kanda kontaktid, müügiarved, tooted ja ressursid Business Centralist Scorosse ning vastupidi. Lõputöö kirjutamise lõpetamise hetkel on rakendus publitseeritud AppSource keskkonda.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 55 leheküljel, 5 peatükki, 58 joonist, 3 tabelit.

Erilist tänu sooviksime avaldada Columbus Eesti AS ettevõtte poolsetele juhendajatele nende suurepärase toetuse eest.

Abstract

Business Central and Scoro connector interfacing and publishing to AppSource

Microsoft Dynamics 365 Business Central is an enterprise resource planning system used and distributed by Columbus Estonia AS. Scoro is an all-in-one business management software developed and distributed by Scoro Software OÜ. The representatives of Columbus Estonia AS saw an overlap of users of both systems and wished to improve the working conditions and efficiency of these users by developing a data synchronization connection between the two softwares. The aim of this connection is to bring client contact info, invoices, products and resources from Scoro to Business Central and also the other way so that the user can choose which system is necessary for their needs without wasting time on manual data entry from one system to the other.

The aim of this thesis is to finish what the team started in their previous course by polishing the code, developing the last features, making sure everything is in working order and preparing the necessary documentation for publication in the Microsoft AppSource. As a result of this thesis, the solution is made available in the AppSource store.

The thesis is in Estonian and contains 55 pages of text, 5 chapters, 58 figures, 3 tables.

We would like to thank our thesis supervisors from Columbus Estonia AS for their excellent support and communication.

Lühendite ja mõistete loetelu

AL	*Application language* programmeerimiskeel Dynamics 365 Business Centrali arendamiseks
API	*Application Program Interface*
AppSource	Microsoft äritarkvara liidestuste pood
BC	Business Central
Business Central	Microsofti ERP tarkvara
CRM	Kliendisuhete haldamine
CSS	*Cascading Style Sheets*
Columbus	Columbus Eesti AS, juhendav ettevõte
ERP	ettevõtte ressursside planeerimise tarkvaralahendus
HTML	*HyperText Markup Language*
js	*JavaScript*
JSON	*JavaScript Object Notation*
REST	Tarkvaraarhitektuuri laad
SaaS	*Software as a service*, tarkvara teenusena
Scoro	Ärihaldustarkvara
ID	Identifikatsioon
CA	*Certificate authority*

Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

1.1 Üldine taust ja projekti lühikirjeldus

Columbus on ülemaailmne digitaliseerimislahendusi ja ärilahendusi pakkuv ettevõte. Ettevõte pakub oma klientidele lisaks kõikidele muudele teenustele, ärijuhtimise protsessi toetamiseks majandustarkvara Microsoft Dynamics 365 Business Central. Majandustarkvaras on ühendatud finantsid, müük, teenused ja operatsioonid, et suurendada tõhusust ja tootlikkust automatiseeritud tööülesannete ja töövoogudega. Tarkvaras on ka võimalik hallata inventuuri, tarneahelat ja logistikat [3]. Scoro on ärihaldustarkvara professionaalsete ja loominguliste teenuste jaoks. See hõlmab endas tööhaldust, müügi- ja kliendi haldust, finantside planeerimist ja haldust, samuti kõike projektidejuhtimisega seonduvat [13].

Projekt valmis Tallinna Tehnikaülikooli tudengite Sken Selge, Kaari Kaasiku, Henrik Hanssoni ja Columbus Eesti AS koostöös kahes erinevas etapis, esialgu meeskonnaprojekti aine raames ja seejärel lõputööna.

Meeskonnaprojekti aine raames sai tiimi ülesandeks Scoro ärihaldustarkvara liidestamine Business Centrali tarkvaraga. Selle tulemusel valmis esialgne Business Centrali laiendus Scoro andmete sünkroniseerimiseks. Scoro tarkvaral on *REST API*, mida kasutab valminud laiendus andmete pärimiseks ja Scorosse tagasi salvestamiseks. Lõputöö raames valmisid lisaarendused Business Centrali laienduses, analüüsiti selle kasutajaliidest, turule toomise võimalusi ja koostati kõik vajalikud dokumendid tarkvara üleslaadimiseks Microsofti AppSource keskkonda.

Microsoft AppSource on äritarkvara liidestuste pood. Microsoft ja tema koostööpartnerite loodud ning sertifitseeritud lahendused on seal saadaval tasuta, katseajaga või tasuliselt. Leidub Dynamics 365, SaaS rakendusi, Microsoft 365 ja muid Microsoft partnerite loodud rakendusi [16].

1.2 Probleem ja projekti eesmärk

Columbus alustas enne tudengite projekti kaasamist liidese arendamisega Scoro müügitarkvarale, kuid see jäi pooleli. Jõuti valmis teha raamistik, kust oli puudu kood, mis paneb tööle nõutud protsessid. TalTechi tiimi ülesandeks sai alustatud töö lõpetamine, et saaks Scorost sünkroniseerida uusi kliente ja arveid Business Centrali andmebaasi ning ka uusi kliente, tooteid ja ressursse tagasi Scorosse sünkroniseerida. Tiimi eesmärk on liidestada kaks infosüsteemi [6]. Aine "Meeskonnaprojekt" raames valmisid esialgsed arendused ja sai valmis esialgne Business Centrali laiendus. Lõputöö käigus pidid valmima Microsofti nõuetele vastavad tarkvara arendused, automaattest, publitseerimiseks vajalikud laienduse dokumendid, sealhulgas: kasutusjuhend, tooteleht, klienditoe leht, logo, test stsenaarium ja turundusmaterjal.

1.3 Lühiülevaade teostatud funktsionaalsusest

Tähtsaim funktsionaalne nõue on klientide, toodete ja ressursside info ülekanne ühest tarkvarast teise ning ülekantud info kuvamine mõlemas süsteemis. Lisaks on sünkroniseeritud müügiarvete liikumine Scorost Business Centrali keskkonda. Klientide sünkroniseerimine toimub süsteemide vahel mõlemas suunas, tooteid ja arveid sünkroniseeritakse ühte pidi Scorost Business Centrali. Ressursside sünkroniseerimine toimub vaid Business Centralist Scorosse, et oleks võimalik neid müügiprotsessis kasutada. Mittefunktsionaalse nõudena saab süsteem olema visuaalselt ühilduv olemasolevate tabelitega.

Peamised kasutusjuhud:

1. Business Centrali kasutajana soovin üle tuua tooted Scorost.
2. Mõlema rakenduse kasutajana tahan Business Centralis muudetud tooted üle viia Scorosse
3. Business Centrali kasutajana soovin üle tuua kliendid Scorost.
4. Mõlema rakenduse kasutajana tahan Business Centralis muudetud kliendid üle viia Scorosse.
5. BC kasutajana soovin ületoodud tooteid ja kliente näha neile vastavates tabelites.

6. BC kasutajana soovin ostuarveid koostada kasutades Scorost ületoodud kliendi andmeid.
7. BC kasutajana soovin varasemalt koostatud müügiarveid ja dokumente tuua Scorost BC'sse.
8. Müügiarvete koostamiseks soovin, et mõlemas süsteemis oleksid ühtivad kliendid, ressursid ja tooted.
9. BC kasutajana tahan liidestada uue Scoro keskkonnaga oma rakenduse ehk soovin nullida viimati sünkroniseeritud kuupäeva.
10. BC kasutajana soovin üle viia ressursid Scorosse, et koostada seal vastavaid arveid.
11. Ettevõtte administraatorina peab mul olema võimalus teha sünkroniseerimiseks vajalikud seadistused.

Käesolev lõputöö annab detailse ülevaate meeskondlikus projektist, valminud funktsionaalsustest ja protsessist üldiselt. Lõputöö koosneb viiest põhilisest osast: metoodika, töö tulemused, analüüs ja järeldused, kokkuvõte ning lisad.

Metoodika peatükis on kirjeldatud objektide detailne kirjeldus, kasutatud tööriistad ja kirjeldatud tööprotsessi. Töö tulemustes on kirjeldatud projekti tehniline dokumentatsioon. Analüüsi ja järelduste all on analüüsitud nõudeid, üldist arhitektuuri, disaini, koodi, teste ja tehtud kirjanduse ülevaade.

2. Metoodika

2.1 Objekti detailne ülevaade

Columbus Eesti AS pakub oma klientidele erinevaid teenuseid. Muuhulgas jagatakse ja arendatakse majandustarkvara Microsoft Dynamics 365 Business Central. Tegemist on ettevõtte haldamiseks ettenähtud majandustarkvaraga, kus on võimalik lihtsalt ja kiiresti hallata oma ettevõtte äriprotsesse, muuta neid efektiivsemaks ja anda neile lisaväärtust. Selle projekti eesmärgiks oli luua liidestus kahe veebirakenduse vahel, mis sünkroniseerib nende API-de vahel erinevaid andmeid. Üheks osapooleks oli nimetatud majandustarkvara Business Central ja teiseks osapooleks ärihaldustarkvara Scoro. Business Central on Microsofti ettevõtte poolt loodud tarkvara, mis pakub klientidele teenuseid läbi pilvetehnoloogiate. Business Centralis jäi meie ülesandeks luua laiendus, et kahe API vahel hakkaksid andmed vastavalt vajadusele liikuma. Scoro tarkvara tiim arendada ei saanud, sealt andmete kättesaamiseks ja sinna salvestamiseks oli ligipääs vaid nende API-le, kuhu sai päringuid saata. Columbus Eesti lõi tiimile vastavate arenduste tegemiseks demo-keskkonna ja andis ligipääsu mõlemale tarkvarale. Projekti peamiseks eesmärgiks oli müügidokumentide ületoomine Business Centrali majandustarkvara keskkonda. Lisaks müügidokumentidele, hakkasid kahe veebirakenduse vahel liikuma ka klientide, ressursside ja toodete andmed.

Projekti esimeses faasis, aine “Meeskonnatööprojekt” raames, valmis arenduste käigus klientide, toodete ja ka müügidokumentide esialgsed sünkroniseerimised kahe tarkvara vahel. Olemasolevale Business Centrali tarkvarale valmis laiendusleht, kus lihtsa nupuvajutusega oli võimalik liigutada andmeid ühest rakendusest teise. Projekti lõpus aga kõikide andmete sünkroniseerimine ei toimunud päris nii nagu peaks, tekkisid erinevad anomaaliad andmete ületoomisel ja kõik ei olnud päris korrektne.

Projekti teises faasis, ehk käesoleva lõputööga, otsustati minna tootega turule ja laadida see rakendus kõigile kasutamiseks Microsoft AppSource keskkonda. Selleks, et seda toodet sinna laadida, oli aga vaja teha veel arendusi, täita vajalikud dokumendid ja luua

kasutusjuhendid, analüüsida kasutajaliidese funktsionaalsust ja selle intuitiivsust. AppSource keskkonda üleslaadimiseks peab rakendus vastama Microsofti poolt ettenähtud nõutele, mis tähendab et olemas peab olema test stsenaarium, turundusmaterjalid, erinevad dokumendid ja koodil peab olema kindel nimetamisstiil.

2.2 Tööriistade kirjeldus

Business Central tarkvarast lähtudes on põhiliseks kasutatavaks programmeerimis-keeleks Business Centrali rakendusspetsiifiline andmebaasi keel AL (Joonis 1). Eelneva kokkupuute puudumise tõttu kulus arendusprotsessist märkimisväärne osa keele erisuste õppimiseks. Õppimiseks kasutasime Microsofti enda dokumentatsiooni. AL keel on tugevate tüüpidega keel, mis sarnaneb süntaksilt paljus populaarsetele keeltele nagu C# ja Java. Andmetüübid erinevad nime poolest, kuid töötavad üldiselt sarnaselt (nt Java *String* on AL'i *Text*). Koodiplokkide (tsüklid, funktsioonid) alustamiseks ja lõpetamiseks kasutatakse loogsulgude asemel võtmesõnu *begin* ja *end.* Muutujaid saab deklareerida nii lokaalselt funktsioonis (AL-is *procedure*) kui ka objektis globaalselt. Funktsioonid ja muutujad võivad samuti olla lokaalselt ligipääsetavad või globaalsed [4].

```
1 reference
local procedure CheckIfProductAlreadyExistsInScoro(): Integer
var
    CustomerRecord: Record Customer;
    NumberOfNewRecords: Integer;
begin
    CustomerRecord.Reset();
    if CustomerRecord.FindSet() then begin
        repeat
            if CustomerRecord."CPC-SC Scoro ID" = 0 then begin
                PostOneCustomerToScoro(CustomerRecord);
                NumberOfNewRecords := NumberOfNewRecords + 1;
            end
        until CustomerRecord.Next() = 0;
    end;
    exit(NumberOfNewRecords);
end;
```

Joonis 1. Näide lihtsast funktsioonist AL keeles, mis kontrollib, kas Scoro klient on juba BC keskkonnas olemas. Kui klienti pole, siis funktsioon saadab kliendi BC-sse. Funktsioon väljastab uute kirjete arvu.

Keeles on kolm põhilist objekt-tüüpi olemit: tabel (*table*), leht (*page*) ja koodiühik (*code unit*). Mitmed äriloogika tabelid on Microsofti enda poolt loodud ning nende puhul on võimalik muuta vaid tabelis olevaid andmeid ning vajadusel lisada täiesti uusi andmevälju (olemasolevad väljad on kohustuslikud). Uute väljade lisamiseks on tabeli laiendi objektitüüp [4].

Scoro programmeerimise poolele ligipääsu meil pole ja suhtlus käib REST API-ga JSON objekte kasutades. Columbuse virtuaalkettal asuvat koodi kirjutatakse Visual Studio Code keskkonnas. Azure DevOpsi kasutame versioonikontrolliks. Enne koodi kirjutamist on Postmanis testitud JSON päringuid.

Turundusliku koduleht sai tehtud Visual Code keskkonnas CSS, HTML ja js keeltes. Seejärel Azure virtuaal programmeerimiskeskkonda üle tõstetud.

Omavahel suhtlemiseks kasutasime Facebook Messengeri ning ettevõtte juhendajatega suhtlemiseks Gmail ja Teamsi videokõnesid. Meeskonnatöö jaoks Google Docs ja protokollide hoiustamine Google Drives. Videokõnede järelvaatamisel oli abiks Sharepoint. Aja jälgimiseks Toggl Track. Turundusmaterjalide jaoks Adobe Illustrator.

2.3 Tööprotsessi kirjeldus

Meeskonnaprojekti ainet jätkates jätkasime samades rollides. Henrik oli arendaja ja Kaari ning Sken olid analüütikud/konsultandid.

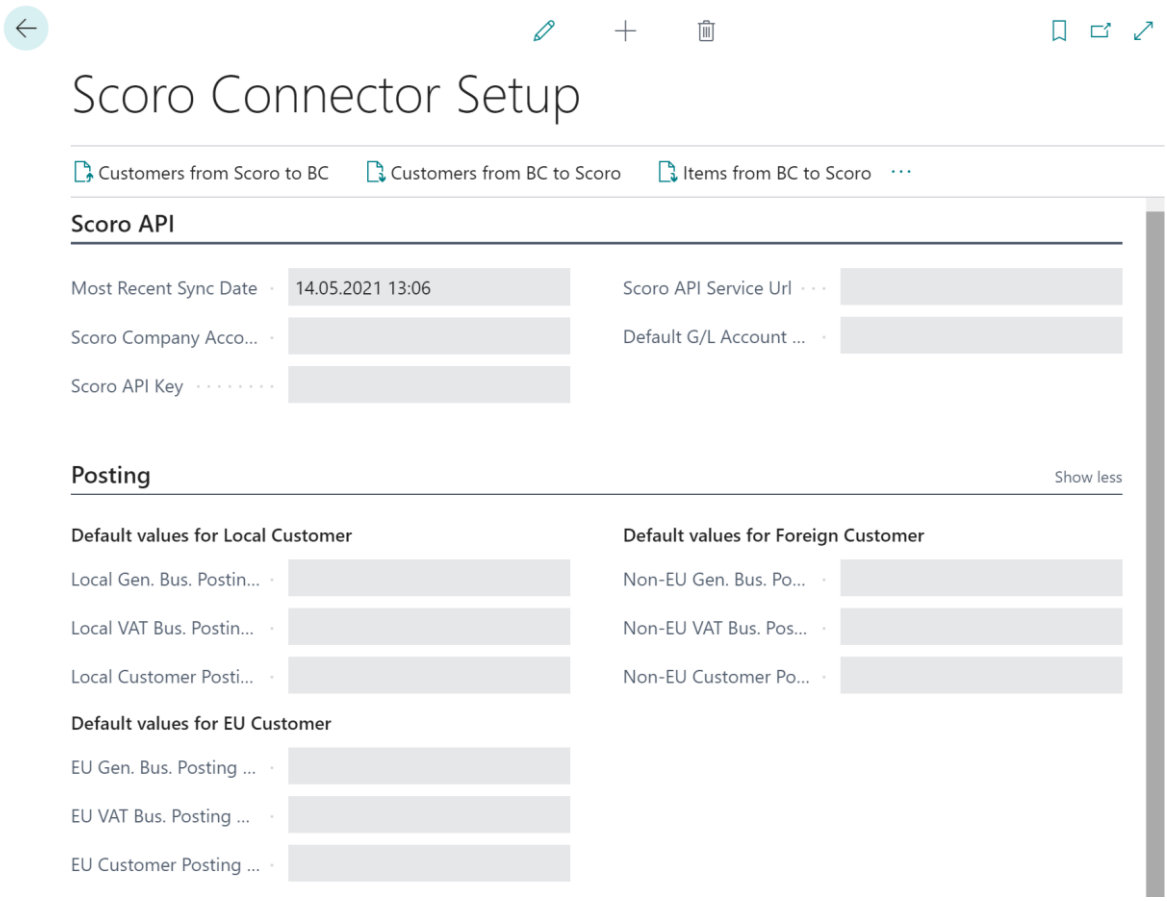
Töö algas juhendajalt arenduse lahtiste otsade nimekirja saamisega, mille Henrik juhendamise abil sai tehtud. Lahtised otsad olid näiteks: nimestandardi jälgimine, koodi puhtus, puuduoleva funktsionaalsuse lisamine, pisivigade parandamine jne. Puuduoleva ressursside andmeväljade nõuded Business Centrali ja Scoro vahel panid paika Kaari ja Sken ning Henrik seejärel arendas nõuete järgi koodi. Lisaks uuris esimesel nädalal kogu tiim läbi eelneva aine raames valminud rakenduse, et leida probleeme, mis vajasid nüüd lahendamist. AppSource jaoks dokumente koostasid Sken ja Kaari. Laienduse jaoks teststsenaariumi ja abilehe koostas Sken, turundusmaterjale tegi Kaari.

Igal nädalal oli ettevõtte juhendajatega koosolek, kus tegime nädala jooksul tehtu ülevaatuse, järgneva nädala plaanide paikapaneku ning kui koosolekul aega, siis lahendasime probleeme ja juhendajad jagasid teadmisi.

3. Tulemused

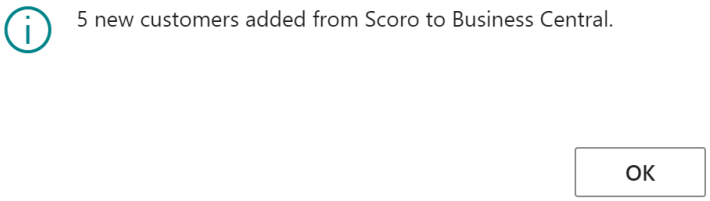
Antud peatükis teeme ülevaate valminud funktsionaalsustest ja rakenduse publitseerimiseks vajalikest dokumentidest. Lähtuvalt olemasolevast BC disainist rakendus Scoro *Connector Setupile* disainimall. *Connector Setup* (Joonis 2)on kasutamiseks ettevõte administraatorile, kes antud seadistusvaates saab teha järgnevat:

- Vaadata, millal toimus viimne sünkroniseerimine ning vastavalt sellele tuuakse kontaktid/teenused/arved/ressursid üle.
- Sisestada sünkroniseerimiseks vajaliku *API* võtme.
- Sisestada vajaliku *Scoro Company Account ID*.
- Sisestada Scoro *API* lõpp-punkti *URL*i.
- Valida vaike kulukonto numbrit, mille alla salvestuvad liigitamata müügiartiklid.
- Seadistada Business Centrali klientide jaoks vajalikud konteeringurühmad.
- Seadistused salvestuvad väljadesse ning pole järgnevatel sünkroniseerimisel vajalik uuesti sisestada.
- Valida, missuguseid ja mis suuna pidi soovitakse sünkroniseerida andmeid kahe rakenduse vahel.

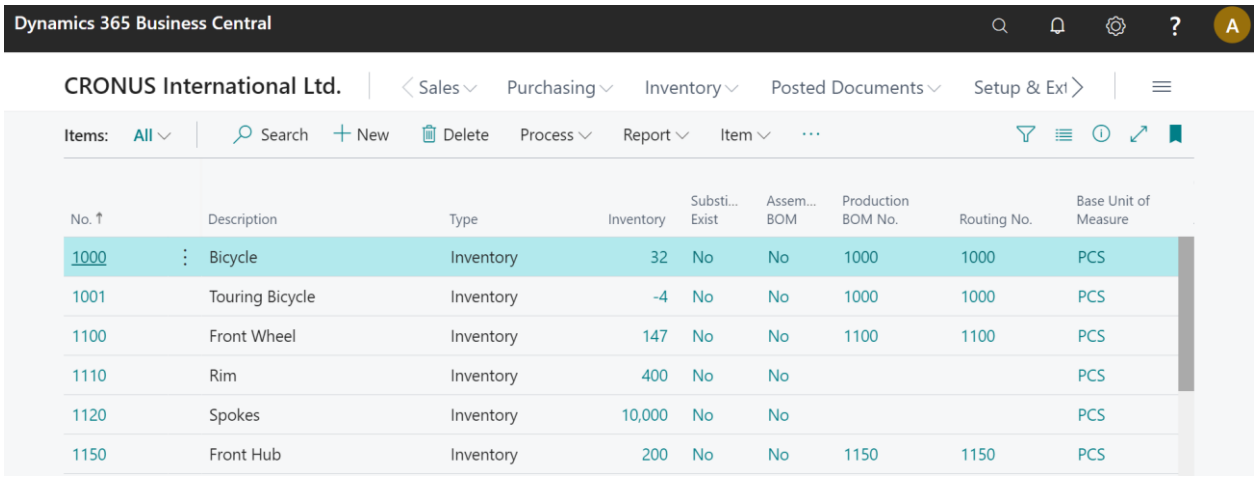


Joonis 2. Scoro Connector Setup page.

Edukal sünkroniseerimisel ilmneb kasutajale teavitus ning andmed on leitavad valitud kohtades (Joonis 3). Business Centralisse sünkroniseeritud klientide andmed jõuavad *Customer*'i tabelite alla, tooted *Items* tabelisse (Joonis 4) ja müügiarved *Invoice* tabelisse. Scoro keskkonda sünkroniseerimisel kliendid paigutatakse klientide loetelusse (Joonis 5), ressursid ja tooted jõuavad mõlemad toodete kataloogi, sealt edasi on võimalik nendega juba luua arveid ja arveid hiljem Business Centrali keskkonda juba tagasi sünkroniseerida.

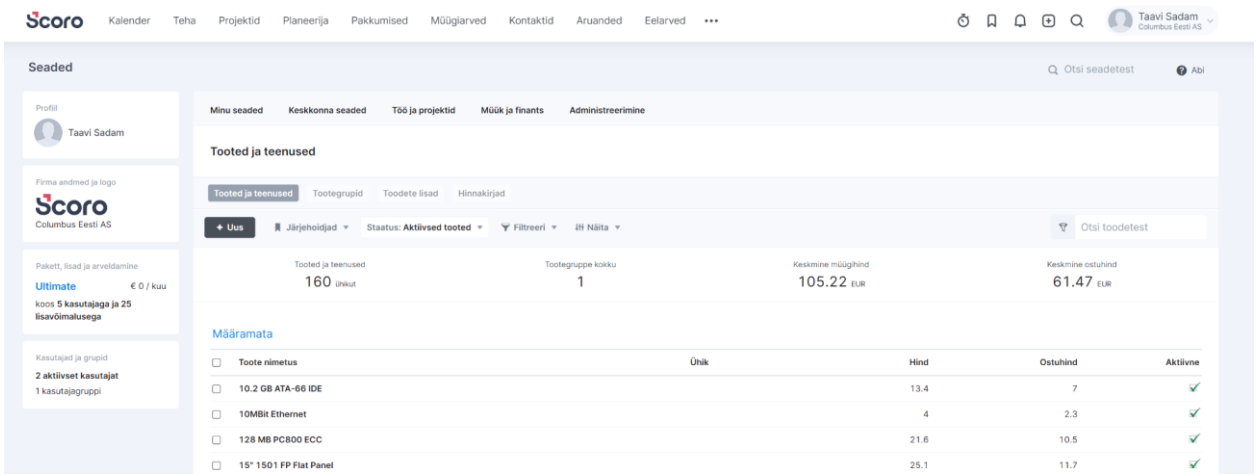


Joonis 3. Edukas sõnum.



No.	Description	Type	Inventory	Substi... Exist	Assem... BOM	Production BOM No.	Routing No.	Base Unit of Measure
1000	Bicycle	Inventory	32	No	No	1000	1000	PCS
1001	Touring Bicycle	Inventory	-4	No	No	1000	1000	PCS
1100	Front Wheel	Inventory	147	No	No	1100	1100	PCS
1110	Rim	Inventory	400	No	No			PCS
1120	Spokes	Inventory	10,000	No	No			PCS
1150	Front Hub	Inventory	200	No	No	1150	1150	PCS

Joonis 4. Toodete tabel BC's.



Joonis 5. Toodete tabel Scoros.

3.1 Arendus

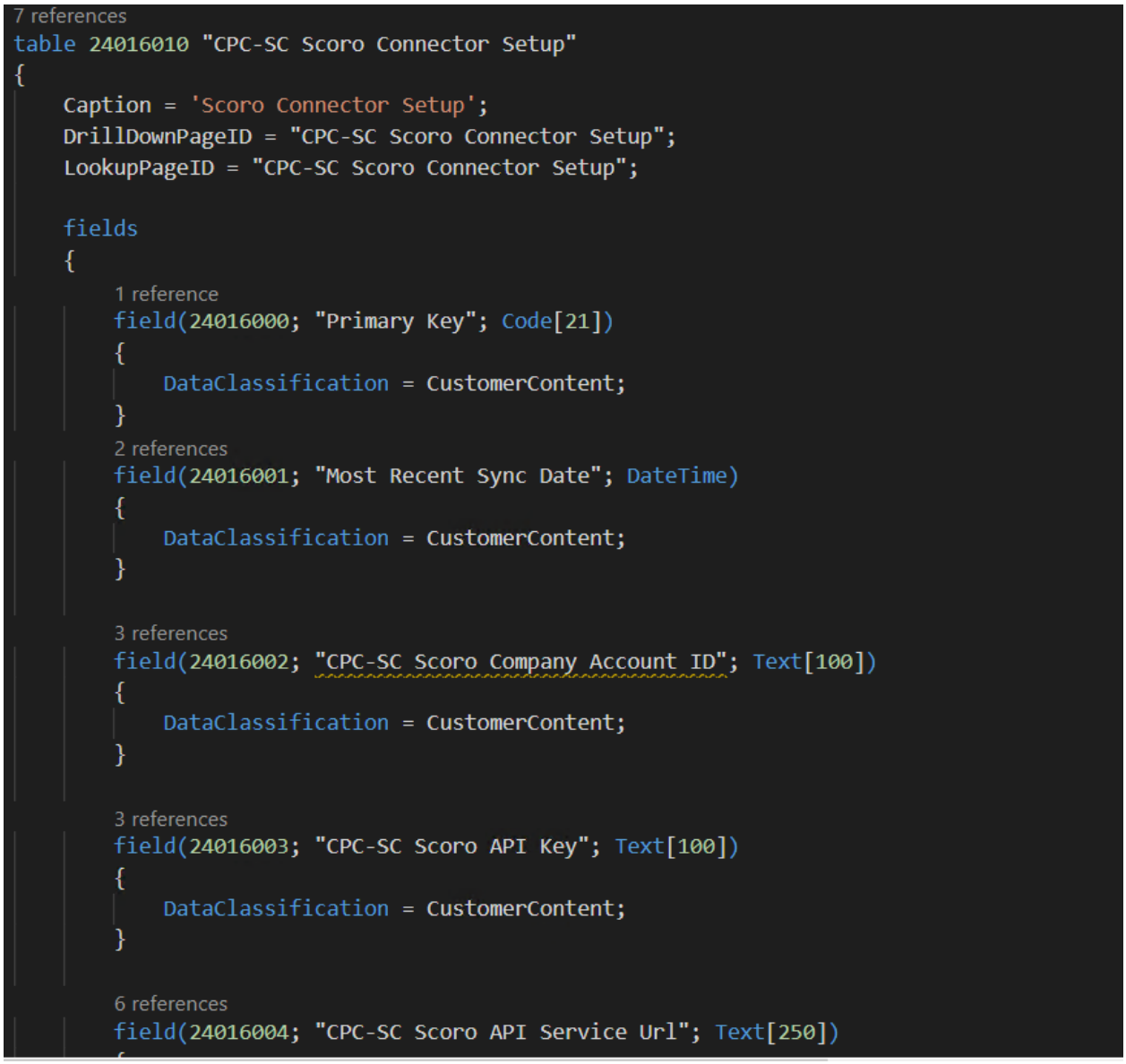
Olemasolevad Business Centrali tabelid, mida oma koodiga muudame (Joonis 6), on klientide, toodete, ressursside ja müügiarvete omad. Kõigile neile lisame laiendiga ka sünkroniseerimisloogikaks vajaliku "*CPC-SC Scoro ID*" välja (Joonis 7). Keskkonnas on võimalik lisada ka täiesti uusi tabeleid enda personaliseeritud väljadega. Seda võimalust kasutame "*Scoro Connector Setup*" tabeli loomisel (Joonis 8), kuhu peidame mitmed sünkroniseerimiseks vajalikud andmed nagu Scoro *API*-ga ühendamiseks vajaliku võtme, URL-i, konteerimisrühmad ja viimase sünkroniseerimisaja.

```
table 9 "Country/Region"
{
    Caption = 'Country/Region';
    LookupPageID = "Countries/Regions";

    fields
    {
        field(1; "Code"; Code[10])
        {
            Caption = 'Code';
            NotBlank = true;
        }
        field(2; Name; Text[50])
        {
            Caption = 'Name';
        }
        field(4; "ISO Code"; Code[2])
        {
            Caption = 'ISO Code';
            DataClassification = CustomerContent;

            trigger OnValidate()
            var
                DotNet_Regex: Codeunit DotNet_Regex;
            begin
                if "ISO Code" = '' then
                    exit;
                if StrLen("ISO Code") < MaxStrLen("ISO Code") then
                    Error(ISOCodeLengthErr, StrLen("ISO Code"), MaxStrLen("ISO Code"), "ISO Code");
                DotNet_Regex.Regex('^[a-zA-Z]*$');
                if not DotNet_Regex.IsMatch("ISO Code") then
                    FieldError("ISO Code", ASCIILetterErr);
            end;
        }
        field(5; "ISO Numeric Code"; Code[3])
```

Joonis 6. Näide Business Centrali keskkonnaga kaasa tulevast, osaliselt muudetavast tabelist “Country/Region”.

```
tableextension 24016002 "CPC-SC Tab-Ext24016002" extends "Country/Region"

    fields
    {
        2 references
        field(24016000; "CPC-SC Scoro Code"; Code[10])
        {
            DataClassification = CustomerContent;
        }
    }

    var
        0 references
        myInt: Integer;
}
```

Joonis 7. Näide tabelilaiendist, mis lisab olemasolevasse riikide tabelisse välja “CPC-SC Scoro Code”.



```
7 references
table 24016010 "CPC-SC Scoro Connector Setup"
{
    Caption = 'Scoro Connector Setup';
    DrillDownPageID = "CPC-SC Scoro Connector Setup";
    LookupPageID = "CPC-SC Scoro Connector Setup";

    fields
    {
        1 reference
        field(24016000; "Primary Key"; Code[21])
        {
            DataClassification = CustomerContent;
        }
        2 references
        field(24016001; "Most Recent Sync Date"; DateTime)
        {
            DataClassification = CustomerContent;
        }

        3 references
        field(24016002; "CPC-SC Scoro Company Account ID"; Text[100])
        {
            DataClassification = CustomerContent;
        }

        3 references
        field(24016003; "CPC-SC Scoro API Key"; Text[100])
        {
            DataClassification = CustomerContent;
        }

        6 references
        field(24016004; "CPC-SC Scoro API Service Url"; Text[250])
```

Joonis 8. Scoro *Connector Setup* tabel.

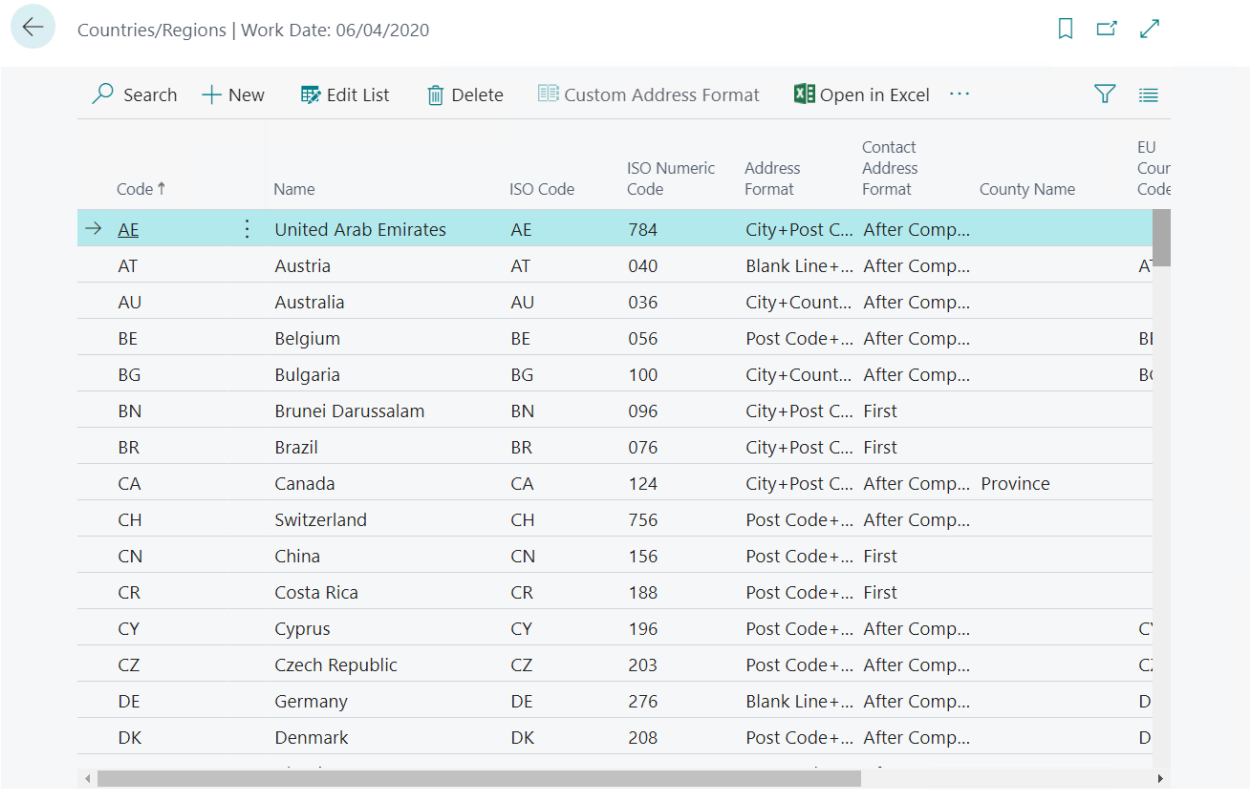
Teine suurem objektitüüp on leht, mis kujutab visuaalselt tabelis olevaid andmeid (igal lehel on oma sisendtabel). Lehti on “nimekirja” ja “kaardi” stiilis (Joonis 12). Esimene sobib hästi kõigi tabelis olevate kirjete ning teine konkreetse kirje kohta täpsema info näitamiseks. Nagu tabelitegi puhul, on olemas nii Business Centrali kohustuslikud lehed, kuhu on võimalik vaid lisada välju lehelaienditega ning muuta olemasolevaid andmeid (Joonis 9,10,11), kui ka on võimalik lisada täiesti uusi personaliseeritavaid lehti. Viimase puhul tegime ühe uue lehe Scoro *Connector Setup* tabelis olevate andmete näitmiseks ning muutmiseks (Joonis 13,14,15). Sellel lehel on ka nupud, mis käivitavad koodiühikutes oleva sünkroniseerimisloogika.

```
page 10 "Countries/Regions"
{
    ApplicationArea = Basic, Suite;
    Caption = 'Countries/Regions';
    PageType = List;
    SourceTable = "Country/Region";
    UsageCategory = Lists;

    layout
    {
        area(content)
        {
            repeater(Control1)
            {
                ShowCaption = false;
                field("Code"; Code)
                {
                    ApplicationArea = Basic, Suite, Invoicing;
                    ToolTip = 'Specifies the country/region of the address.';
                }
                field(Name; Name)
                {
                    ApplicationArea = Basic, Suite, Invoicing;
                    ToolTip = 'Specifies the country/region of the address.';
                }
                field("ISO Code"; "ISO Code")
                {
```

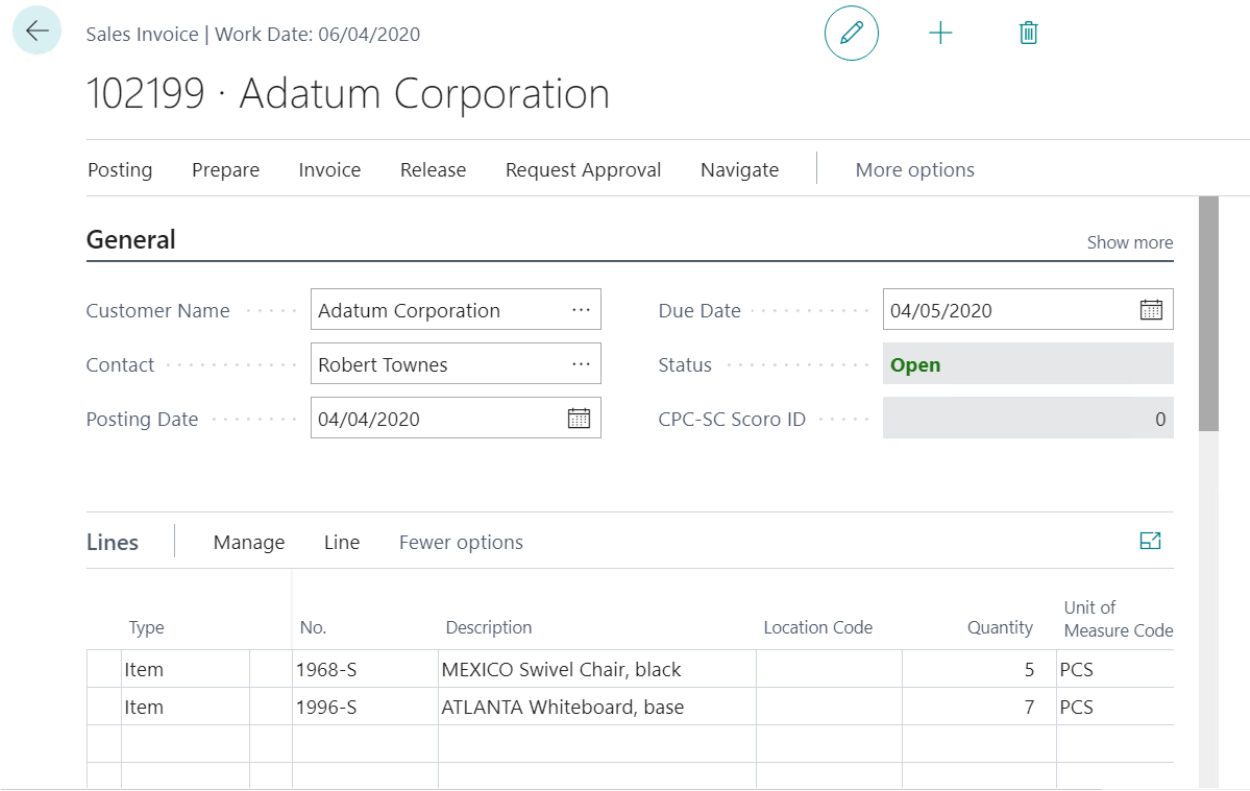
Joonis 9. Koodinäide Business Centrali keskkonnaga kaasatulevast, osaliselt muudetavast lehest *"Countries/Regions"*.

```
pageextension 24016002 "CPC-SC Pag-Ext24016002" extends "Countries/Regions"
{
    layout
    {
        addafter("VAT Scheme")
        {
            field("CPC-SC Scoro Code"; "CPC-SC Scoro Code") {
                ApplicationArea = All;
            }
        }
    }

    actions
    {
        // Add changes to page actions here
    }

    var
        0 references
        myInt: Integer;
}
```

Joonis 10. Näide lehelaiendist, mis lisab olemasolevale riikide lehele välja *"CPC-SC Scoro Code"*.



Joonis 11. *"Countries/Regions"* leht BCs tabelina.



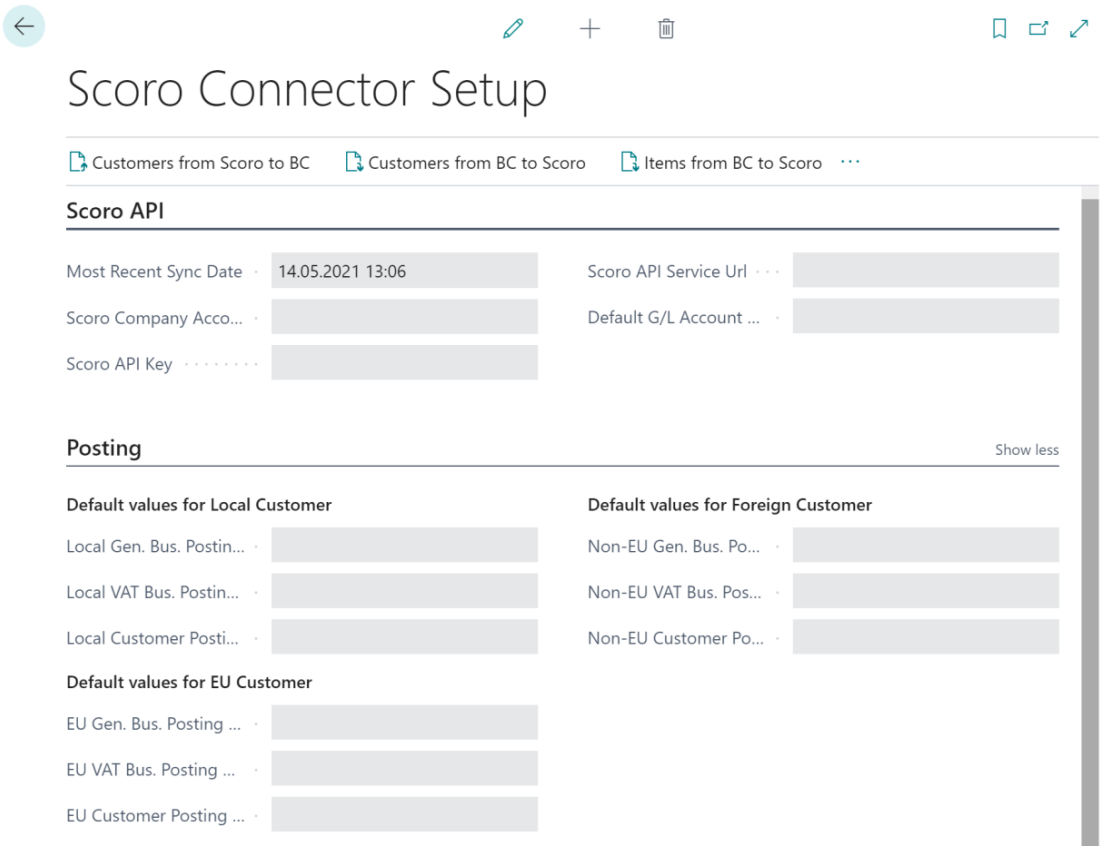
Joonis 12. Näide "kaardi" tüüpi lehest (müügiarve tabeli kirje).

```
2 references
page 24016010 "CPC-SC Scoro Connector Setup"
{
    ApplicationArea = Basic, Suite;
    Caption = 'Scoro Connector Setup';
    DeleteAllowed = true;
    InsertAllowed = true;
    Editable = true;
    PageType = Card;
    PromotedActionCategories = 'New,Process,Report,Approve,New Document,Request Approv
    UsageCategory = Administration;
    SourceTable = "CPC-SC Scoro Connector Setup";

    layout
    {
        0 references
        area(content)
        {
            0 references
            group("Scoro API")
            {
                Caption = 'Scoro API';

                0 references
                field("Most Recent Sync Date"; "Most Recent Sync Date")
```

Joonis 13. Loodud "Scoro *Connector Setup*" leht.

```
            group("Scoro API")
            {
                Caption = 'Scoro API';

                0 references
                field("Most Recent Sync Date"; "Most Recent Sync Date")
                {
                    ApplicationArea = Basic, Suite;
                    ToolTip = 'Updates when some action is done';
                    Editable = false;
                }

                0 references
                field("CPC-SC Scoro Company Account ID"; "CPC-SC Scoro Company Account ID")
                {
                    ApplicationArea = Basic, Suite;
                    ToolTip = 'Company Account ID';
                    Editable = true;
                }

                0 references
                field("CPC-SC Scoro API Key"; "CPC-SC Scoro API Key")
                {
                    ApplicationArea = Basic, Suite;
                    ToolTip = 'API key';
```

Joonis 14. Scoro *Connector Setup* leht koodis.



Joonis 15. Scoro *Connector Setup* leht BCs.

Viimane suur objektitüüp on koodiühik, millega on võimalik automaatselt muuta tabelites ja lehtedel olevad andmeid. Kirjutasime Scoroga sünkroniseerimise tarbeks 7 koodiühikut. 5 koodiühikut tegelevad konkreetselt andmete ületoomisega Scorost BC-sse (*"ScoroToBCCustomers"* ja *"ScoroToBCInvoices"* või vastupidi Business Centralist Scorosse (*"BCToScoroCustomers"*, *"BCToScoroItems"* ja *"BCToScoroResources"*). Andmete saatmine Scorost BC-sse kujutab endast tehniliselt Scoro REST API-le JSON päringu tegemist, saadud *JSON-ist* vastavate väljade kätte saamist ja siis vastavate tabelite täitmist. Andmete saatmine vastupidises suunas tähendab BC andmete põhjal *JSON* objekti koostamist ning selle siis *POST* päringuga Scoro *REST API-le* saatmist. Koodiühik *"SunoAppScoro Management"* sisaldab puhtama koodi eesmärgil arendusloogikat, mida kasutavad kõik eelmainitud ühikud (justkui *parent* klass teistes keeltes). See on näiteks *POST* päringu keha koostamine ja vastuse kontrollimine. Viimane ja väikseim koodiühik *"LastSyncDate"* (Joonis 16) muudab väljakutsumisel Scoro *Connector Setup* tabelis oleva viimase sünkroniseerimisaja välja vastavalt arvuti

ajale. Kirjeldame täpsemalt ühe koodiühiku olemust käesoleva peatüki 4. alapeatükis “Kood”.

```
5 references
codeunit 24016018 "CPC-SC LastSyncDate"
{
    5 references
    procedure SetLastSyncDateTime()
    var
        ScoroConnectorSetupRecord: Record "CPC-SC Scoro Connector Setup";
    begin
        ScoroConnectorSetupRecord.Reset();
        ScoroConnectorSetupRecord.Init();
        if ScoroConnectorSetupRecord.Find('-') then begin
            ScoroConnectorSetupRecord.Validate("Most Recent Sync Date", System.CurrentDateTime);
        end;
        ScoroConnectorSetupRecord.Modify(true);
    end;
}
```

Joonis 16. “LastSyncDate” koodiühik.

Microsofti standarditele vastamiseks kirjutasime ka ühe automaattesti. Testid on täiesti eraldi rakenduses ning sellest täpsemalt käesoleva peatüki viiendas alapeatükis.

3.1.1 Nõuded

Nõudeid esitati Columbus Eesti AS poolt koodile vähe ning konsultantidele ja arendajatele jäeti tulemuse saavutamisel vabad käed. Columbuse soovil sünkroniseerisime Scorost Business Centralisse kliendid ja müügiarved ning Business Centralist Scorosse kliendid, tooted ja ressursid. Andmeväljad, mida ühest keskkonnast üle tõime, sõltusid väljadest teises keskkonnas, mida oli võimalik täita. Väljade “sobivuse” määrasid algselt konsultandid ja siis realiseeris arendaja vastava loogika *AL* keeles. Allpool on välja toodud ressursside vastavad andmeväljad, mida on vaja sünkroniseerimisel üle viia (Joonis 17). Nõue oli veel, et sünkroniseerimisel kuvatakse kasutajale eelmise sünkroniseerimise aeg. Kohustuslik oli ka teatud nimetamise stiil: tabelite, lehtede, koodiühikute ning uute andmeväljade nimed pidid algama viisil “*CPC-SC*”. Üks oluline nõue oli näiteks riigikoodide tabeli täiendamine, sest Scoro ja Business Central kasutavad erinevaid riigikoode. Business Central kasutab kahetähelisi koode ning Scoro kolmetähelisi koode. Ehk riikidega seonduvaid andmeid üle tuues tekkis enne tabeli täiendamist viga.

BC fields	Type	Scoro	Type	Scoro comments
General				
No.	Code,PK	code	String	Product code
Name	Text	Name	String	Product name in the language specified in request object. This field is used for view and list requests only. Modify requests should put the names to the field "names" as an Array (see example).
		names	array	
Type		is_service	Boolean	In scoro product and reources are kept in the same tabel, this boolean separates the product from a resource
Unit Price	Decimal	price	Decimal	
Unit Cost	Decimal	buying_price	Decimal	
Last Date Modified	Date	modified_date	Datetime (YYYY-mm-dd HH:ii:ss)	The date when project was last modified.
CPC Scoro ID	Integer	product_id	Integer	Product ID

Joonis 17. Ressursside kaardistatud andmeväljad.

3.1.2 Arhitektuur

Arhitektuuri tarbes kasutasime olemasolevat Business Centrali rakenduse malli, millele lisasime juurde meie rakendusele spetsiifilist funktsionaalsust personaliseeritud tabelite, lehekülgede ja koodiühikutega (Joonis 18). Kui jagada kood back-end'iks ja front-end'iks, siis esimesse kuuluksid tabelid ja koodiühikud ning teise lehed (ainuke kasutajale visuaalselt nähtav osa). Puhta koodi eesmärgil on üldkasutatav koodiühikute loogika kõik ühes ühikus "*SunoAppScoro Management"*, mille funktsionaalsust siis teised ühikud kasutavad.

Joonis 18. Rakenduse struktuur.

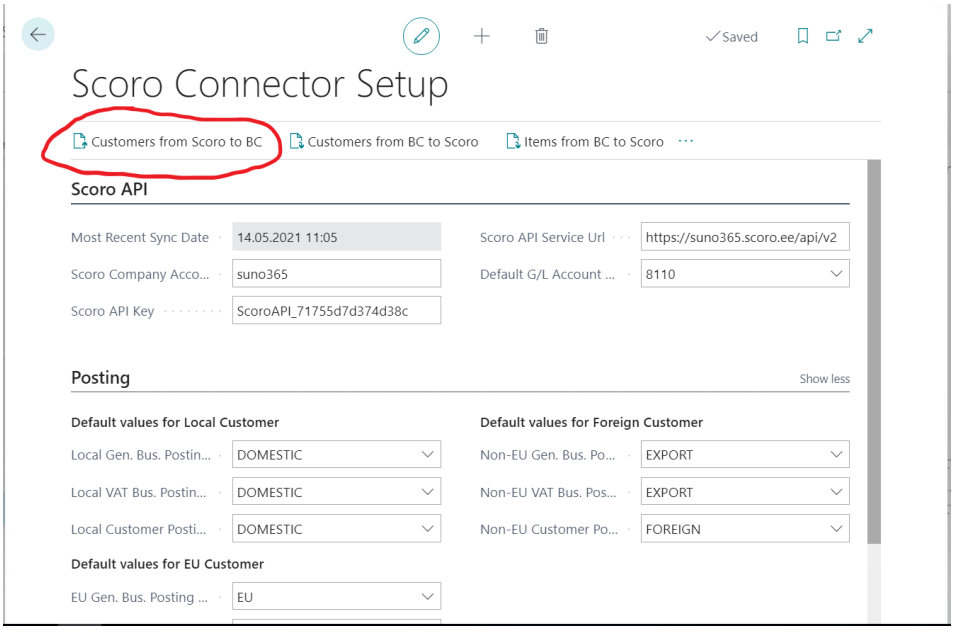
3.1.3 Disain

Ainuke rakenduse osa, mille osas me midagi disainida saime, oli "Scoro *Connector Setup*" leht, kuid ka selle puhul sõltusime suures osas olemasolevast Business Centrali "kaardi" tüüpi lehe raamistikust. See on suhteliselt range ja ei võimalda palju personaliseerimist. Võimalik on vaid lisada teksti, nuppe ja teksti kaste, ülejäänud osa pole muudetav. Lehe ülaosas asuvad nupud, mille taga on koodis *action*'id, mis käivitavad koodiühikutes oleva koodi. Ülejäänud leht koosneb kahest tekstikastide alast: "Scoro *API*" ja "*Posting*". Esimene sisaldab tekstikaste, mida on tarvis Scoro API'le päringu saatmiseks nagu *API* võti ja *URL*. Selles on ka väli *"Most Recent Sync Date"*, mida saab muuta vaid koodiühikuid välja kutsudes. *"Posting"* sektsioon sisaldab andmevälju, mida on tarvis konteerimisrühmad kliendie külge kirjete lisamiseks.

3.1.4 Kood

Kirjutatud koodi täpsemaks kirjeldamiseks kasutame ühe kasutusjuhu näidet. Valisime ühiku *"ScoroToBCCustomers"*, mis tegeleb klientide toomisega Scorost Business Centralisse. Nagu eelnevalt mainitud, tähendab see tehniliselt Scoro REST API-st päringuga saadud JSON objekti lahtipakkimist ja vastavate andmete sisestamist Business Centrali tabelisse.

Koodiühik alustab tööd, kui kasutaja vajutab *"Customers from Scoro to BC"* nuppu *"Scoro Connector Setup"* lehel (Joonis 19).



Joonis 19. "Scoro Connector Setup" leht graafiliselt, kasutatavad *actionid* on näha üleval.

Nupu taga on koodis *action*, mis omab viidet koodiühikule *"ScoroToBCCustomers"* ja kutsub välja selle funktsiooni *"PostRecordsToCustomerTable"* (Joonis 20 punane). Lisaks sellele kutsub *action* välja koodiühiku *"LastSyncDate"* funktsiooni *"SetLastSyncDate"* (Joonis 20 roheline), mis muudab viimase sünkroniseerimisaja kirjet tabelis "Scoro *Connector Setup"* vastavaks selle hetke kellaajaga.

```
actions
{
    0 references
    area(Processing)
    {
        0 references
        action("Customers from Scoro to BC")
        {
            Promoted = true;
            PromotedCategory = Process;
            ApplicationArea = All;
            Image = Import;
            Caption = 'Customers from Scoro to BC';
            ToolTip = 'Adds new contacts from Scoro to Business Central customers table.';
            trigger OnAction()
            var
                customersUnit: Codeunit "CPC-SC ScoroToBCCustomers";
                syncUnit: Codeunit "CPC-SC LastSyncDate";
            begin
                customersUnit.PostRecordsToCustomerTable();
                syncUnit.SetLastSyncDateTime();
            end;
        }
        0 references
        action("Customers from BC to Scoro")
        {
            Promoted = true;
            PromotedCategory = Process;
```

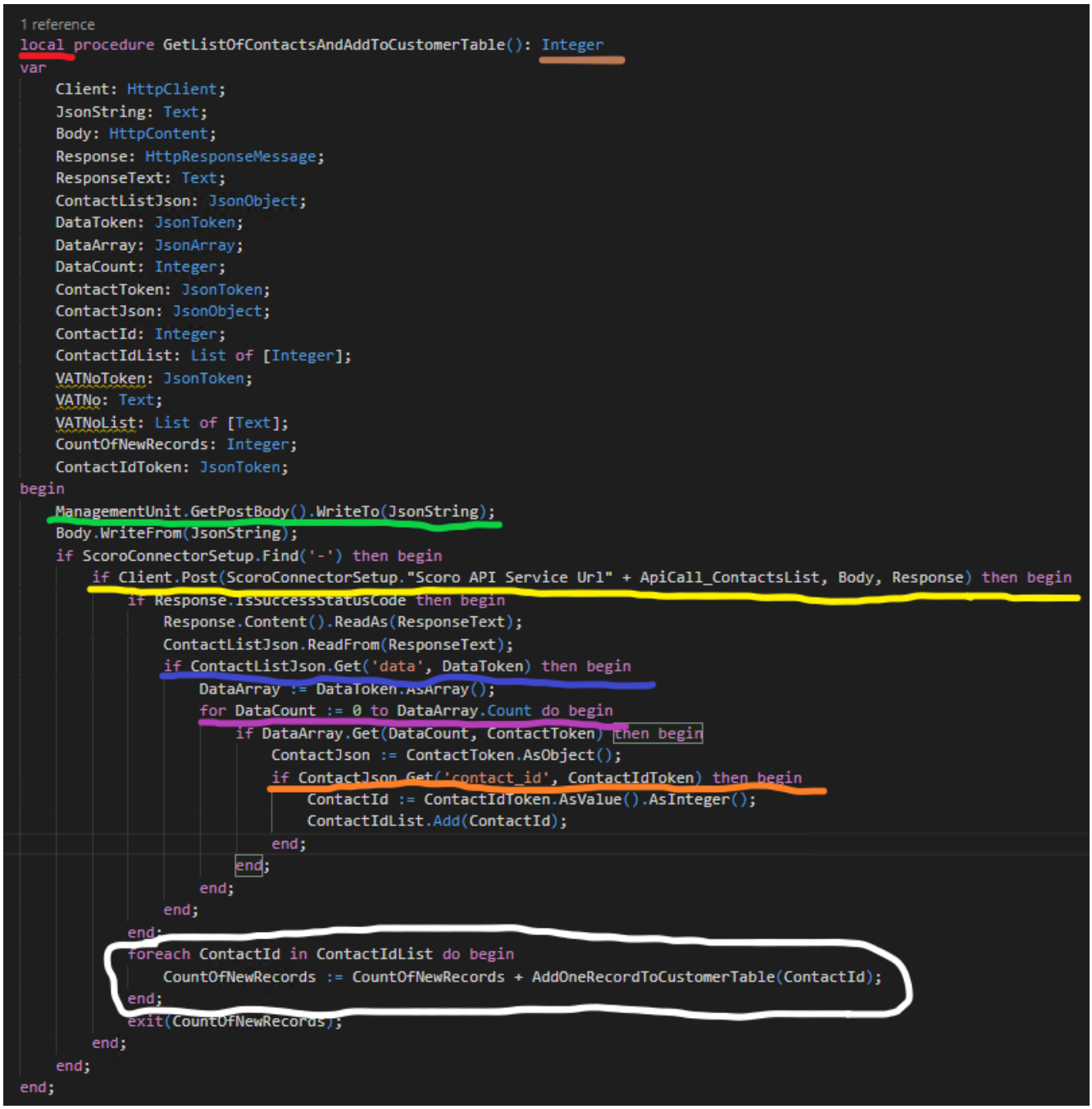
Joonis 20. "Scoro *Connector Setup*" leht koodis, *actionite* osa koodis.

"PostRecordsToCustomerTable" on ainuke avalik meetod terves ühikus ning on sellega ka ainuke ligipääsupunkt selles olevale arendusloogikale. Seda väljendab võtmesõna *local* puudumine sõna *procedure* ees (Joonis 21 punane). Seda tegime puhta koodi eesmärgil. See funktsioon sisuliselt kutsub välja ülejäänud ühikus oleva loogika ja väljastab sõnumi kasutajale, kus on kirjas eelnevast funktsioonist "*GetListOfContactsAndAddToCustomerTable*" saadud uute klientide arv. Funktsioon ühtegi andmetüüpi ei väljasta nagu void tüüpi funktsioon teistes keeltes (Joonis 21 sinine).

```
1 reference
procedure PostRecordsToCustomerTable()
var
    NumberOfRecordsAdded: Integer;
begin
    NumberOfRecordsAdded := GetListOfContactsAndAddToCustomerTable();
    Message(System.Format(NumberOfRecordsAdded) + ' new customers added from Scoro to Business Central.');
end;
```

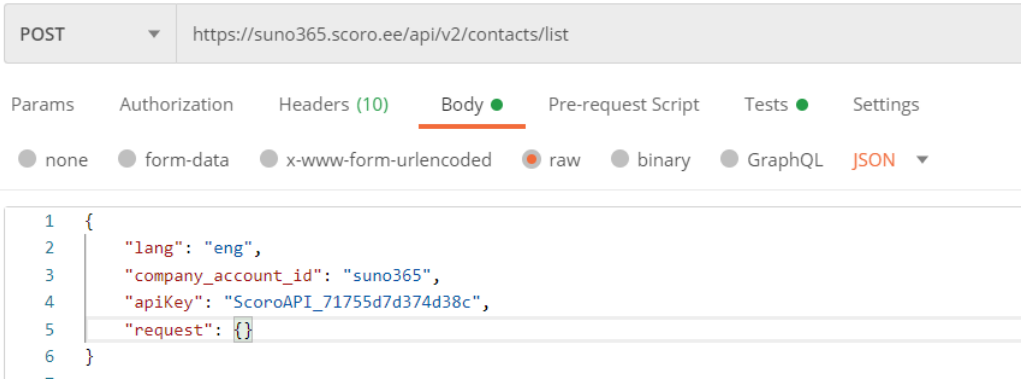
Joonis 21. Avalik "*PostRecordsToCustomer*" funktsioon, mis kutsub välja eelneva loogika ja väljastab kasutajale sõnumi.

Funktsioon "*GetListOfContactsAndAddToCustomerTable*" on juba lokaalne funktsioon (Joonis 22 punane), mis tegeleb mitme kontakti kirje korraga lisamisega. Vaja on teada saada, millised kontaktid on Scoros juba olemas ja millised vaja lisada. Selleks pärime Scorost kõigepealt kõik kontaktid. Vaja on koostada päring, milleks on tarvis *API* key-d, *URL-i* ja *Company Account ID-d* (Joonis 22 kollane, 25,26). Päringu keha koostamine on kõikidele koodiühikutele sama, seetõttu toimub see üldises ühikus *"SunoAppScoro Management*" funktsioonis *"GetPostBody"* (Joonis 22 roheline, 24). *URL* on igale Scorost saadavale andmetüübile erinev, seega lisame selle funktsioonis endas, tarvis on *URL*-ile lisada ka andmetüübile vastav laiend (Joonis 27 punane). Kui päringu tulemus on edukas (Joonis 23), siis eraldame saadud *JSON* objektist andmete osa *"data"* (Joonis 22 sinine, 28). Andmete osas on nimekirja kujul kõik kontaktid. Kontrollimiseks, millised kontaktid on olemas ja millised mitte, leiame tsükliga (Joonis 22 violetne) kõik kontaktide *ID-d* (Joonis 22 oranž) ning lisame listi. Seejärel kutsume tsükliga sama listi põhjal välja funktsiooni "*AddOneRecordToCustomerTable*" (Joonis 22 valge), mis leiab, kas kontakt on olemas või mitte ja lisab siis uue kirje. See lisab uue Scoro kontakti BC tabelisse ja väljastab integer tüüpi arvu, 1 või 0, vastavalt sellele kas vastav kontakt on juba BC tabelis olemas. Tsükkel liidab kõik need arvud kokku ja see ongi funktsiooni "*GetListOfContactsAndAddToCustomersTable*" väljund (Joonis 22 pruun), mida funktsioon "*PostRecordsToCustomerTable*" hiljem sõnumi kujul väljastab.



```
1 reference
local procedure GetListOfContactsAndAddToCustomerTable(): Integer
var
    Client: HttpClient;
    JsonString: Text;
    Body: HttpContent;
    Response: HttpResponseMessage;
    ResponseText: Text;
    ContactListJson: JsonObject;
    DataToken: JsonToken;
    DataArray: JsonArray;
    DataCount: Integer;
    ContactToken: JsonToken;
    ContactJson: JsonObject;
    ContactId: Integer;
    ContactIdList: List of [Integer];
    VATNoToken: JsonToken;
    VATNo: Text;
    VATNoList: List of [Text];
    CountOfNewRecords: Integer;
    ContactIdToken: JsonToken;
begin
    ManagementUnit.GetPostBody().WriteTo(JsonString);
    Body.WriteFrom(JsonString);
    if ScoroConnectorSetup.Find('-') then begin
        if Client.Post(ScoroConnectorSetup."Scoro API Service Url" + ApiCall_ContactsList, Body, Response) then begin
            if Response.IsSuccessStatusCode then begin
                Response.Content().ReadAs(ResponseText);
                ContactListJson.ReadFrom(ResponseText);
                if ContactListJson.Get('data', DataToken) then begin
                    DataArray := DataToken.AsArray();
                    for DataCount := 0 to DataArray.Count do begin
                        if DataArray.Get(DataCount, ContactToken) then begin
                            ContactJson := ContactToken.AsObject();
                            if ContactJson.Get('contact_id', ContactIdToken) then begin
                                ContactId := ContactIdToken.AsValue().AsInteger();
                                ContactIdList.Add(ContactId);
                            end;
                        end;
                    end;
                end;
            end;
            foreach ContactId in ContactIdList do begin
                CountOfNewRecords := CountOfNewRecords + AddOneRecordToCustomerTable(ContactId);
            end;
            exit(CountOfNewRecords);
        end;
    end;
end;
```

Joonis 22. "*GetListOfContactsAndAddToCustomerTable*" funktsioon, mis lisab mitu kirjet tabelisse korraga.



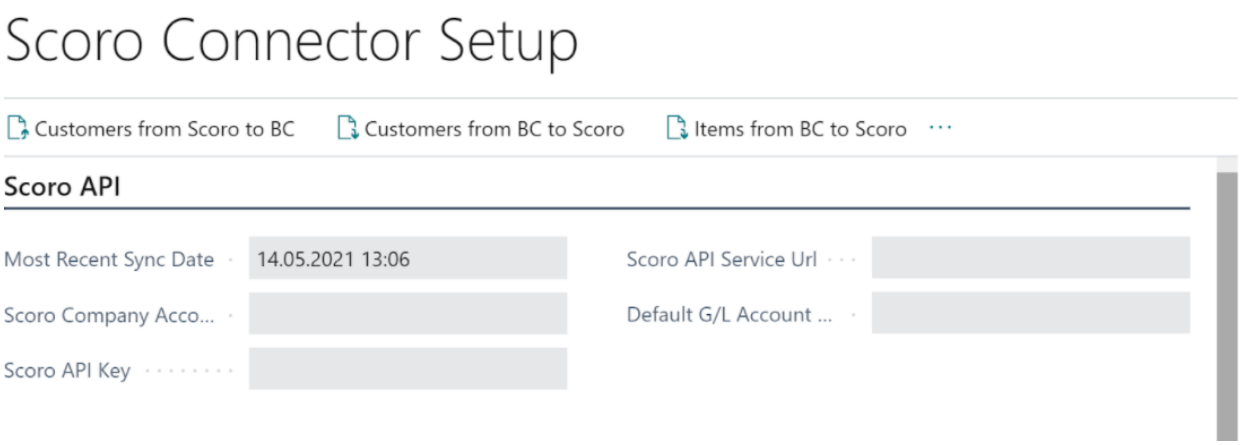
Joonis 23. Kontaktide nimekirja päringu keha.

```
4 references
procedure GetPostBody(): JsonObject
var
    JsonPostBody: JsonObject;
    EmptyJson: JsonObject;
begin
    JsonPostBody.Add('lang', 'eng');
    if ScoroConnectorSetup.Find('-') then begin
        JsonPostBody.Add('company_account_id', ScoroConnectorSetup."Scoro Company Account ID");
        JsonPostBody.Add('apiKey', ScoroConnectorSetup."Scoro API Key");
    end;
    JsonPostBody.Add('request', EmptyJson);
    exit(JsonPostBody);
end;
```

Joonis 24. "*GetPostBody*" funktsioon, mis täidab päringu keha.

```
3 references
field(24016002; "Scoro Company Account ID"; Text[100])
{
    DataClassification = CustomerContent;
    Caption = 'Scoro Company Account ID';
}

3 references
field(24016003; "Scoro API Key"; Text[100])
{
    DataClassification = CustomerContent;
    Caption = 'Scoro API Key';
}

6 references
field(24016004; "Scoro API Service Url"; Text[250])
{
    DataClassification = CustomerContent;
    Caption = 'Scoro API Service Url';
}
```

Joonis 25. "Scoro *Connector Setup*" tabeli väljad, mida kasutatakse päringu koostamiseks.



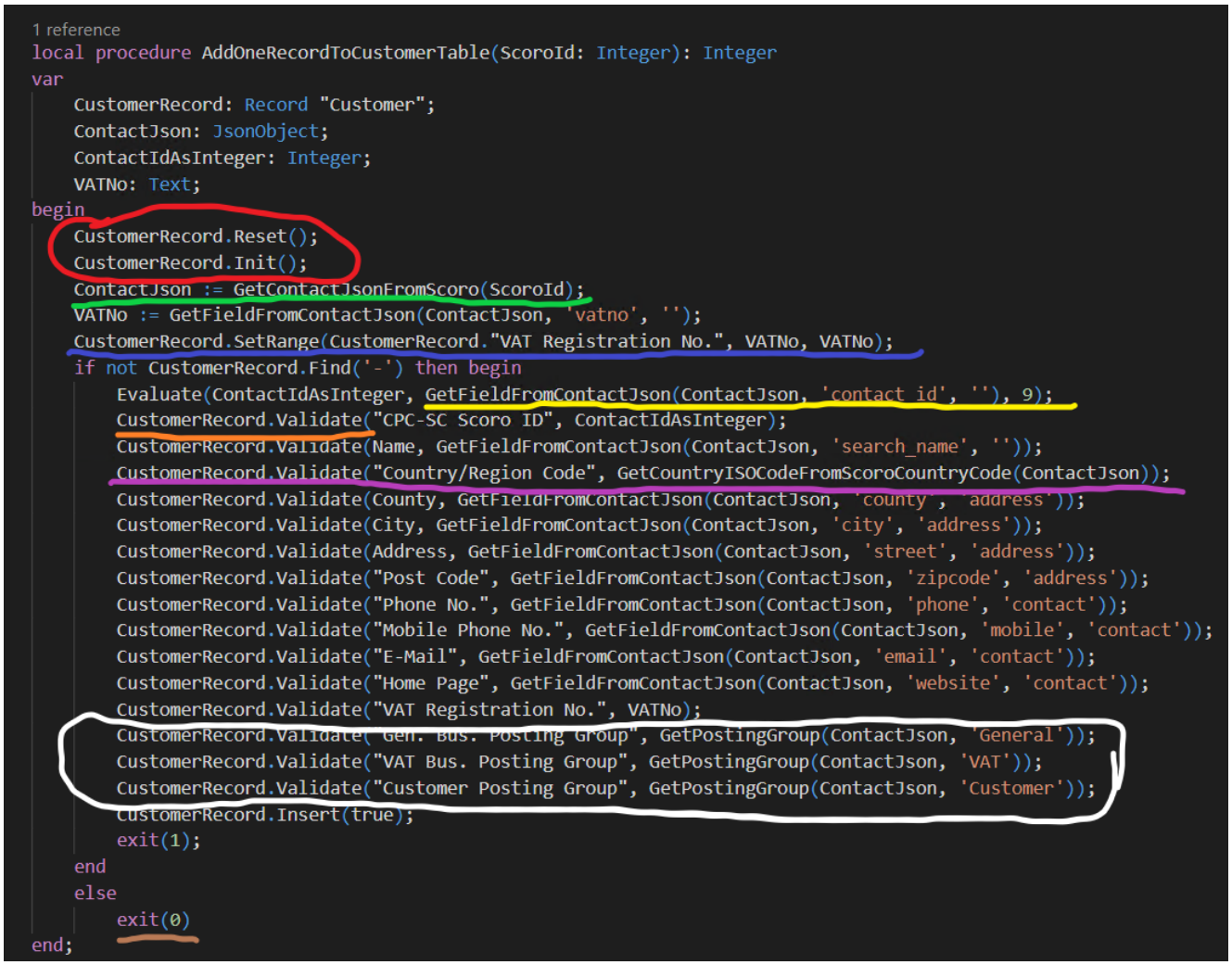
Joonis 26. Väljad "Scoro *Connector Setup*" lehel, mida kasutatakse päringu keha koostamiseks.

```
1 reference
codeunit 24016013 "CPC-SC ScoroToBCCustomers"
{
    var
        2 references
        ManagementUnit: Codeunit "CPC-SC SunoAppScoro Management";
        0 references
        ApiCall_ContactsModify: Label '/contacts/modify';
        1 reference
        ApiCall_ContactsList: Label '/contacts/view/list';
        1 reference
        ApiCall_ContactsView: Label '/contacts/view/';
        13 references
        ScoroConnectorSetup: Record "CPC-SC Scoro Connector Setup";
```

Joonis 27. Kontaktide nimekirja saamisel kasutatav *URL* laiend.

Joonis 28. *JSON-ist* eraldatav andmete osa.

Funktsioon "*AddOneRecordToCustomerTable*" tegeleb Scorost saadud andmete põhjal ühe klientide tabeli kirje lisamisega. Esmalt initsialiseerib see ühe klientide tabeli kirje (Joonis 29 punane). Siis kasutab see funktsiooni *"GetListOfContacts-AndAddToCustomerTable"* tsüklist parameetriga saadud kontakt ID-d õige kliendi JSON objekti saamiseks "*GetContactJsonFromScoro*" funktsiooniga (Joonis 29 roheline). Seejärel toimub kontroll, kas saadud kontakt on juba Business Centralis olemas. Selleks kasutame AL keele *SetRange* funktsiooni, mis filtreerib Scorost saadud käibemaksukoodi alusel Business Centrali klientide tabelit (Joonis 29 sinine). Kui leitakse vastava käibemaksukoodiga kirje, siis funktsioon väljastab lihtsalt arvu 0 (Joonis 29 pruun). Kui kirjet ei leita, siis on tarvis saadud kontaktiobjektist sisestada õiged andmed õigete väljade kohale BC tabelis. Kirje väljade täitmiseks kasutame *AL* keele *Validate* funktsiooni (Joonis 29 oranž). *JSON* objektist väljade eraldamiseks kasutame abifunktsiooni "*GetFieldFromContactJson*", millele anname argumendina kaasa objekti enda, soovitud välja nime ja selle, kas tegemist on kontakti või aadressi tüüpi väljaga (Joonis 29 kollane). Üheks erandiks on konteerimisgrupi väljad, millega kaasneb erinev loogika ja mille puhul kasutame abifunktsiooni "GetPostingGroup" (Joonis 29 valgega). Teiseks erandiks on riigi kood, mille puhul kasutame funktsiooni "GetCountryISOCodeFromScoroCountryCode" (Joonis 29 violetne).



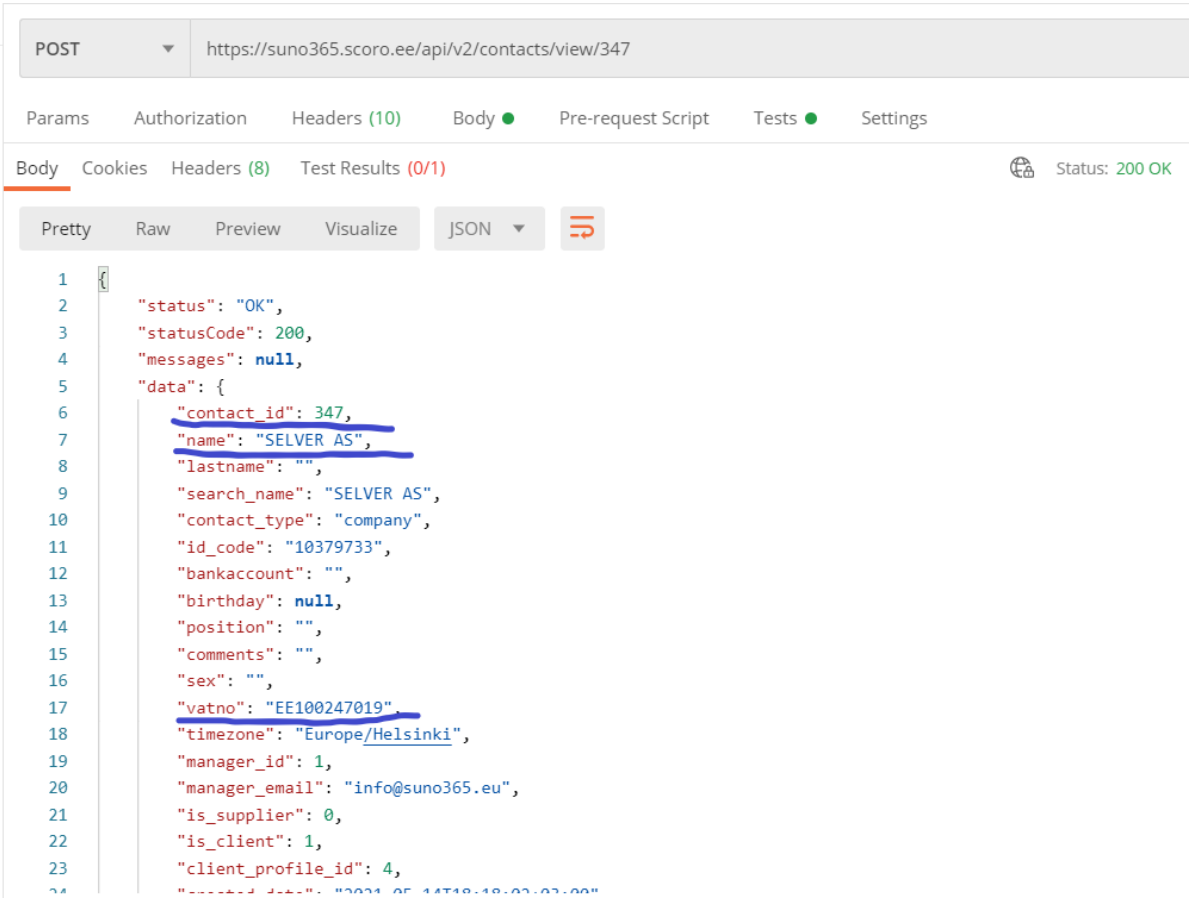
```
1 reference
local procedure AddOneRecordToCustomerTable(ScoroId: Integer): Integer
var
    CustomerRecord: Record "Customer";
    ContactJson: JsonObject;
    ContactIdAsInteger: Integer;
    VATNo: Text;
begin
    CustomerRecord.Reset();
    CustomerRecord.Init();
    ContactJson := GetContactJsonFromScoro(ScoroId);
    VATNo := GetFieldFromContactJson(ContactJson, 'vatno', '');
    CustomerRecord.SetRange(CustomerRecord."VAT Registration No.", VATNo, VATNo);
    if not CustomerRecord.Find('-') then begin
        Evaluate(ContactIdAsInteger, GetFieldFromContactJson(ContactJson, 'contact_id', ''), 9);
        CustomerRecord.Validate("CPC-SC Scoro ID", ContactIdAsInteger);
        CustomerRecord.Validate(Name, GetFieldFromContactJson(ContactJson, 'search_name', ''));
        CustomerRecord.Validate("Country/Region Code", GetCountryISOCodeFromScoroCountryCode(ContactJson));
        CustomerRecord.Validate(County, GetFieldFromContactJson(ContactJson, 'county', 'address'));
        CustomerRecord.Validate(City, GetFieldFromContactJson(ContactJson, 'city', 'address'));
        CustomerRecord.Validate(Address, GetFieldFromContactJson(ContactJson, 'street', 'address'));
        CustomerRecord.Validate("Post Code", GetFieldFromContactJson(ContactJson, 'zipcode', 'address'));
        CustomerRecord.Validate("Phone No.", GetFieldFromContactJson(ContactJson, 'phone', 'contact'));
        CustomerRecord.Validate("Mobile Phone No.", GetFieldFromContactJson(ContactJson, 'mobile', 'contact'));
        CustomerRecord.Validate("E-Mail", GetFieldFromContactJson(ContactJson, 'email', 'contact'));
        CustomerRecord.Validate("Home Page", GetFieldFromContactJson(ContactJson, 'website', 'contact'));
        CustomerRecord.Validate("VAT Registration No.", VATNo);
        CustomerRecord.Validate("Gen. Bus. Posting Group", GetPostingGroup(ContactJson, 'General'));
        CustomerRecord.Validate("VAT Bus. Posting Group", GetPostingGroup(ContactJson, 'VAT'));
        CustomerRecord.Validate("Customer Posting Group", GetPostingGroup(ContactJson, 'Customer'));
        CustomerRecord.Insert(true);
        exit(1);
    end
    else
        exit(0)
end;
```

Joonis 29. “AddOneRecordToCustomerTable”, mis lisab Scorost saadud andmete põhjal klientide tabelisse ühe kirje.

Enne *JSON* objekti lahtipakkimist vaatame funktsiooni “*GetContactJsonFromScoro*”, mis võtab parameetrina kontakti id ja pärib selle alusel Scorost vastava kontakti (Joonis 30 roheline). Kasutame taas Scoro *Connector Setup* tabelisse peidetud *API* key-d ja API URL-i, millele lisame vastava kontakti ID (Joonis 30 punane). Eraldame kohe ka saadud objektist andmete osa, mis sisaldab meid huvitavaid välju (Joonis 30 kollane).

```
1 reference
local procedure GetContactJsonFromScoro(ScoroId: Integer): JsonObject
var
    Client: HttpClient;
    Response: HttpResponseMessage;
    Body: HttpContent;
    ContactJson: JsonObject;
    ResponseText: Text;
    DataToken: JsonToken;
    DataJson: JsonObject;
    JsonString: Text;
begin
    ManagementUnit.GetPostBody().WriteTo(JsonString);
    Body.WriteFrom(JsonString);
    if ScoroConnectorSetup.Find('-') then begin
        if Client.Post(ScoroConnectorSetup."Scoro API Service Url" + ApiCall_ContactsView + System.Format(ScoroId), Body, Response) then begin
            if Response.IsSuccessStatusCode then begin
                Response.Content().ReadAs(ResponseText);
                ContactJson.ReadFrom(ResponseText);
                if ContactJson.Get('data', DataToken) then begin
                    DataJson := DataToken.AsObject();
                    exit(DataJson);
                end;
            end;
        end;
    end;
end;
```

Joonis 30. Funktsioon "*GetContactJsonFromScoro*", mis tegeleb konkreetselt kontakti päringu saatmisega.



Joonis 31. Näide ühest Scorost saadud kontaktide *JSON* objektist.

Vaatleme järgnevalt funktsiooni "*GetFieldFromContactJson*", mis on justkui sõlmpunkt *JSON* objekti lahti murdmisel. Kontakti *JSON-is* on andmed erineval kujul. Osad väljad nagu kontakti ID, nimi ja käibemaksukood (Joonis 31 sinine) on kättesaadavad lihtsalt objekti seest pärides abifunktsiooniga "*GetRegularField*". Aadressiga seotud andmed nagu riik, linn ja tänav on pesastatud teise objekti sisse (Joonis 35 sinine), mis ise omakorda on pesastatud massiivi sisse (Joonis 35 kollane). Kuna sellest andmete kätte saamisega kaasneb lisaloogika, panime vastava koodi puhta koodi eesmärgil eraldi funktsiooni "*GetAddressField*". Sarnane lugu on kontakti kontaktandmetega, mis on igaüks pesastatud massiividesse (Joonis 37 sinine) ning siis ühendatud kokku üheks objektiks (Joonis 37 kollane). Kontaktandmete kätte saamiseks kasutame abifunktsiooni "*GetMeansOfContactField*". Lõpuks konkreetse välja kätte saamiseks kasutame funktsioonide sees AL-i sissekirjutatud Get funktsiooni (Joonis 33, 34, 36 punane).

```
12 references
local procedure GetFieldFromContactJson(ContactJson: JsonObject; FieldName: Text; ContactOrAddress: Text): Text
begin
    if (ContactOrAddress = 'address') then
        exit(GetAddressField(ContactJson, FieldName))
    else
        if (ContactOrAddress = 'contact') then
            exit(GetMeansOfContactField(ContactJson, FieldName))
        else
            exit(GetRegularField(ContactJson, FieldName))
end;
```

Joonis 32. Funktsioon "*GetFieldFromContactJson*", kust algab ülejäänud JSON objekti lahtipakkimine.

```
1 reference
local procedure GetRegularField(DataJson: JsonObject; FieldName: Text): Text
var
    FieldToken: JsonToken;
begin
    if DataJson.Get(FieldName, FieldToken) then
        exit(FieldToken.AsValue().AsText());
end;
```

Joonis 33. "*GetRegularField*" funktsioon, mis eraldab JSON-ist lihtsasti kättesaadava välja.

```
1 reference
local procedure GetAddressField(DataJson: JsonObject; FieldName: Text): Text
var
    AddressArrayToken: JsonToken;
    AddressToken: JsonToken;
    AddressJson: JsonObject;
    NestedAddressFieldToken: JsonToken;
begin
    if DataJson.Get('addresses', AddressArrayToken) then begin
        if AddressArrayToken.IsArray then begin
            foreach AddressToken in AddressArrayToken.AsArray() do begin
                AddressJson := AddressToken.AsObject();
                if AddressJson.Get(FieldName, NestedAddressFieldToken) then
                    exit(NestedAddressFieldToken.AsValue().AsText());
            end;
        end
        else
            exit('');
    end;
end;
```

Joonis 34. “GetAddressField” funktsioon, mis eraldab JSON-ist aadresside massiivi ja sellese pesastatud objekti.


```
"modified_date": "2021-05-14T18:18:02+03:00",
"deleted_date": null,
"addresses": [
    {
        "country": "est",
        "county": "Harju maakond",
        "municipality": "",
        "city": "Tallinn",
        "street": "Pärnu mnt 238",
        "zipcode": "11624",
        "full_address": "Pärnu mnt 238, 11624 Tallinn, Harju maakond, Estonia",
        "contacts_addresses_id": 273
    }
],
"means_of_contact": {
    "phone": [
```


Joonis 35. *JSON* objektis olev pesastatud aadresside massiiv.

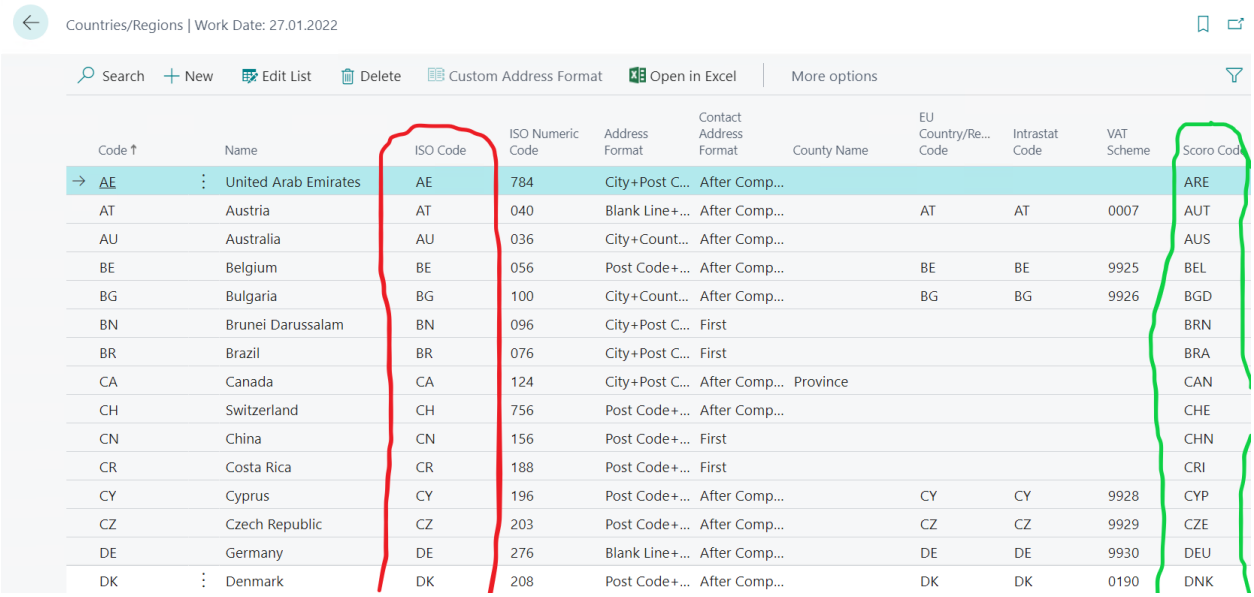
```
1 reference
local procedure GetMeansOfContactField(DataJson: JsonObject; FieldName: Text): Text
var
    MeansOfContactToken: JsonToken;
    MeansOfContactJson: JsonObject;
    NestedMeansOfContactArrayToken: JsonToken;
    NestedMeansOfContactFieldToken: JsonToken;
begin
    if DataJson.Get('means_of_contact', MeansOfContactToken) then begin
        if MeansOfContactToken.IsObject then begin
            MeansOfContactJson := MeansOfContactToken.AsObject();
            if MeansOfContactJson.Get(FieldName, NestedMeansOfContactArrayToken) then begin
                foreach NestedMeansOfContactFieldToken in NestedMeansOfContactArrayToken.AsArray() do begin
                    exit(NestedMeansOfContactFieldToken.AsValue().AsText());
                end;
            end;
        end
        else
            exit('');
    end;
end;
```

Joonis 36. "*GetMeansOfContactField*" funktsioon, mis eraldab *JSON-ist* pesastatud objekti pesastatud massiivid.

Joonis 37. *JSON* objektis olev pesastatud kontaktobjekt ja sellesse pesastatud massiivid.

Riigi ületoomine Scorost BC-sse on pisut erinev, sest Scorost saadav riigi ISO kood on kolmekohaline (Joonis 35 punane), kuid Business Centralis on riigi ISO koodid kahekohalised (Joonis 38 punane). Sellest erisusest tulenevate probleemide lahendamiseks lisasime tabelilaiendiga BC riikide tabelisse uue välja "Scoro *Country Code*" (Joonis 38 roheline), kuhu kasutaja peab ise lisama vastavad kolmekohalised koodid. Funktsioon "*GetCountryISOCodeFromScoroCountryCode*" kasutab siis

SetRange filtrit, et leida sellele kolmekohalisele koodile vastav kahekohaline kood (Joonis 39 punane).



Countries/Regions | Work Date: 27.01.2022

Search | New | Edit List | Delete | Custom Address Format | Open in Excel | More options

Code	Name	ISO Code	ISO Numeric Code	Address Format	Contact Address Format	County Name	EU Country/Re... Code	Intrastat Code	VAT Scheme	Scoro Cod
AE	United Arab Emirates	AE	784	City+Post C...	After Comp...					ARE
AT	Austria	AT	040	Blank Line+...	After Comp...		AT	AT	0007	AUT
AU	Australia	AU	036	City+Count...	After Comp...					AUS
BE	Belgium	BE	056	Post Code+...	After Comp...		BE	BE	9925	BEL
BG	Bulgaria	BG	100	City+Count...	After Comp...		BG	BG	9926	BGD
BN	Brunei Darussalam	BN	096	City+Post C...	First					BRN
BR	Brazil	BR	076	City+Post C...	First					BRA
CA	Canada	CA	124	City+Post C...	After Comp...	Province				CAN
CH	Switzerland	CH	756	Post Code+...	After Comp...					CHE
CN	China	CN	156	Post Code+...	First					CHN
CR	Costa Rica	CR	188	Post Code+...	First					CRI
CY	Cyprus	CY	196	Post Code+...	After Comp...		CY	CY	9928	CYP
CZ	Czech Republic	CZ	203	Post Code+...	After Comp...		CZ	CZ	9929	CZE
DE	Germany	DE	276	Blank Line+...	After Comp...		DE	DE	9930	DEU
DK	Denmark	DK	208	Post Code+...	After Comp...		DK	DK	0190	DNK

Joonis 38. Tabel *"Countries/Regions"* graafilisel kujul.

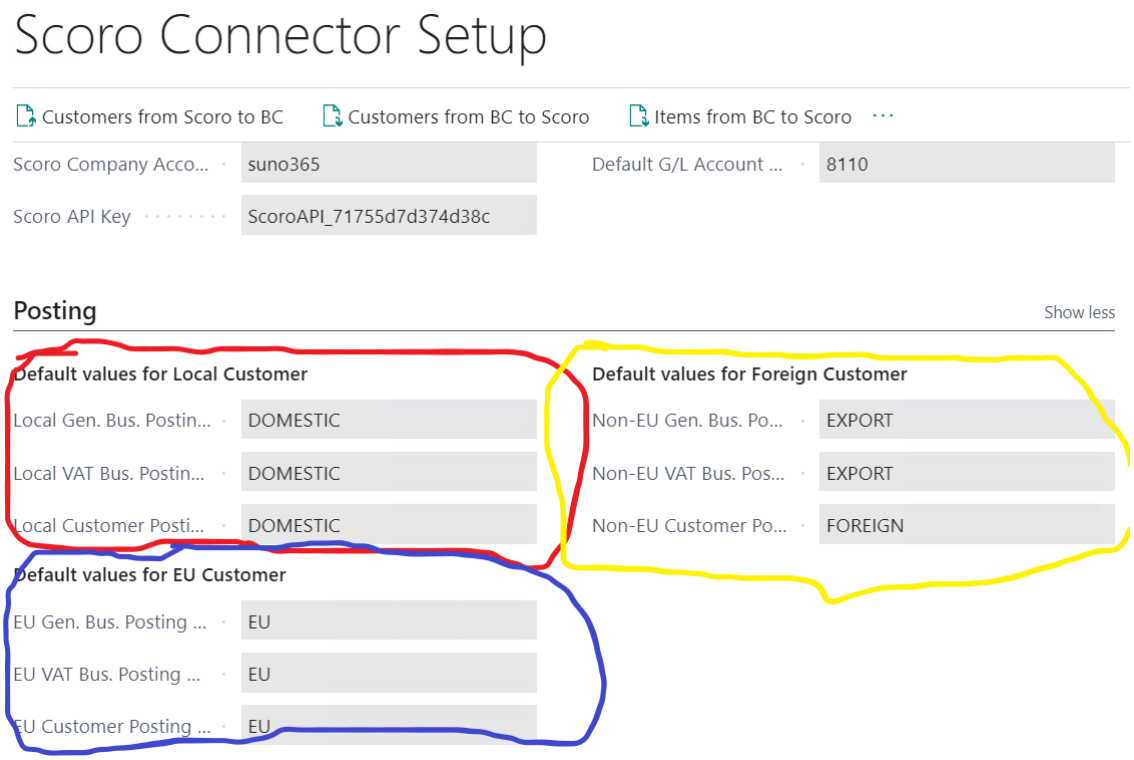
```
2 references
local procedure GetCountryISOCodeFromScoroCountryCode(ContactJson: JsonObject): Code[10]
var
    CountryRecord: Record "Country/Region";
    ScoroCountryCode: Code[10];
begin
    ScoroCountryCode := GetFieldFromContactJson(ContactJson, 'country', 'address');
    CountryRecord.SetRange(CountryRecord."CPC-SC Scoro Code", ScoroCountryCode, ScoroCountryCode);
    if CountryRecord.Find('-') then
        exit(CountryRecord."ISO Code");
end;
```

Joonis 39. "*GetCountryISOCodeFromScoroCountryCode*" funktsioon, mis eraldab JSON-ist kolmekohalise riigi koodi ja leiab vastava kahekohalise koodi riikide tabelist.

Viimane arendusloogika osa koodiühikus "*ScoroToBCCustomers*" puudutab konteerimisgruppe. Konkreetse kliendiga on seotud 3 konteerimisgruppi: üldine (*General Business Posting Group*), käibemaksu oma (*VAT Business Posting Group*) ja kliendi oma (*Customer Business Posting Group*). Iga grupp on seotud samanimelise BC tabeliga, milles on igaühes kolm kirjet. Esimene kirje on kõikides tabelites selle jaoks, kui sissetulev kontakt on samast riigist kui BC keskkonda kasutav ettevõte ise. Teine kirje on selle jaoks kui sissetulev kontakt on Euroopa Liidu riik ning kolmas selle jaoks kui see

pole kumbagi eelnevatest. Kokkuvõttes on seega kolmes konteeringugrupi tabelis kokku üheksa kirjet: lokaalne üldine, lokaalne käibemaksu oma, lokaalne kliendi oma (Joonis 40 punane), EL üldine, EL käibemaksu oma, EL kliendi oma (Joonis 40 sinine), EL-väline üldine, EL-väline käibemaksu oma ja EL-väline kliendi oma (Joonis 40 kollane). Igal keskkonda kasutaval ettevõttel on need väärtused erinevad ning seetõttu on need väärtused vaja seadistada Scoro *Connector Setup* tabelisse.

Konteerimisgruppidega tegeleb funktsioon "*GetPostingGroup*". See võtab parameetrina sisse Scorost saadud *JSON* objekti ja grupi tüübi, üldine, käibemaks või klient (Joonis 41). Uue kontakti lisamisel klientide tabelisse leitakse eelnevalt mainitud "*GetCountryISOCodeFromScoroCountryCode*" funktsiooniga kontakti objektist riigi kood (Joonis 42 punane). Kui see riigi kood vastab BC keskkonda kasutavale riigi koodile (Joonis 42 sinine), kutsub funktsioon välja funktsiooni "*GetLocalPostingGroup*" (Joonis 42 roheline) ning annab argumendina kaasa parameetrina saadud konteeringugrupi tüübi. Kui riigi kood on riikide tabelis EL riikide hulgas, siis kutsutakse välja funktsioon "*GetEUPostingGroup*" (Joonis 42 kollane) ning antakse samuti argumendina kaasa grupi tüüp. Viimane võimalus on see, et riik pole ei kohalik ega EL riik ning siis kutsutakse grupi tüübiga välja funktsioon "GetNonEUPostingGroup" (Joonis 42 valge).



Joonis 40. Konteerimisgruppide väljad "Scoro *Connector Setup*" lehel.

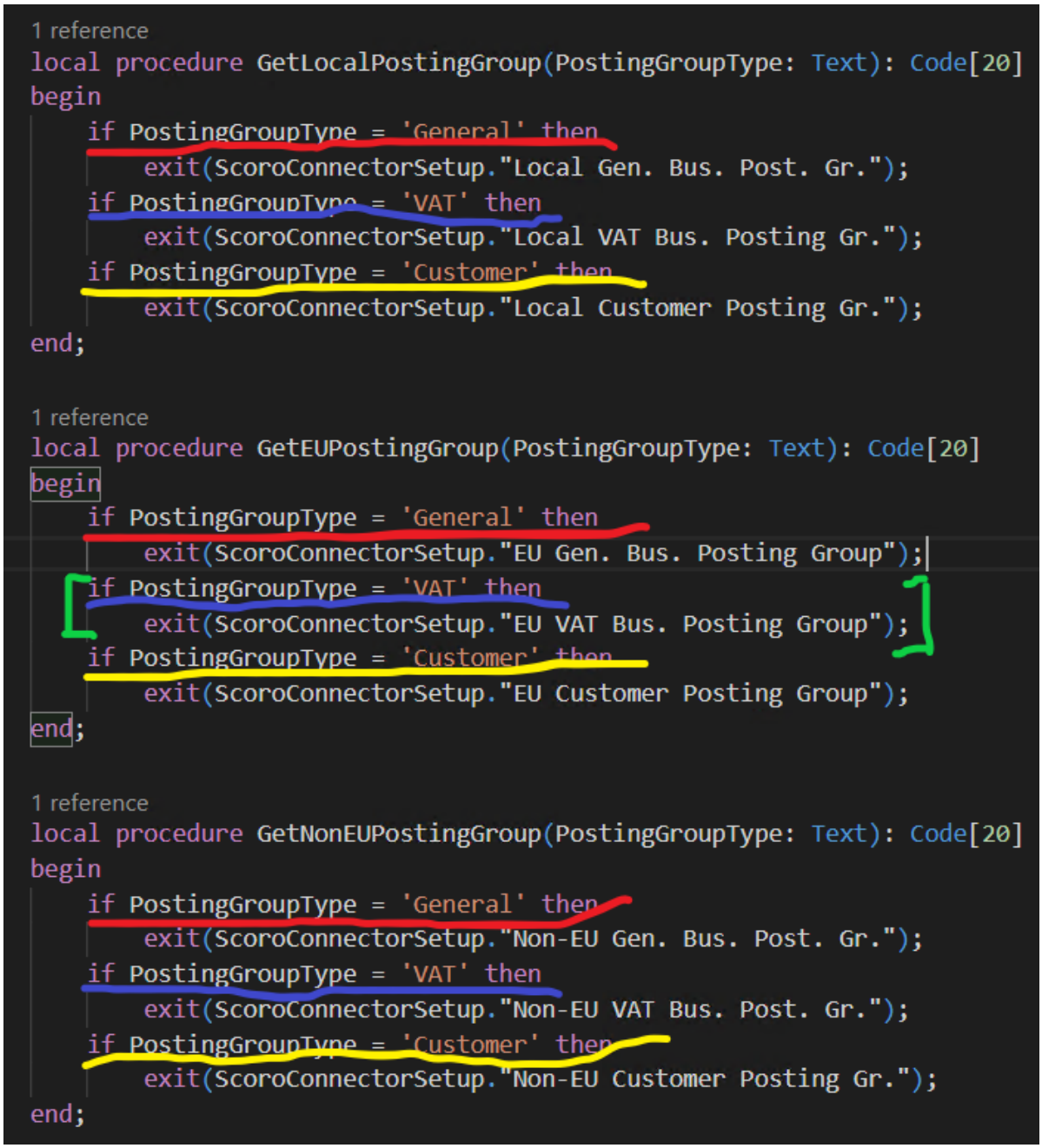
```
CustomerRecord.Validate("Gen. Bus. Posting Group", GetPostingGroup(ContactJson, 'General'));
CustomerRecord.Validate("VAT Bus. Posting Group", GetPostingGroup(ContactJson, 'VAT'));
CustomerRecord.Validate("Customer Posting Group", GetPostingGroup(ContactJson, 'Customer'));
```

Joonis 41. Funktsiooni "*GetPostingGroup*" välja kutsumine.

```
3 references
local procedure GetPostingGroup(ContactJson: JsonObject; PostingGroupType: Text): Code[20]
var
    CompanyRecord: Record "Company Information";
    ScoroCountryISOCode: Code[10];
    CountryRecord: Record "Country/Region";
begin
    ScoroCountryISOCode := GetCountryISOCodeFromScoroCountryCode(ContactJson);
    if ScoroCountryISOCode = '' then begin
        exit('');
    end;
    if CompanyRecord.Find('-') then begin
        if ScoroCountryISOCode = CompanyRecord."Country/Region Code" then
            exit(GetLocalPostingGroup(PostingGroupType));
    end;
    if CountryRecord.IsEUCountry(ScoroCountryISOCode) then
        exit(GetEUPostingGroup(PostingGroupType))
    else
        exit(GetNonEUPostingGroup(PostingGroupType));
end;
```

Joonis 42. Funktsioon "*GetPostingGroup*", mis sisaldab leiab riigi koodi alusel konteerimisgrupi.

Kõik 3 funktsiooni teevad väga sarnast asja, kuid tegelevad erinevate kirjetega. Nad vaatavad parameetrina saadud grupi tüüpi (Joonis 43 üldised punane, käibemaks sinine, klient kollane) ja väljastavad siis vastava tabeli "Scoro *Connector Setupi*" välja. Näiteks kui meetod "*GetEUPostingGroup*" saab parameetrina sisse grupi tüübi "*VAT*", väljastab ta kasutaja poolt välja *"EU VAT Bus. Posting Group"* sisestatud andmed (Joonis 43 roheline).



Joonis 43. "*GetPostingGroup*" funktsiooni abifunktsioonid "*GetLocalPostingGroup*", "*GetEUPostingGroup*" ja "*GetNonEUPostingGroup*", mis väljastavad vastava Scoro Connector Setup tabelisse kirjutatud välja.

3.1.5 Testid

Microsofti nõue AppSource'i publitseerimisel on, et rakendusel peab olema vähemalt üks automaattest. Test asub põhirakendusest "*TalTechAppScoro*" eraldi rakenduses "*TalTechAppScoroTest*", milles on kaks koodiühikut: "*ScoroAppTest Install"* ja

"ScoroApp Setup Test" (Joonis 44). Esimene tegeleb keskkonna ülesseadmise ja põhirakendusest andmete saamisega.

```
codeunit 24013590 "CPC-SC ScoroAppTest Install"
{
    Subtype = Install;

    trigger OnRun()
    begin
        // https://www.kauffmann.nl/2018/11/05/outbound-http-calls-blocked-in-business-central-sandbox/
    end;

    trigger OnInstallAppPerCompany()
    var
        Setup: Record "Company Information";
    begin
        // Code for company related operations
    end;

    trigger OnInstallAppPerDatabase()
    var
        appInfo: ModuleInfo;
    begin
        // Code for database related operations
        NavApp.GetCurrentModuleInfo(appInfo);
        if appInfo.DataVersion = Version.Create(0, 0, 0, 0) then
            HandleFreshInstall // 'DataVersion' of 0.0.0.0 indicates a 'fresh/new' install
        else
            HandleReinstall;
    end;

    3 references
    procedure EnableHttpClient()
    var
        AppSetting: Record "NAV App Setting";
        AppInfo: ModuleInfo;
    begin
        AppSetting."App ID" := AppInfo.Id();
        AppSetting."Allow HttpClient Requests" := true;
        if not AppSetting.Insert() then begin
            AppSetting.Modify();
        end;
    end;

    1 reference
    local procedure HandleFreshInstall();
    begin
        // Do work needed the first time this extension is ever installed for this tenant.
        EnableHttpClient();
    end;

    1 reference
    local procedure HandleReinstall();
    begin
        // Do work needed when reinstalling the same version of this extension back on this tenant.
        EnableHttpClient();
    end;

    var
        0 references
        UnitTest: Codeunit "CPC-SC ScoroApp Setup Test";
}
```

Joonis 44. *"ScoroAppTest Install"* testkoodiühik.

Teine koodiühik *"ScoroApp Setup Test"* tegeleb rakenduse reaalse testimisega, installeerimisühikust saadud andmeid ja põhirakenduse funktsioone kasutades. See seab esmalt üles integratsiooni Scoro *API* ja *HTTP* kliendiga funktsioonis "*SetupScoro*" (Joonis 45 punane). Kui rakendust testida, käivituvad testkoodis *handler*'id (Joonis 45 roheline), mis kutsuvad välja funktsiooni "*TestSetupPage*" (Joonis 45 kollane). See teeb sisuliselt automaatselt läbi ühe teststsenaariumi. Funktsioon avab "Scoro *Connector Setup*" lehe (Joonis 47) ja seab sünkroniseerimiseks vajalikud väärtused *API* võtme, *URL* ja ettevõtte ID väljade kohale (Joonis 46 punane). Siis avab see kliendi lisamise lehe ja täidab ühe kliendi Business Centralist Scorosse toomiseks vajalikud nime ja asukoha väljad (Joonis 46 sinine). Seejärel avab funktsioon taas "*ScoroConnectorSetup*" lehe ja käivitab *"Customers from* BC to Scoro" *action*'i, mis sünkroniseerib vastava kliendi (Joonis 46 kollane). Viimaks kontrollib see, kas ületoomine oli edukas uue kliendi Scoro ID abil (Joonis 46 roheline).

```
codeunit 24013595 "CPC-SC ScoroApp Setup Test"
{
    // [FEATURE] [Scoro Setup]

    Subtype = Test;

    var
        1 reference
        TestInstall: Codeunit "CPC-SC ScoroAppTest Install";
        1 reference
        Assert: Codeunit "Library Assert";

    trigger OnRun()
    begin
        SetupScoro();
        TestInstall.EnableHttpClient();
    end;

    [Test]
    [HandlerFunctions('ResponseHandler,TemplateHandler')]
    0 references
    procedure TestScoroSetup()
    var
    begin
        TestSetupPage();
    end;

    1 reference
    local procedure SetupScoro()
    var
        ScoroInstall: Codeunit "CPC-SC Scoro Install";
    begin
        ScoroInstall.SetupScoroIntegration();
        ScoroInstall.EnableHttpClient();
    end;
```

Joonis 45. *"ScoroApp Setup Test"* testkoodiühik.

```
local procedure TestSetupPage()
var
    ScoroSetupPage: TestPage "CPC-SC Scoro Connector Setup";
    CustomerCardPage: TestPage "Customer Card";
    bScoroId: Boolean;
begin
    ScoroSetupPage.OpenEdit();
    ScoroSetupPage."Scoro Company Account ID".SetValue('suno365');
    ScoroSetupPage."Scoro API Service Url".SetValue('https://suno365.scoro.ee/api/v2/');
    ScoroSetupPage."Scoro API Key".SetValue('ScoroAPI_71755d7d374d38c');
    ScoroSetupPage.Close();

    CustomerCardPage.OpenNew();
    CustomerCardPage.Name.SetValue('Tallink Grupp AS');
    CustomerCardPage."Country/Region Code".SetValue('EE');

    ScoroSetupPage.OpenView();
    ScoroSetupPage."Customers from BC to Scoro".Invoke();
    ScoroSetupPage.Close();

    bScoroId := CustomerCardPage."CPC-SC Scoro Id".Value() <> '';

    Assert.AreEqual(true, bScoroId, 'Scoro ID received');
end;
```

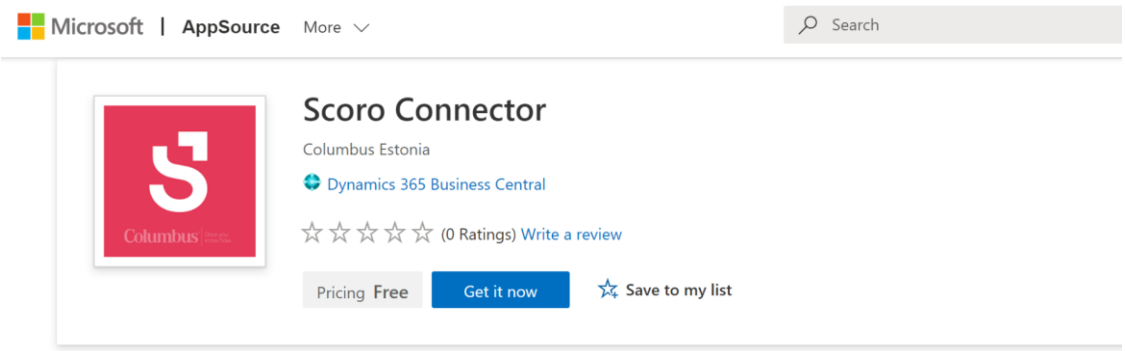
Joonis 46. "*TestSetupPage*" funktsioon "*ScoroApp Test Setup"* ühikus.

```
[StrMenuHandler]
0 references
procedure RequestHandler(Option: Text[1024]; var Choice: Integer; Instruction: Text[1024])
var
begin
    // menu asks user to allow or not to make web service connection
    Choice := 2; //allow once
end;

[MessageHandler]
0 references
procedure ResponseHandler(Msg: Text[1024])
var
begin
    if Msg <> '' then begin
        // ok
    end;
end;

[ModalPageHandler]
0 references
procedure TemplateHandler(var MiniTemplate: TestPage "Config Templates")
var
    Template: Text;
begin
    if MiniTemplate.First() then begin
        Template := MiniTemplate."Template Name".Value();
        MiniTemplate.OK().Invoke();
    end;
end;
```

Joonis 47. Handler'id "ScoroApp Test Setup" ühikus.

3.2 AppSource

Järgnevalt saab ülevaate AppSource'i jaoks valminud dokumentidest ja nõuetes. Nõutavate dokumentide loomiseks soovitasid juhendajad uurida kõigepealt AppSource'is olemasolevat Envoice ning Intra rakendusi. Lähtudes ettevõtte sisendist ning väljatoodud näidetest said tehtud vajalikud failid publitseerimiseks. Ka sai loodud test stsenaarium rakenduse valideerimiseks (Lisa 2). Nüüdseks on Scoro Connector rakendus AppSource keskkonnas (Joonis 48) kõigile vabalt allalaadimiseks olemas [14].



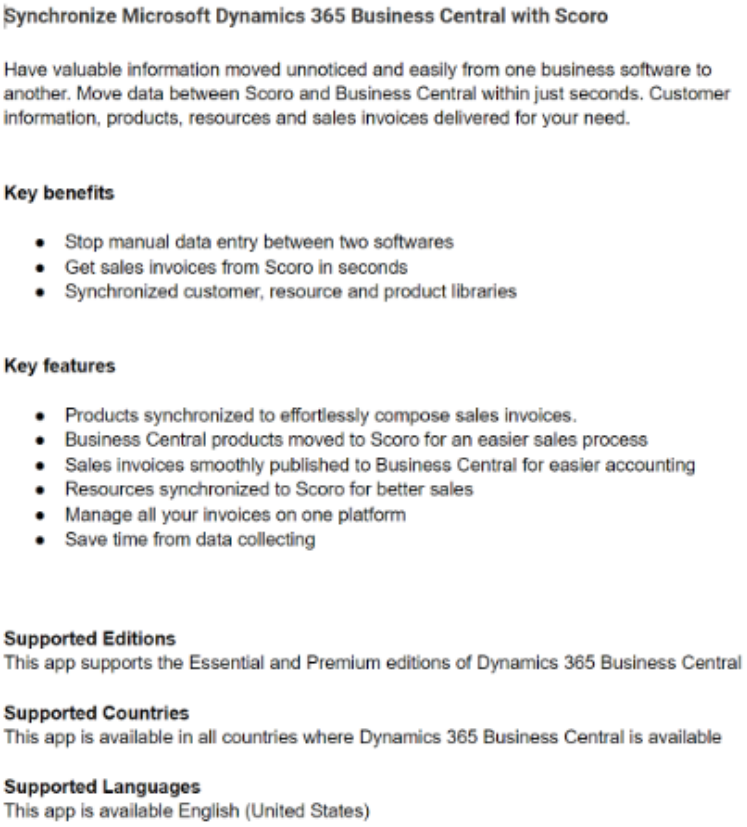
Joonis 48. Scoro Connector edukalt Appsource publitseeritud.

Logo valmis Adobe Illustratoris. Võttes eeskuju Columbuse projekti Envoice logost ning stiili säilitamiseks otsustati ka ühevärvilise ning esitähelise logo kasuks. Logo värvid ning esitähe sümbol on Scoro ametlikult logolt (Joonis 49). Sümboliseerimaks, et toode on Columbuse oma on sümboli all Columbuse logo. Logo oli tarvis väiksel kujul 48 x 48 pikslit ja suurel kujul 216 x 216 pikslit.

Joonis 49. Scoro Connectorile loodud logo.

Kui toode avada AppSource'is, siis sealt saab potentsiaalne klient lisainfot lugeda. Lodud toote kirjeldamiseks oli vaja kirjutada ingliskeelne tootekirjeldus, tuues välja toote

kasutamise peamised eelised ning kasulikud küljed (Joonis 50). Lehele lisamiseks oli vaja ka genereerida lihtsustatud HTML kood.



Synchronize Microsoft Dynamics 365 Business Central with Scoro

Have valuable information moved unnoticed and easily from one business software to another. Move data between Scoro and Business Central within just seconds. Customer information, products, resources and sales invoices delivered for your need.

Key benefits

- Stop manual data entry between two softwares
- Get sales invoices from Scoro in seconds
- Synchronized customer, resource and product libraries

Key features

- Products synchronized to effortlessly compose sales invoices.
- Business Central products moved to Scoro for an easier sales process
- Sales invoices smoothly published to Business Central for easier accounting
- Resources synchronized to Scoro for better sales
- Manage all your invoices on one platform
- Save time from data collecting

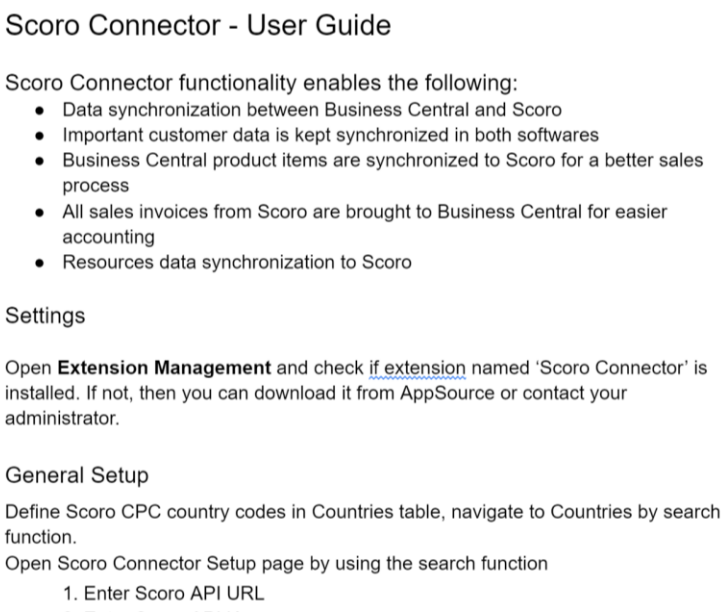
Supported Editions
This app supports the Essential and Premium editions of Dynamics 365 Business Central

Supported Countries
This app is available in all countries where Dynamics 365 Business Central is available

Supported Languages
This app is available English (United States)

Joonis 50. Müügilaused toote võtme funktsioonid ja potentsiaalse kliendi kasum.

Kasutusjuhend tutvustab esmalt põgusalt Scoro Connector laienduse põhilisi funktsionaalsuseid, seejärel aga annab samm-sammulisi juhiseid laienduse installeerimiseks ja algseadistuste tegemiseks. Lisaks ka juhised rakenduse kasutamiseks (Joonis 51).



Scoro Connector - User Guide

Scoro Connector functionality enables the following:

- Data synchronization between Business Central and Scoro
- Important customer data is kept synchronized in both softwares
- Business Central product items are synchronized to Scoro for a better sales process
- All sales invoices from Scoro are brought to Business Central for easier accounting
- Resources data synchronization to Scoro

Settings

Open **Extension Management** and check if extension named 'Scoro Connector' is installed. If not, then you can download it from AppSource or contact your administrator.

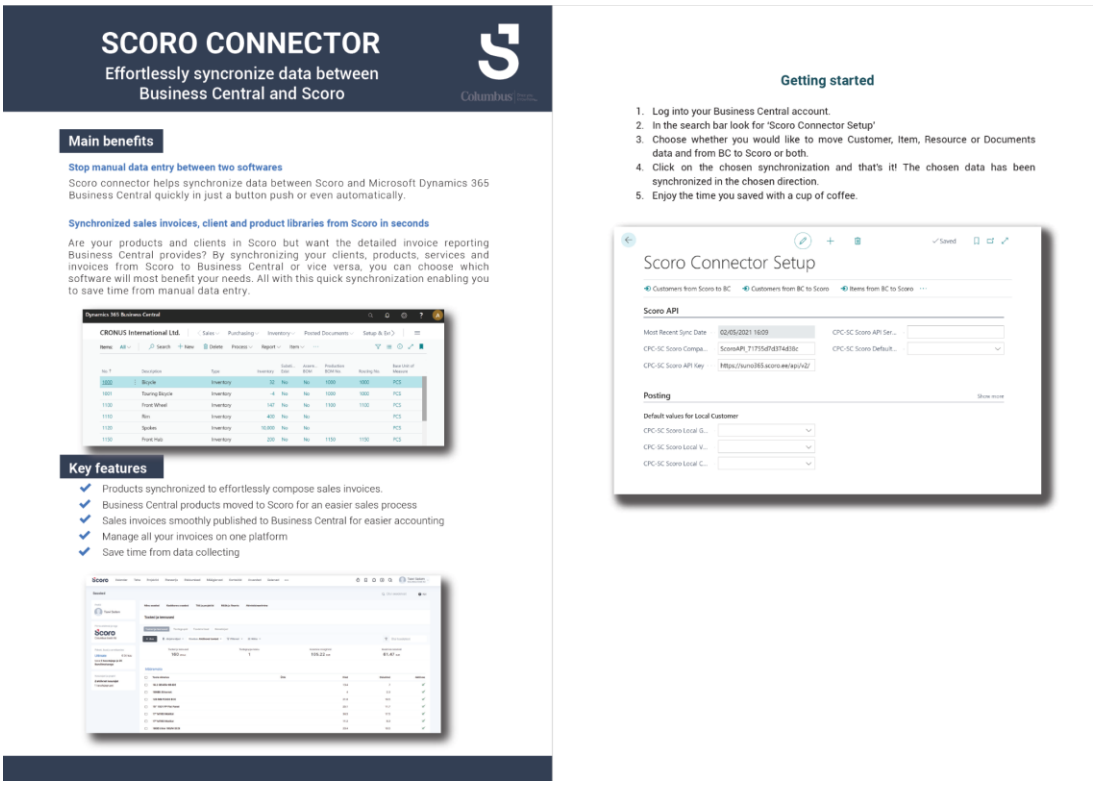
General Setup

Define Scoro CPC country codes in Countries table, navigate to Countries by search function.
Open Scoro Connector Setup page by using the search function

1. Enter Scoro API URL

Joonis 51. Scoro Connector kasutusjuhend.

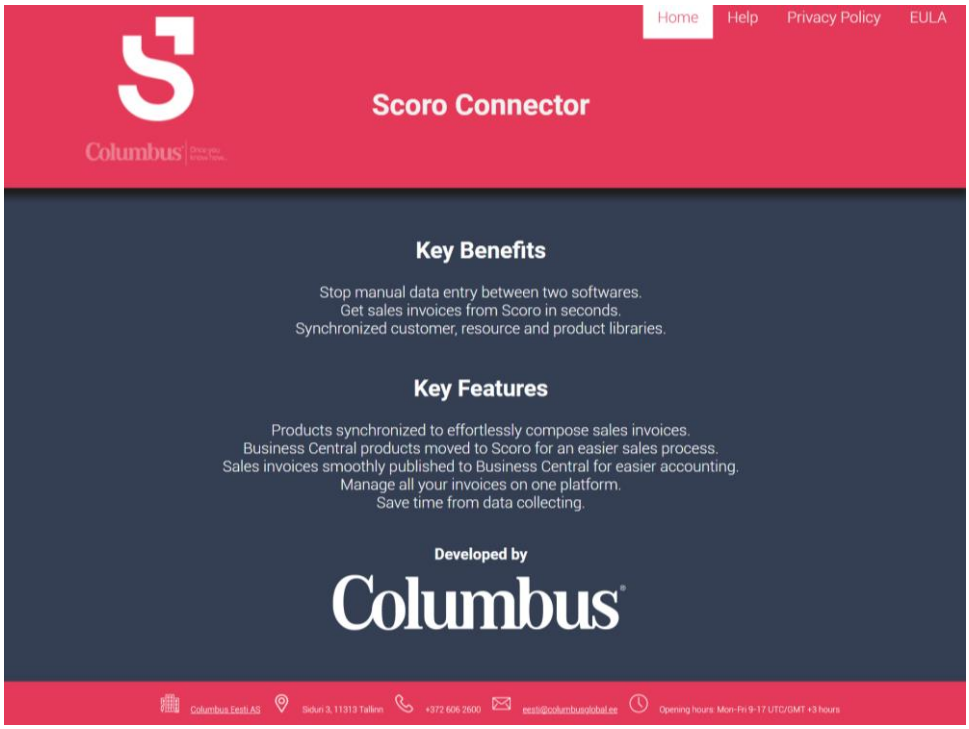
Valminud *fact sheet* on toote turunduslik leht, mis annab potentsiaalsele kliendile kiire ülevaate põhivõimekustest ning kuidas alustada (Joonis 52). Ettevõte andis sisendit, mida *fact sheet* endas sisaldada võiks.



Joonis 52. Valminud tooteleht.

AppSource'i on ka vaja panna privaatsuspoliis ja lõppkasutaja litsentsileping (*End User License Agreement*). Need soovis ettevõtte teha veebilehe kaudu kättesaadavaks. Ka läheb veebilehele ettevõtte kontaktandmed ning kasutusjuhend. Kuna lihtne HTML (Joonis 53) polnud esinduslik, tegi tiim veebilehele disaini (Joonis 54).

Joonis 53. Pealeht enne disaini.



Joonis 54. Pealeht peale disaini.

Koodi kirjutamisel võeti eeskuju internetis tasuta ligipääsuga *CSS* ja *HTML* keeltes kujundatud lehe mallidelt ning *w3schools.com* lehel olevatest õpetustest. Veebileht koosneb neljast *HTMLis* kirjutatud lehest, mis kasutavad ühist *CSS* keeles *style.css* stiilifaili.

Piltide, menüü ja jaluse puhul on kasutuses kood, mis jälgib ekraanisuurust ning muudab piltide ja jaluse suurust vastavalt veebilehe akna suurusele. Menüü muutub väikse akna puhul hamburger menüüks. Veebikood koosneb neljast failist. *Index.htm*, mis on 65 rida. *Help.htm*, mis on 75 rida. *style.css*, mis on 90 rida. Ja üks 9-realine *Javascript* funktsioon.

```
<html>
     <head>
          <link rel="stylesheet" href="style.css">
          <meta    name="viewport"    content="width=device-
width,
           initial-scale=1.0">
          <script src="index.js"></script>
          <title>Scoro Connector</title>
     </head>
     <body>
          <header>
               <link                        rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
               <div class="row">
                    <div class="column">
                         <img     src="help_files/logo.png"
class="logo"/>
                    </div>
```

Joonis 55. Näidis koodijupp pealehe *HTMList*.

4. Analüüs ja järeldused

Antud peatükis analüüsime valminud funktsionaalsusi ning põhjendame nende teostusviisi. Teeme põgusa ülevaate projektiga seotud kirjandusest, esitame teostatud tööde logi ja anname hinnangu projekti teostamise protsessi kohta.

4.1 Tehnilise teostuse põhjendus

Käesoleva tarkvara projekti äriloogika ja sisuline tegevus lähtus Columbus Eesti AS poolt püstitatud ülesandest, milleks oli kahe tarkvara liidestamine. Sellest tulenevalt olid etteantud ka tehnoloogiad ja programmeerimiskeeled, mida need kaks tarkvara kasutasid. Tehnilise teostuse dokumentatsioon ja nõuded valmisid projekti konsultantide poolt koostöös ettevõttepoolsete juhendajatega.

4.1.1 Nõuded

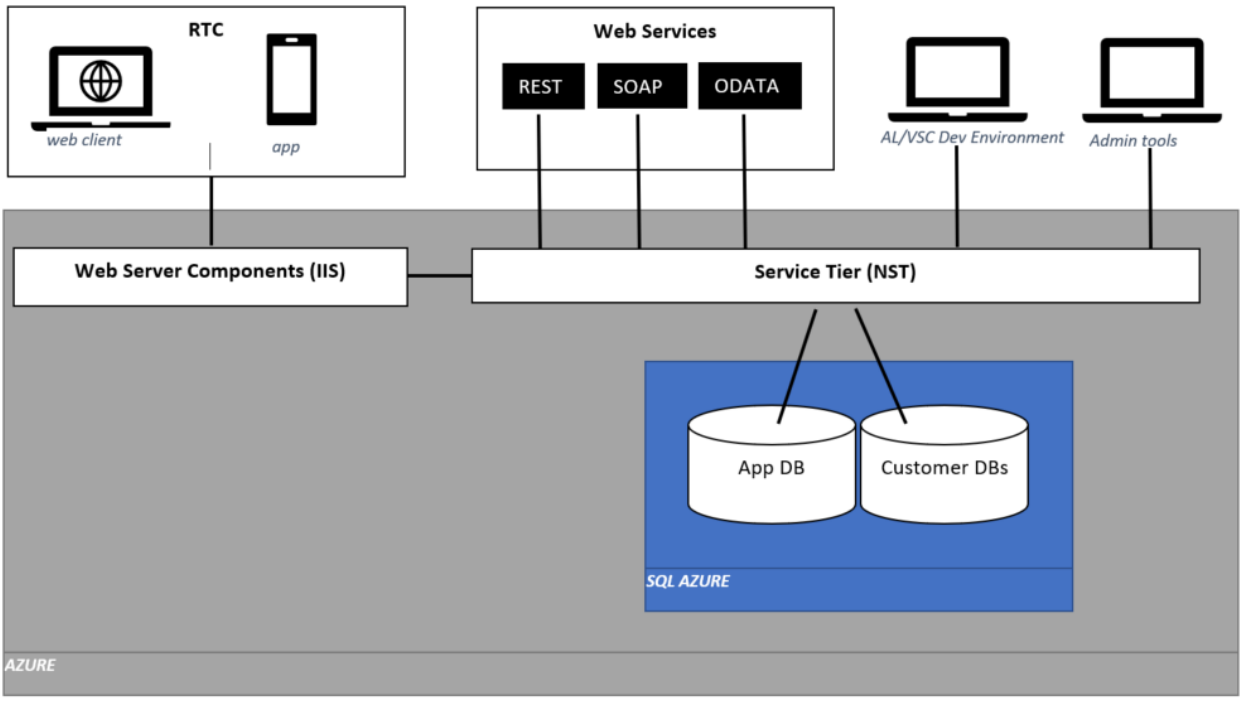
Nõuded tulenesid olukorrast, kus mõni ettevõtte kasutab oma äriprotsesside haldamiseks majandustarkvara, milleks on Microsoft Dynamics 365 Business Central, aga samas toimub osa tema müügiprotsessist teises ärihaldustarkvaras nagu Scoro. Selleks, et ettevõttel oleks terviklik pilt kogu tema müügiarvetest, toodetest, teenustest ja klientidest, on vaja hoida samasuguseid andmeid mõlemas keskkonnas. Ära hoidmaks andmete käsitsi sisestamist, kaardistati konsultantide poolt vajalikud nõuded andmetele, mida on vaja Business Centrali keskkonnas arvete koostamiseks, klientide loomiseks, et ettevõtte teistes äriprotsessides oleksid olemas vajalikud andmed, mis tulenevad klientide, toodete, arvete andmemudelitest.

Andmemudelite nõuded töötati välja koostöös ettevõtte juhendajatega, kust saime sisendit, millised on äriprotsesside toimimiseks vajalikud andmed. Andmete sünkroniseerimise loogika töötati välja meeskonnaliikmete ja Columbus Eesti AS vahel töö käigus.

Eraldi kõiki nõudeid tulemustes välja ei hakatud tooma, kuna nende reprodutseerimine oleks liiga mahukas. Enamus nõuetest on juba kirjeldatud meeskonna projekti raames.

4.1.2 Arhitektuur

Microsoft Dynamics 365 Business Central on pilvetehnoloogial põhinev rakendus. Interneti informatsiooni teenuste lehekülge on vaja, et anda vajalik ligipääs Business Centrali veebikliendile (*Web Server Component*), mis haldab faile, et pakkuda teenuseid ja sisu klientidele üle interneti. Seesama Business Centrali veebiklient on ühendatud teenuste kihiga (NST), mis omakorda pakub ühenduvust andmebaasiga, mis sisaldab rakendust ja äriandmeid (Joonis 56) [15].



Joonis 56. Business Centrali arhitektuur, võetud Microsofti dokumentatsioonist.

Business Centrali ülesehitus on, et sinna on enda vajadustele põhinevalt võimalik juurde arendada laiendusi. Funktsionaalsus Business Centralis on kirjutatud objektidena. Selleks, et modifitseerida olemasolevat objekti nagu näiteks tabel või leht, tuleb teha selleks uus laiendus. Sõltuvalt laienduse vajadusest on võimalik olemasolevaid objekte kas laiendada või teha täielikult uued. Meie näitel oli meil vajalik laiendada juba olemasolevaid objekte nagu näiteks riikide tabel, aga samas luua ka uusi objekte nagu *Scoro Connector Setup Page*. Kõiki objekte hoiustatakse koodina, milleks Business Centrali puhul on *Application Langugage*. Kõiki objekte hoitakse eraldi failis, mille

laiendiks on *.al*. Hiljem kõik objektid pakitakse kokku ja nendest valmib rakenduse faili pakett, mida on võimalik siis juba AppSource keskkonda laadida, et laiendus oleks kõigile kasutatav [15].

Application Langugage (AL) on programmeerimiskeel, mida kasutatakse koodi kirjutamiseks Business Centralis. Koodi ja funktsioonidega on võimalik lugeda, kirjutada ja muuta andmeid. AL keelega on võimalik kontrollida objektide implementatsioone ja suhelda kasutajaga [7].

Võimalikud alternatiivid teistsuguse arhitektuuri rakendamiseks puudusid, kuna Business Centralis on laienduste tegemiseks see kindlalt paika määratud. Samuti poleks võimalik kasutada ka mingit teistsugust programmeerimiskeelt.

4.1.3 Disain

Laienduse disain tuleneb Microsoft Dynamics 365 Business Centrali enda disainist, seepärast ei tehtud ka eraldi disainianalüüsi. Arendaja sai kasutada juba olemasolevaid komponente, millel on kindel disain ja neid muuta ei ole võimalik. Selleks, et sünkroniseerimislaiendus oleks kasutatav ja seadistatav, loodi vastava sisuga seadistamise lehekülg Scoro *Connector Setup Page*. Selle valmimisel järgiti Business Centrali teiste lehekülgede komponentide paigutust, et lehekülg näeks esteetiliselt sarnane välja kogu ülejäänud rakenduse välimusega.

4.1.4 Kood

Arenduse käigus üritasime järgida puhta koodi ja objektorienteeritud programmeerimise printsiipe. Puhas kood on üldlevinud heade programmeerimistavade raamistik, mille ühendas esimesena selle nime all Robert C. Martin oma raamatus “Clean Code: A Handbook of Agile Software Craftmanship” [10]. Üks oluline printsiip, mida meie siinkohal järgisime oli DRY ehk “*don’t repeat yourself*”. Koodiühikud on kirjutatud nii, et meetodites olev loogika ei kordu. Iga funktsioon teeb võimalusel vaid ühte asja ning tekib “väljakutsumise ahel”, kus lõpus käivitatakse vaid ühe meetodi abil kogu ülejäänud loogika. Funktsioonid on lühikesed ja nendes olevat loogikat on võimalik vajadusel lihtsasti välja vahetada ilma ülejäänud koodi muutmata, mis on puhta koodi peamiseid eesmärke. Puhtas koodis on oluline ka sisemise struktuuri peitmine, mille tarbes on enamik meetodeid lokaalsed ja neid ei saa väljaspool originaalset ühikut välja kutsuda.

Sellega väldime koodi kasutamise võimalust ebasobilikul viisil. Väldime ka kommentaare ning eelistame koodi selgitavaid ja lihtsaid nimesid nii ühikute, funktsioonide kui muutujate tasemel. Eesmärk on, et ka inimesel, kellel pole AL keele eriärasustega varasemat kogemust, oleks võimalik koodi põhimõttest aru saada. Funktsioonide nimed (nagu kõik funktsioonid BC-s) on “PascalCase” nimetamisstiilis.

Koodi kirjutamisel tuli ka jälgida, et poleks liigselt globaalseid muutujuaid, sest pärast publitseerimist neid enam muuta ei saa. Saab ainult ülekirjutada või muuta see funktsioon vananenuks ning uus lisada. Lokaalseid funktsioone saaks järgmist versiooni tehes muuta ning seetõttu ka mitmed me funktsioonid on lokaalsed. Rakenduse paketid mida me AppSource keskkonda publitseerime oli vaja enne üleslaadimist allkirjastada Columbuse Eesti AS sertifikaadiga. Allkiri on vajalik selles, et rakenduse kasutajad saaksid kindlad olla, et tegemist ei ole pahavaraga, lisaks peavad kõik AppSource üleslaetud rakendused olema sertifikaadiga allkirjastatud.

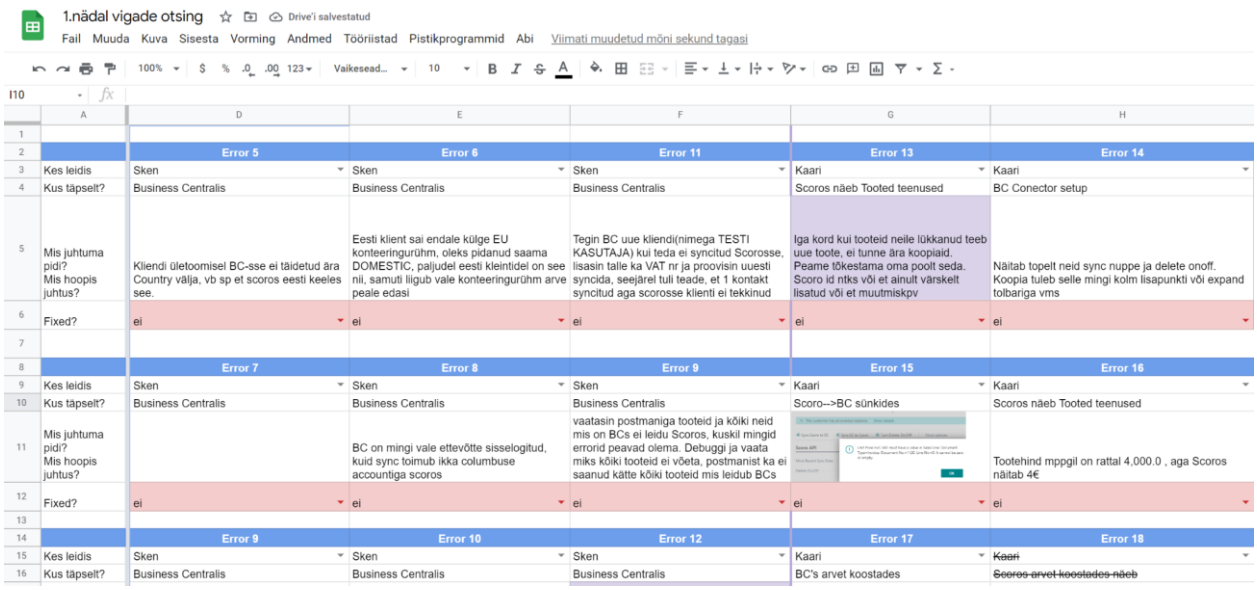
4.1.5 Testid

Microsofti nõude kohaselt peab olema AppSource laetud rakendusele kaasa pandud eraldi automaattestirakendus. Automaattestirakendus peab nõude kohaselt testima vähemalt ühte põhilist rakenduse kasutusjuhtu. Meeskond otsustas, et automaattesti- rakenduse test stsenaariumiks on kliendi sünkroniseerimine, kuna selle arendus oli küllaltki mahukas. Testrakendus teeb sünkroniseerimiseks vajalikud algandmete seadistused, mis sisaldab endas API ühenduse loomiseks vajalike andmete seadistamist. See loob uue testkliendi ning seejärel sünkroniseerib selle Scoro keskkonda, saades Scoro API-st vastuse uue loodud kliendi kohta. Tihtipeale võib testrakenduse loomine olla kordades mahukam, kui põhirakenduse loomine. Seepärast otsustasime meeskonnaga, et automaattesti kirjutame me ainult ühe põhikasutusjuhu kohta, kuna põhirakenduse arenduse peale kulub enamus töömahust. Ülejäänud kasutusjuhud ja kogu rakenduse loogika on läbi testitud mehaaniliselt läbi kasutajaliidese, kõigi meeskonnaliikmete poolt ja Columbus Eesti AS juhendajate poolt.

Automaattesti rakendus on vajalik selleks, et esmakordsel AppSource keskkonda rakenduse lisamisel käivitatakse testid. Nende tulemusest sõltub, kas rakendust on võimalik publitseerida või mitte. Iga Business Centrali uue versiooniga, mis Microsoft välja tuleb, käivitatakse Microsofti poolt automaatselt kõik testrakendused, mis Business

Centrali laienduste jaoks on loodud. Testi tulemustest saab rakenduse looja alati teavituse, kas tema testid läksid läbi või mitte. Juhul kui test rakendus saab negatiivse tulemuse testi jooksutamisel Microsofti poolt, rakendust AppSource keskkonda ei publitseerita [9].

Projekti alguses oli vaja läbi käia kõik võimalikud kasutusjuhud ning leida poolikud kohad, mida on vaja selle lõputöö raames lõpuni viia. Vigade kirjapanekuks võtsime inspiratsiooni aine2 “ITB1708 IT juhtimise ja ülalhoiu alused” tutvustatud juhtumi teavituse pileti vormist. Kõik tiimiliikmed võtsid ette BC ning katsetasid läbi klientide, arvete, toodete, resursside loomise ning proovisid siis sünkroniseerida Scorosse ja vastupidi. Tekkinud errorid kirjutasime tabelisse (Joonis 57), et Henrik teaks parandada. Lisaks saime koodi puhtuse ning BC standardi jälgimiseks ettevõtte poolselt juhendajalt nimekirja kohti, mida tarvis koodis puhastada.



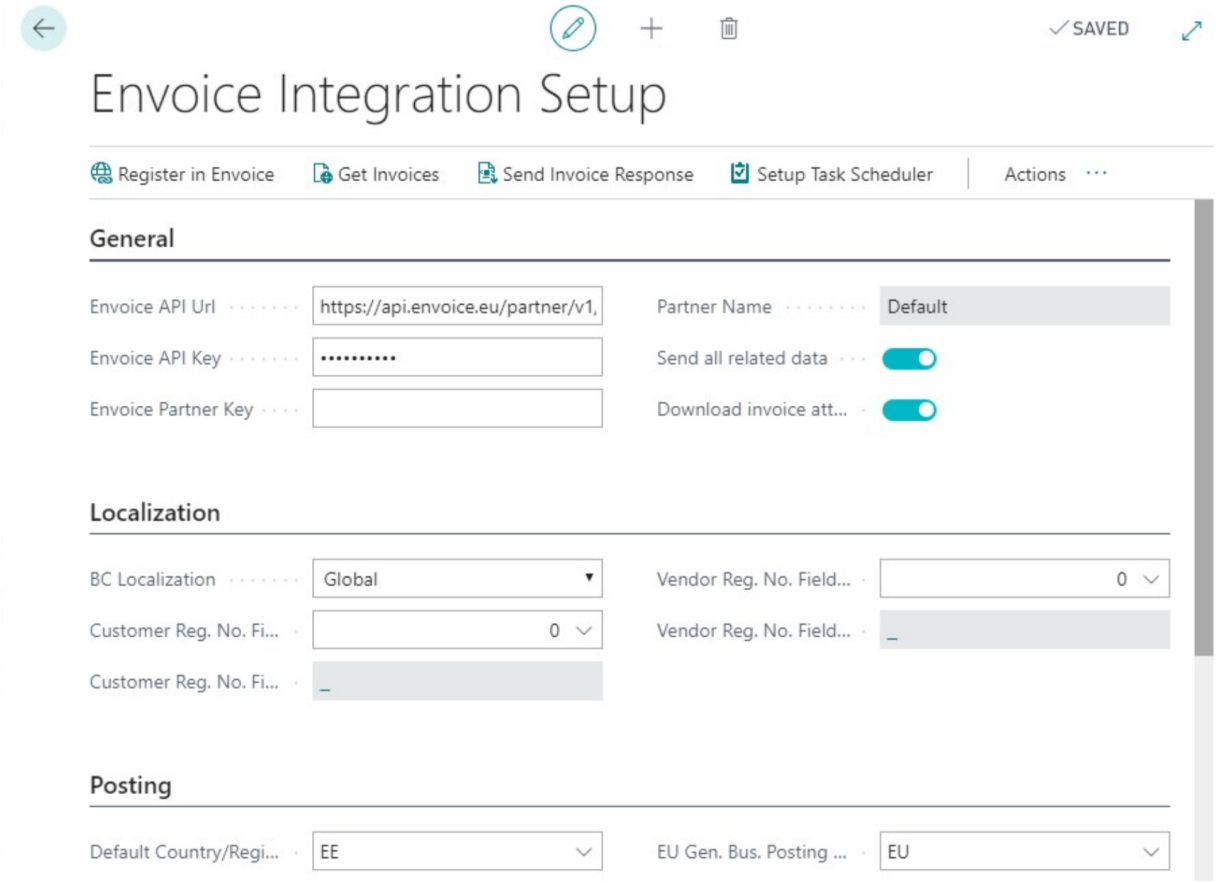
		Error 5	Error 6	Error 11	Error 13	Error 14
Kes leidis		Sken	Sken	Sken	Kaari	Kaari
Kus täpselt?		Business Centralis	Business Centralis	Business Centralis	Scoros näeb Tooted teenused	BC Conector setup
Mis juhtuma pidi? Mis hoopis juhtus?		Kliendi ületoomisel BC-sse ei täidetud ära Country välja, vb sp et scoros eesti keeles see.	Eesti klient sai endale külge EU konteeringurühm, oleks pidanud saama DOMESTIC, paljudel eesti kleintidel on see nii, samuti liigub vale konteeringurühm arve peale edasi	Tegin BC uue kliendi(nimega TESTI KASUTAJA) kui teda ei syncitud Scorosse, lisasin talle ka VAT nr ja proovisin uuesti syncida, seejärel tuli teade, et 1 kontakt syncitud aga scorosse klienti ei tekkinud	Iga kord kui tooteid neile lükkanud teeb uue toote, ei tunne ära koopiaid. Peame tõkestama oma poolt seda. Scoro id ntks või et ainult värskelt lisatud või et muutmiskpv	Näitab topelt neid sync nuppe ja delete onoff. Koopia tuleb selle mingi kolm lisapunkti või expand tolbariga vms
Fixed?		ei	ei	ei	ei	ei
		Error 7	Error 8	Error 9	Error 15	Error 16
Kes leidis		Sken	Sken	Sken	Kaari	Kaari
Kus täpselt?		Business Centralis	Business Centralis	Business Centralis	Scoro-->BC sünkides	Scoros näeb Tooted teenused
Mis juhtuma pidi? Mis hoopis juhtus?			BC on mingi vale ettevõtte sisselogitud, kuid sync toimub ikka columbuse accountiga scoros	vaatasin postmaniga tooteid ja kõiki neid mis on BCs ei leidu Scoros, kuskil mingid errorid peavad olema. Debuggi ja vaata miks kõiki tooteid ei võeta, postmanist ka ei saanud kätte kõiki tooteid mis leidub BCs		Tootehind mppgil on rattal 4,000.0 , aga Scoros näitab 4€
Fixed?		ei	ei	ei	ei	ei
		Error 9	Error 10	Error 12	Error 17	Error 18
Kes leidis		Sken	Sken	Sken	Kaari	~~Kaari~~
Kus täpselt?		Business Centralis	Business Centralis	Business Centralis	BC's arvet koostades	~~Scoros arvet koostades näeb~~

Joonis 57. Vigade otsingu tabel.

4.1.6 Kasutajaliidese ja turu-uuring analüüs

Enamus meie andmetest, mis sünkroniseeritakse, lähevad juba olemasoleva kasutajaliidesega tabelitesse, mistõttu analüüsida saime me vaid enda loodud Scoro *Connector* lehe liidest. Kuna see leht on väga väike osa suurest süsteemist, siis on oluline, et ka meie laiendus näeb esteetilist samasugune välja. Seda aluseks võttes lähtusime ka oma rakenduse lehe tegemisel. Lehe arendamisel võtsime näidiseks Business Centralis oleva Envoice rakenduse (Joonis 58). Meie loodud rakenduse funktsionaalsuse nupud asuvad loogiliselt samas kohas kõigi teiste Business Centrali lehtedega. Väljad kuhu

salvestatakse sünkroniseerimiseks vajalikud andmed on samuti esteetiliselt ja loogiliselt samade kohtade peal kasutajaliideses.



Joonis 58. Envoice rakenduse kasutajaliides.

Suuremahulist turu-uuringut polnud erinevatel põhjustel võimalik läbi viia. Puudusid andmed olemasolevate Scoro kasutajate kohta. Samuti ei olnud võimalik allikate põhjal kindlaks teha Business Centrali kasutajate arvu, kes juba hetkel kasutavad lisaks majandustarkvarale ka ärihaldustarkvara Scoro. Erinevate allikate põhjal empiirilise uuringu käigus, peamiselt Microsofti blogi postituste ja veebilehtede põhjal oli võimalik kindlaks teha, et 26. aprill 2021 seisuga on Business Centrali pilvetehnoloogial üle 15000 kasutaja [5]. Business Centrali kasutajate arv on viimaste aastatega kasvanud 250% aastas [12]. See näitab, et säärasel pilvetehnoloogial põhineva SaaS rakenduse kasutamine ettevõtete seas muutub aina populaarsemaks ning potentsiaalsete Scoro Connector laienduse kasutajate arv kasvab. Hetkel meile kättesaadavate andmete põhjal pole põhjalikumat analüüsi võimalik teostada.

4.2 Kirjanduse ülevaade

Majandustarkvara süsteemid on tänapäeval väga populaarseks saanud ning on saamas iga ettevõtet standard tarkvaraks. Nõudlus pilvetehnoloogial põhinevate majandustarkvarade järgi aina kasvab olenemata sellest, kas need on avalikud või privaatsed pilveteenused. Uuemad majandustarkvara lahendused, mis on mõeldud pigem väiksema ja keskmise suurusega ettevõtetele on pilvetehnoloogiatel põhinevad, pakkudes palju erinevaid funktsionaalsusi [11, lk 8].

2019. aastal korraldatud raporti kohaselt 94% organisatsioonidest kasutavad erinevaid pilvetehnoloogiaid. Nendest 91% kasutavad avalikke pilveteenuseid ja 72% privaatseid teenuseid. Üha rohkem ettevõtteid kasutavad oma äriprotsesside toetamiseks pilvetehnoloogiatel põhinevaid tarkvara lahendusi. Ettevõtte vaatevinklist lähtudes on pilvetehnoloogiatel põhinevad majandustarkvara lahendused odavamad ja nende hinnastatus tihtipeale saab ka määravaks miks ettevõte pilvetehnoloogiate kasuks otsustavad [2, lk 11-20]. Sellistest näitajatest tulenevalt on meie rakendusel kindlasti väga suur potentsiaal kõigi Business Centrali kasutajate hulgas, nii praegu kui ka tulevikus. Eriti näeme on rakenduse kasutamisvõimalusi just väikese ja keskmise suurusega ettevõtete seas.

Isatou Jeng viis 2017. aastal bakalaureusetööks läbi juhtumiuuringu, kus näidisettevõtte ERP süsteemile võeti lisaks kasutusele CRM süsteem. Töö uuringuna leidis autor, et vanade klientide hoidmine, uute saamine ja nende eest hoolitsemine on suur tuluallikas ettevõttele ning müügiinimeste kasutatava CRM süsteemi teadmiste kombineerimisel ettevõtte üldise planeerimisega ERP süsteemis on ettevõtte jaoks väga kasumlik [8, lk 37-39]. Selle töö autoritele tundub, et ka just täpselt Scoros oleva info ühildamine oleks väga kasumlik ettevõtetele, kes kasutava Business Centrali ja Scorot.

Digitaalne sertifikaat on elektrooniline vahend, mille abil on võimalik kindlaks teha võrgus olevaid arvuteid, ettevõtteid või muid olemeid. Erinevate eesmärkide jaoks on olemas mitmeid erinevaid sertifikaate nagu SSH või X.509. Tüüpiliselt sertifikaat koosneb avalikust krüpteerimist võtmest ja omaniku võtmest. Certificate Authority(CA) nimetatakse asutusi, kes verifitseerivad enne sertifikaadi väljaandmist selle omaniku identiteedi ja veenduvad, et tegemist on õige omanikuga. Samal põhimõttel kasutatakse digitaalset allkirja ka tarkvara allkirjastamisel mida nimetatakse *code signing*. Tarkvara

rakendus pakendatakse ja sellele paketile arvutatakse krüptograafiline räsi. Seejärel arendaja krüpteerib sõnumiräsi privaatse võtmega, tekitades allkirjastatud tarkvara komplekti. Teisel osapoolel on aga võimalus saadetud räsi dekrüpteerida arendaja avaliku võtmega ning võrrelda nende räsikoode, kui räsikoodid klapivad võib tarkvara vastuvõtja kindel olla, et tarkvarasse ei ole vahepeal lisatud pahavara [1]. Columbus Eesti AS on verifitseeritud Sertico RSA code signing CA poolt ja kõik kes kasutavad nende poolt allkirjastatud tarkvara saavad kindlad olla, et tegemist ei ole pahavaraga.

4.3 Detailne teostatud tööde logi

Testrakendus koosneb 140 koodireast, mis testib põhilist kasutusjuhtu klientide sünkroniseerimine. Põhi rakendus aga koosneb 1977 koodi reast, mille kirjutas Henrik Hansson ainsa arendajana. Tehniline rakenduse äriloogika valmis meeskonna ühtse töö käigus ja Columbus Eesti AS koostöös. Koostati mitmeid publitseerimiseks vajalike dokumente ja turundusmaterjale. Viidi läbi kasutajaliidese analüüs ja koostati kättesaadavate andmete põhjal turu olukorra analüüs. Järgnevalt on nädalati kirjeldatud tehtud töö.

Tabel 1. Esimese iteratsiooni logi 1.03.2021 -28.03.2021.

<table>
<tr><td></td><td>Henrik</td><td>Sken</td><td>Kaari</td></tr>
<tr><td>1. Nädal</td><td>- Iganädalasel koosolekul osalemine.
- Tiimi standup.
- Jagatavate dokumentide ja keskkonna valmis sättimine.</td><td>- Kasuliku kirjanduse otsing.
- Iganädalasel koosolekul osalemine.
- Tiimi standup.
- Jagatavate dokumentide ja keskkonna valmis sättimine.</td><td>- Dokumentide, jagatud kaustade, lõputöö struktuuri valmispanek tiimi jaoks.
- Iganädalasel koosolekul osalemine.
- Tiimi standup.</td></tr>
<tr><td>2. Nädal</td><td colspan="3" rowspan="2">- Meeskonnaprojektist poolelijäänud projekti ülevaatamine .
- BC ja Scoro läbikatsetamine, et loetleda kõik vead, mis vajavad parandamist (loogika, “ei tööta”, valed sõnumid jne).
- Iganädalasel koosolekul osalemine.</td></tr>
<tr><td>3. Nädal</td></tr>
<tr><td>4. Nädal</td><td>- Iganädalasel koosolekul osalemine.
-Tarkvara testimine.</td><td>- Iganädalasel koosolekul osalemine.
- Kasuliku kirjanduse otsing
-Tarkvara testimine.</td><td>- Iganädalasel koosolekul osalemine.
- Kirjandusliku ülevaate otsing.
-Tarkvara testimine.</td></tr>
</table>

Tabel 2. Teise iteratsiooni logi 29.03.2021 -18.04.2021.

	Henrik	Sken	Kaari
5. Nädal	- Ettevõtte juhendajalt saadud koodinõuete rakendamine ning paranduste sisseviimine.	- Tarkvara testimine. - Turu-uuringu teostamine. - Kasutajaliidese analüüs.	-Tarkvara testimine. - Resursside kaardistamise kontroll, puuduste täitmine. - Kasutajaliidese analüüsiks meetodite ja tööriistade otsing.
6. Nädal	- Iganädalasel koosolekul osalemine. - Resursside lisamine. - Postmaniga resursside API kutsete kontroll.	- Iganädalasel koosolekul osalemine. -Tarkvara testimine. -Kirjandus materjali läbitöötamine. - Arendaja konsulteerimine resurssiküsimustes.	- haige - Arendaja konsulteerimine resurssiküsimustes. -Tarkvara testimine. - Kirjandusliku ülevaate lugemine. - Näidislõputööde otsing.
7. Nädal	- Iganädalasel koosolekul osalemine. - TalTech juhendajatega koosolek. - Resursside lisamise jätk ja lisamise käigus tekkinud vigade parandus.	- Iganädalasel koosolekul osalemine. - TalTech juhendajatega koosolek. - AppSource jaoks nõutud dokumentide uurimine. - Arendaja konsulteerimine resurssiküsimustes.	- Iganädalasel koosolekul osalemine. - TalTech juhendajatega koosolek. - AppSource jaoks nõutud dokumentide uurimine. - Arendaja konsulteerimine resurssiküsimustes.
8. Nädal	- Iganädalasel koosolekul osalemine - Ilmnenud vigade parandamine. - Setup tabeli laiendamine juhendajaga.	- AppSource materjalide ettevalmistamine, tehtud töö kajastamine lõputöös. - Iganädalane koosolek. - AppSource ülevaatlik koosolek konsultandi vaatepunktist.	- Iganädalasel koosolekul osalemine. - AppSource ülevaatlik koosolek konsultandi vaatepunktist. - AppSource materjalide ettevalmistamise algus.

Tabel 3. Kolmanda iteratsiooni logi 19.04.2021 - 16.05.2021.

<table>
<tr><th></th><th>Henrik</th><th>Sken</th><th>Kaari</th></tr>
<tr><td>9. Nädal</td><td rowspan="2">- TalTech juhendajatega koosolek (9.nädal)
- Testimise käigus leitud vigade parandamine
- Iganädalasel koosolekul osalemine</td><td colspan="2">- AppSource materjalide ettevalmistamine
- Tarkvara testimine, et leida kas ikka töötab või on katki
- TalTech juhendajatega koosolek
- Kirjanduse lugemine.</td></tr>
<tr><td>10. Nädal</td><td colspan="2">- Projekti tulemuste kajastamine lõputöös
- Tarkvara testimine, et leida kas ikka töötab või on katki
- Iganädalasel koosolekul osalemine</td></tr>
<tr><td>11. Nädal</td><td>- Lõputöö vormistamine
- Testimise käigus leitud vigade parandamine
- Tiimikoosolek
- Iganädalasel koosolekul AppSource’i publitseerimine</td><td colspan="2">- Lõputöö vormistamine
- AppSource materjalide viimne viimistlemine
- Tiimikoosolek
- Iganädalasel koosolekul AppSource’i publitseerimine
- Veebilehe disain (Kaari)</td></tr>
</table>

4.4 Hinnang projekti teostamise protsessi kohta

Tiimisiseselt juhti ei määratud, küll aga võttis Kaari ette, et oleksid lõputöö mallid ning omavahelise suhtluse ning dokumentide hoiustamise süsteem korras. Kaari oli ka tähtaegade meenutaja. Iganädalaselt toimunud koosolekud olid kõigil kalendris ettevõttepoolse kutsena. Tiim plaanis esialgu hakata tegema iganädalaseid tiimisiseseid *standup*’e, et vaadata üle tehtud töö ning järgnevad plaanid, aga need koosolekud otsustati ära jätta, kuna iganädalasel ettevõttega koosolekul sai kõik need aspektid läbi räägitud. Juhendajad lubasid ka salvestada iganädalaseid koosolekuid, et meeskonnal oleks võimalik tagasi vaadata ning vaiksemas tempos läbi töötada ettevõtte poolsed parandused, vihjed ning soovitused.

Esimese kahe iteratsiooni ajal oli töötempo aeglasem ning intensiivsem töö algas aprilli keskpaigas, kus vaatasime realistlikult projekti ajatelje läbi ning otsustasime, et AppSource’ga tegelemine peab kiiresti algama. Seejärel tõusis tiimi produktiivsus ja töötempo. Kogu projekti vältel tuli lõputöö koostamiseks aega leida õhtustel aegadel ning

neljapäevast pühapäevani. Esimese kuue nädala jooksul sai tehtud tööd 198 tundi ning järgneva viie nädala jooksul sai tehtud tööd 270 tundi.

Ülesannete jaotamine tulenes osaliselt varasema pädevuse olemasolust, näiteks Kaaril turundus- ja disainitaust logo ja tootelehe tegemiseks või Skenil meeskonnaprojekti raames omandatud kontaktide info haldamise teadmised, aga ka ülesande pakilisusest.

Projekti kitsaskohaks loeks tiim seda, et teema oli üpris kitsas. Kuna Business Centralil on juba endal tugevalt sisse juurutatud standardid ning disain, siis tiimil polnud suurt väljakutset disaini kaalumise või lahenduse loomise suhtes. Seetõttu kulus mõnes kohas vähem aega kui plaanitud, aga näiteks koodi parandamise ja kõikide vigade leidmisega läks oodatust rohkem aega, sest kui sai paar viga parandatud, siis ilmnesid järgmised vead parandamiseks.

Kokkuvõttes olid ettevõtte poolsed juhendajad siiralt toetavad ning abivalmid igal hetkel. Paremat poleks osanud lootagi.

4.5 Konsensuslik hinnang meeskonnaliikmete panuse kohta

Seekordsel projektil oli üksikul arendajal kõige suurem koormus oma õlgadel, et saada ikka projekt tähtajaks töötama ning valmis publitseerimiseks. Kogu meeskond käis iganädalastel koosolekutel va haigestumisel ning küsimuste korral suhtlus toimus tiimiliikmete vahel. Igal liikmel oli oma nõrgemaid kohti, aga oma tugevustega balanseerus tiimis töö ja koormus hästi. Seetõttu on tiim üksteist hinnanud kõik hindega 0 ehk kõik panustasid samaväärselt.

5. Kokkuvõte

Lõputöö projekti jooksul teostati Colmbus Eesti AS kasutatava Business Centrali ja Scoro müügitarkvara andmete sünkroniseerimise laienduse täiendused ning AppSource'i publitseerimine Projekti meeskonda kuulus kolm äriinfotehnoloogia viimase kursuse tudengit Henrik Hanson, Kaari Kaasik ja Sken Selge.

Varasemalt puudus andmete ülekandmise laiendus Scoro ja Business Centrali vahel. Seetõttu sai lõputöö projektiks täiendada meeskonnaprojektis algatatut ning ette valmistada vajalikud dokumendid laienduse publitseerimiseks AppSource'i. Lisaks dokumentidele valmisid rakenduse töötamiseks vajalikud nõuded, äriloogika ja kasutusjuhud. Nõuete põhjal valmis arendaja poolt rakenduse kood, milles järgiti puhta koodi ja objektorienteeritud programmeerimise printsiipe. Kood kirjutati nii, rakendust on võimalik hiljem laiendada ja arendada lisa funktsionaalsusi.

Edasi saab laiendust täiendada eestikeelse tõlke failiga ning raportite ning muude andmete sünkroniseerimisega. Sünkroniseerimislaienduse jaoks kirjutati ka eraldi automaattesti rakendus, mis testib ühte põhi kasutusjuhtu. Nüüdseks on rakendus AppSource keskkonda publitseeritud koos kõigi valminud dokumentidega. Projekti võib pidada edukaks, kuna kõik eesmärgid said täidetud ning rakendus on kõigile vabalt allalaetav.

Kasutatud kirjandus

[1] M. Anderson, “Certificates, code signing and digital signatures,” May 2018, p. 7. doi: 10.1117/12.2302618. (14.05.2021)

[2] Y. W. Chang, “What drives organizations to switch to cloud ERP systems? The impacts of enablers and inhibitors,” *Journal of Enterprise Information Management*, vol. 33, no. 3, pp. 600–626, Apr. 2020, doi: 10.1108/JEIM-06-2019-0148.(18.04.2021)

[3] Columbus Eesti AS koduleheküg. [WWW] https://www.columbusglobal.com/et/ (14.04.2021)

[4] Development in AL. [WWW] https://docs.microsoft.com/en-us/dynamics365/business-central/dev-itpro/developer/devenv-dev-overview

[5] J.Gumpert. Dynamics 365 Business Central reaches 15K cloud customers with user count rising even faster. [WWW] https://msdynamicsworld.com/story/dynamics-365-business-central-reaches-15k-cloud-customers-user-count-rising-even-faster (03.05.2021)

[6] H.Hanson, K.Kaasik, S.Selge. Ettevõttes Columbus Eesti AS Business Centrali ja Scoro api liidestamine: ITB1706 Infosüsteemide arendamise meeskonnaprojekt. Tallinna Tehnikaülikool, Tallinn, 2020 (14.04.2021)

[7] Introduction - Cloud architecture for Business Central [WWW] https://docs.microsoft.com/en-us/learn/modules/customize-dynamics-365-business-central/1-cloud (05.05.2021)

[8] I. Jeng, Benefits of having a CRM system in addition to an ERP system Case study: Company X: Bachelor thesis, Helsinki Metropolia University of Applied Sciences, 2017.

[9] Make your solutions available in Microsoft AppSource and within Office [WWW] https://docs.microsoft.com/en-us/office/dev/store/submit-to-appsource-via-partner-center (04.05.2021)

[10] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. 1st ed. United States: Prentice Hall, 2008

[11] I.Orosz, T.Orosz, A.Selmeci, Institute of Electrical and Electronics Engineers, *SAMI 2019 : IEEE 17th World Symposium on Applied Machine Intelligence and Informatics : proceedings : January 24-26, 2019, Herlany, Slovakia.*(22.04.2021)

[12] G.Pezzini. Dynamics 365 Business Central goes fully cloud: reporting from Directons4Partners 2020[WWW] https://www.lsretail.com/blog/dynamics-365-business-central-goes-fully-cloud (03.05.2021)

[13] Scoro koduleheküg. [WWW] https://www.scoro.com/ (14.04.2021)

[14] Scoro Connector. [WWW] https://appsource.microsoft.com/en-us/product/dynamics-365-business-central/PUBID.columbus-estonia%7CAID.scoro%7CPAPPID.eaac4e55-174c-49c6-a95c-15f020b6e290?tab=Overview (18.05.2021)

[15] Tailor, extend, and build applications [WWW] https://docs.microsoft.com/en-us/learn/modules/customize-dynamics-365-business-central/2-applications (05.05.2021)

[16] What is Microsoft AppSource? [WWW] https://docs.microsoft.com/en-us/marketplace/appsource-overview (07.05.2021)

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

Mina, Henrik Hansson, Kaari Kaasik ja Sken Selge

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Business Centrali ja Scoro API liidestamine ning AppSourcei publitseerimine", mille juhendaja on Viljam Puusep ja Karl-Erik Karu
   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. Oleme teadlikud, 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.

18.05.2021

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 – Test stsenaarium

Scoro Connector
User Scenario Documentation

Introduction

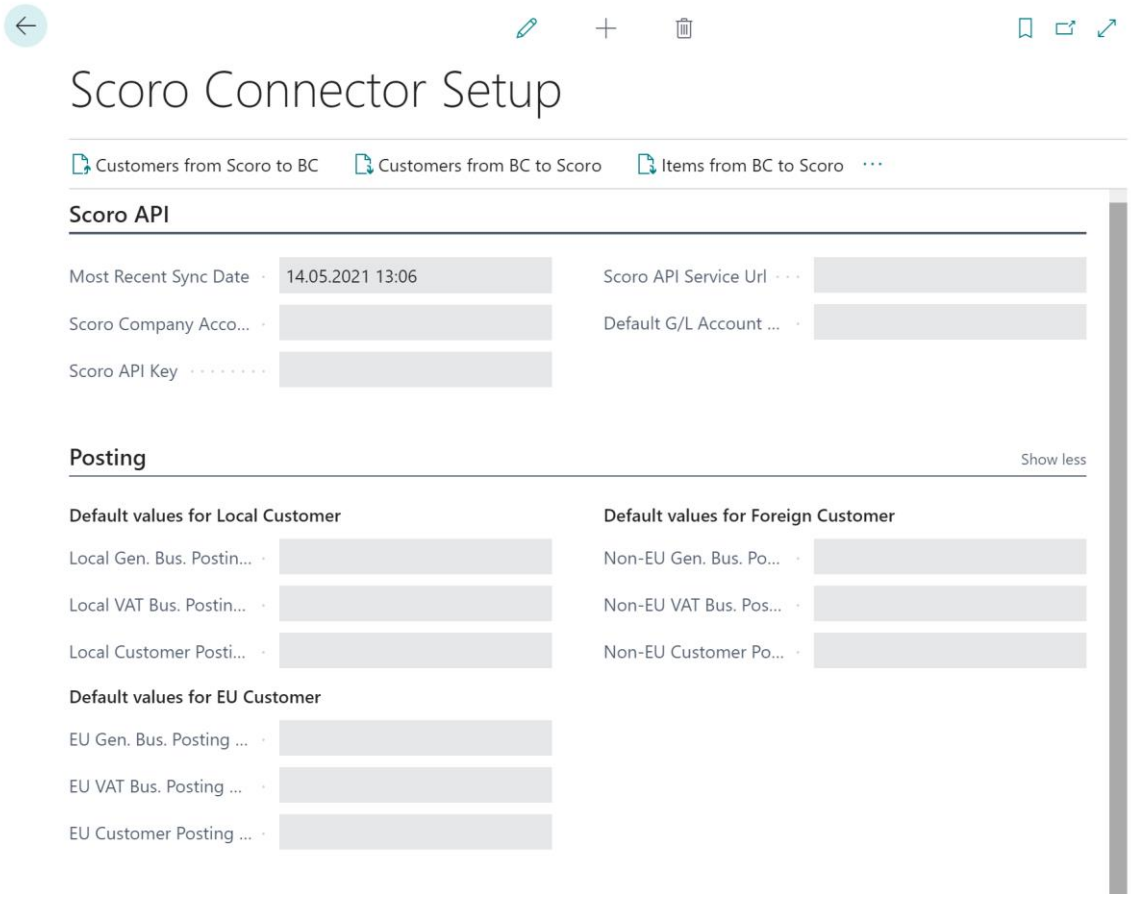
Scoro Connector functionality enables the following:

- Data synchronization between Business Central and Scoro
- Important customer data is kept synchronized in both softwares
- Business Central product items are synchronized to Scoro for a better sales process
- All sales invoices from Scoro are brought to Business Central for easier accounting
- Resources data synchronization to Scoro

Installation

1. Install Scoro Connector application package from AppSource
2. Open Extension Management page by using the search function
3. Select Scoro Connector and click Manage ->Configure

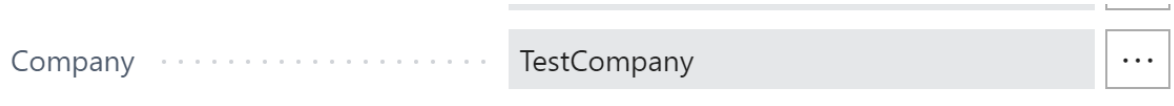
Now you should see an empty Scoro Connector Setup page, we will fill it with data later.



Company Setup
Open Companies page by using the search function

Press New and create new company with name **TestCompany,** wait until company is saved.

Click on Settings button and select My Settings
Change selected Company to **TestCompany**



Setup company with test data

Open **Configuration Packages** by using search function
Select **Process** and then **Import Package** select file test data file NAV17.2.W1.ENU.EVALUATION.rapidstart
Wait until package is being uploaded.

Select **Process** and then **Apply Package**
Wait until data is being imported, this will take a few minutes.

General Setup
Open Countries/Regions page by using the search function

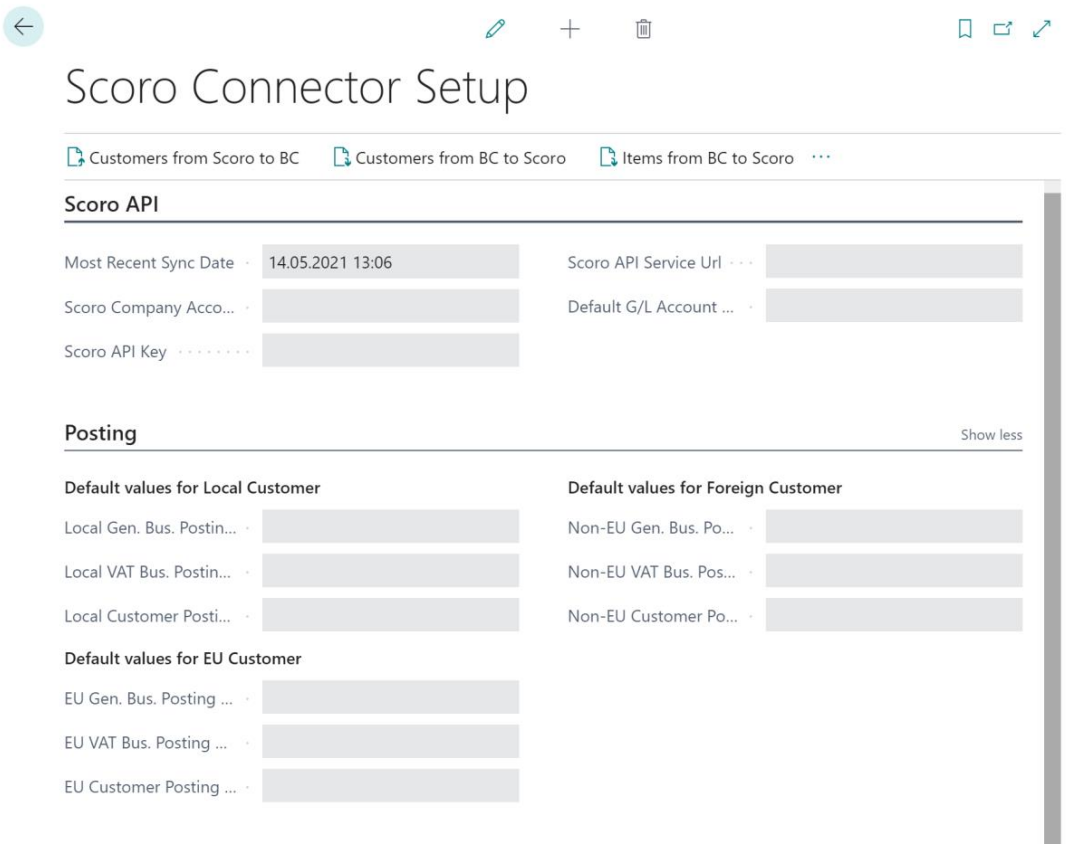
Edit Table and set USA line CPC-SC Scoro Code column to **USA**

Open Scoro Connector Setup page by using the search function

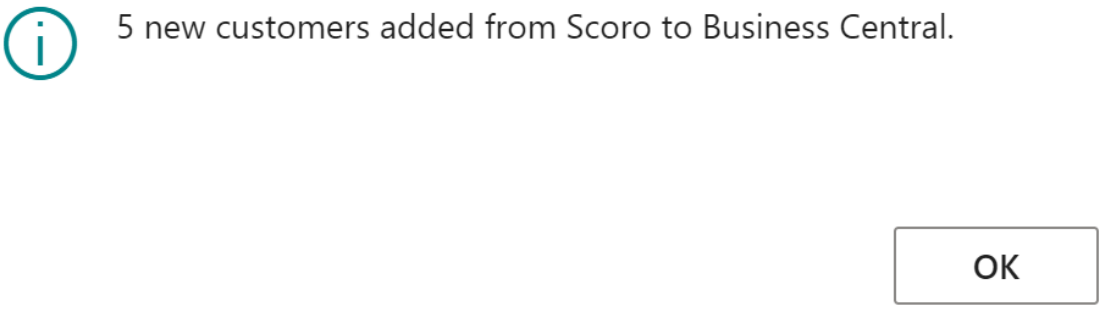
1. Enter Company Account ID: [redacted]
2. Enter Scoro API key: [redacted]
3. Enter Scoro API Service URL: [redacted]
4. Choose Default G/L Account No

Posting Setup
Press Show more

1. DEFAULT VALUES FOR LOCAL CUSTOMER
    a. Local Gen. Bus. Posting Group: DOMESTIC
    b. Local VAT Bus. Posting Group: DOMESTIC
    c. Local Customer Posting Group: DOMESTIC
2. DEFAULT VALUES FOR EU CUSTOMER
    a. EU Gen. Bus. Posting Group: EU
    b. EU VAT Bus. Posting Group: EU
    c. EU Customer Posting Group: EU
3. DEFAULT VALUES FOR FOREIGN CUSTOMER
    a. Non-EU Gen. Bus. Posting Group: EXPORT
    b. Non-EU VAT Bus. Posting Group: EXPORT
    c. Non-EU Customer Posting Group: FOREIGN



Press Customers from BC to Scoro
Defined Customers are now sent to Scoro API



Send Items to Scoro

1. Click Items from BC to Scoro
2. 20 products were sent to Scoro

Get Sales Invoices
Sign in to Scoro environment
Prepare Invoice in Scoro

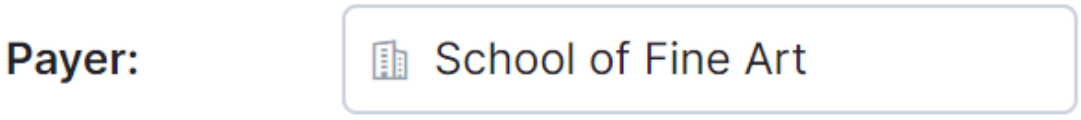
https://suno365.scoro.ee/login

- Email address: [redacted]
- Password: [redacted]

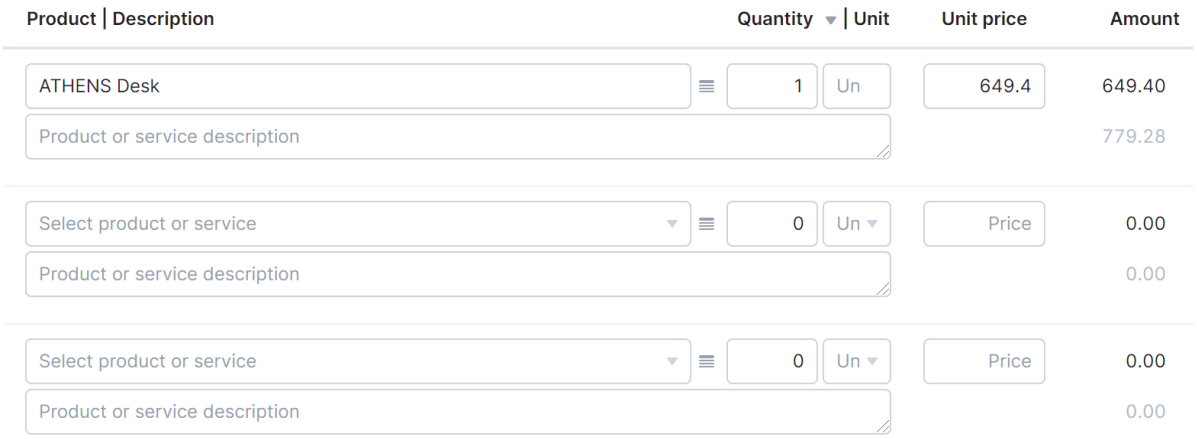
From top menu select Invoices

Press add **+New**

Choose **Payer** from dropdown menu **School of Fine Art**

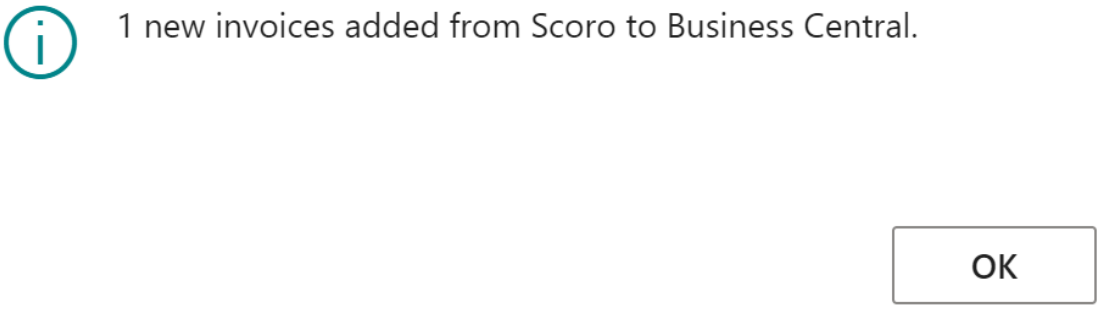


Create new invoice line by choosing **ATHENS Desk** from product menu.



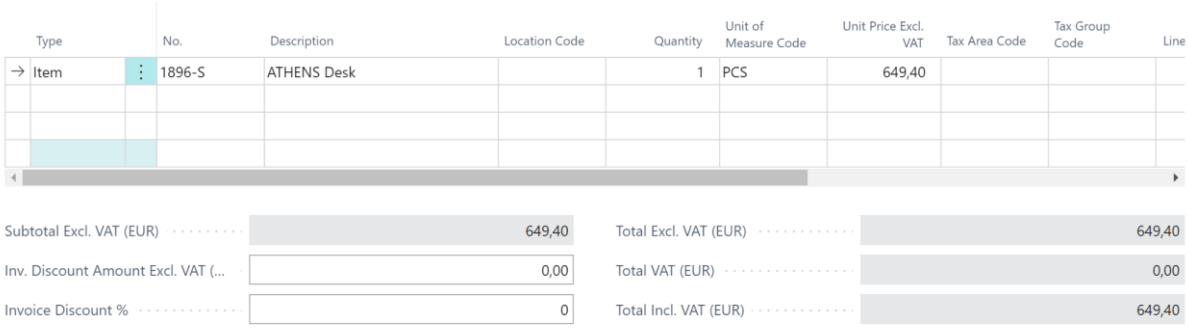
Check that Quantity is 1 and Unit price 649.4
Click Save at the end of the page

Go to Scoro Connector Setup page in Business Central
Click Invoices from Scoro to BC



Check if new Invoice has been created:

1. Open **Sales Invoices** page by using the search function
2. Find Invoice you just created **School of Fine Art**
3. Open Invoice page by clicking on the invoice no
4. Check posting values in Invoice Details
    a. VAT Bus. Posting Group – FOREIGN
5. Check invoice line
    a. Type: Item
    b. No.: 1896-S
    c. Quantity 1
    d. Unit Price Excl. VAT: 649.40
6. Check invoice total amounts
    a. Total Excl. VAT (EUR): 649.40
    b. Total VAT (EUR): 0
    c. Total Incl. VAT (EUR): 649.40



Type	No.	Description	Location Code	Quantity	Unit of Measure Code	Unit Price Excl. VAT	Tax Area Code	Tax Group Code	Line
Item	1896-S	ATHENS Desk		1	PCS	649,40			

Subtotal Excl. VAT (EUR)	649,40	Total Excl. VAT (EUR)	649,40
Inv. Discount Amount Excl. VAT (...	0,00	Total VAT (EUR)	0,00
Invoice Discount %	0	Total Incl. VAT (EUR)	649,40

Lisa 3 – Eneseanalüüs

1. Henrik Hansson

Tehtud töö ülevaade: projekti tehniline realiseerimine ja koodi pidev täiustamine. Arenduse tarbes rakendusspetsiifilise AL keele õppimine. Suhtlus konsultantidega äriloogiliste nõuete mõistmiseks ja tagasiside saamiseks.

Eneseanalüüs: õppisin palju. Äritarkvara arendamisega mul eelnevat kogemust polnud ning lühikese aja jooksul pidin endale selgeks tegema uue keele. AL keele juures pidin sealjuures õppima veel väga spetsiifilist keele osa, milleks on API päringute koostamine ja JSON objektidega tegelemine. Ma ise ütleksin, et 60% töökoormusest oli sügissemestril, kevadel toimus tehtud töö viimistlemine enne publitseerimist. Kasulik kogemus oli kindlasti ka konsultantide ja ettevõttega suhtlemine ning seeläbi reaalses tööprotsessis osalemine.

2. Kaari Kaasik:

Tehtud töö ülevaade: arendaja konsulteerimine, Scoro andmeväljade ja BC andmeväljade ühtimise kontroll. Sünkroniseerimise ja liidese testimine, et leida vigu. Tiimi töö organiseerimine, et tähtaegadest kinni hoitakse ning Google Drive'i valmispanek. Google Drive's tegin valmis lõputöö struktuuri, koosolekute protokollimise, iteratsioonide jälgimine, vigadeotsingu ning *todo-listi* tegemine. AppSource'i jaoks tegin turunduslehe ning disainisin liidestuse kodulehte ning kasutajajuhendi lehte.

Eneseanalüüs: õppisin Microsoft rakenduste publitseerimise protsessi kohta. Varem ei teadnud sellest midagi. Ka sain taaskord proovida oma *CSS* ja *HTML* oskusi, mida ma ammusest ei mäletanud, aga sai *w3school.com* abil meenutatud. Kindlalt sai harjutatud arenduse ja konsulteerimise vaatepunktide erinevust, mis tuli vahepeal arutelude käigus ilmsiks. Ning seetõttu harjutada teise vaatenurga alt seletamist, mis on analüütik/konsultandile oluline.

3. Sken Selge

Tehtud töö ülevaade: rakenduse äriloogika välja kujundamine koostöös ettevõtte ja meeskonna liikmetega. Sünkroniseerimiseks vajalike andmeväljade kaardistamine. Kirjanduse ülevaate jaoks materjalide leidmine ja läbitöötamine. Publitseerimiseks vajalike dokumentide koostamine teststsenaarium (Lisa 2), kasutajajuhendi koostamine ja lehe jaoks HTML kirjutamine. Turu-uuringu teostamine ja kasutajaliidese analüüs. Rakenduse pidev mehaaniline testimine ja arendajale tagasiside andmine.

Eneseanalüüs: projekti suurimaks väljakutseks võiks pidada konsultandi rolli võtmist, kuna selle rolli praktilist osa ülikoolis ei ole võimalik õppida, vaid ikka ainult seda tehes. Konsultandiks olemine avardas selle töö-ala võimalusi ja spetsiifikat mida see endaga kaasa toob. Hästi oli aru saada kuidas konsultandid näevad ühte maailma, ning arendajad näevad hoopis teist.