TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Karlos Heinla 154836IABB

UUSIMATE ARENDUSPRAKTIKATE RAKENDAMINE VABAVARALISTE SISUHALDUSSÜSTEEMIDE PLATVORMIDELE MAGENTO NÄITEL

Bakalaureusetöö

Juhendaja: Tarvo Treier
MSc
Kaasjuhendaja: Riho Pihelpuu
MSc

Tallinn 2019

Autorideklaratsioon

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

Autor: Karlos Heinla

19.05.2019

Annotatsioon

Bakalaureusetöö teema on e-kommerts sisuhaldussüsteemi arhitektuuride analüüsimine ja võrdlemine ning analüüsi abil parima Magento e-poe platvormi arhitektuuri valimine, millele kliendi poolt soovitud e-pood luua.

Töö eesmärk on selgitada välja, millise arhitektuurse versiooni peale e-pood luua ning millised on hetkel võimalikud riskid uue *Decoupled* arhitektuuri rakendamisel e-kommerts platvormile Magento. Eesmärgi saavutamiseks kasutas autor kvalitatiivset ja kvantitatiivset uurimismeetodit ning tegi olulist funktsionaalsust sisaldava prototüübi.

Kvalitatiivsest uuringust selgus, et Magento platvormil *Decoupled* arhitektuuri kasutamisel on veel lahendamata küsimusi ja mitmeid riske. Näiteks informatsiooni, dokumentatsiooni, näidete ning *pluginate* ja moodulite toe vähesus või puudulikkus. See tähendab, et klient või ettevõte ise peab tegema suure investeeringu eelmainitud arhitektuuri uurimisele ja testimisele. Kvantitatiivsest uuringust selgus, et enamik *pluginaid* ja mooduleid arendavatest ettevõtetest ei toeta *Decoupled* arhitektuuril põhineva Magento kasutamist, sest tööjõu- ja finantsressursse on vaja mujale suunata. Siiski on nad valmis tulevikus mainitud arhitektuuri toetama kui selle järele nõudlus suureneb. Prototüübi tegemisel selgus, et *Decoupled* arhitektuuril põhineva e-poe loomine Magentos on keeruline ja aeganõudev, sest näited ja dokumentatsioon on puudulikud ning funktsionaalsuse tööle saamiseks tuleb arendajal proovida ja testida erinevaid võimalusi.

Läbiviidud intervjuude ja küsitluse analüüsi tulemuste ning tehtud prototüübi põhjal soovitab autor hetkel e-poe loomiseks kasutada e-kommerts platvormil Magento *Decoupled* arhitektuuri asemel *Coupled* arhitektuuri.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 31 leheküljel, 6 peatükki, 17 joonist.

Abstract

The Application of the Newest Development Practices to Open Source Content Management Systems Based on Magento

The topic of this Bachelor's Thesis is the analysing and comparing of the architectures of an e-commerce content management system and in addition, by carrying out an analysis selecting the best architecture of the Magento online store platform to which the e-store chosen by the client can be built on.

The aim of the thesis is to find out which architectural version the e-shop to build upon and what the potential risks of implementing the new Decoupled architecture for the commercial platform Magento actually are. In order to achieve this goal, the author used qualitative and quantitative research methods and made a prototype with significant functionality.

The qualitative study revealed that there are still outstanding issues and a number of risks in the application of the Decoupled architecture used on the Magento platform. For example, the lack or inadequacy of the provided information, documentation, examples and the insufficiency or deficiency of the support of the plugins and modules. That is to say, the customer or the company itself must make a major investment in researching and testing the aforementioned architecture.

The quantitative research showed that most of the companies that develop plugins and modules do not support the Decoupled architecture-based Magento because the financial resources of the workforce need to be directed elsewhere. Nevertheless, they are ready to support the architecture in the future if the demand increases.

As a result of the making of the prototype, it became evident that the creation of the e-store Magento, which is based on the architecture Decoupled, is a complicated and time-consuming process, as the examples and documentation are incomplete and the developer needs to try out and test different options in order to get its functionality to work.

Based on the architecture of the e-commerce platform Magento, which the Customer wishes to build the e-store on, and likewise, on the basis of the results of the interviews, the analysis and the prototype, the author of this thesis does currently recommend creating an e-store which would be built on the Coupled architecture instead of Decoupled architecture.

The thesis is in Estonian and contains 31 pages of text, 6 chapters, 17 figures.

Lühendite ja mõistete sõnastik

API	Application Program Interface, rakendusprogrammiliides
Back end	Loogika kiht
C#	.NET raamistikul põhinev objektorienteeritud programmeerimiskeel
CMS	Content Management System, sisuhaldussüsteem
CSS	Cascading Style Sheets
Docker	Tööriist rakenduste loomise, kasutamise ja käivitamise lihtsustamiseks [1].
Front end	Kasutajaliidese kiht
Google reCAPTCHA	Teenus, mis kaitseb veebilehte kuritarvitamise ja rämpsposti eest [2].
GraphQL	Andmete päringukeel
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
JAVA	Objektorienteeritud programmeerimiskeel
JavaScript	Programmeerimiskeel veebiarenduseks
Mikroteenused	Arhitektuuri stiil, mis jagab rakenduse üksteisest sõltumatuteks osadeks
Moodul	Tarkvarakomponent, mis sisaldab ühte või mitut funktsionaalsust, mida kasutatakse mitmel korral [3].
MySQL	Andmebaasisüsteem
Node.js	JavaScripti käitussüsteem
PHP	Hypertext Preprocessor
Plugin	Programmijupp, mis annab olemasolevale programmile lisa võimalusi [4].
PWA	Progressive Web Applications, progressiivsed veebirakendused
React	JavaScripti raamistik kasutajaliidese loomiseks
REST	Representational State Transfer
URL	Uniform Resource Locator

Sisukord

Jooniste loetelu

1 Sissejuhatus

Tänapäeval eksisteerib internetis miljoneid veebilehti ja iga päevaga luuakse neid aina juurde. E-poed, reklaamlehed, uudiste kanalid, otsingumootorid, blogid, foorumid ja kodulehed – need on väike osa kõigist variantidest, mida on võimalik veebis luua. Üks olulisemaid etappe veebilehe loomise juures on eesmärgi läbimõtlemine, sest hiljem peab loodud veebileht nii sisu kui kujunduse poolest eelnevalt mõtestatud eesmärki täitma. Veebilehtede tegemise lihtsustamiseks on erinevad arendusettevõtted loonud kergesti kasutatavad rakendused, mille abil saavad inimesed, kes ei tunne koodi ega oska programmeerida, teha lihtsa disaini ja funktsionaalsusega veebilehti. Tõsiasi on, et enamikel juhtudel nendest rakendustest ei piisa, et veebileht oma eesmärki täidaks. Klientidele ja kasutajatele laiema funktsionaalsuse ja disaini pakkumiseks on vaja kasutusele võtta keerulisemad süsteemid.

Tänapäeval on olemas erinevaid vabavaralisi ja kommertsraamistikke ning -rakendusi, mis lihtsustavad laiema funktsionaalsuse ja disaini arendamist. Erinevatel raamistikel on mitmeid arhitektuuri versioone ning neil kõigil on omad plussid ja miinused. Töö autor vaatleb ja võrdleb ühte spetsiifilist vabavaralist e-kommerts platvormi Magento ning selle võimalusi, kuid töös uuritavad arhitektuurid ja arenduspraktikad on laiendatavad enamikele vabavaralistele platvormidele.

1.1 Taust ja probleem

Bakalaureusetöö teema on e-kommerts sisuhaldussüsteemi (edaspidi CMS) arhitektuuride analüüsimine ja võrdlemine ning analüüsi abil parima Magento e-poe platvormi arhitektuuri välja valimine, millele luua kliendi poolt soovitud e-pood. Teema osutus valituks, kuna ettevõttes kus autor töötab on probleem aktuaalne. Nimelt soovib ettevõtte suurklient viia oma e-poe üle uuele tarkvaralisele platvormile. Magento on üheks valikuks mainitud ettevõttel. Magentol tuli 2018. aasta novembris välja versioon, mis võimaldab kasutada uuemaid arenduspraktikaid põhinedes kihilisel (edaspidi

Decoupled) arhitektuuril. *Decoupled* ideoloogia erineb eelmistes Magento versioonides ainukesena toetatavast monoliitsest (edaspidi *Coupled*) arhitektuurist.

Töö on vajalik, sest mitmed Eesti ja rahvusvahelised ettevõtted, kes tegelevad Magento e-poe platvormile arendamisega, on indekseerinud, et nad-ei taha uut versiooni veel, kas majanduslikel või oskusteabe puudumise tõttu, kasutusele võtta. Töös kirjeldatakse uue Magento e-poe platvormi versiooniga kaasa tulevat *Decoupled front end* arhitektuuri. Autor teeb Magento peale kõige olulisemat funktsionaalsust sisaldava prototüübi ning selle tulemusel teeb järeldused, kas kliendil on mõistlik investeerida uue arhitektuuri kasutusele võtmisesse.

Konkreetsemalt on töö abiks neile, kellel on vaja otsustada, millise CMS-i arhitektuuri peale oma avatud lähtekoodiga raamistikul põhinev rakendus ülesse ehitada, sest antud töös uuritud arenduspraktikad ja arhitektuurid on rakendatavad enamikele vabavaralistele platvormidele. Samuti on töö kasulik neile, kes soovivad oma e-poodi luua Magento platvormile ning peavad otsustama, kas valida *Coupled* või *Decoupled* arhitektuuril põhinev arhitektuur.

1.2 Ülesande püstitus

Käesoleva bakalaureusetöö eesmärk on selgitada välja, millise arhitektuurse versiooni peale e-pood luua ning millised on hetkel võimalikud riskid uue *Decoupled* arhitektuuri rakendamisel e-kommerts platvormile Magento. Eesmärgi täitmiseks püstitab autor kolm uurimisküsimust:

- mis põhjustel Magento ettevõtted, kes on aastaid tegelenud Magento platvormile e-poodide loomisega, ei soovi uuele versioonile üle minna?
- millised on moodulite ja *pluginate* tootjate hetkevõimalused ning tulevikuplaanid seoses Magento uue versiooniga?
- kas ja kui palju Magento uues versioonis lisatud *Decoupled front end* arhitektuuri kasutamise võimalus muudab arendust lihtsamaks, laiendab funktsionaalsuse loomise võimalusi, skaleeruvust, turvalisust ja paindlikust?

Töö oodatavaks tulemuseks on eelneva analüüsi ja uue versiooni peale tehtud prototüübi põhjal välja selgitada kas kliendi soovitud e-pood luua uue või vana Magento versiooni peale ning saada vastused püstitatud uurimisküsimustele.

1.3 Metoodika

Käesolevas töös toob autor välja kõik enamlevinud veebilahenduste *front end/back end* arhitektuurid. Põhilisi neist analüüsib detailselt ja võrdleb omavahel ning toob välja iga arhitektuuri plussid ja miinused. Seejärel uurib autor CMS-i platvormi Wordpress ja Wordpressi e-poe mooduli Woocommerce arhitektuure ja arengut. Töö aluseks valis autor Magento e-poe platvormi, millele klient analüüsib oma lahenduste migreerimist. Magento puhul uurib autor millised olulised muudatused uue versiooniga kaasa tulid.

Esimesele uurimisküsimusele vastuse leidmiseks teostab autor kvalitatiivse uuringu, mille käigus tehakse intervjuu kolme Eesti juhtiva Magento arendusettevõttega. Intervjuude tulemusel soovib autor teada saada kas ja miks Magento ettevõtted, kes on aastaid tegelenud Magento platvormile e-poodide loomisega, kardavad või ei taha uuele *Decoupled* arhitektuuril põhinevale versioonile üle minna.

Teisele uurimisküsimusele vastuse saamiseks viib autor läbi kvantitatiivse uuringu, võttes meili teel ühendust erinevate *pluginaid* ja mooduleid arendavate ettevõtetega ning uurib nende strateegiat *Decoupled* arhitektuuril põhinevate lahenduste arendamisel ja pakkumisel. Sellised platvormid nagu Magento, Wordpress, Drupal jms on paljuski üles ehitatud kogukonna ja ettevõtete poolt arendatavate moodulite kasutamisele, mis lihtsustab antud süsteemide arendamist ja juurutamist. Vajalik on uurida ja testida kuidas käib selliste *pluginate* ja moodulite ökosüsteemist tulevate lahenduste kasutamine uue *Decoupled* arhitektuuriga.

Kolmandale uurimisküsimusele vastuse leidmiseks loob autor olulist funktsionaalsust sisaldava e-poe prototüübi uue arhitektuurse *Decoupled* versiooni peale. Prototüübi tegemist alustab autor keskkonna ülesseadistamisega ja vajaliku funktsionaalsuse loomisega. Autor viib võimalusel läbi katsed prototüübil erinevate moodulite ja *pluginatega* ning uurib nende versioonide ajalugu, et aru saada kuidas käib moodulite ja *pluginate* uuendamise mehhanism *Decopled* lahenduse puhul.

Hinnang töö tulemustele põhineb autori arvamusel, mis omakorda põhineb töö käigus teostatud analüüsil ja saadud kogemustel.

1.4 Ülevaade tööst

Teises peatükis autor analüüsib ning toob välja järgnevate CMS-i arhitektuuride plussid ja miinused – *Coupled*, *Decoupled* ja *Headless front end* arhitektuur. Lisaks seletab autor lahti mõisted API, GraphQL ja REST API, et paremini aru saada eelnimetatud arhitektuuride olemusest ja toimimisest.

Kolmandas peatükis annab autor lühiülevaate Magentost ning toob välja selle olulisemad uuendused ja võimalused uues versioonis. Esimesele uurimisküsimusele vastuse leidmiseks viib läbi kvalitatiivse uuringu ja analüüsi, mis põhineb kolmel intervjuul Eesti ettevõtetega, kes tegelevad Magento platvormile arendamisega. Uurib nende arvamust ja võimalikku hirmu põhjust uue versiooni ees. Teisele uurimisküsimusele vastuse saamiseks teostab autor kvantitatiivse uuringu erinevaid *pluginaid* ja mooduleid arendavate ettevõtete hetkevõimalustest ning tuleviku arendusplaanidest seoses Magento uue versiooniga ja analüüsib saadud vastuseid.

Neljandas peatükis uurib autor tuntud CMS-i platvormi Wordpress ja selle e-poe moodulit Woocommerce, et teada saada, kas antud töö tulemused on üldistatavad ka teistele vabavaralistele platvormidele. Analüüsib millist arhitektuuri nad hetkel kasutavad, kuidas versioonid on muutunud, milline on olnud arhitektuuride areng ja kuidas versioonide areng on mõjutanud kasutajate hulka. Võrdleb kas kolmandas peatükis analüüsitud Magento areng on olnud sarnane Wordpressi ja selle e-kommerts mooduli WooCommerce arengule, et teha järeldused Magento arengusuuna osas.

Viiendas peatükis käsitleb autor Magento uue versiooni peale tehtud prototüüpi, et leida vastus kolmandale uurimisküsimusele. Selgitab prototüübi ülesehitust ja põhjendab miks osutus valituks just see funktsionaalsus, mis prototüübil küljes on.

Lõpuks esitab autor kokkuvõtte, kus on välja toodud püstitatud eesmärgi ja uurimisküsimuste tulemused ja järeldused.

2 Sisuhaldussüsteem ehk CMS arhitektuurid

CMS (*Content Management System*) ehk sisuhaldussüsteem on tarkvara süsteem, mis automatiseerib sisu haldamise ülesandeid. CMS aitab toimetajatel/arendajatel luua, organiseerida, kontrollida, turvata veebilehte ja lõpuks pakkuda sisu. [5] Kasutades CMS-i veebilehe loomiseks, tehakse seda administreerimisliideses, mis on sisuhalduse graafiline osa. See on nähtav ainult veebi administraatorile, mille kaudu on võimalik hallata kogu veebikeskkonda. [6]

Teises peatükis autor analüüsib ning toob välja järgnevate CMS-i arhitektuuride plussid ja miinused – *Coupled, Decoupled* ja *Headless front end* arhitektuur. Lisaks seletab autor lahti mõisted API, GraphQL ja REST API, et paremini aru saada eelnimetatud arhitektuuride olemusest ja toimimisest.

2.1 API, REST API ja GraphQL mõistete seletus

API (*Application Program Interface*) ehk rakendusprogrammi liides on rutiinide, protokollide ja tööriistade komplekt tarkvararakenduste ehitamiseks. Põhimõtteliselt määrab API kuidas tarkvarakomponendid omavahel suhtlevad. Hea API muudab programmi arendamise lihtsamaks, pakkudes kõiki plokke mida on arendajal võimalik kokku panna. [7] Kõik API päringud kasutavad meetodeid HTTP POST, GET, PUT ja DELETE, kus lisaks URL-ile antakse POST sisuna kaasa ka JSON-formaadis objekt, milles peaks kindlasti sisalduma vastuse andmete keel ja tunnus (id) [8]. Näiteks lubab CMS-i API arendajal luua ühenduse oma koodi ja CMS-i vahel ja niimoodi manipuleerida sisuobjektidega [9].

REST (*Representational State Transfer*) on veebipõhine API liides, mida kasutatakse HTTP protokolli abil. REST või RESTful API nimetus tuleneb sellest, et kliendi ja serveri vahelise päringu tegemise ajal ei salvestata vahepeal andmeid [10]. REST API kasutab HTTP meetodeid GET, PUT, POST ja DELETE päringute tegemiseks, mille abil on võimalik andmeid küsida ja muuta [11].

GraphQL on Facebooki poolt 2012. aastal välja töötatud andmepäringu keel. Avalikuks ja kõigile kättesaadavaks tehti see 2015. aastal. GraphQL pakub alternatiivi REST-põhistele arhitektuuridele. Selle eesmärk on kiirendada arendaja tööd ja minimeerida andmemahtu. [12] GraphQL on defineeritud kui andmete päringu keel ja käitusaeg. Oluline on mõista mõlemat osa definitsioonist:

- GraphQL on keel, mida kasutatakse tarkvara kliendipoolsele rakendusele õpetamiseks, et see on võimeline küsima vajaminevaid andmeid *back end* teenuselt, mis suhtleb omakorda GraphQL-iga;
- GraphQL on käitusaeg. See on rakenduskiht, mida serverirakendus saab kasutada GraphQL keeles tehtud taotluste mõistmiseks ja sellele vastamiseks. Põhimõtteliselt tegeleb see kiht GraphQL keele tõlkimisega. [13]

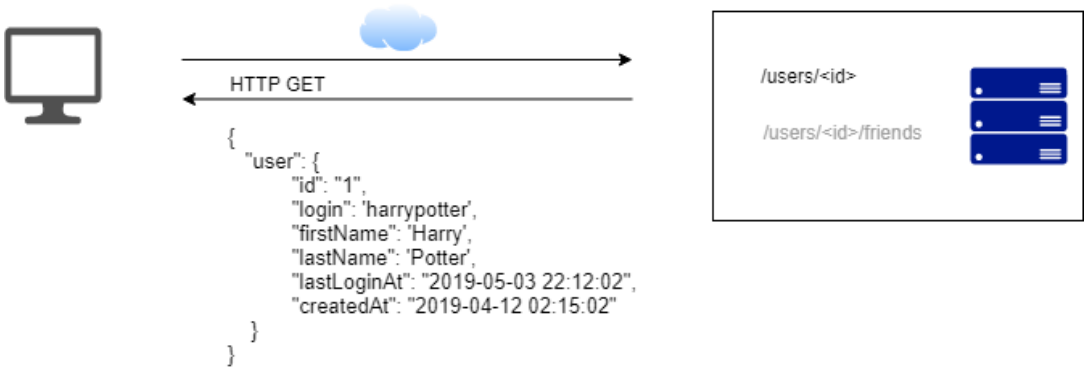
Suurim erinevus REST API ja GraphQL vahel on see, et kui esimene nõuab korraga mitme URL-i laadimist ehk iga andmeallika kohta on erinev lõpp-punkt, siis teine saab kõik vajalikud andmed kätte ühe päringuga ehk tal on üks lõpp-punkt. Isegi kui interneti ühendus on aeglane, töötavad GraphQL kasutavad rakendused ikka kiiresti. [14] Lisaks teeb GraphQL eriliseks võimalus luua objektitüüpe (Joonis 1). See annab võimaluse API andmed struktureerida päringupuuks ja küsida ühe päringuga mitme objekti andmeid [15]. Jooniselt 1 on näha, et tehakse *User* tüüpi objekt, mille sees kutsutakse *Friend* tüüpi objekt. Seejärel kasutatakse *Query* tüüpi objekti sees *User* tüüpi objekti. Seega on võimalik teha ainult üks päring *Query* objektis olevate andmete saamiseks ning automaatselt saadetakse ka *User* ja *Friend* objektis olevate väljade andmed. REST API kasutamisel tuleb teha andmete kättesaamiseks kaks eraldi päringut (Joonis 2). Esimese päringuga saadakse kätte konkreetsele id-le vastav *user* objekt. Teise päringuga tagastatakse *user* alamobjekt *friends*, mis on seotud kindla kasutajaga ehk *user*-iga.

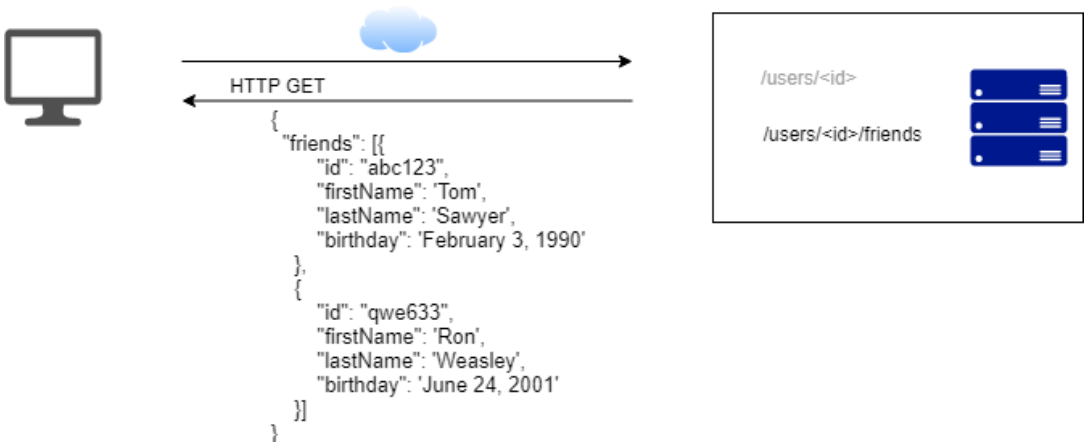
```
type User {
    id: Int!
    login: String!
    firstName: String
    lastName: String
    lastLoginAt: DateTime
    createdAt: DateTime!
    friends: [Friend]
}

type Friend {
    id: String!
    firstName: String
    lastName: String
    birthday: DateTime
}

type Query {
    users: [User]
    user(id: Int!): User
}
```

Joonis 1. GraphQL objektitüübid

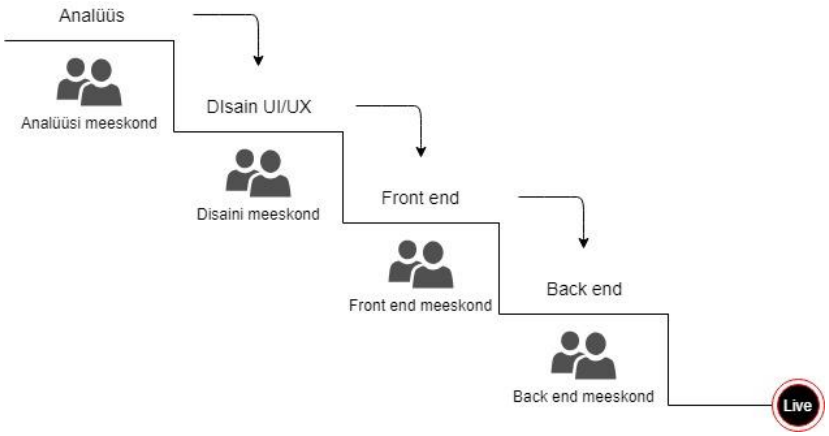




Joonis 2. REST API päring

2.2 *Coupled* ehk monoliitne arhitektuur

Coupled ehk monoliitses arhitektuuris asuvad rakenduse kasutajaliidese kiht ehk *front end*, loogikakiht ehk *back end* ja andmebaas ühes ja samas süsteemis. Kõik kolm komponenti on omavahel seotud ning loodud töötama kindlaks määratud rakenduses. Muutes ühte kolmest komponendist, tuleb ümber kirjutada ka teised kaks. [16] Seetõttu ei ole hea *Coupled* arhitektuuril põhineva rakenduse arendusprotsessis täita erinevaid arendusetappe üheaegselt, vaid pigem liikuda järgmise arendusetapi juurde siis kui eelmine on valmis (Joonis 3). Kliendipoolne kasutajaliides on loodud kasutades *HTML*-i, *CSS*-i ja *JavaScript*-i ning serveripoolne rakendus arendatakse kasutades programmeerimiskeeli nagu *PHP*, *JAVA* või *C#*. Loogikakiht tegeleb HTTP-päringutega, käivitab domeenispetsiifilist loogikat, otsib ja uuendab andmebaasis andmeid ning täidab brauserile saadetava kasutajaliidese vajalike andmetega. [17]



Joonis 3. *Coupled* arhitektuuril põhineva rakenduse arendusprotsess

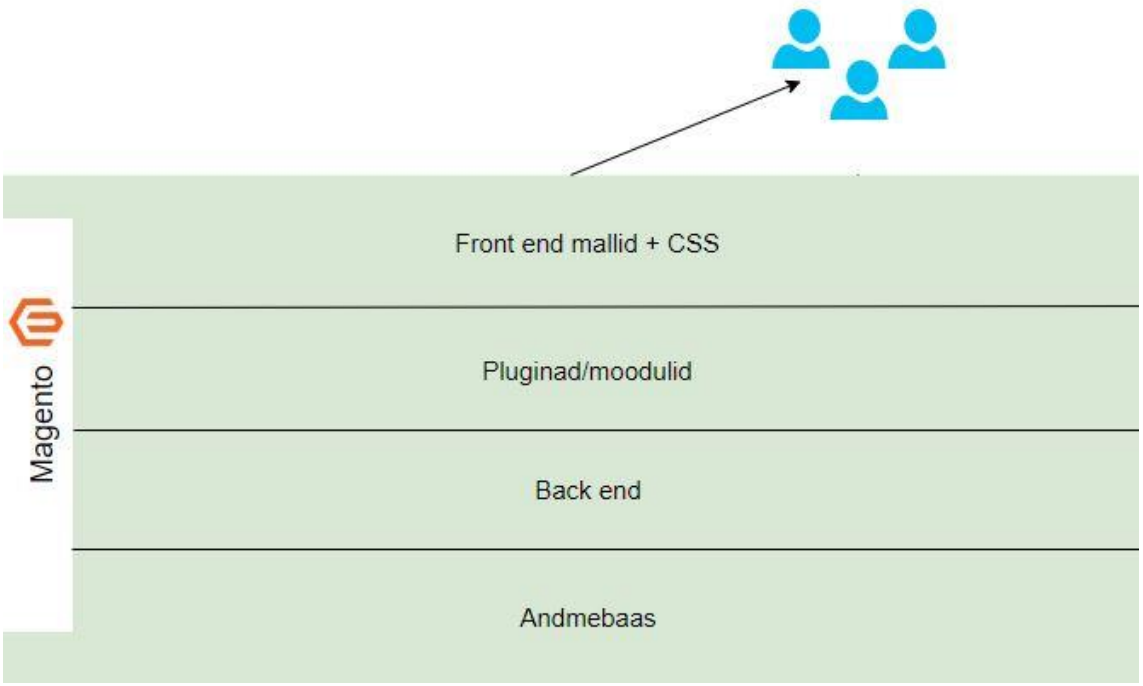
Coupled arhitektuuri plussid on järgmised:

- Kiire ja mugav kasutusele võtmine ning seadistamine;
- Vähene sõltuvus kolmandatest osapooltest ja nende teenustest;
- Täielik kontroll rakenduse üle; [18]
- Rakenduste loomine on lihtsam, sest funktsioonid asuvad ühes süsteemis;
- Rakendust on lihtsam serveris jooksutada; [16]
- Rakendus on kergemini testitav [19].

Coupled arhitektuuri miinused on järgmised:

- Uutel arendajatel läheb kaua aega, et kogu süsteem selgeks saada;
- Raskem teha suuri muudatusi, sest see võib mõjutada kogu süsteemi;
- Rakenduse suurenedes muutub see aeglasemaks; [16]
- Komponendid on süsteemis omavahel tihedalt seotud ja neid on peaaegu võimatu eraldada;
- Raskesti mõistetav, sest kõik asjad on omavahel seotud ja sõltuvuses; [17]
- Viga ühes komponendis kukutab tavaliselt kokku kogu süsteemi [19].

Kasutades *Coupled* arhitektuuril põhinevat CMS platvormi on samuti *front end* ehk kasutajaliidese kiht, *back end* ehk äriloogika kiht ja andmebaas tihedalt seotud ja asuvad samas süsteemis (Joonis 4). Sisu luuakse, hallatakse ja salvestatakse lehe *back end* osas. Lisaks hoitakse seal ka disaini- ja redigeerimisvahendeid. *Back end* ja andmebaas asuvad samas süsteemis *front end*-iga, mis edastab ja muudab sisu nähtavaks seadmetele ning lõppkasutajatele. Seega *Coupled* CMS-i puhul lehe arendajad kirjutavad ja avaldavad sisu samas süsteemis, mida veebilehe külastajad näevad. [20]

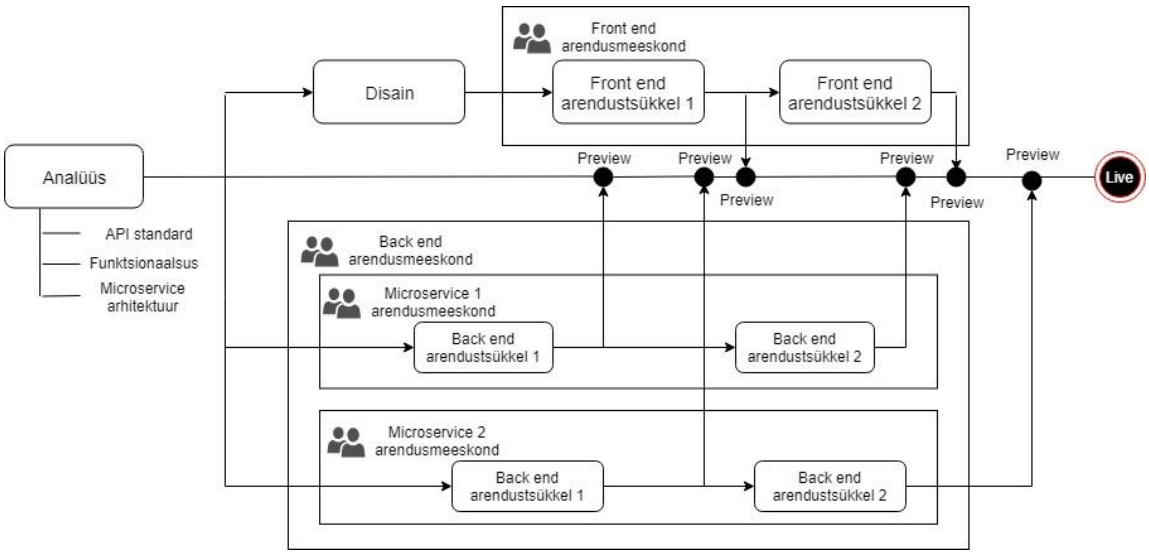


Joonis 4. *Coupled* arhitektuuril põhinev sisuhaldussüsteem

2.3 *Decoupled* ehk kihiline arhitektuur

Mõiste *Decoupled* tähendab kahte või enamat süsteemi, mis on võimelised omavahel suhtlema ilma, et nad asuksid samas keskkonnas. Süsteemid ei mõjuta üksteist ja ühel

süsteemil on tavaliselt vähe teavet mõne teise süsteemi kohta. Enamasti piirduvad need teadmised infoga mida on jagatud omavahel liideste kohta. *Decoupled* arhitektuur võimaldab teha muudatusi mistahes süsteemis ilma, et see mõjutaks teisi süsteeme. [21] Sõltumatus annab võimaluse *Decoupled* arhitektuuril põhineva rakenduse arendusprotsessi üles ehitada nii, et meeskonnad ei sõltu üksteisest (Joonis 5). Arenduse alguses lepivad meeskonnad analüüsi käigus kokku API standardi, rakenduses kasutatava funktsionaalsuse ja mikroserverite arhitektuuri. Seejärel on igal meeskonnal projektis oma osa mida arendada ning lõpuks pannakse mitmest erinevast osast kokku üks suur rakendus.



Joonis 5. *Decoupled* arhitektuuril põhineva rakenduse arendusprotsess

Decoupled arhitektuuri idee on jagada rakendus väikesteks omavahel ühendatud osadeks. See tähendab, et kasutajaliidese – ja loogikakiht ning andmebaas ei asu enam ühes suures süsteemis vaid on eraldatud erinevateks süsteemideks. [22] Rakendus on jagatud mikroteenusteks, kus igal teenusel on oma vastutusala (näiteks isikuandmete töötlemine, arveldus, autentimine) [23]. Osad teenused muudavad ka kasutajaliidese veebilehel nähtavaks. Koos töötamiseks suhtlevad mikroteenused omavahel API kaudu. *Decoupled* arhitektuur mõjutab oluliselt rakenduse ja andmebaasi vahelist suhtlust. Selle asemel, et jagada ühte suurt andmebaasi kõigi teenuste vahel on igal teenusel enda andmebaas. See võib kaasa tuua andmete dubleerimise, aga samas on võimalik igal hetkel teenus süsteemi küljest lahti ühendada ning rakendus töötab jätkuvalt edasi. [22]

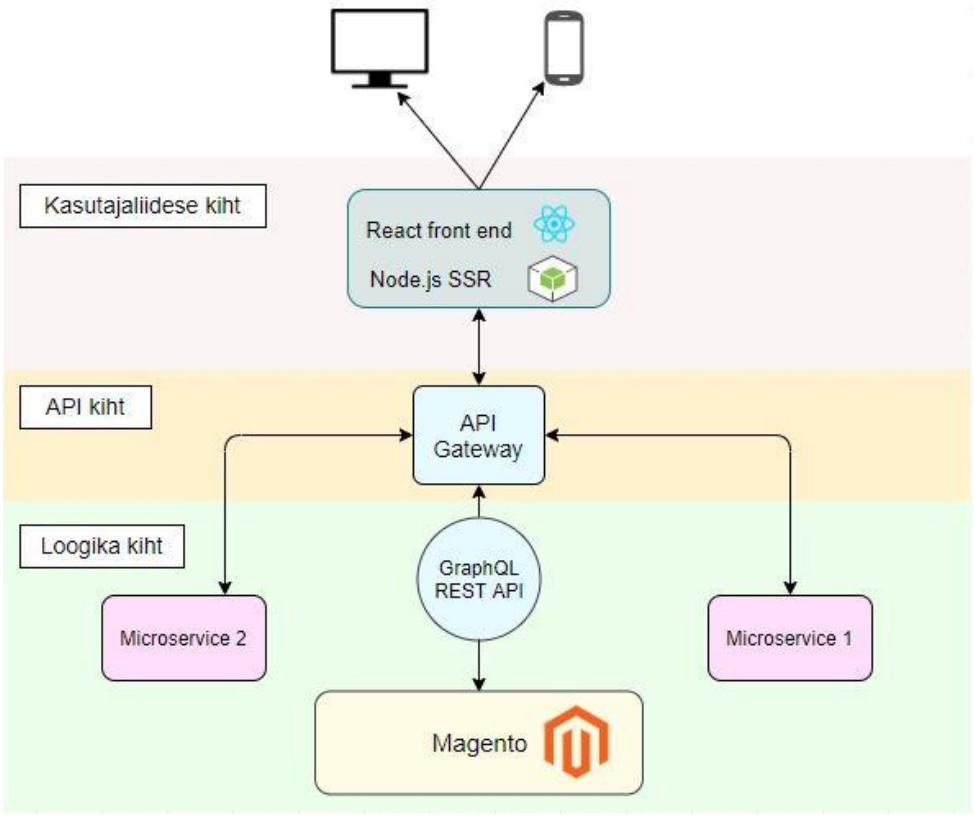
Decoupled arhitektuuri plussid on järgmised:

- Rakenduse ülesehitust on lihtsam mõista ja seetõttu on arendamine kiirem;
- Iga teenust on võimalik eraldi arendada ja see annab võimaluse organiseerida arendustegevust mitme arendaja vahel;
- Arendus keelte ja tehnoloogiate valik ei ole piiratud;
- Võimaldab luua keerulisi rakendusi, sest iga mikroteenus eraldiseisev;
- Rohkem skaleeruv ja võimalik ehitada suurte koormus taluvustega süsteeme; [22]
- Lihtne muuta ning testida koodi [18];
- Kui ühes teenuses tekib viga, siis ei kuku kogu rakendus kokku;
- Võimalus kasutada erinevate kolmandate osapoolte teenuseid ühe süsteemi osana;
- Sama koodi on võimalik kasutada ka teistes projektides. [16]

Decoupled arhitektuuri miinused on järgmised:

- Keerukam kasutusele võtta;
- Mitmele teenusele vastava päringu rakendamine on keeruline;
- Päringute tegemine, mis hõlmab mitut teenust vajab arendajate hoolikat koordineerimist. [24]

Vaadeldes *Decoupled* arhitektuuril põhinevat CMS-i on põhimõte sama. Veebilehe loogikakiht ehk *back end* ja kasutajaliidese kiht ehk *front end* on lahutatud erinevateks süsteemideks (Joonis 6). *Back end* osa on mõeldud sisu loomiseks ja salvestamiseks ning *front end* tegeleb nende andmete kasutamisega ja esitamisega kasutajaliidese kaudu. [20] Seega *front end*-i ei mõjuta kus ja kuidas sisu tekib. Antud arhitektuur annab vabamad käed tehnoloogia valikus ja vähendab sõltuvust ühest konkreetsest platvormist. [25] Kui sisu on *back end* poolel loodud või muudetud, siis kasutab *front end* paindlikke ja kiireid veebiteenuseid ning API-sid, et viia sisu mistahes seadmete kasutajaliideseni [20]. *Front end*-i on võimalik luua mitmel erineval viisil. Näiteks kasutades interaktiivseid *JavaScript* raamistike nagu *React* või *Angular*, staatiliste veebisaitide generaatoreid nagu *Gatsby*, mobiilirakenduste loomiseks *React Native* või teisi CMS-sse. Lehe sisu saab *front end* kätte veebiteenuse API kaudu, mis on enamasti *JSON* või *XML* formaadis ja mida edastatakse REST või GraphQL liidese kaudu. *Decoupled* arhitektuuri puhul kasutatakse andmebaasipõhist CMS-i *back end*-ina, mille kaudu arendaja saab luua, muuta ja salvestada lehe sisu. [26]



Joonis 6. *Decoupled* arhitektuuril põhinev sisuhaldussüsteem

2.4 *Headless* ehk peata arhitektuur

Kui kujutada ette, et CMS-i „*head* ehk pea“ on allsüsteem, mille kaudu tarbijale sisu esitletakse, siis *Headless* arhitektuur on see, mis jääb alles kui antud allsüsteem nii-öelda välja lülitatakse. *Headless* CMS on sisule orienteeritud andmeallikas, mille kaudu on võimalik luua sisupõhiseid rakendusi. *Headless* CMS-is on modelleerimis- ja redigeerimisvahendid, mis aitavad sisu kirjeldada, luua, muuta ja vajadusel sellega manipuleerida. [27]

Headless arhitektuur on tegelikult *Decoupled* arhitektuuri alamhulk ning oma üldise arhitektuuri poolest on nad täpselt samasugused. Lähtudes *Headless* ja *Decoupled* arhitektuuril põhinevast CMS-ist toimub mõlemas sisu haldamine ja salvestamine *back end*-is ning andmeid edastatakse andmebaasist veebiteenuste või API kaudu. Peamine erinevus tekib esitluskihis. Erinevalt *Decoupled* arhitektuurist ei ole *Headless* arhitektuuris eraldi määratletud *front end* süsteemi. [20] Kõige lihtsam on nende erinevust mõista nii: *Decoupled* CMS on ennetav – see valmistab sisu esitamiseks ette

ning seejärel suunab selle esitluskeskkonda. *Headless* CMS on reaktiivne – see loob sisu valmis ja siis ootab kuni mõni päring seda küsima tuleb. [27]

Decoupled CMS-is valmistatakse sisu *back end*-is ette ning seejärel on võimalik loodud sisu ennetavalt pakkuda ja esitleda erinevatele kanalitele. Seevastu *Headless* CMS on ainult sisul põhinev andmeallikas ja tal ei ole funktsionaalsust, et esitada kasutajale ise oma sisu. Sisu luuakse valmis ja oodatakse API päringut, mis edastaks selle rakendustele ja süsteemidele. See tähendab, et *Headless* arhitektuur on API mitte kasutajaliidese põhine. See võib sisu saata ükskõik millisesse internetiühendusega seadmesse või kanalisse. *Headless* arhitektuuri abil võib avaldada sama sisu veebilehele, mobiilirakendusele, nutikellale või muule seadmele, mis on ühenduses internetiga, sest sisu ei ole seotud eelnevalt määratletud kasutajaliidesega. [20]

3 E-kommerts sisuhaldussüsteem Magento

Kolmandas peatükis annab autor lühiülevaade Magentost. Toob välja olulisemad uuendused ja võimalused uues versioonis. Viib läbi kvalitatiivse uuringu ja analüüsi, mis põhineb kolmel intervjuul Eesti ettevõtetega, kes tegelevad Magento platvormile arendamisega. Uurib nende arvamust ja võimalikku hirmu põhjust uue versiooni ees. Teostab kvantitatiivse uuringu erinevaid *pluginaid* ja mooduleid arendavate ettevõtete hetkevõimalustest ning tuleviku arendusplaanidest seoses Magento uue versiooniga ja analüüsib saadud vastuseid.

3.1 Mis on Magento?

Magento on avatud lähtekoodiga e-kommerts platvorm, mis on arendatud PHP keeles. Magento tuli välja 2008. aasta 31. märtsil ja see on arendatud Zend Framework platvormile. Eesmärk oli aidata arendajatel kergema vaevaga luua e-kaubanduse veebilehti. Magento võimaldab veebipõhistel ettevõtetel hallata ja kontrollida oma e-poe ostukorvide süsteemi, funktsionaalsust, sisu ja välimust ning pakub vahendeid turunduse tegemiseks, otsingumootorite optimeerimiseks ja kataloogide haldamiseks. [28] Lisaks on võimalik Magento platvormi külge integreerida erinevaid valmis malle ja teemasid, et lihtsustada arendajate tööd ning muuta kliendi kasutajakogemus mugavamaks [29]. Magentol on kaks erinevat versiooni: Magento Community Edition, mis on kõigile tasuta ja Magento Enterprise Edition, mis on tasuline ning sisaldab rohkem võimalusi. Magento arvatakse olevat üks parimaid e-kaubanduse platvorme, mis tänapäeval saadaval on. [30] Kõigi sisuhaldussüsteemide võrdluses asus Magento 2018. aastalõpu seisuga neljandal kohal, omades 2,3% kogu turust. [31] Seda võib pidada ainult e-kommerts lahendusi pakkuva platvormi kohta kõrgeks tulemuseks. Magento e-poe platvormi kasutavad sellised tuntud ettevõtted nagu Nike, Ford, Jaguar, Helly Hansen, Land Rover, Björn Borg ja paljud teised [32].

3.2 Magento versioonide areng

2015. aasta novembris tuli välja Magento 2, mis oli suur areng edasi. Arhitektuuri poolelt toetas Magento 2 uusimaid *PHP*, *HTML* ja *CSS* versioone. Lehtede laadimiskiirus ning turvalisus suurenes märgatavalt võrreldes Magento 1 versiooniga. Lisaks eemaldati Magento 2 puhul kolmandate osapoolte loodud moodulite kasutamisel tekkinud konfliktid. Administraatori sektsiooni muudeti kasutajasõbralikumaks ja lihtsamaks ning lisati juurde võimalus teha veebipood skaleeruvaks teistele nutiseadmetele. [33]

Magento uusim versioon 2.3 tehti avalikuks 2018. aasta novembris ning tegemist oli taaskord suure arenguhüppega. Magento 2.3 sisaldab uusi tööriistu, mis parandavad ning lihtsustavad nii müüjate kui arendajate tööd. Kõige olulisem muudatus on võimalus luua progressiivseid veebirakendusi PWA Studio abil, mis teisisõnu annab võimaluse arendada *Decoupled* arhitektuuril põhinevat e-poodi. PWA (*Progressive Web App*) on uus *front end* tehnoloogia, millega on võimalik pakkuda seadmetüübile, näiteks mobiiltelefonile, vastavat rakendust, mis on kiire, integreeruv ja usaldusväärne. Progressiivne veebirakendus on ülesehitatud *JavaScript* raamistikule *React, Vue.js* või *Angular* ning sealt on võimalik teha REST API või GraphQL abil päringuid *back end* süsteemi, milleks on Magento. [34] Teine suur uuendus on võimalus kasutada andmete päringukeelt GraphQL. See on paindlik ja hästi toimiv tööriist arendajale, mis on integreeritud Magento API ökosüsteemi [35]. GraphQL annab PWA-le võimaluse otsida ja nõuda täpselt vajaminevaid andmeid ning ühendada need andmed samas päringus. Teisisõnu tuleb andmete kättesaamiseks teha vähem API päringuid ja seega on andmemahud väiksemad. GraphQL kasutamine võimaldab Magento PWA-l olla kiire isegi aeglases võrguühenduses. Lisaks muudeti API päringud asünkroonseks, mis tähendab, et peale päringu tegemist ei pea rakendus ootama serveripoolset vastust vaid võib taustal toimingutega jätkata. See muudab kogu süsteemi toimimise kiiremaks. Turvalisuse poole pealt lisati administraatori paneeli sisselogimiseks kaheastmeline autentimine ning Google *reCAPTCHA*. Viimane kaitseb veebilehte sisselogimise, registreerimise ja kontaktivormi täitmise kuritarvitamise eest. Tegemist on Google teenusega, mis kasutab täiustatud riskianalüüsi meetodeid ja hindab erinevaid aspekte, mis eristavad inimest robotist. [34]

3.3 Individuaalintervjuude tulemuste analüüs

Sissejuhatuses püstitatud esimesele uurimisküsimusele vastuse leidmiseks viis autor läbi kvalitatiivse analüüsi, mis põhineb kolmel intervjuul Eesti juhtivate IT-ettevõtetega, kes kasutavad arendusteenuste pakkumisel Magento platvormi. Intervjuu küsimuste koostamisel lähtus autor uurimisküsimusest – mis põhjustel Magento ettevõtted, kes on aastaid tegelenud Magento platvormile e-poodide loomisega, ei soovi uuele versioonile üle minna. Küsimustik töötati välja koostöös arendusettevõttega, kus autor töötab. Eesmärk oli teada saada, millisel arvamusel on Eestis Magento platvormi kasutavad ettevõtted uuest versioonist ja millised kogemused on neil *Decoupled* arhitektuuril põhineva e-poega. Intervjuud toimusid ajavahemikul 10.04.2019 – 03.05.2019. Vastajad andsid loa vestlust lindistada, kuid palusid töös tagada anonüümsus. Autor tähistab analüüsi osas intervjueeritavad ettevõtted numbritega 1, 2 ja 3. Intervjuu küsimustikku on võimalik vaadata Lisast 1 ja transkriptsiooni Lisast 2.

Intervjuudest selgus, et ettevõte 1 on tegelenud Magento platvormile arendamisega neli aastat, ettevõte 2 kümme aastat ja ettevõte 3 seitse aastat. Kõik kolm ettevõtet kasutavad e-poodide loomisel Magento 2 versioone, kuid varasemast ajast on portfooliosse jäänud ka Magento 1 peale tehtud e-poode. Vanemaid versioone uuendatakse ja viiakse vastavalt klientide soovile ja rahalisele võimekusele üle Magento 2 platvormile.

Ettevõtted 1 ja 3 ei ole enda loodavates lahendustes kasutusele võtnud *Decoupled* arhitektuuri. Peamise põhjusena toodi välja, et see ei ole kliendile hetkel ärilises plaanis kasulik ning läheb talle kalliks maksma, sest suur osa arendaja tööajast kulub uue versiooni ja arhitektuuri tundma õppimisele ning keerulise keskkonna tööle saamisele ja seadistamisele. Siiski on mõlemad ettevõtted valmis turuga kaasa liikuma ning tulevikus pakkuma *Decoupled* arhitektuuri kui nõudlus selle järele suureneb ja on saadaval rohkem erinevaid näiteid ja dokumentatsioone. Ettevõte 3 arendab endale ettevõtte sisest projekti, mis põhineb *Decoupled* arhitektuuril ja mille *front end* kiht ehitatakse kasutades *JavaScript* raamistikku *React*. Nende eesmärk on juba praegu õppida uut versiooni ja arhitektuuri paremini tundma, et seda tulevikus klientidele pakkuda. Ettevõte 2 hakkas esimest *Decoupled* arhitektuuril põhinevat e-poodi arendama paar kuud peale Magento versiooni 2.3 välja tulemist ning teine samal arhitektuuril põhinev projekt oli neil intervjuu tegemise hetkel läbirääkimiste faasis. *Front end* kiht ehitatakse *JavaScript* raamistikule *Vue.js* ning kolmandate osapoolte pakutavaid *pluginaid* ja

mooduleid ei ole veel üritatud külge integreerida, sest projekt on algusfaasis. Ettevõtte 2 väitel on kliendid huvitatud *Decoupled* arhitektuurist tuleviku perspektiive silmas pidades, sest antud arhitektuuri puhul on võimalik rakendus jagada erinevateks mikroserveriteks, mis muudab arenduse lihtsamaks.

Väitega, et hetkel ei ole majanduslikult otstarbekas ehitada uusi e-poode *Decoupled* arhitektuurile, olid nõus ettevõtted 1 ja 3. Ettevõte 1 tõi välja, et kui e-kommerts turul tekib antud arhitektuuri järele nõudlus, siis tekib ka dokumentatsioon. Hetkel on aga nõudlus väike ja seega on dokumentatsiooni ja reaalsete näidete tekkimise protsess aeglane, sest Magento kogukond ei panusta palju aega antud lahenduse uurimisele. See viib põhjuseni, mille tõi välja ettevõte 3, et *Decoupled* arhitektuuri peale ehitamine nõuab hetkel kliendilt liiga suurt investeeringut võrreldes tavalise monoliitse lahendusega, sest *Decoupled* arhitektuuri arendamine on dokumentatsiooni ja tehnilise toe vähesuse tõttu keeruline ja aeganõudev. Ettevõte 2 juba kasutab *Decoupled* arhitektuuri ja pigem ei nõustunud eelmainitud väitega, tuues välja, et Magento versioon 2.3 on stabiilsuse saavutanud ja laiemat pilti vaadates on otstarbekam Magento baasrakenduse külge integreerida erinevaid mikroservereid.

Kõik intervjueeritud ettevõtted näevad *Decoupled* arhitektuuri riskidena *pluginate* ja moodulite toe puudumist. Ettevõtted 1 ja 2 toovad riskina välja ka tööjõu kompententsi puudulikkuse, kuid selle probleemi leevendamiseks toob ettevõte 1 välja, et kuna antud arhitektuuri puhul on võimalik *back end* ja *front end* üksteisest eraldada, siis *front end* arendaja ei pea tingimata olema Magento kompetentsiga inimene. Ettevõte 3 toob riskina välja enda algse investeeringu, mis tähendab, et lisaks hilisemale garantiiperioodile tuleb esimeste projektide tähtajast üle minemisel ettevõttel arendajate ületunnid kinni maksta. Arendajate pikem projektiga tegelemise aeg võib tuleneda uue arhitektuuri kasutamise tundmaõppimisest ja keskkonna seadistamisest. Lisaks tuleb arhitektuur luua nii, et e-pood vastaks Magento versiooni uuendustele ja iga järgneva uuendusega ei peaks suuri muudatusi tegema.

Decoupled arhitektuuril põhineva Magento suurima plussina toob ettevõte 1 välja vabaduse disainida ja luua kasutajaliides täpselt selliselt nagu klient soovib. Ettevõte 2 arvates on põhiline pluss Magentos sisalduv funktsionaalsus, mida on võimalik *Decoupled* arhitektuuri puhul kasutada ja seega olla kliendile kättesaadavam ja odavam, sest ei pea kogu funktsionaalsuse arendamist alustama algusest. Ettevõte 3 toob peamise

plussina välja võimaluse Magentole *Decoupled* arhitektuuril põhineva e-poe loomise. Seega saavad ettevõtted olla kaasaegsed, liikuda ühes tulevikutrendidega ja kasutada arendamiseks moodsat arhitektuuri.

Toetudes intervjuudest saadud vastustele väidab autor, et *Decoupled* arhitektuuril põhineva Magento e-poe tegemisel on hetkel veel palju riske ja lahendamata küsimusi ning antud arhitektuuri kasutamine nõuab nii kliendilt kui ettevõttelt suurt investeeringut. Intervjuudest selgub, et on kliente kes tuleviku perspektiive silmas pidades on nõus tegema alguses suurema investeeringu. Seega ettevõttele, kes juba on võtnud suuna *Decoupled* arhitektuuril põhineva e-poe loomisele, võib investeering lähitulevikus kasumlikumaks osutuda kui neile kes peavad isiklikku investeeringut hetkel majanduslikult ebaotstarbekaks.

3.4 Ankeetküsitluse tulemuste analüüs

Kvantitatiivse uurimisviisi läbiviimiseks võttis autor meili teel ühendust ja saatis küsimustiku 20-le erinevale IT-valdkonna ettevõttele, kes tegelevad Magento *pluginate* ja moodulite arendusega. Küsimustik on leitav Lisast 3 ja küsitluses osalenud ettevõtete nimed Lisast 4. Küsimuste koostamisel lähtus autor eesmärgist ja teisest uurimisküsimusest - millised on *pluginate* ja moodulite tootjate hetkevõimalused ning tulevikuplaanid seoses Magento uue versiooniga. Küsimused valiti koostöös arendusettevõttega kus autor töötab, et aru saada uue Magento versiooni lahenduse riskidest ja kuidas Magento *pluginate* ja moodulite arendajad on neid riske hinnanud ja maandanud. Järgnevalt analüüsib autor kvantitatiivses osas saadud andmeid, et leida vastus uurimisküsimusele.

Küsitlusest selgus, et kahekümnest ainult kolme ettevõtte *pluginad* ja moodulid toetavad *Decoupled* arhitektuuri kasutamist REST API ja GraphQL päringute kaudu. Neist kolmest kahel ettevõttel on tugi hetkel ainult *back end* moodulite jaoks, kuid paari kuu jooksul loovad võimalused ka *front end* moodulitele. Mitte ükski kolmest ettevõttest ei ole teinud ega plaani teha näidislahendusi kuidas integreerida arendaja poolt loodud *front end* nende *plugina*/mooduli külge. Ülejäänud ettevõtete *pluginad* ja moodulid ei toeta hetkel *Decoupled front end* arhitektuuri kasutamist, kuid enamik neist plaanivad luua selle võimaluse lähitulevikus, mis toetaks nii REST API kui GraphQL andmete päringukeeli. Viiel küsimustikule vastanud ettevõttel on plaanis luua õpetused või

näidislahendused *pluginate* ja moodulite *front end*-i külge integreerimiseks, mis on loodud vastavalt *React, Angular* või *Vue,js* raamistikku kasutades. Kaks ettevõtet on valmis kliendile tegema individuaalse pakkumise ning looma *Decoupled* arhitektuuri kasutamiseks vajalikud võimalused. Põhjuseid, miks ei ole loodud võimalusi *Decoupled* arhitektuuri kasutamiseks, oli mitmeid. Peamine põhjus, et ressurssi on vaja suunata mujale ning põhiline fookus on olemasolevate *pluginate* ja moodulite edasiarendamisel ja täiendamisel, sest Magento versioonid uuenevad pidevalt ja seetõttu on vaja uuendada ka *pluginate ja* moodulite versioone. Lisaks toodi välja, et *Decoupled* arhitektuuri vastu on hetkel väike huvi, uus Magento versioon 2.3 on lühikest aega väljas olnud ning informatsioon ja dokumentatsioon nii uue Magento versiooni kui *PWA Studio* kasutamise kohta on puudulik.

Toetudes eelnevatele uuringutele ning *pluginaid* ja mooduleid arendavate ettevõtete vastustele on autor nende põhjendustega nõus. Magento 2.3 versioon on olnud lühikest aega saadaval, Magento enda poolt loodud dokumentatsiooni on veel vähe ja Magento kogukond, mis koosneb asjaarmastajatest ja isehakanud arendajatest on võrdlemisi väike, seega liigub informatsioon aeglaselt. Info vähesuse tõttu peaksid *pluginate* ja moodulite arendajad suunama rohkem tööjõuressurssi proovimisele ja testimisele kuidas *Decoupled* arhitektuuril põhinev Magento töötab. Tulemuste analüüsi järelduste põhjal saaksid nad hakata välja arendama võimalust kasutada antud arhitektuuri enda loodud *pluginates*/moodulites. Ettevõtete vastustest selgub, et enamikel neist puudub selleks ressurss.

4 Sisuhaldussüsteem Wordpress ja selle e-kommerts moodul WooCommerce

Neljandas peatükis uurib autor tuntud CMS-i platvormi Wordpress ja selle e-poe moodulit Woocommerce, et teada saada, kas antud töö tulemused on üldistatavad ka teistele vabavaralistele platvormidele. Analüüsib millist arhitektuuri nad hetkel kasutavad, kuidas versioonid on muutunud, milline on olnud arhitektuuride areng ja kuidas versioonide areng on mõjutanud kasutajate hulka. Võrdleb kas kolmandas peatükis analüüsitud Magento areng on olnud sarnane Wordpressi ja selle e-kommerts mooduli WooCommerce arengule, et teha järeldused Magento arengusuuna osas.

4.1 Mis on Wordpress ja WooCommerce?

Wordpress on avatud lähtekoodiga ja tasuta pakutav sisuhaldussüsteem, mis on loodud kasutades *PHP*-d ja *MySQL*-i [36]. Selle esialgne eesmärk oli pakkuda inimestele võimalust luua endale lihtsalt ja mugavalt veebis baseeruv blogi. Tänaseks on Wordpressi platvormist välja kasvanud üks maailma populaarsemaid veebilehtede loomiseks kasutatavaid sisuhaldussüsteeme. [37] Kõigi sisuhaldussüsteemide võrdluses asub Wordpress esimesel kohal omades 59.7% kogu turust [31]. Sellele on kaasa aidanud aastate jooksul tekkinud suur kogukond, kättesaadav dokumentatsioon ning inimeste poolt arendatud tuhanded moodulid, kujundused ning teemad, mis lihtsustavad erineva ning unikaalse disainiga veebilehtede loomist. Wordpressi on lihtne kasutada, sest see ei eelda programmeerimise oskuseid. Inimene logib sisse administreerimispaneeli, valib välja sobiva kujunduse või malli, lisab moodulite abil vajaliku funktsionaalsuse ning täidab veebilehe väljad vajalike andmetega. [38]

WooCommerce on 2011. aastal välja lastud tasuta Wordpressi moodul, mis võimaldab luua e-kommerts funktsionaalsusega veebilehti [39]. Integreerides WooCommerce mooduli Wordpressi külge lisab see minimaalse e-poe toimimiseks vajaliku funktsionaalsuse, milleks on toodete lisamine ja haldamine, tellimuste töötlemine ja ostukorvi funktsionaalsus. Baasfunktsioone on võimalik täiendada rohkem kui

kolmesaja erineva tasulise WooCommerce laiendusega, näiteks toodete broneerimine, kliendi andmete kogumine ja jälgimine, toodete tagastamine, garantii jne. WooCommerce moodulit ja selle laiendusi arendab igapäevaselt 350 arendajat, kes on kokku koondunud WooThemes nimelise ettevõtte alla. [40]

4.2 Wordpressi ja WooCommerce areng võrreldes Magentoga

Wordpress tehti kõigile kättesaadavaks 2003. aasta 27. mail. Sellest ajast on välja lastud 284 versiooni uuendust mistõttu on raske välja tuua, millised neist on Wordpressi arengut kõige enam mõjutanud. Esimestel aastatel prooviti iga uuendusega aina paremaks muuta administreerimispaneeli. Loodi esimesed *pluginad* ning arendati juurde funktsionaalsust. 2005. aastal tuli välja uuendus, millega oli võimalik luua blogile lisaks eraldiseisev staatiline leht. Kuna 2012. aastal muutusid populaarseks erinevad nutiseadmed, siis lisati juurde võimalus teha veebilehed skaleeruvaks vastavalt ekraani suurusele. Iga versiooni uuendusega muudeti Wordpressi turvalisemaks ning võimalusel uuendati programmeerimiskeelte versioone. [41] *Decoupled* arhitektuurist lähtudes oli kõige olulisem 2016. aasta kui Wordpressis tekkis võimalus teha REST API päringuid, et saata ja vastu võtta *JSON* formaadis andmeid [42]. Tänaseks on olemas Wordpressi *plugin*, mis võimaldab teha ka GraphQL päringuid arendaja enda poolt loodud *front end*-ist Wordpressi [43]. Lisaks on olemas ka WooCommerce-l võimalus teha REST API abil päringuid ning kogukonna liikmed on välja arendanud *plugina*, mis võimaldab kasutada ka GraphQL [44][45].

Üldjoontes on Magento ja Wordpress arenenud samas suunas, viimane on olulised uuendused sisse toonud paar aastat varem ning see on tingitud sellest, et Wordpressil on suur kogukond, mille liikmetel on lihtsa vaevaga võimalik vastavalt vajadusele ise erinevaid *pluginaid* juurde arendada. Mõlemad on pannud suurt rõhku administreerimispaneeli mugavamaks ja kiiremaks muutmisele, muutnud veebilehtede loomist ja haldamist turvalisemaks ning lisanud juurde võimaluse teha veebileht erinevate seadmetele skaleeruvaks. Täna on võimalik mõlema platvormi peale luua *Decoupled* arhitektuuril põhinev veebileht, sest nii Magento kui Wordpress ja selle e-kommerts *plugin* WooCommerce toetavad REST API ja GraphQL abil päringute tegemist. Wordpressi kasutades on seda lihtsam teha, sest see võimalus on pikemalt saadaval olnud ja seetõttu on internetist võimalik leida rohkem õpetusi ja

dokumentatsioone. Magento puhul on *Decoupled* arhitektuuril põhinev veebipood veel lapsekingades ja saadaval on ainult Magento enda dokumentatsioon. Siiski võib järeldada, et Magento liigub *Decoupled* arhitektuuri kasutusele võtmisega õiges suunas. Selle populaarseks muutumine aga võtab aega, sest Magentol on võrreldes Wordpressiga väike kogukond ja seetõttu tekib ja liigub informatsioon aeglasemalt. Võttes arvesse, et Magento ja Wordpressi areng on olnud sarnane väidab autor, et töös uuritud arhitektuurid ja arenduspraktikad on rakendatavad ka Wordpressile ning teistele vabavaralistele CMS ja e-kommerts platvormidele, sest nad on oma olemuselt sarnased.

5 *Decoupled* arhitektuuril põhineva prototüübi realisatsioon Magentos

Viiendas peatükis käsitleb autor Magento uue versiooni peale tehtud prototüüpi. Selgitab prototüübi ülesehitust ja põhjendab, miks osutus valituks just see funktsionaalsus, mis prototüübil küljes on.

5.1 Realiseeritava funktsionaalsuse valimine

Klient saatis ettevõttele nimekirja kogu funktsionaalsusest, mida ta soovib realiseeritavas e-poes näha. Antud listist valis autor enda jaoks kõige olulisemad funktsionaalsused välja, ilma milleta on e-poel keeruline eksisteerida ning integreeris need rakenduse külge. Valituks osutusid järgnevad funktsionaalsused – toodete pärimine Magentost, toodete kuvamine veebilehel, toodete otsimine veebilehel, ostukorvi loomine, toote lisamine ostukorvi ja ostukorvis oleva toote kuvamine. Autor ei valinud ühtegi funktsionaalsust, mida on võimalik realiseerida kasutades kolmandate osapoolte poolt arendatavaid *pluginaid* ja mooduleid, sest eelnevast uuringust selgus, et suur osa neist ei toeta hetkel *Decoupled* arhitektuuri ning kõik *pluginad* ja moodulid on tasulised.

5.2 Prototüübi kirjeldus

Rakendus töötab dockeris, *front end* on loodud kasutades *JavaScripti* raamistikku *React* ja käitussüsteemi *Node.js* ning *back end* osa täidab Magento. API päringute tegemiseks kasutatakse päringukeeli GraphQL ja REST API. Arenduskeskkonnaks kasutati rakendust *WebStorm*. Andmed mida veebilehel kuvatakse ja kasutatakse on Magento platvormis sisalduvad testandmed. Prototüübi visuaalid on leitavad Lisast 5.

Magento käivitamiseks dockeris kasutatakse järgnevat käsku.

```
docker-compose up -d
```

Rakenduse *front end* käivitamiseks kasutatakse alljärgnevat *Node.js* käsku ning rakendus on kättesaadav brauseris aadressil *localhost:3000*.

```
npm run start
```

5.2.1 Rakenduse üldine ülesehitus

Antud peatükis kirjeldatakse veebilehe üldist ülesehitust ning tutvustatakse vajalikke koodijuppe, et aru saada kuidas rakendus töötab. Rakenduse käivitamisel toimub serveri poolel veebilehe renderdamine ehk veebilehe *HTML* luuakse valmis serveri poolel (*Server Side Rendering*). Sel hetkel luuakse klient (*client*), mille abil on võimalik serverist andmeid küsida (Joonis 7). Selleks kasutab autor Apollo platvormi loodud komponenti *ApolloClient*, mis võimaldab GraphQL abil luua klient rakendusi. Järgnevalt antakse *ApolloProvider* nimelise komponendi sisse eelnevalt loodud klient ning siis luuakse konstant nimega *newHTML*, kuhu asendatakse kõik päritud andmed ning saadetakse *public* kaustas olevasse *index.html* faili (Joonis 8), kus on sama id-ga *div*.

```
const client = new ApolloClient({
  link: new createHttpLink({
    uri: 'http://localhost/graphql'
  })
});

const Application = () => (
  <ApolloProvider client={client}>
    <ReduxProvider store={store}>
      <StaticRouter context={context} location={url}>
        <Route path="/" render={props => <App
         type="Server" {...props} />} />
      </StaticRouter>
    </ReduxProvider>
  </ApolloProvider>
);
```

Joonis 7. Kliendi loomine

```
const stringWithData = await getDataFromTree(<Application />);
const newHTML = html.replace(
  '<div id="root"></div>',
  `<div id="root">${stringWithData}</div>`
);
```

Joonis 8. *HTML* loomine ja asendamine

5.2.2 Toodete pärimine ja kuvamine

Toodete pärimiseks Magentost kasutab autor GraphQL päringut (Joonis 9). Selleks luuakse konstant *GET_PRODUCTS,* kus määratakse ära, millist infot soovitakse toodete kohta saada. *Products* väljal täpsustakse, milliseid tooteid ja kui palju päring peab tagastama. Jooniselt 8 on näha, et otsingusõna on *Scarf* ehk sall ja soovitakse saada kümme toodet. Lisaks sisaldab päring veel toote nime, pilti, hinda ja sku-d ehk tootepõhist koodi. Toodete kuvamiseks kasutatakse komponenti *Query* (Joonis 10), mida on võimalik rakendada joonisel 6 näidatud *ApolloProvider* komponendi olemasolul. *Query* komponendi sees käiakse (*map*) kogu toodete list (*array*) läbi ning kuvatakse kõik päritud tooted juhul, kui ei tekkinud viga (*error*) ja laadimine (*loading*) on lõppenud.

```
const GET_PRODUCTS = gql`
query {
  products(search: "Scarf", pageSize: 10) {
    total_count
    items {
      name
      sku
      image
      price {
        regularPrice {
          amount {
            value
            currency
          }
        }
      }
    }
  }
}`;
```

Joonis 9. Toodete pärimine GraphQL abil

```
<Query query={GET_PRODUCTS}>
  {({ loading, error, data }) => {
    if (loading) return 'Loading...';
    if (error) return `Error! ${error.message}`;
    return (
      <ul className="Product__list">
        {data.products.items.map(product => (
          <ProductItem key={product.sku} product={product}/>
        ))}
      </ul>
    );
  }}
</Query>
```

Joonis 10. Toodete kuvamine

5.2.3 Toodete otsing

Toodete otsingu üldine funktsionaalsus sarnaneb toodete pärimise ja kuvamisega. GraphQL päringu osas tekib erinevus *products* väljas, kus toote otsingusõna asendatakse muutujaga *searchValue* (Joonis 11). *Query* komponenti lisatakse juurde väli nimega *variables* ning antud välja sees pannake eelnevalt loodud muutuja väärtuseks kasutaja poolt otsinguväljale sisestatud toote nimi (Joonis 12). Nüüd on muutuja väärtuseks otsingusse sisestatud toote nimi ja selle otsingusõnaga tehakse päring.

```
const GET_PRODUCTS = gql`
query Products($searchValue: String!){
  products(search: $searchValue, pageSize: 10) {
    ...
    ...
    ...
  }
}`;
```

Joonis 11. Muutuja loomine ja asendamine

```
<Query query={GET_PRODUCTS}
       variables={{ searchValue: this.state.inputValue }}>
       ...
       ...
       ...
</Query>
```

Joonis 12. Muutuja väärtustamine otsingusõnaga

5.2.4 Ostukorvi loomine

Ostukorvi funktsionaalsuse loomiseks määratakse kõigepealt REST API lõpp-punkt (*endpoint*), kuhu hakatakse päringuid tegema (Joonis 13). Seejärel luuakse funktsioon *cartCreateRequest*, kus kontrollitakse kas kasutajal on ostukorv juba olemas (Joonis 14). Kui ei ole, siis tehakse *guestCartCreate()* funktsiooni sees POST päring ja luuakse ostukorv ning seejärel tehakse *guestCartGet()* funktsiooni sees GET päring, kuhu lisatakse eelnevalt loodud ostukorvi id (*cartId*).

```
const magentoGuestCart = axios.create({
  baseURL: 'rest/V1/guest-carts/',
  headers: {
    'Content-type': 'application/json',
  },
});
```

Joonis 13. Ostukorvi REST API päringu *endpoint* määramine

```
const cartCreateRequest = () => async (dispatch) => {
  dispatch({ type: CART_CREATE_REQUEST });
  const storedCart = JSON.parse(window.localStorage.getItem('cart'));

  if (storedCart && storedCart.cartId && storedCart.quoteId) {
    dispatch({ type: CART_CREATE_SUCCESS, payload: storedCart });
  } else {
    try {
      const guestCartCreateResponse = await guestCartCreate();
      const guestCartGetResponse = await guestCartGet(
        guestCartCreateResponse.data);

      const cart = {
        cartId: guestCartCreateResponse.data,
        quoteId: guestCartGetResponse.data.id,
      };

      window.localStorage.setItem('cart', JSON.stringify(cart));
      dispatch({ type: CART_CREATE_SUCCESS, payload: cart });
    } catch (error) {
      window.localStorage.removeItem('cart');
      dispatch({ type: CART_CREATE_FAILURE, payload: error });
    }
  }
};
```

Joonis 14. Ostukorvi loomise funktsioon

5.2.5 Toodete lisamine ostukorvi

Toodete lisamiseks ostukorvi luuakse funktsioon *cartAddRequest* (Joonis 15), mille sees käivitatakse funktsioon *guestCartAdd.* Viimasena mainitud funktsioonis tehakse POST päring, kus antakse kaasa eelnevalt loodud ostukorvi id (*cartId*) ja lisatud toote andmed.

```
const cartAddRequest = item => async (dispatch, getState) => {
  dispatch({ type: CART_ADD_REQUEST });
  const { cartId } = getState().cart;

  try {
    const cartItem = {
      cartItem: {
        quote_id: cartId,
        sku: item.sku,
        qty: item.qty,
      },
    };

    const guestCartAddResponse = await guestCartAdd(cartId, cartItem);
    dispatch({ type: CART_ADD_SUCCESS, payload: guestCartAddResponse
});
  } catch (error) {
    dispatch({ type: CART_ADD_FAILURE, payload: error });
  }
};
```

Joonis 15. Toodete lisamine ostukorvi

5.2.6 Ostukorvis olevate toodete kuvamine

Ostukorvi lisatud toodete kättesaamiseks luuakse funktsioon *cartGetRequest*, kus käivitatakse funktsioon *guestCartGet*, mille sisse antakse ostukorvi id (cartId) (Joonis 16). Funktsioonis *guestCartGet* tehakse GET päring, et saada kätte ostukorvis olevad tooted. Seejärel käiakse (*map*) kõik ostukorvis olevad tooted läbi ning kuvatakse need veebilehel (Joonis 17).

```
const cartGetRequest = cartId => async (dispatch) => {
  dispatch({ type: CART_GET_REQUEST });

  try {
    const guestCartGetResponse = await guestCartGet(cartId);
    dispatch({ type: CART_GET_SUCCESS,
               payload: guestCartGetResponse.data });
  } catch (error) {
    window.localStorage.removeItem('cart');
    dispatch({ type: CART_GET_FAILURE, payload: error });
  }
};
```

Joonis 16. Ostukorvi ja selles olevate toodete pärimine

```
renderCartItems = () => {
    const { isLoading, items } = this.props.cart;

    if (isLoading) {
      return 'Loading...';
    }
    const cartItems = items.map(item => (
      <CartItem key={item.sku} item={item} />
    ));
    return cartItems;
  }

  render() {
    return (
      <main className="Cart">
        <h1>Cart</h1>
        <ul className="Cart__list">
          {this.renderCartItems()}
        </ul>
      </main>
    );
  }
}
```

Joonis 17. Ostukorvi sisu kuvamine

5.3 Prototüübi järeldused

Kogu rakenduse ülesseadmine *dockerisse* ja vajalike programmide seadistamine arenduse alustamiseks osutus oodatust raskemaks ning autor tegi seadistamiseks koostööd pikemaajalist kogemust omava arendajaga. Prototüübi funktsionaalsusest osutus kõige keerulisemaks e-poe üldine ülesehitus, mis töötab klient - server põhimõttel, sest autor polnud seda varem teinud. Raskusi valmistas ka ostukorvi

funktsionaalsuse tegemine, sest tuli teha erinevaid päringuid ning loogika väljamõtlemine võttis aega. Prototüübi tegemisel märkas autor, et päringute tegemised võtavad kaua aega ning laadimiprotsessid kestavad vahel mitu sekundit. Lisaks tekkisid paar korda probleemid keskkonna endaga kui see ei käivitunud või ei teinud päringuid korrektselt, kuigi autor ei olnud ühtegi muudatust teinud. Peale *dockerile restardi* tegemist ja rakenduse uuesti käivitamist hakkas prototüüp tööle. Autor arvab, et *Decoupled* arhitektuuril põhineva e-poe loomine Magentos on keeruline ja aeganõudev protsess, sest puuduvad näited, Magento dokumentatsioonist probleemidele vastuste ja lahenduste leidmine võtab aega ning autor pidi ise erinevaid asju proovima ja testima, et funktsionaalsus tööle saada.

6 Kokkuvõte

Bakalaureusetöö eesmärk oli selgitada välja millise arhitektuurse versiooni peale e-pood luua ning millised on võimalikud riskid uue *Decoupled* arhitektuuri rakendamisel e-kommerts platvormile Magento. Eesmärgi täitmiseks püstitas autor kolm uurimisküsimust, millele leidis vastused läbiviidud individuaalintervjuude, küsitluse tulemuste ning tehtud prototüübi analüüsimisel.

Teises peatükis selgitas autor teemaga seonduvaid mõisteid, analüüsis CMS-i arhitektuure ning tõi välja nende plussid ja miinused. CMS-i arhitektuure on kolm – *Coupled*, *Decoupled* ja *Headless.*

Kolmandas peatükis andis autor lühiülevaate Magentost, selle versioonide arengust ning tõi välja olulisemad uuendused ja võimalused versioonis 2.3. Individuaalintervjuude ja ankeetküsitluse tulemuste analüüsist leidis autor vastused kahele püstitatud uurimisküsimusele. Kvalitatiivsest uuringust selgus, et Magento uuele versioonile üleminek tähendab suurt ajakulu arendajale, kes hakkab uut versiooni ja arhitektuuri tundma õppima ning seadistama, mis omakorda tõstab teenuse hinda kliendi jaoks. Oma rahalisest võimekusest sõltuvalt otsustab klient kas soovib sellesse investeerida. Tulevikuperspektiive arvestades on majanduslikult otstarbekas ehitada e-poode *Decoupled* arhitektuurile ning arendusettevõtted on valmis nõudlusega kaasa minema ja teenust pakkuma kui uue versiooni kohta osutub kättesaadavamaks rohkem informatsiooni, näiteid ja dokumentatsiooni. Hetkel on aga lahendamata küsimusi ja mitmeid riske *Decoupled* arhitektuurile üleminemisel, näiteks *pluginate* ja moodulite toe puudumine, esialgne investeering tundub arendusettevõttele liiga suur ja riskantne. Kvantitatiivsest uuringust selgus, et kahekümnest kolme ettevõtte *pluginad* ja moodulid toetavad uue Magento versiooni kasutamist ning nendest vaid ühel ettevõttel on tugi nii *front end* kui *back end* moodulite jaoks, teised kaks loodavad need võimalused lähiajal luua. Ülejäänud ettevõtted on samuti võtnud suuna tulevikus luua võimalused toetamaks *Decoupled front end* arhitektuuri kasutamist, mis toetaks nii REST API kui GraphQL andmete päringukeeli. Mõned ettevõtted plaanivad luua ka õpetusi või näidislahendusi *pluginate* ja moodulite *front end*-i külge integreerimiseks kasutades *React*, *Angular* või

Vue,js raamistikku. Hetkel suunavad ettevõtted oma ressurssi pigem olemasolevate *pluginate* ja moodulite edasiarendamisse ja täiendamisse, sest Magento versioonid on kiirelt uuenevad, mis omakorda tähendab pidevat panustamist ka *pluginate* ja moodulite uuendamisse.

Neljandas peatükis uuris autor CMS-i platvormi Wordpress ja selle e-poe moodulit Woocommerce ning võrdles nende arengut Magentoga. Võrdlusest selgus, et mõlema platvormi areng on liikunud samas suunas, kuid need võimalused, mis tekkisid Magentole uue 2.3 versiooniga, on Wordpressil olemas juba paar aastat. See on tingitud sellest, et Wordpressil on suur kogukond ning selle liikmetel on lihtsa vaevaga võimalik vastavalt vajadusele ise erinevaid *pluginaid* ja mooduleid juurde arendada. Võttes arvesse, et kahe platvormi areng on olnud sarnane, väidab autor, et töös uuritud arhitektuurid ja arenduspraktikad on rakendatavad ka Wordpressile ning teistele vabavaralistele CMS ja e-kommerts platvormidele, sest nad kõik on oma olemuselt sarnased.

Viiendas peatükis käsitles autor Magento uue versiooni peale tehtud prototüüpi ja selgitas ülesehitust. Autor valis prototüübil realiseeritavateks funktsionaalsusteks toodete pärimise Magentost, toodete kuvamise veebilehel, toodete otsimise veebilehel, ostukorvi loomise, toodete lisamise ostukorvi ja ostukorvis olevate toodete kuvamise, sest antud funktsionaalsused olid autori arvates kõige olulisemad e-poe eksisteerimiseks. Autori arvates on *Decoupled* arhitektuuril põhineva e-poe loomine Magentos keeruline ja aeganõudev protsess, sest puuduvad näited, Magento dokumentatsioonist probleemidele vastuste ja lahenduste leidmine võtab aega ning autor pidi ise erinevaid lahendusi proovima ja testima, et funktsionaalsus tööle saada.

Läbiviidud intervjuude ja küsitluse analüüsi tulemuste ning tehtud prototüübi põhjal soovitab autor hetkel e-poe loomiseks kasutada e-kommerts platvormil Magento *Decoupled* arhitektuuri asemel *Coupled* arhitektuuri.

Bakalaureusetöö eesmärk viidi täide ja selgitati välja millise arhitektuurse versiooni peale e-pood luua ning millised on võimalikud riskid uue *Decoupled* arhitektuuri rakendamisel e-kommerts platvormile Magento. Töö edasiarendusena näeb autor võimalust uurida uusimate arenduspraktikate rakendamist teistele vabavaralistele sisuhaldussüsteemide platvormidele näiteks Wordpress ja Drupal.

Kasutatud kirjandus

[1] Opensource.com, „What is Docker?,“. [Võrgumaterjal]. Available: https://opensource.com/resources/what-docker (29.04.2019)

[2] Google, „Google reCAPTCHA,“. [Võrgumaterjal]. Available: https://www.google.com/recaptcha/intro/v3.html (29.04.2019)

[3] Techopedia, „Module,“ . [Võrgumaterjal]. Available: https://www.techopedia.com/definition/3843/module (29.04.2019)

[4] Computer Hope, „Plugin,“ 2018. [Võrgumaterjal]. Available: https://www.computerhope.com/jargon/p/plugin.htm (29.04.2019)

[5] D. Barker, „Web Content Management,“ 2015. [Võrgumaterjal]. Available: http://flyingsquirrelbook.com/glossary/term/content-management-system (29.04.2019)

[6] IThooldus.ee, „Mis on kodulehe sisuhaldussüsteem ehk administreerimisliides?,“. [Võrgumaterjal]. Available: http://www.ithooldus.ee/kusimused-ja-vastused/mis-on-kodulehe-sisuhaldussuesteem-ehk-administreerimisliides (29.04.2019)

[7] V. Beal, „API – application program interface,“. [Võrgumaterjal]. Available: https://www.webopedia.com/TERM/A/API.html (29.04.2019)

[8] Scoro, „API ehk rakendusliides,“. [Võrgumaterjal]. Available: https://help.scoro.ee/manuals/169/?rel=integrations (29.04.2019)

[9] D. Barker, „Application Programming Interface,“ 2015. [Võrgumaterjal]. Available: http://flyingsquirrelbook.com/glossary/term/application-programming-interface (29.04.2019)

[10] M. Rouse, „Application program interface (API),“. [Võrgumaterjal]. Available: https://searchmicroservices.techtarget.com/definition/application-program-interface-API (29.04.2019)

[11] M. Rouse, „RESTful API,“. [Võrgumaterjal]. Available: https://searchmicroservices.techtarget.com/definition/RESTful-API (29.04.2019)

[12] GraphQL, „GraphQL,“. [Võrgumaterjal]. Available: https://foundation.graphql.org/ (29.04.2019)

[13] S. Buna, „Learning GraphQL and Relay,“ 2016. [Võrgumaterjal]. Available: https://books.google.ee/books?hl=en&lr=lang_en&id=j6XWDQAAQBAJ&oi=fn

d&pg=PP1&dq=graphql&ots=CPLy1yVO65&sig=5hYMV1CUq2s91UYtI3Uha8k5Uyw&redir_esc=y#v=onepage&q&f=false (29.04.2019)

[14] Firebear Studio, „PWA & Headless Magento 2," 2018. [Võrgumaterjal]. Available: https://firebearstudio.com/blog/headless-magento-2-pwa.html (29.04.2019)

[15] L. Aguilar, „A Guide to GraphQL in Plain English," 2018. [Võrgumaterjal]. Available: https://medium.freecodecamp.org/a-beginners-guide-to-graphql-60e43b0a41f5 (29.04.2019)

[16] G. Singh, „To Micro or Mono – Pros and Cons of Both Service Architecture," 2018. [Võrgumaterjal]. Available: https://dzone.com/articles/to-micro-or-mono-pros-and-cons-of-both-service-arc (29.04.2019)

[17] J. Lumetta, „Monolith vs Microservices: Which is the best option for you?," 2018. [Võrgumaterjal]. Available: https://www.webdesignerdepot.com/2018/05/monolith-vs-microservices-which-is-the-best-option-for-you/ (29.04.2019)

[18] T. Doron, „Why we need a new breed of hybrid microservices platform," 2019. [Võrgumaterjal]. Available: https://dzone.com/articles/why-we-need-a-new-breed-of-hybrid-microservices-pl (29.04.2019)

[19] S. ul Haq, „Introduction to monolithic architecture and microservices architecture," 2018. [Võrgumaterjal]. Available: https://medium.com/koderlabs/introduction-to-monolithic-architecture-and-microservices-architecture-b211a5955c63 (29.04.2019)

[20] Brightspot staff, „The pros and cons of coupled, decoupled and headless CMS platforms," 2018. [Võrgumaterjal]. Available: https://www.brightspot.com/blog/decoupled-cms-and-headless-cms-platforms (29.04.2019)

[21] V. Beal, „Decoupled,". [Võrgumaterjal]. Available: https://www.webopedia.com/TERM/D/decoupled.html (29.04.2019)

[22] A. Kharenko, „Monolithic vs. Microservices architecture," 2015. [Võrgumaterjal]. Available: https://articles.microservices.com/monolithic-vs-microservices-architecture-5c4848858f59 (29.04.2019)

[23] A. Zhuk, „Mikroteenused agiilsuse võimaldajana," 2018. [Võrgumaterjal]. Available: https://www.ituudised.ee/arvamused/2018/11/11/mikroteenused-agiilsuse-voimaldajana (29.04.2019)

[24] C. Richardson, „Pattern: Microservice architecture," 2018. [Võrgumaterjal]. Available: https://microservices.io/patterns/microservices.html (29.04.2019)

[25] D. Kovagin, „Äripäeva uus veebiplatvorm," 2018. [Võrgumaterjal]. Available: https://www.ituudised.ee/uudised/2018/10/25/aripaeva-uus-veebiplatvorm (29.04.2019)

[26] Pantheon, „Decoupled CMS,“. [Võrgumaterjal]. Available: https://pantheon.io/decoupled-cms (29.04.2019)

[27] D. Barker, „The state of the headless CMS market,“ 2017. [Võrgumaterjal]. Available: https://gadgetopia.com/post/9926 (29.04.2019)

[28] BSSCommerce, „Magento full tutorial from beginner to advanced,“ 2018. [Võrgumaterjal]. Available: https://bsscommerce.com/blog/everything-you-need-to-know-about-magento/ (29.04.2019)

[29] K. Lodge, „Whati is Magento?,“ 2019. [Võrgumaterjal]. Available: https://www.commonplaces.com/blog/what-is-magento/ (29.04.2019)

[30] B. Sarkhedi, „What exactly is Magento?,“ 2015. [Võrgumaterjal]. Available: https://www.quora.com/What-exactly-is-Magento (29.04.2019)

[31] R. Mening, „Popular CMS by market share,“ 2018. [Võrgumaterjal]. Available: https://websitesetup.org/popular-cms/ (29.04.2019)

[32] P. Rogers, „Big brands using Magento,“ 2018. [Võrgumaterjal]. Available: https://vervaunt.com/big-brands-using-magento/ (29.04.2019)

[33] S. Jain, „Magento 1 vs Magento 2: Top 6 differences you must know,“ 2019. [Võrgumaterjal]. Available: https://serverguy.com/magento/magento-1-vs-magento-2/ (29.04.2019)

[34] Magestore, „Magento 2.3 release – is it better than Magento 2.2?,“ 2018. [Võrgumaterjal]. Available: https://blog.magestore.com/magento-2-3/ (29.04.2019)

[35] Magento, „Magento 2.3: New tools to fuel your growth in 2019,“ 2018. [Võrgumaterjal]. Available: https://magento.com/blog/magento-news/magento-2.3-new-tools-fuel-your-growth-2019 (29.04.2019)

[36] Quora, „What is Wordpress? How does it work?,“ 2018. [Võrgumaterjal]. Available: https://www.quora.com/What-is-WordPress-How-does-it-work-2 (29.04.2019)

[37] R. Rooberg, „WordPress mis see on?,“ 2014. [Võrgumaterjal]. Available: http://digiabi.ee/2014/01/wordpress-mis-see-on/) (29.04.2019)

[38] TechTerms, „Wordpress,“ 2014. [Võrgumaterjal]. Available: https://techterms.com/definition/wordpress (29.04.2019)

[39] Ithemes, „What is WooCommerce?,“. [Võrgumaterjal]. Available: https://ithemes.com/tutorials/what-is-woocommerce/ (29.04.2019)

[40] R. Gill, „What is WooCommerce?,“ 2016. [Võrgumaterjal]. Available: https://www.web-savvy-marketing.com/2016/04/what-is-woocommerce/ (29.04.2019)

[41] Wpbeginner, „Evolution of WordPress user interface,“ 2017. [Võrgumaterjal]. Available: https://www.wpbeginner.com/showcase/evolution-of-wordpress-user-interface-2003-2009/ (29.04.2019)

[42] D. Žoljom, „Headless WordPress: The ups and downs of creating a decoupled WordPress,“ 2018. [Võrgumaterjal]. Available: https://www.smashingmagazine.com/2018/10/headless-wordpress-decoupled/ (29.04.2019)

[43] WPGRAPHQL, „GraphQL API for WordPress,“. [Võrgumaterjal]. Available: https://www.wpgraphql.com/ (29.04.2019)

[44] WooCommerce, „Woo REST API,“. [Võrgumaterjal]. Available: https://woocommerce.github.io/woocommerce-rest-api-docs/?shell#introduction (29.04.2019)

[45] Shopz.io, „WooCommerce GraphQL API,“ 2018. [Võrgumaterjal]. Available: https://wordpress.org/plugins/woo-graphql-api/#description (29.04.2019)

Lisa 1 – Individuaalintervjuude küsitlus

1. Kui kaua on teie ettevõte tegelenud Magento platvormile arendamisega?
2. Millist Magento versiooni ja arhitektuuri kasutate hetkel e-poodide loomiseks?
3. Kas olete juba võtnud või plaanite võtta kasutusele enda loodavates lahendustes Decoupled arhitektuuri?
    a. Kui jah, siis millal võtsite või millal plaanite võtta?
    b. Kui ei, siis miks?
4. Kas nõustute väitega, et hetkel ei ole majanduslikult otstarbekas ehitada uusi e-poode Decoupled arhitektuurile, sest puudub põhjalik dokumentatsioon, tehniline tugi ja paljud pluginad ning moodulid veel ei toeta uut versiooni, mis tõttu platvormi eelised võrreldes erilahendustega kaovad?
5. Kui peaksite looma e-poe Magento platvormile Decoupled arhitektuuriga siis milliseid riske te hetkel järgneva 12 kuu jooksul ette näete (pluginate tugi, ebastabiilne core funktsonaalsus, tööjõu kompetentside puudulikkus vms)?
6. Millised plusse näete Magento Decopled arhitektuuris, hetkel ja aasta pärast?

Järgnevad küsimused esitatakse intervjueeritavale juhul, kui ettevõttes juba kasutatakse Decoupled arhitektuuri:

7. Kas kasutate Magento Decoupled arhitektuuris ainult Magento baasrakendust ja endaloodud erilahendusi või mõnda vabavaralist või kommerts pluginat/moodulit?
8. Kui kasutate mooduleid/pluginaid, siis kuidas lahendasite nende integratsiooni ja visuaali näitamine front end poolel ? Kas see tugi tuli juba sisseehitatult mooduli arendaja poolt või pidite ise vastavad liidesed ja esitluskihi arendama?
9. Millise JavaScript raamistiku peale ehitate front end kihi?

Lisa 2 – Intervjuude transkriptsioon

Aeg: 10.04.2019
Intervjueeritav: ettevõte 1
Intervjuu kestis 20 minutit ja 8 sekundit
Edaspidi „K“ tähistab intervjueerija küsimust ning „V“ intervjueeritava vastust.

K: Kui kaua on teie ettevõte tegelenud Magento platvormil arendamisega.
V: Eestis me oleme praeguse seisuga tegelenud üks neli aastat. Meil on ettevõte reaalselt eksisteerinud neli aastat.
K: Millist Magento versiooni ja arhitektuuri kasutate hetkel poodide loomiseks?
V: Versioonid on meil seinast seina. Meil on olemas Magento ühe baasil olevaid platvorme terve rodu kliente, kes kasutavad ikka veel endiselt magento ühte. Nad on kõik nii-öelda viimases patch versioonis küll, aga on siiski magento üks. Ja siis on meil olemas Magento kahest ka terve portfoolio täis ja nemad kasutavad samamoodi erinevaid versioone, on, mõned neist on, 2.0 peal meil ei ole vist ühtegi. Ja siis osad on meil 2.1 peal, osad on 2.2 ja 2.3 peal ei ole meil ka praegu kedagi.
K: Sellest ma järeldan, et hetkel kasutate coupled arhitektuuri, et decoupled peale pole teinud, kuna see vist ei olegi varem varasemate versioonidega võimalik?
V: No see on nii ja naapidi selles mõttes, et ütleme minu jaoks mida üldse tähendab, et decoupled arhitektuur on, see on nagu meeletult lai mõiste. Et kui sa saad kuidagi konkreetsemalt mu käest küsida, et mida sa selle decoupled arhitektuuri all täpselt silmas pead, siis ma võib-olla oskan sulle kohe paremini vastata.
K: Antud töö fookuses ma pean silmas seda, et front end on arendatud täiesti ise ütleme, reactis või, või, või millegis muus , et ei kasuta nagu Magentot ennast front endi tegemiseks, et Magento on puhtalt back end.
V: Ütleme, nii-öelda täielikku lahendust, sellist meil ei ole tõesti, kus kohas meil oleks front end täiesti eraldiseisev siukest täielikku lahendust meil ei ole. Ja meil ei ole ka keegi tulnud meie käest seda otseselt nõu küsima siiamaani, et võiks seda asja kuidagi niimoodi teha. Me ei ole ise ka otseselt kellelegi pakkunud siiamaani seda, et kas te tahate seda ja seda puhtalt sellepärast, et taas kord me oleme nii palju kui frixionit või nagunii-öelda, sest selle staatilist massi kogu selle asja juures mida, mis sellega kaasa tuleb ja kliendid üldjuhul noh, kui oleks meie enda toode, siis oleks nagu teine lugu, ega aga kliendid üldjuhul nemad üritavad saada võimalikult cost effectively kätte, mida iganes oma äriga plaanivad teha, eks ole. Ja ütleme niimoodi, igasugune nagu front endi decouplemine back endist mitte kuidagimoodi cost effective mitte kuskilt otsast, vähemalt veel mitte. Aga see muidugi tõenäoliselt ka muutub tulevikus, eks ole. Kui sul tekib, neil on nagu mingisugune, ütleme, äriline edge selle asja juures on igapäevaselt rohkem, siis nad hakkavad kõik riburada pidi sinna minema. Aga siiamaani ei ole neil ärilises plaanis mingit kasu sellest.
K: Kas olete juba võtnud või plaanite võtta kasutusele enda loodavates lahendustes decoupled arhitektuuril sellisel kujul, siis vastus oleks, et tulevikus plaanid kui, kui see liigub sinna suunas?
V: Selles mõttes jah, kindlasti absoluutselt, ega me meie areneme koos selle turuga, milles me oleme mis tähendab seda, et kui turg sinna suunas areneb, siis ka meie

liigume sinna suunas tehnilisest vaatevinklist. Me ise nii palju me teeme, et me ise exploreme seda ala. Aga nagu reaalne praktiline, selline nagu application praktiline eesmärk selle asjaga tekib alles siis, kui turg on valmis seda nagu sisse võtma.

K: Kas nõustute väitega, et hetkel ei ole majanduslikult otstarbekas ehitada uusi poode decoupled arhitektuurile, kuna puudub põhjalik dokumentatsioon, tehniline tugi ja paljud pluginad ning moodulid veel ei toeta uut versiooni, mistõttu platvormi eelised võrreldes erilahendustega kaovad?

V: Ja ma olen nõus sellega. Ma siin jõudumööda olengi seda põhjendanud, mul jääb selline mulje. Aga põhimõtteliselt küll jah, ma olen sellega nõus nüüd selle vastu noh, räägivad muidugi väga erinevad argumendid osaliselt kattuvad ka sellega, mis sa just ütlesid eks, aga ütleme, et survestuse osa selle või selle põhisurve määrab ikkagi nagu e-kommertsi nagu turg kui selline. Et see on see põhiline surve dokumentatsioonile ja need asjad kui ütleme, et kui seal surve on sellel asjal suur, siis tekib ka dokumentatsioon. Et mida väiksem on see surve, seda vähem community kui selline panustab sellesse asjasse. Noh, praeguse seisuga ongi niimoodi, et mingisugused kindlad piloodid ainult viivad seda asja läbi. Ja mitte Magento, sest ilmtingimata viiakse läbi eelkõige nagu muudes kohtades rohkem, et Magento on jäänud selles osas nagu tahaplaanile. Magentos on ainult üksikud piloodid, kes seda veavad. Ja niikaua, kui need üksikud seal on, nii kaua ei saa olema mitte kunagi head dokumentatsiooni ei saa mitte mingeid häid frameworke ega mitte midagi sellist. Et eks ta jõudu pidi jõudu pidi tuleb, aga see on väga aeglane protsess ja ootab, pigem ütleme siis turu aktiveerumist selle ala pealt, et siis hakkame nägema nagu korraliku dokumentatsiooni me näeme kõike seda põhimõtteliselt.

K: Kui peaksite looma e-poe Magento platvormile decoupled arhitektuuriga, siis milliseid riske te hetkele järgneva ütleme aasta jooksul ette näete? Kas pluginate tugi puudub või tööjõukompetents puudulik või midagi sellist?

V: Ja need kaks asja on kindlasti tõsi, et ütleme, et noh, kompetentsiga on nagu selline asi, et mõnes mõttes mõnes mõttes ma arvan, et kompetents, kompetentsi probleem, nii väga isegi ei ole, sest kui see, kui me saame isoleerida back endi ja front endi ära ja kasutada midagi sellist geneerilist nagu GraphQL siis see tähendab seda, et meil ei pea olema Magento kompetentsiga inimesed tegelikult üldse, et front endi teha. Mis mõnes mõttes nagu natukene leevendab seda probleemi. Ja Magento liigub päris tugeva kiirusega, oma versiooniga edasi ka ja pluginate tüübid peavad ju hoidma seda asja nagu ühilduma kõikide uute versioonidega, mis mõni endast välja tulevad, nad panevad kõik oma ressursid sinna alla tegelikult kinni. Ja samamoodi meie, et meie paneme oma ressursi siis nii-öelda suppordi alla kinni ja siis saab täpselt samamoodi kõik upgraded, mida me Magentole teeme, turva upgraded ja kõik need asjad. Noh nüüd me peame täpselt samamoodi oleme kindlad, et nad on ühilduvad edasi, et ütleme siis jah niivõrd, et ressurss on lihtsalt muude asjade all kinni, mis on tähtsamad.

K: Milliseid plusse näete Magento decoupled arhitektuuris hetkel ja aasta pärast?

V: Ma mõtlen just, et plusside poole pealt kindlasti see nagu vabadus teha UI-ga mida, mida iganes soovitakse teha. Et kuna siis antud juhul ei ole enam vahet nagu, et kas sa näiteks kas sa võtad modifitseerid Magento endale, ütleme siis check-outi või siis sa lood täitsa uue, et see on täiesti vabad käed. Et see on minu arust nagu see põhiline, kõige suurem pluss, et kui saab front endi selle back endi küljest nagu täiesti lahti, sest siis on sul vabad käed integreerida see front end ükskõik näiteks mõnel teisel platvormil, siis sa võid, sa võid selle kasvõi Wordpressi sisse toppida või ükskõik kuidas sa oma front endi lahendad, et ma arvan, et see on see põhiline probleem, et ütleme, Magento UI on selles mõttes üsna restrective praegu, et on palju asju, mida me ei saa kliendile lihtsalt pakkuda, sellepärast et see läheks kliendile kas liiga kulukaks või

põhiliselt nagu selle tõttu, et see on lihtsalt liiga keeruline ära teha või liiga ajamahukas. Et sellest on kahju, et me ei saa seda hetkel teha, ma arvan, et see nii-öelda decoupled Magento selles mõttes lahendaks selle probleemi ära, et inimestele nagu neile kõike seda pakkuda.

Aeg: 16.04.2019
Intervjueeritav: ettevõte 2
Intervjuu kestis 14 minutit ja 59 sekundit
Edaspidi „K“ tähistab intervjueerija küsimust ning „V“ intervjueeritava vastust.

K: Kui kaua on teie ettevõte tegelenud Magento platvormil arendamisega?
V: Aastast 2009
K: Millist Magento versiooni arhitektuuri kasutate hetkel e-poodide loomiseks?
V: Hetkel Magento kahe peale ikkagi viimasele versioonile, aga samas on meil ka neid Magento ühe poode alles. Et me küll sinna uusi ei tee, aga me ikkagi jätkuvalt nii-öelda kõike ei ole veel uuele tarkvarale üle viinud.
K: Magento kahe puhul siis kasutate ikkagi seda coupled versiooni ehk siis kust te kasutate nagu Magentot ennast, et Front end ei ole nagu eraldi loodud kuskil reactis või Angularis?
V: Meil ei ole nii, et selles mõttes meil on nagu töös onju, et nagu mille peale mis on see vue.js storefront, et selle peale, nagu on ka ütleme, üks on töös ja teine sellises nii öelda pakkumise läbirääkimise järgus.
K: Kliendid on siis ikkagi huvitatud sellest?
V: No kliendid on huvitatud ja et just see noh, just nagu lahku lüüa, et sest see, kas selle Magento kahe peale sellise Storefrondi ehitamine on üsna ajamahukas töö ja kliendid on selles mõttes ka sellest aru saanud, et noh, mis siin eelmine aasta Eesti niisugune turu trend oli, selline microservices siis on ja et mida rohkem sa saad nagu asju nii-öelda eraldi teha, eks ju, et see on nagu tuleviku mõttes parem, et otsustasime vue peale teha. Hiljem vahetad selle Magento sealt tagant ära, kui tahad, onju. Et see on väga nagu selline müügiargument hetkel, et turu trend ka ilmselt
K: Millal te selle esimese decoupled versiooni kasutusele võtsite?
V: Millal me võtsime, no selles mõttes me ütleme siin mingisuguse, millal ta nüüd välja tuli. Novembri lõpus vist. Jah, et noh, nüüd vaikselt nagu, ütleme kohe ei võtnud mingi kaks-kolm kuud, et noh, üsna kohe kui ta tegelikult tuli, et siis tulid ka need nii-öelda pakkumised selle peale aga see iseenesest nagu idee, kui selline microservicite peale ehitada, et meie Soome poolel oli ka varem juba selle 2.2 versiooni peale seal noh, nii-öelda läbirääkimisi toimus on ja et prooviti nagu erinevaid asju.
K: Kas nõustute väitega, et hetkel ei ole majanduslikult otstarbekas ehitada uusi poode decoupled arhitektuurile, kuna puudub põhjalik dokumentatsioon, tehniline tugi ja paljud pluginad ja moodulid veel ei toetud versiooni, mistõttu selle platvormi eelised võrreldes erilahendustega kaovad?
V: No selles mõttes ikkagi ma pigem ei ole nagu nõus sellega, et tänaseks päevaks ikkagi ütleme need uuemad versioonid 2, 2.2 ja sealt alates 2.3 on ju nüüd nad on ikkagi sellise stabiilsuse juba saavutanud, et jah, sinna 2.1 peale ma ei soovitaks mitte kellelegi mitte kunagi enam midagi teha, on ju. Et Magento nagu noh et on nagu selline stabiilsus on saavutatud, et julgeks sinna tegelikult ehitada neid asju küll aga just nagu vaatakski laiemalt, et ei keskenduks ainult sellele ühele tarkvarale, vaid nagu kõik see, mis selle ümber ka veel saab töösse panna.
K: Kui peaksite looma e-poe Magento platvormile decoupled arhitektuuriga, siis milliseid riske te hetkele järgneva ütleme aasta jooksul ette näete? Kas pluginate tugi puudub või tööjõukompetents puudulik või midagi sellist?

V: No põhiprobleem on jah see, et kas on piisavalt neid erinevaid mooduleid sinna peale, nii et ka Eesti turule, et see nii-öelda moodulite kaasajastamine kas töötab. Siis teine asi on kindlasti see tööjõud, et ka Eesti turul väga palju neid inimesi võtta ei ole, kes on kokku puutunud ja juba oskavad midagi teha. Jah, Magento puhul vist ei saa kunagi sellest ka mööda vaadata, et see core funktsionaalsus ja sealt võib ikka selliseid üllatusi välja tulla, et ikka on, siis mõtled, et kuidas sa selle ära lahendad.

K: Milliseid plusse näete Magento uues versioonis hetkel ja aasta pärast?

V: No põhi pluss võtame, nagu meie kliendi seisukohast on üks asi on esiteks see, et tegelikult on ju väga palju funktsionaalsust, et kliendi vaatenurgast kõike seda ise nullist endale tellima hakata, see kulu on kordades suurem. Ja ka nagu seda, et noh, ütleme kliendid on harjunud esiteks selle tarkvaraga natuke nagu teavad, mida sealt oodata, et on ikkagist, võimaldab üsna palju, et see nagu see on põhipluss tegelikult miks ta valitakse.

K: Kas kasutate Magento decoupled arhitektuuris ainult Magento baasrakendust ja enda loodud erilahendust või ka mõnda vabavaralist või kommerts pluginat/moodulit?

V: See on küll hea tehniline küsimus. Kui Storefront iseenesest pakub Magento oma me ikkagist oleme vist enamasti oleme ütleme nii-öelda kui mingisuguseid integratsiooni ja selliseid asju ehitatakse Magento enda nii öelda nende API vahendusel. Et selle peale, et noh, ma arvan, et eks lõpuks kõiki asju saab kokku liidestada.

K: Aga olete Magento mooduleid pluginaid selle uue arhitektuuri puhul juba kasutanud ka?

V: 2.3 peale veel ei ole. Pigem on selline nii-öelda et me oleme alles üsna algusfaasis kogu selle asjaga. Selle viimase nii-öelda 2.3 peale projektiga, et sealt nagu ei ole väga niisugust kogemust, millest veel, mida veel nagu jagada.

K: Kas hetkel ongi teil ainult JavaScripti raamistikest Vue kasutusele? Miks te selle otsustasite võtta, miks mitte näiteks reacti või Angulari?

V: Tundus kõige mõistlikum, et ega mina selle valikuprotsess juures ei olnud, et meie nagunii-öelda selline peaarhitekt või sellises rollis inimene, et tema nagu algatusel see tuli, et teeks nüüd nii, onju.

Aeg: 03.05.2019
Intervjueeritav: ettevõte 3
Intervjuu kestis 18 minutit ja 12 sekundit
Edaspidi „K“ tähistab intervjueerija küsimust ning „V“ intervjueeritava vastust.

K: Kui kaua on teie ettevõte tegelenud Magento platvormile arendamisega?

V: Ma arvan, et seitsmes aasta läheb praegu.

K: Millist Magento versiooni ja arhitektuuri kasutate hetkel poodide loomiseks?

V: Nüüd uued uued e-poed, teeme kõik Magento kahe peale. Ja siis alustame enamjaolt alati sellest stabiilsest versioonist, mis on välja tulnud, ehk siis. No hetkel on 2.3 siis, aga meil on hoolduses devopsis on Magento ühed.

K: Kas olete juba võtnud või plaanite võtta kasutusele enda loodavates lahendustesse Decoupled arhitektuuri?

V: Jah, me tegelikult juba, kui see beetaversioon oli väljas, eelmine, eelmise aasta kevadel tuli. Et siis ma hakkasin juba uurima seda ja proovima ja siuke terve meeskond nagu putitas seda, proovis seda tööle saada ja siis me juba nägime, et see on väga-väga ajakulukas, on see asi üldse tööle saada ja vaata, õppimis barjäär ka nagu, et ehk siis betaversiooni kohe midagi reaalset tegelikult tegema ei hakanud, ainult panime tööle selle. Siis nüüd, kui nüüd on välja tulnud see uus lahendus, tegelikult me nüüd ootame veel natukene, et tahaks mingit successcase reaalselt näha ja katsuda ja mõeldes

klientidele nagu, et natuke oodata, et infobaas nagu tõuseks ja foorumites oleks natuke infot rohkem ja et aga me teeme endale ühte sisemist projekti, teeme.

K: Kas nõustute väitega, et hetkel ei ole majanduslikult otstarbekas ehitada uusi poode Decoupled arhitektuurile, kuna puudub põhjalik dokumentatsioon, tehniline tugi ja paljud plugina moodulid veel ei toeta uut versiooni, mistõttu platvormi eelised võrreldes erilahendustega kaovad.

V: Et praegu on, ma olen nõus ja et, et praegu on see sel praegu ta nõuab liiga suurt investeeringut veel tavalise standardse lahendusega, mis nüüd 2.3 jõudes on juba päris noh, päris stabiilsed ja kusjuures on need võimalused juba optimeerida, noh teadmised ja, ja lahendused optimeerida seda tavalahendust ka väga kiireks, mis alguses väga raske oli, aga nendega on võimalik tavalahendus ka tegelikult päris kiireks saada, kui ta alguses juba kiiruse võtad, nagu eesmärgiks. Noh, neid vigu ikka tehakse, kellad-viled on kõige tähtsamad ja siis kiirus jääb lõpuks optimeerida ja siis oled selle lahendusega natuke jännis.

K: Kui peaksite looma e-poe Magento platvormile Decoupled arhitektuurile siis milliseid riske te hetkel järgneva kaheteist kuu jooksul ette näete? Kas näiteks pluginate tugi, ebastabiilne koor, funktsionaalsus, tööjõu kompetentsi puudulikkus või midagi sellist?

V: Ma arvan, et ka esimene risk on üldse esimene risk on alguses see investeering, et, et kindlasti läheb üle aja hinnangutele tegemine siis teine on garantiiperiood selle valulikkus nagu kulu siis. No mis läheb investeeringu alla ilmselt et see asi nagu stabiilseks saada, stabiilsust tagada. Ma kujutan ette, et üsna raske võib olla, oleneb, et noh, kuidas muidugi ehitada. Et see nagu arhitektuur võiks nagu võiks nagu olla nagu vastu nendele versiooni uuendustele, aga ma ikka kardan igat Magento versiooniuuendust, nii et need versiooniuuendused võivad olla alguses niisugused huvitavad.

K: Milliseid plusse näete Magento Decoupled arhitektuuris hetkel ja siis aasta pärast?

V: Hetkel kõige suurem pluss on see, et asi ära tehti, et see, et me oleme nagu selles, noh, võib öelda moodsas arhitektuuris nagu sees, et ei ole seal ainult see monoliitne variant. Ja see on, see on põhipluss praegu. Ja aasta pärast on ma usun, et on juba päris palju lahendusi. Et aasta pärast võiks olla juba moodulite tugi ka üsna selline tavaline, et üle poolte äkki juba saadaolevatest toetavad. Ja ma loodan, et me oleme jõudnud aasta pärast ise ka enda moodulitele need vajalikele moodulitele toed teha.

K: Te enne mainisite, et te arendate praegu ettevõttesisest projekti decoupled arhitektuurile. Kas ma saan õigesti aru, et te sinna veel ei ole üritanud mingeid mooduleid ja pluginaid üldse külge panna?

V: Ei, ei me proovime, see tuleb. See tuleb üsna standardile lähedale, kuna seal nagu see globaalne pood virtuaalsete toodetega, siis ei ole seal checkout ka väga keeruline, nagu et loomulikult uurime vaatama, et kas on mingeid turundusmooduleid või midagi ägedat veel juurde sinna saada, aga noh ilmselt ta jääbki niisugune üsna nagu standardi lähedane, aga me tahaks panna oma kuulutusega.

K: Milles te front endi loote? Mis Javascripti raamistiku te kasutate?

V: Mõtlesin selle küsimuse peale, et minu arust noh, et venia on tehtud ju reacti peale ja eks me oleme küll ja et me läheme sama teed ikka.

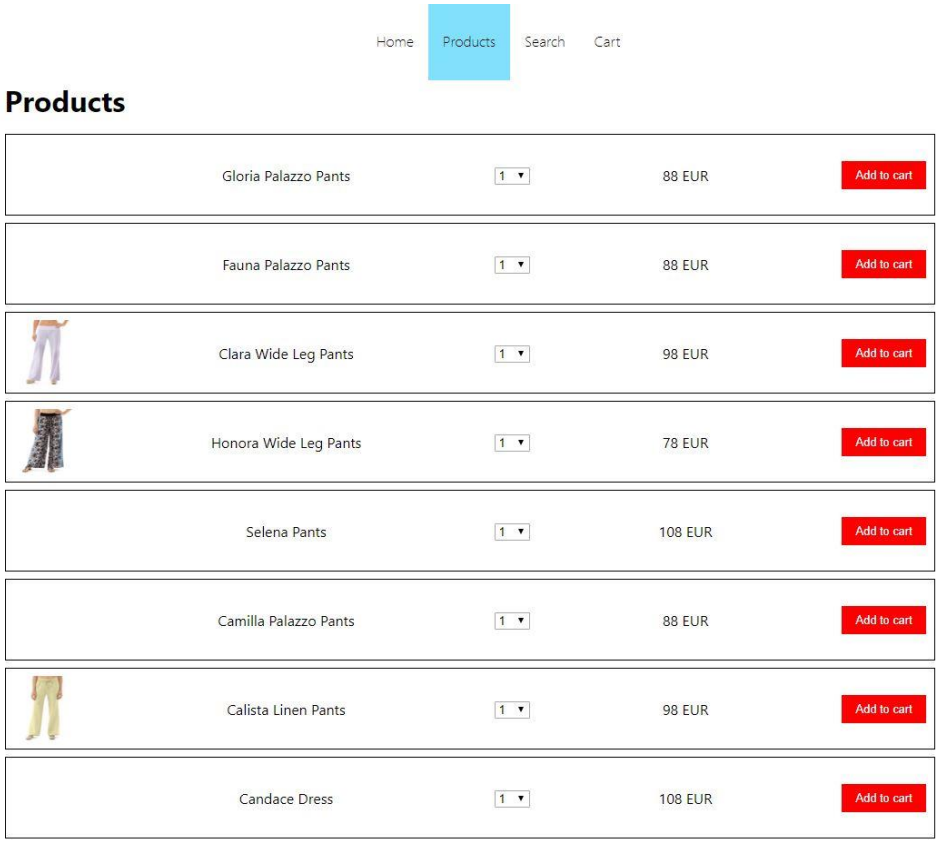
Lisa 3 – Ankeetküsitlus

1. Does any of your Modules/Plugins support Magento decoupled front end architecture?
   a. If YES, are you using REST or GraphQL?
   b. If YES, are you offering also front end implementation examples (React or Angular code snippets etc.)?
   c. If NO, do you plan to offer support for decoupled front end architecture?
      - When?
      - Which protocol REST or GraphQL you plan to use?
      - Do you plan to provide also front end examples (React or Angular code snippets etc)?
2. What will be the update process for the plugin with decoupled front end support?
   a. Will the updates become available same time as for the coupled version or later?
   b. Will the updates include also decoupled front end implementation examples?

Lisa 4 – Ankeetküsitluses osalenud ettevõtted

1. Aheadworks Magento Store. https://ecommerce.aheadworks.com/
2. Aitoc Software LLC. https://www.aitoc.com/
3. Amasty. https://amasty.com/
4. BelVG. https://belvg.com/
5. BSS Commerce. https://bsscommerce.com/
6. Extait. https://extait.com/
7. FME Extensions. https://www.fmeextensions.com/
8. Fooman. https://store.fooman.co.nz/
9. Knowband Store. https://www.knowband.com/
10. MageAnts. https://www.mageants.com/
11. MageComp. https://magecomp.com/
12. Mageplaza. https://www.mageplaza.com/
13. MageWorx Company. https://www.mageworx.com
14. M-Connect Media. https://www.mconnectmedia.com/
15. Mirasvit. https://mirasvit.com/magento-2-extensions.html
16. Neklo, LLC. https://store.neklo.com/
17. ParadoxLabs. https://www.paradoxlabs.com/
18. Plumrocket Inc. https://store.plumrocket.com/
19. Solwin Infotech. https://www.solwininfotech.com/
20. Webkul Software. https://webkul.com/

Lisa 5 – Prototüübi visuaalid



Cart

Clara Wide Leg Pants	1 Item	98 EUR
Camilla Palazzo Pants	1 Item	70 EUR
Carola Infinity Scarf	6 Item	48 EUR