TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Annika Roosleht IADB193370

Klienditeavituste rakenduse juurutamine Zone Media OÜ näitel

Bakalaureusetöö

Juhendaja: Toomas Lepikult
PhD
Kaasjuhendaja: Ats Aim
BSc

Tallinn 2023

Autorideklaratsioon

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

Autor: Annika Roosleht

15.05.2023

Annotatsioon

Käesoleva lõputöö peamiseks eesmärgiks on luua keskne klienditeavituste süsteem, mida oleks lihtne ühtlaselt täiendada ja ajakohasena hoida. Rakendus muudaks Zone Media OÜ kliendi kirjade saatmise nii arenduse- kui ka klienditoe osakonna poolt sujuvamaks. Rakendus, mille baasil lahendust arendatakse, on alates 2016. aastast arenduses olnud ettevõttesisene pärandrakendus, mis omaette kaasab teatud lisapiiranguid ja nõudeid.

Töö analüüsiosas uuritakse eksisteerivaid kolmandaosapoole teavitussüsteeme ning võrreldakse omavahel tehnoloogiaid, mida oleks võimalik kasutada eesmärgi saavutamiseks. Klienditeavituste rakenduse nõuded püstitatakse pärandrakenduse analüüsist lähtuvalt. Bakalaureusetöö praktilises osas tegeletakse lahenduse elluviimisega algul arenduses ning seejärel automaattestimises, lahendades jooksvalt tekkivaid probleeme ja takistusi.

Antud töö rõhuasetus on kasutatavatel tehnoloogiatel: nende kaasaegsus, kasutusmugavus, kasutajabaas ja antud rakenduse skoobi sobivus. Töös on kaetud klienditeavituste rakenduse arendusprotsess ning testimine. Arendusprotsessi tulemuseks valmib prototüüp rakendus, mille abil saab juurutada tervikliku süsteemi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 6 peatükki, 11 joonist, 1 tabelit.

Abstract

Implementation of Customer Notifications Application on the Example of Zone Media OÜ

This thesis aims to create a central customer notification system that would be easy to update and keep contemporary and streamline the sending of Zone Media OÜ customer letters by both the development and customer support departments. The program, based on which the solution is established, has been in development since 2016 as an existing internal application, which includes certain additional restrictions and requirements.

The analysis part of the work studies existing third-party notification systems and compares the technologies that can be used to achieve the goal. The customer notifications application requirements are established based on the analysis of the existing internal application. In the practical part of the bachelor thesis, the implementation of the solution is dealt with first in development, then in the creation of the application interface and then in automatic testing, solving ongoing problems and obstacles.

The emphasis is on the technologies used: their modernity, ease of use, user base, and appropriateness of the given application scope. The thesis covers the development process and testing of the customer notifications program and application interface. The outcome of the development process is a prototype application, which can be used to implement a complete system.

The thesis is in Estonian and contains 41 pages of text, 6 chapters, 11 figures, 1 tables.

Lühendite ja mõistete sõnastik

A/B testimine	Testimismetoodika, milles võrreldakse omavahel kahe veebilehe või rakenduse versiooni, et teha kindlaks, milline neist paremini toimib.
Ad-hoc	Ajutine või ühekordseks otstarbeks loodud lahendus.
API	*Application Interface* ehk rakendusliides.
Bränd	Zone Medias defineeritud kui klientide kogum, kelle puhul kehtivad eri reeglid. Juhised jagunevad omakorda nii piirkonna või teenuse vahel. Zone Media tegeleb Eesti, Soome, Telia ning Euroopa klientidega ja brändid on vastavalt Zone, ZoneFI, Zone Telia ja ZoneEU.
CI/CD	*Continuous Integration/Continuous Delivery*, pidevintegratsioon ja pidevtarne.
Composer	Sõltuvuste haldaja PHP programeerimiskeele jaoks.
CQRS	*Command and Query Responsibility Segregation*, arhitektuuri disainimuster, mis jagab süsteemi toimingud käskudeks ja päringuteks.
Cron	Käsurea utiliit, mida kasutatakse järjepidevate tööde ajastamiseks.
CSRF	*Cross-Site Request Forgery* ehk saitidevaheliste taotluste võltsimise rünnak.
Curl	*Client for URL,* käsurea utiliit andmete edastamiseks serverisse ja serverist.
DAL	*Data Access Layer*, andmete ligipääsukiht arhitektuuris.
DSN	*Data Source Name*, andmete struktuur, mida kasutatakse andmeallikaga ühenduse kirjeldamiseks.
EDA	*Event-Driven Architecture*, tarkvaraarhitekuuri disaini muster, mis edendab sündmuste tuvastamist, tarbimist ja reageerimist.
Git	Versioonihaldustarkvara.
HTML	*Hypertext Markup Language* ehk märgistuskeel veebilehtede märgendamiseks.
HTTP	*Hypertext Transfer Protocol* ehk protokoll teabe edastamiseks arvutivõrkudes.

IDE	*Integrated Development Environment* ehk integreeritud arenduskeskkond.
JSON	*JavaScript Object Notation*, Javascripti objekt notatsioon.
MariaDB	Relatsiooniline andmebaasimootor.
Markdown	Lihtteksti vormindamise süntaks.
MIME	*Multipurpose Internet Mail Extension*
MVP	*Minimum Viable Product* ehk rakenduse minimaalsete funktsioonidega toimiv versioon.
MySQL	Relatsiooniline andmebaasimootor.
PHPStan	PHP programmeerimiskeele staatilise analüüsi tööriist.
PHPUnit	PHP programmeerimiskeele testimisraamistik.
Pipeline	Kogum jadamisi ühendatud andmete töötlemise elemente.
PSR	*PHP Standards Recommendations*, PHP spetsifikatsioon, reeglite ja tavade kogum, mis määravad, kuidas PHP-koodi vormindada ja struktureerida.
REST	*Representational State Transfer*, tarkvaraarenduse API suunis.
SaaS	*Software as a Service* ehk tarkvara teenusena.
SMTP	*Simple Mail Transfer Protocol*, protokoll meiliaadresside edastamisel ja saamisel.
STL	*Standard Template Library*, tarkvara raamatukogu.
Symfony	Avatud lähtekoodiga PHP veebirakenduste raamistik.
Tagarakendus	Kasutajale mittenähtav osa rakendusest, mis sisaldab rakenduse loogikat.
Test-Last Development	Tarkvararendusmetoodika võte, mille käigu kirjutatakse testid arenduse lõppfaasis.
Twig	Mallimootor PHP programmeerimiskeelele.
Webhook	HTTP-põhine tagasikutsumis funktsioon, mis võimaldab sündmuste põhist suhtlust kahe rakendusliidese vahel.
Xdebug	PHP laiendus, mis pakub testimisel silmis- ja profileerimisvõimalusi.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeval on teenuste lahutamatuks osaks teavitused. Zone Media OÜ-l on oma igapäevaste tegevuste käigus vaja välja saata suurtes kogustes erinevaid teavitusi klientide informeerimiseks, teenuste kinnitamiseks ja haldamiseks ning arvete edastamiseks. Seetõttu on tähtis, et ettevõte omaks tõhusat klienditeavituste süsteemi, mis suudaks tagada kvaliteetse kliendikogemuse ning vastata nende ootustele. Teema valimise hetkel oli ettevõttes kasutusel pärandrakendus, mis käesoleval ajahetkel ei vasta enam ettevõtte nõudmistele. Süsteemi tehnoloogilise võla kuhjumise tõttu on selle haldamine ja arendamine muutunud ebaefektiivseks ning sellest tingituna on tekkinud vajadus uuenenud süsteemile.

Töö eesmärgiks on luua keskne klienditeavituste süsteem, mida oleks lihtne ühtlaselt täiendada ja ajakohasena hoida. Loodav lahendus peaks järgima ettevõttesiseseid turvanõudeid. Lahenduse peaks olema jälgitav ning vigu peaks saama hallata. Lahendus peaks olema kaetud ühiktestidega. Lõputöö käigus realiseeritud lahendus on aluseks tulevikus ettevõttes kasutusele võetava keskse klienditeavituste rakendusena.

Bakalaureusetöö analüüsi osas kaardistatakse ning analüüsitakse pärandrakenduses esinenud probleeme ning nende põhjuseid ja pakutakse välja võimalikke lahendusi rakenduse olukorra parandamiseks. Analüüsitakse erinevaid praktikaid ja tehnoloogiaid, mis aitaksid püstitatud probleeme lahendada ning väldiksid taoliste murekohtade esinemist tulevikus.

Töö praktilises osas asutakse projekti jätkusuutlikkuse parandamiseks tehtud analüüsi põhjal realiseerima erinevaid lahendusi, kirjeldatakse selle käigus esinenud katsumisi ning saadud õppetunde. Käesolev osa nõuab nii olemasolevate tööriistade kui ka kasutusele võetavate tehnoloogiate tundmaõppimist. Tulemusena valmib süsteem, mille abil kirjeldatud probleem saab lahendatud.

2 Zone Media OÜ-le loodava klienditeavituste rakenduse taust

Antud peatüki raames tutvustatakse esmalt ettevõtte ja projekti tausta, millega on seotud käesolev lõputöö. Sõnastatakse uuritav probleem ning selle lahendamiseks seotud eesmärgid ja valitud metoodika. Tuuakse välja teema aktuaalsus ettevõttes. Viimaks tutvustatakse töö skoopi.

2.1 Ülevaade ettevõttest Zone Media OÜ

Zone Media OÜ on üks Eesti suurematest Interneti tugi- ja infrastruktuuriteenuseid pakkuv ettevõte, mis tegutseb aastast 1999. Hetkel on Zone Media Eesti suurim veebimajutust pakkuv ettevõte, mille infrastruktuur ulatub täna Beneluxi riikidest kuni Baltimaadeni. 2021. aasta teises pooles laienes Zone oma tegevustega aktiivselt ja ametlikult ka Soome [1].

Zone Media eesmärgiks on pakkuda stabiilset, kuid samas lihtsaid, kiireid ning töökindlaid lahendusi erinevates teenustes ning informatsiooni edastamisel veebis [1]. Ettevõte on pikka aega olnud Eesti domeeninimede registreerimise liider ning samuti Eesti esimene Euroopa Liidu ülemdomeeni (.eu) akrediteeritud registripidaja [2].

Ettevõtte põhitegevusalaks on pakkuda teenuseid nii veebi-, serveri- kui ka e-posti majutustes, domeenide registreerimisel, allhankeid telekommunikatsioonіettevõtetele ja ka teenuseid, mis on suunatud eraklientidele [3]. Klientidena on peamiselt Eesti ettevõtted, eraisikud, riigiasutused ja organisatsioonid, kelle peamine äritegevus ja igapäevane suhtlus on suuresti seotud kodulehe ja/või e-postiga [3] [4]. Ühtlasi kuulub Zone Media Eesti Infotehnoloogia ja Telekommunikatsiooni Liitu [5].

2.2 Probleemi tutvustus ja eesmärk

Klienditeavituste rakenduse väljatöötamise ajendiks on mitmed pärandrakendusega seonduvad aspektid. Probleem seisneb eelkõige selles, et Zone Medias saadetakse pea iga

teenuse sammu juures klientidele teavitusi ning sõltuvalt teavituse tüübist võivad need väljuda süsteemi erikomponentidest. Teavitusi saadetakse hetkel vaid e-kirja põhiselt. Kõikide teavete haldamiseks oleks vaja aga tsentraalselt ettevõtte sisest teavituste süsteemi, mis ei piirduks enam vaid e-kirjadega ning teavituste saatmine toimuks ühest kesksest kohast.

Praegune ettevõttes kasutusel olev pärandrakendus puudutab eelkõige süsteemi hallatavust kui ka vanadust. Selles esineb mitmeid puuduseid ning see ei rahulda enam ettevõtte ärilisi soove. Peamine vajadus uuenenud süsteemile tekib asjaolust, et tehnoloogilise võla kuhjumise tõttu on selle haldamine ja arendamine muutunud ebaefektiivseks. Rakendus koosneb erinevatest komponentidest, mida kasutatakse erinevates kohtades ning on loodud mitmete moodustega. Selle tulemusena puudub süsteemil ühtne struktuur muutes administreerimise keeruliseks ja aeganõudvaks. Zone Media ei pea enam vajalikuks kulutada ressurssi pärandsüsteemi ajakohastamisele, vaid lihtsam on selle asemel luua uus ja ühtsem rakendus, mis võimaldaks tulevikus kõiki täiendusi ühtmoodi teha. Käesoleval rakendusel puudub ühtne bränditavus ja rakendamise käigus probleemidele leitud ajutised lahendused on osutunud püsivateks muudatusteks.

Eesmärgiks on luua süsteem, mis ennekõike eemaldaks eelmainitud probleemid, kuid looks ettevõttele ka lisaväärtust. Rakenduse loomisel on prioriteet kasutuskogemus. Kaardistada ettevõtte vajadused lähtuvalt olemasoleva teavituste süsteemi analüüsist ning nende põhjal luua uus keskne klienditeavituste rakendus. Arendatava lahenduse puhul oleks tähtis, et seda oleks tulevikus lihtne ühtlaselt täiendada ning ajakohasena hoida. Rakendus peab olema kooskõlas tänapäevaste kui ka ettevõttes määratud koodi kirjutamise parimate praktikatega.

2.3 Teema aktuaalsus ettevõttes

Seoses üha suureneva sõltuvusega tehnoloogiast ja kasvava nõudlusega katkematute teenuste järele on tänapäeva ärimaastikul jätkuvalt oluline klientide informeerimine. Seda eelkõige teenuste registreerimiste, muudatuste või uuendustega, mis võivad mõjutada ettevõtete äritegevust. Zone Media pakub veebimajutusteenuseid väga suurele kliendibaasile ning õigeaegsete ja täpsete teadete tähtsust ei saa märkamata jätta. Aastas väljub üle miljoni teavituse ettevõtte süsteemist hetkel e-kirjade kujul. Ajaga on aga

juurde tekkinud uusi kanaleid erinevate digitaalseadmete ja programmide kasutuselevõtuga ning konkurentsis püsimiseks on vaja pakkuda infot ka täiendavate edastusmeetodite kaudu. Klientide kasutuskogemuse ja töökorralduse efektiivsuse parandamiseks on vajadus kaasaegsema teavituste süsteemi järele, mida oleks lihtne ühtlaselt täiendada ja ajakohasena hoida.

2.4 Metoodika

Antud lõputöö raames analüüsitakse esmalt ettevõttes olevat pärandsüsteemi ning tuuakse välja selle peamised puudused Zone Media perspektiivist. Koostöös Zone Media arendustarkvara osakonnajuhiga koguti seejärel uue rakenduse funktsionaalsed ning mittefunktsionaalsed nõuded. Uue süsteemi nõuete kogumine toimus intervjuu käigus arendustarkvara osakonnajuhiga.

Analüüsitakse ka sobivaid tehnoloogiaid arenduseks ning sarnaseid olemasolevate kolmanda osapoolte lahendusi, tuues välja nende peamised eelised ja puudused. Eksisteerivate süsteemide analüüs annab ülevaate, kas turul on olemas sarnaseid lahendusi, mis kataksid ettevõtte vajadused ning mida saaks koheselt kasutusele võtta. Selgitatakse, miks eksisteerivad lahendused on puudulikud ning pakutakse analüüsi alusel ja nõutest lähtuvalt uus lahendus.

Rakenduse realiseerimise faasis selgitatakse arendusprotsessi tehnilisemaid detaile tagarakenduse arenduse poolelt, kirjeldades süsteemi eriosade töötamist ja tervikliku rakenduse tööprotsessi.

Rakenduse valmimisel võetakse pärandsüsteemile lisaks järk-järgult kasutusele uuenenud süsteem ning selle käigus on autoril võimalik koguda tagasisidet nii arendusmeeskonnalt kui ka klienditoelt, mis annaksid ülevaate loodava lahenduse puuduste kohta. Arendatava rakenduse juurutusprotsessi lõpuks loobutakse täielikult pärandsüsteemist.

Lõputöö raames hinnatakse valminud rakenduse kvaliteeti ja funktsionaalsust ning tuuakse välja võimalikud mõttekohad süsteemi edasiarendamiseks.

2.5 Töö skoop

Käesoleva lõputöö skoobis on uue süsteemi MVP (ing. k. *Minimal Viable Product*) ehk minimaalselt töötava teenuse loomine ja selle järkjärguline kasutusele võtmine teavituste edastamiseks ettevõttes Zone Media. Autor on vastutav tagarakenduse süsteemi loomisel, mis hõlmab endas klienditeavituste rakendust ja REST API rakendusliidest. Uue teenuse loomise käigus arvestatakse olemasoleva pärandsüsteemi arhitektuuriga. Töös on kirjeldatud arendusmetoodika, analüüs ja rakenduse teostus. Samuti kuulus töö skoopi pidevvalmiduse ja -integratsiooni lisamine, rakenduse optimeerimine ning olemasoleva andmebaasi tabeli täiendamine.

Lõputöö skoopi ei kuulu esirakenduse loomine ja disain. Zone Media klienditeavituste rakenduse esirakenduse arendus ning disain on loodud eelnevalt kliendipoolse arendusmeeskonna poolt. Teavituste genereerimine toimub peale iga teenuse loomise sammu taustal kasutades andmebaasi päringuid. Saadetud teavitused ilmuvad kliendi *my.zone.eu* esilehele „Teavitused“ lahtrisse. Töö skoopi ei kuulu ka rakendusele integratsiooni testide loomine, vaid piirdutakse ühiktestide kirjutamisega.

3 Klienditeavituse rakenduse analüüs

Käesolevas peatükis analüüsitakse pärandrakendust ja selle kitsaskohti, määratakse funktsionaalsed ja mittefunktsionaalsed nõuded rakenduse minimaalseks toimimiseks, analüüsitakse sarnaseid turul pakutavaid teenuseid ning valitakse välja sobivad tehnoloogiad klienditeavituste rakenduse elluviimiseks.

3.1 Olemasoleva rakenduse analüüs

Järgnevalt analüüsitakse Zone Media OÜ-s eksisteerivat pärandrakendust. Kaardistatakse vead ning puudused, mis annavad parema ülevaate vanast süsteemist, mida uue lahenduse elluviimisel silmas pidada ja vältida.

3.1.1 Pärandrakenduse tutvustus

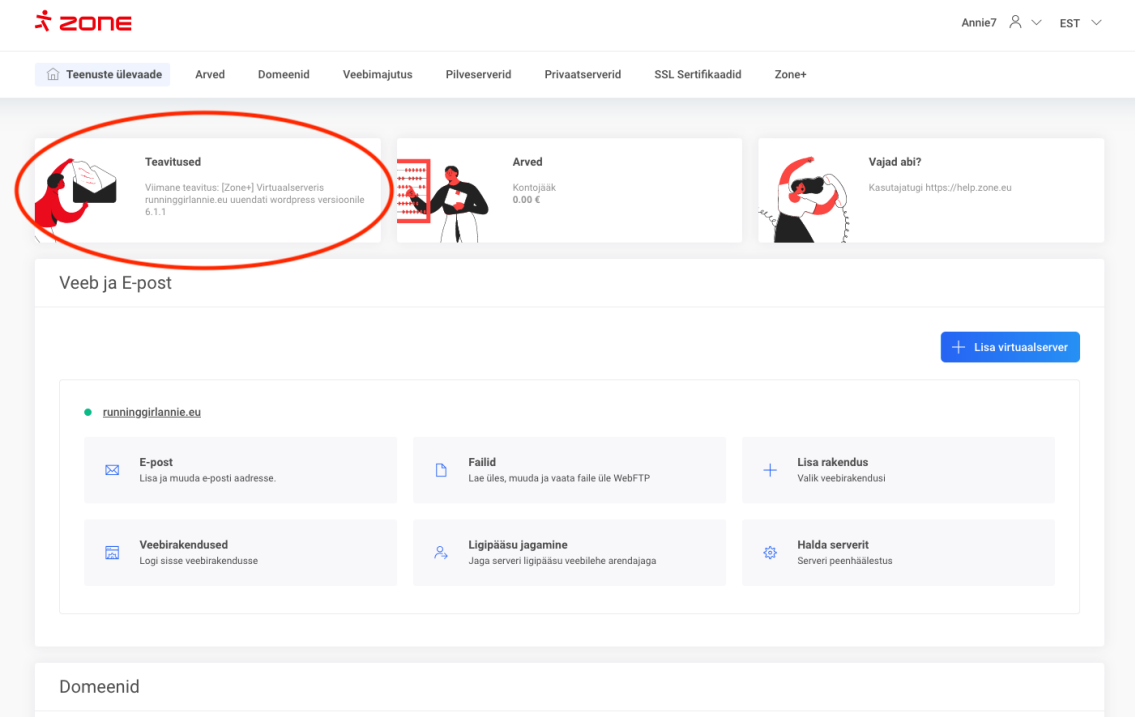
Peatükis 2.2 tutvustati sissejuhatavalt hetkel Zone Medias kasutusel olevat klienditeavituste haldamise süsteemi. Pärandrakendus, mida käesoleva lõputöö raames kaasajastatakse, koosneb kolmest eraldi seisvast süsteemist: teenustega seotud -, mahu- ja veebi *cron*-teavitused. Teenustega seotud teavituste süsteemi võib pidada väljuvate klienditeavituste emasüsteemiks. Vanimad osarakendused, teenustega seotud teavituste- ja veebi *cron*-teavituste süsteemid, on alates 2016. aastast käibel olevad rakendused. Mahuteavituste süsteem on kõige kaasaegsem osarakendus, mis on alates 2019. aastast kasutusel. Otsese integreeritavuse puudumise tõttu teenuste süsteemiga on mahuteavitused kaetud ajutiste lahendustega, mis aja jooksul on osutunud püsivateks muudatusteks. Hetkel saadetakse teavitused klientidele laiali mitmest kohast ja erinevate mooduste abil.

Pärandrakendus on hetkel ettevõtte Zone Media „kekseks“ teavituste väljasaatmis süsteemiks ning on arendatud kasutades PHP programmeerimiskeelt. Pärandsüsteemil on olemas töötav eesrakendus, mille arenduse taga on Zone Media kliendipoolne arendustiim. Taustal toimuvate andmebaasi päringute kaudu genereeritakse teavituste struktuur kliendile iga teenuse sammu juures ning info jõuab andmebaasi *Notifies* tabelisse. Teavituste välja saatmiseks on kasutusel mitu erinevat *cron*-tööd tänu millele

jõuavad teavitused kliendini e-posti teel. Edastatud kirjad kuvatakse kliendile tema Zone avalehel, mille eest vastutab API lõpp-punkt:

```
GET https://my.zone.eu/zone-id-api/v2/me/notification
```

pärides andmebaasi *Notifies* tabelist konkreetsele kliendile laekunud teated. Teavitused on nähtavad *my.zone.eu* avalehel „Teavitused“ lahtris (Joonis 1).



Joonis 1. Kliendi vaatest Zone avalehele ilmuvad teavitused.

3.1.2 Rakenduse puudused Zone Media perspektiivist

Kaardistamaks pärandsüsteemi puuduseid Zone Media perspektiivist, viidi läbi suuline intervjuu tarkvaraarenduse osakonna juhiga. Selle käigus leiti, et pärandrakendus on kirjutatud vastavalt oma ajastule ning kaasajastamine on arvatust ressursimahukam.

Pärandsüsteemil on ka palju aegunud funktsioone, mida arenduses enam ei kasutata ning mis muudavad rakenduse halduse puudulikuks. Järgnevalt on väljatoodud kitsaskohad käesoleva süsteemi komponentide vahel:

Teenustega seotud teavitused on peamine komponent teavituste süsteemist. Antud osarakendus on kirjutatud järgimaks PHP *Pear Mail_MIME* loogikat, mis on ettevõtte siseselt kohandatud vastamaks Zone Media standarditele. *MIME* kujutab endast turvalise meilivahetuse standardit, kus kirjade krüpteerimine, signeerimine, verifitseerimine ja

dekrüpteerimine toimuvad e-posti meiliprogrammide poolel [6]. *Mail_MIME* pakub valmis klasse *MIME*-sõnumite ja töötlemisega tegelemiseks [7]. Kuna viimane uuendus *Pear Mail_MIME*-le on enam kui poolteist aastat tagasi tehtud [7] kujutab see endast suurt turvariski, mida arenduses peaks silmas pidama.

Antud osarakenduse teavituste mallid on brändidud vastavalt brändile, mis omakorda jagunevad keelepõhiselt. Mallid on kirjutatud *Markdown*-is sisaldades endas PHP koodi komponente. Koodistiili kohapealt peaks äriloogika olema eraldi väljundist. Teavitused on puhtalt üles ehitatud kliendi ID-le ning olukorras, kus see puudub, ei oska süsteem andmetega midagi peale hakata. Teavitused saadetakse klientide välja *cron*-i töö jooksul, mis pärib andmebaasi tabelist *Notifies* andmeid iga kahe minuti tagant.

Teenustega seotud teavituste alla kuuluvad ka Zone Media klienditoe poolt välja saadetavad ühekordsed kirjad, mis on seotud hooldustöödega, turvariskide ning ka personaliseeritud teadetega erinevatele klientidele. Need luuakse klienditoe töötajate poolt ning vastavalt vajadusele. Sarnaselt mallidele, kasutavad need märgistuskeelena *Markdown*-i ning lisatakse ühtlasi *Notifies* tabelisse. Ainus erinevus on see, et neid teavitusi saadetakse välja kolmanda osapoole kaudu kasutades *Baboon* tarkvara. Peamine puudus on ühekordsete teavituste eristamine üksteisest, kuna rakendus ei suuda tuvastada, millised teavitused peavad klientideni enne jõudma kui teised.

Mahuteavitused on seotud eelkõige virtuaalserveri pakettides oleva MySQL andmebaasi suuruse piiranguga. Teavitusi saadetakse klientidele laiali ennetavalt kui andmebaas:

- hakkab lähenema piirmahule,
- on ületanud piirmahu,
- või kui kõikide andmebaaside, kõvakettaruumi ja e-posti aadresside maht ületab paketis lubatud kõvakettamahtu.

Mahuteavituste saatmisel on kasutusel *Symfony SwiftMailer* sõltuvus, mille arendus lõpetati 2021. aasta novembris [8]. Kuna nende teavituste puhul kastutatakse teistsugust *cron*-tööd võrreldes teenustega seotud teavitustega, siis sageli lisatakse ajutiste lahenduste abil need teavitused andmebaasi *Notifies* tabelisse. Mahuteavitustel on olemas oma eraldi tabel andmebaasis. Teavitus mallid kasutavad *Twig*-komponendil põhinevat süsteemi.

Veebi *cron*-teavitused on ainsad teavitused Zone Medias, mis saadetakse lihtteksti kujul ning ei ole bränditavad. Saatmisel kasutatakse PHP *mail* sisseehitatud funktsionaalsust. Neid teavitusi ei ole võimalik ühildada ka *Notifies* andmebaasi tabeliga ning puuduvad mallid teavituse edastamisel.

Järgnevalt tuuakse välja pärandrakenduse süsteemi komponentide kokkuvõttev võrdlus. Võrdluspunktideks on pärandrakenduse süsteemi komponentide puudused, mis võetakse arvesse uue lahenduse loomisel (vt Tabel 1).

Tabel 1. Pärandrakenduse süsteemi komponentide puuduste kokkuvõte.

Tehnoloogia tüüp	**Puudused**
Teenustega seotud teavitused	- Teavituste mallid on kirjutatud *Markdownis*, kuid seovad endas PHP koodi; - Saatmiseks kasutatakse PHP *Pear Mail_MIME* loogikat, mille viimane uuendus oli üle pooleteise aasta tagasi; - Teavitused on bränditavad, kuid kliendi ID puudumisel, ei osata hetkel teadet välja saata; - Ühekordsetel teavitustel puudub eristamise võimalus;
Mahuteavitused	- Saatmiseks kasutatakse *Symfony SwiftMailerit*, millele puudub uuenduste tugi; - Osad teavitused lisatakse *Notifies* tabelisse *ad-hoc* meetodipõhjal mahuteavituste tabelist; - Puudub integreeritatavus teenustega seotud teavituste süsteemiga; - Bränditavus teavitustel vaid läbi *ad-hoc* meetodite;
Veebi *cron*-teavitused	- Kasutatakse lihtteksti sisu; - Saatmiseks on kasutusel PHP oma meetod *mail*; - Puudub bränditavuse tugi;

Tarkvaraarenduse osakonna juhi perspektiivist on eraldi komponentidel põhinev rakendus ebaefektiivne. Ajutised lahendused, mis on asunud püsivateks muudatusteks, ei ole dünaamiline lähenemine ning uue arendus lisa korral tuleb ümber kohendada süsteemi. See omakorda suurendab ka andmebaasi koormust, sest päritakse korraga suures koguses andmeid erinevatest tabelitest, et kätte saada klientidele edastamiseks mõeldud teavitused.

3.2 Nõuete määramine uuele lahendusele

Uue klienditeavituste rakenduse arendamisel on oluline määrata nõuded, mis aitaksid tagada selle funktsionaalsuse ja kasutusmugavuse. Selleks et autoril oleks täpsem ülevaade ettevõtte loodava klienditeavituste rakenduse vajadustest, koguti ettevõtte tarkvaraarenduse osakonnajuhilt intervjuu analüüsi käigus funktsionaalseid ja mittefunktsionaalseid nõudeid.

3.2.1 Funktsionaalsed nõuded

Järgnev loetelu sisaldab funktsionaalseid nõueteid, mis määravad kriteeriumid, mida süsteem peab olema võimeline tegema [9]:

- Rakendusel peavad olema seadistatud automaattestid.
- Rakendusele pääsevad ligi ainult Zone Media töötajad, kellel on olemas volitatud õigused.
- Rakendus peab esialgu olema võimeline saatma e-kirju klientidele.
- Rakendus peab toetama kirjade saatmist ka muude kanalite kaudu.
- Rakendus peab suutma töödelda vana struktuuri kirju.
- Rakendus peab võimaldama pärandrakenduse funktsionaalsust ümber teisendada.
- Rakendus peab oskama tagasipõrganud teavitusi kinni püüda.

3.2.2 Mittefunktsionaalsed nõuded

Järgnev loetelu sisaldab mittefunktsionaalseid nõudeid, mis keskenduvad süsteemi käitumise täpsustamisele, mitte konkreetsete funktsionaalsuste või omaduste kirjeldamisele [9]:

- Tagarakendus peab olema kirjutatud PHP uusima versiooni programmeerimiskeeles.
- Rakendus peab olema tugevalt tüübitud (ing. k. *strictly typed)*.
- Rakenduse koodistiil peab vastama *PSR* standarditele.

- Rakenduse kõik otsesed sõltuvused peavad olema usaldusväärsetest allikatest.
- Rakenduse raamistiku välised sõltuvused peavad olema kooskõlastatud Zone Media Infoturbe meeskonnaga.
- Rakenduse arhitektuur ja koodistruktuur peavad olema kooskõlas ettevõttes kasutatava koodi kirjutamise parimate praktikatega.
- Uus süsteem peab olema dokumenteeritud vastavalt ettevõtte poolt loodud dokumentatsiooni mallile.
- Rakenduse teavituste mallid tuleb üle viia *.md.twig* laienditele ehk kasutades *Markdown*-i ja *Twig*-i.
- Tagarakendus peab olema arendatud Symfony raamistikul.
- Tagarakendus peab kasutama *Symfony Mailer* komponenti teavete edastamiseks.
- Andmebaas peab kasutama MariaDB andmebaasisüsteemi.
- Rakendusliides peab olema kooskõlas *OWASP* standardite ja nõuetega.
- Rakendusliides peab vastama RESTful ja *OpenAPI* standarditele.

3.3 Rakenduses kasutatavate tehnoloogiate analüüs

Rakenduse loomise tõhusus ja kvaliteet sõltuvad peamiselt tehnoloogia valikust. Käesolevas peatükis tutvustatakse tehnoloogiaid, mis võetakse kasutusele uue rakenduse arendamisel. Samuti tuuakse välja alternatiivsete tehnoloogiate valiku võimalused, mis võiksid lahendada käesolevaid probleeme.

Valik arendusvahendite osas on piiratud Zone Media suunistega, mis määravad aktsepteeritud tehnoloogiad, mida saab ning mis sobivad töös käsitletava süsteemi loomisel. Rakenduse juurutamisel on kasutusel programmeerimiskeelena PHP, raamistik Symfony ning andmebaas MariaDB. Lähtekoodi versioonihalduseks on kasutatud Git versioonihaldustarkvara ja hoidlana on kasutusel Atlassian-i loodud keskkonna Bitbucket.

3.3.1 Rakenduse programmeerimiskeele valik

Uue süsteemi tagarakenduse arendusel kasutatakse programmeerimiskeelt PHP, mis on kooskõlas ettevõtte arendusosakonna eelistustega. Rakenduse hooldusel ja funktsionaalsuste lisamisel saab edaspidi kasutada Zone Media arendusosakonna töötajaid, mis on kuluefektiivsem variant, kui väljaspool ettevõtet ressursse sisse osta, et rakendus hallata.

PHP on laialdaselt kasutatav avatud lähtekoodiga serveripoolne skriptimiskeel, mis sobib hästi veebiarenduseks [10]. Kood genereeritakse serveri poolel ning seejärel serveeritakse HTML-i kujul veebibrauseris [11].

Alternatiivsed programmeerimiskeelte valikud on Java ja Python, mis on kasutusel vähesel määral ka Zone Media siseselt arenduses.

Java on avatud lähtekoodiga ja platvormist sõltumatu üldotstarbeline objektorienteeritud programmeerimiskeel. Javas kirjutatud programme on lihtne laiendada, võimalik kergesti skaleerida vastavalt vajadustele ning neid on lihtne hooldada [12].

Python on laialdaselt kasutatav üldotstarbeline objektorienteeritud programmeerimiskeel, mida kasutatakse sageli veebi- ja tarkvaraarenduses. Sellel programmeerimiskeelel on lihtne süntaks, mis sarnaneb loomulikule keelele, mida on lihtne lugeda. Populaarne programmeerimiskeel, eriti alustavate programmeerijate seas [13].

Lõputöö autoril on kokkupuude mõlema alternatiivse programmeerimiskeelega. Pythoni kasutuskogemus on võrreldes Java programmeerimiskeelega madal. Võttes arvesse, et programmeerimiskeele õppimine on aeganõudev ning lahenduse loomiseks mõeldud aeg piiratud, ei ole ressursseide puudusest tulenevalt autoril võimalust endale Pythonit uuesti selgeks teha. Java programmeerimiskeelega on kokkupuude ja teadmised märksa paremad, kuid rakenduse arendusel ei pruugi need olla piisavad. Arvestades ettevõtte poolseid nõudmisi rakenduse tehnoloogiate osas ning asjaolu, et autor töötab igapäevaselt PHP programmeerimiskeelega, langeb valik PHP kasuks.

PHP programmeerimiskeele eeliseks on ka see, et pärandrakendus on arendatud PHP keeles ning loodava lahenduse sidumine sellest tulenevalt palju kergem.

3.3.2 Rakenduse raamistiku valik

Kuna rakenduse programmeerimiskeel on piiritletud PHP-ga, siis järgnevalt on analüüsitud ka vastavaid raamistikke. 2023. aastal ettevõtte Atatius poolt tehtud uuringule tuginedes on edetabeli esikolmiku põhjal PHP-raamistike valik järgmine: Symfony, Laravel, CodeIgniter [14].

Symfony on avatud lähtekoodi ja tasuta tarkvaraga kättesaadav PHP veebirakenduste raamistik MVC rakenduste jaoks, mis on välja antud MIT litsentsi alusel [15]. Esmane versioon tuli avalikkuse ette arendaja Fabien Potencieri poolt 2005. aastal [16]. Symfony on loodud veebirakenduste arendamise optimeerimiseks mitme põhifunktsiooni kaudu. Lisaks on olemas arvukalt tööriistu ja klasse, mille eesmärk on lühendada keeruka veebirakenduse arendusaega ning automatiseerida levinumaid toiminguid [17]. Symfony ühildub enamiku saadaolevate andmebaasimootoritega, sealhulgas MySQL, PostgreSQL, Oracle ja Microsoft SQL Server ning töötab mitmel platvormil [17].

Laravel on samuti avatud lähtekoodiga ning üks laialdasemalt levinud PHP raamistikke, mis jälgib MVC disaini mustrit. Raamistik loodi aastal 2011 Taylor Otwelli poolt ning olnud senini pidevas arenguprotsessid. Otwelli soov oli luua alternatiivne raamistik CodeIgniter-ile. Laravel on progressiivne ja üks lihtsamaid raamistikke, millega saab luua keerukaid ja kiireid rakendusi, kasutades kaasaegseid funktsioone [18].

CodeIgniter on avatud lähtekoodiga võimas väga väikesemahuline PHP raamistik, mis on loodud arendajatele, kes vajavad täisfunktsionaalsete veebirakenduste loomiseks lihtsat ja elegantset arendusvahendite komplekti [19]. Raamistik pakub kiireid andmebaasitoiminguid võrreldes teiste PHP raamistikega, kõrget jõudlust ning ei põhine täielikult MVC arhitektuuril, kus mudeli ja vaate klassid on valikulised, kuid kontroller klassid kohustuslikud [20]. 2006. aasta veebruaris tuli USA tarkvarafirma EllisLab turule esimese versiooniga CodeIgniter-ist [21]. CodeIgniter pakub oma sisse-ehitatud mallimootorit ning ühildub mitmete andmebaasimootioritega sealhulgas MySQL, PostgreSQL, Oracle, SQLlite, MSSQL [22, 20].

Võrreldes omavahelisi võimalusi, mida pakuvad edetabeli kolm parimat PHP raamistikku, on igalühel omad eelised ja puudused. Autoril puudub kogemus CodeIgniter raamistiku kasutamisel ning valik langetatakse Symfony ja Laravel raamistiku vahel, millega on olnud varasem kokkupuude. Ühtlasi on Zone Media arenduses kasutusel

mõlemad raamistikud. Arvestades asjaolu, et nõudena on esitatud loodava rakenduse puhul Symfony raamistiku kasutamine, siis valikust langeb välja Laravel.

3.3.3 Rakenduse andmebaasi valik

Andmebaasimootorina kasutatakse MariaDB relatisoonilist andmebaasi. MariaDB üks populaarsematest avatud lähtekoodiga relatsioonilisi andmebaasisüsteeme, mis on algselt loodud MySQL arendajate poolt 2009. aastal ning on osa enamikest pilvepakkujatest ja vaikimisi kasutusel enamikes Linux-i distributsioonides [23].

Vabavaralistest relatsioonilistest andmebaasisüsteemidest on üheks alternatiivseks variandiks PostgreSQL. Tegemist on võimsa avatud lähtekoodiga relatsioonilise andmebaasisüsteemiga, mis kasutab ja laiendab SQL päringukeelt, et andmeid turvaliselt salvestada. PostgreSQL on skaleeritav ka kõige keerukamate jõudlust nõudvate lahenduste tarbeks. PostgreSQL projekti arendust alustati 1986.aastal *University of California at Berkeley* ülikoolis [24].

Mõlemad andmebaasimootorid, MariaDB ja PostgreSQL, on sarnased võttes arvesse ka mõningaid erinevusi ning sobilikud rakenduse loomisel. Eelnevatest peatükkidest väljatulnud nõuete ja analüüsi tulemusena kasutatakse ettevõtte poolt juba seadistatud andmebaasi. Peamiselt Zone Media arendusvahendite suunistest tingituna otsustati andmebaasimootori valikuna MariaDB tarkvara kasuks.

3.3.4 Integreeritud arenduskeskkonna valik

PHP programmeerimiskeelel ja Symfony raamistikul põhineva rakenduse arendamiseks on kõige populaarsemad neli arenduskeskkonda: PhpStorm, Apache Netbeans, Sublime Text ja Notepad++ [25].

PhpStorm ja Apache Netbeans on integreeritud arenduskeskkonnad (*IDE*) ning Sublime Text ja Notepad++ on koodiredaktorid. Integreeritud arenduskeskkonna ja koodiredaktori eristuspiir on väga hägune. Siiski, nende vahel on mõningaid erinevusi. Integreeritud arenduskeskkond on iseseisev pakett, mis võimaldab koodi kirjutada, kompileerida, käivitada ja siluda ühes mugavas kohas. Üldiselt on IDE keskendunud ühele keelele sisaldes sellele omast kompilaatorit ja silurit. Teisest küljest on koodiredaktor nagu tekstiredaktor, millel funktsioonid lihtsustavad koodi kirjutamist kas naiivsete võimaluste või valikuliste pistikprogrammide kaudu. Koodiredaktorid on oma olemuselt üldisema

otstarbega suutes korraga töötada mitme programmeerimiskeele kallal. Arenduskeskkonna valiku eelistamine on suuresti isiklike mugavuste, konkreetsete programmeerimiskeelte ja töövoo küsimus [25].

Järgnevad lõigud kirjeldavad võrdlusi erinevate arenduskeskkondade ja koodiredaktorite vahel:

JetBrains-i poolt loodud PhpStorm, varasemalt tuntud ka kui „Web IDE“, on kasutusel aastast 2010 [26]. PhpStorm toetab Mac OSX, Linux kui ka Windows operatsioonisüsteeme ning sõltumata operatsioonisüsteemidest on neil kõigil kasutusel samad versioonid. Arenduskeskkonna kasutamine on tasuline (individuaalkasutajale alates 99 € kuus) [27], kuid Tallinna Tehnikaülikooli tudengina on selle kasutamine ning allalaadimine tasuta. Lisaks pakub PhpStorm tuge alates PHP versioonist 5.3, silumis- ja testimistööriistu, samuti paljude PHP raamistike ja tehnoloogiate tuge. PhpStormi võimas refaktoreerimistööriist aitab parandada koodi kvaliteeti ja hooldatavust [28].

Apache Netbeans on avatud lähtekoodiga integreeritud arenduskeskkond, mis pakub tuge erinevate programmeerimiskeelte jaoks. Üliõpilasprojektina 1996. aastal Tšehhis alguse saanud ning algselt nime kandnud projektist Xelfi on väljakasvanud populaarne arenduskeskkond. Eesmärk oli kirjutada Java programmeerimis keeles *Delphi*-laadne Java IDE. Netbeans nimi tuleneb sõnadest *Network* ja *Java*, mille idee autor on Jaroslav Tulach. Tema kujundas ühtlasi ka IDE põhiarhitektuuri, et kirjeldada, mida komponendid teevad ning IDE oleks nende edastamis viis [29]. Apache NetBeans toetab Windows, Linux, Mac OSX ja BSD operatsioonisüsteeme. Lisaks pakub nutikaid redigeerimistööriistu ning võimaldab kasutajatel hõlpsasti ümber kujundada koodi. Üldised redaktori funktsioonid hõlmavad koodi vormindamist ja ahendamist, kohandatud klaviatuuri otseteid, nutikat koodi lõpetamist, nime- ja parameetrisoovitusi ning palju muid funktsionaalsuseid [30].

Sublime Text on lähtekoodiredaktor, mis on kirjutatud C++ ja Python-i programmeerimiskeeles. Üle kümne aasta tagasi, 2008. aasta jaanuaris [31], alguse saanud arenduskeskkond toetab nii Mac OSX, Windows kui ka Linux operatsioonisüsteeme [32]. Kasutajatel on võimalik kujundada koodiredaktorit oma äranägemise järgi erinevate teemadega või laiendada selle funktsionaalsust pistikprogrammidega. Üldjuhul on need kogukonna loodud ning neid hooldatakse tasuta

tarkvaralitsentside alusel. Koodiredaktor kasutab minimaalset liidest sisaldades programmeerijatele mõeldud funktsioone, sealhulgas konfigureeritava süntaksi esiletõstmist, koodi ahendamist, terminali väljundakent ja palju muud. Sublime Text-i on võimalik alla laadida tasuta ning viia end süsteemiga kurssi, kuid professionaalse kasutusepoole pealt on siiski vajalik litsentsi olemasolu. [33].

Notepad++ on avatud lähtekoodi redaktor, mis on kirjutatud C++ keeles ning põhinedes võimsal redigeerimiskomponendil Scintilla. Notepad++ kasutab puhast Win 32 API-t ja *STL*-i, tagades suurema täitmiskiiruse ja väiksema programmimahu. Koodiredaktorit esitleti esmakordselt 2003. aasta novembris, mil arendaja Dan Ho sellega avalikkuse ette tuli [34]. Notepad++ toetab vaid Windows operatsioonisüsteemi ning selle kasutamist reguleerib GNU üldine avalik litsents [35].

Eelmainitud arenduskeskkondade vahelised erinevused põhinevad funktsionaalsuse, keerukuse ja kohandamise tasemel. IDE-d, PhpStorm ja Apache NetBeans, pakuvad koodi kirjutamisel eelistatud funktsioone ja tööriistu ning on loodud tagama terviklikuma arenduskeskkonna. Koodiredaktorid, Sublime Text ja Notepad++, pakuvad seevastu lihtsamat liidest ning on sageli kergemini kohandatavad vastavalt projektile [36]. Kuna lõputöö autori enimkasutatud arvuti on MacOS operatsioonisüsteemiga, langeb arenduskeskkonna valikust välja Notepad++. Lähtudes sellest, et uue arenduskeskkonna selgeks tegemine on ajakulukas ning rakenduse loomiseks mõeldud aeg on piiratud, ei ole mõeldav käesoleva lahenduse käigus uue arenduskeskkonna kasutust endale selgeks teha. Seetõttu välistatakse arenduskeskkonna valikuna ka Apache NetBeans. Autori kogemus ja teadmised piirduvad kahe arenduskeskkonnaga: Sublime Text ja PhpStorm. Sublime Text ja PhpStorm on mõlemad väga head arenduskeskkonnad projektide loomisel. Ehkki PhpStorm pakub kõrgtasemeliseks arenduseks rohkem võimalusi Sublime Text-ist, on arenduskeskkonna konfigureerimine virtualiseeritud keskkonnas keerulisem SSH-ühenduse keerukuse tõttu. Seda asjaolu silmas pidades, on autoril mõistlik valida Sublime Text arenduskeskkonnaks. Ettevõttel on selleks olemas ka vastav litsents.

3.4 Sarnaste eksisteerivate lahenduste analüüs

Teavituste koostamine ning saatmine on ettevõtete puhul üks levinumatest protsessidest hoidmaks sidet oma klientidega. Paraku on ettevõtted, kes ei oma teadmisi ja oskuseid

koodi kirjutamisest ning puuduvad vajalikud ressursid, et koostada teavitusi firma siseselt. Töö lihtsustamiseks on võimalus kasutada kolmanda osapoole teavitusrakendusi. Selleks uuriti lähemalt turul olemasolevaid teavitussüsteeme ning tutvustati nende peamiseid eeliseid ja puuduseid. See annab parema ülevaate, kas on olemas teenuseid, mis oleksid kooskõlas ja vastaksid ettevõtte nõuetele ning lahendaksid käesolevaid probleeme.

Leiti turul eksisteerivaid teavitussüsteeme, mis lihtsustaksid teavituste loomist, kujundamist ja saatmist. Analüüsiti olemasolevaid süsteeme nende kirjelduse, funktsionaalsuse ning hinnakvaliteedi suhte põhjal. Lähemalt uuriti kolme teenust:

1. Twilio SendGrid,
2. Mailgun,
3. Postmark.

Kõikide eksisteerivate süsteemide taga on enamasti konkreetsele teenusele spetsialiseerunud meeskonnad. Sageli võivad need olla usaldusväärsemad ja kulutõhusamad kui ettevõttesiseselt luua ning evitada teavitussüsteemi. Arvestada tuleb ka riistvara, tarkvara ja personali kuludega. Ettevõtte kasvades suureneb ka saadetavate teavituste arv ning tehingupõhised meiliteenused saavad hakkama suure hulga teavitustega, põhjustamata viivitusi või süsteemi kokku kukkumist. Välja toodud teenused pakuvad sarnast lähenemist, kus ettevõte saab kolmanda osapoole rakenduse integreerida oma süsteemi, luua vastavalt enda soovidele teavituste põhjad, need salvestada ning andmebaasist tulevate andmete põhjal teavitused kokku panna. Esmapilgul sobiksid antud teenused lahendamaks töös püsitatud probleeme. Suurimaks erinevuseks võib pidada nende lahenduste hinda, mis on iga süsteemi puhul eraldi välja toodud.

Twilio SendGrid on pilve SMTP (*Simple Mail Transfer Protocol*) pakkuja, mis toimib turunduse ja tehnilise meiliedastus mootorina. Lahendatakse probleeme, millega seisavad silmitsi ettevõtted, kes saadavad tehingupõhiseid väljaminevaid e-kirju sealhulgas tarkvararakenduste kaudu edastatavaid meile, nagu registreerumise kinnitused, tarnehoiatused ja teatised [37]. SendGrid (eelnevalt SMTP API) alustas lihtsa SMTP pakkujana 2009. aastal [38]. Twiloga alustati koostööd 2012. aastal lisades SMS-sõnumid

ja tõukemärguanded. 2018. aastal sai Twilio haldusõigused, mille järel nimetati teenus ümber Twilio SendGrid-iks [39]. Teenuse võimaluste hulka kuulub ka malli loomise redaktor, mis annab kasutajale vabaduse kasutada valmis loodud lahendusi või lisada kohandatud HTML koodi [40]. Zone Media kasutab oma loodud ja bränditud malle ning Twilio Sendgrid võimaldab neid kergesti teenusele üle tuua. Lisaks pakub teenus *A/B*-testimis võimalust, et kontrollida kliendile saadetavate teavituste õigsuses [41]. Platvormi teenuste kasutamine on üpriski kulukas. Eeldades, et Zone Media saadab kuus ligikaudu 120 000 teavitust ning ettevõttele sobiv Pro 100K versiooni hinnaks kujuneks 89.95 $ kuus ilma käibemaksu arvestamata [42].

Mailgun on pilvepõhiste tehingute e-posti API teenus, mis on loodud selleks, et aidata arendajatel veebilehitsejatelt või rakendustest e-kirju saata, jälgida ja vastu võtta. Funktsioonide hulka kuuluvad sissetuleva meili marsruutimine, *A/B*-testimine, pakettsaatmine (ing. k. *bulk sending*), e-kirjade isikupärastamine ja ajastamine [43]. Nagu ka Twilio SendGrid, võimaldab Mailgun isikupärastada saadetavaid teavitusi, kas valitud mallide hulgast või genereerides HTML-koodi põhiselt. Lisaks e-posti põhiselt on võimalik saata teavitusi ka SMS sõnumitena üle Mailgun SMS API ning veebihaakide (ing. k. *webhook*) tuge, mida saab kasutada teadete edastamisel välistes süsteemides nagu Discord, Slack [44]. Võrreldes Twilio SendGrid kuumaksega, on selle platvormi teenuste kasutamine kallim. Ettevõttele sobiv *Scale* versioon, mis toetab kuni 100 000 e-kirja saatmist, on kuutasuga 90 $ ning iga 1000 e-maili kohta on lisatasu 0.80 $ [45].

Postmark on SaaS-i (*Software as a Service*) meiliteenuse pakkuja rakendusmeilide edastamiseks, mis pakub SMTP-teenuseid, e-posti API-d ja muid tehingupõhiseid meiliteenuseid ettevõtetele. Pakutavad teenused piirduvad vaid meilidega ning muud teavituskanalid ei ole kasutatavad [46]. Postmark võimaldab samamoodi kohandada vastavalt Zone Media nõutele bränditud teavituste malle nagu Twilio Sendgrid ja Mailgun, kuid teavituste testimine teenuse siseselt puudub [46]. Postmark teenuste kasutamine on võrreldes eelneva kahe platvormiga märksa kallim. Ettevõttele sobiv versioon, mis toetab ligi 125 000 e-kirja saatmist, maksab ligikaudu 115 $ kuus [47].

3.4.1 Eksisteerivate lahenduste analüüsi tulemus

Analüüsi käigus teostati ülevaade eksisteerivate lahenduse võimalustest ning uuriti, kas need oleksid kooskõlas ettevõtte vajaduste ja nõuetega. Tulemustest selgus, et Twilio SendGrid, Mailgun ja Postmark on kõik suurepärased välised teavete haldamise jaoks

loodud lahendused. Neil on kõigil omad plussid ja miinused. Ettevõtte nõuetest lähtuvalt jõuti otsusele, et kolmanda osapoole rakendusi arendusse ei kaasata. Zone Medial endal on nii suur võimekus ja kompetents kirjade välja saatmisel, et välise teenuse kasutamine piiraks teenuse haldamist ning suurendaks oluliselt töömahtu teavitusmallide ümber töötlemisel. Välise teenuse kasutamise pluss on vaid juurutamise kiirus. Zone Medial on olemas oma meiliserveri tarkvara, mida maailmas kasutavad tuntud teenusepakkujad.

Uuenenud klienditeavituste rakendus luuakse ja personaliseeritakse vastavalt Zone Media OÜ nõuetele. See välistaks selle, et Zone Media sõltub kolmanda osapoole süsteemist, vaid omab kontrolli oma rakenduse üle. Tänu sellele ei ole piirangut pakutavate funktsioonide ja funktsionaalsusega ning andmete privaatsus ja turvalisus on ettevõtte siseselt lahendatud. Lisanõuete tekkimisel on klienditeavituste rakenduse edasiarendus lihtne, kuna rakenduse maht on üldpildis väike.

3.5 Analüüsi kokkuvõte

Analüüsis käsitleti funktsionaalseid ja mittefunktsionaalseid nõudeid, teostati pärandrakenduse analüüs ja leiti puudused, mille alusel lõputöö raames rakendust arendama hakatakse. Analüüsiti sarnaseid eksisteerivaid lahendusi, mida võiks kasutusele võtta spetsiifilise koodi kirjutamise asemel. Võttes arvesse Zone Media võimekust ja kompetentsi kirjade saatmisel ning koostamisel, otsustati luua pärandrakendusest tulenevalt uus rakendus.

Pärandrakenduse kaasamine uue lahenduse loomisel annab võimaluse kasutada eelnevalt loodud andmebaasi ning sellega kaasnevaid tabeleid, üldist rakenduse struktuuri ja loogikat, mis säästab üheltpoolt aega ning teiselt poolt võimaldab ettevõtte spetsiifiliselt rakendust arendada.

Serverrakenduse programmeerimiskeele valikuks osutus PHP, mida lõputöö autori oma igapäeva töös kasutab. PHP programmeerimiskeele teadmiste täiendamine on selle võrra lihtsam. Rakenduse raamistiku valiku võrdlus jäi kahe enam levinud raamistike vahel, Laravel ja Symfony, ning tulenevalt ettevõtte poole määratud nõuetest valiti selleks Symfony. Andmebaasiks on Zone Media poolt kasutusel olev MariaDB. Rakendus kirjutatakse valmis integreeritud arenduskeskkonna ja koodiredaktori vahel valituna

Sublime Text 3 peale. Koodihaldus toimub ettevõtte Atlassian Jira Git-i koodihoidlas, kuhu lükatakse muudatused ning mis on kättesaadavad Zone Media volitatud töötajatele.

4 Klienditeavituste rakenduse tehniline teostus

Käesolevas peatükis kirjeldatakse tehnilist teostust serverrakenduse arendamisel. Antakse üldine ülevaade rakenduse arendusmetoodika ja arhitektuuri kohta. Keskendutakse ka rakenduse pidevvalmiduse ja pideva integratsiooni lisamisele, optimeerimisele, turvalisusele ja testimisele.

4.1 Arendusmetoodika

Uue serverrakenduse loomisel kasutati peamiselt agiilset arendusmetoodikat. Agiilne tarkvaraarendus pakub struktuurset ja iteratiivset lähenemist tarkvaraarendamisel ning projektide haldamisel [48]. Arendus toimus iteratiivselt ehk osa tööst valmis saamisel, korraldati koosolek ning vaadati koos tarkvaraarenduse osakonnajuhi ja juhendajaga töö üle. Koosoleku käigus lisati vajadusel täiendused rakendusele, määrati uued tööülesanded ning kavandati koosoleku plaan järgnevaks korraks.

Rakenduse arendusel kasutati *Test-Last Development* tarkvaraarenduse metoodikat. Metoodika põhineb selle, et esialgu luuakse äriloogika ning seejärel kirjutatakse valmis testid, mis põhinevad sellel äriloogikal. Selline arendusviis hoiab ära ka selle, et hilisemas ajajärgus ei hakata testide põhjal koodi ümber kirjutama, vaid koodist lähtuvalt luuakse testid. Kuna testide kirjutamisel ei ole esialgu teada, mida täpselt testitakse, siis toimub sellise arendusprotsessi loomine alles lõpujärgus [49]. Zone Media arendusmeeskond kasutab sellist lähenemist oma igapäeva arendustöös.

4.2 Serverrakenduse arendus

Serverrakendus ehk tagarakendus vastutab nii rakendusele vajalike andmete pärimise, salvestamise ning äriloogika eest. Rakendus on ülesehitatud kasutades Symfony raamistikku.

4.2.1 Symfony rakenduse ülesseadmine

Serverrakenduse loomisel võeti aluseks Symfony raamistiku dokumentatsiooni peatükk, mis käsitleb uue projekti ehitamist [50]. Esialgne rakendus genereeriti abitööriista *Composer* abil käsurealt, mis loob minimaalse valmisrakenduse ehk skeletoni koos vajalike sõltuvustega (Joonis 2). Symfony projekti versiooni 6.2 ülesseadmine eeldab vähemalt PHP versiooni 8.1 ning sõltuvuste haldamise tööriista *Composer*-it [51].

```
composer create-project symfony/skeleton:"6.2.*" notifications
```

Joonis 2. Symfony raamistikul põhineva projekti loomine Composer-i abil.

Skeleton sisaldab endas rakenduse kataloogi struktuuri, genereeritud põhikatalooge ja faile ning laetakse alla mõned sõltuvused (Joonis 3). Arenduse algusfaasis on oluline rakenduse kataloogi struktuuri paika saamine, mis mõjutab hiljem rakenduse hooldatavust, skaleeritavust ja kasutusmugavust [52].

```
notifications/
├─ bin/
│  └─ console
├─ config/
├─ public/
│  └─ index.php
├─ src/
│  └─ ...
├─ var/
│  ├─ cache/
│  ├─ log/
│  └─ ...
├─ vendor/
└─ .env
└─ .gitignore
└─ symfony.lock
└─ composer.lock
└─ composer.json
```

Joonis 3. Rakenduse põhiskeleton.

Enamik kirjutatavast koodist koondub *src* kausta (Joonis 3). Sinna alla kuulub *Kernel.php*, mis on Symfony tuum ning vastutab rakenduses kasutatavate pakettide seadistamise ja neile antavate konfiguratsioonide eest [53]. *Src* kaustast väljas pool paiknevad või luuakse nii *templates, tests, public* (dokumendi juurkataloog, failide avalikult kätte saadav koht), *bin, config, vendor* ja *var* (*cache* ja *log* failid) kaustad.

Kõik raamistikuvälised sõltuvused on määratud rakenduse juurkataloogis *composer.json* failis. Neid on võimalik *Composer*-i käsurealiidese kaudu sõltuvuste loetelusse lisada *composer require* käsu abil. Kõigi sõltuvustega seotud failid asetsevad *vendor* kaustas ning projekti üleslaadimisel Zone Bitbucket-isse neid ei kaasata. Igal arendajal tuleb käsureal installeerimine (ing. k. *composer install*) käivitada enne projekti kasutamist, et sõltuvused oleksid ligipääsetavad.

4.2.2 Serverrakenduse arhitektuuri ja struktuuri paika panek

Rakenduse arhitektuuri jaotamine kihilisteks abstraktsiooniosadeks võimaldab kirjutada kvaliteetset koodi ning vältida tarbetuid probleeme. Klienditeavituste rakenduse arhitektuuril ei ole kindlat arhitektuuri mustrit. Rakendus on kombineeritud *CQRS* (ing. k. *Command and Query Responsibility Segregation*) ja *EDA* (ing. k. *Event-Driven Architecture*) arhitektuuri mustrist ehk käskude ning päringute vastuste eraldamise ja sündmuste juhitud arhitektuurist. Rakenduses käsitletakse eraldiseisvalt REST API-t ning teavituste süsteemi, mis on omavahelises seoses vaid andmebaasi kaudu.

Rakenduses eraldatakse andmete salve lugemise ning värskendamise toimingud kahe erineva mudelina, mis viitab *CQRS*-i arhitektuurimustri aluspõhjale. Päringute põhjal loetakse infot ning saadetakse andmed tagasi kasutajale, käskude kaudu uuendatakse infot ning vastutatakse süsteemi järjepidevuse säilitamise eest [54].

Sündmuste juhitud arhitektuur ehk *EDA*, rõhutab rakenduse sündmuste kasutamist erinevate osadega suhtlemisel ja käivitamisel. Rakendus tuleks üles ehitada ümber keskse sündmuste siini (ing. k. *event bus*), mis vastutab sündmuste saatmise ja haldamise eest [55].

CQRS ja *EDA*-st tuleneva arhitektuuri saavutamiseks jagati loodava klienditeavituste rakenduse kood loogilistest kihtidesse (Joonis 4):

- **Esitluskiht** – vastutab kasutaja sisendi haldamise ja väljundi andmise eest kasutajale. Antud süsteemi puhul on selleks *cron*-töö, mis käivitab klienditeavituste süsteemi.

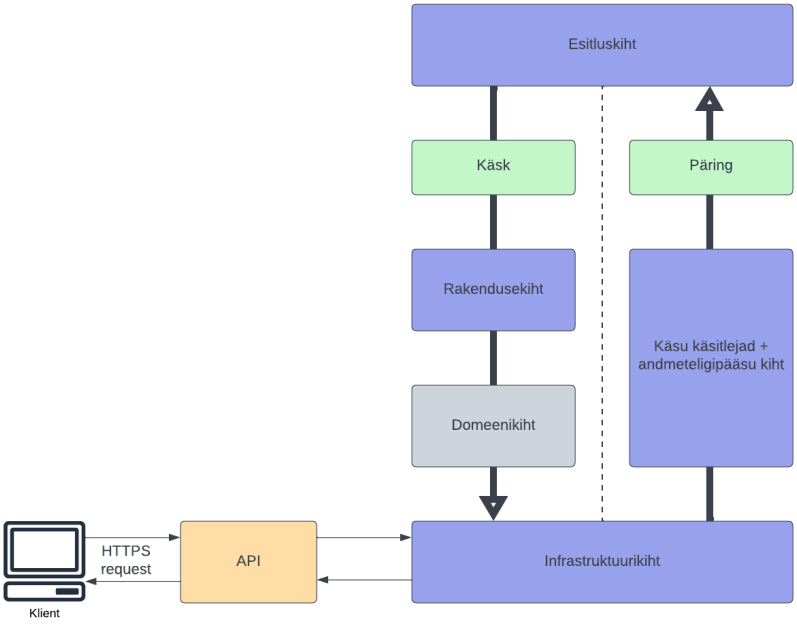
- **Rakenduse kiht** – vastutab rakenduse äriloogika rakendamise eest. Võtab vastu päringud esitluskihilt ning koordineerib vajalike toimingute täitmist domeenikihiga suheldes. Rakenduskihi klassid koguvad teavituste saatmiseks

vajalikud andmed. Kiht käsitleb endas sõnumi klasse, mis edastatakse sõnumiside transpordi komponendile ja sõnumikäitlejate klasse, mis tegelevad sõnumi kätte saamisega ja teavituste koostamisega.

- **Domeenikiht** – määrab rakenduse põhikontseptsioonid ja ärireeglid sisaldades domeenispetsiifilist loogikat ning käitumist, mida rakendus peab toetama. Antud süsteemis on domeenikihiks äriloogika, mis määrab tingimused, mille alusel tuleb teavitused välja saata.

- **Infrastruktuuri kiht** – toetab rakenduse tööd. Sisaldab komponente, mis suhtlevad väliste süsteemide või teenustega nagu andmebaas, meiliserver või sõnumisiin.

- **API** – vastutav kliendi päringu töötlemise ja vastuse tagasi saatmise eest. Ühtlasi tegeleb kliendi teenusega seotud andmete edastamisega andmebaasi *Notifies* tabelisse.



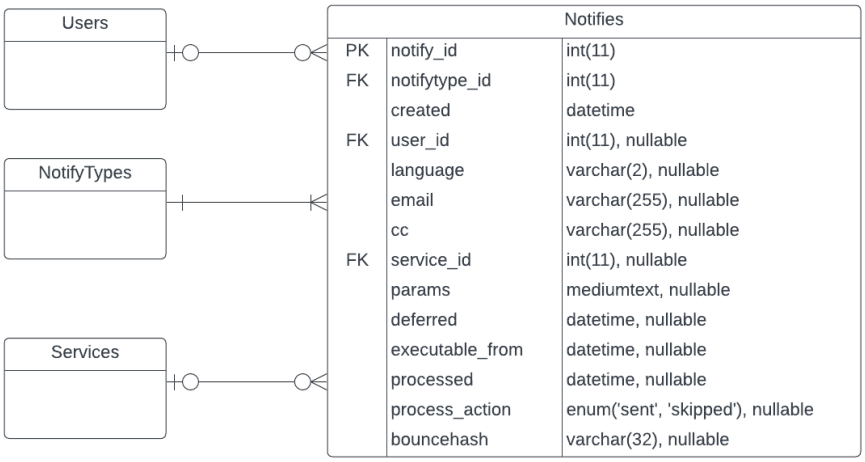
Joonis 4. Klienditeavituste rakenduse tööprotsess arhitektuuri vaatest lähtuvalt.

4.2.3 Andmebaasi kohandamine

Peatükis 3.3.3 selgus, et Zone Media kasutab MariaDB andmebaasimootorit. Kuna kasutatakse eelnevalt ettevõttes loodud andmebaasi, on võimalus taaskasutada olemasoleva äriloogika põhjal andmebaasi andmemudelit ning tänu sellele ka mitmeid

kasutuses olevaid tabeleid. Selle tulemusel saab mugavalt, vähese mõjutusega lisada lahendusest tulenevad muudatused.

Töö skoopi jäävas andmebaasis on kasutusel neli tabelit (Joonis 5). Põhirõhk koondub andmebaasi tabelile *Notifies*.



Joonis 5. Klienditeavitus rakenduse olemi-suhte lihtsustatud diagramm.

Andmebaasi muudatuste kohandamiseks uuele lahendusele tuli genereerida migratsioon olemasoleva *Notifies* tabelisse. Zone Medias kasutavad kõik rakendused andmebaasiga suhtlemisel *DAL* (ing. k. *Data Access Layer*) lahendusel põhinevat repositooriumi. Migratsioonide lisamine andmebaasi tabelitesse tugineb Laraveli raamistikul migratsiooni struktuuril, mis kasutab *up-down* meetodeid [56]. Migratsioonide jaoks loodi uus haru *DAL* repositooriumi ning uus migratsiooni fail uuendamaks *Notifies* tabelit vajalike väljadega. Migratsioonil on järgnevad meetodid:

- *up* – meetod võimaldab uuendada vastavat andmebaasi tabelit,
- *down* – meetod võimaldab muudatused tagasi võtta vastavast andmebaasi tabelist.

Zone Media teenuste ja loodava tagarakenduse koostöö sujumiseks oli vaja valitud andmebaasi *Notifies* tabelisse uusi välju:

- *params_json* – Süsteemi poolt loodud parameetrite väli JSON kujul, mis sisaldab endas kliendile saatetava teavituse infot,
- *flags* – Süsteemi poolt loodud väli, mis eristaks omavahel ühekordseid kirjeid.

Notifies tabelile lisati *nullable array* väärtus *„params_json,“* mille kaudu oli võimalik teenusega seotud teavituse info lisada JSON-kujul. Varasemalt hoiti andmeid massiivi indekseeritud kujul ning JSON parandas olulisemalt välja vaadet ja andmete konverteerimist. Kuniks vana süsteem ei ole veel täielikult üle antud uue lahenduse käsutusse, jääb kehtima ka vana tabeli info pesa, milleks on „*params*“.

Notifies tabel sai ühtlasi täienduse ka *nullable string* väärtuse „*flags*,“ mis võimaldab edaspidi ühekordsete kirjade puhul neid omavahel eristada. Kuna ühekordsed kirjad genereeritakse kasutajatoe poolt, ei olnud varasemalt võimalust teavitustele määrata prioriteete, mis vajavad esmajärgus käsitlemist.

Muudatuste kajastumiseks vastavas andmebaasi tabelis käivitati käsk

```
php sql/migrator do:up all
```

Selle jooksul lisati uuendused andmebaasi tabelisse ning ühtlasi kontrolliti ka kõiki teisi migratsiooni faile. Nii välistatakse konflikte, et mõni tabel on andmebaasis uuendamata jäänud või muutnud oma struktuuri. Korrektne migratsiooni skeemi sisu on toodud välja Lisa 2 all.

4.2.4 Serverrakenduse realiseerimine

Peatükkides 3.4 ja 3.4.1 välja toodud kolmanda osapoole lahendusi ning analüüsi tulemuste põhjal selgus, et klienditeavituste rakendus on vaja iseseisvalt arendada. Olemasolevad lahendused ei kata kõiki nõudeid ning vajadusel saab neid süsteemi juurde lisada täiendavate edastusmeetoditena.

Serverrakendus arendati Symfony raamistikul ning nõutest lähtuvalt kasutades selleks *Symfony Mailer* sõltuvust. Asünkroonselt teadete edastamiseks kaasati rakendusse veel *Symfony Messenger*. Sõltuvused lisati rakendusele *Composer*-it kasutades. Esialgne rakendus pühendub vaid e-kirjade teavituste saatmisele, kuid arvestati täiendavate edastusmeetodite lisamisega tulevikus.

Järgnevalt kirjeldatakse kolme põhikomponenti asünkroonsete teadete edastamiseks rakenduses:

Symfony Mailer

Komponent on vastutav teavituste loomise ja saatmise eest. Kasutab *Transport* komponenti teavituste edastamisel ning võimaldab lihtsat arusaadavat liidest (*MailerInterface*) arendamisel [57]. Liidesel on ainult üks saatmismeetod (ing. k. *send*), mida saab kasutada teavituste sünkroonselt kui ka asünkroonselt saatmisel [58].

Transport

Vastutav komponent teavituste edastamisel läbi kindla transpordiprotokolli. Rakenduse koodi muutmata on võimalik läbi abstraktsioonikihi erinevaid transpordivahendeid hallata [59]. Vaikimisi pakub *Symfony Mailer* teavituste haldamiseks läbi transpordi sisseehitatud *DSN* (ing. k. *Data Source Name*) protokolle *SMTP*, sendmail või native Võimalus on kasutada ka kolmanda osapoole teenuseid, mille sõltuvused tuleb läbi *Composeri* rakendusele lisada [59].

Symfony Messenger

Komponent pakub sõnumisiini, mis aitab rakendusel teistesse rakendustesse või teadete järjekorra (ing. k. *queue*) kaudu teateid edastada ning vastu võtta, kasutades *CQRS* disainimustrit [60]. Järjekorra süsteemid, näiteks *AMQP* (ing. k. *Advanced Message Queuing Protocol*) ehk süsteem, mis juurutab *AMQP* spetsifikatsiooni näiteks *RabbitMQ*, *Redis*, *Doctrine* või mälusisene (ing. k *in-memory*), salvestavad ja edastavad sõnumeid. Marsruudi põhjal määrtakse, millist süsteemi kasutada iga sõnumiklassi või liidese jaoks [61].

Teadete edastamiseks seadistati *mailer.yaml* ja *messenger.yaml* konfiguratsioonid, mis sõltuvuste lisamisel tekkisid *config/packages* kausta (Lisa 3). *Composer* lisas automaatselt *.env* faili vajalikud muutujad *MAILER_DSN* ja *MESSENGER_TRANSPORT_DSN*. Antud rakenduse arendusprotsessi raames seadistati *MAILER_DSN* vaikimisi kasutama *SMTP* serverit, mis põhineb näitel:

`smtp://user:pass@smtp.example.com:port` [59].

Selline URL-kirje ütleb *Mailer*-le, millist serverit või pilveteenust edastamiseks kaasata. *MESSENGER_TRANSPORT_DSN* seadistati vaikimisi *Doctrine* kasutama [62], kuna rakenduses on juba *Doctrine* toetatud. Rakenduse asünkroonne lähenemine viitab sellele,

et serialiseeritud teavete objektid lisatakse järjekorda ning töötlemisel võetakse need ükshaaval järjekorrast ja edastatakse kliendile.

4.2.5 Teavitusmallide ületoomine

Nõuetest lähtuvalt on arendatava rakenduse puhul vajalik teavitusmallid üle viia *Markdown* ja *Twig* (.md.twig) laiendite peale. Peatükis 3.1 analüüsides pärandrakendust selgus, et suurem osa mallidest on kirjutatud kasutades lihtteksti või *Markdown* struktuuri. Valdav osa pärandsüsteemi mallidest sisaldavad endas ka PHP koodi. Hästi arendatud süsteemis on oluline eraldada äriloogika esitluskihist, mida võetakse mallide üle toomisel arvesse. Ainsatena, on mahuteavitused juba varasemalt uuenduse läbinud ning vajavad vaid faililaiendite muutmist.

Teavitusmallide refaktoriseerimisel oli vajalik rakendada järgnevad sammud:

- luua teavitustüüpide ning pealkirjade jaoks vastavad failid, kasutades *.md.twig* laiendeid;
- eraldada nii mallides kui ka pealkirjades olev PHP kood ning asendada *Twig* struktuuri omaste muutujatega;
- ühildada teavitustüübi klassi koodi poolne loogika vastavate malli ja pealkirja muutujatega;
- malle ümbritsev HTML-*layout* sisu, kuhu teavitus tekst paigutatakse, viia PHP muutujalt üle *Twig* muutujale.

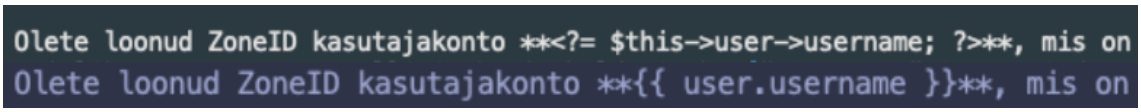
Arvestada tuli ka olukorraga, et mallid on *Markdown* kujul ning teavituste saatmisel tuleb need HTML kujule ümber konverteerida. Pärandrakenduses tegeleb sellega eraldi meetod, kuid antud lahenduse raames pakub *Twig* ise funktsionaalsust eraldi meetodeid kirjutamata. Selleks on võimalus kasutada

```
{% apply markdown_to_html %}...{% endapply %}
```

funktsionaalsust malli enda sees.

Teise arvestatava kohana olid relatsioonidel põhinevad muutujad, mida enam ei olnud võimalik kasutada ning väärtustatud massiivid. Muutujad teisendati ümber mõnel juhul

mitmemõõtmeliseks massiiviks äriloogika poolel ning on ligipääsetavad mallidel punkti kujul (Joonis 6).



Joonis 6. Relatsiooniliste malli muutujate teisendamine Twig kujule.

Esines teavitustüübi malle, mis nõudsid hulgaliselt rohkem mõtlemist, kuidas varasem PHP äriloogika üle tuua. Pärandrakenduses kasutati mallides eraldi massiive, kuhu sisse kirjutatu oli püsiprogrammeeritud.

Järgnevalt selgitatakse lahti, kuidas on teavitused üles ehitatud ning kuvatud klientidele. Teavitusmallid on bränditi mõneti erinevad, kuid üldine struktuur on kõigil sama (vt Joonis 7):

1. Malle ümbritseb raam ehk *Layout*, mis on kirjutatud HTML-is ning sisaldab endas muutujat `{{ content }}`. Igale keelele on omakorda loodud eraldi HTML *Layout* fail.

2. Raamis paiknev muutuja `{{ content }}` sisaldab endas teavitustüübile omast informatsiooni, mida kliendile edastatakse. Muutujad teisendatakse ümber respektiivselt teavitus tüübi klassis. Sisu tuleneb keelest ning teavitustüübist.

3. Signatuur ehk *Macro* sisaldab Zone kontaktinfot kliendile, kuhu küsimuste korral pöörduda. *Macro*-ks nimetatakse seda seetõttu, et tegemist on kasuliku malli fragmendiga, mida saab taaskasutada ning erineb vaid keeleti.

Joonis 7. Teavitus malli komponentide paiknemine kliendivaatest.

4.2.6 Teavituste genereerimine

Rakenduse poolt genereeritud teavitused koostatakse teavitustüübi, kasutaja adressaatide ning *Notifies* tabeli *params_json* välja alusel. Teavitusmallid ja -pealkirjad tagastatakse HTML-sõne kujul *Twig* keskkonna renderdamismeetodi (ing. k. *Twig Environment render*) abil. Meetodile antakse kaasa muutujad teavitus klassist, mis täidavad mallide ning pealkirjade kohatäitjaid. Teavitusklass tagastab massiivikujul info, mis edastatakse sõnumiklassile. Ühtlasi on tegemist vähima hulga teabega, mida läheb vaja teavituste välja saatmisel (Joonis 8).

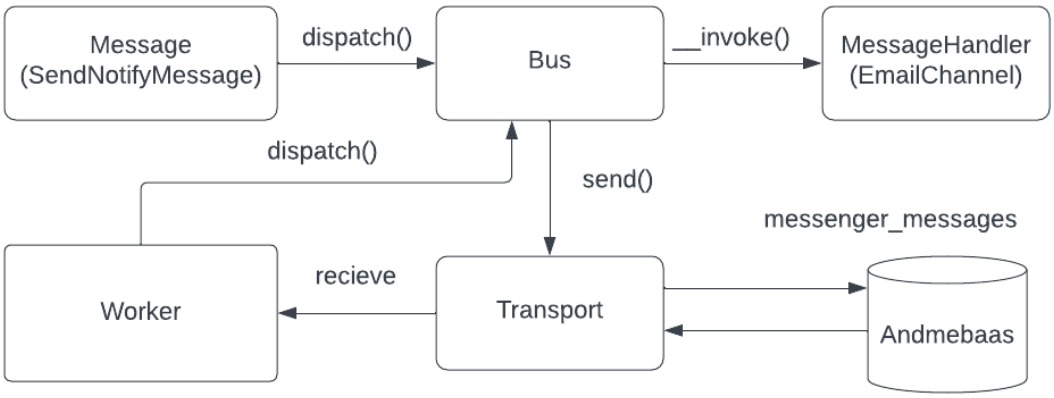
```php
public function send(): array
{
    $result = [];

    $recipients = $this->getRecipients();
    $body = parent::getHtml('                ', $this->setTemplateVariables());
    $subject = parent::getSubject('                ', $this->setTemplateVariables());

    if (!empty($this->_attachments)) {
        $output[] = [
            'attachments' => $this->_attachments,
        ];
    }

    $output[] = [
        'template' => $body,
        'subject' => $subject,
        'recipients' => $recipients,
    ];

    return $output;
}
```

Joonis 8. Teavitusklassi meetod, mille väljund edastatakse sõnumile.

Teavituse saatmisteekond kliendini ühildab endas mitut osapoolt (Joonis 9). Sõnum edastatakse koos teavitus sisuga Symfony sõnumisiinide liidesele (ing. k. *MessageBusInterface*) lähetamismeetodi (ing. k. *dispatch*) abil

```
$this->bus->dispatch(new SendNotifyMessage($notifyClass->send()));.
```

Sõnumiks on PHP objekt, mis toetab serialiseerimist. Edasi liigub sõnum läbi sõnumisiini (ing. k. *Message Bus*) sõnumiside transpordile (ing. k. *Messenger Transport*) ning seejärel andmebaasi *messenger_messages* tabelisse. Paralleelselt tagaplaanil käib töötaja (ing. k. *Worker*), mis kuulab sõnumitootja transpordi järjekorda uute sõnumi tekkel. Sõnumi leidmisel tagastatakse see sõnumisiini, mis oma korda edastab sõnumi edasi sõnumikäitleja (ing. k. *MessageHandler*) klassile. Käitleja klass on defineeritud atribuudiga *#[AsMessageHandler]* ning täiendatud kohustusliku meetodiga *__invoke*. Meetod võtab sisse sõnumi klassi, mis määrab, milliseid sõnumeid käitleja töötlema peab. Symfony automaatika (ing. k. *autowiring*) teab tüübi vihje kujul, kuidas teenused on omavahel seotud [63].



Joonis 9. Teavituse saatmise teekond Symfony-s.

Sõnumikäitleja klassis eraldatakse teavituste saatmisega tegelev meetod. Meetod võtab sisse sõnumi sisu deserialiseeritud massiivi kujul ning paneb selle põhjal kokku teavituse. Sõnumikäitleja klassi lõpptulemusena edastatakse teavitus kliendile (Joonis 10).

Teavituse edastamisel kliendile kasutati *Symfony Mailer* sõltuvus, mis koosneb kahest komponendist: *Mailer* ja *Mime*. *Mime* eesmärk on konfigureerida ja luua teavitus ning *Mailer* tegeleb teavituse saatmisega [64]. Uue teavituse genereerimisel on võimalik kasutada Symfony *Mime* komponente *Email* või *TemplatedEmail* ehk E-kiri või

Mallipõhine e-kiri. Arendusprotsessi jooksul võrreldi kahe komponendi funktsionaalsust ning testitavust, kuid teavitused otsustati genereerida e-kirja baasil. Põhjus seisnes selles, et mallipõhine e-kiri nõuab sisendiks HTML kujul malli asukohta, mida ei ole võimalik ette anda. Mallid on genereeritud *Markdown* kujul ning ei sisalda HTML-i. See eeldab omakorda lõpptulemuse jaoks eraldi meetodi loomist mallide genereerimisel ning täiendavalt tuleb malli muutujad eraldi kaasa anda. Teiseks põhjuseks oli pealkirja lisamine. Igal mallil on teavitus tüübile vastav pealkiri, mis võib sisaldada samuti muutujaid. Mallipõhisele e-kirjale saab ette anda vaid sõne kujul pealkirja. Sellisel juhul tuleks luua veel üks meetod, mis saab kätte teisendatud kujul teavituse pealkirja ning annab selle teavituse koostamisel ette. Ei ole välistatud, et olukord muutub tulevikus.

```php
/**
 * Send actual emails out
 *
 * @param array $notifyData
 */
public function sendNotify(array $notifyData): void
{
    $email = (new Email())
        ->from('                    ')
        ->to(new Address($notifyData['recipients']['email']))
        ->subject($notifyData['subject'])
        ->html($notifyData['template']);

    if (in_array('attachments', $notifyData)) {
        $email->addPart(new DataPart(new File($notifyData['attachments'])));
    }

    try {
        $this->mailer->send($email);
        echo 'New notify has been sent out';
    } catch (TransportExceptionInterface $exc) {
        echo 'Couldn\'t send an notify to address. Exception ';
        echo $exc->getMessage();
    }
}
```

Joonis 10. Sõnumikäitleja klassi meetod teavituste edastamiseks kliendile.

4.2.7 REST rakendusliides

Serverrakendusest klientidele teavituste saatmiseks oli vajalik REST-rakendusliidest, mis edastaks vajaliku info *Notifies* tabelisse andmebaasis. Rakendusliidese kiht vastutab teenuste päringute vastuvõtmise ja vastuste tagasisaatmise eest kasutajaliidesesse.

REST (*Respresentational State Transfer*) rakendusliides on tarkvaraarhitektuuriline lähenemisviis veebiteenuste loomisel, mis kasutab HTTP protokolli ressurssidega suhtlemiseks. Sarnaselt teiste arhitektuuristiilidega on REST-il omad põhimõtted ja tavad, mis peavad olema täidetud, et rakendus vastaks RESTful rakenduse standarditele [65].

REST API kontrolleri klassid Symfony rakenduses laiendavad abstraktset kontrollerklassi (ing. k. *AbstractController*) ning iga meetod kontrolleris defineeritakse atribuudiga

```
#[Route('/<kontrolleri_nimetus>, name: 'app_<kontrolleri_nimetus>, methods: ['GET/POST/PUT/DELETE'])],
```

kus *route* ehk marsruut kaarditatakse, milliselt URL-lt päringut on võimalik teostada, *name* ehk nimetus on vastava marsruudi nimi, mida saab kasutada URL-i loomisel teistes meetodites, kui on vajalik näiteks antud kontrollerile tagasi pöörduda ning *methods* ehk HTTP meetodid, mis määravad ära päringu tüübi.

Andmete töötlemiseks saadetakse kindlale lõpp-punktile HTTP-päring koos vastava HTTP-verbiga. Antud rakenduses kasutatakse HTTP-verbe GET üksikressursikogumi lugemiseks ja POST andmete sisestamiseks *Notifies* tabelisse. Andmed edastatakse päringuobjektina (ing. k. *Request*) ning tagastatakse kliendile vastuseobjekt (ing. k. *Response*). Nii edastus kui ka tagastus vastuse vorming on JSON-objekti kujul.

Antud lõputöö raames on REST API lahendus minimaalne täitmaks vajalikud käsud, et kliendi andmed teenuse kohta edastada andmebaasile.

4.2.8 Rakenduse turvalisus ja ligipääsetavus

Rakenduse turvalisus ja ligipääsetavus on kriitilised aspektid veebiteenuste arendamisel. Turvameetmeid ei ole mõtet ise looma hakata, kuna turvalisusega seotud probleemidele võib olla keeruline ise lahendust leida ning suure tõenäosusega võivad need sisaldada turvaauke.

Andmete juurdepääsu turvalisuse tagamiseks on käesolevas rakenduses kasutusel kasutaja autentimiseks ZoneID-d ja genereeritud API parooli. Klient, kes soovib päringuid teostada läbi API tarkvara või kasutades selleks käsurea *curl*-i, peab esmalt looma endale *my.zone.eu*-s API võtme. Autentimine toimub HTTP põhisel autentimismeetodil (ing. k. *HTTP Basic Auth*).

4.3 Pidev integratsioon ja evitamine

Pideva integratsiooni ja evitamise protsessid uue rakenduse puhul aitavad arendajatel varakult tuvastada vigu, parandada koodi kvaliteeti ning kiirendada rakenduse

arendusprotsessi. Eesmärk on testida uute muudatuste kooskõla äriloogikaga, mis annab võimaluse automaatselt ja kiiremini uusi versioone välja lasta. Üha enam on arenduses kasutusele võetud CI/CD (ing. k. *Continuous Integration/Continuous Delivery*) metoodikat, pidev integratsioon ja pidev edastamine, mis keskendub pidevale tarkvaralahenduse tarnimisele kliendile, tuues automatiseerimise tarkvaralahenduse arendamise etappidesse [66].

Rakenduse evitamiseks uute muudatuste korral ehitatakse rakendusest valmis uus versioon peaharusse. Sinna kogutakse kokku uuendused, mis testkeskkonnast peavad jõudma lõpliku faasina klientideni. Arendusmeeskond viib läbi manuaalse testimise põhifunktsionaalsusega, käivitatakse uuesti kõik testid ning vaadatakse üle kõik koodi muudatused. Pideva integratsiooni puhul on loodud eraldi skriptid, mis käivitatakse iga kord kui soovitakse uut versiooni turule väljastada. Antud projekti raames kasutatakse *Jenkins*-i integratsiooni tööriista, mis võimaldab automatiseerida evitusprotsessi ja testimist. Kuna inimsilm ei pruugi näha alati vigu, mis võivad arendus protsesse mõjutada, kasutatakse selleks automatiseeritud lahendusi. Skriptid on omavahelises seoses *Jenkins*-iga, mis kontrollivad automaattestide põhjal koodi uuenduste mõjuala. Automaattestide käivitamiseks *Jenkins*-is luuakse keskkonnas uus *pipeline*, mis seostatakse rakenduse automaattestide Atlassian Jira Bitbucket repositooriumiga. Automaattestide repositooriumis asub juurkaustas fail nimega *Jenkinsfile*, mille sees määratletakse testide käivitamise masin.

4.4 Rakenduse optimeerimine

Iga uue süsteemi puhul on vajalik mõelda selle optimeerimisele, et tagada rakenduse skaleeritavus, töökindlus ja tõhusus. Käesolev peatükk keskendub rakenduse optimeerimise olulisusele ning vajadusele arendusprotsessis.

4.4.1 *Cron*-tööd

Vähendamaks arendaja tööd ning klienditeavituste süsteemi automatiseerimist luuakse skript, mida käsitletakse *cron*-tööna.

Antud skript kujutab endast tööd või protsessi, mis teatud aja möödudes serveri poolt automaatselt käivitatakse. *Cron*-tööd vaadeldakse kui üksikobjekti *crontab* failist Unix süsteemis. Sellised skriptid töötavad perioodiliselt antud ajakava alusel, mille ajavahemik

määratakse skripti eesmärgist lähtuvalt [67]. Skript seatakse üles Zone Media kuuluvasse serverisse, kus paiknevad kõik teised *cron*-tööd.

Cron-töö eelis klienditeavituste rakenduses on teavituste automaatne välja saatmine ning andmebaasi *Notifies* tabeli mahu vähendamine. Arendaja ei pea ise silmas pidama, et teavitused on vaja laiali saata, vaid süsteem teeb seda tema eest. Nii minimeeritakse olukord, kus kliendid saavad teavitused viivitusega või andmebaas koormatakse teavitustega üle. Klientidele saadetavate teavituste *cron*-töö seadistatakse käima iga kahe minuti tagant.

4.5 Rakenduse testitavuse tagamine

Käesoleva peatüki kontekstis mõeldakse testimise all klienditeavituste rakenduse manuaalset testimist, ehk autori enda poolt, ja automaatsete testide koostamist. Kirjeldatakse testide loomisprotsessi tagarakenduses ning koodikattuvust. Automaattestide koostamisel kasutatakse serverrakenduse puhul *PHPUnit* testimisraamistikku.

4.5.1 Tagarakenduse testimine

Rakenduse testimine on oluline osa arendusest veendumaks rakenduse toimivuses. Selle tulemusel tagatakse, et ei esineks vigu, mis võiksid hilisemas järgus teenuse tööprotsesse mõjutada.

Arendusprotsessi vältel kasutati *PHPStan* staatilise analüüsi tööriista tuge ning teostati manuaalset testimist tagamaks rakenduse toimivuse ja ühtlasi õigetel alustel teavituste genereerimise. *PHPStan* võimaldab leida koodibaasist vigu ilma koodi käivitamata enne kui rakendus tootmisesse suunatakse [68].

Manuaalse testimise käigus kasutati API päringute testimisel Postman-tarkvara ning kirjutati käsurea skript, mille alusel päriti andmebaasi tabelist *Notifies* välja saatmata teavitused. Rakendus genereeris andmebaasist tuleva info põhjal (sisendid, bränd, keel, meiliaadress(id)) õige teavituse subjekti ehk pealkirja, sisu, ning kellele teavitus edastada. Esialgsed teavitused suunati järjekorda ning *Symfony Mailer* käsu

```
php bin/console messenger:consume async -vv
```

käivitamisel käsurealt saadeti need autori e-posti aadressile.

Lahenduse valmimise lähenemisel hakati koostama äriloogilisi automaatteste, vähendamaks muudatuste käigus tekkivaid vigu ning valideerimaks koodi. Ühiktestidega kaeti rakenduse tuuma moodustavad funktsioonid nagu teavituste kättesaamine, õigete teavituste ja mallide genereerimine, sisendite valideerimine ning erindite püüdmine vigade esinemisel.

Testid kirjutatakse vastavalt Zone Media testimisdokumentatsioonile ning testide nimed kirjutatakse kaamelkirjas (ing. k. *CamelCase*). Nimedest on võimalik välja lugeda nõutud funktsionaalsus, mida testitakse. Näiteks *testEmailChannelWillSendOutASuccessfulNotify* näitab, et e-maili aadressi teel saadetav teavitus edastatakse õigetel alustel ning jõuab kliendini. Testimist teostatakse *Mock*-objektide kaudu, mis väldivad suhtlust tegeliku andmebaasiga ning jäljendavad kõigest klasside meetodite käitumist. Testimise käigus midagi andmebaasi lisada ei tohiks. Eeldavate tulemuste ning testist tagasi saadava info põhjal luuakse *Assert* või *Expect* laused, mis peaksid tõestama koodi ootuspärast käitumist.

Järgneval joonisel on toodud näide eelmise lõigus kirjutatud testi ülesehituse põhjal, mis demonstreerib, kuidas töös testimist teostati (joonis 11).

```php
public function testEmailChannelWillSendOutASuccessfulNotify(): void
{
    $htmlTemplate = '<p>Täname, et valisite Zone oma teenusepakkujaks!
                     Olete loonud ZoneID kasutajakonto **testing**,
                     mis on vajalik Zone teenuste tellimiseks ja haldamiseks
                     [“Minu Zone” iseteeninduses](https://www.my.zone.eu).</p>';

    $subjectTemplate = '[Zone] ZoneID kasutaja testing loodud';

    $mailer = $this->createMock(MailerInterface::class);
    $mailer->expects($this->once())
        ->method('send')
        ->with($this->isInstanceOf(Email::class));

    $twig = $this->createMock(Environment::class);

    $channel = new EmailChannel($mailer, $twig);

    $notifyData = [
        'template' => $htmlTemplate,
        'subject' => $subjectTemplate,
        'recipients' => [
            'email' => 'test.user@example.com',
        ],
    ];

    $message = new SendNotifyMessage($notifyData);

    $channel->sendNotify($notifyData);

    $this->expectOutputString('New notify has been sent out');
}
```

Joonis 11. Ühiktesti näidis serverrakenduses.

4.5.2 Koodikattuvuse testimise

Koodikattuvus on pideva edastamise ehk CD (ing. k. *Continuous Delivery*) üks testimise strateegiatest. Võimaldab protsendiliselt määrata, kui suures mahus käivitatakse koodi. Pidev edastamine kasutab programmi või lähtekoodi kvaliteedi määramisel automatiseeritud testide komplekti. Ehkki automaatsed testid on kasulikud koodi ehituse kvaliteedi hindamisel, ei võimalda see kindlaks määrata kui suur osa tegelikult lähtekoodist arenduses käivitatakse [69].

Antud rakenduse raames kasutatakse koodikattuvuse testimisel *Xdebug*-i. *Xdebug* on PHP *de-facto* silumislaiendus, mis pakub nii silumis- kui ka koodikattuvuse võimalusi. Laiend võimaldab teavet saada kooditee kattuvusest kui ka „surnud koodi“ osast, mida mitte kunagi ei käivitata [70].

Testimata kood võib potentsiaalselt viidata vigadele ning omakorda mõjutada lõppkasutaja kasutuskogemust. Pideva integratsiooni korral ei soovita, et ükski osa lähtekoodist jääks testimata. Ideaalis võiks rakenduse koodikattuvus olla ligilähedane 100%-le, kuid projekti ja funktsionaalse poole kasvades on märksa keerulisem katta koodi täielikult testidega. Suured projektid muudavad lähtekoodi halduse tülikaks ning 100%-lise katvuse eesmärgi ebareaalseks. Reaalsuses on umbes 80% ulatuses testidega kaetud kood suurepärane tulemus [69].

4.6 Arendusprotsessi kokkuvõte

Arendusprotsessi jooksul tutvuti uue klienditeavituste rakenduse loomisega Symfony raamistikul. Selgitati Symfony rakenduse ülesseadmist ning sellele arendatava *Symfony Mailer*- ja *Messenger*-i kaasamist. Rakenduse arenduse jooksul tuli kohandada andmebaasi vastavalt uue lahenduse nõuetele ning viia vanad pärandrakenduse teavitusmallid osaliselt üle teistele laienditele. Teavitus mallide teisendamisel eraldati PHP koodi äriloogika HTML ja *Markdown* sisust ning teavituste muutujad viidi täielikult üle puhtale *Twig* kujule.

Rakenduse puhul on äärmiselt oluline selle pidev integratsioon ning optimeerimine.

Rakenduse testimisprotsess toimus esialgu manuaalse testimise läbi serverrakenduses kui ka rakendusliideses. Rakenduse valmimise hetkel asuti kirjutama ühikteste, et katta

süsteemi tuuma moodustavad funktsioonid. Testimise puhul kasutati *PHPStan* sõltuvust koodi õigsuse kontrollimiseks ja *PHPUnit* sõltuvust raamistikku testide koostamisel.

Tehnoloogilise evolutsioon käigus on tekkinud juurde erinevaid kanaleid teavete edastamiseks. Lahenduse arendamisel võeti arvesse rakenduse disaini jätkusuutlikust, mis võimaldaks tulevikus juurde evitada täiendavaid edastusmeetodeid.

5 Hinnang loodud mikroteenusele

Käesolevas peatükis antakse hinnang rakenduse kasutatavusele arendusprotsessi järgselt. Ühtlasi tuuakse välja ka võimalused klienditeavituste rakenduse edasiarenduseks.

Lõputöö raames valmis Zone Mediale minimaalselt töötava teenuse rakendus, klienditeavituste haldamiseks. Rakendust võib hinnata oodatud ja saavutatud funktsionaalsuse alusel.

Arendusprotsessi jooksul täiendati teadmisi nii Symfony raamistiku käsitlusest, rakenduse haldamisest kui ka PHP versiooni kaheksa uuendustest. Neid teadmisi plaanib töö autor edaspidi ka oma igapäevatöös rakendada.

5.1 Saavutatud kasutatavus

Lõputöös arendatud rakendust saab hinnata realiseeritud funktsionaalsuse abil. Nõuded, mis on välja toodud kolmandas peatükis, on enamuses täidetud lõputöö kirjutamise protsessi ja rakenduse arendamise jooksul. Projektil puudub lõplik ametlik realiseerimise kuupäev ning klienditeavituste rakenduse juurutamine jätkub ka peale lõputöö valmimist. Käesoleva lõputöö vastab MVP rakenduse lahendusele ning valminud vastavalt planeeritule.

Arendatud klienditeavituste rakendus on loodud Zone Media nõuete kohaselt ning mõeldud kasutamiseks vaid Zone Media siseselt. Käesolev rakendus võimaldab paralleelselt töös hoida ka pärandsüsteemi, kuniks uus lahendus täielikult valmib. Tagarakendus sai võrreldes pärandsüsteemiga uue ja kaasaegsema süsteemi ning võimaldab kõik teavitussüsteemide komponendid pärandrakendusest üle tuua edaspidi ühtse lahendusena. Kliendid oma silmaga selles osas muudatusi ei näe, kuid antud rakendus võimaldab Zone Media arendusosakonnal ning ka klienditoel edaspidi ajakohasemat süsteemi kasutada. Autori hinnangul vastab loodud lahendus tänapäeva tarkvaraarenduse parimatele praktikatele. Rakenduse paindlikkus on nii ärilistele kui ka tehnoloogilistele muudatustele.

5.2 Võimalikud edasiarendused tulevikus

Neljandas peatükis mõistliku lõputöö skoobi töömahu säilitamiseks loodi MVP. Vaatamata sellele, et loodud serverrakendusel on küllaltki kitsas kasutusala, on selle täiustamiseks ja parandamiseks sellegipoolest võimekus edasiarendusteks.

Valminud lahendus Zone Media näitel ei ole täiuslik ning arenguruumi veel on. Seda tõestavad ka nõuded, mis määrati uue lahenduse osas ega käsitle kõiki teavitus tüüpe. Arendusprotsessi jooksul loodud minimaalselt töötav teenus saatmaks välja kirju mingisuguses valikus ning nõutud mallide põhjal. Tulevikus oleks vaja kõik teavitusklassid pärandrakendusest üle viia uuele lahendusele ning teisendada vanad teavituste mallid. Eeltöö selle jaoks on tehtud ning edaspidine arendus kujutab endast vaid ümber töötlemist uuenenud lahenduse alusel. Antud rakenduse puhul võiks mõelda ka sellele, et rakendus muutuks vaid sündmuste põhiseks ning teateid saaks edastada koheselt ka API kaudu.

Rakenduse nõuete poolest mõeldi arenduses võimalusele lisada ka täiendavaid edastusmeetodeid. Need jäid lõputöö skoobist välja, kuna esialgne süsteem nõudis vaid paralleelset töötamist pärandrakendusega. Zone Media ei ole veel kindlalt paika pannud, milliseid kanaleid hakatakse tulevikus rakendama süsteemi raames, kuid täiendavate edastuskanalite lisamine kujutab endast vaid uute klasside loomist ning süsteemi juurutamist.

Rakendusliidese poolt lihtsustati antud lõputöö raames. Zone Media ei ole kindlaks määranud, millised konkreetsed lõpp-punktid on planeeritud API poolele ning millised neist hakkavad kasutama avalikku ja mitte-avalikku dokumentatsiooni. Eeltöö on selle jaoks samuti teostatud.

Klienditeavituste rakendusega on suurel määral seotud ka klienditugi. Nende kasutusalas on teavitusredaktor, mis võimaldab neil olemasolevaid teavitusmalle korrigeerida vastavalt nõudmistele ning luua ka täiesti uusi. Lõputöö skoopi ei kuulunud teavitusredaktori kaasamine arendusprotsessi, kuid klienditeavituste rakenduse valmimise hetkel on vajalik kõik uuenenud teavitusmallid üle viia ka klienditoe rakendusse. Autor on näinud omasilmaga teavitusredaktori töökäiku, kuid kokkupuude on olnud minimaalne. Teavitusredaktori üles seadmine Zone Media arenduskeskkonnas on aeganõudev.

6 Kokkuvõte

Käesoleva bakalaureusetöö käsitleb klienditeavituste rakenduse arendust ettevõttele Zone Media OÜ, mida oleks edaspidi lihtne ajakohasena hoida ja ühtlaselt täiendada.

Eesmärgi saavutamiseks uuris autor Zone Media klienditeavituste pärandsüsteemi ning selle puuduseid. Pärandsüsteemi analüüsist lähtuvalt peeti tarkvaraarenduse osakonna juhiga intervjuu, mille käigus formuleeriti funktsionaalsed ja mittefunktsionaalsed nõuded loodavale rakendusele. Nõuete põhjal otsiti võimalike lahendusi ning analüüsiti võimalusi. Oluline oli tuua üle varasem funktsionaalsus, mida kasutati pärandrakenduses.

Analüüsi käigus uuriti kolme turul olevat teenust, mis pakuvad e-kirjadel põhinevat teavitussüsteemi. Antud lahendused ei lahendanud püsitatud probleemi. Analüüsist lähtuvalt on turul olemasolevate teavitussüsteemidel omad eelised ja puudused, mis ei kata täielikult uue arendatava klienditeavituste rakenduse nõudeid. Ühtlasi on Zone Medial endal nii suur kompetents ja võimekus teavituste välja saatmisel, et väliste teenuste kasutamine oleks piiranguks. Sellest tulenevalt otsustas autor rakenduse realiseerimise algusest lõpuni ise valmis ehitada.

Töö käigus analüüsiti erinevaid tehnoloogiaid, millega oleks kõige efektiivsemalt võimalik rakenduse MVP luua. Lähtuvalt ettevõtte nõuetest valiti tehnoloogiateks tagarakenduses PHP programmeeriskeel ja Symfony raamistik, andmebaasina oli kasutusel MariaDB. Nõudeid arvestades loodi vastav rakendus valitud vahenditega.

Klienditeavituste rakendus pole lõputöö esitamise hetkeks ettevõttes täielikult kasutatav. See nõuab lõplikku arendust ka teistele teavitustüüpidele ja meetoditele, mis käesoleva lõputöö raames jäid töö skoobist välja. Valminud serverrakendus vastas enamikele kolmandas peatükis seatud nõuetele ning nii Zone Media tarkvaraarendus meeskond kui ka klienditugi on loodud lahendusega üldiselt rahul. Edaspidi saab tagasiside põhjal loodud lahendust veelgi paremaks muuta ja vaja minevaid funktsionaalsused juurde lisada. Valminud lahenduse prototüüp aitab tulevikus parandada nii töökorralduse efektiivsust kui ka klientide kasutuskogemust.

Kasutatud kirjandus

[1] Z. M. OÜ, „Zone EE ettevõttest,“ [Võrgumaterjal]. Available: https://www.zone.ee/et/ettevottest/. [Kasutatud 3 märts 2023].

[2] Z. M. OÜ, „Zone.ee: Esimene volitatud .EU domeenide registreerija Eestis,“ [Võrgumaterjal]. Available: https://www.zone.ee/et/zone-ee-on-esimene-volitatud-eu-domeenide-registreerija-eestis/. [Kasutatud 3 märts 2023].

[3] Z. M. OÜ, „Zone sai 11 aastat vanaks,“ [Võrgumaterjal]. Available: https://www.zone.ee/et/zone-sai-11-aastat-vanaks/. [Kasutatud 3 märts 2023].

[4] Z. M. OÜ, „Miks valida Zone?,“ [Võrgumaterjal]. Available: https://www.zone.ee/et/miks-zone/. [Kasutatud 3 märts 2023].

[5] Z. M. OÜ, „Zone.ee liitus ITL-iga,“ [Võrgumaterjal]. Available: https://www.zone.ee/et/zone-ee-liitus-itl-iga/. [Kasutatud 3 märts 2023].

[6] „S/MIME,“ Zone Media OÜ, 28 01 2023. [Võrgumaterjal]. Available: https://help.zone.eu/kb/smime/. [Kasutatud 20 märts 2023].

[7] „Package Information: Mail_Mime,“ The PHP Group, [Võrgumaterjal]. Available: https://pear.php.net/package/Mail_Mime/. [Kasutatud 20 märts 2023].

[8] F. Potencier, „The end of Swiftmailer,“ Symfony SAS, 19 08 2021. [Võrgumaterjal]. Available: https://symfony.com/blog/the-end-of-swiftmailer. [Kasutatud 15 märts 2023].

[9] V. V. j. J. Laanpere, „Tarkvaraanalüüs ja testimine - Tarkvaraarendus nõuded,“ Tallinna Ülikool, [Võrgumaterjal]. Available: https://web.htk.tlu.ee/digitaru/testimine/front-matter/introduction/. [Kasutatud 17 veebruar 2023].

[10] T. P. Group, „What is PHP?,“ [Võrgumaterjal]. Available: https://www.php.net/manual/en/intro-whatis.php. [Kasutatud 05 Märts 2023].

[11] „What Is PHP?,“ hostinger.com, 01 03 2023. [Võrgumaterjal]. Available: https://www.hostinger.com/tutorials/what-is-php/. [Kasutatud 12 märts 2023].

[12] I. Inc., „Web programming languages: the best languages for web development,“ IONOS Inc., 11 03 2019. [Võrgumaterjal]. Available: https://www.ionos.com/digitalguide/websites/web-development/web-programming-languages/. [Kasutatud 09 märts 2023].

[13] „What Is Python Used For? A Beginner’s Guide,“ Coursera Inc, 14 aprill 2023. [Võrgumaterjal]. Available: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python. [Kasutatud 20 aprill 2023].

[14] A. S, „Top 10 PHP Frameworks in 2023,“ NamLabs Technologies Pvt Ltd, 04 01 2023. [Võrgumaterjal]. Available: https://www.atatus.com/blog/top-php-frameworks/. [Kasutatud 05 märts 2023].

[15] E. Ecosystem, „Symfony History,“ Education Ecosystem, [Võrgumaterjal]. Available: https://educationecosystem.com/guides/programming/symfony/history. [Kasutatud 10 märts 2023].

[16] F. Potencier, „symfony 1.x legacy website,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/legacy. [Kasutatud 10 märts 2023].

[17] F. Z. a. F. Potencier, The Definitive Guide to Symfony, New York: Apress, 2007.

[18] E. Ecosystem, „Laravel History,“ Education Ecosystem, [Võrgumaterjal]. Available: https://educationecosystem.com/guides/programming/laravel/history. [Kasutatud 10 märts 2023].

[19] C. Foundation, „CodeIgniter- The small framework with powerful features,“ CodeIgniter Foundation, [Võrgumaterjal]. Available: https://codeigniter.com/. [Kasutatud 10 märts 2023].

[20] „Which PHP Framework to Choose: Laravel vs CodeIgniter vs Symfony,“ OurCodeWorld, 07 07 2020. [Võrgumaterjal]. Available: https://ourcodeworld.com/articles/read/1252/which-php-framework-to-choose-laravel-vs-codeigniter-vs-symfony. [Kasutatud 10 aprill 2023].

[21] I. Inc., „CodeIgniter – The lightweight of the PHP frameworks,“ IONOS Inc., 16 03 2020. [Võrgumaterjal]. Available: https://www.ionos.com/digitalguide/websites/web-development/codeigniter-the-lean-php-framework/. [Kasutatud 10 märts 2023].

[22] „Service Requirements- Supported Databases,“ CodeIgniter Foundation, [Võrgumaterjal]. Available: https://www.codeigniter.com/user_guide/intro/requirements.html#id3. [Kasutatud 10 aprill 2023].

[23] M. Foundation, „MariaDB Server: The open source relational database,“ MariaDB Foundation, [Võrgumaterjal]. Available: https://mariadb.org/. [Kasutatud 06 märts 2023].

[24] T. P. G. D. Group, „What is PosgreSQL?,“ The PostgreSQL Global Development Group, [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 06 märts 2023].

[25] I. Ali, „Top Code Editors and IDE for PHP Development,“ Cloudways Ltd, 18 01 2023. [Võrgumaterjal]. Available: https://www.cloudways.com/blog/top-ide-and-code-editors-php-development/. [Kasutatud 05 märts 2023].

[26] JetBrains, „PHP Turns 25,“ [Võrgumaterjal]. Available: https://www.jetbrains.com/lp/php-25/. [Kasutatud 06 märts 2023].

[27] J. s.r.o, „PhpStorm: Subscription options & pricing,“ JetBrains s.r.o., [Võrgumaterjal]. Available: https://www.jetbrains.com/phpstorm/buy/#personal. [Kasutatud 06 märts 2023].

[28] J. s.r.o., „PhpStorm: Features,“ JetBrains s.r.o., [Võrgumaterjal]. Available: https://www.jetbrains.com/phpstorm/features/. [Kasutatud 10 märts 2023].

[29] T. A. S. Foundation, „Apache Netbeans History,“ The Apache Software Foundation, [Võrgumaterjal]. Available: https://netbeans.apache.org/about/history.html. [Kasutatud 06 märts 2023].

[30] T. A. S. Foundation, „Code Assistance in the NetBeans IDE Java Editor: A Reference Guide,“ The Apache Software Foundation, [Võrgumaterjal]. Available: https://netbeans.apache.org/kb/docs/java/editor-codereference.html#_general_editor_features. [Kasutatud 10 märts 2023].

[31] J. Skinner, „1.0!,“ Sublime HQ Pty Ltd, 18 01 2007. [Võrgumaterjal]. Available: https://www.sublimetext.com/blog/articles/one-point-oh. [Kasutatud 10 märts 2023].

[32] S. H. P. Ltd, „Sublime Text,“ Sublime HQ Pty Ltd, [Võrgumaterjal]. Available: https://www.sublimetext.com/. [Kasutatud 10 märts 2023].

[33] S. H. P. Ltd, „Download,“ Sublime HQ Pty Ltd, [Võrgumaterjal]. Available: https://www.sublimetext.com/download. [Kasutatud 10 märts 2023].

[34] C. Hope, „Notepad++,“ Computer Hope, 16 05 2020. [Võrgumaterjal]. Available: https://www.computerhope.com/jargon/n/notepad-plus-plus.htm. [Kasutatud 10 märts 2023].

[35] D. Ho, „What is Notepad++,“ [Võrgumaterjal]. Available: https://notepad-plus-plus.org/. [Kasutatud 10 märts 2023].

[36] S. Mino, „IDE vs Code Editor--Why and When to Use Them,“ Jobsity, LLC, 29 04 2022. [Võrgumaterjal]. Available: https://www.jobsity.com/blog/ide-vs-code-editor-why-and-when-to-use-them. [Kasutatud 14 märts 2023].

[37] SendGrid, „Twilio SendGrid,“ SendGrid, [Võrgumaterjal]. Available: https://sendgrid.com/. [Kasutatud 15 märts 2023].

[38] W. Inc, „Interview with Isaac Saldana of SendGrid,“ WhoAPI Inc, 13 12 2017. [Võrgumaterjal]. Available: https://whoapi.com/blog/sendgrid-became-one-of-largest-api-companies-in-the-world-isaac-saldana-interview/. [Kasutatud 15 märts 2023].

[39] G. David, „Twilio SendGrid: The Cloud SMTP Provider,“ Cellenza, 22 06 2022. [Võrgumaterjal]. Available: https://blog.cellenza.com/en/cloud/twilio-sendgrid-the-cloud-smtp-provider/. [Kasutatud 15 märts 2023].

[40] „Design & Code Editor,“ Twilio Inc, [Võrgumaterjal]. Available: https://docs.sendgrid.com/ui/sending-email/editor. [Kasutatud 10 aprill 2023].

[41] „A/B Testing Your Single Send,“ Twilio Inc, [Võrgumaterjal]. Available: https://docs.sendgrid.com/ui/sending-email/a-b-testing. [Kasutatud 10 aprill 2023].

[42] SendGrid, „Email API Plans,“ SendGrid, [Võrgumaterjal]. Available: https://sendgrid.com/pricing/. [Kasutatud 15 märts 2023].

[43] S. A. Inc, „Mailgun,“ Software Advice Inc, [Võrgumaterjal]. Available: https://www.softwareadvice.com/email-verification/mailgun-profile/. [Kasutatud 15 märts 2023].

[44] „Your Guide To Webhooks,“ Mailgun, 05 12 2022. [Võrgumaterjal]. Available: https://www.mailgun.com/blog/email/your-guide-to-webhooks/. [Kasutatud 10 aprill 2023].

[45] Mailgun, „Flexible plans built for email delivery,“ Mailgun, [Võrgumaterjal]. Available: https://www.mailgun.com/pricing/. [Kasutatud 15 märts 2023].

[46] A. Goel, „Postmark Review 2023 (Features, Limitations, Pricing),“ GMass, Inc, 06 02 2023. [Võrgumaterjal]. Available: https://www.gmass.co/blog/postmark-review/. [Kasutatud 10 aprill 2023].

[47] L. ActiveCampaign, „Transparent pricing with no surprises,“ ActiveCampaign, LLC, [Võrgumaterjal]. Available: https://postmarkapp.com/pricing. [Kasutatud 15 märts 2023].

[48] „What is Scrum?,“ Scrum.org, [Võrgumaterjal]. Available: https://www.scrum.org/learning-series/what-is-scrum. [Kasutatud 19 märts 2023].

[49] J. Arundel, „Introducing Test-Last Development (TLD),“ 4 05 2021. [Võrgumaterjal]. Available: https://bitfieldconsulting.com/golang/test-last-development. [Kasutatud 10 aprill 2023].

[50] „Creating Symfony Applications,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/setup.html#creating-symfony-applications. [Kasutatud 29 märts 2023].

[51] „Installing & Setting up the Symfony Framework,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/setup.html. [Kasutatud 19 märts 2023].

[52] „File Structure,“ Broad Research Communication Lab, [Võrgumaterjal]. Available: https://mitcommlab.mit.edu/broad/commkit/file-structure/. [Kasutatud 12 aprill 2023].

[53] „The Kernel Class,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/configuration/front_controllers_and_kernel.html#the-kernel-class. [Kasutatud 29 märts 2023].

[54] „CQRS pattern,“ Microsoft, [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs. [Kasutatud 11 aprill 2023].

[55] „Event-driven architecture style,“ Microsoft , [Võrgumaterjal]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/architecture-styles/event-driven. [Kasutatud 11 aprill 2023].

[56] „Database: Migrations,“ Laravel LLC, [Võrgumaterjal]. Available: https://laravel.com/docs/10.x/migrations#migration-structure. [Kasutatud 26 märts 2023].

[57] „Sending Emails with Mailer,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/mailer.html#installation. [Kasutatud 11 aprill 2023].

[58] „How to use symfony/mailer without the Symfony Framework,“ doeken.org, 09 09 2021. [Võrgumaterjal]. Available: https://doeken.org/blog/using-symfony-mailer-without-framework. [Kasutatud 6 aprill 2023].

[59] „Sending Emails with Mailer- Transport Setup,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/mailer.html#transport-setup. [Kasutatud 2 aprill 2023].

[60] A. Garay, „CQRS with Symfony Messenger,“ DEV Community, 12 08 2022. [Võrgumaterjal]. Available: https://dev.to/adgaray/cqrs-with-symfony-messenger-2h3g. [Kasutatud 6 aprill 2023].

[61] „Messenger: Sync & Queued Message Handling,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/messenger.html. [Kasutatud 11 aprill 2023].

[62] „Messenger: Sync & Queued Message Handling - Doctrine Transport,“ Symfony SAS, [Võrgumaterjal]. Available: https://symfony.com/doc/current/messenger.html#doctrine-transport. [Kasutatud 6 aprill 2023].

[63] A. Fatrah, „Handle asynchronous tasks in Symfony with Messenger,“ 26 10 2022. [Võrgumaterjal]. Available: https://aicha-fatrah.medium.com/handle-asynchronous-tasks-in-symfony-with-messenger-bbaf6ed36dbf. [Kasutatud 10 aprill 2023].

[64] Y. Biberoğlu, „Symfony 5 Contact Form with Mailer and Mime component(MailerInterface),“ 02 06 2020. [Võrgumaterjal]. Available:

https://yusufbiberoglu.medium.com/symfony-contact-form-with-mailer-and-mime-component-mailerinterface-4e0a753c58c2. [Kasutatud 10 aprill 2023].

[65] L. Gupta, „What is REST,“ 07 04 2022. [Võrgumaterjal]. Available: https://restfulapi.net/. [Kasutatud 20 aprill 2023].

[66] „What is CI/CD?,“ Red Hat, Inc, 11 05 2022. [Võrgumaterjal]. Available: https://www.redhat.com/en/topics/devops/what-is-ci-cd#continuous-integration. [Kasutatud 26 märts 2023].

[67] „CronJob,“ The Kubernetes Authors, 07 03 2023. [Võrgumaterjal]. Available: https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/. [Kasutatud 20 aprill 2023].

[68] O. Mirtes, „PHPStan,“ [Võrgumaterjal]. Available: https://phpstan.org/. [Kasutatud 18 aprill 2023].

[69] H. Escafit, „Code Coverage: Everything You Need to Know,“ The Mergify Blog, 14 12 2022. [Võrgumaterjal]. Available: https://blog.mergify.com/code-coverage-everything-you-need-to-know/. [Kasutatud 6 aprill 2023].

[70] „PHP Code Coverage Tools,“ PHP.Watch, 12 10 2020. [Võrgumaterjal]. Available: https://php.watch/articles/php-code-coverage-comparison. [Kasutatud 6 aprill 2023].

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

Mina, Annika Roosleht

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Klienditeavituste rakenduse juurutamine Zone Media OÜ näitel“, mille juhendajateks on Toomas Lepikult ja Ats Aim
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

15.05.2023

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

Lisa 2 Migratsioon andmebaasi tabeli *Notifies* täiendamiseks

```php
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Capsule\Manager as Capsule;

class AddParamsJsonAndFlagsToNotifies extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Capsule::connection('   ')->getSchemaBuilder()->table('notifies', function (Blueprint $table) {
            $table->array('params_json')->after('params')->default(array);
            $table->string('flag')->nullable()->default(null);
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Capsule::connection('   ')->getSchemaBuilder()->table('notifies', function (Blueprint $table) {
            $table->dropColumn('params_json');
            $table->dropColumn('flag');
        });
    }
}
```

Lisa 3 – Symfony Mailer-i DSN ja transpordi konfiguratsiooni failid

config/packages/mailer.yaml

```
framework:
    mailer:
        dsn: '%env(MAILER_DSN)%'
```

config/packages/Messenger.yaml

```
framework:
    messenger:
        failure_transport: failed
        transports:
            async: '%env(MESSENGER_TRANSPORT_DSN)%'
            failed: 'doctrine://default?queue_name=failed'
        routing:
            'App\Message\SendNotifyMessage': async
```