Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Jaka Krajnc

Uporaba platforme Salesforce za razvoj aplikacij v zdravstvu

diplomsko delo

visokošolski študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Damjan Vavpotič

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednje delo:

Tematika dela:

V diplomskem delu najprej predstavite vodilne standarde, ki so se z leti oblikovali na področju razvoja zdravstvenih informacijskih sistemov. Nato predstavite platformo Salesforce in analizirajte njeno primernost za razvoj aplikacij v zdravstvu. Pri tem upoštevajte pomembnejše zahteve, ki jih morajo izpolnjevati zdravstveni informacijski sistemi ter se še posebej posvetite vprašanju njihove medobratovalnosti. Izdelajte prototip, s pomočjo katerega boste preverili primernost platforme Salesforce za potrebe izmenjave zdravstvenih podatkov skladno s standardom, ki ga boste pred tem identifcirali kot najprimernejšega za uporabo v okviru omenjene platforme. Rezultate dela kritično ovrednotite.

Zahvaljujem se mentorju doc. dr. Damjanu Vavpotiču za nasvete in strokovno pomoč pri izdelavi diplomskega dela. Zahvala gre tudi staršema, ki sta mi omogočila študij. Hvala prijateljem, sošolcem in vsem bližnjim za spodbudo in podporo tekom študija.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CRM	Customer Relationship Management	Sistem za upravljanje odnosov s strankami
ERP	Enterprise Resource Planning	Poslovni informacijski sistem
EHR	Electronic health record	Elektronski zdravstveni zapis
IT	Information technology	Informacijske tehnologije
SOA	Service-oriented architecture	Storitveno usmerjena arhitektura
XML	eXtensible Markup Language	Označevalni jezik
OOM	Object oriented methodology	Objektno usmerjena metodologija
WSDL	Web services description language	Jezik za opis spletnih storitev

Povzetek

Naslov: Uporaba platforme Salesforce za razvoj aplikacij v zdravstvu

Avtor: Jaka Krajnc

Na področju zdravstva v zadnjem času prihaja do velike želje po digitalizaciji podatkov in procesov. Kot lahko opazimo pri obisku zdravstvenih ustanov, je v uporabi še vedno veliko papirnate dokumentacije. Za digitalizacijo takšnih procesov in podatkov potrebujemo sistem, ki izpolnjuje ustrezne funkcionalnosti ter je sposoben medobratovalnega delovanja, kar pomeni, da si lahko z drugimi sistemi učinkovito in na standarden način izmenjuje podatke. Na področju zdravstva so se skozi čas oblikovali standardi za načrtovanje takšnih sistemov in standardi za medsebojno izmenjavo sporočil. V prvem delu diplomskega dela smo pregledali vodilne standarde na področju razvoja zdravstvenih sistemov in ocenili njihovo primernost za uporabo v okviru platforme Salesforce. V drugem delu smo analizirali primernost omenjene platforme za razvoj aplikacij v zdravstvu na podlagi ključnih funkcionalnosti, ki jih zahtevajo takšni sistemi. Za konec pa smo prikazali še podporo platforme za izmenjavo podatkov v standardnem formatu s pomočjo standarda, za katerega smo ocenili, da bi bil v okviru platforme najbolj primeren za uporabo.

Ključne besede: HL7, FHIR, EHR, Salesforce, CRM, medobratovalnost.

Abstract

Title: Use of Salesforce platform for application developement in healthcare

Author: Jaka Krajnc

In the healthcare, there is an increased need to digitize data and processes, especially in the last few years. As we can see in health institutions, there is a lot of paperwork still in use. To digitize such data and processes, we need a system that meets the appropriate functionalities and that is capable of interoperability operations with other systems. That means that it can effectively exchange information with other systems in a standard way. Through years of development, many standards have been developed in healthcare for the purpose of architectural design and to ensure interoperability between systems. In the first part of the thesis we reviewed the leading standards in the healthcare industry and assessed their suitability with the Salesforce platform. In the second part, we analyzed the suitability of Salesforce platform in healthcare based on the key functionalities required by such systems. For the end, we also checked the support of platform for the data exchange in a standard format. We used a standard that we felt would be most suitable for use within the platform.

Keywords: HL7, FHIR, EHR, Salesforce, CRM, Interoperability.

Poglavje 1

Uvod

V marsikaterem podjetju je v zadnjih letih moč opaziti korenite spremembe pri obravnavi svojih strank, zaposlenih in ostalih vpletenih v dejavnost podjetij s pomočjo digitalizacije in prenove poslovnih procesov. Pojavljajo se nova orodja in metode, ki podjetjem omogočajo, da so v poslu učinkovitejša od konkurence. Če izvzamemo mikro podjetja, se danes že odraža dejstvo, da večja podjetja brez takšnih sistemov praktično ne morejo več uspešno poslovati. Eden od sistemov, o katerih govorimo, je tudi sistem CRM (angl. *Customer Relationship Model)*, ki predstavlja stično točko med stranko in podjetjem. Sistemi CRM podjetjem omogočajo iskanje novih strank, zbiranje podatkov o obstoječih strankah, spremljanje dobičkonosnosti ter še mnogo več. Sistem CRM zajema vse aktivnosti, da bi podjetje obstoječe stranke obdržalo, pridobilo nove, gradilo dobre odnose ter, kar je najpomembneje, ohranjalo zadovoljstvo strank.

Koncept takšnega sistema bi si želeli tudi v zdravstvu, vendar ko prihajamo v stik z zdravstvom, opazimo, da je obravnava uporabnikov zdravstvenih storitev še vedno v veliki meri podobna tisti, ki smo jo bili vajeni pred nekaj desetletji. To nakazuje na to, da je na področju zdravstva še veliko prostora za vpeljavo rešitev, ki bi uporabnikom zdravstvenih storitev izboljšale njihovo izkušnjo pri obiskih zdravstvenih ustanov. Kot lahko opazimo, se v zdravstvu še vedno uporablja veliko papirnate dokumentacije, prav tako

pa je potrebno za veliko rutinskih zadev še vedno obiskati svojega osebnega zdravnika. Tudi sistemi med različnimi oddelki v veliki meri med sabo niso povezani, zato na primer specialist za posamezno zdravstveno področje, ki ne pozna pacientovega ozadja, nima vpogleda v pacientov osebni karton.

Po drugi strani pa bi si marsikdo želel naročila k zdravniku kar preko mobilne aplikacije. S tem bi se odprle še marsikatere druge možnosti, ki jih ponujajo sodobne tehnologije. S pomočjo takšnega sistema bi lahko na podlagi zbranih podatkov izvajali tudi napredno podatkovno analitiko in glede na življenjski slog napovedali tveganja za določena obolenja. Takšen sistem bi ponudil ogromno možnosti in najverjetneje izboljšal izkušnjo pri uporabi zdravstvenih storitev.

Ker bomo v nadaljevanju večkrat uporabili besedo medobratovalnost, bomo najprej pojasnili njen pomen. Medobratovalnost (angl. *Interoperability*) je zmožnosti izdelka ali sistema, katerega vmesniki so popolnoma razumljivi, za delo z drugimi izdelki ali sistemi, brez kakršnih koli omejitev [17].

V zadnjih letih je tudi na področju elektronskega zdravja prisotna želja po informatizaciji in medsebojni integraciji sistemov. Opazen je trend povečanja uporabe elektronskih zdravstvenih zapisov v bolnišnicah in preostalih zdravstvenih ustanovah [12]. To predstavlja dodatno prednost, saj smo z uporabo elektronskih zdravstvenih zapisov na pragu nove dobe, kjer bomo lahko na podlagi teh zapisov izvajali bolj uspešne zdravstvene raziskave. Pomembno vlogo tukaj igrajo sistemi, ki bodo morali biti zmožni izmenjave informacij s pomočjo mednarodnih standardov za zagotavljanje medobratovalnosti, saj bomo tako zagotovili konsistentnost, popolnejše beleženje in izmenjavo podatkov za različne skupine bolnikov [4]. Pri razvoju zdravstvenih sistemov se pojavi veliko izzivov, predvsem zaradi različnih sistemov, iz katerih črpamo podatke, in podatkov, ki se lahko vsebinsko hitro spremenijo [25].

Da bi omogočili uporabnikom zdravstvenih storitev sodobnejšo in učinkovitejšo obravnavo, potrebujemo sistem, ki bo v veliki meri podoben sistemu CRM ter bo zmožen medobratovalnega delovanja. Na trgu sicer že obstajajo sistemi,

ki ponujajo rešitve CRM za potrebe zdravstva, ker pa Salesforce trenutno velja za vodilni splošno namenski sistem CRM na trgu, ki se redno posodablja in nadgrajuje z novimi in izboljšanimi obstoječimi funkcionalnostmi ter ponuja ogromno naprednih možnosti, smo v okviru dela analizirali le tega. Ker želimo, da bo naš sistem skladen in medobratovalen, bomo v diplomskem delu najprej pregledali vodilne standarde, ki se danes uporabljajo za razvoj sistemov in izmenjavo informacij na področju zdravstva. Na podlagi rezultatov analize bomo izmed standardov izbrali najprimernejšega za uporabo na platformi Salesforce. V nadaljevanju si bomo na kratko pogledali dotični CRM-sistem Salesforce in analizirali njegovo uporabo za namen razvoja aplikacij v zdravstvu. Pri tem bomo upoštevali ključne funkcionalnosti, ki jih zahtevajo takšni sistemi. Na koncu pa bomo izdelali prototip standardne izmenjave sporočil med našo aplikacijo in zunanjim sistemom. S tem bomo preverili podporo platforme Salesforce za standardno izmenjavo sporočil ter težavnost takšne implementacije na platformi.

Namen dela je pregled trenutno vodilnih standardov na področju elektronskih zdravstvenih zapisov in ovrednotenje njihove primernosti za uporabo v okviru platforme Salseforce ter analiza primernosti platforme Salesforce za razvoj aplikacij v zdravstvu na podlagi funkcionalnosti, ki jih zahtevajo zdravstveni informacijski sistemi. Poleg tega je namen dela tudi preučiti posebnosti, ki jih je potrebno upoštevati pri razvoju na omenjeni platformi ter izpostaviti prednosti in slabosti, ki jih takšen razvoj prinese. To pa bomo prikazali v okviru implementacije prototipa, v katerega bomo vključili tudi vodilni standard za zagotavljanje medobratovalnosti.

Poglavje 2

Elektronski zdravstveni sistemi

V tem poglavju si bomo pogledali, kaj pomeni pojem e-zdravje, zakaj potrebujemo standarde ter s kakšnimi izzivi se srečujemo pri izdelavi informacijskih sistemov v zdravstvu.

2.1 E-zdravje

Elektronsko zdravje je izraz, ki zajema uporabo informacijsko-komunikacijskih tehnologij v zdravstvu in je postal neločljiv del vizije sodobne zdravstvene oskrbe. V preteklih letih se je na področju zdravstva pričel vse večji razvoj v smeri informatizacije, vendar kljub tehnologiji, ki jo imamo na voljo, sistemi še vedno niso na dovolj visokem nivoju, kakršnem bi lahko v tem trenutku bili. Tekom razvoja sodobnih informacijskih sistemov se je pojavilo tudi veliko izkušenj na področju digitalizacije elektronskih zapisov in se izkazalo, da obstajajo še številni pomembni izzivi, eden od njih je tudi standardna izmenjava podatkov med različnimi sistemi [25].

2.2 Standardi

Da bi zagotovili medobratovalnost med sistemi, potrebujemo standarde. Standard je definicija, nabor pravil ali smernic, oblika ali dokument, ki določa

enotne inženirske ali tehnične specifikacije, merila, metode, procese in prakse. Poleg tega morajo biti standardi odobreni s strani priznane organizacije za razvoj standardov ali pa jih mora sprejeti panoga. Na tem mestu bi omenili tudi pobudo eZdravje (angl. *eHealth*) [19]. Gre za neodvisno in neprofitno organizacijo, ki spodbuja zdravnike in paciente k sodelovanju pri standardizaciji in preoblikovanju zdravstvene informacijske tehnologije. Ta definira standard kot pristop, ki podpira poslovni proces in je bil dogovorjen s strani skupine strokovnjakov in bil javno preverjen. Standarde je mogoče opisati tudi kot skupino smernic v zvezi z bistvenimi zahtevami, ki jih mora izpolnjevati določen proces, izdelek ali storitev za doseganje kakovostnih ciljev. Standardi omogočajo, da se podatki delijo med različnimi sistemi in posameznimi zainteresiranimi stranmi ne glede na njihovo vlogo. Pravzaprav so standardi temelj medobratovalnosti, saj brez njih ni mogoče graditi medobratovalnih sistemov [19].

2.2.1 Vloga standardov

Težave s pomanjkanjem standardov za izmenjavo zdravstvenih podatkov in zagotavljanjen medobratovalnosti niso trivialne. Mnogo zdravstveno-informacijsko-tehnoloških projektov je bilo ne glede na njihovo velikost neuspešnih, ker zdravstvene IT storitve in aplikacije niso bile medobratovalne. V zdravstvenih sistemih je težko doseči učinkovito izmenjavo informacij zaradi pomanjkanja standardov. Študije kažejo, da pomanjkanje standardov ustvarja oviro za ljudi, da bi uspešno sodelovali, ker njihovi informacijski sistemi niso medobratovalni, zaradi česar je težko deliti ali interpretirati podatke med sistemi [19]. Za integracijo različnih zdravstvenih sistemov je treba podatke preko sistema v sistem prenašati preko vmesnikov.

Če torej želimo med seboj povezati n nemedobratovalnih sistemov, potem potrebujemo

$$\frac{N^2}{N-1} \tag{2.1}$$

vmesnikov, kar pomeni, da število vmesnikov med sistemi narašča eksponentno s številom sistemov, ki jih želimo povezati.

2.3 Integrating the Healthcare Enterprise (IHE)

Integrating the Healthcare Enterprise je pobuda zdravstvenih strokovnjakov in panoge za izboljšanje načina izmenjave informacij med različnimi zdravstvenimi sistemi. IHE spodbuja usklajeno uporabo uveljavljenih standardov za obravnavanje specifičnih zdravstvenih potreb v podporo optimalni oskrbi pacientov. Sistemi, razviti v skladu z IHE, komunicirajo med seboj bolje ter omogočajo ponudnikom oskrbe bolj učinkovito uporabo informacij. Zdravniki, specialisti, medicinske sestre, administratorji in ostali izvajalci zdravstvene oskrbe si želijo, da se bodo informacije brez težav pretakale med sistemi in oddelki ter bodo na voljo v vsakem trenutku. IHE je zasnovan tako, da pomaga pri uresničitvi te vizije z izboljšanjem stanja medobratovalnosti sistemov in izboljšanjem pretoka informacij, kar pa posledično pomeni kakovostnejšo zdravstveno oskrbo [16].

2.4 Splošna tveganja tekom razvoja zdravstvenih sistemov

Problem, ki se največkrat pojavi v času izvajanja in načrtovanja projekta, je pomanjkanje sodelovanja zdravstvenih delavcev, ki so končni uporabniki sistema. To se zgodi iz več razlogov, in sicer zaradi pomanjkanja strokovnih sodelavcev, ki bi lahko sodelovali na projektu, ali pa nezmožnost pravilne komunikacije med razvojno in implementacijsko ekipo. Rezultat je najpogosteje sistem, ki povzroča številne težave zdravstvenim delavcem pri njihovem vsakdanjem delu, saj ni bil zgrajen v skladu z njihovimi pričakovanji. Težave se lahko pojavijo tudi kmalu po uvajanju sistema predvsem zaradi dinamičnosti zdravstvenih podatkov in postopkov, saj se le ti razvijajo in oblikujejo postopoma. Zaradi napredkov na področju medicine je prej ali slej potrebna

sprememba v vsebini podatkov v našem sistemu, kar pa povzroča dodatne stroške za samo vzdrževanje [25].

Tekom diplomskega dela nas bo tako med drugim zanimalo tudi, kakšne so zmožnosti platforme Salesforce za prilagajanje obstoječe rešitve prav zaradi sprememb, ki so v zdravstvu dokaj pogoste.

Poglavje 3

Pregled vodilnih standardov

3.1 OpenEHR

OpenEHR je virtualna skupnost, ki se ukvarja s preoblikovanjem zdravstvenih podatkov iz fizične v elektronsko obliko in zagotavljanjem medobratovalnosti med vsemi oblikami elektronskih zdravstvenih zapisov. Primarna prizadevanja so osredotočena predvsem na elektronske zdravstvene zapise ter z njimi povezane sisteme. Glavni razlog zasnove in razvoja standarda openEHR je vse večja želja po informatizaciji procesov v zdravstvu. Ker med sistemi želimo čim večjo skladnost in medobratovalnost, je treba standardizirati tako konceptualne kot strukturne načrte sistemov. Standard OpenEHR k razvoju sistema pristopa z večnivojskim modeliranjem znotraj storitveno usmerjene programske arhitekture (angl. *Service-oriented arhitecture*), v kateri so modeli, ki jih strokovnjaki uporabljajo za posamezna področja, zgrajeni vsak v svojem sloju [25].

3.1.1 Arhitektura openEHR

Arhitektura standarda sestoji iz naslednji ključnih elementov:

- Referenčni model (RM),
- Arhetipni model (AM),

- Model storitev (SM),
- Arhetipni poizvedovalni jezik (AQL).

Referenčni model

Temelj standarda openEHR je referenčni model. Gre za zelo stabilen informacijski model, ki definira logične strukture zdravstvenih zapisov in demografskih podatkov. Za vpis kakršnega koli zdravstvenega zapisa v sistem openEHR moramo upoštevati referenčni model. OpenEHR fundacija zagotavlja specifikacijo referenčnega modela, ki je formalno logična opredelitev informacij in ne konkretna fizična podatkovna shema [25].

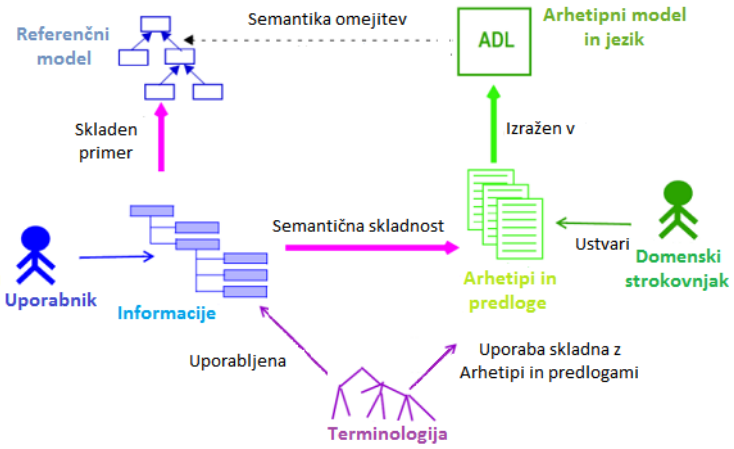
Arhetipni model

Arhetipni model vsebuje modele, potrebne za opis semantike arhetipov (angl. *Archetype*), in predlog (angl. *Teplate*) za njihovo uporabo v openEHR. Arhetip je formalna definicija informacij o ravni domene, opredeljena na podlagi pravil nad referenčnim modelom. Za definiranje arhetipov uporabljamo jezik *ADL – Archetype Definition Language*. To je formalni jezik za izražanje arhetipov. Na voljo sta dve pomembnejši verziji, in sicer ADL 1.4 in pa modernejša ADL 2, ki se počasi širše sprejema. Arhetipe lahko med sabo povezujemo in jih ponovno uporabimo. Prav tako jih lahko v openEHR uporabimo za generiranje uporabniškega vmesnika ali pa na primer nekega točno določenega dokumenta. Za lažjo predstavo lahko zbirko arhetipov razumemo kot knjižnico definicij vsebine, ki jih je mogoče ponovno uporabiti, pri kateri vsak arhetip deluje kot enota, katere vsebina se sooblikuje, pregleduje in objavlja [25].

Predloge so uporabljene za logično predstavitev podatkovnega nabora, specifičnega za določeno uporabo. Na primer podatki, ki so vključeni v odpustnico pacienta iz bolnišnice. Predloga je sestavljena tako, da se sklicuje na ustrezne postavke iz številnih arhetipov. Tehnično predloge ne morejo kršiti semantike arhetipov, iz katerih so zgrajene. Predloge se skoraj vedno zgra-

jene za lokalno uporabo s strani razvijalcev in zdravstvenih analitikov in so običajno definirane za obrazce v uporabniškem vmesniku, definicije sporočil ali dokumentov [25]. Koncept modeliranja arhetipov in predlog je prikazan na sliki 3.1.

Mednarodna knjižnica arhetipov openEHR trenutno vsebuje okoli 500 arhetipov. Prednost izdelave arhetipov je v tem, da jih lahko modeliramo z zdravstvenimi strokovnjaki, ki nimajo tehnološkega predznanja o sistemih EHR, vendar dobro poznajo svoje problemsko področje [25].



Slika 3.1: Prikaz koncepta modeliranja arhetipov in predlog [26]

Arhetipi in dvonivojsko modeliranje

Ena od ključnih paradigem, na katerih temelji openEHR, je tako imenovano dvonivojsko modeliranje. V okviru tega je najprej modeliran referenčni model, medtem ko so formalne definicije zdravstvene vsebine v obliki arhetipov in predlog modelirane naknadno. V programski opremi je implementirana le prva raven – referenčni model, kar znatno zmanjša odvisnost sistema in podatkov, ki večinoma vsebujejo spremenljivo vsebino. Posledično imajo sistemi možnost, da postanejo bistveno manjši in lažji za vzdrževanje kot enonivojski sistemi. Prav tako se samodejno prilagajajo, saj so zgrajeni za

uporabo arhetipov in predlog, ki se bodo v prihodnosti še lahko spreminjali.

Model storitev

Modeli storitev (angl. *Service model*) opredeljujejo dostop do ključnih zalednih storitev, medtem ko se za dostop do aplikacij uporablja večinoma aplikacijske vmesnike, ki temeljijo na protokolu REST (angl. *Representational state transfer protocol*). Storitveni model vključuje opredelitve osnovnih storitev v zdravstvenem informacijskem okolju, ki so osredotočene na elektronske zdravstvene zapise. To so storitve, ki omogočajo kreiranje elektronskih zdravstvenih zapisov, njihovo poizvedovanje ter njihovo urejanje. Prav tako nam omogočajo dostop do repozitorija arhetipov in predlog.

Arhetipni poizvedovalni jezik (AQL)

AQL nam omogoča poizvedovanje na podlagi arhetipov namesto poizvedovanja po dejanski fizični podatkovni shemi. S tem nam olajša delo, saj nam ni potrebno poznati podrobne zasnove dejanskega fizičnega podatkovnega modela. Eden od najpomembnejših ciljev openEHR je zagotoviti skladen in ponovno uporaben tip sistema za znanstveno in zdravstveno uporabo. "Jedro" referenčnega modela, ki je najnižja plast, zagotavlja identifikatorje, podatkovne tipe in podatkovne strukture, ki jih je mogoče ponovno uporabiti v višjih plasteh referenčnega modela in enako v plasteh arthetipnega modela in modela storitev. Odvisnosti vedno obstajajo le iz zgornjih slojev v spodnje sloje [26].

3.1.2 Prednosti pristopa openEHR

Sistemi, ki so bili razviti s pomočjo pristopa openEHR, imajo bistveno boljšo podporo pri računanju z zdravstvenimi podatki napram ostalim sistemom. Ključna prednost uporabe pristopa openEHR je v tem, da sistem vnaprej ne potrebuje informacij o zdravstvenih podatkih, ki jih bo obdeloval (kot so vitalne funkcije, diagnoze, naročanja itd.), saj so modeli za obdelavo teh in-

formacij razviti ločeno od sistema. Prav tako so posebej razviti tudi arhetipi in predloge. Na podlagi teh pa se generira uporabniški vmesnik, ki je kasneje viden uporabnikom sistema. To nam omogoča novo generacijo sistemov, ki se lahko zaradi zgoraj opisane arhitekture redno prilagajajo novim zahtevam [26].

3.1.3 Slabosti

OpenEHR standard ima v praksi manj primerov implementacij, zato je na tem področju tudi manjša skupnost strokovnjakov in posledično slabša podpora. Večino razvoja vodi eno samo podjetje (Ocean Informatics). Zaradi uporabe arhetipov je kompleksnost implementacije višja [2].

3.1.4 OpenEHR pristop v praksi

Komponente in sistemi, ki so skladni s standardom openEHR, so odprti v smislu podatkov, modelov in aplikacijskih vmesnikov. Ključna prednost pristopa openEHR je prilagodljivost, saj arhetipi niso del programske opreme in velik del programske opreme izhaja iz arhetipov. Specifikacija arhetipov je postala ISO standard (ISO 13606-2). Arhetipi so sedaj uporabljeni v več državah za specifikacijo standarda na nivoju nacionalnega elektronskega zdravstva [25].

3.1.5 Združljivost s platformo Salesforce

Specifikacija standarda openEHR definira, kako zgraditi sistem na podlagi dvonivojskega pristopa in uporabo dobrih praks pri implementaciji zdravstvenih sistemov. Standard pa se ne omejuje na točno določen programski jezik za implementacijo. Pri razvoju na platformi Salesforce moramo upoštevati arhitekturo, ki je že zastavljena s strani Salesforca, zato tukaj nimamo vpliva na izbiro tehničnih detajlov zalednega sistema. Ker je implementacija arhetipov v sistem že sama po sebi kompleksna, bi bila vpeljava takšnega koncepta

v Salesforce še zahtevnejša in nesmiselna, saj bi s tem rušili koncept platforme in izgubili prednosti, ki nam jih platforma že v osnovi ponuja. Kot smo omenili, nimamo vpliva na tehnično ozadje delovanja platforme, zato na platformi ne bi bilo mogoče implementirati arhetipov, kot jih definira openEHR, prav tako ne bi bilo mogoče uporabiti poizvedovalnega jezika AQL.

Kljub temu pa lahko koncepte pristopa openEHR uporabimo pri razvoju aplikacij in postavitvi arhitekture sistema na platformi Salesforce. Na platformi lahko modeliramo objekte, ki jih lahko v tem primeru obravnavamo kot neke vrste preslikavo arhetipov v objekte, razumljive s strani platforme. Objekt na platformi Salesforce si bomo lažje predstavljali kot tabelo v podatkovni bazi, atribute nekega objekta pa kot imena stolpcev, ki jih tabela vsebuje. Vsaka instanca objekta na platformi predstavlja zapis v omenjeni tabeli. Sami arhetipi, ki jih zagotavlja organizacija openEHR, so tudi sicer jezikovno nevtralni in jih lahko uporabimo v katerem koli programskem jeziku. V kolikor bi želeli na platformi Salesforce uporabiti določene arhetipe kot objekte, bi morali arhetip preslikati v definicijo objekta na platformi. Za takšno preslikavo bi bilo potrebno implementirati modul, ki bi znal arhetip prevesti v definicijo objekta, le to pa bi lahko kasneje uporabili v več sistemih. Druga, manj prijazna rešitev pa je, da objekt ustvarimo ročno. Ko imamo objekt ustvarjen, nam Salesforce omogoča prenos definicije objektov med različnimi organizacijami, ki uporabljajo platformo Salesforce, v našem primeru med zdravstvenimi sistemi, razvitimi na platformi Salesforce [22]. Prav tako lahko aplikacije, razvite na platformi, objavimo v Salesforcovi trgovini, imenovani AppExchange. Iz trgovine si lahko v našo organizacijo tudi nameščamo že obstoječe aplikacije.

Druga skupna lastnost s konceptom openEHR je ta, da v primeru, da pride do razširitve arhetipa z dodatnim atributom, to ne bo vplivalo na samo delovanje sistema, saj z razvijalskega vidika tukaj ni vpliva na podatkovno bazo, ampak za vse spremembe, ki se zgodijo v ozadju, poskrbi platforma sama [6]. V kolikor bi želeli razširiti objekt z dodatnim atributom, bi na platformi s klikom dodali novo polje ali pa bi uvozili novo definicijo objekta.

V trenutku ko dodamo novo polje, ga lahko nemudoma pričnemo uporabljati, prav tako pa nam ga platforma ponudi za prikaz na podrobnostih zapisa in vnosnih maskah.

Težava se pojavi, v kolikor bi želeli odstraniti določen atribut, vendar samo v primeru, da nad tem atributom izvajamo razširjene standardne funkcionalnosti, kjer se v kodi sklicujemo na izbrisan atribut. V tem primeru nas platforma opozori, da je polje uporabljeno v določenem delu programske kode, zato je potreben ročni poseg v kodo, kar pa ni v skladu s konceptom standarda openEHR.

3.2 Organizacija Health Level Seven International (HL7)

Health Level Seven International je neprofitna organizacija, ki razvija standarde, namenjene zagotavljanju celovitega ogrodja in s tem povezanih standardov za izmenjavo, povezovanje, deljenje in pridobivanje zdravstvenih informacij, ki podpirajo zdravstvene prakse, vodenja, izvajanja in vrednotenja zdravstvenih storitev. HL7 podpira več kot 1600 članov iz več kot 50 držav, med njimi tretjino članov, ki zastopajo izvajalce zdravstvenega varstva, storitev, farmacevtskih podjetij in ostalih članov zdravstvene stroke. Vizija organizacije HL7 je, da lahko kdorkoli varno dostopa in uporablja elektronske zdravstvene zapise, kjerkoli jih potrebuje [13]. Število sedem v imenu HL7 se nanaša na sedmi nivo sedemslojnega komunikacijskega modela *Open Systems Interconnection* – OSI (aplikacijski nivo). Poslanstvo organizacije HL7 je zagotoviti standarde, ki omogočajo globalno medobratovalnost med zdravstvenimi sistemi [13].

3.2.1 Pomembnejši standardi organizacije HL7

Standardi, za katere se zdi, da so najpogosteje uporabljeni in uveljavljeni na ravni HL7, so:

- HL7 verzija 2,
- HL7 verzija 3,
- FHIR (Fast Healthcare Interoperability Resources).

3.3 HL7 verzija 2

Standard za sporočanje HL7 verzije 2.x je standard za elektronsko izmenjavo podatkov v zdravstveni domeni in verjetno najbolj razširjen standard, ki se uporablja v zdravstvenih sistemih po svetu. Namenjen je podpori centralnemu sistemu zdravstvenega varstva in tudi bolj porazdeljenim sistemom, kjer se podatki nahajajo v oddelkih. Verzija 2 je uveljavljena kot eden najbolj razširjenih standardov za izmenjavo zdravstvenih podatkov na svetu. Prvič je bila objavljena leta 1987 kot aplikacijski protokol za elektronsko izmenjavo podatkov v zdravstvenih sistemih. V letu 2011 je bila objavljena verzija 2.7, ki predstavlja najnovejšo posodobitev standarda verzije 2. Zaradi široke uporabe bo verzija 2 še naprej igrala sestavni del sporočil o zdravstvenih podatkih, tudi s standardno različico HL7 verzije 3. HL7 se zavzema za podporo in razširitev verzije 2, vzporedno z verzijo 3. Verzija 2 ima več podverzij (2.2, 2.3, 2.3.1 ...), vse pa so združljive tudi s predhodnimi [13]. Najbolj pomembni verziji, ki ju uporablja organizacija IHE, sta verziji v2.3.1 in v2.5 [23].

3.3.1 Arhitektura

Ker so standardi HL7 sporočilni standardi, bomo v tem razdelku opisali zgradbo sporočila. Sporočila v verziji 2 ne uporabljajo sintakse XML, ampak temeljijo na podlagi sporočila, sestavljenega iz segmentov in enoznačnih ločil med njimi. Segmenti vsebujejo polja, ki so prav tako ločena z enoznačnimi ločili polja. Prav tako pa lahko imajo tudi polja dodatna podpolja ter ta svoja enoznačna ločila. Ločilo predstavlja ponovitev vnosa na istem nivoju.

Ločimo tri osnovne nivoje sporočila, ki si med sabo sledijo v hierarhiji:

- segment,
- polje,
- podpolje.

Vsak segment se prične s triznačno kratico, ki predstavlja tip segmenta. Prvi segment sporočila je vedno segment MSH, ki nam pove, za kateri tip sporočila gre ter katere segmente lahko v sporočilu pričakujemo. Vsak segment sporočila vsebuje eno specifično informacijo. Vrste segmentov, ki se uporabljajo v določeni vrsti sporočila, določi oznaka sheme, ki se uporablja v standardih HL7 [13]. Slika 3.2 prikazuje primer sporočila, slika 3.3 pa prikazuje segmente, ki jih vsebuje sporočilo za sprejem pacienta.

Primer sporočila

```
MSH|^~\&|GHH LAB|ELAB-3|GHH OE|BLDG4|200202150930||ORU^R01|CNTRL-3456|P|2.4
PID|||555-44-4444||EVERYWOMAN^EVE^E^^^^L|JONES|19620320|F|||153 FERNWOOD DR^
OBR|1|845439^GHH OE|1045813^GHH LAB|15545^GLUCOSE|||200202150730|||||||||
OBX|1|SN|1554-5^GLUCOSE^POST 12H CFST:MCNC:PT:SER/PLAS:QN||^182|mg/dl|70_105|H|||F
```

Slika 3.2: Sporočilo HL7 verzije 2

MSH	[1, 1]	Glava sporočila
EVN	[1, 1]	Tip dogodka
PID	[1, 1]	Identifikator pacienta
[{ NK1}]	[0, 3]	Bližnji sorodniki
PV1	[0, 1]	Informacije o obisku
PV2	[0, 1]	Informacije o obisku – dodatne informacije
[{AL1}]	[0, *]	Informacije o alergijah

[1, 1] – Obvezen segment z eno ponovitvijo

[0, 3] – Neobvezen segment z največ tremi ponovitvami

[0, 1] – Neobvezen segment z največ eno ponovitvijo

[0, *] – Neobvezen segment z več ponovitvami

Slika 3.3: Segmenti sporočila za sprejem pacienta

3.3.2 Prednosti

Prednost standarda je predvsem odprtost in zelo široka uporaba v svetu. Pri oblikovanju standarda so se odločili za pristop 80/20, kar pomeni, da se je predefiniralo samo osemdeset odstotkov vmesnika, preostalih dvajset odstotkov pa je ostalo odprtih za prilagajanje. Prav zaradi prilagodljivosti je standard dosegel širšo uporabnost in mednarodno sprejetje in je danes najbolj uporabljen standard v zdravstvenih sistemih. Standard je združljiv z vsemi svojimi predhodnimi verzijami.

3.3.3 Slabosti

Čeprav se je standard uveljavil prav zaradi njegove odprtosti, pa je po drugi strani to tudi slaba stran standarda. Slabost so nekonsistentni tipi sporočil, ki omogočajo preveč prožnosti in ne dovolijo popolno definirane rešitve. V standard so vpeljani tako imenovani Z-segmenti, ki nam omogočajo, da jih definiramo sami, kar pa lahko povzroči neskladja med sistemi. Standard nima

formalnih metodologij za modeliranje podatkovnih elementov in sporočil, kar povzroča neskladnost znotraj standarda in težave pri razumevanju, kako se elementi sporočila nanašajo drug na drugega. Standard nima podpore za nove tehnologije (XML, WEB, OOM). Prav tako je slaba stran standarda format sporočila, ki več ne sledi sodobnim smernicam in je tudi človeško slabo čitljiv. Standard ni združljiv z novejšim standardom HL7 verzije 3.

3.3.4 Uporaba standarda v praksi

Kot že omenjeno, je standard HL7 verzije 2 najbolj uporabljan standard za izmenjavo zdravstvenih podatkov. Največ se uporablja v ZDA. Standard ni popolna rešitev, vendar pa ima osnove, ki jih potrebuje zdravstvena panoga. Da bi zapolnili vrzeli, ki jih pušča omenjeni standard, so razvijalci razvili novejše standarde, ki vključujejo najboljše prakse ter rešujejo pomisleke, ki so se pojavile v celotnem ciklu razvoja standardov HL7. S tem sta se razvila standarda HL7 verzija 3 ter FHIR, ki ju bomo opisali v nadaljevanju.

3.3.5 Združljivost standarda s platformo Salesforce

V kolikor bi standard uporabili za izmenjavo podatkov v okviru platforme Salesforce, bi bilo potrebno razviti vsaj naslednje module [15]:

- razčlenjevalnik sporočila,
- validacijski modul in
- modul za sestavo sporočila.

Ob predpostavki, da sporočilo pridobimo v sistem, moramo sporočilo najprej razčleniti ter ga validirati. To predstavlja dodatno delo, saj standard definira več tipov sporočil. Tukaj nastane veliko možnosti za napačno interpretacijo sporočila predvsem zaradi potencialne nekonsistentnosti sporočil med različnimi sistemi, ki so posledica odprtosti standarda, kar pa bi dodatno pomenilo ponovno prilagajanje razčlenjevalnika za različne sisteme. Prav

tako bi nastalo dodatno delo z generiranjem sporočila, ki bi ga želeli poslati iz sistema, saj bi bilo potrebno logiko generiranja v celoti implementirati po meri.

Ker platforma Salasforce uporablja novejše spletne tehnologije, standard HL7 verzije 2 ni najbolj primeren standard za neposredno uporabo na platformi, saj predstavlja veliko nepotrebnega dodatnega dela z razčlenjevanjem, validiranjem in generiranjem sporočil, ki je precej kompleksno. Zato je bolje uporabiti novejše standarde organizacije HL7, ki temeljijo na sodobnih spletnih tehnologijah, saj lahko z njimi na bolj enostaven način izvajamo razčlenjevanje in generiranje sporočil. Kljub temu pa obstajajo alternativne možnosti podpore standarda, ki jih bomo obravnavali v petem poglavju.

3.4 HL7 verzija 3

HL7 verzija 3 pomeni veliko več kot samo novo različico med razvojem standarda. HL7 verzija 3 sledi novi paradigmi. Ta sprememba paradigme ni bila kratek korak, ampak dolgoročen in kontradiktoren proces. Osnove komunikacijskega standarda HL7 verzije 3 temeljijo na obsežni razvojni metodologiji okvirja *Version 3 Message Development Framework (MDF)* ta pa je bil nadaljnje zamenjan in razširjen v *HL7 Development Framework (HDF)* [24]. HDF je okvir za razvoj specifikacij, ki omogočajo medobratovalnosti med sistemi zdravstvenega varstva. Modeli, ki se uporabljajo v razvojni metodologiji HDF, uporabljajo sintakso UML. Gre za najnovejšo izdajo razvojne metodologije HL7 verzije 3. Če HL7 verzija 2 strogo sledi paradigmi sporočil, HL7 verzija 3 uveljavlja naslednja različna načela [18].

- Stopenjski pomik iz sporočila v arhitekturno paradigmo z uporabo HDF.
- Uvedba specifikacije sporočil, ki temeljio na referenčnem informacijskem modelu (angl. *Reference information model*).

HL7 verzija 3 predstavlja nov pristop k izmenjavi zdravstvenih informacij,

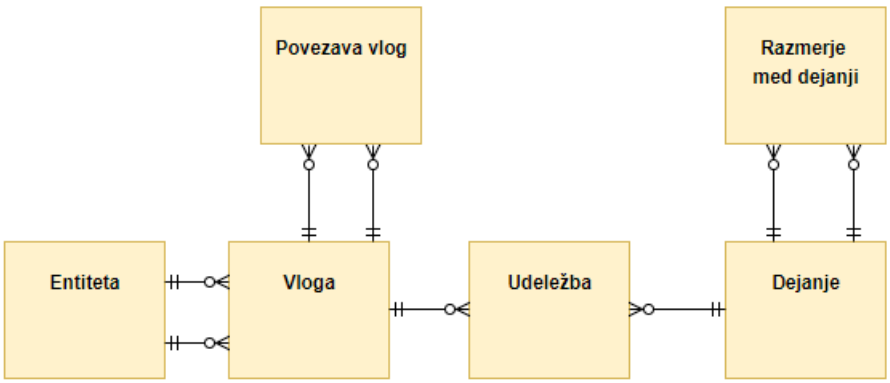
ki temelji na metodologiji modela in izdeluje sporočila in elektronske zapise, izražene v sintaksi XML. Cilj različice 3 je podpreti vse zdravstvene tokove [13].

3.4.1 Arhitektura HL7 verzije 3

Referenčni informacijski model

Referenčni informacijski model (RIM) je temelj razvojnega procesa HL7 verzije 3 in bistveni del razvojne metodologije HL7 verzije 3. RIM izraža vsebino podatkov, ki je potrebna v določenem zdravstvenem ali administrativnem kontekstu in zagotavlja eksplicitno predstavitev povezav, ki obstajajo med informacijami, ki jih prenašajo polja sporočil HL7 [18]. Slika 3.4 prikazuje diagram razredov v RIM. Referenčni model opisuje šest glavnih razredov za objekte zdravstvene domene kot tudi povezave med njimi: [18]

- entitete (angl. *Entities*) – fizične informacije objekta ali akterja v določeni domeni (organizacija, ljudje, material);
- vloge (angl. *Roles*) – definira vloge v procesu (pacient, zdravnik, zaposleni);
- udeležbe (angl. *Participations*) – to so udeležbe entitet, ki igrajo vloge v posebnih dejanjih (npr. izvajalec, avtor);
- dejanja (angl. *Acts*) – akcije določene entitete (npr. opazovanje, postopek);
- povezave med vlogami (angl. *Role links*) – upravljanje odnosov med entitetami glede na njihove vloge;
- razmerja med dejanji (angl. *Act relationships*) – razmerje med dvema dejanjema.



Slika 3.4: Diagram razredov RIM [28]

Sporočilo HL7 verzije 3

Ker gre tudi pri standardu HL7 za sporočilni standard, bomo na kratko opisali samo strukturo sporočila. Sporočilo v standardu HL7 verzije 3 temelji na referenčnem informacijskem modelu RIM, ki opredeljuje vsebino podatkov, potrebnih pri specifičnih zdravstvenih in administrativnih postopkih [14]. Sporočila v standardu HL7 verzije 3 so definirana s pomočjo definicije XSD (angl. *XML schema definition*). V nadaljevanju si bomo ob primeru standardnega sporočila ogledali strukturo sporočila HL7 verzije 3 [14]. Sporočilo je v grobem sestavljeno iz treh delov, to so:

- koren,
- dejanje in
- telo.

Koren sporočila (angl. *Root element*) prenaša informacije o samem sporočilu. Vsebina v korenu sporočila definira identifikator sporočila, čas ustvarjanja, vrsto sporočila, sistema, med katerima se sporočilo pošilja, ter dejanje, ki se pošilja v sporočilu. Struktura korena sporočila je prikazana v kodi 3.1.

```
<POLB_IN224200 ITSVersion="XML_1.0" xmlns="urn:hl7-org:v3"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <id root="2.16.840.1.113883.19.1122.7" extension="CNTRL-3456"/>
  <creationTime value="200202150930-0400"/>
  <versionCode code="2006-05"/>
  <interactionId root="2.16.840.1.113883.1.6" extension="POLB_IN224200"/>
  <processingCode code="P"/>
  <processingModeCode nullFlavor="OTH"/>
  <acceptAckCode code="ER"/>
  <receiver typeCode="RCV">
    <device classCode="DEV" determinerCode="INSTANCE">
      <id extension="GHH LAB" root="2.16.840.1.113883.19.1122.1"/>
      <asLocatedEntity classCode="LOCE">
        <location classCode="PLC" determinerCode="INSTANCE">
          <id root="2.16.840.1.113883.19.1122.2" extension="ELAB-3"/>
        </location>
      </asLocatedEntity>
    </device>
  </receiver>
  <sender typeCode="SND">
    <device classCode="DEV" determinerCode="INSTANCE">
      <id root="2.16.840.1.113883.19.1122.1" extension="GHH OE"/>
      <asLocatedEntity classCode="LOCE">
        <location classCode="PLC" determinerCode="INSTANCE">
          <id root="2.16.840.1.113883.19.1122.2" extension="BLDG24"/>
        </location>
      </asLocatedEntity>
    </device>
  </sender>
  <controlActProcess>
    ...
  </controlActProcess>
</POLB_IN224200>
```

Koda 3.1: Koren sporočila

Dejanje sporočila (angl. *Control act)* identificira dejanje, ki se prenaša v sporočilu in zagotavlja dodatne informacije, povezane z njim. To vključuje izvajalca dejanja, prejemnike, katerim naj se pošljejo informacije, ter vsebino domene, ki je povezana s tem dejanjem. Glede na tip sporočila so lahko prisotna še druga dodatna polja, ki pa jih ne bomo podrobneje opisovali. Dejanje sporočila je prikazano v kodi 3.2.

```
<controlActProcess moodCode="RQO|EVN">
  <id root=' ' extension=' '/>
  <code code='QUPC_TE043100UV'/>
  <effectiveTime value=' '/>
  <languageCode code=' '/>
  <authorOrPerformer typeCode=' '/>
  <informationRecipient typeCode=' '/>
  <subject>
      ...
  </subject>
</controlActProcess>
```

Koda 3.2: Dejanje sporočila

Telo sporočila (angl. *Message body)* vsebuje vsebino domene, ki se prične z lastnim korenskim elementom – dejanjem. V tem primeru gre za opazovanje (angl. *Observation*). Elementi znotraj določajo vrsto opazovanja, identifikator, čas opazovanja, status in preostale atribute za to dejanje. V svojih sekcijah pa so vidni še podatki o dejanju, pacientu in izvajalcu dejanja. Telo sporočila je prikazano v kodi 3.3.

```
<Observation>
  <!-- This is the Observation event at the root of the message -->
  <!--- ID is the filler order number ... -->
  <id root="2.16.840.1.113883.1122" extension="1045813"/>
  <code code="1554-5" codeSystemName="LN" displayName="GLUCOSE^POST 12H
    CFST:MCNC:PT:SER/PLAS:QN"/>
  <statusCode code="completed"/>
  <!-- time the sample was taken -->
  <effectiveTime>
    <center value="20020215073000"/>
  </effectiveTime>
  <!-- time of the actual lab test -->
  <activityTime>
    <center value="20020215083000"/>
  </activityTime>
  <priorityCode code="R"/>
  <value xsi:type="PQ" value="182" unit="mg/dL"/>
  <interpretationCode code="H"/>
  <author>
    ...
  </author>
  <recordTarget>
    ...
  </recordTarget>
  <inFulfillmentOf>
    ...
  </inFulfillmentOf>
  <referenceRange>
    <referenceInterpretationRange>
      <value xsi:type="IVL_PQ">
        <low value="70" unit="mg/dL"/>
        <high value="105" unit="mg/dL"/>
      </value>
    </referenceInterpretationRange>
  </referenceRange>
</Observation>
```

Koda 3.3: Telo sporočila

3.4.2 Prednosti

Standard HL7 verzije 3 je že bolj "pravi" standard in ne dopušča toliko odprtosti kot njegov predhodnik HL7 verzije 2. Odprtost standarda HL7 verzije 2 je bila sprva prednost in vzrok za njegovo uveljavitev, vendar je standard dopuščal preveč fleksibilnosti in opcijskih polj, kar je lahko povzročilo ne-

skladja s stališča medobratovalnosti. Prav zato standard HL7 verzije 3 ne dopušča tolikšne odprtosti in fleksibilnosti in s tem rešuje omenjeni problem. Standard temelji na referenčnem informacijskem modelu RIM, ki zagotavlja konsistentnost skozi celoten standard. Standard vsebuje najboljše prakse razvoja programske opreme. Sporočila so v sodobnejšem XML formatu, s čimer postane tudi sporočilo bolj pregledno in lažje za interpretacijo. Standard je rezultat več kot desetletja dela strokovnjakov na tem področju.

3.4.3 Slabosti

Standard ni združljiv z različico HL7 verzije 2. Uporaba napram standardu HL7 verzije 2 je časovno zahtevnejša in dražja. Prav tako je standard sam po sebi bolj kompleksen kot njegov predhodnik in ima manj implementacij v praksi. Standard je močno kritizirala panoga, ker je bil v svoji lastni dokumentaciji preveč nekonsistenten, preveč kompleksen in ker je standard drag za implementacijo v realnih sistemih. Prav tako so standard okrivili kot vzrok k številnim neuspešnim implementacijam sistemov [3].

3.4.4 HL7 verzija 3 v praksi

Sporočila HL7 verzije 3 se v svetu še niso široko uveljavila. Strokovnjaki menijo, da bodo sporočila HL7 verzije 3 prej zastarala, kot pa dosegla visoko stopnjo uveljavljenosti. Razlog je predvsem uspeh sporočil HL7 verzije 2 v državah, ki niso želele sprememb. Drug razlog je to, da je standard bolj abstrakten in ni domač analitikom in programerjem. Standard je precej obširen in težje razumljiv ter ima veliko krivuljo učenja. Posledično je zaradi obširnosti in kompleksnosti standarda tudi sporočilo v formatu XML daljše in bolj razvlečeno, saj vsebuje veliko podatkov [2]. Tretji večji razlog pa je objava najnovejšega standarda FHIR, ki združuje dobre lastnosti obeh svojih predhodnikov. Standard FHIR bomo podrobneje opisali v nadaljevanju.

3.4.5 Kompatibilnost s platformo Salesforce

Standard HL7 verzije 3 uporablja sintakso XML in sodobnejše principe razvoja programske opreme, zato je s stališča uporabljenih tehnologij bolj skladen za implementacijo na platformi Salesforce, vendar ostajajo pomisleki glede kompleksnosti standarda.

Za transport sporočil potrebujemo spletne servise, s katerimi bomo zmožni sprejemati in pošiljati različne tipe sporočil med sistemi. Spletni servisi bodo morali na naši strani sprejemati različne tipe sporočil in iz njih razbrati pomen ter ustrezno vršiti določene akcije. Prav tako bodo morali biti spletni servisi zmožni generirati različne tipe XML sporočil in jih pošiljati v zunanje sisteme bodisi kot zahtevo bodisi kot odgovor na določeno sporočilo. Tako moramo v sistemu podpreti logiko, ki bo znala reagirati na določen tip sporočila ter definirati sporočila, ki jih bomo pošiljali iz sistema. Ker so sporočila v standardu HL7 verzije 3 zelo obširna in ker je standard arhitekturno kompleksen, bi tudi implementacija omenjenega standarda predstavljala visoko stopnjo zahtevnosti, kar posledično pomeni več dela ter večje stroške implementacije. Druga slaba stran pa je, da bi morali za podporo različnim tipom sporočil implementirati tudi veliko spletnih servisov, saj standard definira več tipov sporočil napram ostalim standardom.

3.4.6 Razlike med HL7v2 in HL7v3

HL7 verzija 3 ima bistveno bolj strogo strukturo kot verzija 2, zato je slednji bistveno lažji za implementacijo, saj je preprostejši in v strukturi dopušča opcijska polja, kar pa sicer ni dobro z vidika medobratovalnosti. Izbira primerne različice za implementacijo pomembno vpliva na samo implementacijo in kakovost izmenjave podatkov. HL7 verzije 2 je tipično cenejši in hitrejši za implementacijo, medtem ko je implementacija HL7 verzije 3 običajno daljša in zahtevnejša. Po drugi strani pa z bolj zapleteno izmenjavo podatkov HL7 verzije 3 uvaja referenčni informacijski model, ki obravnava pomanjkljivosti iz standarda HL7 verzije 2 [20].

3.5 FHIR (Fast Healthcare Interoperability Resources)

FHIR je osnutek standarda, ki opisuje podatkovne strukture in elemente ter aplikacijske vmesnike za izmenjavo elektronskih zdravstvenih zapisov [1]. Standard je ustvarila organizacija HL7 in združuje najboljše lastnosti predhodnikov iz družine HL7 hkrati pa izkorišča najnovejše spletne standarde in se osredotoča na implementacijo. Rešitve FHIR so zgrajene iz sklopa modularnih komponent, imenovanih viri (angl. *Resources*). Te vire je mogoče zlahka združiti v sisteme. Viri temeljijo na enostavnih strukturah XML ali JSON, ki se prenašajo s protokolom RESTful na osnovi protokola HTTP. Tudi tehnično je standard oblikovan za splet. FHIR je primeren za uporabo v različnih kontekstih uporabe – mobilnih aplikacijah, računalništvu v oblaku, izmenjavi podatkov na podlagi elektronskih zdravstvenih zapisov, komunikaciji med strežniki v večjih zdravstvenih centrih in še veliko več [10]. Potreba po FHIR se je zvišala, saj je standard HL7 verzije 3 zapleten in potrebuje več časa za razvoj, medtem ko je standard HL7 verzije 2 star in ni združljiv z novimi tehnologijami, kot so mobilne aplikacije in oblačne storitve [27] [13].

3.5.1 Arhitektura FHIR

Standard FHIR vsebuje dve primarni komponenti, in sicer: [8]

- viri (angl. *Resources*) – so zbirka informacijskih modelov, ki opredeljujejo podatke, omejitve in razmerja najbolj relevantnih zdravstvenih "poslovnih modelov". Viri so predstavljeni v formatu XML ali JSON, trenutno pa obstaja 109 različnih vrst virov, ki so opredeljeni v specifikaciji FHIR.

- aplikacijski vmesniki (angl. *Application interfaces*) – so zbirka natančno opredeljenih vmesnikov za zagotavljanje medobratovalnosti med

dvema aplikacijama. Čeprav ni potrebna, je specifikacija FHIR namenjena RESTful vmesnikom za izvajanje aplikacijskih vmesnikov.

Na sliki 3.5 je viden primer sporočila v standardu FHIR, ki temelji na strukturi vira in je sestavljen iz treh večjih sklopov:

- razširitev z URL do definicije (oranžno),
- povzetek sporočila v formatu HTML (sivo),
- standardno definirana vsebina podatkov (zeleno).

```
<Patient xmlns="http://hl7.org/fhir">
  <extension url="http://www.goodhealth.org/consent#trials">
    <valueCode value="renal"/>
  </extension>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456</p>
    </div>
  </text>
  <identifier>
    <use value="usual"/>
    <label value="MRN"/>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender>
    <text value="Male"/>
  </gender>
  <birthDate value="1932-09-24"/>
  <managingOrganization>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </managingOrganization>
  <active value="true"/>
</Patient>
```

Slika 3.5: Primer sporočila FHIR [31]

3.5.2 Prednosti FHIR

Standard FHIR ima močan poudarek na implementaciji. Je hiter in enostaven za implementacijo, kar se kaže v tem, da je več razvijalcev v samo enem dnevu razvilo preproste vmesnike za komunikacijo med sistemi. Na voljo je veliko implementiranih knjižnic in primerov za začetek razvoja. Specifikacija je brezplačna za uporabo in brez omejitev. Prednost standarda so tudi prilagoditve v okviru razširitev, kar pomeni, da lahko zbirke virov uporabimo takšne, kot so, kljub temu pa jih je mogoče prilagoditi. Viri vsebujejo človeško čitljiv format zapisa za lažjo uporabo s strani razvijalcev. Standard ima močne temelje na standardih, kot so: XML, JSON, HTTP ter OAuth, ter ima podporo za RESTful arhitekture. Specifikacija je jedrnata in lahko razumljiva.

3.5.3 Slabosti FHIR

Standard je arhitekturno osredotočen na vire, kar omogoča enostavno implementacijo osnovnih entitet, vključenih v zdravstveno varstvo. Vendar pa je malo smernic, kako se ti osnovni viri zgradijo v večje zbirke in njihove odnose, kar lahko postane območje razhajanja med sistemi, ki vodi do pomanjkanja medobratovalnosti [3]. Standard je še dokaj nov in je zaenkrat izdan kot osnutek standarda za preizkusno uporabo – trenutno je na voljo tretja izdaja.

3.5.4 Fleksibilnost FHIR

Glavni izziv za standarde zdravstvenega varstva je, kako ravnati s spremenljivostjo podatkov, ki jih povzroča spreminjanje določenih procesov. Skozi čas je lahko k specifikaciji dodanih več dodatnih polj, s čimer se povečuje kompleksnost implementacije. Alternativa se opira na razširitve po meri, vendar le te ustvarijo še dodatne implementacijske probleme. FHIR rešuje ta izziv tako, da definira preprost način za razširitve in prilagajanje obstoječih virov. Vsi sistemi, ne glede na to, kako so bili razviti, lahko razširitve iz vira eno-

stavno preberejo in jih uporabijo. Vsaka razširitev vira vsebuje tudi URL z definicijo razširitve. Poleg tega vsak vir vsebuje človeško berljivo predstavitev besedila z uporabo HTML kot alternativnega prikaza. To je še posebej pomembno pri zapletenih zdravstvenih podatkih [10].

3.5.5 Združljivost s platformo Salesforce

Ker platforma Salesforce temelji na sodobnih spletnih tehnologijah, se zdi, da je FHIR najbolj primeren standard za izmenjavo zdravstvenih podatkov med platformo in drugimi zdravstvenimi sistemi. Standard se poslužuje novejših tehnologij, kot sta formata sporočil JSON in XML, ter uporabo RESTful spletnih storitev, kar se dobro sklada z arhitekturo omenjene platforme. Prav tako se operacije vršijo preko spletnih servisov, ki se jih na platformi realizira v razmeroma kratkem času. FHIR želi razvijalcem omogočiti prav to, da zgradijo spletne aplikacije, ki omogočajo dostop do podatkov ne glede na to, kateri zdravstveni sistem stoji na uporabnikovi strani. Za analizo podpore platforme Salesforce standardu FHIR smo v okviru diplomskega dela realizirali tudi prototip, ki je predstavljen v petem poglavju.

3.6 Primerjava standardov

Standard	Prednosti	Slabosti
OpenEHR	Modeli so ločeni od sistema, enostavno prilagajanje novim zahtevam, boljša podpora pri računanju z zdravstvenimi podatki napram drugim sistemom.	Manj primerov implementacije v praksi, manjša skupnost strokovnjakov, kompleksnost zaradi uporabe arhetipov.
HL7 v2	Prilagodljivost standarda, visoka stopnja uporabe standarda v svetu, združljivost s predhodnimi verzijami.	Nekonstistenti tipi sporočil, format sporočila je zastarel, pomanjkanje metodologij za modeliranje podatkov.
HL7 v3	Temelji na referenčnem informacijskem modelu, ki zagotavlja konsistentnost skozi celoten standard, sporočila so v sodobnejšem XML formatu.	Ni združljiv z različico HL7 verzije 2, standard je sam po sebi bolj kompleksen in ima manj implementacij v praksi.
FHIR	Hiter in enostaven za implementacijo, močni temelji na spletnih standardih, podpora RESTful arhitetkture.	Malo smernic, kako graditi vire v večje zbirke, standard izdan v preizkusni različici.

Tabela 3.1: Primerjava standardov

Poglavje 4

Platforma Salesforce

Salesforce je oblačna programska rešitev za sisteme za upravljanje odnosov s strankami (CRM), za prodajo, storitve, trženje, analitiko in izdelavo prilagojenih mobilnih aplikacij. Salesforce velja po Gartnerjevi analizi za trenutno najboljši sistem CRM na tržišču [11].

Slika 4.1: Magični kvadrant sistemov CRM [21]

4.1 Tehnologije

4.1.1 Apex

Apex je programski jezik, ki ga platforma Force.com ponuja razvijalcem. Force.com je platforma tipa "Platforma kot storitev" *(PaaS – Platform-as-a-Service)*, gre za platformo, ki ponuja storitve računalništva v oblaku in strankam omogoča razvijanje, upravljanje in izvajanje aplikacij brez zapletenosti gradnje in vzdrževanja infrastrukture. Platforma Force.com je prav tako ponujena s strani podjetja Salesforce in omogoča izdelavo lastnih aplikacij in prilagajanje standardnih. Sintaksa jezika Apex je zelo podobna programskima jezikoma Java in C#. Apex se lahko uporablja za izvedbo programski funkcij med večino procesov na platformi Force.com. Pobuda za izvedbo Apex kode lahko pride s strani spletnih storitev in prožilcev na objektih.

4.1.2 Visualforce

Visualforce je tehnologija za nadzor pogleda (angl. *View*). Je zelo podobna HTML, hkrati pa omogoča, da lahko z uporabo Visualforce značk bistveno lažje in z manj kode implementiramo funkcionalnosti na uporabniškem vmesniku. Koda, napisana v tehnologiji Visualforce, se kasneje prevede v dejansko HTML kodo. Za implementacijo Visualforce strani lahko uporabljamo tako Visualforce kot HTML značke.

4.1.3 Lightning

Leta 2014 je Salesforce ponudil svoje ogrodje za razvoj uporabniškega vmesnika, imenovan Lightning. Ogrodje je namenjeno grajenju uporabniškega dela programske opreme (angl. *frontend*) in temelji na komponentni zasnovi. Lightning olajša izdelavo odzivnih aplikacij za vsako napravo. Z uporabo ogrodja Lightning si olajšamo prilagajanje in razvijanje aplikacij, prav tako pa lahko razvite komponente ponovno uporabljamo. Dejansko sta mobilna

aplikacija Salesforce in Salesforce Lightning Experience zgrajeni z elementi Lightning.

4.2 Podpora Salesforca na področju zdravstva

Na področju zdravstva je Salesforce pred dvema letoma predstavil svoj produkt *Salesforce Health Cloud*. Produkt je še relativno mlad in v praksi še ne širše uporabljen. Izvajalcem zdravstvenega varstva omogoča, da dobijo boljši pogled na pacienta, sprejmejo boljše in pametnejše odločitve in lažje določijo oskrbo pacienta.

4.2.1 Salesforce Health Cloud

Salesforce Health Cloud je paket, nameščen na platformo Salesforce. Paket se namesti preko Salesforcove trgovine AppExchange. Z namestitvijo paketa na platformo je Salesforce Health Cloud pripravljen za uporabo. Kar pravzaprav pridobimo z nameščenim paketom, je podatkovni model sistema s pripadajočimi standardnimi funkcionalnostmi, kar vključuje delo z definiranimi objekti ter standardne maske za vnos in urejanje podatkov. Kar velja izpostaviti, je to, da je podatkovni model v produktu Salesforce Health Cloud namenoma modeliran tako, da je zelo podoben sporočilom standarda FHIR. To nam omogoča bolj preprosto in lažje razumljivo izmenjavo zdravstvenih podatkov z drugimi sistemi, saj moramo ob izmenjavi podatkov le te preslikati iz sporočila FHIR v ustrezne objekte na platformi. Poleg tega lahko še vedno dodajamo atribute in povezave na že definirane objekte.

Vsekakor je za implementacijo sistema dobro uporabiti omenjeni produkt, saj nam ponuja že dobro definirane objekte in nam s tem olajša delo pri samem razvoju sistema sploh v primeru, če želimo za integracijo uporabiti standard FHIR. V kolikor bi pri načrtovanju sistema ugotovili, da gre za zelo specifično rešitev, ki ne bi imela dovolj skupnih točk s produktom Salesforce Health Cloud in bi potrebovali veliko prilagajanja standardnih mask in definicij objektov, potem bi bilo smiselneje izhajati iz klasičnega CRM

sistema. Licence zanj so cenejše, obe rešitvi pa temeljita na isti platformi, saj je produkt Salesforce Health Cloud dejansko zgrajen na platformi Salesforce. Razlika med njima je v tem, da s produktom Salesforce Health Cloud pridobimo podatkovni model za potrebe zdravstva in določene funkcionalnosti, ki pa jih lahko razvijemo tudi na klasični platformi Salesforce. Glede na velikost projekta ter število licenc, ki jih bomo v sistemu potrebovali, se je tukaj potrebno vprašati, koliko nas bo stal razvoj takšnega sistema brez uporabe produkta Salesforce Health Cloud oziroma koliko nas bi za to stale licence.

Poglavje 5

Analiza primernosti platforme Salesforce za razvoj aplikacij v zdravstvu

V tem poglavju bomo izvedli analizo primernosti platforme Salesforce za razvoj aplikacij v zdravstvu. Pregledali bomo ključne funkcionalnosti ter lastnosti, ki naj bi jih zagotavljali zdravstveni sistemi, ter jih primerjali s funkcionalnostmi in možnostmi, ki jih ponuja platforma Salesforce. Zanimalo nas bo, ali ima platforma orodja, s katerimi lahko zagotovimo te funkcionalnosti oziroma če so te funkcionalnosti že privzete na platformi. Ker smo tekom diplomskega dela spoznali pomembnost medobratovalnosti, bomo preverili tudi podporo platforme za izmenjavo elektronskih zdravstvenih zapisov z drugimi sistemi. S stališča medobratovalnosti želimo doseči, da bomo lahko v našo aplikacijo na standardni način pridobili podatke iz drugih sistemov ter da bo naša aplikacija zmožna ponuditi svoje podatke v standardnem formatu tudi drugim sistemom.

5.1 Ključne funkcionalnosti zdravstvenih sistemov

Za določitev ključnih funkcionalnosti, ki jih morajo omogočati zdravstveni sistemi, smo obravnavali več študij [5] [30], na podlagi teh pa smo ugotovili, da so so bili pri določitvi ključnih funkcionalnosti zdravstvenih sistemov upoštevani naslednji kriteriji [30]:

- Podpora zagotavljanju učinkovite oskrbe pacientov – učinkovitost oskrbe v tem kontekstu pomeni zagotavljanje zdravstvenih storitev tistim, ki bi od teh storitev lahko imeli koristi, hkrati pa se vzdrževati zagotavljanja storitev tistim, ki teh koristi ne bi imeli. V študiji je bilo ugotovljeno, da samo polovica Američanov prejme priporočeno zdravstveno oskrbo, ki je v skladu z vodilnimi smernicami [30].

- Olajšan nadzor nad kroničnimi boleznimi – več kot polovica bolnikov s kroničnimi boleznimi ima tri ali več ponudnikov zdravstvenih storitev in poroča, da pogosto prejmejo nasprotujoče si informacije od različnih ponudnikov. Poleg tega mnogi navajajo podvojene preiskave, zdravniki pa poročajo o težavah pri usklajevanju oskrbe takšnih bolnikov.

- Izboljšanje učinkovitosti – potrebno je najti metode za povečanje učinkovitosti zdravstvenih delavcev in zmanjšanje administrativnih stroškov in stroškov dela v zdravstvu. V številnih poklicih zdravstvenega varstva je težava pomanjkanje kadra, s čimer se vrši še dodaten pritisk na stalno izboljševanje postopkov zdravstvene oskrbe.

- Izboljšanje varnosti bolnikov – varnost pomeni preprečevanje škode za bolnike, povzročene zaradi nepravilne zdravstvene oskrbe.

Na podlagi kriterijev so bile oblikovane ključne funkcionalnosti [30], ki naj bi jih zagotavljal vsak elektronski zdravstveni sistem. Te so bile upoštevane tudi pri študiji, kjer so strokovnjaki ocenjevali funkcionalnosti elektronskih zdravstvenih sistemov, ki jih pretežno uporabljajo splošni zdravniki v Franciji [5]. Poleg teh so bile v študiji zajete še nekatere dodatne, vendar se bomo v sklopu analize osredotočili le na ključne. Identificirane ključne funkcionalnosti zdravstvenih sistemov so naslednje:

- razpoložljivost zdravstvenih podatkov in informacij,
- upravljanje z rezultati,
- upravljanje in vnos naročil,
- podpora odločitvam,
- podpora pacientom,
- podpora administrativnim procesom,
- poročanje in upravljanje podatkov o zdravju prebivalstva,
- elektronska komunikacija in povezljivost.

Funkcionalnosti so podrobneje opisane v podpoglavjih, ki sledijo, prav tako pa bomo podporo za vsako funkcionalnost analizirali tudi v okviru platforme Salesforce.

5.1.1 Razpoložljivost zdravstvenih podatkov in informacij

Čeprav to zares ni prava funkcionalnost, saj gre bolj za lastnost, pa je zelo pomembno, da sistem vsebuje določene podatke o pacientih. Zdravniki in drugi ponudniki oskrbe potrebujejo določene informacije, da se lahko na podlagi teh pravilno odločajo, vendar pa njihove potrebe po informacijah pogosto niso izpolnjene. To pomanjkanje informacij lahko vodi do manj

kakovostne in neučinkovite zdravstvene oskrbe. Če vzamemo za primer že izvedene laboratorijske rezultate, lahko ugotovimo, da ti znatno zmanjšajo število odvečnih testov in s tem ne le prihranimo na denarju, temveč tudi preprečimo nepotrebne preizkuse, ki so bili opravljeni že v preteklosti. Prav tako lahko izpostavimo pomembnost informacij o alergijah pri predpisovanju zdravil. Po drugi strani pa je pomembno opozoriti, da lahko preveč podatkov uporabnika odvrne od uporabe sistema, zato je zelo pomembno, da so uporabniški vmesniki zasnovani tako, da prikazujejo podatke, ki so v danem trenutku obravnave relevantni [30].

Razpoložjivost zdravstvenih podatkov in informacij – Salesforce

Salesforce ima dobro podporo za upravljanje in prikaz relevantnih podatkov o zapisih v sistemu. Prva dobra lastnost je, da Salesforce omogoča enostavno prilagajanje podatkovnega modela, kar pomeni, da lahko enostavno dodamo določene atribute ali nov objekt v obstoječi sistem. V primeru, da želimo dodati nov objekt, ga na platformi le ustvarimo, nanj pa določimo atribute, za katere določimo ustrezne podatkovne tipe. Prav tako lahko obstoječim objektom določimo dodatne atribute. Kar lahko opazimo kot prednosti, je to, da v tem primeru ni potrebno skrbeti, kako bodo podatki shranjeni v dejanski fizični podatkovni bazi, ampak za to poskrbi platforma sama na podlagi izbranega podatkovnega tipa. S tem, ko nam Salesforce omogoča enostavno prilagajanje, lahko dosežemo tudi večjo razpoložljivost podatkov, saj lahko glede na potrebe na enostaven način prilagodimo obstoječi model in si s tem zagotovimo dodatne informacije o določenem zapisu. Novi atributi, ki bodo razširili obstoječi objekt, pa bodo tudi dodani v vnosne maske ter prikaz zapisa. Platforma Salesforce za shranjevanje podatkov v ozadju uporablja podatkovno bazo Oracle, vendar za poizvedbe po podatkovni bazi uporabljamo jezik SOQL (*Salesforce Object Query Language*), ki je zelo podoben jeziku SQL (*Structured Query Language*), vendar je jezik SOQL specializiran za poizvedovanje po podatkih, ki jih hrani platforma Salesforce. SOQL je zelo močan v kontekstu razvoja na omenjeni platformi, saj

nam omogoča enostavnejše dostope do povezanih podatkov ter omogoča lažji in hitrejši razvoj. Ker smo v okviru funkcionalnosti omenjali prikazovanje relevantnih podatkov, velja izpostaviti, da Salesforce omogoča prilagajanje nabora podatkov, ki jih želimo prikazati ob pogledu določenega zapisa. To je bistvena prednost, saj za prilagajanje ni potreben poseg v kodo, ampak lahko na enostaven način v pogled zapisa dodamo dodatne atribute oziroma jih lahko poljubno umestimo na uporabniški vmesnik. Podobno velja za povezane objekte, saj lahko na zapisu prikažemo tudi tako imenovane povezane sezname objektov (angl. *Related lists*), ki vsebujejo informacije o zapisih, ki so povezani na prikazan zapis. Kot primer lahko vzamemo objekt pacient, ki vsebuje določene atribute (ime, priimek, naslov, čas kreiranja v sistemu itd.). Vse te atribute želimo prikazati, medtem ko podatka o času kreiranja pacienta ne želimo prikazati uporabniku in ga zato ne umestimo v prikaz zapisa. Povezani objekti so v tem primeru lahko obiski pacienta v ambulanti, ki jih spet lahko opcijsko prikažemo. Skratka Salesforce omogoča v sklopu uporabniškega vmesnika standardne funkcionalnosti, ki so zelo uporabne tudi za potrebe informacijskega sistema v zdravstvu in ne zahtevajo dodatnega dela s posegi v kodo, ampak lahko sistem prilagajamo z le nekaj kliki na platformi.

5.1.2 Upravljanje z rezultati

Pri upravljanju rezultatov smatramo kot rezultat vse vrste izvidov in testov. Elektronska hramba takšnih rezultatov ima več prednosti napram rezultatom, zapisanim na papirju, saj se s tem bistveno izboljša kvaliteta oskrbe. Digitalni rezultati so lažje dostopni, do njih lahko dostopamo kjerkoli in kadarkoli. S tem tudi omogočimo hitrejše prepoznavanje in zdravljenje zdravstvenih težav. Poleg tega prikaz prejšnjih rezultatov testa omogoča zmanjšanje odvečnih in dodatnih testov, s čimer se ne samo izboljša učinkovitost zdravljenja, ampak tudi zmanjšuje stroške. Elektronski rezultati omogočajo boljšo interpretacijo in lažje odkrivanje nepravilnosti [30].

Upravljanje z rezultati – Salesforce

V okviru platforme Salesforce lahko enostavno upravljamo z rezultati raznih preiskav. Pomembno je, da jih smiselno povežemo na ustrezne zapise. Salesforce nam omogoča hranjenje raznovrstnih datotek, ki jih lahko pripnemo zapisom. S tem dosežemo to, da lahko za vsak zapis vidimo datoteke ali razne druge preiskave, povezane z njim. Kar v tem kontekstu ni namen platforme Salesforce, pa je beleženje velikih količin transakcij, kjer bi dnevno nastajalo ogromno podatkov v sklopu raznih meritev. V primeru, da bi želeli spremljati srčni utrip za določenega pacienta, bi bilo smiselneje podatke shranjevati na zunanjo storitev in ob zahtevi podatke prikazati na platformi, saj bi s tem manj obremenili sistem. Težavo bi predstavljalo veliko takšnih spremljanj, saj Salesforce v osnovi ni namenjen transakcijskim funkcionalnostim v smislu zelo pogostega vzorčenja podatkov. Salesforce je torej primeren za upravljanje z rezultati in je le pogojno primeren za spremljanje velikih količin vzorčenih podatkov.

5.1.3 Upravljanje in vnos naročil

Prednosti računalniškega vnosa naročil, kot je na primer predpisovanje zdravil, je, da imamo vse recepte evidentirane v sistemu. Z vnosom naročil v sistem se izognemo napakam, ki se lahko zgodijo zaradi nečitljivega pisanja zdravnika, morebitnih izgub naročil ali podvajanja le teh. Ta funkcionalnost vključuje tudi uporabo validacij za napake pri vnosih odmerkov zdravil, pogostosti odmerjanja, preverjanje alergij ter medsebojnega delovanja zdravil. Pokazalo se je, da so relativno preprosti sistemi za upravljanje z naročili bistveno zmanjšali število napak pri omenjenih scenarijih [30].

Upravljanje in vnos naročil – Salesforce

Na platformi Salesforce lahko vnašamo razna naročila, pregledujemo zgodovino in se tako izognemo morebitnemu podvajanju le teh. Vsekakor pa so funkcionalnosti odvisne od implementacije sistema. Velja izpostaviti, da

nam Salesforce že sam po sebi omogoča, da za zapise vidimo, kdaj so bili nazadnje spremenjeni, kdo jih je spremenil, kdo jih je ustvaril ter omogoča obnovitev izbrisanih zapisov v roki petnajstih dni. Za določene zapise je mogoče spremljati tudi zgodovino sprememb.

Salesforce nas v tej funkcionalnosti ne omejuje, je pa res, da zaenkrat sam po sebi v sklopu zdravstva ne ponuja podpore, ki bi omejevala validacijo raznih naročil. Nam pa Salesforce zato omogoča, da sami definiramo validacijska pravila na objektih. Ta pravila kasneje onemogočajo ustvarjanje zapisa, v kolikor validacijski pogoji niso izpolnjeni. Pravila definiramo tako, da določimo pogoje, kdaj je lahko določen zapis ustvarjen. Pogoji so logični izrazi, v katerih se lahko sklicujemo na vrednosti zapisa, ki so vnešene s strani uporabnika. Kadar pogoji niso izpolnjeni, se uporabniku na maski prikaže opozorilo in tako zapisa ni mogoče shraniti. Prednost takšnih pravil je, da lahko brez posega v kodo definiramo in urejamo validacijska pravila na objektih. V našem primeru bi na zapis recept lahko definirali validacijsko pravilo, ki bi na podlagi izbranega predpisanega zdravila preverilo, če je količina odmerka v skladu z referenčnimi vrednostmi. Prav tako bi lahko poleg tega definirali še več drugih pravil. Lahko rečemo da je v tej funkcionalnosti Salesforce primeren, saj nam omogoča vnašanje raznih naročil obenem pa nam omogoča tudi definiranje validacijskih pravil na zapisih, v tem primeru validacijo naročil.

5.1.4 Podpora odločitvam

Sistemi računalniško podprtega odločanja so pokazali svojo učinkovitost pri izboljšanju uspešnosti pri številnih vidikih zdravstvenega varstva, vključno s preventivo, predpisovanjem zdravil, postavitve diagnoze in obvladovanjem ter odkrivanjem neželenih dogodkov in izbruhov bolezni. V dveh analizah je bilo ugotovljeno, da so opomniki in opozorila znatno izboljšali preventivne prakse na področjih, kot so cepljenja ter razni preventivni testi [5]. Obstaja tudi majhna, a vedno večja dokazna baza za učinkovitost takšnih sistemov na področju računalniško podprtega diagnosticiranja ter zdravljenja in upra-

vljanja bolezni [30].

Podpora odločitvam – Salesforce

Na podlagi podatkov, ki jih hranimo v sistemu, lahko na platformi Salesforce implementiramo tudi podporo odločitvam. To lahko podpremo z raznimi opomniki, filtriranimi pogledi zapisov na podlagi izbranih kriterijev, hkrati pa nam Salesforce omogoča tudi pošiljanje elektronskih sporočil pacientom na precej enostaven način, ki jih lahko izkoristimo za razna avtomatska obveščanja. V kontekstu podpore odločitvam bi omenili specifiko platforme Salesforce. To je v platformo vgrajena umetna inteligenca, imenovana Salesforce Einstein. Gre za funkcionalnosti, ki zagotavljajo razne napovedi, priporočila in avtomatizacije na podlagi podatkov, ki jih imamo v sistemu. V našem primeru so to elektronski zdravstveni zapisi. Funkcionalnost Salesforce Einstein v okviru zdravstva omogoča izračune ocen tveganja in tako identificira visoko rizične paciente. Prav tako na podlagi podatkov o pacientih ponuja nadzorne plošče s prikazanimi podatki o raznih tveganjih. Vse te funkcionalnosti avtomatsko pridobimo, v kolikor uporabljamo produkt Salesforce Health Cloud, v nasprotnem primeru pa nam Salesforce Einstein ponuja aplikacijske vmesnike, s katerimi si lahko imlementiramo svoje funkcionalnosti v okviru platforme.

5.1.5 Podpora pacientom

Izobraževanje bolnikov je pokazalo pomembno učinkovitost pri izboljšanju nadzora nad kroničnimi boleznimi. Več študij je pokazalo izvedljivost spremljanja meritev na domu s strani pacientov. V študiji je bilo na primer ugotovljeno, da je samopreizkus spirometrije pri bolnikih z astmo zagotavljal veljavne rezultate, primerljive s testi, ki so bili izvedeni pod nadzorom zdravnika [5]. Za izvedbo takšnih meritev in izobraževanj potrebujejo pacienti dostop do svojih zdravstvenih zapisov za zagotavljanje interaktivnega izobraževanja ter pomoč in beleženje rezultatov pri izvajanju domačega spremljanja in samotestiranja [30].

Podpora pacientom – Salesforce

Salesforce omogoča prijavo v sistem tudi pacientom, kar je pri tej funkcionalnosti praktično nujno, saj sicer ne moremo vzpostaviti interakcije s pacientom na njegovem domu. Ko so pacienti prijavljeni, jim lahko omogočimo dostop do podatkov, kar pomeni, da lahko pregledujejo razne rezultate ter ostale zabeležene zapise v sistemu, do katerih imajo dostop. S tem jim lahko omogočimo tudi izvajanje raznih meritev na domu ter ustrezno poročanje zdravnikom, seveda pa se tu odpre še veliko drugih možnosti. Dobra lastnost platforme je tudi ta, da Salesforce omogoča dostop do platforme tudi preko mobilne aplikacije, kar pomeni dostop kjerkoli in kadarkoli. Ker so zdravstveni podatki občutljivi, je pomembno, da imajo pacienti dostop samo do tistih podatkov, do katerih so upravičeni. Tudi to funkcionalnost Salesforce odlično rešuje, saj za uporabnike sistema definira različne vloge. Glede na vlogo pa se tako določi nivo pravic, ki jih pridobi uporabnik. Prav tako lahko dodatno ustvarimo tako imenovani nabor pravic (angl. *Permission set*), ki vsebuje skupek pravic, ki jih lahko dodelimo nekemu specifičnemu uporabniku ne glede na njegovo vlogo. Za vsak objekt je mogoče določiti nivo dostopa, prav tako pa se dostop določa tudi na nivoju atributov objekta. Dostop je lahko le bralni, lahko pa omogočimo tudi dostop za urejanje. S tem pridobimo veliko prednost, saj se lahko dostopnost podatkov enostavno ureja in ni potrebnih nobenih posegov v kodo. Glede izvajanja meritev na domu pa ponovno velja omeniti, da Salesforce sicer omogoča beleženje meritev, vendar se pojavi težava, kadar bi želeli meritve izvajati v zelo kratkih periodah in vsako meritev takoj prikazati na platformi. V tem primeru Salesforce ni najbolj primeren za shranjevanje takšnih podatkov, saj bi za več pacientov nastajalo ogromno zapisov. Za takšne količine podatkov bi bilo smiselneje uporabiti pomoč zunanje storitve.

5.1.6 Podpora administrativnim procesom

Pod podporo administrativnim procesom štejemo računalniško podprta orodja za izboljšanje učinkovitosti bolnišnic in klinik. Sistemi elektronskega načrtovanja bolnišničnih sprejemov in bolnišničnih ter ambulantnih postopkov ne samo povečujejo učinkovitost organizacij zdravstvenega varstva, temveč tudi zagotavljajo boljše in pravočasnejše storitve za bolnike [30].

Podpora administrativnim procesom – Salesforce

Z uporabo platforme Salesforce lahko v zdravstvu podpremo in vodimo tudi administrativne procese, kot so naročanja in razni drugi podobni postopki. Tukaj imamo veliko možnosti za implementacijo in prilagoditve. Kako želimo takšne procese podpreti, je le stvar implementacije sistema. Salesforce tudi v tej smeri omogoča dobro podporo, zato ne vidimo ovire, da na platformi ne bi mogli pokriti podobnih procesov tudi v zdravstvu.

5.1.7 Poročanje in upravljanje podatkov o zdravju prebivalstva

Institucije imajo trenutno več zahtev javnega in zasebnega sektorja za poročanje na državni in lokalni ravni za varnost in kakovost oskrbe pacientov ter zdravja prebivalstva. Poleg tega prizadevanja za izboljšanje kakovosti znotraj zdravstvenih organizacij zdravnikom omogočajo rutinsko poročanje o ključnih kazalnikih kakovosti (tako imenovane zdravstvene nadzorne plošče). Večino podatkov za ta poročila je treba izvleči iz podatkov raznih preiskav in pisnih dokumentov, zato je proces zbiranja teh podatkov intenziven in časovno zahteven. Prav iz teh razlogov večinoma v poročila tudi niso vključeni vsi podatki, ki so na razpolago. Poleg tega v poročilih prihaja do določene stopnje napak. Zdravstveni podati, ki so zbrani v sistemu, bi najverjetneje bistveno zmanjšali breme zbiranja podatkov, prav tako pa bi s tem dosegli večjo natančnost podatkov ter zmanjšali stroške [30] [5].

Poročanje in upravljanje podatkov o zdravju prebivalstva – Salesforce

Platforma Salesforce nam omogoča tudi poročanje. Poročila si lahko sestavimo po lastnih potrebah. Podatke, ki smo jih na kakršen koli način pridobili v sistem, lahko vključimo v poročila in nadzorne plošče, ki jih lahko prikazujemo na strani ali pa jih izvozimo iz sistema v različnih formatih. Podatke si lahko praktično ogledamo v nešteto kombinacijah. Poročanje je standardna funkcionalnost, ki nam jo ponuja platforma Salesforce, zato v ta namen ni potreben dodaten razvoj. Zato lahko tudi v tej funkcionalnosti rečemo, da jo Salesforce izpolnjuje.

5.1.8 Elektronska komunikacija in povezljivost

Učinkovito komuniciranje med člani skupine zdravstvenih delavcev, oskrbovalnimi partnerji in pacienti je ključnega pomena za zagotavljanje kakovostne zdravstvene oskrbe. Njegovo pomanjkanje lahko prispeva k nastanku neželenih dogodkov. Izboljšana komunikacija med različnimi sistemi lahko poveča varnost in kakovost oskrbe pacientov in izboljša nadzor javnega zdravja. Elektronska povezljivost zdravstvenih sistemov je bistvenega pomena, zlasti za bolnike s kroničnimi boleznimi, ki imajo značilno več ponudnikov zdravstvenih storitev, ki se morajo med seboj uskladiti za dober načrt oskrbe pacientov [30]. Podporo platforme Salesforce za zagotavljanje standardne komunikacije in povezljivosti bomo preverili v nadaljevanju z implementacijo prototipa.

5.2 Elektronska komunikacija in povezljivost – Salesforce

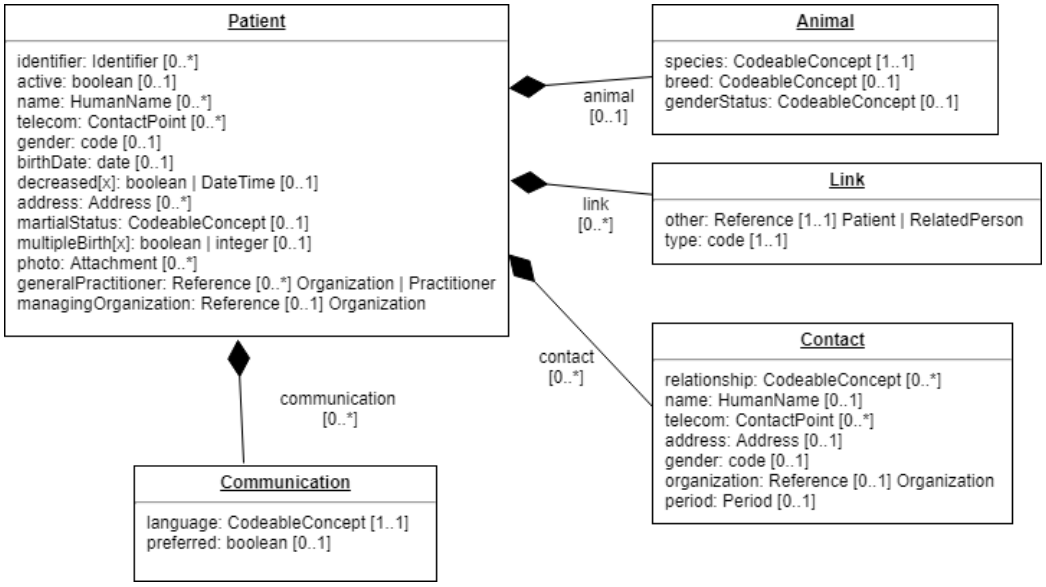
Ker smo tekom dela spoznali pomembno vlogo medobratovalnosti zdravstvenih sistemov ter ugotovili, da gre za eno ključnih funkcionalnosti, ki jo mora omogočati takšen sistem, bomo preverili podporo platforme Salesforce za

izmenjavo podatkov z drugimi sistemi. V prototipu si bomo podatke izmenjevali s pomočjo standarda FHIR, ki trenutno velja za enega najperspektivnejših standardov na področju elektronske izmenjave podatkov v zdravstvu in se dobro sklada z arhitekturo platforme Salesforce, saj je namenjen za splet z uporabo sodobnih spletnih tehnologij. V prototipu si bomo s strežnikom izmenjevali podatke o pacientih. Pacient bo torej naš vir (angl. *resource*). Preden nadaljujemo, velja ponovno razjasniti besedo vir. Vir je struktura, definirana v standardu FHIR, na kateri temelji sporočilo. Najprej bomo oblikovali podatkovni model, ki nam bo omogočal hranjenje podatkov o pacientih, nato pa bomo implementirali metode, ki bodo iz testnega strežnika pridobile podatke o pacientih v obliki standardnega sporočila FHIR in jih shranile v naš sistem. Prav tako bomo implementirali spletni servis, ki bo zmožen vračati podatke o pacientih, ki jih hranimo v našem sistemu. Tudi ta sporočila bodo morala biti vrnjena v standardnem formatu FHIR. Za potrebe implementacije prototipa bomo uporabili klasično platformo Salesforce, kjer bomo sami modelirali podatkovni model in s tem predstavili tudi enostavnost in prilagodljivost takšnega modeliranja. Kot smo že omenili, je produkt Salesforce Health Cloud le nadgradnja in prilagoditev klasične platforme Salesforce, osnovne funkcionalnosti pa se med njima ne razlikujejo, saj je produkt Salesforce Health Cloud "zgrajen" na klasični platformi.

5.2.1 Priprava podatkovnega modela v Salesforcu

Podatkovni model v naši aplikaciji bomo oblikovali tako, da bo vanj mogoče shraniti informacije, ki jih bomo pridobili v vsebini sporočila FHIR. Po drugi strani pa nam mora podatkovni model omogočati tudi delo znotraj same aplikacije. V ta namen lahko imamo na objektih različna dodatna polja, ki jih bomo uporabili za podporo določenim dodatnim funkcionalnostim v aplikaciji. FHIR za definicijo virov uporablja dve vrsti podatkovnih tipov, in sicer primitivne ter kompleksne podatkovne tipe, ki v sebi združujejo več različni kompleksnih in primitivnih podatkovnih tipov. Slika 5.1 prikazuje UML diagram vira pacient.

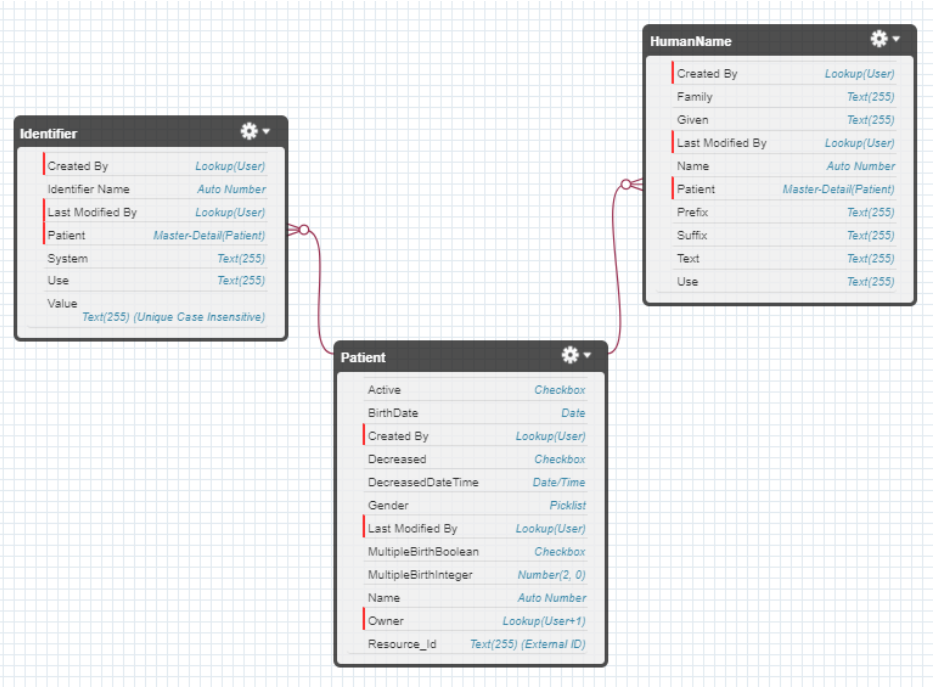
Kako bodo podatki shranjeni v aplikaciji, ne igra bistvene vloge, pomembno je le, da jih pravilno predstavimo v sporočilu in da jih znamo iz sporočila pravilno preslikati v podatkovni model aplikacije. Vsekakor pa nam podatkovni model, ki je podoben strukturam FHIR, omogoča lažje razumevanje in delo pri preslikavi podatkov v standardna sporočila.



Slika 5.1: UML diagram vira pacient s pripadajočimi atributi [9]

Pri oblikovanju podatkovnega modela bomo na platformi Salesforce ustvarili objekt *pacient*, ki bo vseboval atribute vira pacient, da bomo lahko kasneje podatke iz sporočila shranili v podatkovni model naše aplikacije. Na objektu bomo definirali vse obvezne podatkovne tipe. Ker vir pacient vsebuje več opcijskih podatkovnih tipov, ki jih sporočilo izrecno ne zahteva, bomo za implementacijo prototipa na strani platforme Salesforce shranili le nekaj ključnih podatkov. Kljub temu pa bomo imeli preostale podatke iz sporočila na razpolago za shranjevanje. V okviru prototipa bomo modelirali še dva dodatna objekta, ki bosta predstavljala predstavitev objekta pacient na strani aplikacije, to bosta identifikator (angl. *Identifier*) ter ime (angl. *HumanName*). Objektu pacient bo lahko tako pripadalo več imen ali iden-

tifikatorjev. S tem pa bomo posledično preverili tudi podporo za ustrezno povezovanje podatkov na strani platforme Salesforce.



Slika 5.2: Podatkovna shema v aplikaciji na platformi Salesforce

Na sliki 5.2 je prikazana podatkovna shema objekta pacient na strani naše aplikacije. Na sliki lahko vidimo tri objekte, ki so med sabo povezani in predstavljajo predstavitev objekta pacient v podatkovni bazi. Takšna struktura nam omogoča hranjenje več imen in identifikatorjev za določenega pacienta. Rdeče oznake na objektih predstavljajo polja, ki smo jih določili kot obvezna. Nekatera izmed polj so že sistemsko določena kot obvezna in jih ne moremo urejati, na primer čas zadnjega urejanja zapisa ali osebe, ki je zapis ustvarila.

5.2.2 Testni strežnik

Za izmenjavo zdravstvenih zapisov bomo uporabili testni strežnik FHIR, dostopen na naslovu *https://fhirtest.uhn.ca*. Gre za enega izmed testnih

strežnikov, ki jih organizacija HL7 priporoča za uporabo za potrebe testiranja.

5.2.3 Izmenjava podatkov – dostop do podatkov drugega sistema

Priprava razredov

Za pridobitev podatkov o pacientih smo na strani testnega strežnika uporabili spletno storitev RESTful, ki nam za podani vir vrne sporočilo v formatu JSON. Za začetek smo potrebovali informacijo o zgradbi sporočila, ki ga bomo prejeli s strani testnega strežnika, da bomo lahko na podlagi tega pripravili pomožne razrede v programskem jeziku Apex, v katere bomo razčlenili sporočilo. Ti razredi nam bodo kasneje omogočali enostavno delo s podatki, ki bodo vsebovani v sporočilu. Apex nam omogoča razčlenitev sporočila JSON v dejanski razred s pripadajočimi vrednostmi v sporočilu. Da lahko to funkcionalnost razčlenjevanja uporabimo, moramo najprej definirati razrede z atributi, ki bodo vsebovani v sporočilu. Da nam razreda ni potrebno definirati ročno, smo uporabili za to namensko orodje *JSON to Apex*, ki nam iz podanega JSON sporočila generira razrede Apex. V orodje smo kot vhod poslali sporočilo z vsemi možnimi atributi, ki jih lahko vir pacient vsebuje v sporočilu. S tem smo si pripravili razrede za razčlenitev sporočila in hranjenje podatkov za nadaljnje procesiranje. Torej če želimo uvoziti kakršenkoli vir v naš sistem, moramo definirati razred, v katerega se bo razčlenila vsebina sporočila. Za vsak tip sporočila pa moramo kasneje definirati še preslikavo, kako se bodo podatki iz sporočila shranili v naš sistem.

Pridobivanje podatkov o pacientih

Ko smo pripravili podatkovni model ter razrede za razčlenitev sporočila, smo pričeli z implementacijo metode, ki bo klicala testni strežnik in iz njega pridobila podatke o pacientih. Ustvarili smo nov razred Apex z imenom *ImportPatient*, ki vsebuje metodi *importPatient* ter metodo *storePatient*. Prva

metoda, ki je prikazana v kodi 5.1, pokliče spletno storitev testnega strežnika s podanim identifikatorjem pacienta. V odgovoru naša aplikacija prejme podatke o pacientu ter jih razčleni v pomožni razred s pripadajočimi vrednostmi v sporočilu. Pridobljene podatke je sedaj potrebno še preslikati v podatkovni model aplikacije. V ta namen smo uporabili drugo metodo *storePatient*, prikazano v kodi 5.2, ki kot argument prejme pomožni razred s pripadajočimi vrednostmi v sporočilu in iz njega ustvari ustrezne zapise v našem sistemu. Metoda tako poskrbi tudi za pravilno shranjevanje identifikatorjev ter imen in podatke med sabo ustrezno poveže in shrani v podatkovno bazo aplikacije.

```
1  public static void importPatient(String patientId) {
2      // Http zahtevek
3      Http http = new Http();
4      HttpRequest request = new HttpRequest();
5      request.setEndpoint(SERVER_BASE_URL + patientId + FORMAT_JSON);
6      request.setMethod("GET");
7      HttpResponse response = http.send(request);
8      // Odgovor in shranjevanje zapisa za izbranega pacienta
9      if (response.getStatusCode() == 200) {
10         patient pat = patient.parse(response.getBody());
11         storePatient(pat);
12     } else {
13         throw new CustomException("importPatient status: " + response.getStatusCode());
14     }
15 }
```

Koda 5.1: Metoda za uvoz podatkov o pacientu

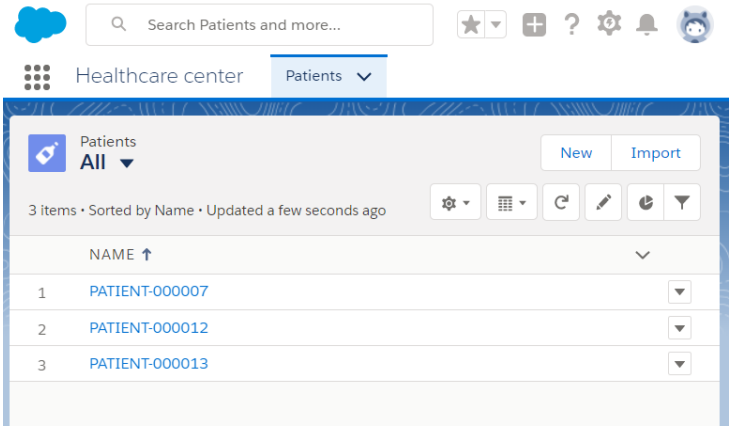
Metoda *importPatient* v prvem delu najprej ustvari zahtevek HTTP ter ga pošlje na naslov testnega strežnika. Pošlje se zahtevek *GET*, v naslovu URL pa prav tako določimo parameter, ki strežniku pove, da želimo odgovor v formatu JSON. Prejeto sporočilo JSON razčlenimo v pomožni razred ter nato shranimo s pomočjo pomožne metode *storePatient*.

```
1  public static void storePatient(patient patientData) {
2      // Ustvarimo zapis za pacienta
3      Patient__c p = new Patient__c();
4      p.Resource_Id__c = patientData.id;
5      insert p;
```

```
6       // Inicializacija seznamov imen in identifikatorjev
7       List < HumanName__c > humanNames = new List < HumanName__c > ();
8       List < Identifier__c > identifiers = new List < Identifier__c > ();
9       // Mapiranje imen v zapis HumanName
10      if (patientData.name != null) {
11          for (patient.cls_name name: patientData.name) {
12              HumanName__c humanName = new HumanName__c();
13              humanName.Family__c = name.family;
14              if (name.given != null) {
15                  humanName.Given__c = name.given[0];
16              }
17              // Imenu nastavimo pacienta, ki mu pripada
18              humanName.Patient__c = p.Id;
19              humanNames.add(humanName);
20          }
21      }
22      // Mapiranje identifikatorjev
23      if (patientData.identifier != null) {
24          for (patient.cls_identifier identData: patientData.identifier) {
25              Identifier__c ident = new Identifier__c();
26              ident.Use__c = identData.use;
27              ident.System__c = identData.systemCustom;
28              ident.Value__c = identData.value;
29              ident.Patient__c = p.Id;
30              identifiers.add(ident);
31          }
32      }
33      // V podatkovno bazo vstavimo pridobljene podatke
34      insert humanNames;
35      insert identifiers;
36  }
```

Koda 5.2: Metoda za shranjevanje in preslikavo podatkov

Z izvršitvijo prejšnjih dveh metod smo podatke o pacientu shranili v našo aplikacijo. Zdaj lahko ta zapis na strani aplikacije urejamo ter uporabljamo za potrebe naše aplikacije. Na sliki 5.3 so prikazani zapisi pacientov, ki so bili pridobljeni v aplikacijo iz testnega strežnika, na sliki 5.4 pa so prikazane podrobnosti za posamezni zapis.

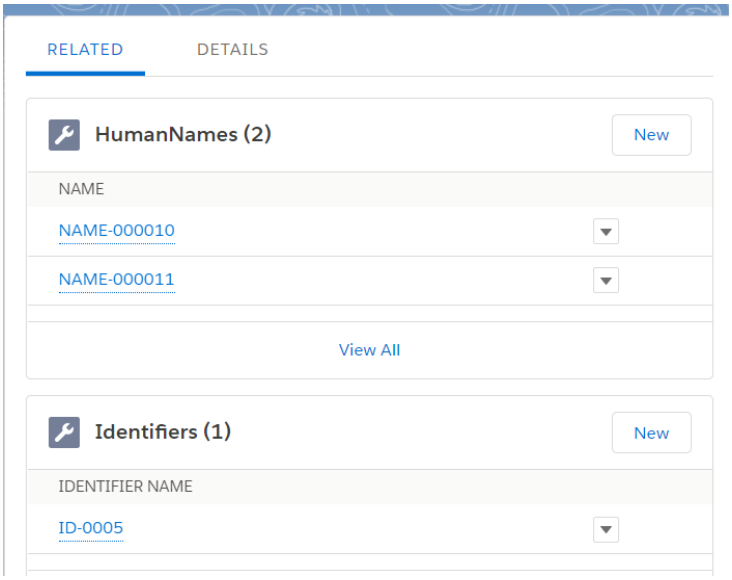


Slika 5.3: Pacienti, uvoženi iz testnega strežnika

5.2.4 Omogočanje dostopa do podatkov drugim sistemom

V naslednjem koraku smo omogočili dostopnost podatkov tudi drugim sistemom, ki bi želeli pridobiti podatke, ki jih hranimo v naši aplikaciji. Dostop smo omogočili s pomočjo spletne storitve RESTful. Spletno storitev implementiramo tako, da izbranemu razredu Apex pripišemo opombo "*@RestResource*" in ga tako izpostavimo kot spletno storitev. Prav tako je potrebno ustrezno opombo določiti tudi metodi, ki bo izvajala akcijo ob določeni zahtevi (GET, POST, PUT ...). Torej če želimo neko metodo izpostaviti preko zahteve GET, moramo metodi pripisati opombo "*@HttpGet*". Tukaj se pojavi tudi omejitev, da lahko znotraj enega razreda določeno opombo uporabimo le enkrat. V kolikor bi želeli implementirati dve metodi GET, bi ju morali ločiti v dva ločena razreda.

V našem primeru smo implementirali spletni servis za dostop do podatkov o pacientih. Spletni servis preko klica HTTP GET za podan identifikator pacienta (*resource Id*) vrne sporočilo v standardnem formatu FHIR v obliki sporočila JSON. Za implementacijo spletnega servisa smo uporabili že prej pripravljene pomožne razrede, le da smo tokrat objekte iz platforme preslikali v pomožni razred, tega pa smo serializirali v sporočilo JSON in ga vrnili v



Slika 5.4: Pacientu pripadajoča imena ter identifikatorji

odgovoru našega spletnega servisa. Atribute, ki niso vsebovali nobenih informacij, pa smo v sporočilu ignorirali in jih nismo prikazali v vsebini sporočila, tako kot to definira standard FHIR.

```
1  @RestResource(urlMapping = "/Patient/*")
2  global class ExportPatient {
3
4      @HttpGet
5      global static void show() {
6          // Pridobimo id pacienta
7          RestRequest req = RestContext.request;
8          String resourceNumber = req.requestURI.substring(req.requestURI.lastIndexOf("/") +
               1);
9          // Pridobimo objekt pacient iz podatkovne baze
10         Patient__c result = [SELECT Id, Name, Resource_Id__c FROM Patient__c
11             WHERE Resource_Id__c =: resourceNumber LIMIT 1
12         ];
13         // Preslikava v objekta v razred patient
14         patient mappedPatient = mappPatientForExport(result);
15         // Serializiramo razred in zamenjamo rezervirane besede v odgovoru
16         String serialized = JSON.serialize(mappedPatient, true);
17         serialized = serialized.replaceAll("systemCustom", "system");
18         // Vrnemo informacije o viru
```

```
19          RestContext.response.addHeader("Content-Type", "application/json");
20          RestContext.response.responseBody = Blob.valueOf(serialized);
21      }
22      ...
```

Koda 5.3: Spletni servis za pridobitev podatkov o pacientu

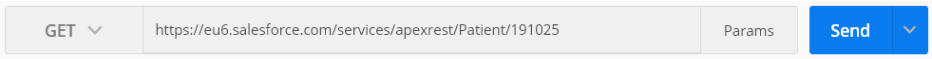
V spletnem servisu, ki je prikazan v kodi 5.3, najprej iz URL zahteve pridobimo identifikator pacienta (*resource Id*), za katerega je prišla zahteva v našo aplikacijo. Nato na podlagi tega v podatkovni bazi poiščemo ustreznega pacienta. V naslednjem koraku moramo podatke o pacientu preslikati v pomožni razred *patient*. Storimo enako kot pri uvozu, le da namesto da iz razreda podatke beremo, tokrat atribute razreda ustrezno nastavimo. Preslikava je prikazana v kodi 5.4. Ker so bile v atributih sporočila vsebovane tudi rezervirane besede v programskem jeziku Apex, smo te atribute preimenovali, zato jih moramo ustrezno preimenovati tudi, preden sporočilo vrnemo kot odgovor.

```
1  private static patient mappPatientForExport(Patient__c p) {
2      patient patientData = new patient();
3      // Nastavimo resource id
4      patientData.id = p.Resource_Id__c;
5      // Preslikava imen
6      patientData.name = mapHumanNames(p);
7      // Preslikava identifikatorjev
8      patientData.identifier = mapIdentifiers(p);
9      return patientData;
10 }
```

Koda 5.4: Metoda za preslikavo imen in identifikatorjev v pomožni razred patient

S tem ko smo implementirali spletni servis, smo omogočili dostop do podatkov naše aplikacije tudi drugim sistemom. Zanimalo nas je, če vsebina vrnjenega sporočila ustreza standardnemu formatu sporočil FHIR ter vsebuje informacije, ki jih hranimo v sistemu za določenega pacienta. Za simulacijo zdravstvenega sistema, ki poizveduje na našo spletno storitev, smo izvedli klic s pomočjo orodja Postman. Z orodjem Postman lahko izvedemo klic na neko

spletno storitev, v katerem določimo parametre klica. Klic spletne storitve je prikazan na sliki 5.5. Kot rezultat smo prejeli sporočilo JSON v formatu FHIR, ki je prikazan v kodi 5.5. Da smo se res prepričali, če je struktura sporočila pravilna, smo isto sporočilo prenesli na testni strežnik in na njem poskusili ustvariti nov zapis na podlagi istega sporočila. Testni strežnik je sporočilo sprejel brez napak in zapis ustvaril, iz česar lahko sklepamo, da je naša aplikacija vrnila sporočilo v pravilnem standardnem formatu FHIR.



Slika 5.5: Klic spletnega servisa z orodjem Postman

```
{
    "resourceType": "Patient",
    "name": [{
        "given": [
            "Charlie"
        ]
    }, {
        "family": "Smith"
    }],
    "identifier": [{
        "value": "12345",
        "system": "urn:system"
    }],
    "id": "191025"
}
```

Koda 5.5: Odgovor spletnega servisa v formatu JSON, ki je strukturiran po standardu FHIR

5.2.5 Rezultati izmenjave podatkov

V prvem delu implementacije prototipa smo uspešno modelirali podatkovni model z novimi objekti ter atributi, ki smo jih kasneje uporabili za shranjevanje podatkov o pacientih. Uspešno smo tudi pridobili podatke o pacientih

iz testnega strežnika FHIR ter jih shranili na stran aplikacije. Podatki so bili ustrezno shranjeni v podatkovni bazi kot zapisi objekta pacient s pripadajočimi atributi, imeni ter identifikatorji. V drugem delu, kjer smo podatke o pacientih izpostavili na voljo drugim sistemom, pa smo izvedli simulacijo klica zunanjega sistema na naš strežnik s pomočjo orodja Postman. S klicem na implementirani spletni servis smo podatke o pacientih prejeli v standardnem formatu FHIR v obliki JSON. V podatkih so bile upoštevane tudi spremembe na zapisih, ki so se pred tem zgodile v aplikaciji.

Po izvedenem testu smo želeli preveriti še obnašanje platforme ob dodajanju novega atributa na že obstoječi podatkovni model. V ta namen smo na objektu pacient dodali dodatni atribut, ki za določenega pacienta pove, ali je samoplačnik. Atribut je bil dodan le z nekaj kliki ter bil avtomatsko dodan na vnosno masko ter prikaz zapisa pacient. Kljub temu, da smo izvajali spremembe podatkovne baze, pa so naši spletni servisi še vedno nemoteno delovali, prav tako ni bilo potrebno skrbeti za spremembe, ki so se med tem dogajale v podatkovni bazi. Test izmenjave podatkov za vir pacient ocenjujemo kot uspešen, saj smo pokazali, da platforma Salesforce ponuja dobro podporo za implementacijo izmenjave podatkov med sistemi, prav tako pa ima dobro podporo za implementacijo različnih vrst spletnih storitev, pri čemer smo v našem primeru uporabili storitev REST. Izkazalo se je, da lahko na zelo hiter način zgradimo podatkovni model, prav tako pa lahko na že obstoječi model samo z nekaj kliki dodamo dodatni atribut brez kakršnega koli posega v kodo, kar nam lahko olajša razvoj sploh pri kasnejših dodelavah v primeru, da bi se izkazala potreba po dodatnem atributu. V našem primeru smo uvažali vir pacient. Za podporo izmenjave preostalih virov pa bi morali implementirati spletne storitve, ki bi bile zmožne sprejemati sporočila tudi za druge vire ter jih ustrezno preslikati v model aplikacije in obratno. Težava med implementacijo spletnih storitev se je pojavila pri preslikovanju med sporočili zaradi rezerviranih besed, ki jih ne moremo uporabiti kot spremenljivk v pomožnih razredih za razčlenjevanje, zato moramo v sporočilih pred razčlenjevanjem ustrezno zamenjati določene atribute z nadomestnimi spre-

menljivkami, enako pa moramo storiti tudi, preden sporočilo vrnemo drugim sistemom.

5.3 Alternativne možnosti izmenjave

V podpoglavjih, ki sledijo, bomo predstavili tudi alternativne možnosti izmenjave, ki jih je mogoče implementirati s platformo Salesforce. Platforma nudi svoja orodja, s katerimi se v kontekstu CRM sistema sicer tipično povezujemo na centralni sistem v podjetju (ERP), v našem primeru pa bomo govorili o povezavi na druge zdravstvene sisteme. Prav tako bomo preverili alternativne možnosti izmenjave podatkov s pomočjo vmesne programske opreme (angl. *middleware*), ki lahko reši problematiko s kompleksno obdelavo sporočil na strani platforme Salesforce.

5.3.1 Salesforce Data Loader

Salesforce Data Loader je odjemalska aplikacija za uvoz in izvoz velikih količin podatkov, ki se veliko uporablja za integracije v sistemih CRM. Uporabljamo jo lahko za vstavljanje, posodabljanje, brisanje ali izvoz podatkov platforme Salesforce. Data Loader lahko uporabljamo na več načinov. Podatke lahko prebere in naloži iz datotek v formatu CSV (gre za datoteko, v kateri so vrednosti ločene z vejico) ali pa podatke črpa neposredno iz povezave na podatkovno bazo nekega zunanjega sistema. Tako nam Data Loader po eni strani omogoča, da podatke programsko uvozimo v našo aplikacijo, po drugi strani pa nam omogoča tudi prenos podatkov med sistemi. V primeru da podatke med sistemi prenašamo s pomočjo aplikacije Data Loader, moramo aplikaciji podati konfiguracijsko datoteko, v kateri definiramo, katere podatke iz podatkovne baze želimo prenašati v naš sistem ter v katere objekte in polja se bodo ti podatki preslikali. Data Loader nam tako omogoča samodejno izmenjavo podatkov med platformo Salesforce in zunanjimi sistemi.

V kontekstu zdravstvenih sistemov nam Data Loader lahko nudi pomoč pri uvozu podatkov, ki jih še nimamo zabeleženih v sistemu, saj jih lahko z

njegovo pomočjo bistveno lažje prenašamo, kot pa bi to počeli ročno, sploh pri uvozih iz datotek. Za prenašanje podatkov med sistemi pa nam Data Loader ne omogoča dobre medobratovalnosti. Kot smo že uvodoma dejali, morajo biti sistemi zmožni medobratovalnega delovanja ne glede na njihov podatkovni model. V primeru, da bi za izmenjavo podatkov med sistemi uporabljali Data Loader, bi morali za vsak sistem določiti preslikave polj med objekti na platformi Salesforce in vnosi v podatkovni bazi zunanjega sistema, kar pa s stališča medobratovalnosti ni dobro, saj bi morali graditi neke vrste vmesnik za vsak nov sistem, s katerim bi si želeli izmenjevati podatke.

5.3.2 Podpora standardu HL7 verzije 2

Pokazali smo, da si lahko na platformi Salesforce izmenjujemo sporočila FHIR, kljub temu pa še vedno ostajajo pomisleki glede podpore starejšim standardom, ki jih še vedno podpira večina današnjih sistemov. Pri pregledu standardov smo dejali, da Salesforce v osnovi ni najbolj primeren za obdelavo sporočil HL7 verzije 2, saj bi imeli veliko dela z validiranjem in interpretiranjem prejetih sporočil, prav tako pa bi morali implementirati logiko za generiranje sporočil, ki bi jih pošiljali drugim sistemom. Takšno obdelavo bi bilo na platformi težje implementirati, saj ne bi mogli vključiti kakšne izmed zunanjih knjižnic, kar bi lahko naredili na nekem sistemu, ki bi deloval v programskem jeziku Java, saj za omenjeni jezik že obstajajo podobne knjižnice, medtem ko jih za programski jezik Apex zaenkrat še ni.

Kljub temu pa obstaja boljša rešitev za podporo HL7 verzije 2. Za podporo standarda bi bilo potrebno uporabiti sloj vmesne programske opreme, ki bi sprejemala sporočila HL7 verzije 2, jih razčlenila, iz njih razbrala pomen ter sprožila akcije na platformi Salesforce. Takšna programska oprema na trgu tudi že obstaja, kot je na primer produkt *MuleSoft*. Razlog, zakaj je to mogoče, je v tem, da nam Salesforce omogoča integracijo preko različnih aplikacijskih vmesnikov, zlasti preko SOAP in REST [29]. To nam omogoča programski dostop do informacij znotraj platforme z uporabo pre-

prostih, močnih in varnih aplikacijskih vmesnikov. Zaradi razpoložljivosti programskih vmesnikov na platformi Salesforce lahko MuleSoft dostopa do definicij objektov, podatkov in ima možnost ustvarjanja in urejanja zapisov. Zato lahko rečemo, da je Salesforce ena izmed preprostejših platform, ki jih je mogoče povezati. V orodju MuleSoft moramo tako za vsak tip sporočila določiti, katere akcije naj se prožijo na platformi, kar pa v praksi pomeni, da za vsak tip sporočila določimo akcijo oziroma preslikavo v objekt na platformi Salesforce. MuleSoft nam prav tako omogoča, da sporočila HL7 verzije 2 pretvorimo v format FHIR ter sporočilo posredujemo na platformo Salesforce. Definicijo za preslikavo določenega vira lahko najdemo na uradni spletni strani organizacije HL7.

5.4 Omejitve

V okviru razvoja na platformi Salesforce velja omeniti omejitve na transakcijah. Ker Apex deluje v večnamenskem okolju, uveljavlja določene meje, s čimer zagotovi, da določeni procesi oziroma koda Apex ne bodo monopolizirali skupnih virov. V kolikor koda Apex preseže določeno omejitev, se sproži izjema, ki je ni mogoče obravnavati. Omejitve so določene glede na različne vrste transakcij.

Pri razvoju na platformi Salesforce se je vedno potrebno zavedati omejitev. So pa omejitve vsekakor smernice, ki nas prisilijo v programiranje brez nepotrebne porabe dodatnih virov in ne predstavljajo ovire, da nekatere stvari ne bi bile izvedljive na platformi. Za transakcije, ki zahtevajo veliko procesne moči, Salesforce ponuja prav za to namenske funkcionalnosti.

5.5 Splošni rezultati analize

Na podlagi ključnih funkcionalnosti zdravstvenih sistemov, ki so bile identificirane s strani strokovnjakov, lahko rečemo, da te Salesforce v veliki meri pokriva. Izkazalo se je, da Salesforce ponuja učinkovita orodja za modeliranje

podatkovnega modela ter urejanje objektov brez potrebe poznavanje dejanske fizične sheme. Prav tako smo za vse objekte, ki smo jih vključili v podatkovni model, že pridobili standardne maske za vnos in urejanje podatkov. Salesforce ponuja tudi specializiran jezik za poizvedovanje po podatkovni bazi, ki temelji na zapisih, ki jih hrani platforma, s čimer nam omogoča hitre dostope do povezanih objektov. Salesforce ponuja tudi enostavno prilagajanje uporabniškega vmesnika brez posegov v kodo. Študije so pokazale, da kljub temu da je bilo v razvoj elektronskih zdravstvenih sistemov vloženega veliko dela, se izkaže, da zdravstveni delavci teh sistemov ne znajo uporabljati ali pa jih slabo razumejo [5]. Salesforce daje na področju uporabniške izkušnje smernice, kako graditi dobre in učinkovite uporabniške vmesnike. Prav tukaj lahko Salesforce z dobro uporabniško izkušnjo in izboljšano estetiko, ki jo prinaša tehnologija Lightning, ter enostavno prilagoditvijo uporabniškega vmesnika pripomore tudi k boljšemu razumevanju procesov. Mnogo podjetji je poročalo, da je z uporabo Lightning tehnologije bistveno izboljšalo produktivnost, sodelovanje in uspešnost poslovanja [7]. Poleg tega lahko na platformi Salesforce hranimo tudi razne rezultate, dokumente, naročila ter vodimo razne druge administrativne postopke. Prav tako nam Salesforce omogoča definiranje validacijskih pravil na objektih, kar omogoča validacijo vnosov pri predpisovanju zdravil ali podobnih naročil. Platforma Salesforce vključuje tudi modul Einstein, ki je namenjen analizi podatkov in pripomore k lažjim odločitvam in identificiranju rizičnih bolnikov. Salesforce nam omogoča tudi vključitev pacientov v proces zdravstvenih storitev. To jim lahko omogočimo preko spletne ali mobilne aplikacije in s tem še dodatno povečamo dostopnost njihovih podatkov in sodelovanje v procesih zdravstvenih storitev. Salesforce prav tako zelo dobro rešuje pravice dostopa do podatkov za specifičnega uporabnika. Na področju obdelave podatkov ter izdelave raznih poročil pa nam Salesforce ponuja tudi odlična orodja za generiranje poročil nad podatki, ki jih hranimo v sistemu.

Salesforce nam s stališča integracije med različnimi sistemi ponuja aplikacijske vmesnike, s katerimi lahko preberemo definicije objektov in urejamo

zapise znotraj platforme. S tem se nam odprejo večje možnosti za integracijo z drugimi sistemi, saj lahko podatke znotraj platforme urejamo preko standardnih aplikacijskih vmesnikov, kar pa pomeni, da je platformo dokaj enostavno povezati z drugimi sistemi, sploh če uporabljamo vmesni sloj programske opreme, ki zna glede na vhod pravilno vršiti operacije nad zapisi, ki jih hranimo v sistemu.

Ugotovili smo tudi, da ni namen platforme shranjevanje ogromnih količin podatkov, ki bi jih pridobili z vzorčenjem srčnega utripa ali podobnih vitalnih funkcij pacientov, v ta namen je za shranjevanje smiselneje uporabiti zunanje storitve in podatke kasneje na zahtevo pridobiti v sistem. Za to področje sama platforma ni najbolj primerna.

S pomočjo implementacije prototipa smo prikazali tudi dobro podporo platforme za implementacijo vmesnikov za izmenjavo podatkov, saj smo lahko zelo hitro vzpostavili povezavo s testnim strežnikom. Ovira, ki se je tukaj pojavila, so bile rezervirane besede spremenljivk v pomožnih razredih za razčlenjevanje, vendar to ni predstavljalo večje težave, saj smo jih pred razčlenjevanjem sporočila zamenjali z ustreznimi nadomestnimi imeni.

Kar pa smo tekom razvoja ugotovili, je tudi to, da je platforma na splošno zelo prilagodljiva, saj lahko zelo enostavno uredimo uporabniške vmesnike, dodajamo atribute in zelo hitro razvijamo aplikacije.

Ker platforma sama skrbi za infrastrukturni nivo, se lahko tekom implementacije takšnih sistemov bistveno bolj osredotočimo na funkcionalnosti samih aplikacij in s tem gradimo boljše in uporabnikom bolj prijazne aplikacije, saj lahko čas, ki bi ga namenili za postavitev in ukvarjanje z infrastrukturo, namenimo razvoju. Vendar pa se moramo pri razvoju vedno zavedati tudi omejitev, saj na takšni infrastrukturi nismo sami.

Poglavje 6

Sklepne ugotovitve

V diplomskem delu smo si najprej pogledali, kaj so to elektronski zdravstveni zapisi ter kakšne so dobre prakse pri implementaciji sistemov v zdravstvu. Spoznali smo, zakaj potrebujemo standarde ter da je ena izmed ključnih zahtev takšnih sistemov zmožnost medobratovalnega delovanja, ki ga zagotovimo z uporabo standardov. Pregledali smo vodilne standarde na področju razvoja zdravstvenih sistemov ter organizacije, ki se ukvarjajo z razvojem standardov. Nekateri izmed njih so predvsem zaradi tehnologij, z uporabo katerih lahko standarde implementiramo, ter njihove kompleksnosti in uporabnosti v realnem okolju za uporabo v okviru platforme Salesforce bolj, drugi pa manj primerni. Spoznali smo, da je standard FHIR standard nove generacije, ki uporablja sodobne spletne protokole in omogoča enostaven prenos standardno zapisanih podatkov med sistemi, zato smo ga identificirali kot najprimernejšega za uporabo na platformi. Standard HL7 verzije 2 definira sporočila v starejšem formatu, kar bi predstavljalo dodatno delo z razčlenjevanjem sporočila, prav tako pa pušča preveč odprtosti, kar posledično privede do pomanjkanja medobratovalnosti. Standard HL7 verzije 3 je bistveno bolj kompleksen glede na ostale, standard OpenEHR pa definira, kako zgraditi sistem na podlagi dvonivojskega pristopa, ki pa ga na platformi Salesforce ne moremo uporabiti, saj nimamo možnosti izbire tehničnih lastnosti platforme, ampak moramo uporabiti platformo takšno, kot je, kljub temu pa

lahko v procesu razvoja uporabimo določene koncepte standarda OpenEHR. V nadaljevanju smo izvedli analizo primernosti platforme Salesforce za razvoj aplikacij v zdravstvu na podlagi ključnih funkcionalnosti, ki jih morajo takšni sistemi zagotavljati. Ugotovili smo, da platforma Salesforce nudi dobro osnovo za implementacijo večine funkcionalnosti. Kljub temu pa smo ugotovili, da platforma ni najbolj primerna za shranjevanje transakcij, kjer bi nastajale ogromne količine zapisov, na primer spremljanje srčnega utripa v realnem času pri več pacientih. Prav tako pa smo ugotovili, da se je potrebno pri razvoju na omenjeni platformi zavedati tudi omejitev pri transakcijah, saj uporabljamo skupno infrastrukturo.

Ker delo temelji na uporabi vodilnih standardov za zagotavljanje medobratovalnega delovanja, smo prikazali tudi podporo platforme Salesforce za medobratovalno delovanje s pomočjo standarda FHIR. Razvili smo prototip aplikacije, ki je zmožna izmenjave podatkov z zunanjimi sistemi, v našem primeru s strežnikom FHIR. Najprej smo postavili podatkovni model, nato pa smo naredili poizvedbo na testni strežnik ter vrnjene podatke shranili v našo aplikacijo. Kasneje smo na strani aplikacije razvili tudi spletni servis, ki omogoča dostop do podatkov v standardnem formatu FHIR. S temi koraki smo dokazali, da platforma ponuja dobro podporo pri razvoju takšnih aplikacij, poleg tega pa omogoča tudi veliko drugih funkcionalnosti, kot so poročila, dostop do platforme preko mobilne aplikacije, podatkovno analitiko ter še mnogo več. Poleg tega platforma zagotavlja integracijo preko različnih aplikacijskih vmesnikov, kar nam omogoča programski dostop do informacij znotraj platforme, to pa pomeni, da lahko preko njih preberemo definicije objektov ter urejamo podatke znotraj platforme, zato je platformo tudi enostavneje povezati z drugimi sistemi. Ugotovili smo, da ima platforma dober potencial, na ta potencial pa kaže tudi to, da se je v zadnjih letih tudi Salesforce pričel osredotočati na področje zdravstva.

Literatura

[1] Kelly J. Abrams, Shirley Learmonth, and Candace J. Gibson. *The Canadian Health Information Management Lifecycle*. Lulu Publishing Services, 2017.

[2] Mojca Paulin in Saša Javorič Anka Bolka, Brane Leskošek. Primerjava standardov hl7 : openehr in priporočila za uveljavljanje standardov v zdravstveni informatiki v Sloveniji. *Odbor za zdravstveno informacijske standarde*, 2009.

[3] Duane Bender and Kamran Sartipi. Hl7 fhir: An agile and restful approach to healthcare information exchange. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*. IEEE, 2013.

[4] Pascal Coorevits, M Sundgren, Gunnar O Klein, A Bahr, B Claerhout, C Daniel, M Dugas, D Dupont, A Schmidt, P Singleton, et al. Electronic health records: new opportunities for clinical research. *Journal of internal medicine*, 2013.

[5] David Darmon, Rémy Sauvant, Pascal Staccini, and Laurent Letrilliart. Which functionalities are available in the electronic health record systems used by french general practitioners? an assessment study of 15 systems. *International journal of medical informatics*, 2014.

[6] Salesforce data model. Dosegljivo: https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/data_model.html, 2017. [Dostopano: 17. 12. 2017].

[7] Felicia Duarte and Rachelle Hoffman. Introducing salesforce lightning. In *Learn Salesforce Lightning*, pages 1–51. Springer, 2018.

[8] FHIR Overview - Architects. Dosegljivo: http://hl7.org/fhir/2016sep/overview-arch.html. [Dostopano: 17. 12. 2017].

[9] FHIR Resource Patient. Dosegljivo: http://www.hl7.org/fhir/patient.html, 2017. [Dostopano: 23. 12. 2017].

[10] FHIR Summary. Dosegljivo: http://hl7.org/fhir/summary.html. [Dostopano: 18. 11. 2017].

[11] The Magic Quadrant for the CRM Customer Engagement Center. Dosegljivo: https://www.salesforce.com/blog/2017/05/salesforce-gartner-crm-customer-engagement.html. [Dostopano: 17. 12. 2017].

[12] J Henry, Yuriy Pylypchuk, Talisha Searcy, and Vaishali Patel. Adoption of electronic health record systems among us non-federal acute care hospitals: 2008-2015. *The Office of National Coordinator for Health Information Technology*, 2016.

[13] HL7. Dosegljivo: http://www.hl7.org/. [Dostopano: 11. 11. 2017].

[14] HL7 Version 3 Guide. Dosegljivo: ftp://ftp.ihe.net/International/Europe/HL7%20V2.5/Version%203%20Ballot%20Cycle%204%20-%20DRAFT/v3ballot4/html/foundationdocuments/helpfiles/v3guide.htm, 2017. [Dostopano: 17. 12. 2017].

[15] Ean-Wen Huang, Sheng-Hsiung Hsiao, and Der-Ming Liou. Design and implementation of a web-based hl7 message generation and validation system. *International Journal of Medical Informatics*, 2003.

[16] IHE. Dosegljivo: http://wiki.ihe.net. [Dostopano: 11. 11. 2017].

[17] Interoperability. Dosegljivo: http://interoperability-definition.info/en/. [Dostopano: 15. 1. 2017].

[18] Semantic Interoperability. Dosegljivo: http://www.hl7.org/documentcenter/public_temp_3CF21336-1C23-BA17-0CF1FAA4C0135219/wg/t3f/general/blobel-schattauer_9_2006_4_343.pdf. [Dostopano: 11. 11. 2017].

[19] Boonchai Kijsanayotin and Win Min Thit. 12 health information standards and interoperability. *Global Health Informatics: Principles of EHealth and MHealth to Improve Quality of Care*, page 149, 2017.

[20] Jessica Wan-Yi Kuo and Alex Mu-Hsing Kuo. Integration of health information systems using hl7: A case study. In *ITCH*, pages 188–194, 2017.

[21] Magic quadrant for the crm customer engagement center. Dosegljivo: https://www.gartner.com/doc/3706717/magic-quadrant-crm-customer-engagement. [Dostopano: 6. 1. 2017].

[22] Force.com migration tool. Dosegljivo: https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_deploying_ant.html, 2017. [Dostopano: 17. 12. 2017].

[23] Frank Oemig, Bernd Blobel, and HL Germany. Hl7 v2+: The future of hl7 version 2? *EJBI*, 2016.

[24] Frank Oemig and Robert Snelick. *Healthcare Interoperability Standards Compliance Handbook*. Springer, 2016.

[25] Open EHR. Dosegljivo: http://www.openehr.org. [Dostopano: 5. 5. 2017].

[26] Open EHR Arhitecture. Dosegljivo: http://www.openehr.org/releases/BASE/Release-1.0.3/docs/architecture_overview/architecture_overview.html. [Dostopano: 17. 5. 2017].

[27] Sarita Pais, Dave Parry, and Yunfeng Huang. Suitability of fast healthcare interoperability resources (fhir) for wellness data. 2017.

[28] Introduction to: HL7 Reference Information Model (RIM). Dosegljivo: https://www.hl7.org/documentcenter/public_temp_2D9C9810-1C23-BA17-0C1C0AA8D7BC0746/calendarofevents/himss/2011/HL7%20Reference%20Information%20Model.pdf. [Dostopano: 11. 11. 2017].

[29] Salesforce - Which API Do I Use. Dosegljivo: https://help.salesforce.com/articleView?id=integrate_what_is_api.htm&type=5. [Dostopano: 17. 12. 2017].

[30] PC Tang et al. Key capabilities of an electronic health record system. *Washington, DC, Institute of Medicine of the National Academies*, 2003.

[31] Introducing HL7 FHIR. Dosegljivo: https://www.hl7.org/fhir/DSTU1/summary.html. [Dostopano: 11. 11. 2017].