UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sebastjan Štucl

Mobilna aplikacija za inventuro osnovnih sredstev

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE
STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2018

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Sebastjan Štucl

Mobilna aplikacija za inventuro osnovnih sredstev

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Rok Rupnik

Ljubljana, 2018

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Platforma IBM Maximo omogoča celovito upravljanje z osnovnimi sredstvi. Veliko je podjetij, ki imajo svoja osnovna sredstva na terenu in ne le v eni ali dveh proizvodnih halah: elektro energetska podjetja, podjetja, ki upravljajo s plinovodi in naftovodi ter energetska podjetja. Takšna podjetja nujno potrebujejo mobilno aplikacijo, ki omogoča zajem podatkov o stanju osnovnih sredstev na terenu.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani, Sebastjan Štucl, izjavljam, da sem avtor diplomskega dela z naslovom:

Mobilna aplikacija za inventuro osnovnih sredstev

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Rok Rupnik-a,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. januarja 2018 Podpis avtorja:

ZAHVALA

Rad bi se zahvalil svojemu mentorju, doc. dr. Roku Rupniku, za strokovno svetovanje, potrpežljivost in spodbudo pri nastajanju diplomskega dela.

Iskrena hvala tudi staršem in bratu, ki so ves čas verjeli vame, me podpirali in spodbujali ob vzponih in padcih ter punci Katarini za vso podporo in ljubezen.

Kazalo

KAZALO SLIK

Seznam uporabljenih kratic

kratica	angleško	slovensko
MVC	Model View Controller	Model-Pogled-Krmilnik
DBMS	Database Management System	Sistem za upravljanje podatkovnih baz
ISO	International Organization for Standardization	Mednarodna organizacija za standardizacijo
IT	Information technology	Informacijska tehnologija
SCADA	Supervisory Control And Data Acquisition	Nadzorna kontrola in akvizicija podatkov
DCS	Distributed control system	Razdeljen sistem za upravljanje
GPS	Global Positioning System	Globalni sistem pozicioniranja
KPI	Key performance indicator	Ključni kazalnik uspešnosti
SOA	Service oriented architecture	Storitveno orientirana arhitektura

Povzetek

Namen diplomske naloge je predstaviti pomembnost učinkovitega upravljanja z osnovnimi sredstvi. Podjetja morajo izvajati letno inventuro osnovnih sredstev, saj s tem ugotavljajo dejansko stanje sredstev in zagotavljajo pravilen prikaz stanja v poslovnih knjigah in izkazih. Za učinkovitejše izvajanje letne inventure je bila razvita mobilna aplikacija, ki prenaša odgovornost upravljanja z osnovnimi sredstvi na zaposlenega. Diplomska naloga je sestavljena iz ekonomskega in računalniškega dela. Na začetku diplomska naloga teoretično opredeljuje upravljanje z osnovnimi sredstvi, opis relevantnih standardov, opis programske rešitve IBM Maximo ter razvoj mobilne aplikacije. Teoretični del je podprt z uporabniškim in programerskim opisom mobilne aplikacije, ki predstavlja ključni del diplomske naloge. Le-ta opisuje postopke razvoja programske opreme, uporabljeno razvojno okolje ter funkcionalnosti, ki jih vsebuje razvita mobilna aplikacija.

Ključne besede: mobilna aplikacija, osnovna sredstva v podjetju, upravljanje z osnovnimi sredstvi, Xcode, Swift, ISO 55000, iOS

Abstract

The aim of this thesis is to present the importance of effective fixed asset management. Companies are obligated to perform yearly inventory of fixed assets. Based on that they establish actual state of assets and ensure presentation of the correct status in the business records. For the inventory to be more effective, mobile application was developed, which enables transfer of responsibility of fixed asset management to an employee. The graduation thesis is divided into economic and programming part. At the beginning, the graduation thesis theoretically defines fixed asset management, description of relevant standards, description of software solution (IBM Maximo), and mobile application development. Theoretical part is supported by the user and programmable description of mobile application, which presents the key part of this graduation thesis. It describes the development of software procedure, used developmental environment, and certain code parts of the mobile application.

Key words: mobile application, company's fixed assets, fixed asset management, Xcode, Swift, ISO 55000, iOS.

Poglavje 1 Uvod

Diplomska naloga opisuje mobilno aplikacijo, ki služi kot pomoč pri upravljanju z opredmetenimi osnovnimi sredstvi. Zakon o gospodarskih družbah navaja, da morajo vsa podjetja izvajati letne inventure na vseh opredmetenih osnovnih sredstvih. Poleg tega želijo podjetja tudi v dobro poslovanja izvajati redne preglede in s tem podaljševati življenjski cikel osnovnega sredstva.

Eden izmed ciljev diplomske naloge je prikazati rešitev, kako na enostaven način del odgovornosti upravljanja z osnovnimi sredstvi prenesemo na uporabnika, ki to osnovno sredstvo uporablja. Poleg letnega inventurnega popisa osnovnih sredstev se lahko mobilna aplikacija uporablja tudi za druge namene. Za ta namen smo oblikovali mobilno aplikacijo, ki je bolj priročna za uporabnika in nam omogoča transparentno, hitrejše ter enostavno upravljanje z osnovnimi sredstvi. Vsak zaposleni je v podjetju odgovoren za svoja osnovna sredstva, ki jih dobi v uporabo za upravljanje dela. Mobilna aplikacija, razvita v okviru te diplomske naloge, omogoča podjetju, da opravi letno inventuro osnovnih sredstev s pomočjo končnih uporabnikov (zaposlenih), ki so odgovorni za dodeljena sredstva. Zaposleni tako s pomočjo mobilne aplikacije enkrat letno potrdijo prostor nahajanja svojega sredstva ter jamčijo, da je le-to v skladu s politiko podjetja. Razvita mobilna aplikacija omogoča potrditev letne inventure in lokacijo sredstva s pomočjo mobilne kamere, ki zajame sliko sredstva oziroma z uporabo čitalca QR/črtne kode. Poleg zgoraj naštetega aplikacija omogoča še prijavo napake. To pomeni, da zaposleni preko mobilne aplikacije pošlje sliko okvarjenega sredstva na center pomoči in s tem odpre zahtevek za pomoč. Ta način ukrepanja omogoča lažje in hitrejše popravilo sredstva, kar vodi v zmanjšanje stroškov podjetja. Podjetje s tem pridobi dodaten vpogled v to, kje se nahajajo osnovna sredstva. Tako se izloči možnost kraje in poškodbe predmetov. Poleg tega je to pomembno zaradi računovodstva in izračuna letne amortizacije ter davka. Tekom diplomske naloge smo s strokovnjaki opravili razgovor s področja upravljanja z osnovnimi sredstvi. Ker so ti strokovnjaki lahko zaposleni tudi na delovnih mestih, kjer je ob izvajanju inventure tudi tveganje poškodb (na primer: transformatorska postaja), je v mobilni aplikaciji dodan tudi gumb za sprožitev tekstovnega sporočila za pomoč.

Poglavje 2 Osnovna sredstva in obvladovanje osnovnih sredstev

2.1 Kaj so osnovna sredstva?

Diplomska naloga se osredotoča na osnovna sredstva. Iz računovodskih zapisov podjetja je razvidno, kaj prištevamo med osnovna sredstva in kakšen je njihov življenjski cikel (izbira, nabava, namestitev, vzdrževanje, popravilo ter odpis osnovnega sredstva) [25].

Pomembno je, da natančno opredelimo, kaj se uvršča med osnovna sredstva v podjetjih. Od vseh teh značilnosti je odvisno, ali podjetje lahko priznava amortizacijo, ki je davčno priznani odhodek podjetja, in ali se posamezni strošek v tekočem poslovnem letu šteje za davčno priznani odhodek. Vse to tudi vpliva na neto dobiček podjetja ob koncu leta [25].

Osnovna sredstva delimo na opredmetena in neopredmetena osnovna sredstva ter dolgoročne finančne naložbe.

Neopredmeteno sredstvo je razpoznavno nedenarno sredstvo, ki praviloma fizično ne obstaja. Praviloma je neopredmeteno dolgoročno sredstvo. Neopredmetena sredstva, katerih posamične vrednosti ne presegajo 500 EUR, se lahko štejejo kot strošek v obdobju, v katerem jih podjetje pridobi.

Neopredmetena osnovna sredstva so le tista neopredmetena dolgoročna sredstva, ki soustvarjajo poslovne zmogljivosti in niso zgolj postavke dolgoročnih časovnih razmejitev (patenti, licence, blagovne znamke in podobno).

Opredmeteno osnovno sredstvo je sredstvo v lasti ali v finančnem najemu, ki se bo uporabljalo več let. Med opredmetena osnovna sredstva štejemo zemljišča, zgradbo, opremo, večletne nasade, zalogo investicijskega materiala, biološka sredstva itd.

Opredmeteno osnovno sredstvo, katerega posamična nabavna vrednost po dobaviteljevem obračunu ne presega vrednosti 500 EUR, se lahko izkazuje skupinsko kot drobni inventar ali kot osnovno sredstvo. Vendar pa je potrebno paziti, da določenega enakega sredstva (npr. računalnik) ne moremo enkrat knjižiti med osnovno sredstvo in drugič med inventar. V primeru, da se nam pri nakupu osnovnega sredstva pojavi dvom, opredelimo sredstvo v skladu z njegovo vsebino. Če nam osnovno sredstvo služi za opravljanje temeljne poslovne dejavnosti

in se bo uporabljalo dlje kot eno leto, ga je smiselno opredeliti kot osnovno sredstvo (npr. računalnik, katerega vrednost je 400 EUR). Pomembno je, da se osnovna sredstva amortizirajo [25].

Dodatno lahko sredstva delimo na:

- stvarno obliko (gradbeni objekti, zemljišča, oprema itd.),
- pravice, patente, licence, blagovne znamke, koncesijo itd.,
- dolgoročne finančne naložbe.

Pomembnost pravilnega vodenja osnovnih sredstev se pozna v bilanci stanja podjetja. Letno mora podjetje ob izkazu finančnega izida prikazati stanje vseh osnovnih sredstev.

2.2 Vrste vrednosti osnovnih sredstev

Nabavna vrednost kupljenega osnovnega sredstva je vrednost, ko nakupni ceni, zmanjšani za popust, prištejemo nevračljive nakupne dajatve ter stroške, ki jih lahko pripišemo direktno za nameravano uporabo, še posebej stroške dovoza in namestitve in ostale stroške, ki so nastali pri nakupu [27].

Nabavna vrednost (NV) je osnova za izračun amortizacije in dejanska naložba v sredstvo oziroma vrednost, po kateri sredstvo ovrednotimo ob začetnem pripoznanju.

Odpisana vrednost je vrednost že obračunane amortizacije osnovnega sredstva. Odpisana vrednost je vidna na popravkih vrednosti posameznih osnovnih sredstev.

Neodpisana vrednost ali sedanja vrednost osnovnega sredstva je razlika med njegovo nabavno vrednostjo in popravkom vrednosti.

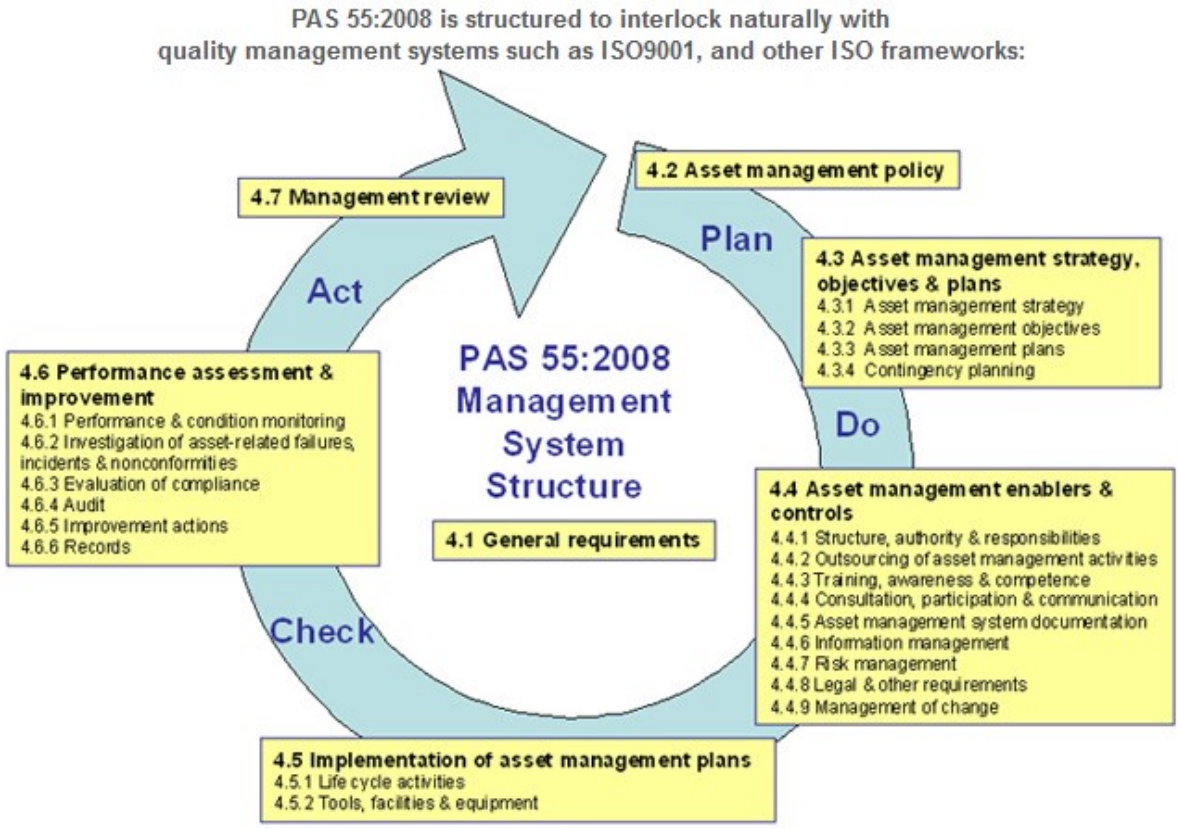
Poštena vednost z računovodskega vidika pomeni ocenjeni znesek denarnih sredstev oziroma ustreznikov, za katerega lahko zamenjamo sredstvo med dobro obveščenim in voljnim kupcem ter dobro obveščenim in voljnim prodajalcem v razumnem poslu.

Zaradi zgoraj navedenih dejstev sklepamo, da je za rast podjetja in pravilen prikaz stroškov zelo pomembno pravilno beleženje in upravljanje osnovnih sredstev [28].

2.3 Standardi za upravljanje z osnovnimi sredstvi

2.3.1 PAS 55

PAS 55 je standard, ki narekuje optimizirano upravljanje z osnovnimi sredstvi. Razvit je bil s strani BSI (podjetja za upravljanje z osnovnimi sredstvi iz Londona) in inštituta za upravljanje osnovnih sredstev. Vsebuje definicije in 28 točk, ki so potrebne za učinkovito upravljanje z osnovnimi sredstvi za vse tipe industrij [23].



Slika 2.1: Diagram za življenjski cikel upravljanja z osnovnimi sredstvi PAS55:2008 [23]

Kot je razvidno iz diagrama (slika 2.1), je za upravljanje s sredstvi pomemben celoten cikel. Ob nakupu sredstva je pomembna politika podjetja, ki mora skrbeti za pravilno upravljanje s sredstvom, saj nam le-ta lahko dolgoročno prinese pomembne prihranke.

O PAS 55 lahko rečemo naslednje:

- opredeljuje seznam za dobre prakse pri načrtovanju življenjskega cikla, stroškov/ optimizacije tveganja in skupnega razmišljanja,
- je standard z zelo veliko bazo uporabnikov,

- je v razvoju že več kot šest let, v sodelovanju z več kot 50 javnih in zasebnih organizacij v desetih državah in petnajstih različnih sektorjih,
- predstavlja možnost uporabe za vse sektorje in vse vrste sredstev,
- podaja podrobna navodila in primere dobre prakse.

2.3.2 ISO 55000 – Standard za upravljanje s sredstvi

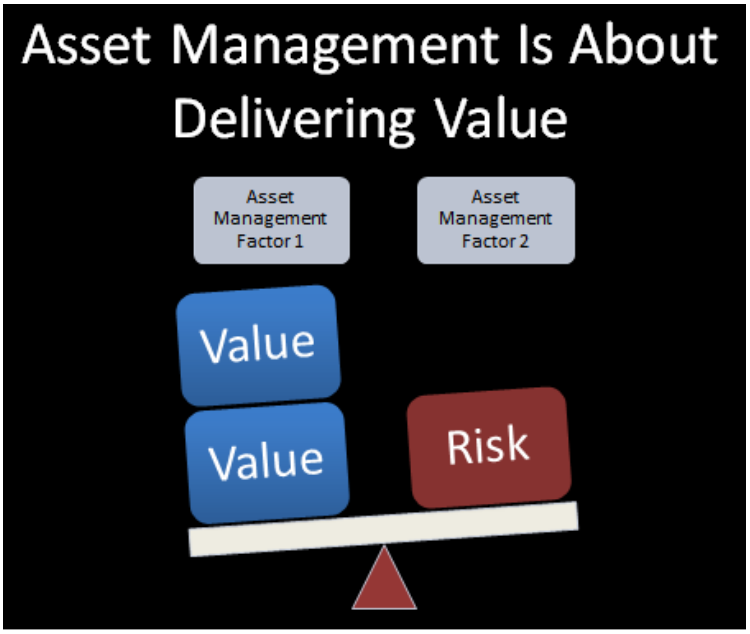
ISO (International Organization for Standardization) je svetovna zveza nacionalnih organov za standardizacijo (ISO). Za pripravo ISO standardov skrbi skupina tehničnih specialistov za določeno področje, ki ga standard obravnava. ISO standardi so zelo pogoste smernice, ki jim današnja podjetja sledijo. Poleg tega podjetja od izvajalcev zahtevajo, da so skladni z določenimi standardi, saj s tem zagotavljajo minimalne pogoje za uspešno izvedene projekte.

Leta 2002 je nastal standard BSI PAS 55 o načinu in pravilnem upravljanju z osnovnimi sredstvi, opisan v tej diplomski nalogi. Leta 2008 so pri ISO organizaciji sprejeli BSI PAS 55 kot osnovo in naredili standard ISO 55000, ki se ukvarja in vpeljuje nove standarde na področja upravljanja z osnovnimi sredstvi [21].

Standard 55000 se predvsem ukvarja s celovitim upravljanjem osnovnih sredstev. Namenjen je:

- podjetjem, ki se sprašujejo, kako realizirati vrednost njihovih sredstev za njihovo organizacijo,
- vsem, ki so vpleteni v realizacijo, implementacijo, vzdrževanje in posodobitev sistema za upravljanje z osnovnimi sredstvi,
- vsem, ki so vpleteni v planiranje, oblikovanje in implementacijo vseh aktivnosti in življenjskega cikla osnovnega sredstva.

V primeru, da podjetje pravilno izpolnjuje zahteve tega standarda, bo dosegalo cilj v zvezi z efektivnim upravljanjem s sredstvi. Aplikacija sistema za upravljanje z osnovnimi sredstvi prinese zagotovilo za doseganje zastavljenih ciljev uporabnikov.



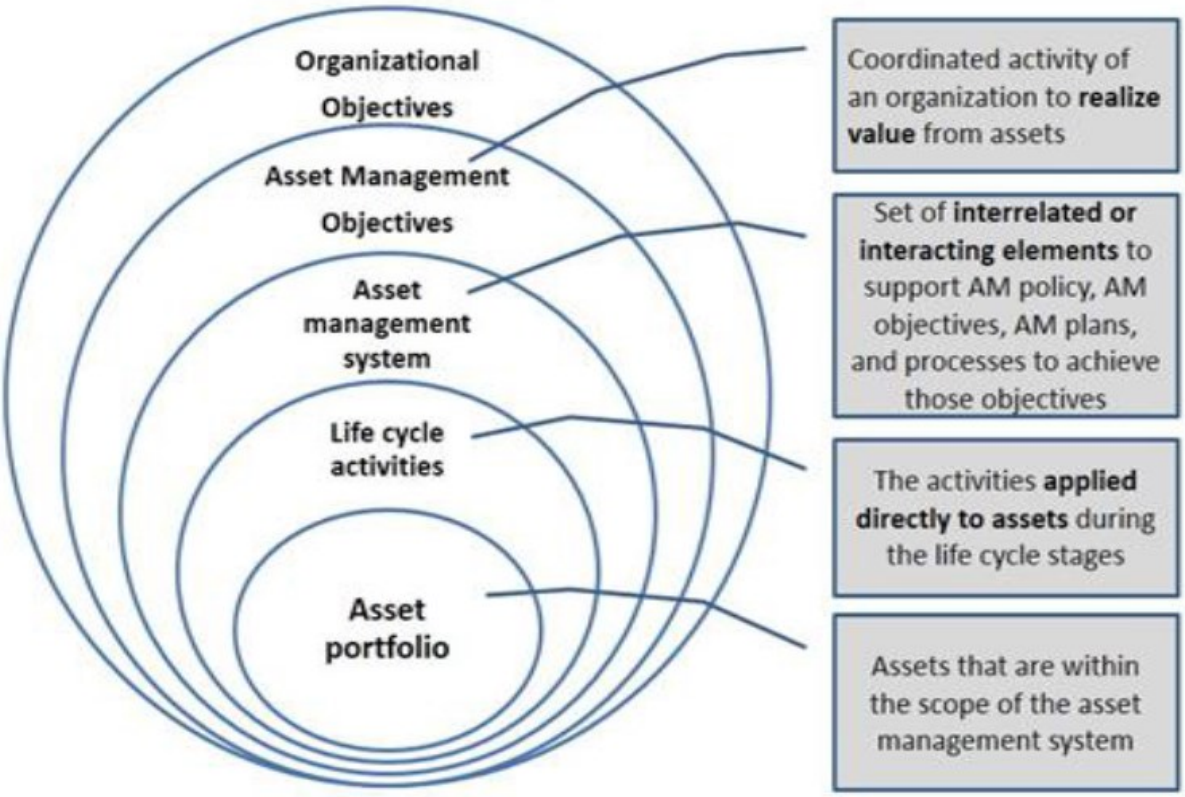
Slika 2.2: Tehnica prikazuje pomembnost v teži vrednosti sredstva [39]

Glede na ISO 55000 je osnovno sredstvo predmet, ki ima vrednost oziroma potencialno vrednost.

Eden izmed pomembnih vidikov, ki ga standard ISO 55000 [22] ne pokriva, je tako imenovan »risk management« oziroma upravljanje s tveganji. Zaradi navedenega je ISO organizacija naredila nov standard ISO 31000, ki se ukvarja s tveganji (primer slike 2.2). Pri upravljanju osnovnih sredstev je pomemben vidik tveganj, ki predstavlja pomemben gradnik za podjetje, saj predvideva in pravilno reši težave na sredstvih. Kot primer navajam krajo računalnika v podjetju, kjer lahko podjetje s pravilno napisanim procesom omogoči zaposlenemu, ki mu je bil ukraden računalnik, novo sredstvo z njegovimi podatki in mu ne povzroči dodatne izgube. Drugi primer upravljanja s tveganji je tudi, kaj se lahko zgodi z določenimi sredstvi, če niso pravilno in pravočasno servisirani oziroma deli niso zamenjani. Vzemimo na primer lokomotivo, ki predvideva zamenjavo določenega dela v roku. Če dela ne zamenjamo pravočasno, lahko pride do nedelovanja lokomotive in v skrajnem primeru tudi do nesreče. Pomembno je, da se zavedamo negativnih posledic v primeru nepravilnega upravljanja s sredstvi [22].

Družina standardov ISO 55000 je sestavljena iz:

- ISO 55000 opisuje izraze in definicije,
- ISO 55001 opisuje zahteve,
- ISO 55002 opisuje priporočila.



Slika 2.3: Shema aktivnosti pri upravljanju z osnovnimi sredstvi [39]

Kot opisuje zgornja shema, je koordiniranje aktivnosti pomemben cilj upravljanja z osnovnimi sredstvi, ki pomaga organizaciji, da oceni vrednost, ki jo le-ta prinaša. Zelo pomembno je, da ob začetku življenjskega cikla (ob nakupu) pravilno vnesemo vse podatke o sredstvu in načrtujemo celoten cikel (slika 2.3).

Podjetje izhaja iz osnovnega portfelja sredstev (asset portfolio) in na to dodaja ustrezne aktivnosti:

- aktivnosti vezane na življenjski cikel osnovnih sredstev (life cycle activities),
- vpeljava sistema za vodenje evidence osnovnih sredstev (asset management system),
- vzpostavitev ciljev in meril za učinkovito realizacijo vrednosti iz osnovnih sredstev (asset management objectives),
- spremembe in ozaveščanje uporabnikov na vseh nivojih o pomembnosti pravilnega vodenja osnovnih sredstev (organisational objectives).

2.3.3 Koristi pri uporabi standarda ISO 55000

Za lastnike podjetij predstavlja skladnost s standardom ISO 55000 dodatno zaupanje v njihove strateške investicije. ISO 55000 spodbuja, da se oceni vrednost vsake intervencije, ki se načrtuje izvesti na sredstva. To nam omogoča, da se ugotovi njena ustreznost in prioriteta [34]. Zelo pomembna je pravilna priprava plana za upravljanje z osnovnimi sredstvi. Pri vpeljavi politike za upravljanja s sredstvi mora podjetje tudi predstaviti meritve te politike. Podjetje mora določiti KPI (Key performance indicator) za meritve uspešnosti pri upravljanju z osnovnimi sredstvi. Glavni KPI so kratkost in dolgoročnost osnovnih sredstev, upravljanje s tveganji, življenjska doba in učinkovitost.

2.4 Primer osnovnih sredstev v informatiki

V vsakem podjetju imajo osnovna sredstva v oddelkih informatike. Med osnovna sredstva prištevamo:

- osebni računalnik,
- monitor,
- licenčna programska oprema,
- čitalec RFID kartic za prihod/odhod,
- strežniška oprema,
- mrežna oprema,
- tiskalniki.
- ostala opredmetena osnovna sredstva..

Vsako podjetje ima dandanes v lasti opredmetena osnovna sredstva, ki spadajo pod stalna sredstva. Vprašanje je, kje je meja, ko vzdrževanje teh sredstev ni več obvladljivo s preprostimi razpredelnicami. V proizvodnji obstajajo naprednejše rešitve, ki lahko na podlagi preteklih podatkov o strojih predvidijo okvaro določenega dela in s predhodno zamenjavo tega podaljšajo življenjsko dobo sredstva. To je mogoče le, če podjetje pravilno hrani vse podatke o sredstvu, s pomočjo uporabe napredne programske opreme. Programska oprema, ki omogoča največ integracij z zalednimi sistemi in specifično bazo znanja za izbrano industrijo, je tudi najboljša rešitev za podjetje. Primer tega so sredstva v jedrski elektrarni, ki imajo drugačne vrednosti in drugačen pomen kot sredstva v hidroelektrarni. Programska oprema mora omogočati integracijo z avtomatiko. Primeri so daljinsko vodenje, napredni SCADA sistemi (Supervisory Control and Data Acquisition – sistemi, ki so namenjeni nadzorovanju in krmiljenju različnih tehnoloških procesov z računalnikom) in določeni DCS sistemi (Distributed control system – celotni krmilni sistem jedrske elektrarne).

2.5 IBM Maximo platforma za upravljanje z osnovnimi sredstvi

Na tržišču poznamo več programskih rešitev za upravljanje z osnovnimi sredstvi. V tej diplomski nalogi je predstavljena programska rešitev Maximo.

IBM Maximo je programska oprema podjetja IBM. Namen aplikacije je upravljanje, vzdrževanje in razpolaganje z osnovnimi sredstvi. MAXIMO EAM in IT Service Management (ITSM) IBM Maximo Asset Management je informacijski sistem za računalniško podprto upravljanje sredstev in storitev, ki na podlagi učinkovite podpore upravljanja sredstev, upravljanja dela, skladiščnem poslovanju ter nabavi pomaga podjetjem pri maksimiranju produktivnosti in podaljšanju življenjske dobe ključnih sredstev ter povrnitvi investicij v njih [30].

Podjetjem omogoča zasnovati in vpeljati strategijo celovitega upravljanja sredstev in storitev. Omogoča centralno zajemanje in hranjenje z upravljanjem in vzdrževanjem sredstev povezanih podatkov ter na podlagi podatkov iz sistema zagotavljanje informacij za učinkovito planiranje, organiziranje, izvajanje, analiziranje in optimizacijo vzdrževalnih del ter sprejemanje s tem povezanih odločitev [8].

Z uporabo sistema Maximo lahko podpremo upravljanje vseh vrst sredstev, minimiziramo nepredvidene zastoje, zmanjšamo ne planirano vzdrževanje, optimiziramo skladiščne zaloge, zagotovimo pravočasno razpoložljivost rezervnih delov, učinkovito obvladujemo nabavo, zmanjšamo stroške vzdrževanja in povečamo operativno učinkovitost.

IBM Maximo na isti platformi vsebuje vsa ključna orodja in funkcionalnosti za razvoj, konfiguriranje, uporabo, avtomatizacijo, analitsko poročanje, obveščanje, varnost dostopa, uvoz podatkov, integracijo z drugimi sistemi ter podporo uporabnikom (Help Desk).

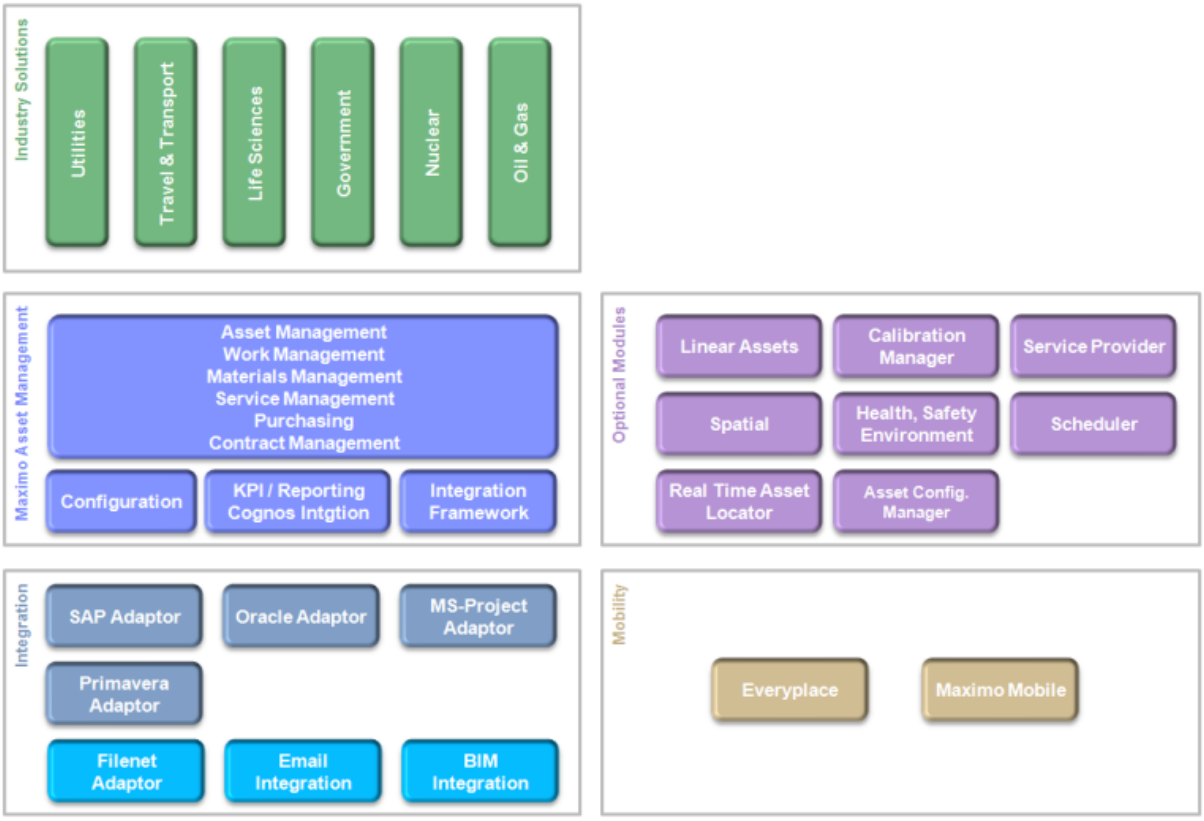
Maximo podpira naslednje industrijske panoge:

- proizvodnja (olje, plin, kemična proizvodnja, hrana),
- infrastruktura (železnice, ceste, telekomunikacije, vodni sistemi, električne distribucije),
- transport (letalski, cestna in železniška vozila, plovila),
- nepremičnine,
- IT (Informacijska tehnologija).

2.5.1 Arhitektura IBM Maximo

IBM Maximo je zgrajena po konceptu J2EE service-oriented (SOA) arhitekturi. Kot prikazuje slika 2.4, Maximo združuje industrijsko orientirano okolje s programskimi rešitvami (integracije, mobilnost, poročila, upravljanje ...). V našem primeru sem se osredotočil na del z integracijami prek API vmesnika, ki mi je omogočal dostop do podatkov, shranjenih v IBM Maximo. Informacije so dostopne v JSON zapisu ali v obliki XML datoteke. Maximo vključuje šest glavnih modulov v obliki SOA arhitekture, kot so upravljanje s sredstvi, upravljanje virov, upravljanje storitev, upravljanje pogodb, upravljanje zalog in nabava.

Torej lahko rečemo, da je SOA (storitveno orientirana arhitektura) optimalna in učinkovita uskladitev poslovnih in informacijskih tehnologij, ki jo podpirajo [9].



Slika 2.4: J2EE modularna arhitektura programske rešitve IBM Maximo [33]

2.5.2 IBM Maximo API (Application Programming Interface)

IBM Maximo omogoča dostop do podatkov, shranjenih v podatkovni bazi, prek spletnega vmesnika in nabora ukazov, ki se imenujejo RESTful. REST predstavlja arhitekturo, kjer je vsak vir predstavljen kot spletna storitev z enoznačnim naslovom URL. Za dostopanje in

spreminjanje virov se tako uporabljajo standardne metode HTTP: GET, PUT, POST in DELETE.

Odločil sem se za dostop do spletnih virov z uporabo REST HTTP protokola. Lahko bi uporabil tudi prenos XML datotek in interpretacijo teh, a je v primeru mobilnih aplikacij priporočljivo, da se pošiljanje zahtevkov na strežnik zmanjša, saj s tem uporabniku tudi pospešimo prikaz in obdelavo podatkov. S tem, ko bi pošiljali XML datoteko na strežnik, bi povečali čas dostopa in zmanjšali zadovoljstvo uporabnika [7].

Če želimo dostopati do osnovnega sredstva z inventarno številko 123 in uporabo XML datotek, moramo spodnji XML poslati na strežnik in počakati na odgovor.

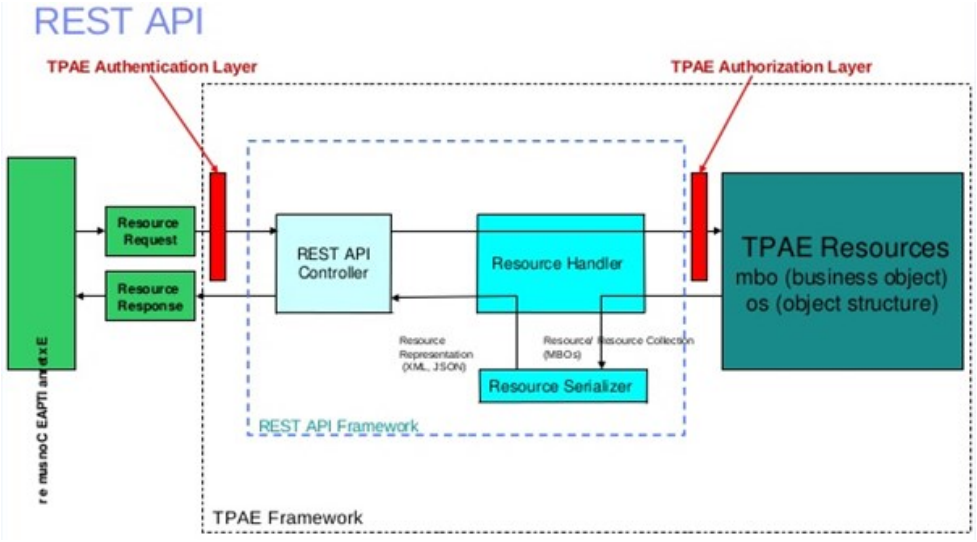
Primer XML datoteke:

```
  <?xml version="1.0" encoding="UTF-8"?>
<QueryMXASSET
 xmlns="http://www.ibm.com/maximo" xmlns:xsi="http://www.w3.org/
 2001/XMLSchema-instance"    transLanguage="EN">
  <MXASSETQuery>
   <ASSET>
    <ASSETNUM>123<ASSETNUM>
   </ASSET>
    </MXASSETQuery>
   </QueryMXASSET>
```

S klicem preprostega URL naslova preko uporabe REST lahko dosežemo enak učinek:

```
GET …/maxrest/rest/mbo/asset?assetnum=123
```

V obeh primerih nam strežnik odgovori s podatki o izbranem osnovnem sredstvu. V prvem primeru nam tudi iz programerskega vidika vzame več časa, saj je potrebno generirati in potrditi XML datoteko. Spodnja slika prikazuje logično shemo dostopa do podatkov s pomočjo REST api Controllerja. IBM Maximo celotno okolje za obdelavo zahtevkov imenuje »TPAE Framework«.



Slika 2.5: Prikaz komunikacije s pomočjo REST API krmilnika za dostop do podatkov v IBM Maximo [2]

V našem primeru nam strežnik odgovori v obliki JSON, kot je prikazano na spodnji sliki.

Slika 2.6: Izpis podatkov o osnovnem sredstvu v JSON formatu

Kot je razvidno iz slike 2.6 lahko iz strežnika pridobimo vse potrebne informacije o osnovnem sredstvu. Za eno osnovno sredstvo lahko dobimo več kot štirideset različnih podatkov, s katerimi lahko razpolagamo. V primeru, da bi izdelali aplikacijo za finance, bi lahko s pomočjo zgornjega URL zahtevka izračunali amortizacijo iz polja »PURCHASEPRICE« in »INSTALLDATE«. Trend je, da se večina podatkov posreduje s pomočjo REST HTTP protokola, kot prikazuje slika 2.5. S tem se izognemo direktnemu povezovanju na podatkovne baze, kar posledično omogoča tudi varnejši dostop do podatkov [3].

Poglavje 3 Izvajanje letne inventure

3.1 Kaj je letna inventura?

Zakon o gospodarskih družbah (54. členom Zakona o gospodarskih družbah - ZGD-1) opredeljuje, da je potrebno enkrat letno preveriti, ali se stanje aktivnih in pasivnih postavk v poslovnih knjigah ujema z dejanskim stanjem [37]. Zaradi tega razloga mora podjetje letno preveriti vsa opredmetena sredstva, njihovo stanje in lokacijo. Dejstvo je, da si lahko podjetje s sprotnim in hitrim vzdrževanjem osnovnih sredstev podaljšuje življenjsko dobo sredstva in doseže večjo učinkovitost sredstva. Podjetja se pogosto soočajo s tem, kako lahko z nizkimi stroški dosežejo učinkovitost osnovnih sredstev in hkrati izpolnijo zakonske obveznosti.

Popis osnovnih sredstev pomaga podjetju uskladiti knjižna stanja z dejanskim stanjem. Za ta namen se ustanovi popisna komisija, ki skrbi za verodostojnost podatkov. Komisija vključuje vsaj 3 člane in ima nalogo popisati ugotovljene razlike med evidentiranimi in dejanskimi podatki, katere je potrebno tudi uskladiti. Zavedati se namreč moramo, da podatki, ki jih dobimo na podlagi letnega popisa, vplivajo na višino kapitala, višino poslovnega izida in davčne osnove.

Dejstvo je, da se zaposleni vsaj enkrat letno z **veliko nejevoljo lotevajo** tega opravila, saj se pogosto zgodi, da se številke ne ujemajo, pa tudi sam postopek prepoznavanja in pregleda izdelkov je časovno potraten. To je tudi eden izmed glavnih razlogov razvoja prototipa mobilne aplikacije v tej diplomski nalogi.

3.2 Učinkovito izvajanje letne inventure

Glede na zakonske zahteve, omenjene v točki 3.1, mora podjetje letno izvajati inventuro nad osnovnimi sredstvi. Izziv predstavlja dodaten strošek, ki ga tovrstni pregled prinese. Predvsem je strošek za delo delavca in informacijskega sistema za učinkovito vodenje evidence osnovnih sredstev. Razvita mobilna aplikacija v tej diplomski nalogi omogoča letni popis osnovnih sredstev in s tem zniževanje stroškov dela ter posledično dviguje učinkovitost. Zaposleni lahko iz mobilne aplikacije tudi prijavi napako na izbranem osnovnem sredstvu. Posledično je zaradi mobilne aplikacije odprava napake na sredstvu hitrejša. Center pomoči ob prijavi napake dobi vse informacije o sredstvu, točno lokacijo sredstva in informacijo o odgovorni osebi. Z mobilno aplikacijo lahko znatno zmanjšamo čas odprave napake na sredstvu. Odgovornost zaposlenega je tudi, da vsako leto preko mobilne aplikacije potrdi lokacijo sredstva. S tem podjetje izpolni zakonske zahteve in se hkrati izogne kraji in dodatnemu strošku, saj prenese del odgovornosti na zaposlenega.

Inventura ali popis sredstev in obveznosti pa ni le predpisano opravilo, temveč vodi poslovodstvo k sprejemu novih organizacijskih ukrepov in poslovne politike podjetja, ki vodi k bolj smotrnemu, racionalnemu, urejenemu in odgovornemu ravnanju vseh segmentov poslovanja podjetja.

Poglavje 4 Prototip mobilne aplikacije za izvedbo inventure

Mobilna aplikacija je napisana v programskem jeziku Swift. Za razvoj in testiranje aplikacije je potrebno od podjetja Apple kupiti uraden certifikat za izdelavo mobilnih aplikacij. Ta certifikat omogoča namestitev mobilne aplikacije na mobilni telefon proizvajalca Apple (iOS) in uporabo Apple Xcode okolja. Med razvojem aplikacije sem aplikacijo testiral na mobilnem telefonu iPhone 5s.

4.1 Pregled uporabljenih tehnologij za razvoj programske rešitve

4.1.1 Swift

Programski jezik Swift je produkt podjetja Apple. Uporablja se za naprave, ki uporabljajo operacijski sistem za mobilne naprave iOS in OS X . Swift programski jezik je bil javnosti prvič predstavljen junija leta 2014. Kot nadomestilo za svoj predhodnik Objective-C (obstoječi programski jezik podjetja Apple za razvoj mobilnih aplikacij) je osvojil svetovni trg. Swift omogoča tudi integracijo knjižnic iz Objective-C, kar uporabniku omogoči lažji prehod iz starejšega programskega jezika v nov, uporabniku prijaznejši programski jezik Swift. Swift naj bi bil bolj odporen na napake kot njegov predhodnik Objective-C. Poleg tega omogoča, da se za isto rešitev porabi manj programske kode [1].

Razvoj programskega jezika Swift se je začel leta 2010, a do leta 2014 ni bil pripravljen za izdajo. V osnovi gre za prestrukturiranje jezika Objective-C, ki vpeljuje moderni način programiranja z uporabo moderne sintakse in konceptov. Swift prav tako opušča uporabo kazalcev na dele pomnilnika in s tem zmanjšuje možnosti napak pri klicanju določene vrednosti iz pomnilnika. Swift jezik se pogosto primerja z novejšimi programskimi jeziki kot so Java, C++.

Primer za izpis znanega besedila »Hello, world«:

Objective-C:

```
NSString *str = @"Hello,";
str = [str stringByAppendingString:@" world"];
NSLog(@"%@", str);
```

Swift:

```
var str = "Hello,"
str += " world"
print (str)
```

Glede na zgoraj prikazani primer je razvidna razlika med starejšim jezikom Objective-C in novejšim jezikom Swift [19, 20]. Pri Swift jeziku so v jedro implementirali deklaracije spremenljivk, kar uporabniku omogoča lažjo manipulacijo z določenimi podatkovnimi tipi.

Dodatne razlike so:

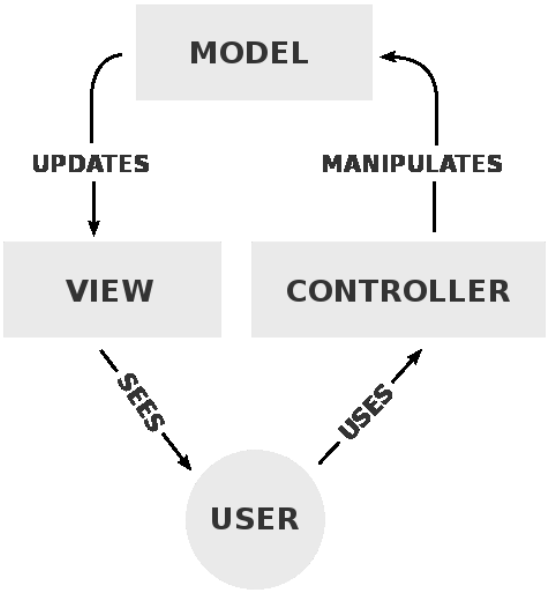
- programski stavki ne potrebujejo »;« na koncu stavka,
- implementacija dodatnih knjižnic ni potrebna,
- lažje razumevanje,
- prepis funkcij v izbranem razredu,
- v primeru napake/izjeme dodatno programiranje ni potrebno.

Zaradi zgoraj navedenih izboljšav je bila mobilna aplikacija te diplomske naloge razvita v novejšem programskem jeziku Swift [18].

4.1.2 MVC – Model, View, Controller

Mobilna aplikacija je narejena po MVC arhitekturi. MVC pomeni Model, View in Controller. »Model« predstavlja podatke, »View« predstavlja prikaz vsebine uporabniku, »Controller« predstavlja krmilnik med pogledom, podatki in funkcijami. Vse funkcije in manipulacije s podatki se dogajajo v »Controller« delu.

Kot prikazuje spodnji diagram, se uporabniku vedno prikaže vsebina (»View«), kjer »Controler« skrbi za dostop do podatkov in jih posreduje prikazovalniku vsebine (»View«). Do »Modela« uporabnik nima neposrednega dostopa, saj za to skrbita »View« in »Controller«. S pomočjo tega pristopa programiranja se obranimo zlonamernim napadom, znanim kot »SQL Injection«. To je tehnika napada, kjer pridobimo podatke iz podatkovne baze brez neposrednega vdora v računalnik.



Slika 4.1: Prikaz toka MVC arhitekture [35]

Pogled poskrbi za predstavitev podatkov v obliki, ki bo primerna za uporabnika in je vmesnik med uporabnikom in sistemom. Krmilnik se odziva na akcije uporabnika in komunicira z modelom in s pogledom ter ju povezuje in upravlja. Model poskrbi tudi za podatkovno abstrakcijo, da pogled in krmilna logika dostopata do podatkov samo prek modela. S tem se izognemo vezanosti na posamezno podatkovno bazo. MVC arhitektura ima več prednosti (slika 4.1). Omogoča enostavnejše vzdrževanje, koda je preglednejša in naloge se lažje delijo na več uporabnikov. Ti so lahko specializirani za uporabniške vmesnike in se lahko posvetijo samo pogledu, krmilno logiko pa prepustijo drugim. Ena od posledic arhitekture je tudi, da v primeru uporabe aplikacije na različnih napravah ni potrebno spreminjati in podvajati krmilne logike in modela, temveč se izdela samo več pogledov [10].

4.2 Razvoj mobilne aplikacije

Namen mobilne aplikacije je izvedba letne inventure osnovnih sredstev. Mobilna aplikacija je bila razvita v razvojnem okolju podjetja Apple, imenovanem Xcode [36]. Okolje omogoča razvoj za vse Applove naprave na tržišču (iPhone, iPad, iWatch, Mac, tvOS). Xcode ima tudi zelo napreden sistem za odpravo napak (debug), ki uporabniku zelo olajša delo [36]. Poleg tega okolje vsebuje tudi simulator, ki nam omogoča izvajanje mobilne aplikacije, razvite v xCodu, brez mobilne naprave. Edina slabost, ki sem jo odkril med razvijanjem aplikacije, je, da simulator nima na voljo kamere. Ker za skeniranje osnovnega sredstva QRcode in črtne kode

potrebujemo kamero, je bilo potrebno vsako verzijo mobilne aplikacije prenesti in izvajati na mobilni napravi.

Za razvojno okolje Xcode lahko ugotovimo naslednje [16,17]:

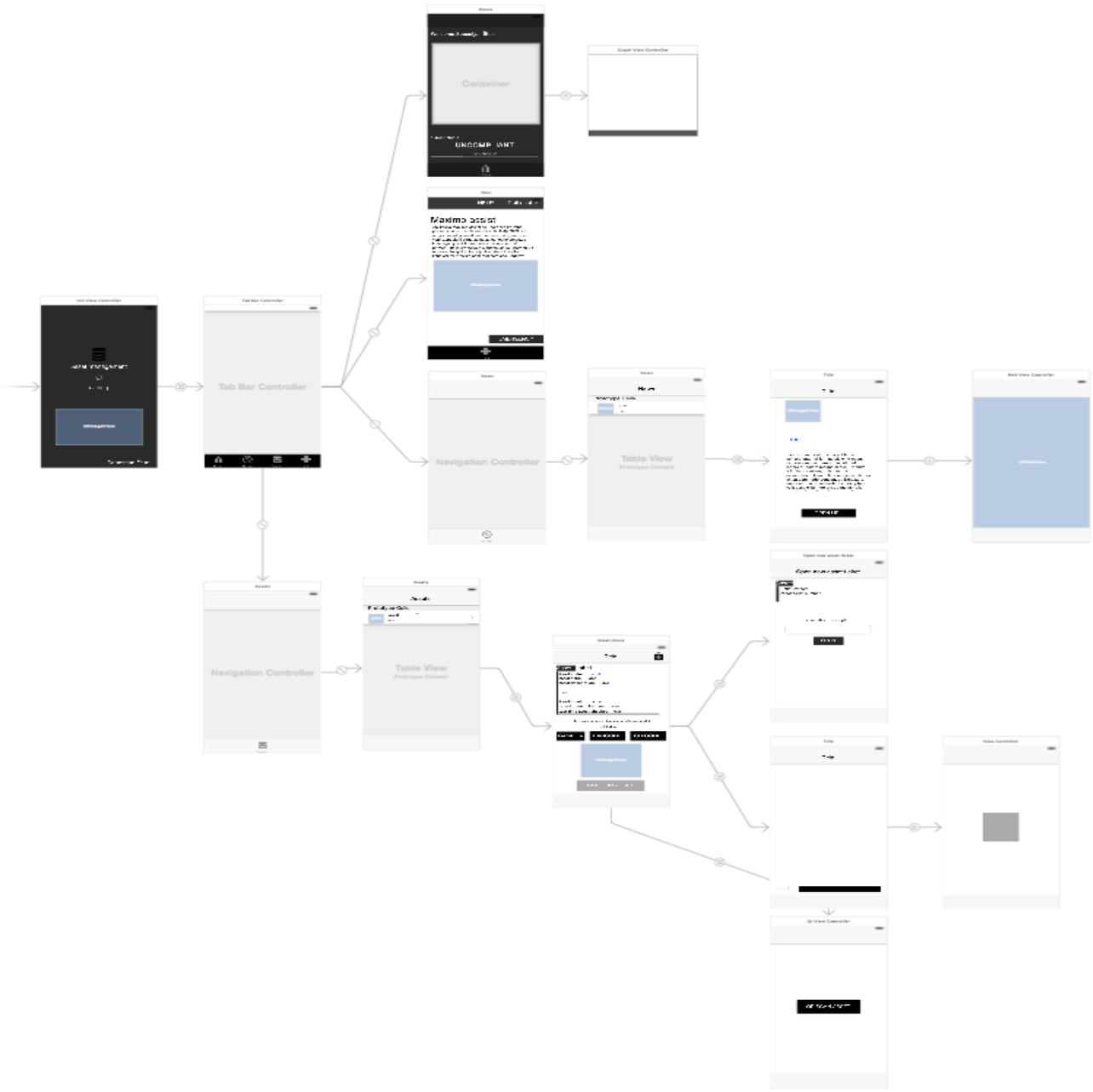
- sestavljajo ga inovativna orodja, ki pomagajo pri razvoju aplikacij,
- ima poenostavljen vmesnik, ki je enostaven za uporabo,
- vsebuje profesionalni urejevalnik kode (iskanje knjižnic in funkcij),
- ima vgrajeno LLVM tehnologijo (infrastruktura prevajalnika napisana v C++ jeziku), ki poišče in popravi napake,
- vsebuje inštrumente za testiranje vizualnih zmogljivosti (emulatorji).

4.2.1 Funkcionalnosti mobilne aplikacije

Na kratko bomo za lažjo predstavitev opisali strukturo mobilne aplikacije in njene forme:

- **Prijava** (izvedba avtentikacije do centralnega sistema Maximo),
- **Prikaz trenutnega stanja** (prenos podatkov iz Maxima o sredstvih in stanje letne inventure),
- **Potrditev osnovnega sredstva** (ob letni inventuri, potrditev verodostojnosti osnovnega sredstva),
- **Seznam osnovnih sredstev** (seznam osnovnih sredstev, ki so pisani na uporabnika),
- **Klic v sili** (klic ali tekstovno sporočilo na center pomoči).

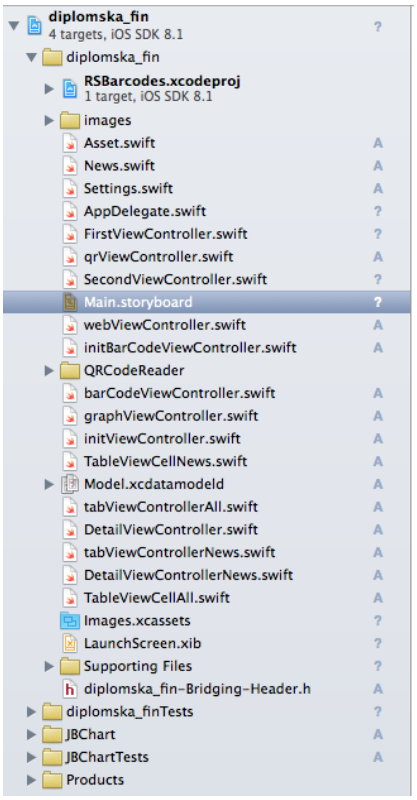
Spodnja slika prikazuje krmilnike in komunikacije med njimi, zgrajene v programskem jeziku Swift. Kot smo že omenili, sledi arhitektura MVC konceptu. Na spodnji sliki 4.2 je grafično vidna tudi komunikacija med pogledom in krmilnikom. To se vidi pri krmilniku Navigation Controller, ki skrbi za prikaz seznama sredstev in Table view pogledom, ki prikaže ta seznam uporabniku. Aplikacija je sestavljena iz štirih glavnih pogledov - Home, News, Assets in Help.



Slika 4.2: Celotna shema klicev med krmilniki za razvito mobilno aplikacijo

Na sliki 4.2 prikazana shema klicev je med krmilniki je žal nepregledna, vendar je potrebno poudariti, da je cilj prikaza sheme klicev prikazati vlogo krmilnikov in posredno tudi strukturo (arhitekturo) aplikacije. V kasnejših poglavjih je arhitektura iz slike 4.2 opisana po posameznih delih kode.

Krmilniki upravljajo zahtevek, obdelujejo podatke, vrnjene iz modela, in nalagajo poglede, da pošljejo odziv uporabniku. V aplikaciji sem razvil štirinajst različnih krmilnikov, ki skrbijo za prikaz in obdelavo podatkov kot prikazuje slika 4.3.

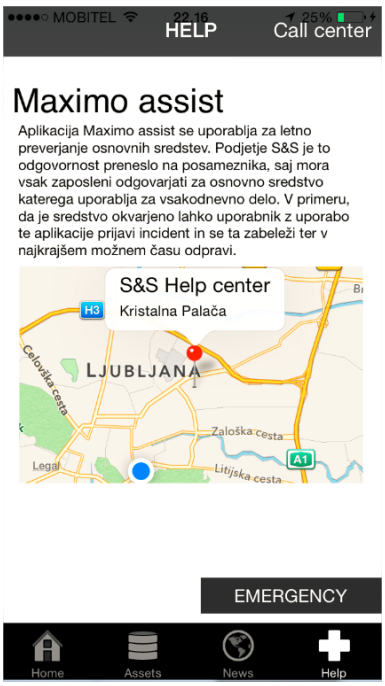


Slika 4.3: Seznam uporabljenih krmilnikov (Controllers)

Uporabniku se ob prijavi, naloži zadnje stanje osnovnih sredstev, ki so v upravljanju uporabnika. To se izvede s pomočjo REST ukaza GET in sicer v izpisu formata JSON iz centralnega strežnika. Uporabnik dobi popolne informacije o sredstvih, kot so ime sredstva, serijska številka, lokacija sredstva, stanje sredstva in ostalo. Uporabniku se naloži tudi zadnja slika sredstva. Nato lahko uporabnik s pomočjo mobilne aplikacije sredstvo prijavi - potrdi podjetju, da ima v lasti to sredstvo ter da je sredstvo v skladu s politiko podjetja. To naredi s pomočjo zajema slike ter z uporabo Qrcode/BARcode skenerja, ki potrdi pristnost sredstva. Aplikacija avtomatsko shrani GPS koordinate sredstva v samo sliko in omogoča administratorju, da preveri, kje se je sredstvo nahajalo ob zajemu slike. Poleg osrednje funkcionalnosti aplikacija tudi omogoča prijavo napak na osnovnem sredstvu. Uporabnik ima s klikom možnost prijaviti napako na izbranem sredstvu in s tem skrajšati čas vnašanja serijske številke in tipa sredstva v brskalnik na računalniku oziroma pri prijavi napake po telefonu.

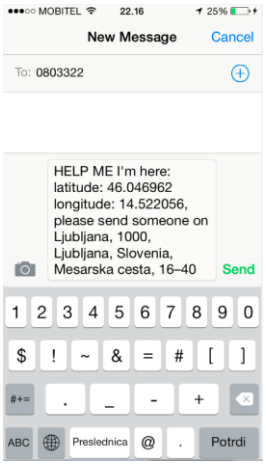
Izbrana tehnologija Maximo podjetja IBM nam omogoča povezave do podatkov s pomočjo API klicev. Z uporabo REST ukazov uporabnik pridobi seznam sredstev, s katerimi upravlja. Hkrati se sredstva uporabniku prikažejo na mobilni aplikaciji. Po končani prijavi sredstva uporabnik z gumbom "CHECKIN" pošlje potrditev na centralno aplikacijo IBM Maximo. Ko uporabnik

uspešno potrdi vsa sredstva, se mu v meniju "Home" spremeni stanje iz "UNCOMPLIANT" (neskladno) v "COMPLIANT" (skladno).



Slika 4.4: Dostop do centra pomoči v mobilni aplikaciji

Poleg tega aplikacija ponuja dve dodatni funkcionalnosti, in sicer News ter Help. »News« prikazuje zadnje novice iz podjetja, ki v JSON formatu iz centralnega strežnika naloži zadnje novice (prikaz novic iz intraneta). Druga možnost je »Help«, ki uporabniku v primeru iskanja pomoči nudi vse potrebne informacije. V primeru, da je uporabnik aplikacije v nevarnosti, je tu še gumb EMERGENCY (primer na sliki 4.4), ki s pomočjo SMS-a pošlje trenutno lokacijo na centralo podjetja (ta zahteva je bila podana iz strani uporabnikov). To se ponazori na primeru elektrotehnikov, ki so lahko pri delu (torej tudi pri izvajanju inventure) na transformatorjih življenjsko ogroženi.



Slika 4.5: Primer tekstovnega sporočila za pomoč

Glavne funkcionalnosti mobilne aplikacije so:

- prikaz osnovnih sredstev v lasti uporabnika iz centralnega sistema za upravljanje z osnovnimi sredstvi – Maximo,
- prijava in potrditev osnovnih sredstev s pomočjo zajema slike,
- prijava in potrditev osnovnih sredstev s pomočjo branja QR in črtne kode,
- prijava napake na osnovnem sredstvu v center pomoči,
- prikaz zadnjih novic iz podjetja,
- klic v sili (telefon in sms).

4.2.2 Programske komponente mobilne aplikacije

4.2.2.1 Upravljanje s podatki (MVC - Module)

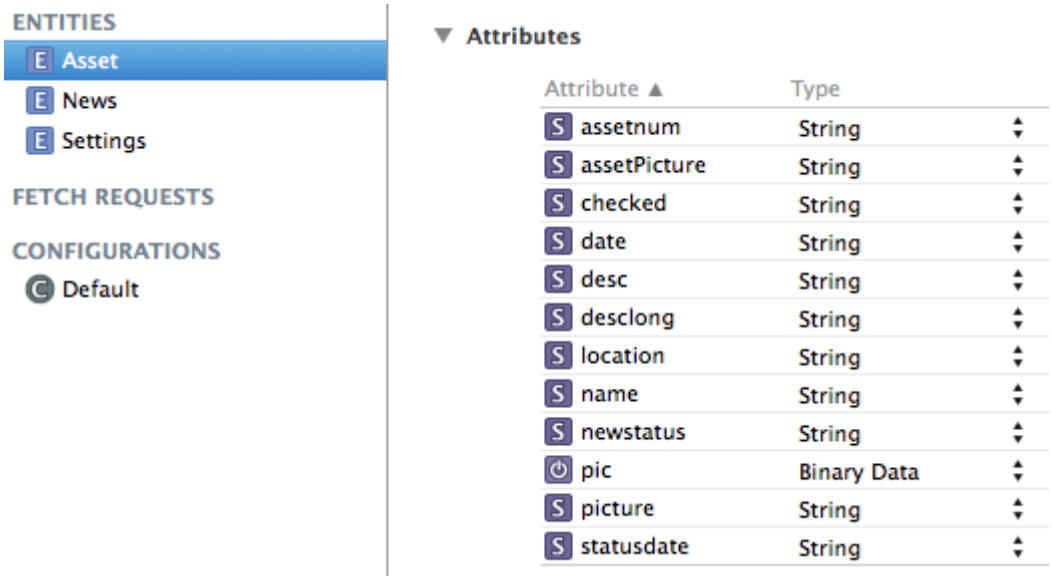
V primeru mobilnih aplikacij na platformi iOS je za pravilno uporabo in shranjevanje podatkov na mobilnik potrebno inicializirati knjižnico **CoreData**. CoreData omogoča uporabo metod, s katerimi dostopamo do podatkov, shranjenih v podatkovni bazi mobilne naprave. Uporabnik lahko podatke shrani v obliki XML zapisa, binarnih podatkov (slike, videa) in relacijske baze (SQLite). Uporabil sem zadnji dve obliki zapisa. V primeru fotografiranje osnovnega sredstva (zajema slike osnovnega sredstva), sem sliko shranil v binarni zapis v spremenljivko **pic**, kot tudi prikazuje slika **4.6**. Dostop do podatkov, shranjenih v CoreData, je možen samo prek objektov, kar pomeni, da direkten dostop do podatkov ni možen. Uporabnik mora najprej ustvariti objekt, ki kaže na podatke, do katerih želi dostopati.

```
let appDelegate = UIApplication.sharedApplication().delegate as AppDelegate

let managedContext = appDelegate.managedObjectContext!

//2
let fetchRequest = NSFetchRequest(entityName:"Asset")

//3
var error: NSError?

let fetchedResults =
managedContext.executeFetchRequest(fetchRequest,
    error: &error) as [NSManagedObject]?

if let results = fetchedResults {
    asset = results
} else {
    println("Could not fetch \(error), \(error!.userInfo)")
}
```

Slika 4.6: Prikaz enega izmed načinov dostopa do podatkov, shranjenih v CoreData

Kot je prikazano na sliki 4.6, uporabnik s pomočjo ukaza appDelegate iz knjižnice CoreData ustvari objekt, ki služi kot »tunel« med trenutnim krmilnikom in podatki na mobilni napravi. S pomočjo NSFetchRequest izberemo tabelo, iz katere želimo manipulirati s podatki. Ukaz executeFetchRequest, ki je metoda v objektu managedContext, nam omogoči prenos podatkov iz glavnega pomnilnika na mobilni napravi v delovni pomnilnik in uporabo oziroma prikaz teh podatkov.

Swift nam omogoča grafično dodajanje novih entitet znotraj Xcode programerskega okolja. To je za programerja zelo uporabno in enostavno. Ustvaril sem tri različne entitete s posameznimi atributi za delovanje mobilne aplikacije. Prva entiteta je »Asset«, v kateri so shranjeni podatki o osnovnem sredstvu. Druga entiteta se imenuje »News«. V njej so shranjene zadnje novice iz intraneta podjetja. Tretja entiteta pa je »Settings«, ki ima shranjene uporabnikove nastavitve. Xcode nam tudi omogoča spremembo trenutne oblike modela.



Slika 4.7: Prikaz modela podatkov (podatkovne baze)

Slika 4.7 prikazuje implementacijo objekta Asset, ki nam omogoča dostop do podatkov, shranjenih v podatkovni bazi mobilne naprave. Kot je razvidno iz slike 4.8, zahteva jezik deklaracijo vsake posamezne spremenljivke. V primeru spremenljivke "pic" uporabimo podatkovni tip "NSData", ki predstavlja binarni zapis vrednosti (v tem primeru shranim v to spremenljivko sliko, zajeto iz strani uporabnika).

```
import Foundation
import CoreData

@objc(Asset)
class Asset: NSManagedObject {

    @NSManaged var assetPicture: String;
    @NSManaged var checked: String;
    @NSManaged var date: String;
    @NSManaged var name: String;
    @NSManaged var pic: NSData
    @NSManaged var picture: String;
    @NSManaged var desc: String;
    @NSManaged var desclong: String;
    @NSManaged var location: String;
    @NSManaged var newstatus: String;
    @NSManaged var statusdate: String;
    @NSManaged var assetnum: String;

}
```

Slika 4.8: Prikaz deklaracije metode za dostop do podatkov

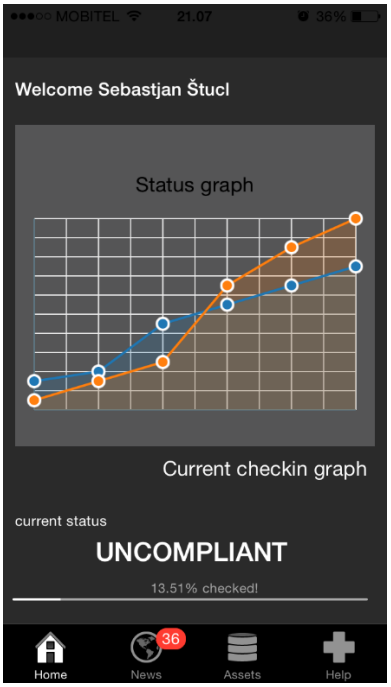
To izbrano zbirko kličemo s pomočjo ukaza:

Let fethedResults = managedContext.executeFetchRequest(fetchRequest,error: &error) as [NSManagedObject]?

Slika 4.8 prikazuje tudi strukturo modela znotraj mobilne aplikacije, kjer so shranjeni podatki.

4.2.2.2 Graf uspešnosti

FirstViewController predstavlja prvo stran aplikacije, ki se uporabniku naloži ob zagonu. Uporabnik lahko na grafu preveri njegovo trenutno število potrjenih sredstev.



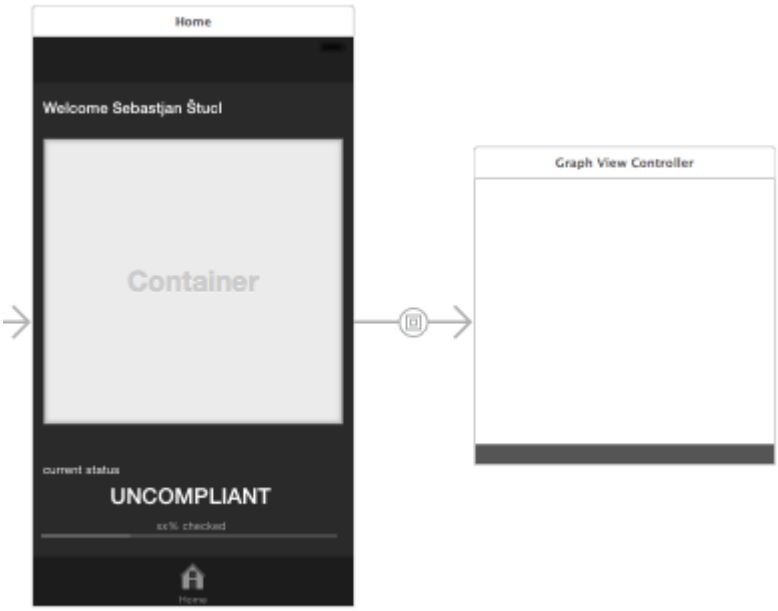
Slika 4.9: Graf o uspešnosti, prikazan na mobilni aplikaciji

Osrednji del predstavlja graf, ki prikazuje trenutno stanje uporabnika. Uporabnik lahko preveri svoje rezultate o tem, kdaj je bila v tekočem letu prijavljena napaka na njegovih osnovnih sredstvih. Oranžna črta pa prikazuje stanje o številu potrjenih osnovnih sredstev.

Izris grafa se dogaja s pomočjo krmilnika graphViewControllerja. V krmilniku FirstViewController je ustvarjen poseben kontejner, v katerega se naloži krmilnik graphViewController. Ta krmilnik določa, kako naj se graf izriše. To se naredi s pomočjo odprtokodne knjižnice JBChart [6].

Navedena knjižnica je napisana v Objective-C jeziku, vendar Xcode urejevalnik omogoča tako imenovani most med obema jezikoma za uporabo Objective-C knjižnic v Swift jeziku.

Spodaj je prikaz **pogleda** iz MVC arhitekture za lažjo predstavo integracije krmilnika graphViewController v krmilnik FirstViewController.



Slika 4.10: Prikaz krmilnika Graph View Controller

Poleg navedenega grafa ima aplikacija na osnovnem pogledu tudi "progress bar" oziroma vrstico napredka, ki se avtomatsko povečuje s tem, ko uporabnik uspešno prijavi osnovno sredstvo po pravilnem postopku (zajame sliko sredstva in skenira sredstvo s pomočjo Qreaderja ali BARcode readerja). Aplikacija v bazi avtomatsko preveri vsa sredstva in njihov status (primer: **Checked)**. Letno, ko podjetje zahteva ponovno validacijo osnovnih sredstev, se ta spremenljivka na vseh sredstvih spremeni iz "true" v "false", kar pomeni, da se vrstica napredka spremeni v 0 %. V tem primeru mora uporabnik ponovno potrditi vsa sredstva, ki so mu dodeljena. Ob vsaki uspešni validaciji se v spremenljivko **Checked** shrani vrednost "true".

4.2.2.3 Čitalec QR kode

Qrviewcontroller skrbi za delovanje kamere za prepoznavo QR kode (QR koda je matrična oz. dvodimenzionalna (2D) črtna koda). Pri tem je uporabljena knjižnica AVSFoundation. Podjetje Apple v novejših verzijah omogoča uporabo bralcev kod, kot sta QRReader in BARReader brez dodatnega programiranja kamere. V tej diplomski nalogi je bila uporabljena koda uporabnika yannickl iz githuba [4], ki omogoča enostavno implementacijo Qrcode kamere. Koda na spodnji sliki prikazuje implementacijo tega razreda. V glavi razširimo osnovni ViewController razred s QRCodeReaderDelegate razredom in s tem omogočimo uporabo metod za klic in branje QR

kode. Qrcode je tekstovni zapis v grafični obliki, ki omogoča lažje in hitrejše branje zapisa. Lahko bi rekli, da je to nadgradnja starejše črtne kode.

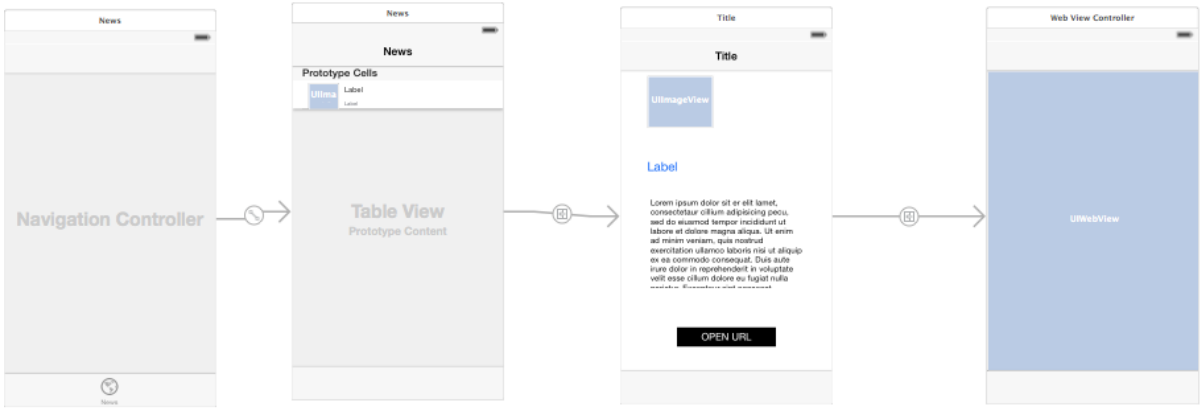
```
class qrViewController: UIViewController, QRCodeReaderDelegate {
        lazy var reader: QRCodeReader = QRCodeReader(cancelButtonTitle: "Cancel")

    @IBOutlet var label: UILabel!


    @IBAction func scanAction(sender: AnyObject) {
            reader.modalPresentationStyle = .FormSheet
            reader.delegate               = self

            reader.completionBlock = { (result: String?) in
                println(result)
            }

            presentViewController(reader, animated: true, completion: nil)
        }

        // MARK: - QRCodeReader Delegate Methods

        func reader(reader: QRCodeReader, didScanResult result: String) {
            self.dismissViewControllerAnimated(true, completion: nil)
            label.text = result //primer
        }

        func readerDidCancel(reader: QRCodeReader) {
            self.dismissViewControllerAnimated(true, completion: nil)
        }
}
```

Slika 4.11: Koda za branje QR kode iz mobilne kamere in uporaba knjižnice

Kot je prikazano na sliki, tekst na zaslon, ki se skriva za izbrano QR kodo, uporabniku preprosto prezentiramo s pomočjo ukaza »label.text = result«. Ob tem ko krmilnik zazna zapis QR kode, sproži funkcijo reader in shrani rezultat v spremenljivko **»result«** tipa string.

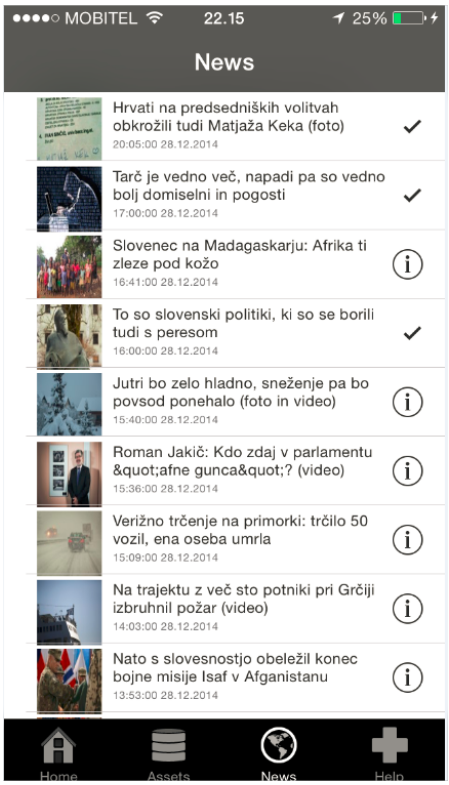
4.2.2.4 Prikaz novic

Second view krmilnik predstavlja prikaz novic za uporabnika. V primeru izdelave mobilne aplikacije za Apple store (spletni prostor za shranjevanje mobilnih aplikacij, ki so dostopne vsem uporabnikom) je zelo pomembna izkušnja uporabnika. Tukaj je namen prikaz novic, ki prihajajo iz podjetja, po navadi iz internega (javnosti nedostopnega) intraneta.



Slika 4.12: Prikaz klicev krmilnikov za prikaz novic iz spleta

Slika 4.12 prikazuje uporabo krmilnika za tablični prikaz. Pri tem je pomembno, da vsako celico sprogramiramo posebej, saj služi za prikaz želenih podatkov. Ob izbiri gumba News[2], uporabnik dobi zadnje novice iz intraneta. Ob kliku na novico mu aplikacija prikaže podrobnejše informacije o novici in uporabniku omogoči prikaz celotne novice znotraj mobilne aplikacije brez uporabe zunanjega brskalnika.



Slika 4.13: Primer prikaz novic na mobilni aplikaciji

[2] News – Novice – poimenovanje gumba za prikaz novic na mobilni aplikaciji, razviti za to diplomsko nalogo

4.2.2.5 Uporaba integriranega brskalnika

Webviewcontroller je krmilnik, ki skrbi za prikaz spletnih strani znotraj mobilne aplikacije. S to možnostjo uporabniku ni potrebno odpirati dodatne aplikacije, ki bi prikazovala spletno stran. Pri aplikaciji v okviru te diplomske naloge se krmilnik uporablja v primeru dostopa do novice na intranetu.

Ob kliku na novico s pomočjo vmesnika Segue[3] [11] prenesemo želeno stran v ta krmilnik. Krmilnik nato sproži zahtevo in uporabniku prikaže želeno stran.

Spodnja programska koda prikazuje preprostost uporabe krmilnika za prikaz želene strani. Nastavitev webView nam omogoča prenos zahteve iz krmilnika v pogled uporabniku.

```
override func viewDidLoad() {
        super.viewDidLoad()

                let url = NSURL(string: "http://maximointranet.int/novica/5")

                let request = NSURLRequest(URL: url)

        webView.loadRequest(request)
}
```

4.2.2.6 Čitalec črtne kode

BarCodeViewController se uporablja za inicializacijo kamere pri branju črtne kode. Naredil sem tako imenovani »most« med jezikoma Objective-C in Swift. To pomeni, da lahko knjižnice, ki so napisane v Objective-C jeziku, izvajaš tudi v Swift jeziku. Uporabil sem odprtokodno rešitev iz odprtokodnega spletnega naslova github, imenovano RSBarcodes_SWIFT [5].

```
override func viewDidLoad() {
    super.viewDidLoad()

    self.focusMarkLayer.strokeColor = UIColor.redColor().CGColor

    self.cornersLayer.strokeColor = UIColor.yellowColor().CGColor

    self.tapHandler = { point in
        println(point)
    }

    self.barcodesHandler = { barcodes in
        for barcode in barcodes {
            println(barcode)
        }
    }
}
```

[3] Vmesnik za prenos informacij iz pogleda v krmilnik

Swift znova prikazuje enostavno implementacijo funkcije za branje črtne kode. Koda zgoraj prikazuje nastavitev črte pri zaznavi črtne kode na rumeno barvo in s tem uporabniku olajša prepoznavanje črtne kode. V primeru, ko krmilnik zazna črtno kodo, jo spremeni v tekst in jo shrani v spremenljivko v pomnilniku »barcode«.

4.2.2.7 Graf

GraphViewController je krmilnik za prikaz grafa uporabniku. Tukaj lahko uporabnik na grafu preveri njegovo trenutno stanje glede na število potrjenih sredstev. Za izris grafa sem implementiral odprtokodno rešitev **JBChart** [6]**.**

4.2.2.8 Povezava z Maximo

Osrednji del aplikacije je initViewController, saj skrbi za povezavo do centralne aplikacije za upravljanje z osnovnimi sredstvi – Maximo APIja. V tem krmilniku aplikacija preveri trenutno stanje osnovnih sredstev uporabnika. Z uporabo REST ukaza GET [12] iz Maxima dobimo zadnji seznam osnovnih sredstev.

```
func searchItunesFor(searchTerm: String) {

    // The iTunes API wants multiple terms separated by + symbols, so replace spaces with + signs
    let itunesSearchTerm = searchTerm.stringByReplacingOccurrencesOfString(" ", withString: "+", options: NSStringCompareOptions.CaseInsensitiveSearch, range: nil)

    // Now escape anything else that isn't URL-friendly
    if let escapedSearchTerm = itunesSearchTerm.stringByAddingPercentEscapesUsingEncoding(NSUTF8StringEncoding) {
        let urlPath = "http://gts.ibm.si:9080/maxrest/rest/mbo/asset?_format=json&_compact=1&siteid=IBMSLOGT"
        let url = NSURL(string: urlPath)
        var session = NSURLSession.sharedSession()
        var request = NSMutableURLRequest(URL: NSURL(string: "http://gts.ibm.si:9080/maxrest/rest/os/mxperson?displayname=~eq~MAXADMIN&status=~eq~ACTIVE&_format=json")!)
        request.HTTPMethod = "GET"
        var params = ["username":"maxadmin", "password":"********"] as Dictionary<String, String>

        let username = "maxadmin"
        let password = "P@22w0rd"
        let loginString = NSString(format: "%@:%@", username, password)
        let loginData: NSData = loginString.dataUsingEncoding(NSUTF8StringEncoding)!
        let base64LoginString = loginData.base64EncodedStringWithOptions(nil)


        let urlConnection = NSURLConnection(request: request, delegate: self)

        var err: NSError?

        let config = NSURLSessionConfiguration.defaultSessionConfiguration()
        let authString = "Basic \(base64LoginString)"
        config.HTTPAdditionalHeaders = ["Authorization" : authString]
        session = NSURLSession(configuration: config)
        session.dataTaskWithURL(url!) {
            (let data, let response, let error) in
            if let httpResponse = response as? NSHTTPURLResponse {
                let dataString = NSString(data: data, encoding: NSUTF8StringEncoding)


                var jsonResult = NSJSONSerialization.JSONObjectWithData(data, options: NSJSONReadingOptions.MutableContainers, error: &err) as NSDictionary
                    if(err != nil) {
                        println("JSON Error \(err!.localizedDescription)")
                    }
                var results: NSDictionary = jsonResult["ASSETMboSet"] as NSDictionary
                var rez: NSArray = results["ASSET"] as NSArray

                dispatch_async(dispatch_get_main_queue(), {
                                self.tableData = rez

                                self.searchNews()
                            })
            }
            }.resume()
```

Slika 4.14: Prikaz implementacije povezave med mobilno aplikacijo in centralnim sistemom za upravljanje z mobilnimi aplikacijami

Kot je razvidno na sliki 4.14, do podatkov, ki jih potrebujemo za validacijo osnovnih sredstev, dostopamo preko spodaj navedenega spletnega naslova:

http://gts.ibm.si:9080/maxrest/rest/mbo/asset?_format=json.

JSON format je strukturiran po principu ključ+vrednost, kar pomeni, da je potrebna uporaba podatkovne strukture NSDictionary. Ta omogoča klicanje vrednosti po njenem ključu in obratno (podobno kot pri dostopu do sklada). Primer je zapis ASSETID: 123, kar nam omogoča, da lahko po klicanju ključa ASSETID dostopamo do vrednosti 123. Vse skupaj je zapisano v globalni tabeli **self.tableData**. Pomembna lastnost Swift-a je klicanje spremenljivke z uporabo predpone self. Ta nakazuje klic in uporabo globalne spremenljivke iz trenutnega razreda. S tem lahko ločimo lokalne in globalne spremenljivke.

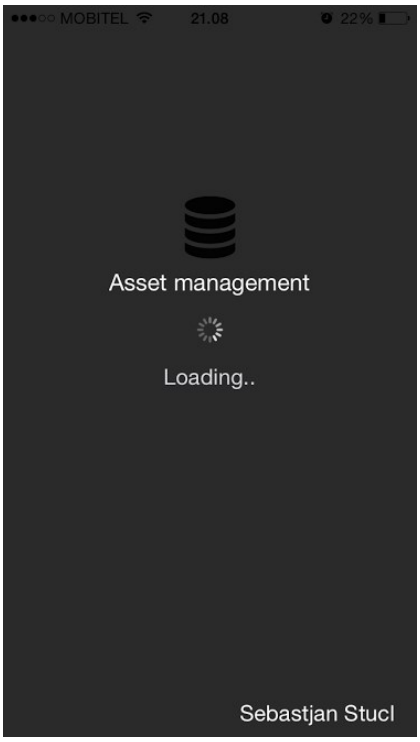
Druga pomembna funkcija krmilnika je zapis v podatkovno strukturo na mobilno napravo.

```
if results.count > 0
{

    for tab in self.tableData
    {
        for newAssets in results
        {
            if tab["SERIALNUM"] as String == newAssets.name as String
            {
                exists = 1
            }
        }
            if exists == 0
            {
                thisAsset = Asset(entity: ent, insertIntoManagedObjectContext: context)
                thisAsset.name = tab["SERIALNUM"] as String!
                thisAsset.picture = "http://accountancysage.com/wp-content/uploads/2014/04/assets.jpg"
                thisAsset.picture = tab["artworkUrl60"] as String!
                thisAsset.assetPicture = ""
                thisAsset.date = tab["INSTALLDATE"] as String!
                thisAsset.desc = tab["DESCRIPTION"] as String!
                thisAsset.desclong = tab["DESCRIPTION_LONGDESCRIPTION"] as String!
                thisAsset.location = tab["SALOCATIONDESC"] as String!
                thisAsset.newstatus = tab["NEWSTATUS"] as String!
                thisAsset.assetnum = tab["ASSETNUM"] as String!
                thisAsset.statusdate = tab["STATUSDATE"] as String!
                context.save(nil)
            i++
            self.loading.text = "Loading.. \(i)"
            }
        exists = 0
    }
```

Slika 4.15: Poizvedovanje po osnovnem sredstvu

Kot je že bilo omenjeno, se v tabelo ASSET zapišejo vse pomembne lastnosti izbranega sredstva. To so: slika, ime, inventarna številka, serijska številka, datumi, trenutno stanje sredstva in lokacija. V tem primeru se pot do slike sredstva zapiše kar v obliki URL naslova. Swift omogoča tudi zapis slike v obliki binarnega zapisa. Pomembno je bilo implementirati funkcijo, ki preveri, ali že zapis obstaja v bazi, saj Swift ne omogoča nadzora unikatnosti (enoličnosti) zapisov v bazi, ampak mora programer to nadzorovati sam.

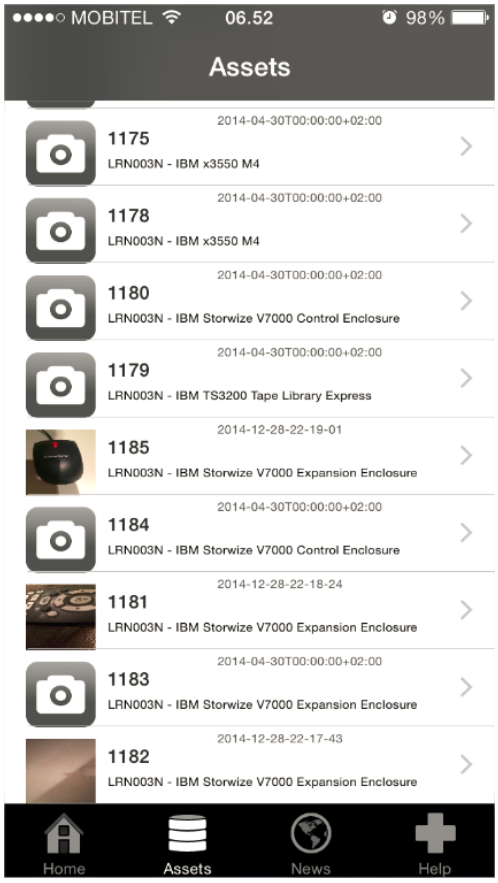
Zaradi **uporabniške izkušnje** (uporabniška izkušnja je področje, ki ukvarja z navadami uporabnikov in načini interakcije s programsko opremo [38]) se ob zagonu aplikacije naložijo vsi podatki o osnovnih sredstvih. Med nalaganjem podatkov se uporabniku prikaže stran, iz katere je jasno razvidno, da poteka nalaganje podatkov (slika 4.16).



Slika 4.16: Prikaz nalaganja podatkov ob zagonu mobilne aplikacije

4.2.2.9 Prikaz osnovnih sredstev

Krmilnik tabViewControllerAll skrbi za prikaz sredstev v obliki razpredelnice. Na mobilnih napravah je takšen način prikazovanja podatkov zelo pogost, saj uporabniku omogoča lažjo izbiro med podatki. Kot je razvidno iz spodnje slike, lahko uporabnik takoj preveri, katera sredstva so mu na voljo. V primeru, ko za sredstvo ni prepričan, kako izgleda, lahko iz seznama prav tako vidi sliko.



Slika 4.17: Prikaz seznama osnovnih sredstev

Ob kliku na izbran element v mobilni aplikaciji se sproži prenos podatkov iz enega v drug krmilnik. Ko uporabnik izbere svoje sredstvo, moramo podatke o sredstvu prenesti v drug krmilnik.

```
override func prepareForSegue(segue: UIStoryboardSegue, sender: AnyObject?) {
    if segue.identifier == "AssetDetails" {
        let controller: DetailViewController = segue.destinationViewController as DetailViewController
        let indexPath: NSIndexPath = self.tableView.indexPathForCell(sender as UITableViewCell)!

        //Fetch the data from CoreData
        let appDel = (UIApplication.sharedApplication().delegate as AppDelegate)
        let context = appDel.managedObjectContext
        let request = NSFetchRequest(entityName: "Asset")
        request.returnsObjectsAsFaults = false

        let results: NSArray = context?.executeFetchRequest(request, error: nil) as NSArray!


        let userData: Asset = results[indexPath.row] as Asset

        controller.nameString = userData.name
        controller.photoFullURL = userData.assetPicture
        controller.indexS = indexPath.row
        controller.assetDetail = userData

    }
}
```

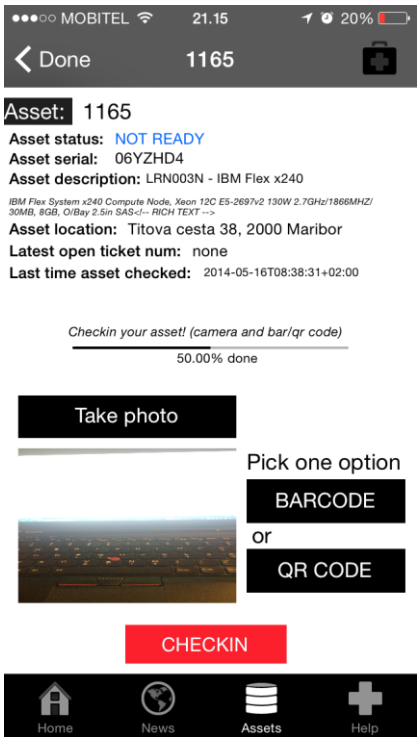
Slika 4.18: Prikaz prenosa podatkov med različnimi krmilniki

S pomočjo posebne funkcije prepareForSegue [13] lahko ta prenos izvršimo. Ob tem ko inicializiramo drug krmilnik v lokalno spremenljivko, nam ta omogoča, da shranimo podatke v krmilnik, v katerega prehajamo.

V našem primeru bomo v krmilnik DetailViewController prenesli vrednost indexS, ki vsebuje zaporedno številko izbrane celice.

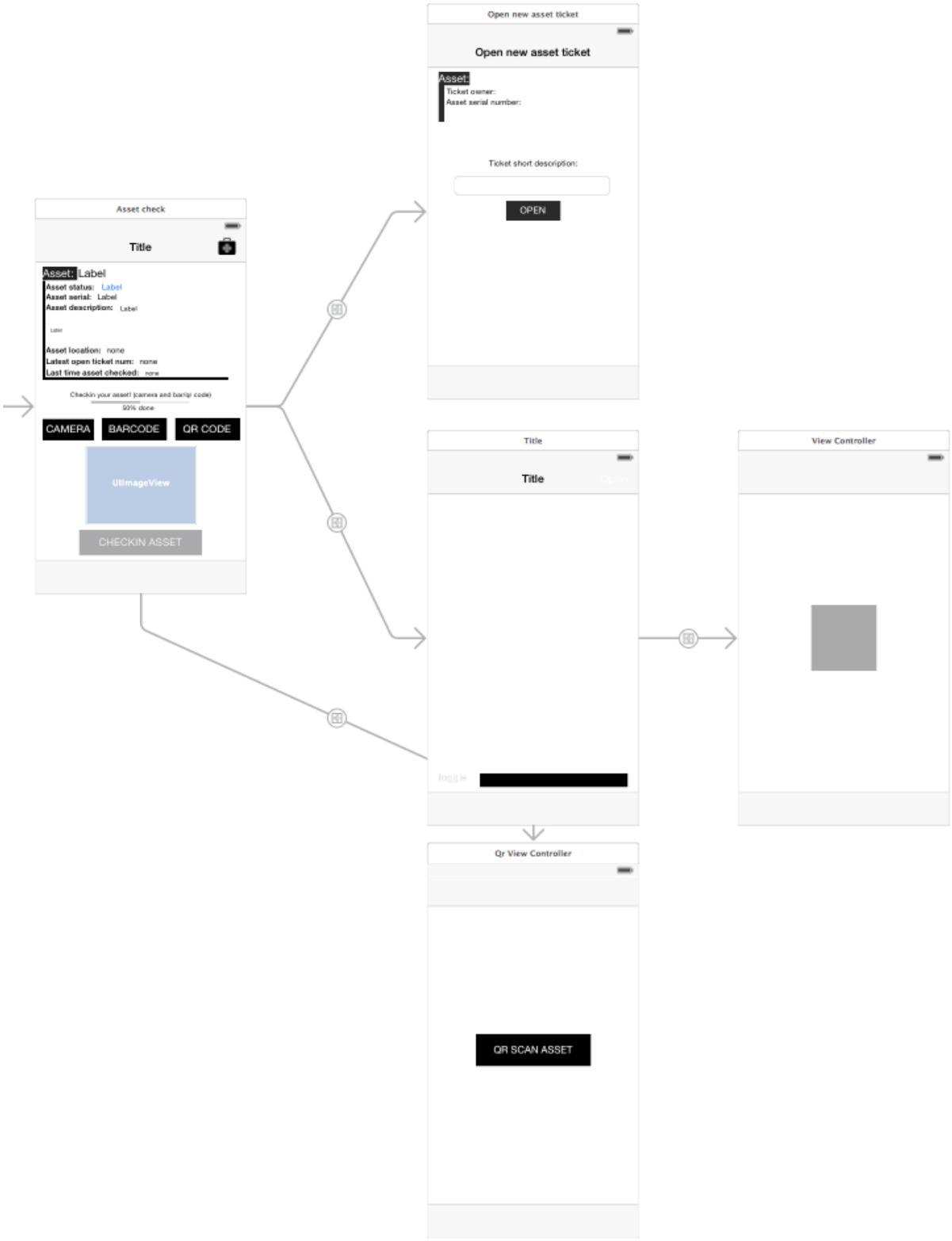
4.2.2.10 Vpogled v osnovno sredstvo

DetailViewController je krmilnik, ki skrbi za podroben prikaz podatkov o izbranem osnovnem sredstvu. Poleg tega povezuje štiri sisteme, in sicer sistem za zajem slike, sistem za branje QR kode, sistem za branje črtne kode ter sistem za prijavo napak na izbranem sredstvu. S pomočjo REST zahtevka se na Maximo API pošlje informacija o sredstvu in opis napake.



Slika 4.19: Prikaz podatkov o osnovnem sredstvu po poizvedbi iz sistema IBM Maximo

Kot prikazuje zgornja slika, vsak UIWeb gumb povezuje svoj krmilnik. Poleg gumbov je prikazan natančen opis sredstva in prikaz slike.



Slika 4.20: Prikaz klicev krmilnikov o posameznem osnovnem sredstvu in možnostih izbire akcije

Zgornja slika prikazuje povezave med posameznimi krmilniki in pogledi. Iz te slike je razviden tudi diagram poteka, kako uporabnik uporablja aplikacijo.

Pri implementaciji je bil pomemben tudi razvoj zajema slike, ki je potreben za podjetje pri pregledu sredstev. Spodaj je iz programske kode razvidno, da program zajame sliko in jo shrani točno pod določenim imenom. Ime vsake slike vsebuje točen datum in uro zajema te slike. To je pomembno za podjetje, saj lahko s tem preveri, kdaj je bila slika narejena. Na spodnji sliki je primer, kjer zapišemo v podatkovno bazo pot do slike. Slike ne shranimo v podatkovno bazo v obliki binarnega zapisa, temveč v galerijo slik uporabnika.. Naslov do slike se shrani v spremenljivko thisAsset.assetPicture.

```
var dateFormat = NSDateFormatter()
dateFormat.dateFormat = "yyyy-MM-dd-HH-mm-ss"
let now:NSDate = NSDate(timeIntervalSinceNow: 0)
let theDate: NSString = dateFormat.stringFromDate(now)
// Set URL for the full screen image
self.photoFullURL = NSString(format: "/%@.png", theDate)
// Save the full screen image via pngData
let pathFull: NSString = documentsDir.stringByAppendingString(self.photoFullURL!)
let pngFullData: NSData = UIImagePNGRepresentation(newImage)
pngFullData.writeToFile(pathFull, atomically: true)
var appDel = (UIApplication.sharedApplication().delegate as AppDelegate)
var context: NSManagedObjectContext = appDel.managedObjectContext!
var request = NSFetchRequest(entityName: "Asset")
request.returnsObjectsAsFaults = false
var results: NSArray = context.executeFetchRequest(request, error: nil) as NSArray!
//get contents and put into cell
var thisAsset: Asset = results[indexS!] as Asset
thisAsset.date = theDate
thisAsset.assetPicture = self.photoFullURL!
context.save(nil)
```

Slika 4.21: Prikaz kode za shranjevanje slik iz mobilne kamere

Pri zajemu slike lahko uporabimo osnovno knjižnico operacijskega sistem iOS, ki se imenuje AVFoundation [14]. Na spodnji sliki je primer, ki povezuje gumb takePhoto iz pogleda s funkcijo iz krmilnika za zagon kamere na uporabnikovi mobilni napravi. Uporaba je zelo enostavna, saj so funkcije že pripravljene za enostavno uporabo kamere na mobilni napravi. Za namen te aplikacije je uporabniku omogočeno, da ob zajemu slike prilagodi sliko oziroma naredi nov zajem slike. S tem mu prihranimo čas ob nepravilnem ali meglenem zajemu slike. To omogočimo s pomočjo metode allowsEditing [15].

```
@IBAction func takePhoto(sender: AnyObject) {
    if (UIImagePickerController.isSourceTypeAvailable(UIImagePickerControllerSourceType.Camera)){
        var picker = UIImagePickerController()
        picker.delegate = self
        picker.sourceType = UIImagePickerControllerSourceType.Camera
        var mediaTypes: Array<AnyObject> = [kUTTypeImage]
        picker.mediaTypes = mediaTypes
        picker.allowsEditing = true
        self.presentViewController(picker, animated: true, completion: nil)


    }
    else{
        NSLog("No Camera.")
    }

}
```

Slika 4.22: Primer uporabe kamere za zajem slike

Slika 4.22 prikazuje implementacijo metod za zajemanje slik s pomočjo kamere na mobilniku. Gre za uporabo knjižnice in standardnih metod za klic funkcije takePhoto, ki omogoča povezavo in preverjanje mobilne naprave ali sploh ima kamero. Kamera je bila pri razvoju te mobilne aplikacije osrednja funkcionalnost pri reševanju težave z letnim potrjevanjem osnovnih sredstev. Zajem slike nam zagotavlja avtentičnost prijave osnovnega sredstva, ker lahko preverimo uporabnika, lokacijo zajema in pristnost slike. S tem preprečujemo zlorabe in omogoča nam lažje upravljanje z osnovnimi sredstvi.

Poglavje 5 Sklepne ugotovitve

Največji izziv pri izdelavi diplomske naloge je bilo učenje novega programskega jezika Swift. Ob učenju programskega jezika in razvoju mobilne aplikacije sem dobil potrditev, da mi zaradi poznavanja drugih programskih jezikov in znanja pridobljenega na fakulteti učenje novega visokonivojskega programskega jezika ne predstavlja težav [24].

Ugotovitev diplomske naloge je med drugim tudi, da lahko s pomočjo relativno preproste mobilne aplikacije pomagamo podjetju pri učinkovitejšem upravljanju z osnovnimi sredstvi, in sicer na področju izvajanja inventure. Mobilna aplikacija razvita v okviru diplome ima štiri glavne funkcije: popis osnovnih sredstev s pomočjo kamere, prijava napake na osnovnem sredstvu, klic v sili in prikaz zadnjih novic iz strani podjetja.

Pristop izvajanja inventure preko mobilne aplikacije podjetju olajša spremljanje njihovih osnovnih sredstev. Podjetje ima z mobilno aplikacijo možnost pridobivanja dodatnih informacij, kot so na primer lokacija osnovnega sredstva, vlaga v okolici osnovnega sredstva, vreme v okolici osnovnega sredstva, GPS lokacija sredstva, pogostost napake na osnovnem sredstvu itd. Z vsemi naštetimi dodatnimi informacijami lahko podrobneje analiziramo osnovno sredstvo in z večjo verjetnostjo napovemo njegovo okvaro.

Literatura

[1] A Swift Tour. Dostopno na: https://developer.apple.com/library/ios/documentation/Swift/Conceptual/Swift_Programming_Language/GuidedTour.html#//apple_ref/doc/uid/TP40014097-CH2-ID1

[2] (2012) Integration Framework. Dostopno na: https://www.slideshare.net/sunk818/2012-07-17steintegrationframework

[3] (2011) RESTing with Maximo. Dostopno na: https://www.ibm.com/developerworks/community/blogs/a9ba1efe-b731-4317-9724-a181d6155e3a/entry/resting_with_maximo1?lang=en

[4] QRCodeReader. Dostopno na: https://github.com/yannickl/QRCodeReader.swift

[5] RSBarcodes_Swift. Dostopno na: https://github.com/yeahdongcn/RSBarcodes_Swift

[6] JBChartView. Dostopno na: https://github.com/Jawbone/JBChartView

[7] Sistemska arhitektura in komponente. Dostopno na: https://www.ibm.com/support/knowledgecenter/sl/SSLKT6_7.5.0.5/com.ibm.mam.doc/mam_install/c_mam_components.html

[8] Maximo Asset Management. Dostopno na: https://www-03.ibm.com/software/products/sl/maximoassetmanagement

[9] M. P. Papazoglou, W. J. van den Heuvel, "Service oriented architectures: approaches, technologies and research issues", *The VLDB Journal*, št. 16, str. 389-415, mar. 2007

[10] Peter Stegnar, *Razvoj spletnih aplikacij z vzorci MVC na strežniku in odjemalcu*, diplomsko delo na UL FRI, 2013, pogl. 3

[11] Using Segues. Dostopno na: https://developer.apple.com/library/content/featuredarticles/ViewControllerPGforiPhoneOS/UsingSegues.html

[12] Representational state transfer. Dostopno na: https://en.wikipedia.org/wiki/Representational_state_transfer

[13] prepareForSegue. Dostopno na: https://developer.apple.com/reference/uikit/uiviewcontroller/1621490-prepareforsegue

[14] AVFoundation. Dostopno na: https://developer.apple.com/reference/avfoundation

[15] allowsEditing. Dostopno na: https://developer.apple.com/reference/uikit/uiimagepickercontroller/1619137-allowsediting

[16] Xcode. Dostopno na: https://developer.apple.com/Xcode/

[17] Dokumentacija programskega jezika SWIFT (Apple). Dostopno na: https://developer.apple.com/swift/

[18] Mohit Deshpande, *Swift Programming for Human Beings*. Dostopno na: https://swiftludus.org/free-ebook-swift-programming/

[19] Nick Smith, *Programming Swift! Swift 2* , AppSmith Books, 1st edition, sep. 2015

[20] Nick Smith, *Programming Swift! Mac Apps 1* , AppSmith Books, 2016

[21] Opis ISO 55000 standarda za upravljanje z osnovnimi sredstvi. Dostopno na: http://www.assetmanagementstandards.com/iso-55000-standards-for-asset-management/

[22] ISO 55000. Dostopno na:https://www.iso.org/obp/ui/#iso:std:iso:55000:ed-1:v2:en

[23] Applying the Guidelines of BSI PAS 55 to Facility Management. Dostopno na: http://reliabilityweb.com/articles/entry/applying_the_guidelines_of_bsi_pas_55_to_fac/

[24] Visokonivojski programski jezik. Dostopno na: https://sl.wikipedia.org/wiki/Visokonivojski_programski_jezik

[25] (2005) Osnovno sredstvo ali drobni inventar (Barbara Fekonja univ.dipl.ekon). Dostopno na: http://www.racunovodja.com/clanki.asp?clanek=703/Osnovno_sredstvo_ali_drobni_inventar

[26] (2016) Kako pa vi obračunavate amortizacijo? Dostopno na: http://bankazapodjetnike.si/novice/kako-pa-vi-obracunavate-amortizacijo/

[27] (2008) Amortizacija. Dostopno na: http://www.eracunovodstvo.org/blog/racunovodstvo/amortizacija/

[28] Žarko Rankov, *Poštena vrednost opredmetenih osnovnih sredstev,* diplomsko delo na UL EF, 2009, pogl. 4.

[29] Glosar. Dostopno na: http://www.rs-rs.si/rsrs/rsrs.nsf/Glosar?OpenForm

[30] IBM Maximo Asset Management. Dostopno na: http://www.troia.si/produkti-resitve/ibm-maximo-asset-management

[31] Tatjana Černač Stupar, *Dolgoročna sredstva in drobni inventar,* GZS Ljubljana, 2011, pogl. 3.

[32] PANTEON Business operating system. Dostopno na: ftp://ftp.datalab.si/Marketing/Brosure/SI_Pantheon_print.pdf

[33] (2014) Raja Pleci, Overview of Maximo Industry Solutions and Extensions. Dostopno na: http://rajapleci.blogspot.co.uk/2014/05/overview-of-maximo-industry-solutions.html

[34] I. Woodbury, »Amping your enterprise: Realising the benefits of asset management«, *The ASSET Journal*, št. 10, zv. 4, str. 20-24, 2016

[35] Model-view-controller (wikipedia). Dostopno na: https://en.wikipedia.org/wiki/Model–view–controller

[36] Boris Šavc, *Izdelava aplikacije iOS*, revija Monitor, september 2016, str. 62

[37] (2010) Predpisi, ki urejajo letni popis (popis, inventura). Dostopno na: https://www.racunovodja.com/clanki.asp?clanek=2955/Predpisi_ki_urejajo_letni_popis_popis_inventura

[38] User experience. Dostopno na: https://en.wikipedia.org/wiki/User_experience

[39] (2015) Asset Management PAS 55/ISO 55000. Dostopno na: http://www.maintenanceonline.co.uk/article.asp?id=9813