UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Jernej Logar

RFID implementacija sledenja v preskrbovalni verigi

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mira Trebar

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01688/2010
Datum: 01.09.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **JERNEJ LOGAR**

Naslov: **RFID IMPLEMENTACIJA SLEDENJA V PRESKRBOVALNI VERIGI**
RFID IMPLEMENTATION OF TRACEABILITY IN A SUPPLY CHAIN

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Kandidat naj preuči standarde za področje sledenja živil v preskrbovalni verigi. Na osnovi podanih zahtev sledljivosti naj z uporabo tehnologije RFID in specifikacij EPCglobal definira in uporabi EPC Information Service (EPCIS) za shranjevanje in izmenjavo podatkov o označenih izdelkih. Zasnuje naj aplikacijo za zajem podatkov in njihovo predstavitev na informacijskem kiosku, kjer ima kupec dostop do podatkov zbranih v postopku priprave, transporta in prodaje živila. Predlagano rešitev naj izvede za primer sledenja gojenih rib od mesta ulova pa vse do prodaje v ribarnici.

Mentor:
Mira Trebar
doc. dr. Mira Trebar

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

IZSTAVITEV

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani Jernej Logar,

z vpisno številko 63000205,

sem avtor diplomskega dela z naslovom:

RFID implementacija sledenja v preskrbovalni verigi

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom
  doc. dr. Mire Trebar,

- 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 v zbirki »Dela FRI«.

V Ljubljani, dne 13.12.2010. Podpis avtorja:

Zahvala

Za strokovno pomoč, hiter odziv na vprašanja in nasvete pri pisanju diplomske naloge se zahvaljujejm mentorici doc. dr. Miri Trebar.

Posebna zahvala gre družini, ki mi je bila tekom celotne študijske poti v vsestransko oporo.

Kazalo

Seznam uporabljenih kratic in pojmov

DI **Dependency Injection**	Vstavljanje odvisnosti
Domain Model	Model domene
EPC Electronic Product Code	Elektronska koda izdelka
EPCIS **Electronic Product Code Information Services**	EPC informacijske storitve
IDE Integrated Development Environment	Integrirano razvojno okolje
IoC Inversion of Control	Inverzija nadzora
Microsoft .NET Framework	.NET ogrodje za razvoj programske opreme
NHibernate	Odprto kodni ORM
ORM Object Relational Mapper	Preslikovalnik med objekti v objektno orientiranem programskem jeziku in relacijskim modelom
RFID **Radio Frequency Identification**	Identifikacija z radijskimi valovi
RFID tag	RFID značka
SQLite	Lahka relacijska podatkovna baza na osnovi datotek
TCP/IP	Standardiziran sklad protokolov, na katerem temelji internet
TraceCore XML	XML format za zajem in izmenjavo podatkov o sledljivosti med vpletenimi v preskrbovalni verigi
UID Unique IDentifier	Enolični identifikator
WPF Windows Presentation Fundation	Ogrodje za razvoj uporabniških vmesnikov na Windows sistemu
XML eXtensible Markup Language	Razširljivi označevalni jezik

Povzetek

Sledenje v preskrbovalni verigi v zadnjem času vse bolj pridobiva na veljavi. Izdelanih je že veliko pilotskih projektov in tudi realnih izvedb, predvsem na področju logistike, kjer obstajajo številni primeru sledenja tudi v vsakodnevni uporabi.

Diplomsko delo vsebuje splošen pregled področja, zahtev in zakonskih okvirov ter industrijskih standardov, ki se uveljavljajo na tem področju. Opisane so tudi tehnologije, ki so na voljo za izvedbo informatiziranega sistema za sledenje, predvsem značilnosti RFID tehnologije. Predstavljeni so standardi za izmenjavo podatkov med posameznimi členi verige, s poudarkom na EPCglobal EPCIS standardu.

Obravnavan je postopek načrtovanja in implementacije povezanega sistema za sledenje živil v preskrbovalni verigi. Sistem je sestavljen iz štirih delov. Dva v sistem dodajata podatke o izdelkih označenih z RFID značkami, tretji te podatke streže, četrti pa predstavlja odjemalca, kot je na primer kupec. Zajeto je snovanje modela podatkov s pripadajočo poslovno logiko, načrtovanje programske arhitekture, uporaba strojne opreme za branje RFID značk in implementacija komunikacije med posameznimi deli sistema po EPCIS standardu za izmenjavo podatkov. Za praktični prikaz branja RFID značk je opisan Feig i-scan UHF MRU200i čitalnik. Uporabljena so programska orodja, ki so na voljo za izvedbo takega sistema v Microsoft .NET ogrodju. Končna izvedba je prirejena in testirana za primer sledenja gojenih rib vse od trenutka ulova pa do prodajnega mesta, kjer lahko kupec spremlja pridobljene podatke.

Ključne besede: RFID, EPCglobal, EPCIS, sledljivost, preskrbovalna veriga, .NET

Abstract

Tracking in supply chains is gaining in importance lately. Many pilot projects have been made and also real implementations. The field of logistics is in the forefront with many systems deployed into everyday use already.

A review covers a general overview of the problem area, requirements and laws and industrial standards that are being established in the area. Described are the technologies available for the implementation of an information system for tracking, mainly the properties of RFID technology and standards for data exchange between separate links of the supply chain with emphasis on the EPCglobal EPCIS standard.

The thesis presents procedure of designing and implementing a connected system for tracking in the supply chain. The system consist of four parts. Two of them add data about products marked with RFID tags into the system. The third serves the data and the fourth represents the client, a buyer of the product for example. The thesis includes the creation of the data model with the respective business logic, designing of the software architecture, the use of hardware components for reading RFID tags and the implementation of the communication using the EPCIS standard for data exchange. Feig i-scan UHF MRU200i reader is used for a practical demonstration of reading RFID tags. The final implementation is designed and tested for an example of farmed fish from the time of catch to the point of sale where a consumer can access the supply chain data.

Key words: RFID, EPCglobal, EPCIS, tracking, supply chain, .NET

1 Uvod

Za moderno potrošniško družbo je značilno, da se hrana in druge dobrine dobavljajo iz oddaljenih krajev in preko cele vrste posrednikov. S tem se povečuje tudi možnost, da se v času dostave do kupca izdelek pokvari oziroma se z njim rokuje na nedovoljen ali nepredviden način. Transport lahko traja dalj kot predvideno, na vsakem koraku lahko pride do vmešavanja s strani tretje osebe. Tako prihaja do potrebe po sledenju izdelka od njegovega izvora do prodaje končnemu potrošniku.

Hkrati se med kupci in inštitucijami na področju prehrane povečuje zavest o varni hrani, kar vključuje zagotavljanje podatkov o poreklu vseh sestavin, o poti, ki jo je izdelek prepotoval in kako je bil skladiščen. Zato se na vsakem koraku pričakuje nadzor nad kakovostjo izdelkov in surovin, blago ne sme biti prestaro, hranjeno mora biti pod točno določenimi pogoji, vsak korak v verigi dobaviteljev mora biti pregleden in podobno. To nas privede do sledljivosti izdelkov v preskrbovalnih verigah in k tehnologijam, ki nam to omogočajo.

V ta namen se že dalj časa uporablja različne metode sledenja v preskrbovalni verigi. Že pred časom so ročno obdelavo s popisom v knjige zamenjali računalniški informacijski sistemi. Sam vnos pa je podprt z različnimi tehnologijami, največkrat s črtnimi kodami. Naslednji korak predstavlja radiofrekvenčna identifikacija RFID (ang. Radiofrequency identification), ki je sicer na voljo že dalj časa, vendar vse do pred nekaj leti ni prišla v širšo uporabo. Z razvojem tehnologije in predvsem vse nižjimi cenami RFID značk bo postalo označevanje posameznega izdelka z RFID značko in s tem sledenje od proizvajalca do končnega uporabnika ekonomsko uresničljivo.

Za vsak tak tehnološki prehod so potrebne prilagoditve, ki niso majhen zalogaj za podjetja. Take prilagoditve se zato ponavadi uveljavijo šele, ko jih kot obvezne uvede kakšno veliko podjetje ali inštitucija. V zvezi z RFID sta tako največkrat omenjena Wal-Mart [3] in obrambno ministrstvo ZDA [24], ki sta od vseh svojih dobaviteljev zahtevala uporabo RFID značk na pošiljkah. Tudi sicer je v svetu že kar nekaj projektov, ki kažejo na tehnično in ekonomsko izvedljivost takega načina sledenja. Izkazalo se je že, da se sledljivo blago v nekaterih primerih bolje prodaja [26].

Cilj diplomskega dela je z uporabo trenutno dosegljivih tehnologij in metod razviti informacijski sistem, ki omogoča vnos podatkov o gojenih ribah v podjetju in prenos teh podatkov k drugim členom v verigi ter do končnega kupca preko računalnika z zaslonom na dotik in čitalnikom RFID značk. Podatki vljučujejo vzreditelja, vrsto ribe, težo posameznega primerka in drugo. Velik poudarek je na uporabi obstoječih standardov z mednarodno veljavo.

Rezultate tega dela se bo lahko uporabilo tudi v okviru evropskega projekta »RFID from Farm to Fork« [20], ki se trenutno izvaja na Univerzi v Ljubljani. Eden izmed namenov projekta je izvedba pilota za vpeljavo RFID tehnologije v preskrbovalni verigi. V pomoč bi lahko bile

predvsem izkušnje pridobljene z uporabo standardov na področju sledenja v preskrbovalni verigi.

2 Sledenje v preskrbovalni verigi

V preskrbovalni verigi se srečamo z izrazoma sledljivost in sledenje. Izraza imata različen pomen, zato je pomembno, da ju opredelimo.

»Sledljivost je zmožnost ugotoviti pretekle ali trenutne lokacije predmeta ter poznati njegovo zgodovino.« [13]

Razširjeno lahko sledljivost opredelimo kot zmožnost, da v kronološkem vrstnem redu ugotovimo kaj se je dogajalo s posamezno enoto, ki ji sledimo, in popolnost podatkov o vsakem koraku v procesni verigi. Omogoča nam, da lahko za vsak trenutek izvemo kje se je določena enota nahajala.

Sledenje pa je proces zajemanja in shranjevanja podatkov o dogodkih povezanih s posamezno enoto v preskrbovalni verigi.

Sledljivost in sledenje sta lahko izvedena bodisi analogno v obliki zapisov na papirju ali digitalno z zapisi v informacijskem sistemu. Seveda pa tu informacijski sistem ponuja enostavno obvladovanje večje količine podatkov, veliko večjo natančnost in lažjo analizo pridobljenih podatkov, vključno z različnimi načini iskanja. To pa je nekaj, kar je z ročnim prebiranjem arhivov praktično nemogoče.

Dogodki, ki jih običajno beležimo:

- menjava lastnika,
- sprememba stanja,
- združevanje v serije, pakete, na palete,
- razdruževanje.

2.1.1 Sledenje v prehrambeni verigi

S prehrambeno verigo imamo v mislih pot prehrambenega izdelka od virov surovin (kmetije, farme, ribogojnice ipd.) preko predelovalcev (klavnice, predelovalni obrati, ribarnice itd.) in transporta (logistični centri, skladišča ipd.) do prodajnih mest ter končnih kupcev. Vse to skupaj povzemamo z izrazoma »od vil do vilic« ali »od njive do mize« (ang.»Farm to Fork«).

Sledenje v preskrbovalni verigi je zelo pomembno iz naslednjih razlogov:

- **nadzor nad kakovostjo**
  Sledljivost nam omogoča pregled nad vsemi uporabljenimi sestavinami, o pogojih hranjenja in prevoza ter drugimi podatki. Institucije zdravstvenega varstva in nadzorni organi imajo tako vpogled v podatke, ki so nujni za nadzor kakovosti.
- **preklic pokvarjenih ali škodljivih živil**
  V primeru, da se za živilo v prodaji odkrije neprimernost za končnega uporabnika, je odpoklic teh živil iz prodaje veliko lažji in bolj dosleden. Za vse izdelke se točno ve kam in od kod so bili odposlani.

- **odkrivanje in preprečevanje ponarejanja izdelkov**
  Problem za lastnike znanih trgovskih znamk in njihove stranke je tudi ponarejanje. Zaradi uveljavljenega imena imajo višjo ceno. Enako velja za ponarejene izdelke. Sledljivost in enolična identifikacija, ki jo prinaša RFID značka na vsakem artiklu, se lahko uporabita za odkrivanje in preprečevanje ponaredkov.
- **komunikacija s končnim uporabnikom**
  Proizvajalec lahko kupcu poleg podatkov o hrani sporoči tudi dodatne koristne informacije glede izdelka in njegove uporabe, kot na primer nasvete za uporabo in hranjene, recepte ali informacije o drugih povezanih izdelkih. Za proizvajalce je to lahko tudi priložnost za krepitev svojih znamk.

2.1.2 Sledenje gojenih rib

Ker so ribe precej občutljivo in hitro pokvarljivo živilo, je sledljivost na tem področju potrebna in še toliko bolj zaželjena.

Prva odločitev pri sledenju rib (in tudi pri ostalih živalih) je ali rabimo podatke že o gibanju žive ribe ali šele v času, ko je bila riba zajeta. Žive ribe označujemo kadar želimo slediti podatke o njihovi lokaciji (v katerem bazenu ali kletki so se nahajale) v določenih obdobjih v času rasti. Podatke o lokaciji potem povežemo s temperaturo vode na teh lokacijah, z vrsto krmil in vrsto zdravil, ki so jih ribe dobile. Kadar se v živilih pojavijo sestavine v količinah nad dovoljenimi, je pomembno, da je vedno zabeleženo, katere vrste zdravil ali dodatkov so gojene ribe prejele. V zvezi s tem so predvsem v živinorejski in mlekarski industriji na tem področju znani škandali. Če je sledenje urejeno na zgoraj opisani način, potem je v primerih nepravilnosti preprosto ugotoviti kako je do nepravilnosti prišlo in na katere ribe je imela nepravilnost vpliv.

Vsako ribo ali bolj pogosto zaboj z ribami se lahko opremi s pasivno RFID značko, ki se jo prebere ob prehodih skozi senzorje postavljene na strateške lokacije, kot so na primer vhodi v hladilnice, nakladalna mesta, razkladalna mesta ipd. Tako se zabeleži vse dogodke, ki so pomembni v okviru poti od ribogojnice do končnega porabnika.

Ribe se ponavadi označuje na enega izmed sledečih načinov:

1. RFID značke v obliki implantanta (slika 1) se vbrizga (injekcija na sliki 2) v ribje mladice. Implantat je podolgovate oblike in običajno velik približno toliko kot večje zrno riža in ribe, razen v redkih primerih, ne moti. Običajno je vbrizgavanje implantantov precej zamudna naloga, saj je treba ribe za določen čas prestaviti v poseben bazen, ki vsebuje raztopino sredstev proti infekcijam. Ribam se nato naredi majhen zarez v kožo na trebušnem predelu in injecira RFID značko. Ribe se potem vrnejo v bazen in šele čez določen čas, ko se rana zaceli, se lahko vrnejo v običajno kletko.

Slika 1. Biomark TXP RFID značka [4].

Slika 2. Biomark MK7 »injekcija« za RFID značke [4].

2. Manj invazivne so značke, ki se samo zataknejo na ribo (slika 3). Tudi te so namenjene za uporabo na živih ribah. Značke so podolgovate oblike in imajo na delu, ki se zagozdi v ribje telo, zatič oblike T ali kavlja. Take značke se precej lažje in hitreje nanašajo, vendar so namenjene večjim oziroma odraslim ribam.

Slika 3. Hallprint PDXL značka [14].

3. RFID značke, ki so pripete na ribo in se pripnejo na ulovljeno oziroma mrtvo ribo. Načini pripenjanja so različni (obesek, s pomočjo žice skozi škrge ipd).
4. RFID značke na plastičnih zabojih ali škatlah, ki vsebujejo večjo količino rib. S tem se nekoliko zmanjša cena označevanja in poveča tudi zanesljivost branja značk, saj se bere neprimerno manjša količina značk. Seveda pa to ne zagotavlja sledenja posameznih rib v škatlah. V večjih pristaniščih in na ribjih dražbah se uporabljajo večje plastične škatle z vgrajenimi RFID značkami (slika 4), pri manjših uporabnikih, kot je npr. Fonda d.o.o. pa se uporabljajo navadne stiroporne izolacijske škatle namenjene transportu (slika 5), na katere se lahko nalepi RFID značka v obliki samolepilne etikete. Kot je opisano spodaj, se lahko ribam dodaja tudi aktivne RFID značke z namenom sledenja lokacije in temperature hranjenja v verigi (slika 5).

Pri ribah je zelo pomembno, da so ves čas shranjene na primerni temperaturi, zato se pogosto pri sledenju zajema tudi temperatura. V ta namen se uporablja aktivna RFID značka, ki ima tudi svoj spomin v katerega beleži temperaturo v enakomernih časovnih razmakih. Značka se običajno vstavi v škatle oziroma zabojčke v katerih so večje količine rib, saj so ti podatki vezani na celotno pošiljko rib. Če ima značka dovolj spomina, lahko

tako spremljamo tudi daljše transporte rib (na primer prekooceanske) in imamo zagotovljen nadzor kakovosti dostavljenih rib.

Slika 4. Pack and sea škatla z vgrajeno RFID značko [19].

Slika 5. Običajna stiroporna škatla z dodano aktivno RFID značko.

Tovrstno zbiranje podatkov se je že izkazalo za učinkovito v okviru MainSafeTrace projekta [10]. Sam projekt se ukvarja z razvojem in testiranjem namenskega sistema za sledenje v preskrbovalni verigi skuš nalovljenih v morju okrog Norveške in prodanih na Japonskem. Preizkušali so uporabnost RFID temperaturnih senzorjev pri prevozu skuš na Japonsko. Edina resna ovira je zanesljivost senzorjev, saj je bilo po transportu, ki je trajal okrog 60 dni, pri -27°C do -23°C možno prebrati le še 17 od 24 značk. Vendar tudi na tak način lahko

zagotovimo dovolj dober nadzor nad razmerami pri prevozu, kvaliteta aktivnih RFID značk pa se tudi izboljšuje, tako da lahko v prihodnosti pričakujemo boljše rezultate.

Sledenje v kombinaciji z GPS tehnologijo

V primeru, da imamo na določenem delu poti kako vidno odstopanje od zahtevanih temperatur, nam to v kombinaciji z GPS sledenjem omogoča natančno analizo podatkov in sklepanje kje in zakaj so težave nastale. Posledično se lahko tem težavam v prihodnje izognemo in s tem seveda tudi zmanjšamo stroške.

2.1.3 Obdelava gojenih rib v Fonda d.o.o.

Ob vpeljavi RFID tehnologije in EPCIS sistema bomo obdržali delovni proces podoben kot je bil prej, ne da bi ga spreminjali in s tem vnašali zmedo v delovanje podjetja. Običajno se že prehod na uporabo nove tehnologije izkaže za dovolj zahtevnega, brez spreminjanja samega procesa.

V podjetju Fonda d.o.o. uporabljajo ročni zajem in obdelavo podatkov. Najprej sprejmejo naročilo od stranke (preko telefona, elektronske pošte ali telefaksa) in ga vpišejo v zvezek. Naročilo vsebuje podatke o stranki, količine in teže rib, željeno obdelavo (filirane, očiščene, v soli). Po dvigu iz kletk se ribe pri sortiranji zopet popišejo v zvezek, nato pa se vnesejo še dobavnice in računi v računovodski sistem.

Če bi bil projekt širšega obsega, bi hkrati lahko izvedli še integracijo z obstoječim obračunskim sistemom v podjetju. Stična točka bi lahko bila kar izdelava naročilnice ali računa iz označene škatle z ribami.

V obstoječem postopku je precej mest, kjer bi lahko vstavili točke sledenja z uporabo RFID značk.

1. Ob dvigu rib iz kletk. Predpogoj tu je, da so ribe označene že kot mladice, ali, da se označijo sproti. Sprotno označevanje pa verjetno ni sprejemljivo, ker na ladji ni kapacitet za označevanje, niti časa.
2. Ob raztovarjanju v pristanišču. Podoben pogoj, kot pri točki 1, vendar imamo tu na voljo prostor in smo manj časovno omejeni.
3. Pri sortiranju in filiranju. Na tem mestu se že sedaj vpisujejo podatki v zvezek. Zato je to mesto najboljša izbira za točko sledenja.

Za potrebe pilotnega projekta bomo postavili točko sledenja le na eno mesto in sicer ob sortiranju rib v zabojčke za stranke.

2.2 Splošne zahteve in pravila na področju EU

Zakoni v EU definirajo sledljivost, kot

»...zmožnost, da sledimo katerikoli hrani, krmi, živali za hrano ali substanci, ki se bo zaužila, skozi vse stopnje proizvodnje, predelave in distribucije.« [8]

Vidik, ki ga ta zakonodaja zavzema, je predvsem varnost pri prehrambenih izdelkih in zmanjševanje nevarnosti za porabnike. Zakonodaja na področju EU od leta 2005 naprej predpisuje takoimenovano »one step back - one step forward« zahtevo. Na ta način mora vsak člen v preskrbovalni verigi imeti informacije o tem od kod je blago prišlo in kam je šlo.

Cilji te zakonodaje so v primeru nevarnosti izdelkov za porabnike:

- Takojšnji odpoklic prizadetih izdelkov s tržišča in, če je to potrebno, tudi od porabnikov.
- Uničenje celotne serije, pošiljke krme, ki ne zadosti zahtevam po varni hrani.

Seveda je tu še veliko prostora za izboljšave, predvsem v smeri večje preglednosti in približevanja vseh zbranih informacij kupcem. Veliko pa bi lahko od tega imela tudi podjetja, ki bi lažje informatizirala svoje poslovanje.

2.3 Industrijski standardi in smernice za sledenje

Kot v vsaki industriji so se tudi pri preskrbovalnih verigah že oblikovali določeni standardi. Kot krovna in neprofitna organizacija se za standardizacijo in izboljšavo v preskrbovalnih verigah pojavlja GS1. Ustanovljena leta 1977, se *»zavezuje k izdelavi in implementaciji globalnih standardov in rešitev za izboljšavo učinkovitosti in transparentnosti preskrbovalnih verig globalno in v različnih sektorjih«* [12].

2.4 Tehnologije označevanja izdelkov

Če hočemo nekemu objektu slediti v preskrbovalnih verigah skozi različne organizacije in okolja, mu moramo določiti nek enolični identifikator. Prav tako moramo ta identifikator na nek način vsakič prebrati, za kar so nam na voljo različni načini označevanja.

2.4.1 Identifikatorji

Identifikator je enoličen izraz, ki določa identiteto predmeta. Pri vseh izmenjavah in nadzorih je potrebno imeti neke vrste enolični identifikator, ki nam omogoča, da vsak opazovan predmet enolično označimo in identificiramo.

Predstavlja povezavo med fizično spremljano enoto, torej izdelkom z značko, in njeno podatkovno predstavitvijo. S pomočjo identifikatorja ob vsakem srečanju objekt prepoznamo in mu pripišemo ustrezne atribute ter zabeležimo dogodke povezane z njim.

Če je identifikator globalno enoličen, dobimo povezavo med predstavitvijo predmeta v različnih sistemih, ki so lahko med seboj zelo različni, saj največkrat služijo popolnoma različnim namenom. Primer dveh takih sistemov sta sistem za računovodstvo in sistem za

vzdrževanje voznega parka. V sistemu za računovodstvo nas za nek predmet oziroma njegovo predstavitev zanimata njegova vrednost in njeno padanje skozi čas, medtem ko nas pri vzdrževanju zanima bolj koliko kilometrov ima neko vozilo še do naslednjega servisa.

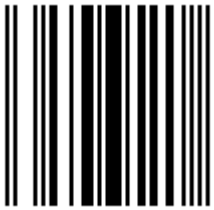
Med standardi za identitifkatorje je najbolj uveljavljen GTIN (Global Trade Item Number). Oblikovala ga je organizacija GS1. Vsebuje tudi EAN (European Article Number, kasneje preimenovan v International Article Number, vendar se je kratica ohranila) in UPC (Universal Product Code) standarda, ki ju uporabljata enako imenovana standarda za črtne kode omenjena spodaj. Razširjen v SGTIN (Serial Global Trade Item Number), se GTIN lahko uporablja tudi v EPC (Electronic Product Code) standardu. GTIN sam namreč označuje le vrsto izdelka, SGTIN v EPC pa služi za identifikacijo vsakega izdelka.

2.4.2 Črtna koda

Prvi način označevanja objektov ali vsaj vrst le-teh se je pričel razvijati okrog leta 1948 za potrebe branja podatkov o izdelkih pri prodajnem pultu oziroma pri plačilu in nekoliko kasneje pri prepoznavanju vagonov. Tekom zgodovine so se uporabljali različni načini kodiranja identifikacijskih podatkov, od raznobarvnih odbojnih nalepk, preko koncentričnih krogov oziroma kod, do sodobnih črtnih kod.

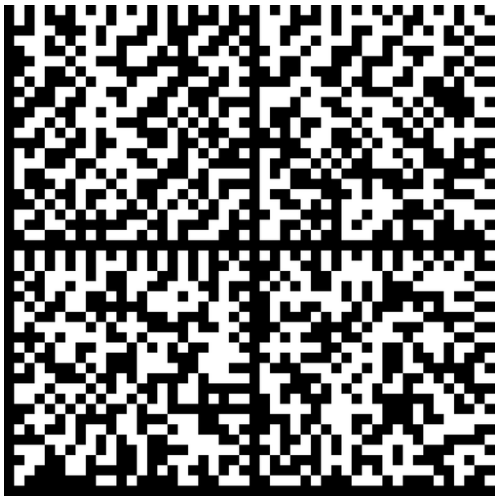
V uporabi so različne vrste črtnih kod z različnimi kodiranji:

- Linearne črtne kode
  Črtna koda kot jo pozna večina ljudi. Sestavljena je iz vzporednih črt različnih debelin, ki nosijo informacijo. Najbolj razširjena je UPC koda (slika 6).



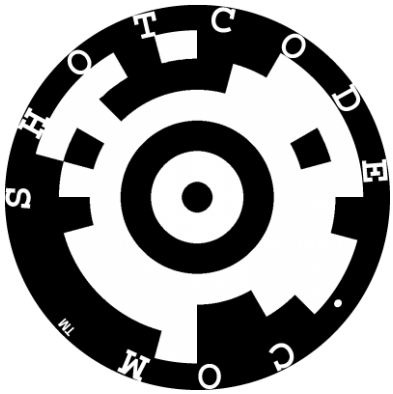
Slika 6. UPC črtna koda [32].

- Matrične črtne kode (slika 7)
  Črtna koda, ki je lahko različnih oblik. Običajno je njena prednost v večji količini podatkov, ki jih lahko predstavi.



Slika 7. DataMatrix 2D črtna koda [17].

- Bolj zahtevne kode
  Kode, ki poleg črt in kvadratov uporabljajo tudi bolj zahtevne oblike za hranjenje informacije, kot na primer barve in kroge (taka je tudi ShotCode koda prikazana na sliki 8). V take kode lahko kodiramo večje količine podatkov, tudi zvok in sliko.



Slika 8. ShotCode koda [23].

2.4.3 RFID sistemi

Prvi koraki v razvoju RFID tehnologije, kot jo poznamo danes, so bili narejeni že v času druge svetovne vojne. Podobno kot pri mnogih drugih tehnologijah, ki jih kasneje s pridom uporablja civilna družba, so se ideje porodile v vojaške namene. V drugi svetovni vojni so vse strani začele uporabljati radarje, da so zaznale bližajoča se letala. Pri tem pa je nastala težava – kako ločiti sovražna letala od prijateljskih. Nemška letala so naredila obrat ob bližanju k svoji bazi, kar je spremenilo odboj radijskih signalov. Britanci pa so na svoja letala dodali aktivne radijske oddajnike, ki so ob prejemu signala odgovorili z ustreznim odgovorom. RFID še vedno uporablja podoben koncept.

Kot RFID sistem danes pojmujemo tri komponente: RFID značko, RFID čitalnik in računalnik. RFID značka je nosilec identifikatorja, RFID čitalnik skrbi za prenos identifikatorja na

računalnik, le-ta pa opravlja vse akcije povezane s predmeti označenimi z RFID značkami. Velikokrat sta čitalnik in računalnik kar združena v en integriran sistem.

RFID značka je najpogosteje sestavljena iz antene in čipa. Sistem RFID deluje na dva možna načina, s pomočjo magnetnega polja ali pa elektromagnetnega valovanja. V prvem primeru čitalnik povzroča magnetno polje s pomočjo katerega se v tuljavi na znački inducira napetost na katero se značka nato odzove in signal odda nazaj čitalniku. Pri elektromagnetnih RFID sistemih pa izkoristimo odboj radijskega signala in ga moduliramo.

RFID značke in čitalniki le-teh delujejo na različnih frekvenčnih območjih. Vsaka frekvenca ima svoje prednosti in slabosti glede na svoje zmogljivosti. V splošnem nižja frekvenca pomeni manjši doseg in nižjo hitrost branja podatkov, po drugi strani pa večjo sposobnost branja v bližini ali na kovinskih ali tekočih površinah.

LF (ang. Low Frequency)

Značke v nizkem frekvečnem območju od 125 do 134,2 kHz in 140 do 148,5 kHz imajo razmeroma kratek doseg (okrog 0,5m) in nižjo hitrost branja glede na naprave z višjimi frekvencami. Sistemi s takšnimi značkami se najbolje obnesejo za branje značk v okolju z večjimi količinami kovin ali vode. Uporabljajo se za označevanje živali v živinoreji ali hišnih ljubljenčkov.

HF (ang High Frequency)

Značke v višjem frekvenčnem območju 13,56 MHz imajo večji doseg (okrog 1m) in večjo hitrost branja. Tudi cena posamezne značke je nižja od LF značk. Običajne uporabe vključujejo sledenje v knjižnicah, v zdravstvu in sledenje letalski prtljagi.

UHF (ang. Ultra High Frequency)

Značke, ki delujejo v frekvenčnem območju od 868 do 928 MHz, imajo približno enako ceno kot HF značke. Območje dosega je ponavadi do približno 3m. Te značke se najslabše obnesejo pri branju v povezavi s kovinskimi ali tekočimi materiali. Take značke se običajno priporočajo v distribucijski in logistični industriji in predstavljajo osnovo EPC standarda.

Glede na način napajanja oddajnika značke poznamo tri vrste RFID značk:

1. Pasivne
   So najbolj preproste in so večinoma sestavljene samo iz antene in čipa. Energija potrebna za oddajanje povratnega signala se inducira v anteni oziroma tuljavi. Ker je energija, ki jo tako pridobijo, majhna, imajo zelo omejen domet in so precej odvisne od moči signala čitalnika.

Slika 9. Pasivna RFID značka Alien ALN-9640 "Squiggle®" [2].

2. Aktivne

   Imajo lasten vir napajanja (baterijo) in ne rabijo energije čitalnika, da oddajo svoj signal. To jim daje več prednosti pred pasivnimi in polpasivnimi. Imajo večjo moč oddajanja, večji domet tudi do 100m, so bolj zanesljive, saj odziv ni odvisen od moči čitalnikovega signala, vsebujejo pa lahko tudi druge komponente, ki potrebujejo dodatno napajanje, kot na primer temperaturni senzor, spominska vezja za dodatne informacije in beleženje. Običajno beležimo lokacijo na kateri se značka nahaja, temperaturo, vlago pri občutljivejših izdelkih. So pa tudi večje in dražje od pasivnih, kar pomeni, da jih lahko uporabimo samo za identifikacijo predmetov z večjo vrednostjo. Prav tako je njihova življenjska doba omejena s trajanjem baterije.

Slika 10. Aktivna RFID značka s temperaturnim senzorjem Shenzen Friendcom [1].

3. Polaktivne

   So mešanica aktivnih in pasivnih značk. Pravtako vsebujejo baterijo, ki služi za napajanje aktivnih komponent na znački, vendar za oddajanje signala uporabijo inducirano napetost, tako kot pasivne.

Pri vse večji informatizaciji in elektronifikaciji poslovnih procesov pa se je pojavila tudi potreba po hitrejšem branju brez direktne vidne linije in hkratnem branju večih identifikatorjev. Hkrati se pojavlja tudi želja po večji natančnosti in avtomatizaciji. Prav to pa omogoča tehnologija RFID. Edina prava ovira je cena (črtna koda stane približno 0,05$, medtem ko stane tiskana pasivna RFID značka okrog 0,15$ v količinah nad deset tisoč kosov [25]) in v nekaterih primerih zanesljivost.

Kot za vsako tehnologijo in uvedbo le-te v vsakdanjo rabo, je tudi za uporabo RFID še nekaj nerešenih vprašanj. Med najbolj moteče sodi zasebnost ljudi. Kupcu bi lahko prodali izdelek, ki bi brez njegovega vedenja vseboval RFID značko, ki bi jo lahko, prav tako brez njegovega privoljenja, brali in sledili ter tako veliko izvedeli o njegovem gibanju in navadah. Najlažje si je to predstavljati na primeru obleke z všito pasivno RFID značko.

2.5 Standardi pri delu z RFID

Druga večja težava je povezana s standardizacijo. Tudi tu se še ni izoblikoval enoten standard za frekvenčna območja, ki bi bil podprt povsod po svetu. V ZDA se razlikujejo od tistih na Japonskem in v Evropi, tako da je težko z enako opremo in značkami slediti predmetom širom sveta.

Organizacija ISO je definirala serijo standardov ISO 18000, ki pokriva področje avtomatske identifikacije z brezžičnim prenosom. Zelo verjetno je, da se bo kot ISO 18000-6 standard sprejel EPC Gen 2, ki se je izoblikoval kot »de facto« standard v industriji.

2.5.1 RFID strojni standardi in EPC

Kot na vsakem področju, tudi tu obstajajo različni, med seboj konkurenčni standardi proizvajalcev strojne opreme. Predvsem pa ni enotne krovne organizacije, ki bi urejala področje RFID. Vsak kontinent, če že ne vsaka država, ima svojo organizacijo (ZDA FCC, Evropa ERO, CEPT, ETSI), ki to ureja na svoj način.

EPC (Electronic Product Code oz. elektronska koda izdelka) in EPC Gen 2

EPC je načrtovan kot univerzalni identifikator z namenom zagotavljanja enolične identifikacije vsakega objekta na svetu, za vedno. Definiran je s strani organizacije EPCglobal. To je skupno podjetje GS1 in GS1 US, ki deluje na področju mednarodnih standardov za uporabo RFID (predvsem pasivne vrste) v podjetjih na področju preskrbovalnih verig. EPCglobal je bil ustanovljen v devetdesetih letih z namenom poenotenja množice protokolov v svetu RFID.

Sprva so sprejeli 2 protokola znana kot Class 0 in Class 1, ki sta se precej na široko uporabljala v obdobju od 2002 do 2005. Decembra leta 2004 pa je Hardware Action Group (ena izmed skupin v okviru EPCglobal) na podlagi pripomb in pomanjkljivosti protokola Class 1 definirala še standard EPC Class 1 Generation 2. EPC Gen 2, kot ga tudi imenujejo, deluje v območju med 860 do 960 MHz in definira fizične in logične zahteve za čitalnike kot tudi značke. Zelo verjetno je, da bo ravno ta standard tudi v prihodnje imel precejšnjo vlogo na trgu RFID in se razvil v »de facto« standard na tem področju. To v veliki meri potrjuje tudi množična uporaba v industriji.

2.5.2 Programski standardi za izmenjavo podatkov pri sledenju izdelkom na področju EU

S strani Evropske komisije je bilo v preteklosti s programi FP5 (Fifth Framework Programme), FP6 (Sixth Framework Programme) in FP7 (Seventh Framework Programme) podprtih že veliko število projektov na temo sledenja v preskrbovalni verigi. Izmed teh so bili najbolj odmevni oziroma prodorni:

- **TraceFish** – skrajšano za Traceablitiy of Fish Products [29]
  Projekt, ki ga je koordiniral norveški inštitut za ribarjenje in vodno rejo (Fiskeriforskning). V njegovem okviru naj bi razvili načine za zbiranje in izmenjavo informacij o transakcijah z ribami v distribucijskih verigah. Izid projekta so bili trije standardi na področju sledenja ribjim izdelkom: za gojene ribe, za ulovljene ribe in tehnični standard. Prva dva opredeljujeta katere informacije je treba zajemati in zapisovati pri sledenju ribam skozi celotne distribucijske verige. Zadnji pa opisuje, kako se podatki kodirajo in pošiljajo v elektronski obliki. Predvideva izmenjavo v obliki

XML sporočil. Ta projekt je osnova za mnoge druge, ki imajo za cilj elektronsko izmenjavo informacij o sledljivosti hrane.

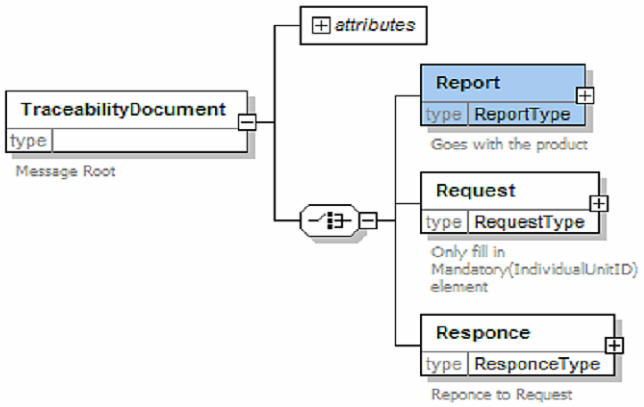
- **TRACE** [27]
  Projekt na celotnem področju sledljivosti od verifikacije porekla preko kemične analize do dobrih praks sledljivosti (ang. Good Traceability Practice).
- **Seafood Plus in Telop trace** [22]
  Raziskovalni projekt v okviru FP6, katerega cilji so zmanjšanje zdravstvenih problemov med Evropskimi uporabniki s ciljem promocije zdravja in varne morske hrane.
- **TraceFood** [30]
  Ogrodje v okviru TRACE projekta, ki je logično nadaljevanje od točke, kjer sta končala TraceFish in SeafoodPlus.

TraceFood

Projekt TRACE je v okviru TraceFood ogrodja definiral več verzij TraceCore XML standarda. Za osnovo so vzeli temelje postavljene s TraceFish tehničnim standardom.

Sam standard je razdeljen na dva dela, enega za splošne informacije, neodvisne od področja uporabe (v okviru trgovanja s pokvarljivimi prehrambenimi izdelki), in drugega za dodatne informacije, ki predstavljajo razširitev namenjeno posameznemu področju.

V času svojega delovanja so v projektu razvili več delovnih različic standarda, nekako tako kot je pritekal denar. Od TraceCore XML (del strukture prikazan na sliki 11), preko TraceCore XML 2, TraceCore XML 2.1 do TraceCore XML 2.2, ki se je končal pri različici »Release Candidate 2«. Vse skupaj je potekalo v obdobju od leta 2002 do leta 2009.



Slika 11. Del strukture TraceCore XML [28].

TraceCore XML podaja osnovo za sledljivost prehrambenih izdelkov med posameznimi strankami v preskrbovalni verigi ne glede na področje uporabe. Opredeljuje osrednji del

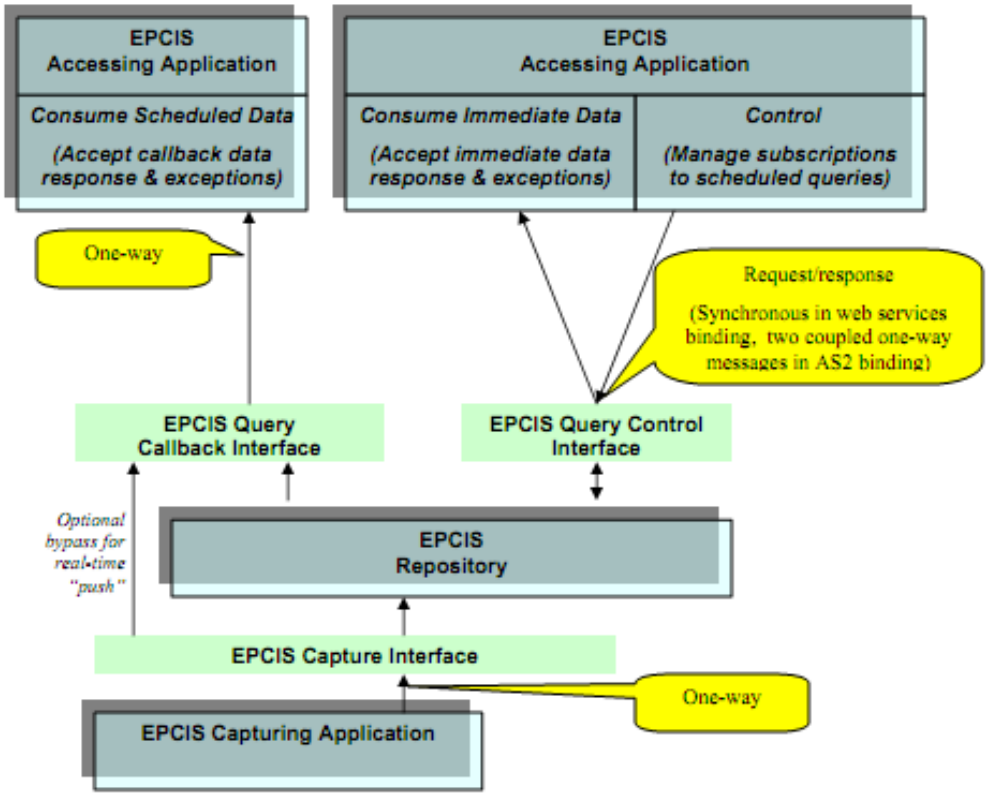
standarda, ki vsebuje vse potrebne podatke za sledljivost prehrambenih izdelkov v splošnem. Lastnosti izdelkov na nivoju posameznega področja (npr. morska hrana, mineralna voda, med, piščančje meso, žitarice, meso itd.) pa naj bi definirale razširitve standarda kot naprimer TraceFish XML, TraceCereal XML in podobne.

Na koncu projekta so definirali še TraceCore Abstract model, ki naj bi služil preslikavi TraceCore sledljivostnega modela na več kot le en format za izmenjavo, niti ne nujno XML.

Poravnava TraceCore z EPCIS in EPCIS standard

Z letom 2009 pa se je TRACE zaključil, s tem se je ustavil tudi dotok denarja za delovanje na standardu TraceCoreXML. Aprila 2008 so člani delovne skupine razpravljali o nadaljnih možnostih za TraceCore XML. Prišli so do zaključka [16], da je za nadaljevanje najboljša opcija poravnava s kakim drugim standardom oziroma sistemom, ki ima sledeče lastnosti:

- čim več skupnih točk s TraceCore XML;
- čim večje zaledje obstoječih uporabnikov v prehrambeni industriji;
- je že preizkušen v praksi in vključuje izkušnje iz realne uporabe;
- ima na voljo sredstva za vzdrževanje standarda.



Slika 12. EPCIS arhitektura [6].

Pred izbiro so preučili več možnih kandidatov (ISO, OASIS, GS1, EPCIS), na koncu pa izbrali EPCIS.

EPCIS (EPC Information Services) je standard, določen s strani EPCGlobal, za izmenjavo podatkov povezanih z EPC med trgovinskimi partnerji [7]. Razvit je bil v sodelovanju z velikimi podjetji in izkorišča njihove izkušnje na tem področju. Sistem je predstavljen z diagramom na sliki 12. Deli se na tri glavne dele:

1. »EPCIS Query Interface«
2. »EPCIS Repository«
3. »EPCIS Capture Interface«

»EPCIS Query Interface« predstavlja modul EPCISa s katerim komunicirajo odjemalci informacij o RFID značkah in je razdeljen na dva dela. Prvi (»EPCIS Query Control Interface«) je namenjen pridobivanju podatkov po potrebi in prijavam na poizvedbe, drugi (»EPCIS Query Callback Interface«) pa vračanju rezultatov poizvedb na katere je prijavljena aplikacija. Definiranih je nekaj standardnih klicev, ki jih mora poznati vsaka implementacija. Metode so sledeče:

subscribe – prijava na poizvedbo, ki se bo izvajala periodično;
unsubscribe – odjava od poizvedbe;
poll – enkratna poizvedba;
getQueryNames – vrne imena vseh podprtih poizvedb na voljo za poizvedovanje preko subscribe in poll metod;
getSubscriptionIDs – vrne identifikatorje obstoječih prijav na poizvedbe z določenim imenom;
getStandardVersion – vrne verzijo podprtega standarda;
getVendorVersion – vrne verzijo proizvajalčevih razširitev;

Ker za to nalogo primarni cilj ni bil popolna implementacija vseh funkcionalnosti, ki jih predpisuje EPCIS, ampak le prikaz izvedljivosti in uporabe standarda za namen sledenja gojenim ribam, vmesnika »EPCIS Query Callback Interface« in prijav na poizvedbe nismo realizirali.

»EPCIS Repository« je skladišče EPCIS dogodkov po katerih se z »EPCIS Query Interface« povprašuje. »EPCIS Capture Interface« definira način sporočanja EPCIS dogodkov od aplikacij, ki te dogodke zajemajo, do repozitorijev, ki jih hranijo.

Komunikacija med posameznimi deli sistema

EPCIS za komunikacijo med odjemalci podatkov označenih izdelkov in strežnikom predvideva več načinov. Za »ECPIS Query Control Interface«:

- SOAP over HTTP (WSDL)
- XML over AS2

Uporabili smo SOAP over HTTP.

3 Implementacija sledenja rib z RFID

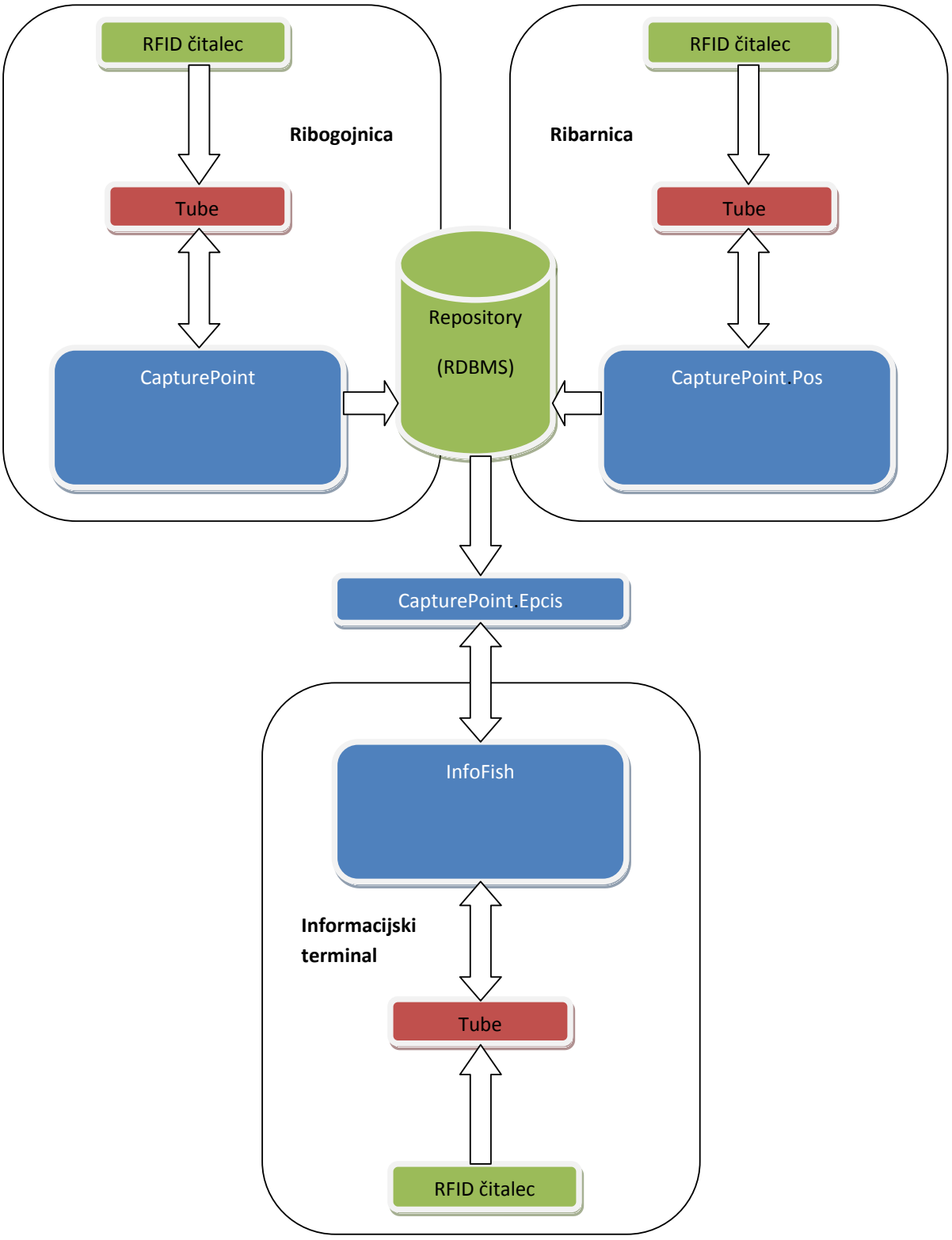
V naši implementaciji smo prišli do podobne programske arhitekture, vendar ne tako kompleksne, saj so naša ciljna skupina mala in srednje velika podjetja, medtem ko GS1 EPCGlobal z EPCIS cilja na velika podjetja. Programsko arhitekturo naše rešitve podaja slika 13.

Če bi morali sistem razširiti in ga uporabiti za večja podjetja, bi se lahko katerakoli komponenta zamenjala z bolj natančno implementacijo po EPCIS standardu oziroma v taki obliki, da bi zadostila novim zahtevam.

Tako smo uporabili le obliko sporočil in vmesnik, kot ju za izmenjavo sporočil med posameznimi porabniki določa EPCIS. Oblika temelji na XML standardu in omogoča razširjanje s podatki, ki so pomembni za specifično področje uporabe. Razširitve za to delo smo določili sami, saj TraceCore XML standard, ki naj bi jih določal, še ni dokončan.

Osredotočili smo se na del, ki nam s strani odjemalca omogoča poizvedovanje po podatkih preko EPCIS Query Control Interface. EPCIS sicer podaja tudi osnovno obliko za zajem podatkov (EPCIS Capture Interface), vendar smo se v tej nalogi osredotočili na izmenjavo podatkov po tem standardu.

Omeniti je treba tudi, da smo v okviru izvorne kode (poimenovanje spremenljivk, razredov ipd.) in celotnega sistema za nadzor različic izvorne kode uporabljali angleški jezik. Razlogov za to je več, izpostaviti pa velja predvsem to, da se angleški jezik bolje podaja poimenovanju spremenljivk in da so vsa ogrodja, ki smo jih uporabljali tudi napisana in poimenovana v angleškem jeziku. V primeru uporabe slovenskega jezika za poimenovanja bi hitro prišlo do neskladja med jeziki.



Slika 13. Programska arhitektura implementacije.

3.1 Zahteve

Pri izvedbi sistema smo zahtevali:

- Uporabo obstoječih standardov
  Obstoječi standardi imajo najboljše možnosti, da bodo uporabni tudi na dolgi rok. Standard, ki je bil razvit v obstoječih projektih Evropske komisije je v kar največji meri poravnan z zakonskimi zahtevami, ki bodo vpeljane v prihodnosti.
- Možnost samostojnega delovanja brez pretiranih zahtev za nameščanje
  Tu imamo v mislih možnost čim lažje postavitve v uporabniško okolje, po možnosti brez namestitve dodatnih programskih komponent kot so relacijska podatkovna baza. Namen tega je čim krajši cikel od implementacije dela sistema do možnosti preizkusa pri stranki in s tem hitrejši razvojni cikel.
- Preprosto uporabo
  Za uporabnika mora biti uporaba del delovnega procesa. Tak sistem ne obremenjuje uporabnika z dodatnimi postopki, ampak je del vnosa artiklov v sam sistem.

Ker gre za diplomsko nalogo nismo imeli dejanskih zahtev s strani naročnika, v nasprotnem primeru bi zgornji seznam dopolnili še z naročnikovimi zahtevami.

3.2 Metodologija

Pri razvoju smo poizkušali uporabiti sodobne metode zasnove programske opreme.

Proces razvoja smo razdelili na 4 faze:

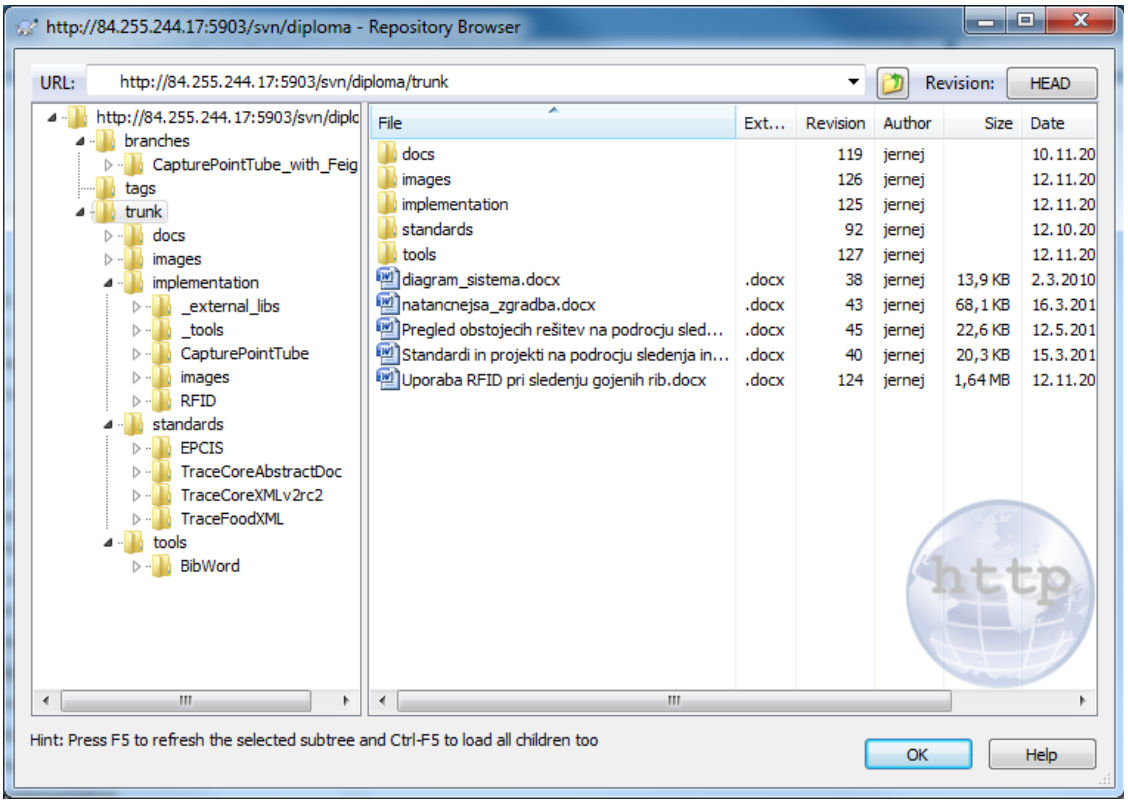
1. Načrtovanje in implementacija programske arhitekture
   V tem delu smo na grobo razdelili potrebne module programske arhitekture in določili tehnologije v katerih bi jih bilo najbolje izvesti. Opredelili smo aplikacije, ki so pomembne za dobavitelja rib in aplikacijo za končnega kupca.
2. Načrtovanje modela domene
   Faza v kateri smo določili osnovne pojme v poslovni logiki sistema in definirali interakcijo med njimi.
3. Načrtovanje uporabniškega vmesnika
   Določili smo okna in način uporabe osnovnih modulov sistema.
4. Implementacija prejšnjih točk in preurejanje (ang. refactoring)
   Če prejšnje faze vzamemo kot pripravljalne, potem ta faza predstavlja izvedbo vsega načrtovanega s potrebnimi spremembami in prilagoditvami. Kot vedno pri razvoju programske opreme se tudi tu pokaže veliko stvari, ki jih nismo predvideli v času načrtovanja. Preurejanje pa se je v praksi izkazalo za nujno »zlo« za dolgoročno zmožnost vzdrževanja sistema.

Velja omeniti predvsem naslednje metode pri razvoju programske opreme.

Upravljanje konfiguracije (ang. Configuration management)

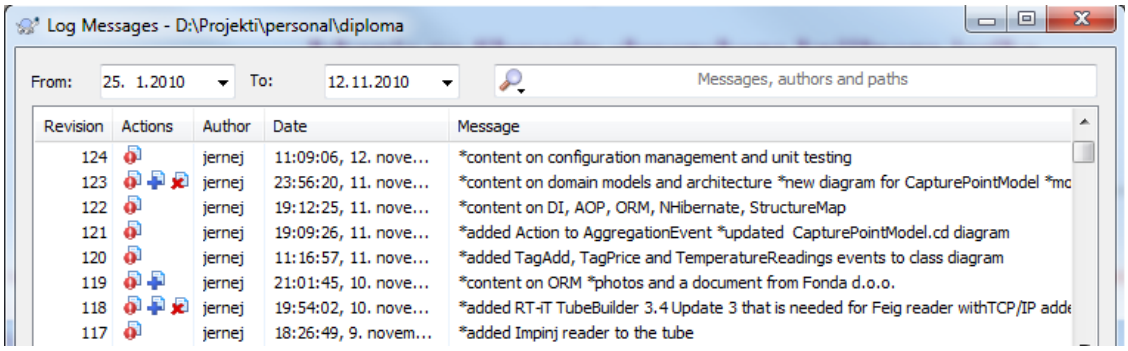
»Upravljanje konfiguracije se nanaša na proces s katerim se vsi elementi povezani z vašim projektom in povezave med njimi hranijo, vračajo, enolično označijo in spreminjajo.« [15]

Tega smo se v tem delu zelo dosledno držali, saj smo imeli vse kar je bilo povezano z izvedbo, vključno s tem dokumentom, v shrambi za nadzor verzij (ang. version control repository). S tem si omogočimo takojšno ponovno postavitev delovnega okolja, brez zamudnega iskanja predpisanih verzij orodij in knjižnic. Prav tako pridobimo ponovljivost vsake verzije programske opreme. Če revizije (ang. revision) v repozitoriju ustrezno označujemo, potem lahko kadarkoli spet vzpostavimo razmere, kot so bile v trenutku, ko smo svoje delo potrdili (ang. commit) v repozitorij (vsebina prikazana na sliki 14).



Slika 14. Vsebina repozitorija za nadzor verzij.

Še ena dobra lastnost nadzora verzij se pokaže, ko nas zanima kdo in zakaj je naredil neko spremembo na kateremkoli delu našega sistema. Če se za vsako potrditev držimo pravila, da mora imeti vsaka potrdiv v repozitorij kratek opis, imamo zagotovljeno zgodovino in namen vseh sprememb na sistemu (slika 15).



Slika 15. Primer izpisa dnevnika potrditev v repozitorij.

Enotsko testiranje (ang. Unit Testing)

V računalniškem programiranju je enotsko testiranje metoda s katero posamezne enote izvorne kode testiramo ali so primerne za uporabo. Enota je najmanjši možni del, ki se ga v aplikaciji še da testirati. [31]

Pojem je tesno povezan z zagotavljanjem kakovosti programske opreme in vzdrževanjem le-te na daljši rok. Znan pa je tudi pojem testno gnanega razvoja (ang. Test Driven Development, krajše TDD), kjer se celotno načrtovanje programske opreme in njenih gradnikov podreja zmožnosti testiranja. Posledica takega načina razvoja je na splošno boljša kvaliteta zasnove programske opreme, saj nas že sam princip vodi k bolj modularni zasnovi in jasno deljenim odgovornostim v posameznih enotah.

Ločiti je potrebno enotsko testiranje od integracijskega in prevzemnega (ang. acceptance) testiranja, ki sta razširjeni obliki testiranja in zajemata celotne sklope ali celo sisteme. Pri enotskem testiranju se koncentriramo na majhne samostojne in zaključene enote, v primeru objektnega programiranja je to razred ali v skrajnem primeru celo metoda v razredu. Pri integracijskem testiranju gre predvsem za tesiranje delovanja večjih sklopov sistema.

Model domene in domensko usmerjeno načrtovanje (ang. Domain Driven Design, krajše DDD)

»Poslovna logika, s katero ima program opravka, je lahko zelo kompleksna. Pravila in logika opisujejo veliko različnih primerov, oblik in kombinacij obnašanja. Prav s takšnimi zapletenimi opravili v mislih so bili ustvarjeni objekti. Model domene ustvari mrežo med seboj povezanih objektov, kjer vsak objekt predstavlja smiselnega posameznika...« [11]

Model domene se ponavadi uporablja v povezavi z domensko gnanim razvojem, ki je samo po sebi dovolj kompleksno področje, zato ga v tem delu nismo obravnavali bolj podrobno. Ukvarja pa se predvsem s tem, kako čim bolje s pomočjo objektnega programiranja simulirati probleme v resničnemu življenju.

3.3 Orodja

Pri implementaciji modernih informacijskih sistemov seveda ne gre brez uporabe že pripravljenih orodij oziroma komponent. S tem imamo v mislih predvsem programska ogrodja, programske knjižnice, razvojna orodja in podporne tehnologije. Nepredstavljivo je namreč, da bi vsak del sistema razvijali sami. Tudi v tem delu smo zato uporabili precej že razpoložljivih rešitev za delo s zunanjimi servisi (npr. podatkovno bazo, RFID čitalnik) in za pomoč pri raznih opravilih.

V veliko pomoč so nam odprtokodne rešitve. Microsoft, kot proizvajalec .NET ogrodja, je tradicionalno vedno deloval bolj usmerjeno v lastniške oziroma zaprtokodne rešitve, vendar se je tudi na področju .NET ogrodja v zadnjih letih prebudil določen krog razvijalcev in podjetij, ki aktivno podpirajo razvoj odprtokodne programske opreme.

3.3.1 Strojna oprema in vmesna programska oprema

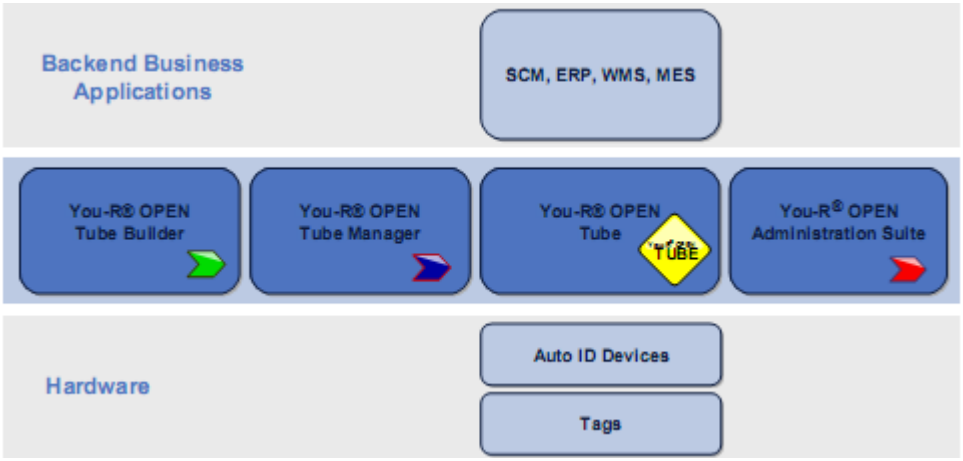
Edino strojno orodje s katerim smo imeli pri tem delu opravka je bil RFID čitalnik. Kot pove že ime je RFID čitalnik naprava, ki skrbi za zaznavanje in branje RFID značk. V našem primeru gledamo na čitalnik kot na komponento sistema, ki nam sproža dogodke povezane z identifikatorji značk. Te dogodke potem beležimo v sistem oziroma na njih reagiramo in uporabniku prikažemo povezane podatke ali mu ponudimo možnost reakcije.

Med strojno opremo (čitalnikom priključenim na računalnik) in višjenivojsko programsko opremo stoji vmesna programska oprema (ang. middleware). Vmesna programska oprema je plast abstrakcije, ki nam omogoča lažji razvoj aplikativne programske opreme, brez poznavanja implementacijskih detajlov dostopa do strojne opreme. Še ena prednost uporabe vmesne opreme je, da nam ponuja pogled na strojno opremo preko programskega vmesnika (ang. Application Programming Interface), ki ni odvisen od dejanske vrste strojne opreme, in nam omogoči neopazno menjavo strojne opreme, seveda z določenimi omejitvami, kot na primer enaka oblika zapisa na značkah.

Za vmesno programsko opremo smo uporabili paket »detego You-R Open« podjetja RF-iT Solutions [21]. Vsebuje vse, kar potrebujemo za uporabo širokega spektra čitalnikov. Posamezne aplikacije v paketu so:

- »Administration Suite« – nadzor aplikacij in storitev
- »(Mobile) Tube Manager« – povezava med Administration Suite in Tube-i
- »Tube Builder« – razvoj aplikacij in storitev
- »Verification Client«– testiranje aplikacij

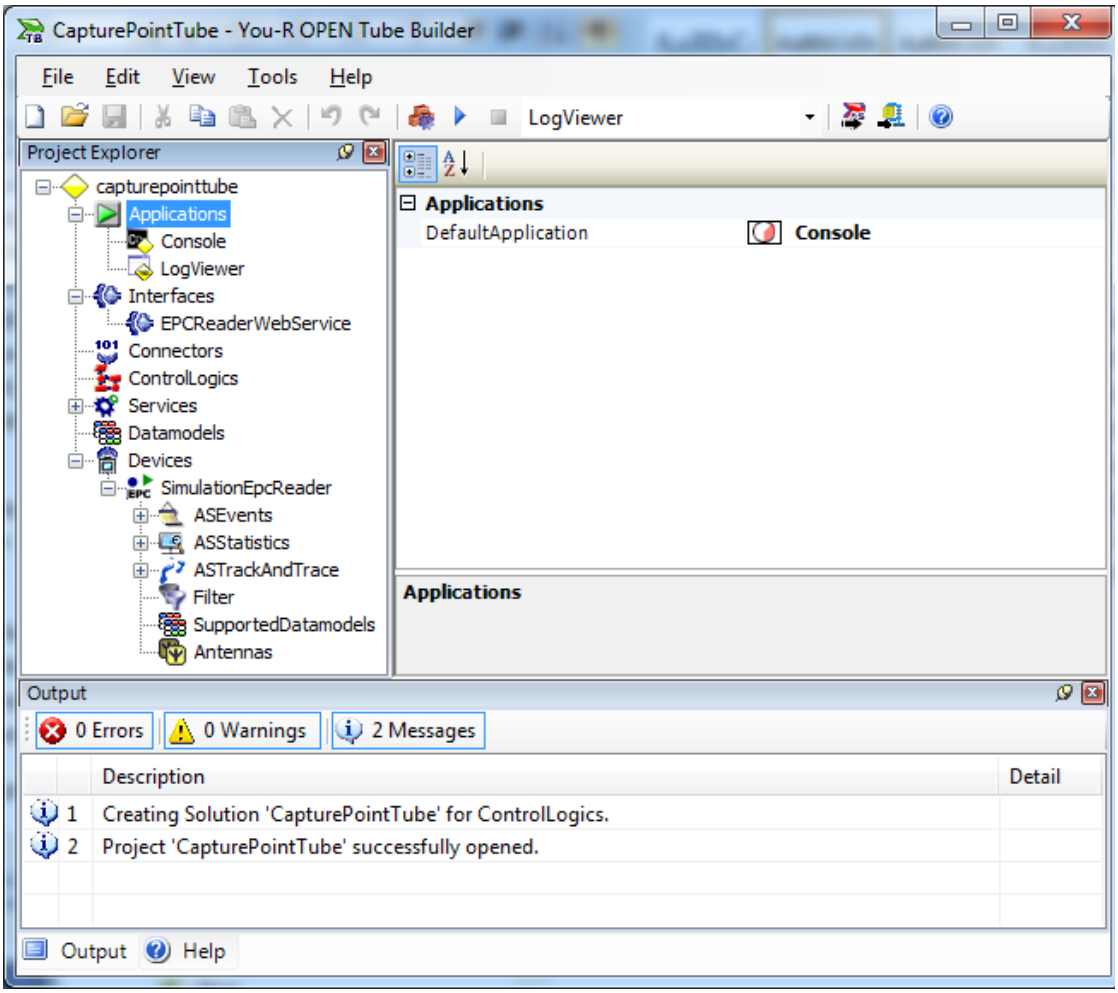
Celoten sklad teh aplikacij je prikazan na sliki 16.



Slika 16. Sklad aplikacij, ki sestavljajo programski paket detego You-r Open [34].

Na voljo so nam vmesniki za LF, HF, UHF naprave, kot tudi za čitalnike črtnih kod, različnih proizvajalcev. Paket vsebuje več aplikacij, ki nam omogočajo realizacijo celotnih projektov z uporabo RFID in črtnih kod. Osrednji pojem v gradnji sistema je »Tube«. »Tube« je komponenta, zadolžena za povezovanje podatkov iz lokalnih identifikacijskih točk do drugih informacijskih sistemov [34]. Sestavljena je lahko iz mnogih delov, od katerih so poglavitni: vmesnik za dostop do naprav, abstrakcija naprav, preslikava podatkov v podatkovni model, kontrolna logika, grafični vmesnik, povezovalniki (and connector) in storitve. Za naše potrebe si lahko »Tube« predstavljamo kot dvojček naprave in storitev. Napravo tukaj vidimo kot abstraktno in se nam ni treba ukvarjati s tehničnimi detajli komunikacije z njo, od nje samo sprejemamo dogodke in podatke zapisane na znački. V »Tube« tako le dodamo ustrezno vrsto naprave, nastavimo parametre potrebne za zahtevani način povezovanja (npr. številko vrat za povezovanje preko RS-232 vmesnika ali IP številko omrežnega čitalnika) in naprava nam je na voljo za uporabo v storitvah, ki jih nudimo drugim komponentam našega sistema.

Za razvoj »Tube« uporabljamo »Tube Builder« in za testiranje »Verification Client«. V »Tube Builder« dodamo nov »Tube«, vanj povežemo napravo, ustrezen podatkovni model in »Tube« izpostavimo preko nekega vmesnika zunanjemu svetu. Običajno se za ta vmesnik uporabi spletni servis (ang. web service). Ko imamo ta del opravljen, zaženemo »Tube« in »Verification Client«, ki se poveže na novo definiran »Tube«. »Verification Client« nam v obliki dnevnika prikazuje vse dogodke, ki jih sporoča storitev v našem »Tube«. Če smo torej pravilno sestavili »Tube«, potem bi nam na primer pristavitev značke k čitalniku ali vklop simulirne naprave moral povzročiti nov vnos v dnevniku, ki ga beleži »Verification Client«. Na ta način se prepričamo, da deluje povezava med storitvijo, ki jo nudimo navzven, in strojno opremo. »Tube Builder« je prikazan na sliki 17.



Slika 17. You-R Open TubeBuilder.

Za potrebe razvoja so nam na voljo simulacijski čitalniki RFID značk. Eden od dosegljivih je tudi »SimulationEpcReader«, ki nam vrača naključno generirane ali vnaprej določene EPC kode. To nam omogoča testiranje funkcionalnosti našega sistema brez prisotnosti dejanske strojne opreme. Seveda bi nam to prišlo prav tudi v primeru, ko bi pri razvoju sistema sodelovalo več razvijalcev hkrati, na voljo pa bi imeli le majhno število čitalnikov.

V kombinaciji z vmesno programsko opremo »detego You-R Open« smo uporabili čitalnik Feig i-scan MRU200i (slika 18). Gre za industrijski čitalnik RFID značk, ki omogoča priklop preko različnih vmesnikov: USB ethernet in RS232. V našem primeru smo imeli na voljo model, ki podpira RS232 in ethernet. Odločili smo se za slednjega, saj nam to omogoča večjo fleksibilnost, ker lahko čitalnik postavimo na katerokoli mesto znotraj mreže v podjetju, ne glede na oddaljenost od dejanske delovne postaje, ki ga krmili oziroma prejema podatke od čitalnika. Današnji računalniki tudi nimajo več na voljo RS232 serijskega vmesnika, kar pomeni, da bi si z izbiro tega vmesnika precej zožali nabor delovnih postaj.

Čitalnik deluje v UHF frekvenčnem območju, torej od 865MHz do 928MHz z dosegom okrog 2,5m. Posebnost čitalnika sta dve vgrajeni anteni. Prva je takoimenovana »Near field«

antena, ki deluje po principu backscatter coupling in omogoča zaznavanje majhnih transponderjev ter s tem majhnih objektov, ki so označeni z njimi. »Far field« antena, ki je tudi že vgrajena v čitalnik, omogoča zelo usmerjeno branje, tako da ne dobivamo dogodkov povezanih z značkami, ki nas ne zanimajo. Tak dogodek je na primer zaznavanje značke, ki se nahaja na sosednjem tekočem traku v tovarni. Čitalnik ima tudi možnost delovanja v načinu z nizko močjo (ang. Low power mode) v katerem se še dodatno omeji doseg in s tem še poveča selekcija prebranih značk.

Podprto je branje značk po standardu EPC Gen 2 in opcijsko tudi ISO 18000-B/-C, vendar čitalnik, ki smo ga uporabili, te opcije ni imel. Preizkušali smo predvsem EPC Gen 2 in z njim povezani EPCIS, tako da to ni bil problem.

Slika 18. Feig i-scan UHF čitalnik [9].

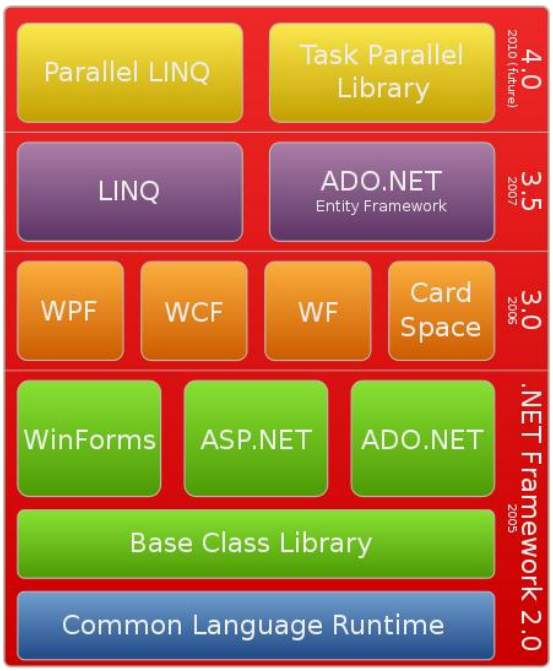
Za preizkusne značke smo uporabili nalepke UPM Web podjetja UPM Raflatac. To so običajne pasivne značke po standardu EPC Class 1 Gen 2 in delujejo v frekvenčnem območju 860 MHz do 960 MHz, primerne za uporabo pri prodajnih terminalih in označevanje posameznih artiklov. Velike so 22mm x 40 mm, neobčutljive na usmeritev glede na čitalnik in imajo vgrajen 96-bitni pomnilnik za EPC [33].

3.3.2 Programska oprema

Microsoft .NET

Je programsko ogrodje (predstavljeno z diagramom na sliki 19) namenjeno računalnikom z operacijskim sistemom Microsft Windows. Vsebuje množico pripravljenih rešitev za izvrševanje pogostih nalog v računalništvu. Podpira delo v večih programskih jezikih, ki jih prevajalniki, ki so del ogrodja, prevedejo v skupni vmesni jezik (ang. Common Intermediate language, krajše CIL), tega pa potem izvršuje navidezni stroj. CIL je neodvisen od strojne opreme in se lahko izvršuje na vsaki strojni opremi, ki podpira izvajalno okolje .NET. Ta strojna oprema vključuje vse od »polnopravnih« računalnikov do telefonov. Poleg Microsoftovega .NET okolja na Windows platformi obstaja namreč še odprtokodni projekt

Mono, ki ni vezan na Windows operacijski sistem. V teoriji lahko torej programe pisane v kateremkoli .NET programskem jeziku izvajamo na različnih platformah.



Slika 19. Diagram ogrodja .NET [18].

V osnovi so v ogrodju zajeta orodja za dejavnosti na naslednjih področjih, če naštejemo samo nekatere:

- grafični uporabniški vmesnik,
- spletne aplikacije,
- komunikacije,
- dostop do podatkov, povezovanje z bazami podatkov,
- kriptografija.

Za to ogrodje smo se odločili predvsem zaradi poprejšnjega poznavanja in pozitivnih izkušenj pri uporabi v že izvedenih projektih. Ogrodje se aktivno razvija in je skozi leta doživelo že več posodobitev, trenutna različica je 4.0, ki smo jo tudi uporabili v tej nalogi.

WCF (ang. Windows Communication Foundation)

Za komunikacijo in razvoj spletnih servisov smo uporabili Windows Communication Foundation, del Microsoft .NET ogrodja namenjen razvoju storitveno orientiranih aplikacij preko enotnega programskega modela, ki komunicirajo preko omrežja. EPCIS Query Control Interface predstavlja natanko to – storitev, ki nam ponuja informacije o dogodkih povezanih z značkami, ki imajo EPC identifikator. ECPIS standard nam definira obliko storitev, WCF pa nam mogoča tehnično izvedbo prenosa sporočil po množici protokolov. Pri razvoju z WCF za

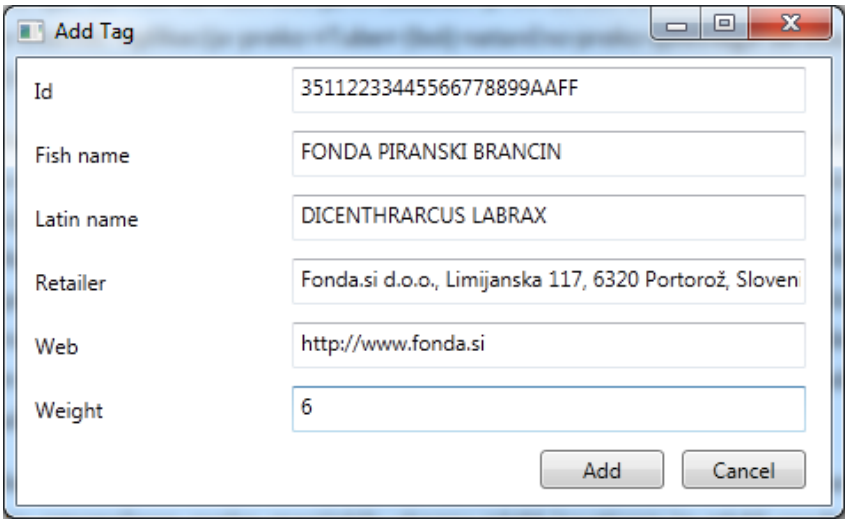
storitev definiramo na strani strežnika vmesnik (interface) in končno točko (endpoint) na katero se po tem vmesniku povežemo.

WPF (ang. Windows Presentation Foundation)

Za grafični uporabniški vmesnik smo uporabili WPF, ki je del .NET ogrodja od različice 3.5 dalje. Omogoča bolj deklarativen razvoj grafičnega vmesnika v jeziku XAML (slika 20), kot smo bili običajno navajeni pri tehnologijah za razvoj namiznih aplikacij. Temelji na XML in se zdi podoben jeziku HTML in je zato blizu tudi spletnim razvijalcem. Z njim deklariramo elemente uporabniškega vmesnika, vezi na podatke (ang. databinding), povežemo upravljalce dogodkov (ang. event handler) v takoimenovani code-behind datoteki z dogodki na vmesniku in drugo.

```
<Window x:Class="RFID.CapturePoint.AddTag.AddTagView"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Add Tag" Width="475" Height="290">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="150" />
            <ColumnDefinition Width="300"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
        <Label Content="Id" />
        <TextBox Text="{Binding TagId}" Grid.Column="1" IsReadOnly="True" />
        <Label Content="Fish name" Grid.Row="1"/>
        <TextBox Text="{Binding FishName}" Grid.Column="1" Grid.Row="1" />
<!--
More content...
-->
    </Grid>
</Window>
```

Slika 20. Primer XAML izvorne kode.



Slika 21. Izgled okna definiranega z XAML kodo prikazano na sliki 20.

Microsoft Visual Studio 2010

Najbolj razširjeno integrirano razvojno okolje za .Net ogrodje je Microsoft Visual Studio (uporablili smo različico Visual Studio 2010). Omogoča nam razvoj programskih rešitev od začetka do konca na vseh nivojih, od zasnove podatkovnega modela do uporabniškega vmesnika.

Tu je potrebno omeniti tudi nepogrešljivo razširitev Resharper. Ta nam omogoča učinkovito delo z veliko bližnjicami za pogosta opravila, kot so na primer ustvarjanje razredov iz vmesnikov in obratno, poimenovanje spremenljivk. Predvsem pa se vrednost razširitve pokaže ob preurejanju programske kode, saj namesto nas poskrbi za preimenovanje, spremembo definicij, premikanje med datotekami in druga zamudna opravila.

Vnašanje odvisnosti in StructureMap

Pri razvoju razvejanih in modularnih aplikacij, sploh v povezavi z razvojem pri katerem obširno uporabljamo testiranje modulov (ang. unit testing), je zelo koristna uporaba principa vnašanja odvisnosti (ang. Dependency Injection). Sam princip je definiran takole:

»Vnašanje odvisnosti v objektno orientiranem programiranju je načrtovalni vzorec za ločevanje vedenja od razreševanja odvisnosti. Z drugimi besedami: tehnika za zmanjšanje vezanosti med zelo odvisnimi programskimi komponentami.« [5]

Vse to lahko strnemo v to, da neki programski komponenti (npr. razredu v objektnem programiranju) ni treba skrbeti za to, kako bo pridobila vse zunanje servise (odvisnosti), razvijalci se lahko osredotočijo na to, kako bodo razvijali poslovno logiko in programski opremi dodajali funkcionalno vrednost.

Dodatna prednost, ki jo pridobimo z uporabo DI, je aspektno orientirano programiranje (ang. Aspect Oriented Programing, krajše AOP). Le-to nam omogoča, da nekatere infrastrukturne oziroma »prerezne« skrbi dejansko preložimo na infrastrukturo. Predvsem se to velikokrat pokaže za koristno, ko moramo pri veliko metodah poskrbeti za to, da se ustvari transakcija na podatkovni bazi. Odgovornost za to lahko prenesemo na DI ogrodje, ki ob vsakem klicu npr. metode poskrbi, da se ustvari transakcija, in po končanem klicu ali v primeru napake poskrbi, da se okolje pospravi.

Za to delo smo uporabili odprtokodno ogrodje imenovano StructureMap v kombinaciji z odprtokodno knjižnico Castle DynamicProxy, ki skrbi za ustvarjanje takoimenovanih proxy razredov s pomočjo katerih lahko potem prestrezamo klice na razred in izvajamo aspektno orientirano programiranje.

SQL in relacijska podatkovna baza

Za shranjevanje podatkov oziroma persistenco stanja sistema smo izbrali relacijsko podatkovno bazo. Za poizvedovanje v večini takih podatkovnih baz služi SQL.

Uporabili smo dve izvedbi relacijske podatkovne baze. Obe sta dosegljivi brez stroškov in si ju lahko prenesemo s spleta, razlika med njima pa so v popolnosti in obremenitvi za računalnik, ki ju gosti, in seveda v namembnosti.

- **SQLite**
  SQLite je programska knjižnica, ki služi kot lahka, samostoječa (brez namenskega strežnika), preprosto nastavljiva transakcijska relacijska podatkovna baza. Programska koda je prosto dostopna. Seveda pa ima ta »lahkost« svojo ceno. Knjižnici manjka nekaj bolj zahtevnih delov SQL standarda (shranjene procedure in podobno), zato za projekte, kjer se te naprednejše funkcije potrebujejo, ni primerna. Prav tako ni primerna za okolja, kjer do podatkovne baze hkrati dostopa in v njej spreminja podatke več uporabnikov.
  V tej nalogi smo jo uporabili predvsem pri razvoju. Eden izmed načinov delovanja je kar v delovnem spominu računalnika. To omogoča zelo hitre integracijske teste, saj je podatkovna baza postavljena in spet izbrisana v roku nekaj milisekund. Kreiranje in brisanje podatkovne baze pa je nujno za neodvisnost takih testov eden od drugega. Prav tako lahko deluje kar iz datoteke. To pa nam koristi pri pošiljanju razvojne različice aplikacije v pregled naročniku, saj nam za to ni treba nameščati strežnika za podatkovno bazo.
- **Microsoft SQL Server 2008 Express**
  Microsoftova izvedba relacijske baze. Je funkcijsko popolna relacijska podatkovna baza, ki implementira vse kar predpisuje ANSI SQL in veliko Microsoftovih razširitev. Potrebuje primerek strežnika na računalniku dosegljivem preko omrežja. Za razliko od SQLite je podatkovna baza skrita za večimi datotekami in se jo upravlja preko posebne programske opreme (SQL Server Management Studio) s svojim

uporabniškim vmesnikom. Različica Express je nekoliko okrnjena glede na plačljivi SQL Server 2008, vendar za potrebe informacijskega sistema, kot smo ga razvili za to delo, to ne igra vloge.

V pravi namestitvi sistema bi uporabili tak strežnik, saj lahko tako bazo namestimo kamorkoli v omrežje naročnika in dobro podpira večuporabniški način dela. Na voljo je tudi veliko število izdelkov, ki jih potrebujemo za zagotavljanje obstojnosti podatkov.

Objektno relacijska preslikava in NHibernate

Orodje za objektno-relacijsko preslikavo (ang. Object-Relational Mapping ali krajše ORM), ki skrbi za lažje premoščanje takoimenovanega impedančnega nesorazmerja med objektno orientiranimi programskimi jeziki in relacijskimi podatkovnimi bazami. Omogoča nam uporabo objektno orientiranega pristopa k razvoju, ki ga nato preslikamo na relacijsko podatkovno bazo. Pri tem pridobimo tudi neodvisnost od implementacije relacijske baze (MSSQL, MySQL, Oracle in drugi). Prednost tega je, da programsko opremo brez večjih težav prilagodimo za uporabo z drugo relacijsko podatkovno bazo. Pomaga nam tudi z zmanjševanjem količine izvorne kode, ki jo moramo napisati za dostop do podatkov. Namesto, da pišemo za vsako SQL poizvedbo posebej lahko operiramo direktno z objektnim modelom, iz tega pa nam ORM orodje dinamično generira SQL stavke, jih izvede in potem preslika nazaj v naš objektni model.

Prenosljivost na drugo podatkovno bazo pa nam omogoča tudi zelo učinkovite integracijske teste. V kombinaciji z SQLite podatkovno bazo, ki se lahko v celoti zaganjanja v glavnem pomnilniku in je zato zelo hitra, lahko napišemo avtomatske teste, ki preverjajo ustreznost fizičnega podatkovnega modela objektom napisanih v izbranem objektnem programskem jeziku. Ravno ti testi ustreznosti pri shranjevanju v trajno hranjenje (ang. persistence tests) so ponavadi vir mnogih težav in se v primeru, da jih izvajamo na pravem RDBMS, zelo dolgo izvajajo.

Seveda pa tudi ORM orodja niso brez slabosti. Največkrat se problemi pojavijo pri uporabi orodja v napačne namene. Tu imamo v mislih predvsem »batch« procesiranje podatkov kot na primer množično brisanje ali popravljanje. To privede do cele množice nepotrebnih SQL stavkov, ki jih razvijalec s svojim poznavanjem vsebine lahko mnogo bolje optimizira oziroma vse skupaj opravi z manj zahtevki na bazo. Velika napaka, ponavadi neizkušenih in nepozornih, razvijalcev pa je tudi, da pustijo ORM orodju čisto svobodo in jih to pripelje do zelo slabo načrtovane baze. Fizični podatkovni model se popolnoma pokori objektnemu modelu in postane popolnoma neprimeren za uporabo na relacijski podatkovni bazi.

Za ORM orodje smo uporabili dobro znano odprtokodno rešitev NHibernate, ki je že dalj časa najbolj priljubljena izbira med razvijalci na .NET ogrodju. V svoji osnovi je prepis prav tako odprtokodnega ogrodja iz javanskega sveta Hibernate. Na tej osnovi je bila razvita verzija 1,

od takrat naprej pa se je razvijalo po svoje, tako da je v sedanji različici 2 in v prihajajoči 3 vedno manj skupnega s prednikom.

Omeniti velja tudi razširitev za Nhibernate Fluent Nhibernate, ki nam omogoča konfiguracijo in preslikavo s takoimenovanim tekočim načinom (ang. fluent interface), kjer objekte uporabljamo v obliki, ki malce spomnja na tekoči naravni jezik.

Običajno se Nhibernate preslikave konfigurira v obliki XML datotek, s pomočjo Fluent Nhibernate pa to opravimo kar v C# z dejansko referenco na razrede, ki jih preslikujemo v fizični podatkovni model. Prednost tega je uporaba istega razvojnega okolja in s tem znanih orodij, ki jih uporabljamo za običajni razvoj. Tudi pri preurejanju nam tu lahko pomagajo vgrajena orodja in prevajalnik, ki nam že za časa prevajanja javi napake pri preslikavah. Če imamo preslikave definirane v XML datotekah pa to ni mogoče.

Beleženje, Common.Logging in log4net

Z redkimi izjemami rabimo pri vsaki programski opremi vsaj osnovno beleženje zanimivih dogodkov. Običajno gre tu za za dogodke, ki nam pomagajo pri analizi delovanja programa ali za dogodke zanimive za uporabnike oz za uporabnike, da vedo ali se je res zgodilo tisto kar ni čisto očitno.

Naša rešitev ni nobena izjema. Uporabili smo odprtokodno knjižnico Common.Logging, ki je posplošitev knjižnic za beleženje. Nanjo preko konfiguracije priključimo eno izmed mnogih podprtih implementacij beleženja (Log4Net, NLog, Enterprise Library Logging). Odločili smo se za Log4Net.

3.4 Model domene

Poslovna logika našega sistema sicer ni med bolj zapletenimi, vendar nam uporaba modela domene omogoča:

- razvoj z manj napakami,
- lažje avtomatsko testiranje,
- hkratno obravnavo procesov, stanja sistema (v povezavi z objektno-relacijskim preslikovanjem) in dogodkov v njem.

Ker je sistem sestavljen iz repozitorija dogodkov (podobno kot to predvideva EPCIS) in informacijskega sistema na strani ribogojnice, smo razvili dva ločena modela domene. Njuno stanje je v naši implementaciji shranjeno v isti relacijski bazi, ker se vsa dejavnost odvija na enem mestu, vendar tega pri sistemu z ločenimi enotami oziroma podatki na različnih lokacijah ni težko ločiti.

Model na strani repozitorija vsebuje podatke o dogodkih povezanimi z značkami, kot so ustvarjanje značke, združevanja v večje enote, transakcije z značkami.

Model na strani ribogojnice pa je bližje običajnemu prodajnemu terminalu (ang. POS) ali informacijskemu sistemu kot si ga običajno predstavljamo.

Celoten, sicer ohlapno povezan sistem, povezuje EPC enolični identifikator, ki določa RFID značko. Tu imamo z ohlapno povezanim sistemom v mislih sistem, ki je povezan le preko vmesnikov. Posamezne komponente ne poznajo podrobnosti o implementaciji drugih komponent. Vsa komunikacija in zahteve po podatkih o značkah se izvajajo na podlagi tega identifikatorja. Z njim se dejansko izvede povpraševanje v podatkovnem modelu.

3.4.1 Model repozitorija

Sam domenski model na strani repozitorija je namenjen zgolj shranjevanju zajetih dogodkov in poizvedovanju po njih s pogoji, ki jih določajo EPCIS sporočila za poizvedbe (»EPCIS Query Control Interface«), zato so tudi zelo podobni delom sporočil, ki jih vračajo te poizvedbe.

EPCIS ločuje podatke v dva razreda – podatke o dogodkih (event data) in glavne podatke (master data). Glavni podatki so shranjeni v zapisih razredov:

- **Location**
  Lokacija na kateri se nahaja objekt potem, ko se je zgodil nek dogodek. Primeri lokacij: ribogojnica, ribarnica, predelovalni obrat, skladišče.
- **Reader**
  Čitalnik v našem sistemu. Vedno je povezan z eno lokacijo. Na eni lokaciji pa je lahko več čitalnikov, na primer v skladišču je najverjetneje en čitalnik na vhodu, drugi pa na izhodu. Za vsak dogodek je znano, kateri čitalnik ga je zabeležil, in s tem tudi kje je do dogodka prišlo.

Podatki o dogodkih:

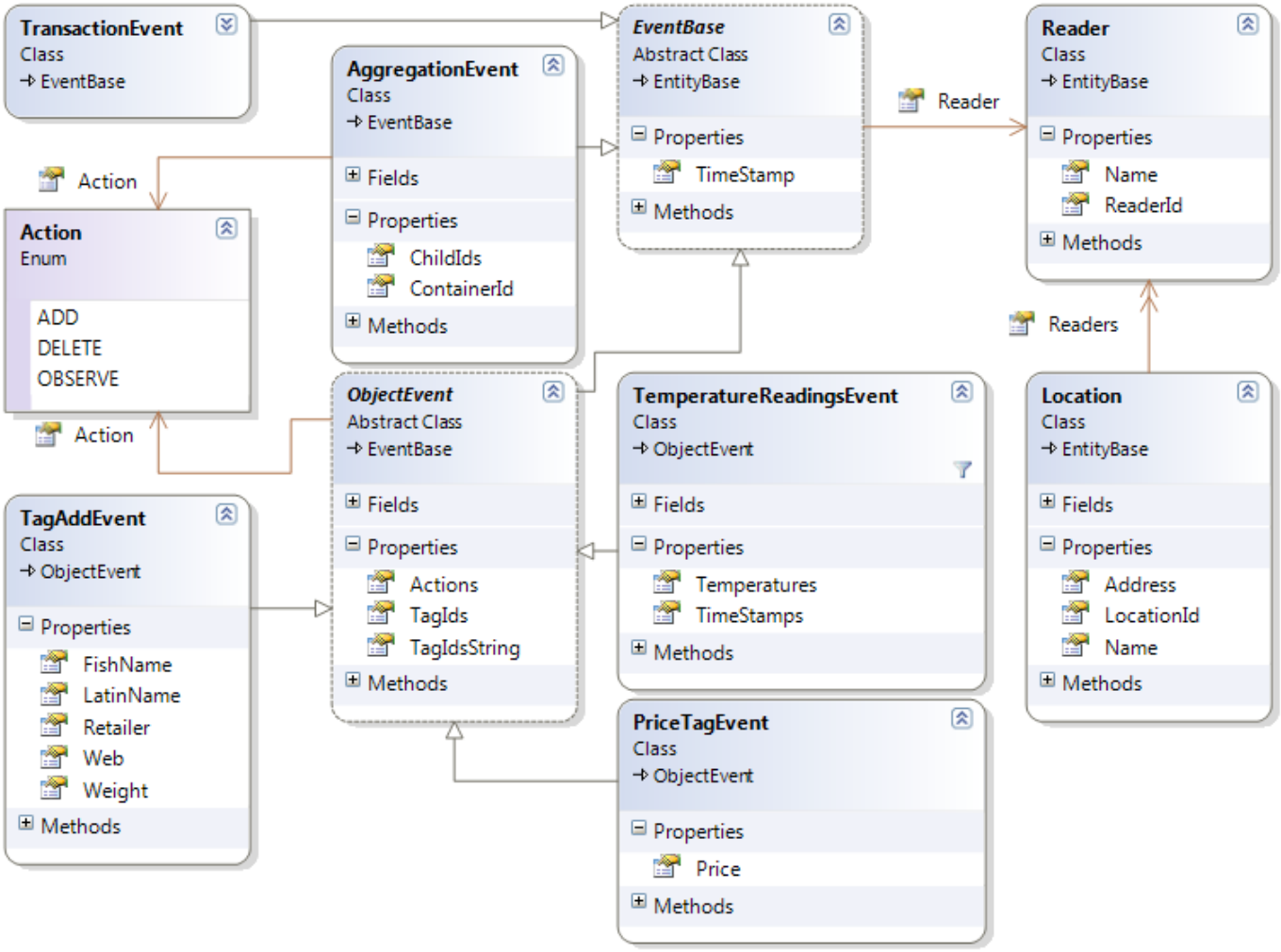
- **EventBase**
  Abstraktni razred, ki združuje lastnosti skupne vsem razredom, ki določajo dogodke. Vsak dogodek zabeležen v sistemu ima podatek o času (TimeStamp) in o tem na katerem čitalniku (Reader) se je zgodil. Tako imamo vsak dogodek umeščen v čas in prostor.
- **AggregationEvent**
  Dogodek, ki predstavlja združevanje ali razdruževanje večih objektov v enega. Določeno ima eno starševsko značko (ContainerId) in množico podrejenih značk (ChildIds), ki so od tega dogodka naprej združene v starševsko in vsi dogodki povezani z njo veljajo tudi za podrejene. Na primer pakiranje rib z značkami v posodo za transport, ki vsebuje aktivno značko za merjenje temperature. Temperature, ki jih zabeleži aktivna značka v škatli vedno veljajo za ribe, ki so bile pakirane v tej škatli.
- **ObjectEvent**
  Abstraktni razred za dogodke, namenjene dodajanju objektov v sistem, zaključku njihove uporabe ali spremembi stanja objekta. Naprimer: odpošiljanje objekta iz

obrata, sprejem objekta na drugi lokaciji ipd. Izvedenke tega razreda so tudi nosilec dodatnih informacij o predmetu označenem z določeno značko.
Konkretne izvedbe ObjectEvent razreda:

- **TagAddEvent**
  Predstavlja dodajanje značke v sistem. Obenem pa zapisu o dogodku pripišemo tudi podatke o sami ribi. Ti podatki so vrsta ribe (npr. Piranski Brancin), latinsko ime ribe (npr. Dicenthrarcus Labrax), prodajalec ali dobavitelj (npr. Fonda d.o.o.), spletni naslov (npr. http://www.fonda.si/) in teža v kilogramih. Ustrezne lastnosti na objektu so: FishName, LatinName, Retailer, Web in Weight.
- **PriceTagEvent**
  Pripis cene objektu (lastnost Price), ki ga predstavlja značka. Običajno dogodek ob sprejetju v prodajalno.
- **TemperatureReadingsEvent**
  Pripis zapisov o temperaturi v določenem časovnem obdobju (na primer med prevozom od ribogojnice do lokacije, kjer ribe filirajo). Vsebuje čase posameznih merjenj in temperature. Časi so zabeleženi v lastnosti TimeStamps, temperature pa v Temperatures. Lastnosti sta polji z enakim številom elementov tipa DateTime ali double.

- **TransactionEvent**
  Dogodek, ki opisuje povezavo ali prekinjanje povezave med fizičnim objektom opremljenim z RFID značko in eno ali večimi poslovnimi transakcijami. Ker tega dogodka v našem sistemu nismo obravnavali, nismo dodajali nobenih lastnosti, razen tistih, ki jih že definira EventBase.

Model repozitorija je prikazan na sliki 22.



Slika 22. Razredni diagram modela domene za repozitorij.

3.4.2 Model za podporo prodajnemu terminalu

Namen aplikacije za ribarnico je praktičen prikaz informacijskega sistema v poslovalnici, kjer se za sledenje oziroma ravnanje z izdelki uporablja RFID značke. Model (slika 23) je zelo okrnjen, ker cilj naloge ni bil implementacija informacijskega sistema za podporo prodajnemu mestu.

Glavni razredi:

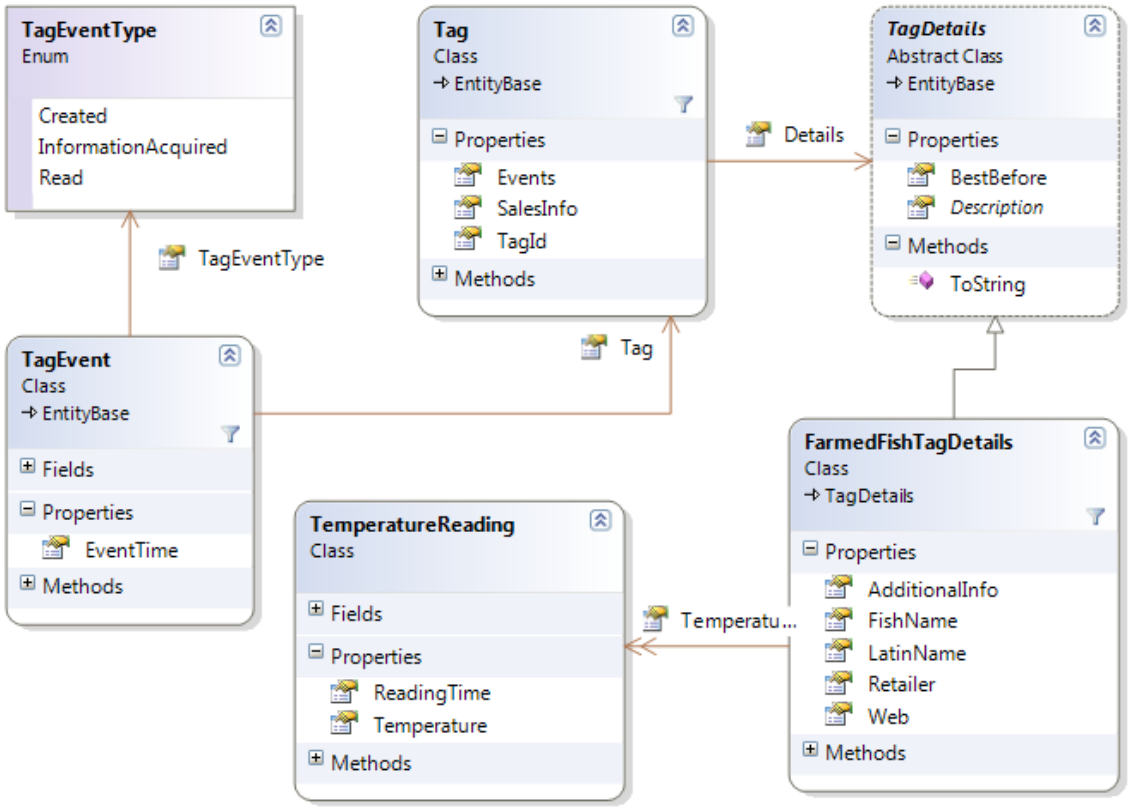
- **Tag**
  Je glavna entiteta v sistemu na katero se sklicujejo tudi praktično vsi ostali. Predstavlja RFID značko in nosilca enoličnega identifikatorja. Vsebuje tudi podatke o njegovi ceni (lastnost tipa SalesInfo) in dogodkih (lastnost, ki vsebuje seznam objektov tipa TagEvent).
- **TagDetails in njegove izpeljanke**
  Abstrakten razred, ki vsebuje bolj natančne podatke o posameznih značkah in omogoča razširljivost na nivoju modela domene. V primeru, da bi potrebovali novo vrsto značke ali da bi morali hraniti več podatkov o značkah, bi vpeljali novo izpeljanko razreda TagDetails z ustreznimi lastnostmi.
- **TagEvent**
  Predstavlja dogodke znotraj sistema ribarnice. Najbolj osnovni dogodki, ki smo jih predvideli so branje, ustvarjanje zapisa o znački in prevzem podatkov o znački s strani zunanjega vira informacij preko EPCIS sporočila. Vsebuje podatke o tem za katero značko gre (lastnost Tag), kdaj se je dogodek zgodil (EventTime) in tipu dogodka (lastnost tipaTagEventType).

Podporno oštevilčenje je tip TagEventType, ki predstavlja vse možne dogodke v tem modelu. Elementi, ki smo jih dodali in uporabili Created (dodajanje podatkov o artiklu), InformationAcquired (sprejem informacij o artiklu preko EPCIS sporočila), Read (tak dogodek se zabeleži vsakič ko preberemo značko). Služi notranji sledljivosti v aplikaciji za prodajni terminal.

V nalogi smo se osredotočili na gojene ribe, kar se odraža na lastnostih, ki jih vsebuje razred **FarmedFishTagDetails**, tj. konkretna implementacija razreda TagDetails. Ker se naloga koncentrira predvsem na prenos podatkov preko EPCISa in ne toliko na samo vsebino, je večina teh podatkov zgolj preslikava tistih, ki jih prejmemo preko EPCIS sporočil ob poizvedbi v repozitorij. Vsebovani podatki so:

- FishName
  Ime ribe, direktno preslikano iz EPCIS sporočila.
- LatinName
  Latinsko ime ribe, direktno preslikano iz EPCIS sporočila.
- Retailer
  Dobavitelj ribe, direktno preslikano iz EPCIS sporočila.

- Web
  Spletni naslov dobavitelja, direktno preslikano iz EPCIS sporočila.
- TemperatureReadings
  Polje vsebuje podatke o ribah med prevozom od dobavitelja do kraja, kjer se ribe skladiščijo in prodajajo (npr. ribarnice).
- AdditionalInfo
  Dodatne informacije, ki jih lahko uporabnik pripiše izdelku.



Slika 23. Razredni diagram modela domene.

3.5 Programsko ogrodje

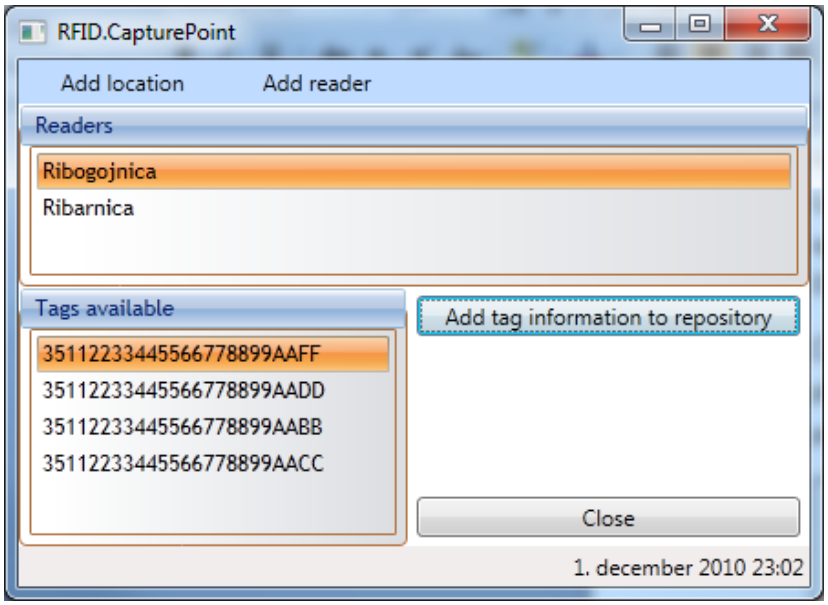
Za potrebe aplikacij v tej nalogi smo razvili manjše razvojno ogrodje, ki se deli na:

- Uporabniški vmesnik
  Podpora razvoju uporabniškega vmesnika z obrazcem MVVM (ang. Model-View-ViewModel). Vsebovano v zbiru (ang. assembly) Core.Ui. Zajeta so orodja oziroma podporni razredi za lažje upravljanje z deli uporabniškega vmesnika kot so prikazovanje oken, vezava vnosnih polj na prikazne modele (ang. ViewModel), sporočanje o dogodkih na uporabniškem vmesniku na prikazne modele, sporočanje o dogodkih v aplikaciji (npr. prebrana RFID značka, prispeli podatki o RFID znački) in podobno.

- Model
  Vsebuje vse razrede modela domene, v zbiru CapturePoint.Model za EPCIS repozitorij in v istem zbiru v imenskem prostoru CapturePoint.Model.Pos za prodajno mesto. Ta zbir praktično ni odvisen od nobene druge programske komponente in nam tako omogoča, da se pri delu na njem koncentriramo res le na razvoj modela domene.
- Shranjevanje stanja
  Vse potrebno za shranjevanje entitet iz modela v relacijsko bazo. Vsebovano v zbiru CapturePoint.Persistence. Vključuje shranjevanje s pomočjo Nhibernate ogrodja v SQLite ali SQL Server relacijsko podatkovno bazo.
- Interakcijo s strojno opremo
  Abstrakcija povezovanja s strojno opremo. Vsebuje vse, kar potrebujemo za povezovaje s »Tube« in proženje dogodkov povezanih s tem v naših aplikacijah. Nahaja se v zbiru imenovanem HardwareInteraction.

3.6 Aplikacije

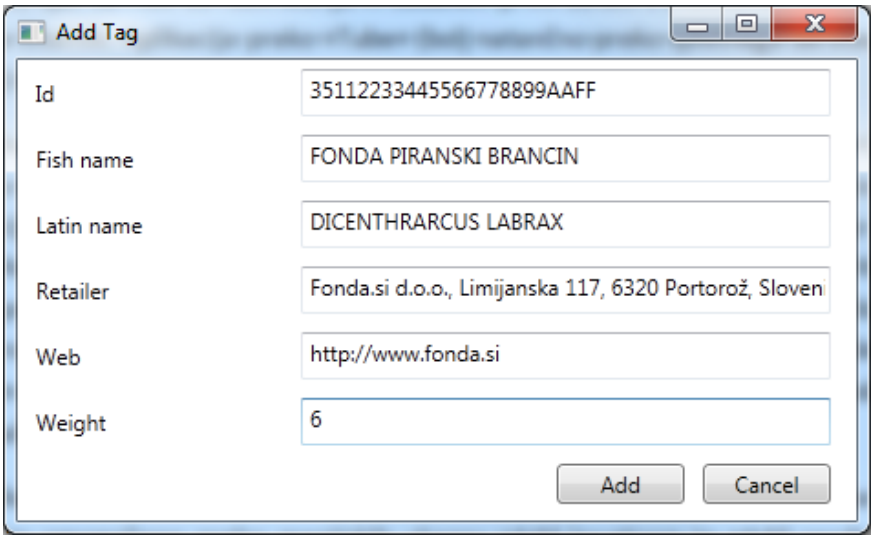
Razvili smo tri aplikacije in en servis, ki prikazujejo povezovanje preskrbovalne verige s pomočjo RFID značk in EPCIS standarda.



Slika 24. CapturePoint.

CapturePoint

Aplikacijo, ki smo jo izdelali za zajem (ang. capture) podatkov, smo poimenovali CapturePoint, torej točka zajemanja. Prikazana je na sliki 24. Namenjena je namestitvi v ribogojnico, kjer se ribo ob razvrščanju in obdelavi prvič označi. Ribo se nato postavi na čitalnik. Aplikacija preko »Tube« (bolj natančno preko spletnega servisa, ki ga le-ta nudi) izve, da je prebran identifikator RFID značke.



Slika 25. Pogovorno okno za dodajanje značke v repozitorij.

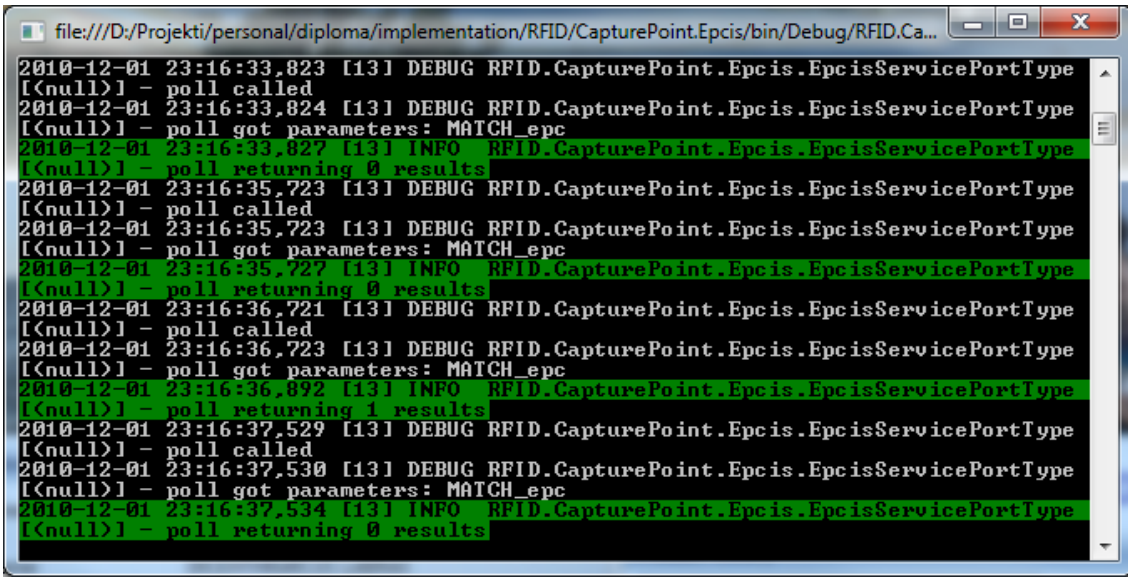
Uporabniku se na zaslonu izpiše seznam značk (zaslonski seznam »Tags available«), ki so trenutno v dosegu čitalnika. Sam čitalnik (Reader v modelu oziroma po EPCISu) in s tem tudi lokacija se izbere v seznamu Readers. Sedaj lahko izbere značko in zanjo preko gumba »Add tag information to repository« doda informacijo v sistem. Ob pritisku na gumb se prikaže pogovorno okno prikazano na sliki 25. Vnese lahko podatke o ribi, ki se potem shranijo kot TagAddedEvent v repozitorij. Od tega trenutka dalje so podatki na voljo vsem odjemalcem preko aplikacije CapturePoint.Epcis. Vnosna maska za podatke o ribi je v naši implementaciji že popolnjena z privzetimi podatki.

V tej aplikaciji ima uporabnik na voljo tudi urejanje lokacij in čitalnikov, ki so na voljo. Urejanje je omogočeno preko ukazov »Add location« in »Add reader«. Ko uporabnik doda lokacijo, lahko le-to izbere tudi pri dodajanju čitalnika. Vsak dodan čitalnik pa je na izbiro v seznamu Readers.

Ker je čitalnik RFID značk običajno hkrati tudi pisalnik, bi v tej aplikaciji lahko omogočili tudi vnos in pisanje podatkov na značko, ne le v repozitorij. S to dodatno informacijo na znački nam potem za osnovne podatke ne bi bilo treba povpraševati v EPCIS repozitoriju.

CapturePoint.Epcis

Servis, ki drugim aplikacijam nudi dostop do podatkov o dogodkih povezanih z RFID značkami oziroma EPC identifikatorji. V času razvoja smo servis razvili kot konzolno aplikacijo, ki jo je v primerjavi s pravim Windows servisom (ang. Windows service) dosti lažje opazovati (trenutno stanje in dogodki se izpisujejo na zaslon) in razhroščevati. V pravem produkcijskem sistemu bi uporabili isto programsko kodo, le poganjali bi jo na drugačen način.



Slika 26. CapturePoint.Epcis.

Aplikacija (slika 26) navzven izpostavlja spletno storitev kot jo določa EPCIS Query Control Interface. Implementacija je izvedena z WCF ogrodjem in na nastavljenem naslovu čaka na EPCIS poizvedbe v obliki SOAP XML sporočil. Ko taka poizvedba prispe, opravi SQL poizvedbo na relacijski bazi v kateri je shranjeno stanje modela repozitorija. Najdene podatke pretvori v objekte in jih vrne odjemalcu v obliki XML sporočila. Del XML sporočila odgovora je izpisan na sliki 27.

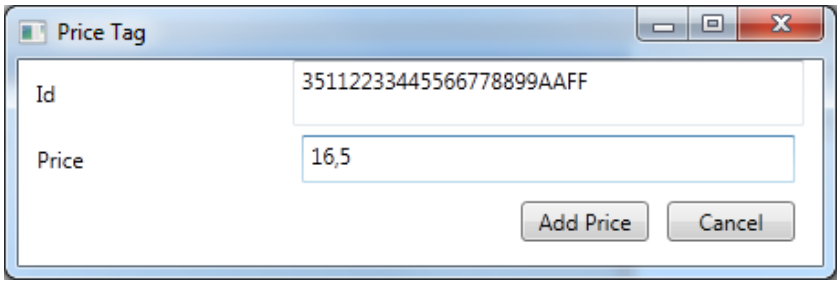
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<epcis:EPCISDocument xmlns:epcis="urn:epcglobal:epcis:xsd:1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   creationDate="2005-07-11T11:30:47.0Z"    schemaVersion="1">
 <EPCISBody>
  <EventList>
   <ObjectEvent>
    <eventTime>2005-04-03T20:33:31.116-06:00</eventTime>
    <eventTimeZoneOffset>-06:00</eventTimeZoneOffset>
    <epcList>
     <epc>urn:epc:id:sgtin:0614141.107346.2017</epc>
    </epcList>
    <action>OBSERVE</action>.
```

Slika 27. Del EPCIS XML sporočila.

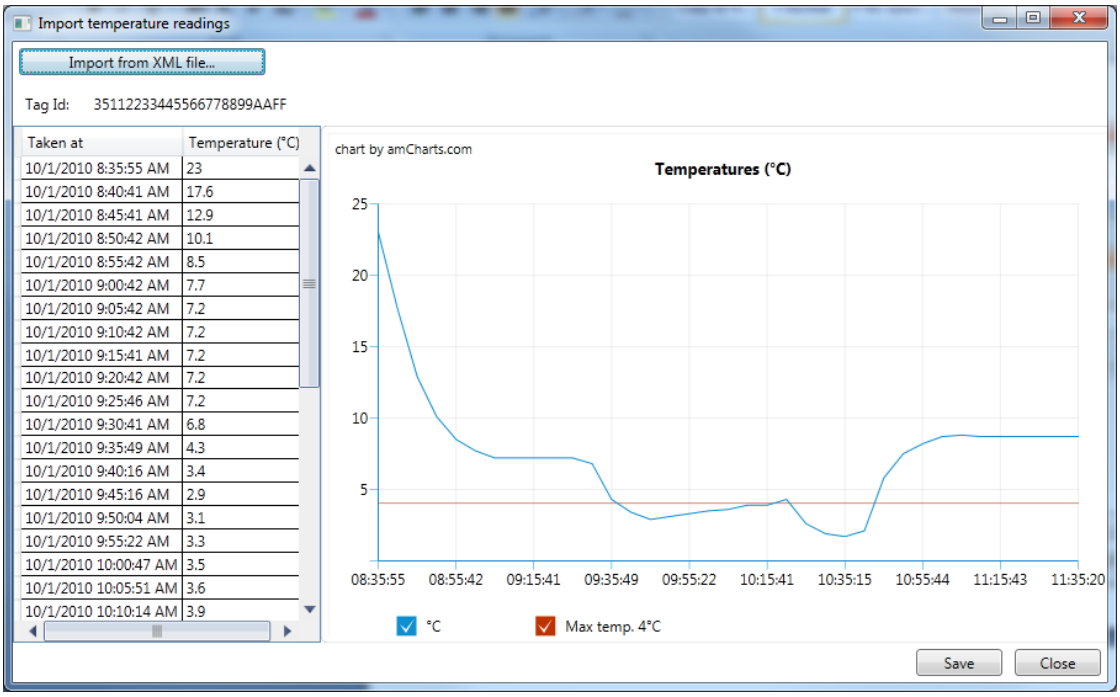
CapturePoint.Pos

Aplikacija za prodajalca rib. Omogoča vnos cene, temperatur pri prevozu od ribogojnice do ribarnice in pridobivanje podatkov od CapturePoint.Epcis.

Tako kot pri CapturePoint aplikaciji, je tudi tu pred vnosom podatkov treba izbrati RFID čitalnik, ki nam določa lokacijo na kateri zajemamo podatke. Ribo se primaknemo k čitalniku, nato jo lahko izberemo in zanjo vnesemo ceno. Klik na gumb »Add price« prikaže pogovorno okno na sliki 28. Ob potrditvi se v repozitorij shrani PriceTagEvent.



Slika 28. Pogovorno okno za dodajanje cene.



Slika 29. Pogovorno okno za uvoz temperatur.

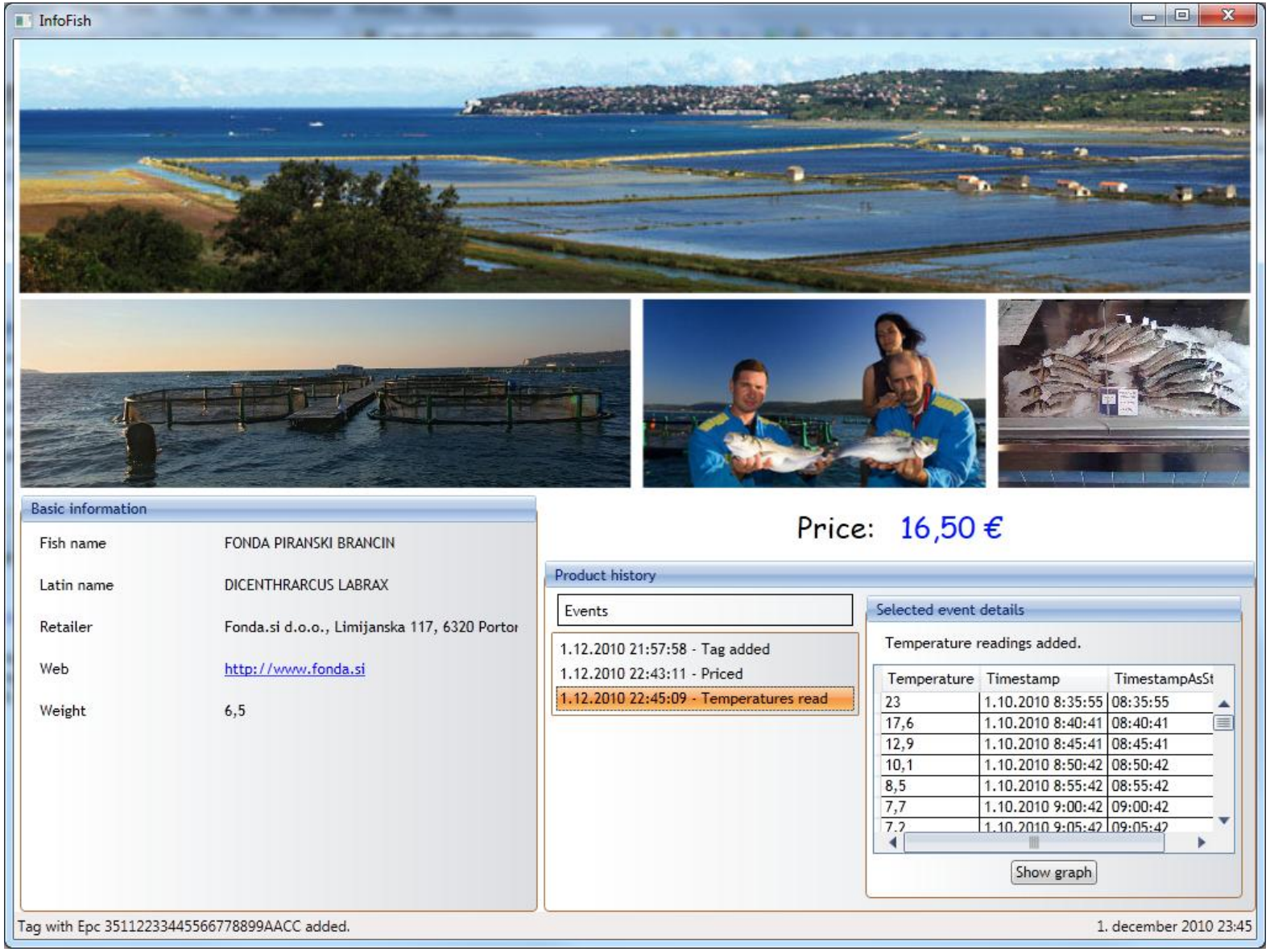
Ob kliku na gumb »Import temperatures« se prikaže pogovorno okno na sliki 29, kjer imamo na voljo uvoz temperature iz XML datoteke, ki jih izvozi aplikacija za branje aktivnih značk s

senzorjem za temperaturo. Pritisnemo gumb »Import temperatures from XML file...« in izberemo datoteko. Program te podatke prebere in prikaže v obliki tabele in grafa kot je razvidno na sliki. Ob potrditvi se v repozitorij shrani TemperatureReadingsEvent s prebranimi podatki o temperaturah in časih.

InfoFish

Aplikacija (slika 30) namenjena namestitvi na terminal z zaslonom občutljivim na dotik v trgovini ali ribarnici. Preko nje končni kupec izve podatke o izdelku, ki ga namerava kupiti. Kupec s približanjem izdelka označenega z RFID značko aktivira poizvedbo na spletni servis po EPCIS standardu. V naši implementaciji je to CapturePoint.Epcis opisan zgoraj. Lahko pa bi bil katerikoli repozitorij po tem standardu. Edina razlika je v tem, da InfoFish pozna razširitve EPCIS, ki smo jih definirali.

Ob prejetju informacij od CapturePoint.Epcis servisa se za izdelek v levem delu (»Basic information«) izpišejo osnovni podatki o ribi označeni z značko. Izpiše se tudi cena. V desnem delu pa se v razdelku »Product history« v kronološkem vrstnem redu izpišejo dogodki povezani z značko. Kot že omenjeno so v naši implementaciji ti dogodki: dodajanje značke, dodajanje cene za značko in dodajanje informacije o temperaturah med prevozom. Ob izbiri časa dogodka se v razdelku »Selected event details« izpišejo še detajli o tem dogodki. Najbolj zanimiv je seveda dogodek o temperaturah, zanj se v tabeli izpišejo temperature v časovnem zaporedju. Ob kliku na gumb »Show graph« pa se zanje izriše še graf.



Slika 30 InfoFish.

4 Zaključek

V diplomskem delu smo obravnavali RFID implementacijo sledenja v preskrbovalni verigi v programskem okolju, kot ga ponuja Microsoft .NET ogrodje. Za strojno osnovo smo uporabili EPC Gen 2 RFID značke in Feigov UHF čitalnik. Vse komponente smo med seboj povezali v sistem, kot bi ga za svoje delovanje lahko uporabilo podjetje, ki se ukvarja s prodajo gojenih rib. Pri povezavi smo uporabili standard EPCIS.

Izbrano programsko ogrodje in programske knjižnice (odprtokodne in zaprtokodne) so se izkazali za prave in bi jih uporabili tudi v nadaljnem razvoju. Aplikacije, ki smo jih razvili v okviru diplomskega dela so prikaz izvedljivosti sistema sledljivosti. Za praktično uporabo v vsakodnevnem poslovanju podjetja pa bi bilo treba še marsikaj dopolniti. Informacijski sistem bi bilo potrebno prilagoditi zahtevam podjetja tako da omogoča čim bolj enostavno uporabo v uporabniku prijaznem okolju.

Za resnično umestitev v preskrbovalno verigo bi morali implementirati še druge dogodke EPCIS (najpomembnejša sta združevanje in razdruževanje). Za boljši izkoristek računalniških sredstev bi morali razviti tudi »EPCIS Query Callback Interface« in v celoti implementirati »EPCIS Query Control Interface«.

Področje RFID je tehnološko zrelo. Za vse ravni obstajajo rešitve. Tudi branje značk deluje zelo zanesljivo. Manjkajo pa definitivni standardi s strani organizacije ISO. Glede na to, da se standardi razviti v okviru projektov Evropske komisije poravnavajo z uporabljenim standardom EPCIS, je le-ta zelo verjetno dobra izbira za uporabo v razvoju podobnih sistemov.

Možnosti za razširitev opisanih rešitev je veliko. Delo bi lahko razširili z implementacijami za različne čitalnike in dodali tudi podatke drugih ponudnikov, tako kot smo to naredili za uvoz temperatur med prevozom. Vsak dobavitelj se povezuje z logističnimi podjetji, ki zanj opravljajo prevozne storitve. Če imajo ta podjetja že v uporabi svoj sistem sledenja z RFID, potem lahko zabeležene dogodke izpostavijo preko EPCIS storitev in se tako še razširi število dogodkov, ki so na voljo. Omeniti pa velja tudi možnost povezave sistema z računovodskim programom v podjetju. Ob vnosu podatkov o artiklu v implementirani sistem bi se ti podatki lahko avtomatsko prenesli tudi v računovodski sistem.

Drugi sistemi v projektu »RFID from Farm to Fork« bi se lahko na sistem razvit v okviru te naloge povezovali preko EPCIS storitev. Prav tako se lahko informacijski kiosk razširi z uporabo podatkov, ki jih nudijo drugi servisi tega projekta.

Literatura

[1] (2010, Avgust) Active RFID tag with temperature sensor. Dostopno na: http://news.directindustry.com/press/shenzhen-friendcom-technology-development-co/active-rfid-tag-with-temperature-sensor-53968-341173.html

[2] (2010) Alien Technology. Dostopno na: http://www.alientechnology.com/tags/index.php

[3] (2008, Januar) Sam's Club Tells Suppliers to Tag or Pay. Beth Bacheldor. Dostopno na: http://www.rfidjournal.com/article/articleview/3845/1/1/

[4] Biomark. Dostopno na: http://www.biomark.com/RFID-tags.htm

[5] (2010) Dependency Injection. Dostopno na: http://en.wikipedia.org/wiki/Dependency_injection

[6] EPCglobal. (2007, September) EPC Information Services (EPCIS) Version 1.0.1. PDF.

[7] (2010) EPCIS. Dostopno na: http://www.epcglobalinc.org/standards/epcis

[8] Health and Consumer Directorate-General European Commision. (2007) Food Traceability Factsheet. PDF dokument.

[9] (2008, Maj) Feig Electronic. Dostopno na: http://www.feig.de/images/stories/feig/Bilder/OBID/Produkte/mru200i-bearb.jpg

[10] (2010) Results from temperature monitoring at storage and transport between Norway and Japan. Eskil Forås. Dostopno na: http://www.tracefood.org/images/8/81/Temperature_documentation_.pdf

[11] Martin Fowler, "Domain Model," in *Patterns of Enterprise Application Architecture*.: Addison-Wesley, 2003, p. 116.

[12] GS1. Dostopno na: http://www.gs1.org/about/overview

[13] GS1 Slovenija. Dostopno na: http://www.gs1si.org/sntportal.asp?m=63&p=94

[14] Hallprint. Dostopno na: http://www.hallprint.com/1327/1354.html

[15] Jez Humble and David Farley, "Managing Software Configuration," in *Continuous Delivery*.: Addison Wesley, ch. 2, p. 31.

[16] Steinar Kjærnsrød, "Aligning TraceCore with EPCglobal XML formats", 2009, pp. 1-3.

[17] (2009, Januar) Lorem Ipsum barcode. Dostopno na: http://en.wikipedia.org/wiki/File:Lorem_Ipsum.png

[18] (2007, Oktober).NET framework diagram. Dostopno na: http://upload.wikimedia.org/wikipedia/commons/thumb/d/d3/DotNet.svg/500px-DotNet.svg.png

[19] Pack and Sea. Dostopno na: http://packandsea.dk/conic-fish-crate/

[20] (2010) RFID from Farm To Fork. Dostopno na: http://www.rfid-f2f.eu/

[21] (2010) RF-iT Solutions. Dostopno na: http://www.rf-it-solutions.com/

[22] (2009) Seafoodplus. Dostopno na: http://www.seafoodplus.org/

[23] (2007, Nov.) ShotCode. Dostopno na: http://en.wikipedia.org/wiki/File:Shotcode.png

[24] Introduction to RFID / EPC Adoption for DoD Suppliers. Louis Sirico and Carl Brown. Dostopno na: http://rfid.net/basics/compliance/218-introduction-to-rfid-epc-adoption-for-dod-suppliers

[25] (2010, September) High Demand Keeps Tag Prices Steady. Claire Swedberg. Dostopno na: http://www.rfidjournal.com/article/view/7901/1

[26] (2010, Junij) Scandinavian Group Finds That Tracked Cod Sells Better. Claire Swedberg. Dostopno na: http://www.rfidjournal.com/article/view/7699/1

[27] (2010) Trace. Dostopno na: http://www.trace.eu.org/

[28] (2002) TraceCore - XML Standard Guidelines. PDF.

[29] (2005) TraceFish. Dostopno na: http://www.tracefish.org/

[30] (2010) TraceFood. Dostopno na: http://www.tracefood.org/

[31] (2010) Unit Testing. Dostopno na: http://en.wikipedia.org/wiki/Unit_testing

[32] (2007, Maj) UPC barcode. Dostopno na: http://upload.wikimedia.org/wikipedia/en/7/7a/UPC-E-654321.png

[33] (2010, Maj) UPM Raflatac Web Data Sheet. PDF dokument.

[34] (2009) You-R Open Technical Overview. PDF dokument.