SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Vedran Ivanac, Nenad Katanić, Goran Kopčak

WildLife Observer

Programska potpora praćenju divljih životinja

Zagreb, 2009.

Ovaj rad izrađen je na Zavodu za primijenjeno računarstvo pod vodstvom prof. dr. sc. Krešimira Fertalja i predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2008/2009.

Sadržaj

1. Uvod

Danas kada razvoj ljudske civilizacije napreduje prema neviđenim razinama i kada ne vidimo granicu svoje superiornosti ne shvaćamo da na jednoj razini postajemo sve nazadniji. Svojim napretkom uništavamo okoliš u kojem živimo, njegove prirodne resurse i bogatstva, a sve pod paskom kapitala i zarade. Čovjek će biti superioran tek kada shvati prirodu i omogući održavati ravnotežu između prirodne ljepote i svoga napretka, razinu koja se odnosi na održivi razvoj i zaštitu okoliša. Znanstvenici na početku 21.stoljeća predviđaju izumiranje životinjskih vrsta i uništavanje životinjske raznolikosti na Zemlji. Postoje projekti pod vodstvom znanstvenika i istraživača koji pokušavaju zaštititi najugroženije životinjske vrste brinući o uvjetima u kojem žive, čimbenicima koji utječu na njihovu populaciju i postupcima koji bi vratili životinjsku vrstu u ravnotežu. Kako danas tehnologija napreduje izuzetnom brzinom trebalo bi iskoristiti njezin potencijal i upotrijebiti je u zaštiti okoliša i održivosti razvoja.

Jedna od metoda zaštite ugroženih životinjskih vrsta je praćenje životinje u njezinom prirodnom staništu kako bi se shvatilo što može pozitivno i negativno utjecati na njezino preživljavanje. Ovaj rad objašnjava s kojim ciljevima i problemima smo se suočavali, kakva smo teoretska i praktična znanja morali naučiti i primijeniti te što smo ostvarili. U početnom dijelu bit će objašnjeno kako se može odrediti lokacija životinje na temelju matematičkih modela, kako izgleda višeslojna aplikacija te kako se sinkroniziraju podaci u bazi podataka replikacijom. Nakon toga bit će objašnjeno što je „WildLife Observer“, od kojih se podsustava sastoji te će se detaljno objasniti faze i različiti pogledi na izgled i rad sustava. Tokom cijelog razvoja sustava tim je bio u uskoj suradnji sa znanstvenikom sa Veterinarskog fakulteta u Zagrebu, dr.sc. Josipom Kuskom koji radi na projektu „Velike zvijeri Hrvatske“ kao voditelj projekta te koji planira koristiti WLO sustav u svakodnevnom radu.

2. Opći i specifični ciljevi rada

Glavni cilj rada na projektu je pružanje podrške radu znanstvene i istraživače zajednice u praćenju i zaštiti divljih životinja u prirodi. Kroz pružanje podrške definirani su podciljevi koji su se odnosili na zahtjeve korisnika i uvjete korisnikovog rada.

- **Podrška za praćenje životinja** – Pružiti podršku za praćenje životinja putem radio telemetrije na način da se koristi postupak i ocjena pogreške opće prihvaćen od znanstvene zajednice.
- **Mobilnost –** Tokom terenskog rada sustav ne smije ometati rad istraživača na način da smanjuje kvalitetu. Podsustav na džepnom računalu ne smije ovisiti o mogućnosti spajanja na mrežu tokom svog primarnog rada na terenu.
- **Globalnost –** Istraživanje ne provodi jedna osoba već grupa osoba pa sustav mora omogućiti da korisnici rade istovremeno i da podaci prikupljeni od drugih osoba budu brzo i jednostavno biti dostupni svima koji koriste sustav.
- **Lakoća korištenja –** Korištenje sustava mora biti lako za naučiti i koristiti.

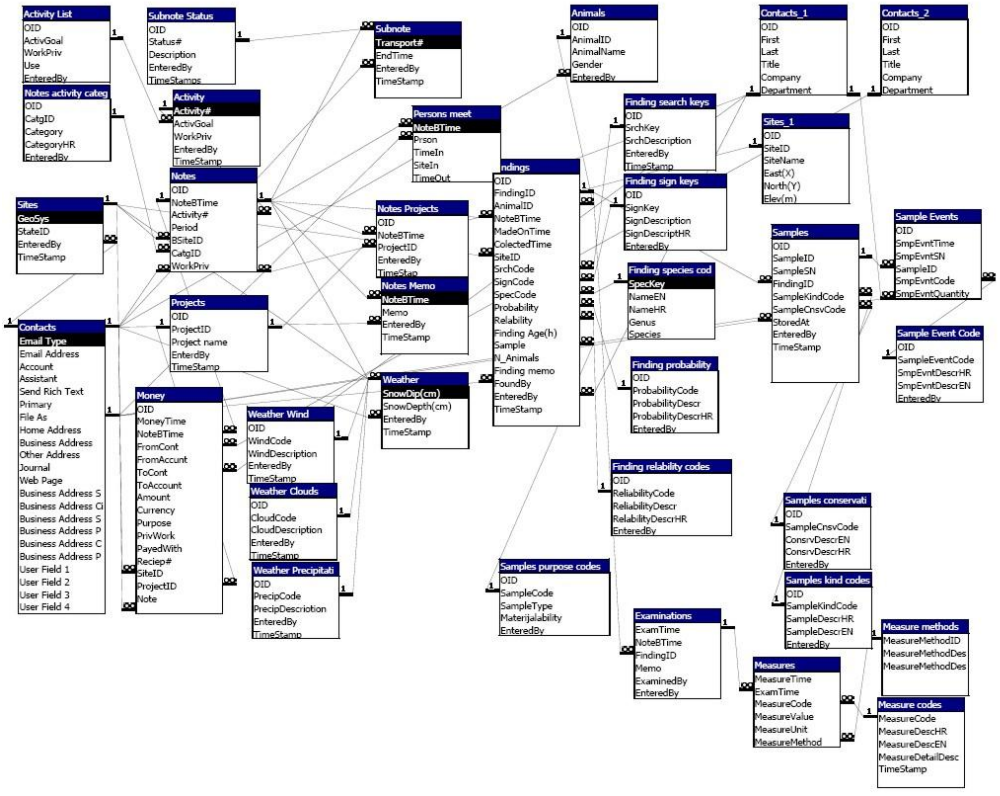
Tokom analize, intervjuiranja i prikupljanja zahtjeva shvatili smo probleme i potrebe krajnjeg korisnika [22]. Uočeno je da korisnik koristi neprikladna rješenja za uvjete na terenu i u radu. Korisnik koristi rješenja koja nisu međusobno povezana pa mora više puta iste podatke upisivati na različita mjesta. U Access bazu podataka u koju je upisivao podatke teško ili nikako nije mogao dijeliti sa svojim suradnicima, te je imao slabu mogućnost u daljnjem pretraživanju i analizi podataka prikupljenih na terenu. Tokom rada na terenu bio je primoran koristi više računalnih platformi, jednu za navigaciju, a drugu za lociranje životinja radiotelemetrijom koristeći program na DOS sustavu. Korisnikove potrebe su definirane podciljevima navedenih na početku što ne bi bilo moguće korištenjem do tada korištenog sustava.

3. Analiza postojećeg rješenja

U ovom poglavlju dan je kratki osvrt na postojeću programsku potporu koja se koristi pri terenskom istraživanju divljih životinja u Hrvatskoj. Svi podaci prikupljeni su od dr. Josipa Kuska i predstavljaju opis programske potpore koju on koristi u svakodnevnom znanstvenom radu. Cjelokupnu programsku potporu koju kao pomoć pri radu koristi dr. Josip Kusak predstavljaju baze podataka koje služe za pohranu i arhiviranje povijesnih podataka o divljim životinjama.

3.1 Access baza podataka

Glavna baza podataka za unos podataka prikupljenih na terenu izrađena je u programu Microsoft Access 2003. Baza je namijenjena pohrani svih vrsta podataka potrebnih za znanstvenu djelatnost dr. Josipa Kuska. Relacijski dijagram baze podataka dan je slikom 3-1.



Slika 3-1: Access baza podataka

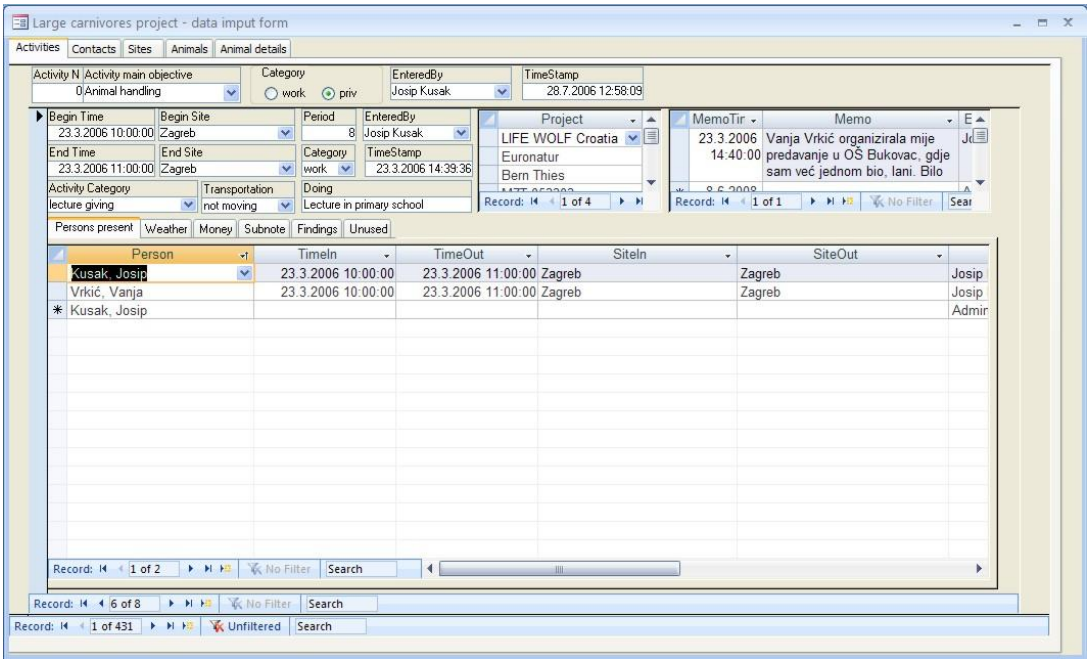
3.1.1 Model podataka

Prikazana baza podataka sadrži tablice namijenjene unosu različitih vrsta podataka prikupljenih s terena. Slijedi kratki opis najvažnijih tablica baze podataka.

- **Animals –** pohrana podataka o jedinkama koje se promatraju
- **Activity –** podaci o pojedinoj istraživačkoj (ili nekoj drugoj) aktivnosti
- **Sites –** podaci o različitim lokalitetima
- **Notes –** podaci o pojedinim terenima na kojima se vrše istraživanja, a povezani su s nekom aktivnošću koja je definirana u tablicu „Activity“
- **Findings –** podaci o različitim opažanjima s terena
- **Samples –** uzorci prikupljeni na terenu

3.1.2 Prikaz i unos podataka

Za pregled podataka pohranjenih u bazi koriste se forme za prikaz i unos podataka izrađene također u programu Microsoft Access 2003. Izgled formi s prikazanim podacima dan je na slici 3-2.



Slika 3-2: Forma za pregled i unos podataka

Prikazane forme omogućuju pregled i ažuriranje podataka iz baze podataka:

- Aktivnosti
- Terene povezane uz određenu aktivnost
- Sudionike na pojedinom terenu
- Atribute terena (temperatura, vrijeme...)
- Podatke o posjećenim lokalitetima
- Opažanja i prikupljene uzorke
- Podatke o jedinkama
- Podatke o osobama (kontaktima)

Opisani model i forme za unos i ažuriranje podataka sadrže brojne nedostatke, a njihov detaljniji opis i prijedlozi za poboljšanje dani su u poglavlju 3.3.

3.2 Paradox baza podataka

Drugi dio postojeće programske potpore predstavlja baza podataka za unos podataka o geografskim lokacijama životinja. Ovi podaci prikupljani su dugi niz godina i to na dva osnovna načina:

1. **Telemetrija –** postupak mjerenja radiosignala sa VHF/UHF ogrlice (prethodno postavljene na jedinku) i određivanje lokacije jedinke korištenjem algoritma triangulacije
2. **GPS/GSM –** praćenje lokacije jedinke korištenjem GPS/GSM ogrlice (također prethodno postavljene na jedinku). Omogućuje dohvat točnih geografskih koordinata jedinke putem SMS poruke poslane s GPS/GSM ogrlice.

Ovi podaci pohranjuju se u bazu podataka Paradox (ručno, izravno u pojedinu tablicu) koja je potpuno neovisna od Access baze podataka opisane u poglavlju 3.1 (što je također jedan veliki nedostatak modela). U nastavku je dan popis najbitnijih tablica baze podataka i kratki opis njihovog značenja na temelju čega ćemo moći producirati zahtjeve na novi, poboljšani model podataka kojeg želimo izraditi.

- **Animals –** podaci o jedinkama (onima koje nose ogrlice)
- **Bearings –** podaci o mjerenim signalima (i smjerovima) u postupku telemetrije – potrebni za određivanje položaja jedinke primjenom algoritma triangulacije

- **Collars –** podaci o raspoloživim ogrlicama (model, proizvođač, tip, aktivnost i sl.)
- **CollarsUse –** povezanost ogrlica s jedinkama koje ih nose
- **Locations –** podaci o lokacijama jedinki izračunatih algoritmom triangulacije

	Date	Time	Animal	Station	X_coord	Y_coord	Bearing	Activity	Person#	SignalQ	SignalB
1	26.11.2002	10:36	24	0	5458204	5040151	272	a	1	2	1
2	13.1.2006	16:25	24	0	5463218	5034619	0	a	1	6	2
3	13.1.2006	16:20	23	0	5463218	5034621	0	a	1	6	2
4	12.3.2006	14:45	19	0	5459470	5025150	275	p	63	3	2
5	5.9.2006	16:27	6	0	5465921	5050612	97	a	60	2	1
6	6.9.2006	11:37	6	0	5464900	5046355	165	a	60	4	1
7	6.9.2006	11:26	6	0	5463353	5047028		0	60	0	0
8	6.9.2006	12:16	6	0	5466119	5042924	174	a	60	3	1
9	6.9.2006	12:10	6	0	5466472	5042856	313	a	60	3	1
10	6.9.2006	11:55	6	0	5465455	5044911	203	a	60	3	1
11	6.9.2006	11:44	6	0	5465448	5046513	209	a	60	3	1
12	6.9.2006	12:22	6	0	5465783	5042463	328	a	60	3	2
13	6.9.2006	12:05	6	0	5466318	5043447	0	0	60	0	2
14	7.9.2006	11:40	6	32	5467680	5040800		0	60	0	0
15	7.9.2006	12:24	6	0	5468155	5046543		0	60	0	0
16	7.9.2006	11:30	6	0	5466913	5047665		0	60	0	0
17	7.9.2006	17:27	6	21	5473007	5036220	296	a	60	2	1
18	7.9.2006	19:21	6	0	5464541	5045705		0	60	0	0

Slika 3-3: Podaci iz tablice "Bearings" baze podataka Paradox

3.3 Nedostaci postojećeg modela

Kao što već u prethodnim poglavljima napomenuli, prethodno opisana „programska potpora“ koja se koristi pri terenskom istraživanju divljih životinja sadrži brojne nedostatke. Točnije, mogli bi reći da programska potpora u punom smislu niti ne postoji za ovakvu vrstu terenskog istraživanja. Budući da trenutno na raspolaganju imamo samo bazu podataka (bez ikakve prateće programske podrške) u nastavku ćemo nešto detaljnije analizirati njezinu strukturu i opisati neke najuočljivije nedostatke.

1) Baza podataka ne zadovoljava treću normalnu formu (3NF) – nije normalizirana

Postupci normalizacije baze podataka omogućuju nam da se postupno, točno definiranom metodom, odredi zamjena za loše koncipiranu relacijsku shemu. Loše oblikovana shema baze podataka rezultira različitim problemima pri pohrani podataka:

- Redundancija podataka
    - neracionalno korištenje prostora za pohranu podataka
    - anomalija unosa novih podataka

- anomalija izmjene već upisanih podataka
- anomalija brisanja podataka

- Pojava lažnih n-torki

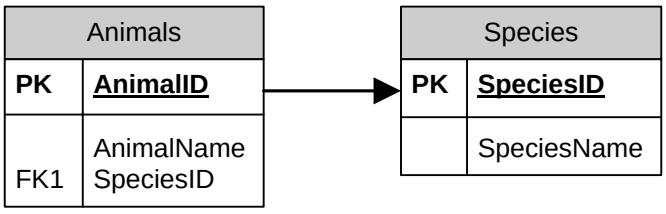
Iako redundanciju podataka na postojećem modelu možemo uočiti na više mjesta, pokažimo je na primjeru tablice „Animals“ čiji su podaci prikazani na slici 3-4.

	AnimalTmID	AnimalID	AnimalName	Species
1	20	W13	Chiara	Wolf
2	19	W12	Sara	Wolf
3	0	WX	Unknown	Wolf
4	23	W14	Noah	Wolf
5	9	W07	Jelica	Wolf
6	10	W08	Felix	Wolf
7	14	W11	Eva	Wolf

Slika 3-4: Podaci iz tablice "Animals" baze podataka Paradox

Na primjeru tablice „Animals“ prikazane na slici vidimo očit primjer redundancije podataka. Točnije, na više mjesta u tablici pojavljuje se isti redundantni podatak – atribut „Species“. Svaki puta kad se unosi novi podatak u bazu (nova jedinka), mora se ponovo unositi i vrsta kojoj ta jedinka pripada iako zapis o toj konkretnoj vrsti već postoji zapisan u tablici (za neku drugu jedinku). U prikazanom primjeru, za svaku jedinku unešenu u tablicu moralo se ručno unositi da ona pripada vrsti „Wolf“. Nadalje, kod ovakve strukture treba paziti da za sve jedinke koje pripadaju istoj vrsti, atribut „Species“ ima jednaku vrijednost da bi se zadržala konzistentnost podataka. Osim toga, ukoliko odlučimo promijeniti zapis o određenoj vrsti, taj ćemo podatak morati ručno mijenjati za sve jedinke koje pripadaju toj vrsti (anomalija izmjena). Naposljetku, ukoliko obrišemo sve jedinke koje pripadaju određenoj vrsti, podatak o toj vrsti više neće biti zapisan u bazi podataka (anomalija brisanja).

Rješenje ovog problema pronalazimo u normalizaciji sheme relacije „Animals“ na treću normalnu formu (3NF). Ispravna shema relacije „Animals“ (nakon provedene normalizacije) prikazana je na slici 3-5.



Slika 3-5: Povezanost relacija nakon normalizacije

Normaliziranjem sheme relacije „Animals“ nastaju dvije nove (međusobno povezane) relacije:

- **Animals**
  - AnimalID – jedinstveni identifikator jedinke
  - AnimalName – ime jedinke
  - SpeciesID – identifikator vrste kojoj jedinka pripada (strani ključ koji se poziva na novonastalu relaciju „Species“)
- **Species**
  - SpeciesID – jedinstveni identifikator pojedine vrste
  - SpeciesName – naziv (ime) vrste

Normaliziranjem sheme relacije na način prikazan u prethodnom primjeru uklanjamo problem redundancije podataka. Ukoliko sada, primjerice, odlučimo promijeniti zapis o nekoj vrsti, dovoljno je to učiniti samo jedanput u tablici „Species“ promjenom atributa „SpeciesName“ za konkretnu vrstu kojoj mijenjamo ime. Osim toga, sada u bazi podataka imamo samo jedan zapis za svaku pojedinu vrstu (racionalno iskorištenje prostora za pohranu podataka).

2) Dvije odvojene baze podataka

Još jedan veliki nedostatak postojeće programske potpore nalazimo u činjenici da za podatke namijenjene istoj vrsti terenskog istraživanja imamo dvije odvojene baze podataka (različitog tipa – Access i Paradox) koje, kako je već objašnjeno, nisu normalizirane. Ponovo, spomenuti nedostatak najbolje ćemo pokazati na primjeru. Ukoliko je istraživač metodom telemetrije (i korištenjem algoritma triangulacije) locirao određenu jedinku te s nje prikupio nekakve uzorke, dobivene podatke mora ručno unositi u dvije odvojene baze podataka. Telemetrijske podatke mora unijeti u bazu podataka Paradox, a podatke o prikupljenim uzorcima u Access bazu podataka. Osim toga, neki podaci će biti potpuno istovjetni, a upisani u dvije odvojene baze podataka (primjerice, ime jedinke).

Opisani slučaj je prilično otežavajuća okolnost pri terenskom radu. Tim više što nad opisanim modelom podataka ne postoji nikakva poslovna logika koja bi povezivala i obrađivala pohranjene podatke na način da korisniku omogući automatiziran unos podataka, ali i njihov jednostavan dohvat prema različitim kriterijima postavljenim od strane samog korisnika.

3) Prikaz i unos podataka

U poglavlju 3.1.2. opisane su zaslonske maske (korisničko sučelje) izrađene u programu Microsoft Access 2003, a koje se koriste za prikaz, unos i ažuriranje podataka pohranjenih u bazi podataka. Neki od glavnih nedostataka ovog korisničkog sučelja su slijedeći:

- **loš dizajn i raspored kontrola**
  - narušena preglednost podataka
  - prekomjerno sažimanje kontrola na malom prostoru
- **neintuitivno sučelje**
  - korisnik mora „pogađati“ što pojedini podatak predstavlja
- **redundancija podataka**
  - isti podatak nepotrebno se pojavljuje na više različitih mjesta (uzrok tome leži u lošoj shemi baze podataka)
- **loša navigacija kroz podatke**
  - korisnik mora pogađati koji se navigator koristi za koju logičku skupinu podataka
- **nedostatak područja za prikaz statusa obrade podataka**
- **kompliciran unos novih podataka**
  - nedostatak zajedničke forme za ažuriranje podataka šifrarnika – svaki novi podatak mora se ručno unositi otvaranjem tablice šifrarnika i dodavanjem željenog zapisa
- **nedostatak poslovne logike**
  - pretraživanje podataka prema različitim kriterijima
  - automatizirani unos jedne logičke cjeline podataka
  - filtriranje i sortiranje podataka

3.4 Specifikacija zahtjeva na sustav potpore mobilnom biologu

Analizom postojećeg rješenja koje se koristi pri terenskom istraživanju divljih životinja u Hrvatskoj, te provođenjem niza intervjua s dr. Josipom Kuskom, koji se opisanom programskom podrškom služi u praksi, producirano je mnoštvo zahtjeva na novi model podataka i prateću programsku potporu koja se želi izraditi s ciljem da se znanstvenicima olakša svakodnevni posao.

U nastavku je dan popis zahtjeva na sustav i željenih funkcionalnosti, te njihovo sažeto objašnjenje.

1) Baza podataka

Oblikovati i ugraditi novu, normaliziranu višekorisničku bazu podataka na poslužitelju. Nova baza podataka mora omogućiti pohranu svih vrsta podataka koje se pojavljuju u opisanim, postojećim bazama podataka (Access i Paradox). Konkretno, potrebno je omogućiti evidenciju podataka za:

- Aktivnosti
- Terene i lokalitete
- Opažanja s terena
- Jedinke
- Osobe (kontakte) i sudionike istraživanja
- Resurse
- Događaje
- Vremenske uvjete
- Uzorke i nalaze

Nad bazom podataka potrebno je implementirati spremljene procedure (engl. Stored Procedure) za automatizirani unos, izmjenu i brisanje zapisa za svaki pojedini entitet baze podataka (Jedinka, Opažanje, Osoba ...). Nadalje, potrebno je omogućiti replikaciju podataka na druga stolna računala i mobilne uređaje.

2) Klijentska aplikacija

Potrebno je razviti stolnu (desktop) aplikaciju za rad s podacima pohranjenim u ugrađenu SQL bazu podataka. Slijedi popis osnovnih funkcionalnosti koje aplikacija mora sadržavati

- Omogućiti automatiziran unos, izmjenu i brisanje zapisa pojedinih entiteta (pozivanje ugrađenih pohranjenih procedura)
- Oblikovati novo korisničko sučelje
    - ugraditi dobru navigaciju kroz različite zapise međusobno povezanih podataka
    - ugraditi izbornike za brzi pristup pojedinim skupinama podataka
    - rasporediti podatke na različite zaslonske maske (logički grupirati podatke)

- uvažavanje sadržaja – u svakom trenutku mora biti vidljiva informacija o:
  - dijelu obrade podataka (unos, izmjena, brisanje)
  - vrsti prikazanih podataka (jedinke, osobe, opažanja...)
  - količini podataka koji se prikazuju (zapis m od n)
  - mogućim akcijama (aktivne kontrole)
- ugraditi generičku zaslonsku masku za brzo i efikasno ažuriranje zapisa šifrarnika
- omogućiti pozivanje određenih dijelova programa ubrzavajućim kraticama s tipkovnice (engl. *Accelerator Key*)
- ugraditi podršku za pretraživanje, sortiranje i filtriranje podataka po različitim kriterijima

3) Mobilna aplikacija

Potrebno je razviti mobilnu aplikaciju za prikupljanje i analizu podataka na terenu. Osnovne funkcionalnosti koje aplikacija mora sadržavati:

- Omogućiti unos, brisanje i izmjenu podataka o mjerenjima radiosignala te perzistenciju tih podataka korištenjem mobilne baze podataka
- Implementirati algoritam triangulacije
- Omogućiti određivanje lokacije jedinke pomoću algoritma triangulacije (temeljem izvršenih mjerenja) i prikazivanje te lokacije na karti

4. Materijal i metode

4.1 Određivanje položaja objekta

4.1.1 Triangulacija

Triangulacija je algoritam koji nam omogućava da odredimo položaj objekta u prostoru na osnovu mjerenja signala koji taj objekt emitira. Kod praćenja životinja vrše se mjerenja radio signala. Primjena triangulacije je najčešća u potragama za crnim kutijama iz aviona i praćenju životinja sa radio ogrlicama. Triangulacija je postupak koji ne zahtjeva mnogo podataka. Potrebno je osigurati minimalno tri mjerenja u kojem je svako mjerenje određeno koordinatom položaja mjerenja i smjerom iz kojeg je dolazio najjači signal. Kao rezultat triangulacije dobivamo izračunati položaj objekta i površinu elipse pogreške koja nam daje objektivnu ocjenu uspješnosti mjerenja. Ako je površina prevelika tada izračunati položaj ne odgovara stvarnom položaju objekta i moraju se ponoviti mjerenja.

Obavili smo mjerenja i sada može započeti računanje položaja objekta u prostoru. Odredimo da mjerenja možemo jednoznačno definirati sa položajem (x_i, y_i) i smjerom θ_i u radijanima u odnosu na sjever odnosno kada koristimo kompas $\theta_i = (\pi / 180°)(90° - \text{očitanje s kompasa})$. Ako smo napravili *n* mjerenja tada ćemo imati listu od *n* elemenata $[(xi,yi),\theta i]$. Pretpostavit ćemo da su točke precizno određene, a smjerovi nezavisne varijable. Prava pozicija objekta određena je pozicijom (x, y). Cilj je odrediti *procijenjenu maksimalnu vjerojatnost* (engl. maximum likelihood estimate, skraćeno **MLE**) za (x,y), a u tu svrhu koristimo Von Mises distribuciju čija funkcija gustoće izgleda ovako [1]:

$$f(\theta_i;\ \mu_i, \kappa) = [2\pi I_0(\kappa)]^{-1} e^{[\kappa \cos(\theta_i - \mu_i)]} \qquad (4.1)$$

Gdje je *Io* modificirana Besselova funkcija i κ je koeficijent koncentracije. Pretpostavljamo da je κ jednak za sva mjerenja. Iz takve formule možemo odrediti logaritamsku funkciju

$$L = C - n\ln I_0(\kappa) + \kappa \sum_{i=1}^{n} \cos(\theta_i - \mu_i) \qquad (4.2)$$

koja se derivira po x i y i dobivaju se derivirane funkcije:

$$\frac{\partial L}{\partial x} = \kappa \sum_{i=1}^{n} \sin(\theta_i - \mu_i) \frac{\partial \mu_i}{\partial x} \qquad (4.3)$$

$$\frac{\partial L}{\partial y} = \kappa \sum_{i=1}^{n} \sin(\theta_i - \mu_i) \frac{\partial \mu_i}{\partial x} \qquad (4.4)$$

gdje je varijabla $\mu_i = \tan^{-1}[(y - y_i)/(x - x_i)]$, pa se jednadžbe (4.3) i (4.4) mogu izraziti preko x i y varijabli. Jednadžbe (4.3) i (4.4) možemo izjednačiti sa 0, ali tada su mogući problemi što L nije definiran za svako mjerenje te nije konveksan. Srećom, u realnim situacijama to nam ne predstavlja problem. Nakon što smo jednadžbe (4.3) i (4.4) izjednačili s 0 potrebno ih je napisati u drugačijem obliku bez parcijalne derivacije μ_i po x i y.
Prvo definiramo nove varijable $s_i = \sin\theta_i$, $c_i = \cos\theta_i$, $d_i = [(x - x_i)^2 + (y - y_i)^2]^{0.5}$. Potom izračunamo parcijalnu derivaciju μ_i po x i y.

$$\frac{\partial \mu_i}{\partial x} = -\frac{(y - y_i)}{{d_i}^2} \qquad (4.5)$$

$$\frac{\partial \mu_i}{\partial y} = -\frac{(x - x_i)}{{d_i}^2} \qquad (4.6)$$

Dodatno koristeći nove varijable dobivamo da je $\sin(\theta_i - \mu_i) = [s_i(x - x_i) - c_i(y - y_i)]$.
Sada možemo prikazati jednadžbe (4.3) i (4.4) u novom obliku koji neće ovisiti o κ:

$$L_x = -\sum_{i=1}^{n} (y - y_i)[s_i(x - x_i) - c_i(y - y_i)]/{d_i}^3 = 0 \qquad (4.7)$$

$$L_y = -\sum_{i=1}^{n} (x - x_i)[s_i(x - x_i) - c_i(y - y_i)]/{d_i}^3 = 0 \qquad (4.8)$$

MLE od (x,y) je rješenje sustava jednadžbi (4.7) i (4.8) i takvo rješenje će predstavljati najveću vjerojatnost lokacije s obzirom na mjerenja i pravu lokaciju objekta. Pošto se algoritam izvršava na računalu potrebno je osmisliti kako numerički riješiti sustav jednadžbi. Pošto L nije konveksan pojedini algoritmi kao što je Newtonova metoda nisu pouzdane. Sljedeći algoritam je predstavljen kao algoritam koji efikasno rješava sustav jednadžbi (4.7) i (4.8). Definiramo nove varijable ${s_i}^* = (y - y_i)/{d_i}^3$, ${c_i}^* = (x - x_i)/{d_i}^3$ i $z_i = s_i x_i - c_i y_i$, a jednadžbe (4.7) i (4.8) prikažemo kao linearni sustav Ax=b.[1]

$$\begin{bmatrix} \sum s_i\, {s_i}^* & -\sum c_i\, {s_i}^* \\ -\sum s_i\, {c_i}^* & \sum c_i\, {c_i}^* \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \sum {s_i}^* z_i \\ \sum {c_i}^* z_i \end{bmatrix} \qquad (4.9)$$

Algoritam se temelji na iterativnom rješavanju linearnog sustava (4.9). U prvoj iteraciji se definira izuzetak da je ${s_i}^* = s_i$ i ${c_i}^* = c_i$ u linearnom sustavu. Modificirani sustav se riješi, a rješenje je prva procjena lokacije $(\hat{x}_0, \hat{y}_0)$ koja se koristi u daljnjim iteracijama. Sljedeće

iteracije započinju tako da se na temelju prethodne procjene lokacije izračuna vrijednost varijabli d, s_i^*, c_i^* koje se onda koriste u rješavanju linearnog sustava (4.9). Rješenje postaje nova procjena za sljedeću iteraciju. Iteracije se ponavljaju sve dok razlika prethodne procjene i nove procjene ne bude manja od nekog jako malenog broja ε. ε se može definirati po potrebi ovisno o želji za preciznošću i brzinom algoritma. Podrazumijevana vrijednost je $\varepsilon = 10^{-5}$.

Osim osnovnog algoritma triangulacije postoje i varijacije algoritma i verzije algoritama koje su robusne i sprečavaju unošenje pogrešnih mjerenja. Varijacije algoritama se moraju koristiti ako imamo sistematičnu pogrešku u mjerenju. Sistematične pogreške mogu nastati korištenjem rotacijskih antena koje nisu u savršenoj fazi. [1][5]

4.1.2 Određivanje pogreške izračunate lokacije

U poglavlju 4.1.1 smo odredili lokaciju objekta s obzirom na mjerenja signala objekta. Kako bi se mogla odrediti preciznost određene lokacije i njezina vjerodostojnost potrebno je odrediti pogrešku izračunate lokacije. Pogrešku ćemo prikazati u obliku elipse pogreške koja grafički prikazuje područje gdje se objekt može nalaziti s određenom vjerojatnošću, a numerički u obliku površine elipse prema kojoj kvantitativno možemo odrediti da li je pogreška izračunate lokacije velika ili mala.[5] Da bi odredili pogrešku izračunate lokacije potrebno je izračunati vrijednosti standardne greške po x i y osi i korelaciju. U narednim jednadžbama koriste se varijable iz poglavlja 4.1.1, a njihova vrijednost se računa na temelju lokacije (x,y). Preduvjet izračunavanja navedenih vrijednosti je potreba za izračunavanjem inverza procijenjene vrijednosti varijable koncentracije $\hat{\kappa}^{-1}$,[1]

$$\hat{\kappa}^{-1} = 2(1-\bar{C}) + (1-\bar{C})^2\,[0.48794 - 0.82905\bar{C} - 1.3915\bar{C}^2]/\bar{C} \qquad (4.10)$$

s tim da je $\bar{C} = \sum \cos(\theta_i - \hat{\mu}_i)/n$. Sada možemo izračunati vrijednost standardne greške po x i po y i međusobnu korelaciju.

$$seX = \sqrt{\left|\frac{\hat{\kappa}^{-1}\sum c_i c_i^*}{\sum s_i s_i^* \sum c_i c_i^* - \sum c_i s_i^* \sum s_i c_i^*}\right|} \qquad (4.11)$$

$$seY = \sqrt{\left|\frac{\hat{\kappa}^{-1}\sum s_i s_i^*}{\sum s_i s_i^* \sum c_i c_i^* - \sum c_i s_i^* \sum s_i c_i^*}\right|} \qquad (4.12)$$

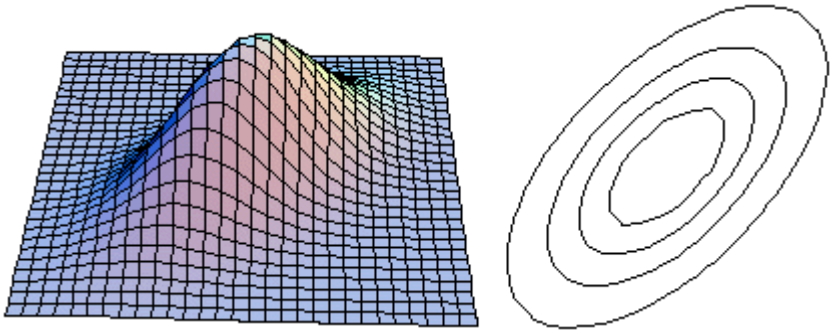
$$corr = \frac{(\sum c_i s_i^* + \sum s_i c_i^*)}{2\sqrt{\sum s_i s_i^* \sum c_i c_i^*}} \qquad (4.13)$$

Nakon što smo definirali i izračunali standardnu pogrešku po x i y i korelaciju možemo

započeti sa postavljanjem i izračunom elipse pogreške. Koristit ćemo dvosmjernu normalnu razdiobe (engl. bivariate normal distribution) koja se koristi kada želimo prikazati zajedničku distribuciju dvije slučajne varijable X,Y. Funkcija gustoće (slika 4-1) je definirana kao

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} exp\left\{\frac{-1}{2\sqrt{1-\rho^2}}\left[\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right) + \left(\frac{y-\mu_y}{\sigma_y}\right)^2\right]\right\} \quad (4.14)$$

gdje je standardne devijacije od X i Y, $\sigma_x = seX$, $\sigma_y = seY$, korelacija između X i Y $\rho = corr$, a μ_x i μ_y srednja vrijednost slučajnih varijabli X i Y koje imaju vrijednost koordinata lokacije (x,y) dobivene algoritmom u poglavlju 4.1.1. [2]



Slika 4-1: Funkcija gustoće dvosmjerne normalne razdiobe

Kako bi dobili elipsu pogreške potrebno je na nekoj visini K „odrezati“ ravninu funkcije gustoće, odnosno postaviti $f(x,y) = K$ i tada dobivamo jednadžbu elipse

$$\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right) + \left(\frac{y-\mu_y}{\sigma_y}\right)^2 = (1-\rho^2)c^2 \quad (4.15)$$

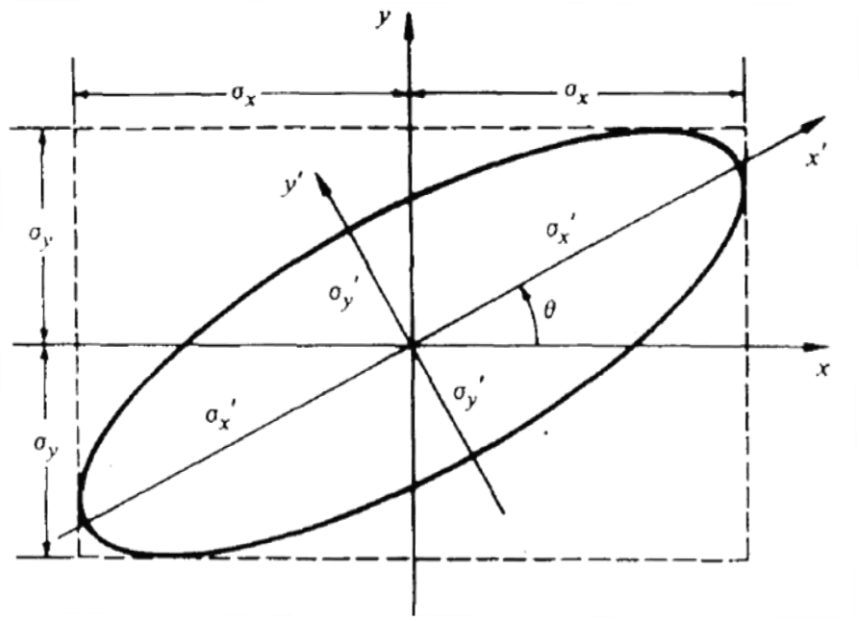
gdje je $C^2 = ln\left[4\pi K^2{\sigma_x}^2{\sigma_y}^2(1-\rho^2)\right]^{-1}$ konstanta. [2][8]

Za ocjenu površine, kuta i veličine poluosi elipse predstavit ćemo standardnu elipsu pogreške (engl. standard error ellipse) koja je definirana u ishodištu koordinatnog sustava sa c = 1

$$\left(\frac{x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x}{\sigma_x}\right)\left(\frac{y}{\sigma_y}\right) + \left(\frac{y}{\sigma_y}\right)^2 = (1-\rho^2) \quad (4.16)$$

Standardna elipsa pogreške predstavlja područje u kojem se objekt koji se locira nalazi sa 39.35% vjerojatnosti i kroz standardnu elipsu ćemo izvesti jednadžbe koje će nam eksplicitno dati vrijednosti kuta i poluosi, a koje ovise o σ_x, σ_y i ρ prikazanih na slici 4-2. Na slici 4-2 možemo vidjeti da osi elipse x' i y' nisu podudarne sa osima koordinatnog sustava x,y

odnosno razliku kutova između osi x' i x ocjenjujemo kao kut elipse i označavamo sa θ.



Slika 4-2: Standardna elipsa pogreške sa označenim parametrima pogreške i kuta

Da bi izračunali vrijednosti poluosi standardne elipse (označit ćemo ih sa σ_x' i σ_y') potrebno je izvršiti transformaciju standardnih pogrešaka σ_x i σ_y iz koordinatnog sustava x,y u koordinatni sustav x',y'. Ono što ćemo učiniti je da ćemo izraziti standardne pogreške slučajnih varijabli X,Y kao standardne pogreške slučajnih varijabli X',Y'. Jednadžba kojom možemo transformirati vektor iz jednog koordinatnog sustava u drugi je

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X' \\ Y' \end{bmatrix} \qquad (4.17)$$

i na taj način korelirane slučajne varijable možemo transformirati u nekorelirane slučajne varijable. Ako vektore slučajnih varijabli prikažemo kao kovarijacijske matrice

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} {\sigma_x}^2 & \sigma_{xy} \\ \sigma_{xy} & {\sigma_y}^2 \end{bmatrix} \qquad (4.18)$$

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} {\sigma_x}'^2 & 0 \\ 0 & {\sigma_y}'^2 \end{bmatrix} \qquad (4.19)$$

gdje je $\sigma_{xy} = \sigma_x \sigma_y \rho$. Onda jednadžba (4.17) postaje

$$\begin{bmatrix} {\sigma_x}'^2 & 0 \\ 0 & {\sigma_y}'^2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} {\sigma_x}^2 & \sigma_{xy} \\ \sigma_{xy} & {\sigma_y}^2 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad (4.20)$$

Rješavanjem jednadžbe (3.20) dobivamo eksplicitne formule za poluosi standardne elipse

pogreške σ_x', σ_y' i kuta elipse θ. [8]

$$\sigma_x'^2 = \frac{\sigma_x^2+\sigma_y^2}{2} + \left[\frac{\left(\sigma_x^2-\sigma_y^2\right)^2}{4} + \sigma_{xy}^2\right]^{0.5} \qquad (4.21)$$

$$\sigma_x'^2 = \frac{\sigma_x^2+\sigma_y^2}{2} - \left[\frac{\left(\sigma_x^2-\sigma_y^2\right)^2}{4} + \sigma_{xy}^2\right]^{0.5} \qquad (4.22)$$

$$\tan 2\theta = \frac{2\sigma_{xy}}{\sigma_x^2-\sigma_y^2} \qquad (4.23)$$

Nakon što smo definirali standardnu elipsu pogreške potrebno je odrediti elipsu pogreške za neku proizvoljnu vjerojatnost. Pretpostavit ćemo da slučajne pogreške X i Y nisu korelirane i prema tome će jednadžba standardne elipse pogreške (4.16) za nedefinirani c postati

$$\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2 = c^2 \qquad (4.24)$$

Točka koja je definirana slučajnom pogreškom X i Y će ležati unutar elipse pogreške ako vrijedi $\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2 \leq c^2$. Ako znamo da je slučajna varijabla $U = \left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2$ i da ima hi-kvadratnu razdiobu sa dva stupnja slobode, onda znamo da U ima funkciju gustoće $f(u) = \frac{1}{2}e^{-\frac{u}{2}} \quad za\ u > 0$. Prema tome vjerojatnost da točka leži unutar elipse je određena sa

$$P[U \leq c^2] = \int_0^{c^2} \frac{1}{2} e^{-u/2} du = 1 - e^{-c^2/2} \qquad (4.25)$$

Sada možemo definirati elipse pogreške za različite vjerojatnosti, a tada će poluosi standardne elipse pogreške trebati pomnožiti sa c, dok se kut ostati isti. Naravno varijabla c će se mijenjati ovisno o vjerojatnosti. [2][8]

4.2 Geografski informacijski sustav

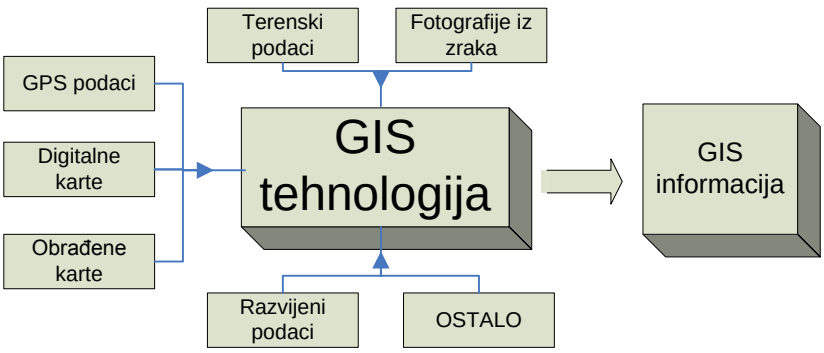
U današnje vrijeme sve je izraženiji razvoj geografskih informacijskih sustava (GIS) koji omogućavaju integraciju računala i podataka u odnosu na njihov geografski aspekt. Štoviše, može se reći da je ovo jedno od trenutno najbrže rastućih tehničkih područja u kojima se primjenjuje računalo. Karakteristike takvih sustava su mogućnosti da prikupljaju podatke iz više izvora i nakon analize ponude cjelovito rješenje.

Geografski informacijski sustav je kolekcija računalne opreme, aplikacija i geografskih podataka za prikupljanje, analizu i prikaz svih oblika geografskih informacija. Što se tiče osnovnih metoda koje se koriste, može se reći da one nisu revolucionarne i da postoje već dugo. Promatranje, informacije o terenu, mjerenje udaljenosti i ostale mjerne metode postoje odavno, ali uz pomoć GIS tehnologije i primjene računala postaju puno korisnije i pružaju dodatne funkcionalnosti za analizu čovjekove okoline. Dodatna prednost GIS tehnologije je ta što pruža korisniku mogućnost pregleda informacija na više načina, ovisno o njegovim potrebama i interesima. [13]

4.2.1 Osnove GIS-a

GIS sustave karakterizira pretvorba geografskih podataka u korisniku potrebne informacije. Svi podaci koji se prikupe mnogobrojnim metodama moraju se adekvatno pohraniti, geografski obraditi i pripremiti za prikaz.

Prije početka obrade podataka, potrebno ih je prikupiti. Podaci koji se prikupljaju dolaze u još neobrađenom obliku i karakteriziraju ih atributi okoline iz kojih su prikupljeni.



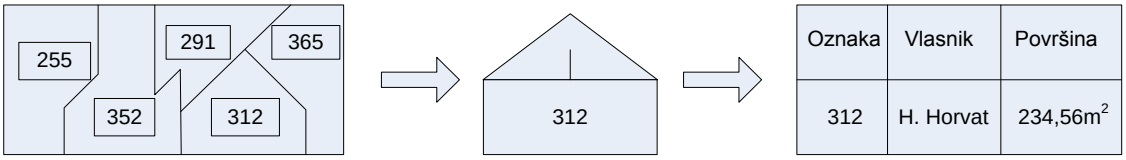
Slika 4-3: Prikaz izvora podataka

4.2.2 Geografski podaci

Geografski podaci predstavljaju interpretaciju fizičkih lokacija na površini Zemlje ili u njenoj neposrednoj blizini. Podaci su često opisani atributima koji karakteriziraju sve podatke istog tipa.

Podaci se mogu modelirati tako da se prikazuju u vektorskom prostoru. Vektorski prostor možemo zamisliti kao platformu za geografske podatke koji su predstavljeni x-y koordinatama te opisuju karakteristike nekog područja.[13]
Standardni formati za prikaz GIS vektorskih podataka se sastoje od nekoliko elemenata:

- Glavna datoteka koja sadrži svojstva podataka (eng. feature file) ,
- Identifikacijska datoteka koja sadrži identifikatore (eng. index file),
- Povezana atributna tablica sadrži atribute prostornih varijabli (eng. attribute table)



Slika 4-4: Opis vektorskih podataka

Drugi tip prikaza podataka nazivamo raster tip podataka i predstavljaju digitalne prikaze koji nastaju kao nakupine ćelija i polja podataka. Tip prikaza i boja koja se prikazuju određuju svojstva i samu pojavu točke ekranskog prikaza. Za razliku od vektorskog prikaza, raster tip podataka je bolji za prikaz kontinuiranog toka podataka te one slučajeve u kojima nije strogo definiran rubni element objekta.

Kada se prikazuju realni objekti iz prirode na izvoru podataka, prikazujemo ih kao zasebne geometrijske oblike. Postoje tri glavna geometrijska oblika kojim definiramo elemente:

- Točke (eng. Point) ,
- Linije (eng. Line) ,
- Poligoni (eng. Polygon).

Postoji i potkategorija ove podjele koja nastaje kao kombinacije osnovne tri i naziva se polilinija (eng. polyline). [13]

4.2.3 MapWindow

MapWindow je geografski informacijski sustav otvorenog koda koji može biti korišten na više načina:

- Kao stolna aplikacija za korištenje GIS tehnologije,
- Za analizu i prosljeđivanje podataka drugim korisnicima,
- Kao okruženje za razvoj samostalnih aplikacija koje se temelje na GIS tehnologiji.

Komponenta MapWindow sustava koja je korištena za razvoj WLO stolne aplikacije naziva se MapWinGIS. MapWinGIS predstavlja ActiveX koja se može implementirati u bilo koju samostalno razvijanu aplikaciju. Jedini uvjet za implementiranje je taj da je samostalna aplikacija pisana u programskom jeziku koji podržava ActiveX kontrolu. Komponenta je zapravo programski objekt koji se doda kao referenca pri razvoju vlastitog projekta.

Unutar same ActiveX kontrole postoji nekoliko klasa koje omogućuju korisniku baratanje sa podacima, bilo u smislu analize ili samog prikaza podataka:

Tablica 4-1: Popis klasa ActiveX komponente [14]

ESRIGridManager	Extents	Field	Grid	GridColorBreak
GridColorScheme	GridHeader	Image	ShapefileColorBreak	**Point**
Shape	ShapeNetwork	Shapefile	ShapefileColorScheme	**Map**
Table	Tin	Utils	Vector	

Nakon što se stvori instanca klase Map unutar vlastitog objekta, korisniku je omogućeno korištenje mnoštvo funkcija i događaja sa kojima može upravljati sa dodanom komponentom. Bitno je naglasiti da se unutar vlastito razvijane aplikacije komunicira uz pomoć dodane .dll datoteke sa MapWindow okružjem. [13]

4.3 Višeslojna programska arhitektura

Prilikom razvoja bilo koje vrste programske podrške, jedan od najvažnijih elemenata o kojem ovisi kvaliteta cjelokupne podrške jest odabir programske arhitekture. No, pri samom odabiru arhitekture, potrebno je obratiti pozornost i na niz drugih elemenata kao što su skalabilnost i performanse, te mogućnost buduće lake nadogradnje sustava. Arhitektura koja podjednako uzima u obzir sva tri navedena elementa je višeslojna programska arhitektura (engl. *multilayered arhitecture*).

Višeslojna arhitektura odnosi se na arhitekturu aplikacije koja ima najmanje tri nezavisna i odvojena logička sloja. Kod ovakve arhitekture svaki pojedini sloj komunicira samo s onim slojem koji se nalazi direktno ispod njega te ima točno definiranu funkcionalnost koju obavlja. Sve radnje koje sloj interno obavlja su potpuno skrivene drugim slojevima što nam omogućuje obavljanje izmjena na nekom sloju bez ikakvog utjecaja na ostale slojeve. Ovo je vrlo važno svojstvo višeslojne arhitekture, jer nam omogućuje jednostavnu nadogradnju nekog dijela sustava bez potrebe za izmjenama drugih dijelova. Osim logičke odvojenosti slojeva (kada se svi slojevi nalaze na istom računalu), slojeve je moguće odvojiti i fizički, u smislu da se svaki sloj nalazi na zasebnom računalu. U sklopu ovog rada govorimo o višeslojnoj arhitekturi u kojoj su pojedini slojevi logički odvojeni jedan od drugoga.
Neke bitne prednosti višeslojne programske arhitekture su:

- bolja raspodjela opterećenja sustava
- povećana skalabilnost – mogućnost ekspanzije (primjerice, povećanje broja korisnika bez preopterećenja ili potrebe za promjenom procedura)
- jednostavnije održavanje i nadogradnja sustava
- timski razvoj

S druge strane, ova arhitektura sadrži i neke nedostatke kao što su:

- vrlo složen dizajn i razvoj
- problem raspodjele podataka, procesa i sučelja
- povećano opterećenje mreže

U većini slučajeva, višeslojna arhitektura odnosi se na arhitekturu koja se sastoji od tri sloja: podatkovnog, poslovnog i prezentacijskog. U nastavku teksta bavimo se takvom vrstom višeslojne arhitekture.

U poglavlju 4.4. iznesen je kratki teorijski uvod u problematiku troslojne programske arhitekture, a u poglavlju 4.5. detaljan opis troslojnog modela i njegove implementacije na

primjeru aplikacije „wloDesktop“.

4.4 Troslojna arhitektura korisnik-poslužitelj

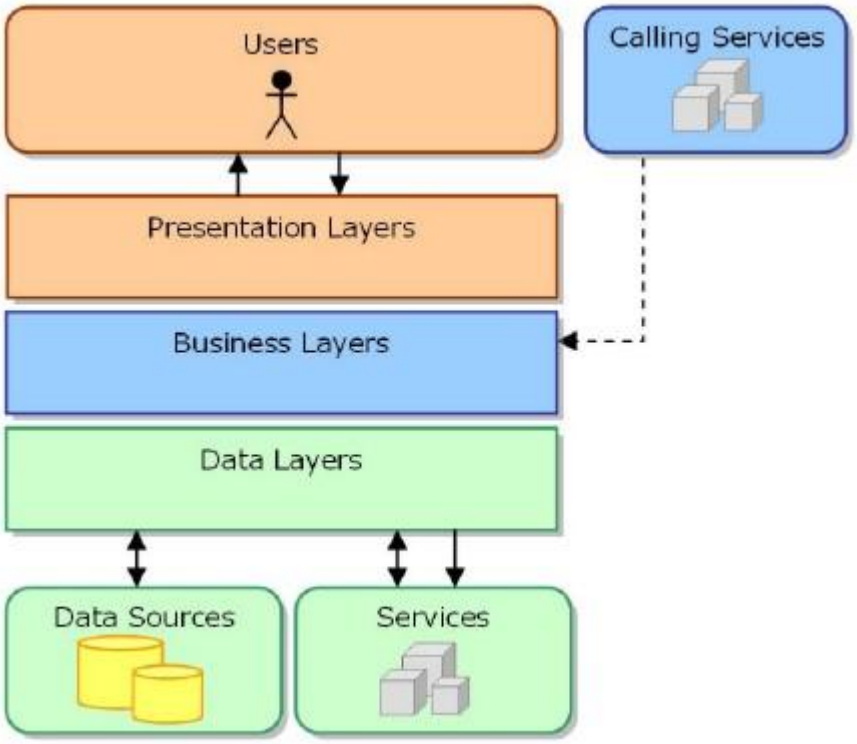
Kao što smo i ranije napomenuli, troslojna programska arhitektura je višeslojna arhitektura koja se sastoji od tri međusobno odvojena sloja [19]:

- **Podatkovni sloj (eng. *Data Layer)***
  - sloj pristupa podacima (engl. *DAL – Data Access Layer*)
- **Poslovni sloj (engl. *Business Layer*)**
  - sloj poslovne logike (engl. *BLL – Business Logic Layer*)
- **Prezentacijski sloj (engl. Presentation Layer)**
  - grafičko korisničko sučelje (engl. *GUI – Graphical User Interface*)

Troslojna programska arhitektura se najčešće koristi u slučajevima kada želimo postići efikasnu distribuiranu klijent-server arhitekturu koja nam nudi dobre performanse, skalabilnost, povećanu fleksibilnost, jednostavno održavanje i nadogradnju sustava te ponovnu iskoristivost pojedinih dijelova sustava. Nadalje, troslojna arhitektura pokazuje se kao odličan izbor pri razvoju neke aplikacije u timu. No, ova činjenica ni ne čudi, budući da je svaki sloj neovisan od ostalih, pa svaki sloj možemo razvijati neovisno o drugima i to na drugoj platformi (pa čak i u drugom programskom jeziku).

Bez obzira na sve naveden prednosti, implementacija troslojne arhitekture je iznimno težak i kompleksan posao. Jedan od potencijalnih problema na koje možemo naići je taj da način na koji ćemo razdvojiti prezentaciju podataka od poslovnu logike i pristupa podacima, nije uvijek potpuno jasan. Osim toga, samostalan razvoj troslojne aplikacije zahtjeva mnogo uloženog truda i vremena u što smo se i sami uvjerili radeći na razvoju troslojne aplikacije wloDesktop.

U nastavku teksta dan je kratki opis pojedinih slojeva troslojne programske arhitekture, a radi boljeg shvaćanja podjele arhitekture na slojeve, na slici 4-5 izložen je grafički prikaz slojeva troslojne programske arhitekture.



Slika 4-5. Troslojna programska arhitektura [7]

4.4.1 Podatkovni sloj

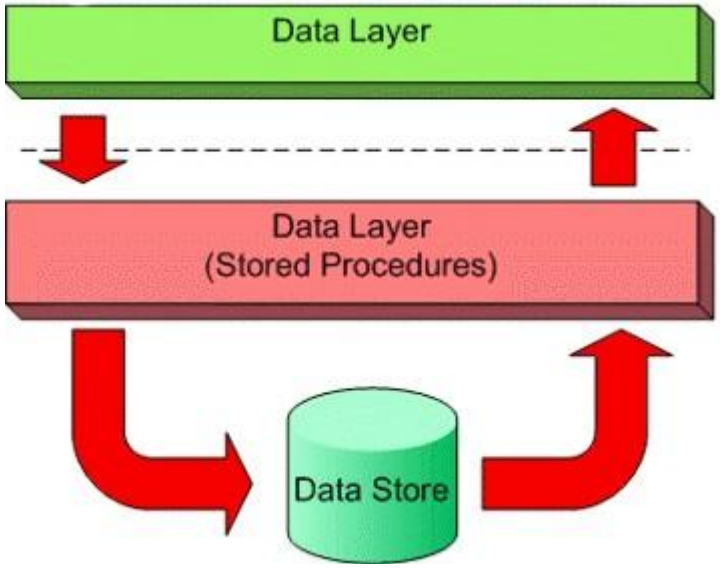
Podaci su ključna komponenta gotovo svake aplikacije [18]. Stoga, podatkovni sloj predstavlja temelj cjelokupne troslojne arhitekture, a realizira se kao zasebna komponenta čija je jedina uloga da dohvati podatke te da ih na siguran i efikasan način dostavi onome tko ih je zatražio (najčešće je to sloj poslovne logike). Ovakvim pristupom, podaci mogu biti logički ponovno iskorišteni, u smislu da različiti dijelovi aplikacije koji koriste iste podatke mogu pozivati istu metodu podatkovnog sloja za dohvat podataka (umjesto pisanja istog upita za svaki pojedini dio aplikacije). Ovo je vrlo važno svojstvo i temelj je za buduće lako održavanje sustava.

Podatkovni sloj najčešće predstavlja baza podataka (ili neki drugi izvor podataka). U .NET programskom okruženju to su u većini slučajeva SQL ili Access baze podataka, iako to može biti primjerice i Oracle baza podataka, pa čak i XML.

U mnogim implementacijama podatkovnog sloja, on je podijeljen u dva dijela (dva podsloja). Prvi se sastoji od skupa pohranjenih procedura (engl. *Stored Procedure*) koje su implementirane izravno u samoj bazi podataka, a omogućuju automatiziran unos, izmjenu,

brisanje i dohvat podataka iz baze. Drugi podsloj sastoji se od skupa klasa i metoda koje pozivaju pohranjene procedure i na taj način upravljaju podacima (engl. *DAL – Data Access Layer*). Potrebno je ugraditi po jednu klasu za svaki pojedini entitet, odnosno za svaku skupinu procedura koje omogućuju osnovne operacije nad tim entitetom (dohvat, unos, izmjena i brisanje). Spomenuti skup operacija nad nekim entitetom baze podataka poznat je i pod skraćenicom CRUD (create, read, update, delete). Bitno je napomenuti da se DAL sloj ponekad spominje i kao podsloj poslovnog sloja. U tom slučaju, podatkovni sloj predstavlja samo baza podataka s ugrađenim pohranjenim procedurama.

Slikoviti prikaz podjele podatkovnog sloja na dva podsloja izložen je na slici 4-6.



Slika 4-5. Podjela podatkovnog sloja na dva podsloja

4.4.2 Poslovni sloj

Iako aplikacija inače može komunicirati izravno s bazom podataka, u troslojnoj arhitekturi ta se komunikacija odvija preko još jednog dodatnog sloja – sloja poslovne logike. Sloj poslovne logike zaslužan je za pristup podatkovnom sloju s ciljem dohvata, izmjene, unosa ili brisanja podataka. Ovaj sloj posreduje komunikaciji između podatkovnog i prezentacijskog sloja i to na način da filtrira podatke, odnosno da vrši validaciju podataka u oba smjera (prije pohrane podataka u bazu podataka i prije isporuke podataka prezentacijskom sloju). Ovo svojstvo nam jamči da su ulazni podaci ispravni prije pohrane podataka, kao i da su izlazni podaci ispravni prije prezentacije korisniku. Opisani skup

operacija za validaciju ulaznih i izlaznih podataka naziva se poslovnim pravilima (engl. *business rules*) u smislu da su to pravila kojima se koristi poslovni sloj s ciljem donošenja određenih zaključaka vezanih uz podatke. No, s druge strane, poslovna pravila ne odnose se samo na validaciju podataka, već na bilo koju operaciju nad podacima koja je implementirana upravo unutar poslovnog sloja. Inače, potrebno je što više poslovne logike smjestiti upravo u poslovni sloj što nam kasnije omogućuje da tu istu poslovnu logiku iskoristimo na nekom drugom sustavu.

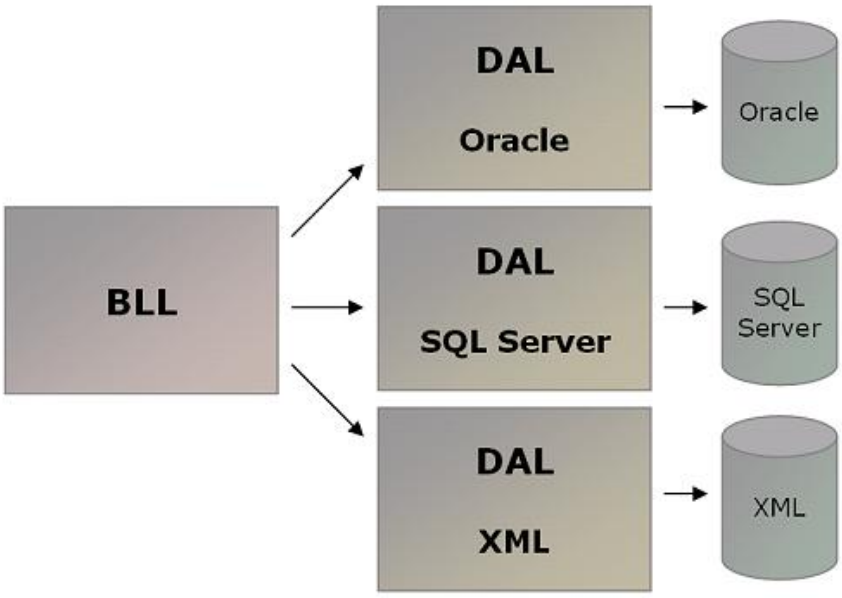
U .NET okruženju, poslovni sloj najčešće koristi *SqlClient* i *OleDb* objekte za manipulaciju podacima iz baze podataka te *DataReader* ili *DataSet* objekte za isporuku dohvaćenih podataka prezentacijskom sloju [21].

Slično opisanoj situaciji kod podatkovnog sloja, sloj poslovne logike također možemo podijeliti na dva podsloja:

- sloj pristupa podacima (engl. *DAL – Data Access Layer*)
    - u ovom slučaju DAL sloj je sastavni dio poslovnog, a ne podatkovnog sloja
- sloj poslovne logike (engl. *BLL – Business Logic Layer*)

BLL sloj nalazi se iznad DAL sloja u smislu da BLL sloj poziva i koristi klase i metode implementirane u DAL sloju. DAL sloj zadužen je za dohvat podataka iz baze podataka i prosljeđivanje tih podataka BLL sloju. Nadalje, BLL sloj obrađuje primljene podatke, validira ih i naposljetku prosljeđuje prezentacijskom sloju.

Kod opisa podatkovnog sloja spomenuli smo da on može biti realiziran bilo kojom vrstom baze podataka ili nekim drugim podatkovnim izvorom. S tim u vezi, podjelom poslovnog sloja na BLL i DAL podslojeve možemo implementirati poslovnu logiku koja će ispravno raditi bez obzira na vrstu izvora podataka. Ovo je također jedno od bitnih svojstava troslojne arhitekture. Primjerice, u slučaju dodavanje podrške za drugu vrstu izvora podataka (npr. zamjena Access baze podataka SQL bazom) potrebno je samo implementirati novi DAL podsloj za pristup podacima (bez ikakvih promjena na BLL podsloju). Ovaj princip podjele poslovnog sloja prikazan je slikom 4-7.



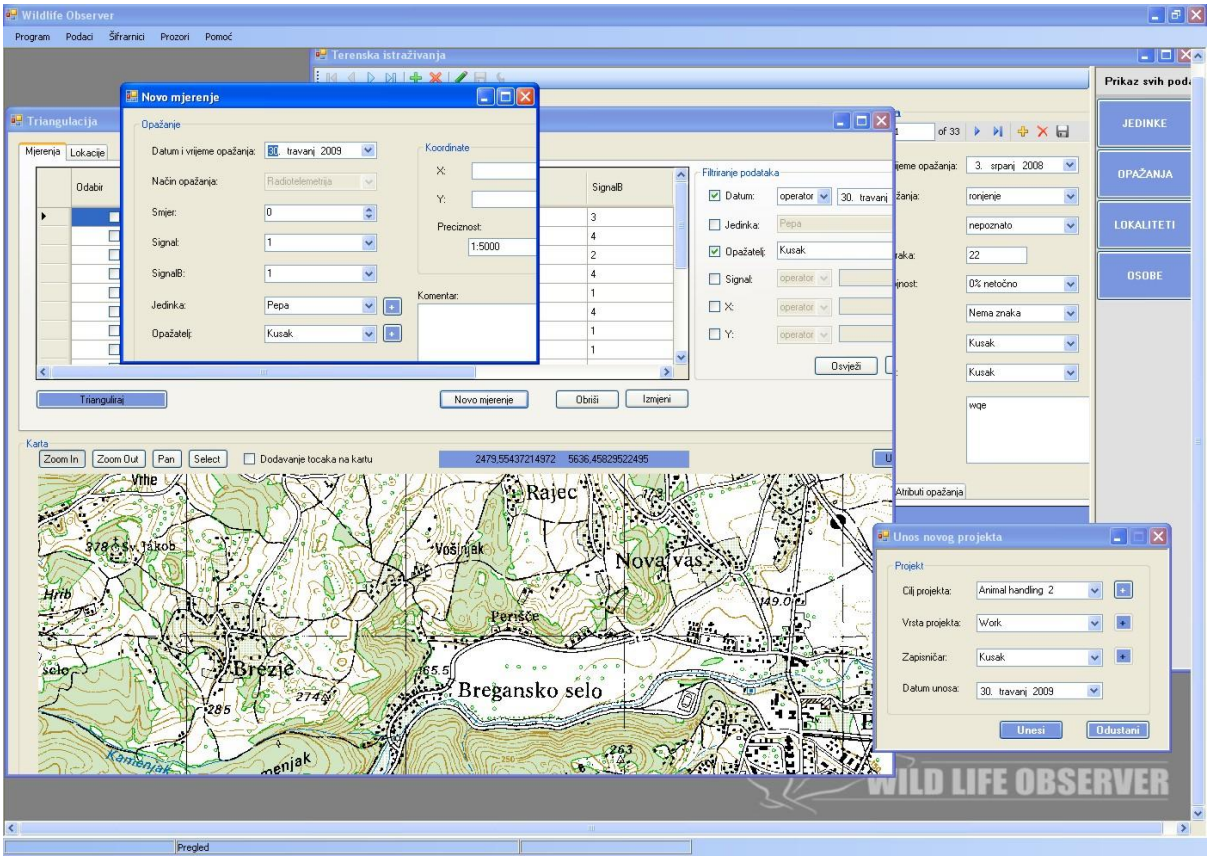
Slika 4-6: Podrška za različite izvore podataka

4.4.3 Prezentacijski sloj

Najviši sloj u hijerarhiji troslojne programske arhitekture je prezentacijski sloj. Ovaj sloj zadužen je za prezentaciju podataka korisniku, ali služi i kao sučelje preko kojeg sustav dobiva naredbe od strane korisnika. Prezentacijski sloj ne obuhvaća samo skup kontrola i formi prikazanih na ekranu korisnika (engl. *GUI – Graphical User Interface*), već i pozadinski skup svih klasa koje se koriste u pripremi prikaza podataka. Možemo reći da je ovaj sloj možda i najvažniji u cjelokupnoj troslojnoj arhitekturi iz jednostavnog razloga što je to sloj kojeg krajnji korisnik vidi i kojeg izravno koristi. Stoga, funkcionalnost i jednostavnost korištenja sustava uvelike ovise o realizaciji prezentacijskog sloja. Bez obzira na dobar model podataka i kvalitetno implementiranu poslovnu logiku, loša prezentacija rezultirat će i lošim pogledom na cjelokupni sustav.

Dobra praksa pokazuje da je što više poslovne logike potrebno izbaciti iz prezentacijskog sloja i smjestiti je upravo u poslovni sloj. Ovakav pristup omogućuje nam jednostavnu naknadnu izmjenu prezentacijskog sloja (korisničkog sučelja) bez utjecaja na poslovni sloj (i druge slojeve). Ovaj princip možemo povezati i s podjelom poslovnog sloja na podslojeve BLL i DAL u smislu da nad istom poslovnom logikom može raditi više različitih korisničkih sučelja (npr. Windows aplikacija i web aplikacija) kao što ista poslovna logika može raditi nad više različitih izvora podataka.

Primjer prezentacijskog sloja aplikacije prikazan je slikom 4-8.



Slika 4-7. Prezentacijski sloj aplikacije „wloDesktop“

4.5 Implementacija troslojne arhitekture

Na temelju opisa troslojne programske arhitekture dane u poglavlju 4.4 i zahtjeva na novi sustav popisanih u poglavlju 3.4, pokazat ćemo implementaciju modela troslojne arhitekture na primjeru aplikacije „wloDesktop“.

U narednim poglavljima dan je detaljan opis aplikacije „wloDesktop“ po pojedinim slojevima.

4.5.1 Korištene tehnologije

Prije samog opisa implementacijskih detalja aplikacije, slijedi popis tehnologija korištenih u njezinom razvoju.

1. **Microsoft SQL Server 2005 i SQL Server Management Studio**

- razvoj i implementacija modela podataka opisanog u poglavlju 4.5.2
- ugradnja pohranjenih procedura za rad s podacima iz baze podataka
- ugradnja podrške za replikaciju podataka

2. **Microsoft Visual Studio 2005 i .NET Framewok 2.0**
   - implementacija pojedinih slojeva aplikacije korištenjem programskog jezika C#
   - izrada grafičkog korisničkog sučelja

3. **Corel PHOTO-PAINT 12**
   - izrada i obrada slika korištenih pri dizajniranju grafičkog korisničkog sučelja
4. **IcoFX 1.6**
   - izrada i obrada ikona korištenih pri dizajniranju grafičkog korisničkog sučelja

4.5.2 Model podataka

Na temelju postavljenih zahtjeva na novi model podataka oblikovana je i ugrađena relacijska baza podataka u sustavu MS SQL Server. Rad cijelog WLO sustava, odnosno svih pojedinih aplikacija koje su sastavni dio tog sustava, oslanja se na SQL bazu podataka. Opisani model baze podataka temeljen je na postojećoj bazi podataka razvijenoj u sklopu projekta „Fauna Croatica Database“ te je taj model prilagođen pohrani podataka o terenskim istraživanjima divljih životinja uz mogućnost spojne replikacije podataka. Baza je projektirana na način da omogućuje efikasan i jednostavan unos novih podataka u bazu, s točno definiranim i odvojenim dijelovima za pohranu različitih vrsta podataka.

U ovom poglavlju dan je detaljan opis modela baze podataka. Budući da je baza namijenjena radu cjelokupnog WLO sustava, detaljno su opisani samo oni dijelovi modela koji se koriste u sklopu troslojne aplikacije wloDesktop.

4.5.2.1 Model entiteti-veze

U modelu baze podataka koriste se tablice (npr. Jedinka, Ogrlica, Opazanje ...) koje služe za pohranu pojedinačnih instanci entiteta iz stvarnog svijeta, odnosno za pohranu zapisa. Entiteti predstavljaju skupove podataka opisane atributima. Primjerice, entitet „Jedinka“ je opisan slijedećim skupom atributa: „IdJedinke“, „OznPrimjerka“,

„ImePrimjerka“, „IdSkupine“, „IdVrste“. Vrijednosti atributa su stvarni podaci, npr. vuk imena ImePrimjerka=„Hilda“ s identifikacijskim brojem IdJedinke = 1.

U slijedećoj tablici 4-2 prikazan je popis svih tablica baze podataka i kratkog opisa njihova značenja. Važno je napomenuti da su neke od tablica koje su sastavni dio baze podataka namijenjene budućem razvoju sustava i još uvijek se ne koriste od strane do sada razvijenih aplikacija WLO sustava [6].

Tablica 4-2: Popis tablica baze podataka

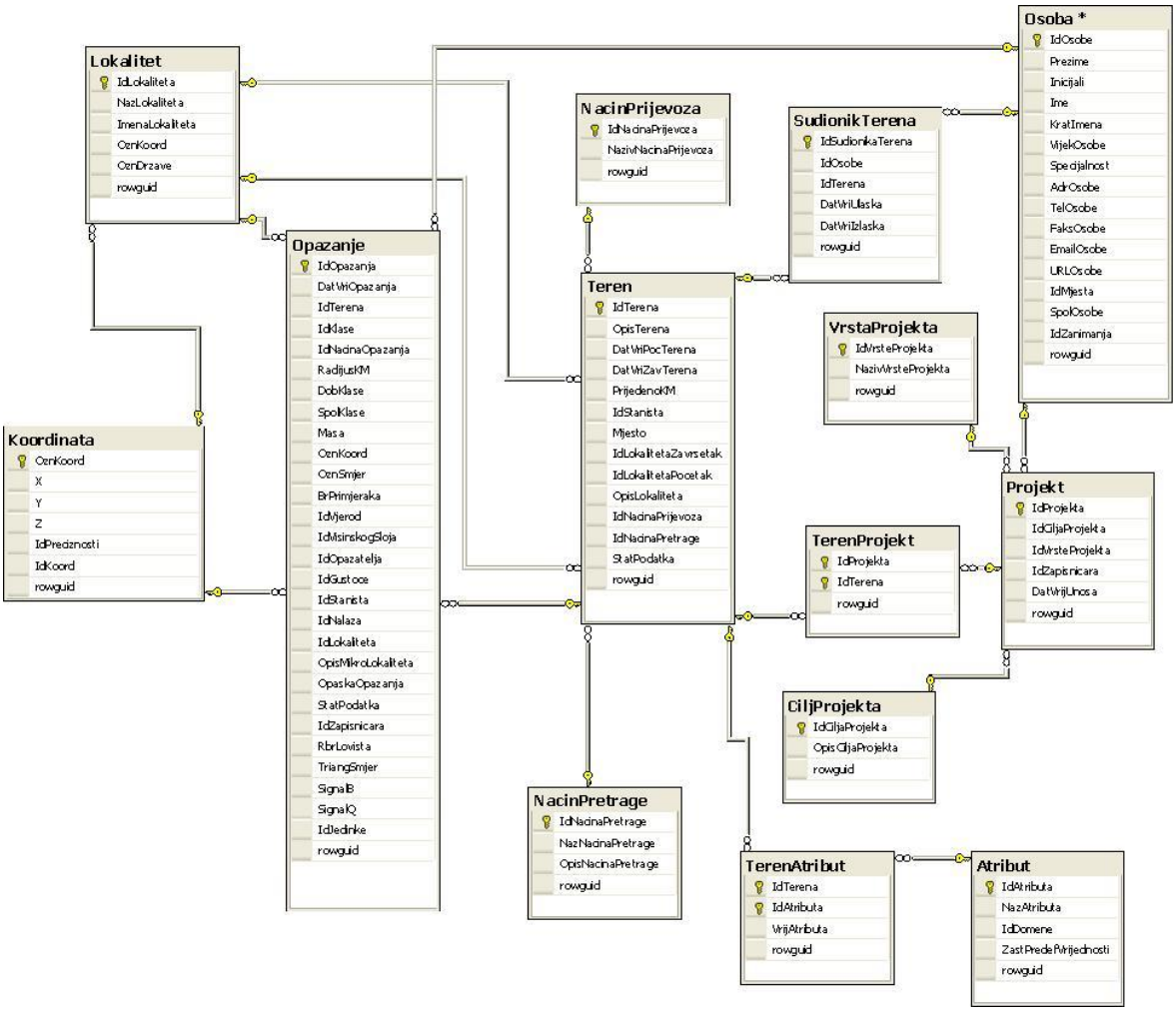
Naziv tablice	Značenje
Atribut	Korisnički definirani atributi
DioTijela	Dio tijela životinje (npr. rep)
DioTijelaUzorka	Dijelovi tijela preuzeti s nekog uzorka
Domena	Područje vrijednosti korisnički definiranog atributa
DomenaVrijednost	Vrijednosti koje pripadaju konkretnoj domeni
Drzava	Popis država
Gustoca	Gustoća pojavljivanja opažene skupine
Jedinka	Jedinka ili skupina koju možemo razlikovati od drugih (ima oznaku, ime)
Klasa	Cjelokupna klasifikacija životinjskih vrsta: carstva, rodovi, vrste ...
Koordinata	Geografska koordinata, točka
Lokalitet	Korisnički definirano mjesto ili područje
Mjesto	Popis mjesta
NacinOpazanja	Način na koji je neka jedinka opažena (npr. fotografiranjem)
NacinPretrage	Način pretrage terena (npr. monitoring)
NacinPrijevoza	Prijevozno sredstvo (npr. automobil)
Nalaz	Kategorija opažanja (npr. trag, ulov ...)
NarodnoIme	Narodno ime vrste
Odredio	Osoba koja je odredila neki uzorak
Ogrlica	Popis svih ogrlica i podataka o njima
OgrlicaJedinka	Povezuje pojedine ogrlice sa jedinkama koje ih koriste
Opazanje	Opažanje jedinke ili skupine na terenu, pohrana podataka o smjerovima za triangulaciju
OpazanjeAtribut	Dodatni korisnički atributi opažanja
Osoba	Bilo koja osoba u sustavu
OsobaKontakt	Povezanost osobe sa korisnički definiranim kontaktima
Pohrana	Način pohrane uzorka
Preciznost	Preciznost koordinate
Prikupljanje	Način uzimanja uzorka (npr. rukom)
Projekt	Aktivnost vršena na više različitih terena
ReferencaOpazanja	Ukoliko je opažanje literaturni podatak
Smjer	Smjer na kompasu
Staniste	Popis staništa po nacionalnoj klasif. staništa

SudionikTerena	Osoba (ili više njih) koja je sudjelovala u terenskom istraživanju
Teren	Terenska aktivnost (npr. praćenje tragova)
TerenAtribut	Korisnički atributi vezani uz terenske aktivnosti
TerenProjekt	Povezanost terena s određenom aktivnošću (projektom)
TriangOpazanje	Povezuje odabrane smjerove iz tablice Opazanje za izračun triangulacije sa rezultatima triangulacije pohranjenim u tablici Triangulacija
Triangulacija	Rezultati triangulacije
Uzorak	Jedinka ili dio jedinke opažen ili uzet s terena
UzorakAtribut	Korisnički atributi vezani uz određeni uzorak
VisinskiSloj	Visinski (i dubinski) sloj
Vjerodostojnost	Vjerodostojnost (pouzdanost) opažanja
Vrsta	Popis vrsta jedinki (npr. vuk – Canis Lupus)
VrstaKontakta	Korisnički definirani kontakti osobe (npr. e-mail, faks...)
VrstaProjekta	Vrsta projekta (poslovni ili privatni)
Zamka	Zamka postavljena pri obradi terena
Zupanija	Popis županija

U nastavku su izloženi različiti pogledi na pojedine podmodele koji su dio istog modela baze podataka, a predstavljeni su dijagramima koji opisuju logičku povezanost podataka. Na dijagramima su prikazani i svi atributi tablica te povezanost s ostalim tablicama modela. Objašnjeni su samo oni dijelovi razvijenog modela na temelju kojih je izgrađen podatkovni sloj aplikacije wloDesktop.

4.5.3 Podmodeli baze podataka

4.5.3.1 Terenska istraživanja



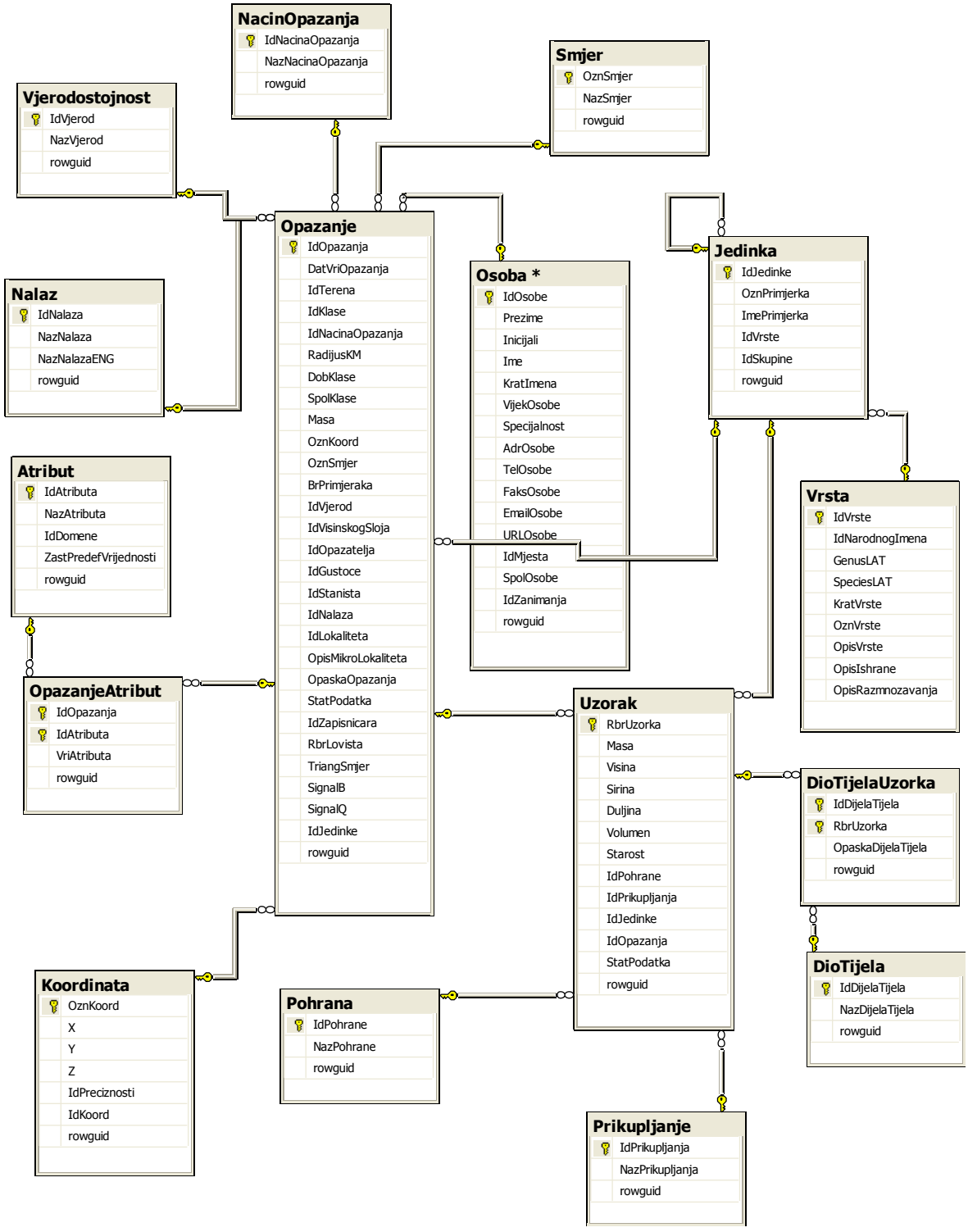
Slika 4-8: Terenska istraživanja

Prikazani podmodel baze podataka koristi se za pohranu podataka o terenskim istraživanjima i predstavlja temeljni dio modela baze podataka kojeg koristi aplikacija wloDesktop. Ovaj dio baze podataka omoguće evidenciju podataka za:

- **Projekte –** predstavljaju aktivnosti u sklopu kojih se vrše terenska istraživanja. Omogućena je podjela aktivnosti na poslovne i privatne uz pohranu podataka o glavnom cilju aktivnosti
- **Terene –** tereni na kojima se vrše istraživanja u sklopu nekog projekta. Pohranjuju se podaci o:
    - lokalitetima te datumu i vremenu ulaska i izlaska na pojedini teren
    - načinu pretrage uzoraka i vrsti korištenog prijevoznog sredstva

    - opisu terena te korisnički definiranim atributima (tablice „Atribut“ i „TerenAtribut“)
- **Opažanja –** opažanja uočena pri terenskom istraživanju
- **Sudionike terena –** osobe koje sudjeluju pri nekom terenskom istraživanju
- **Koordinate –** pohrana podataka o koordinatama lokaliteta i opažanja

4.5.3.2 Opažanja



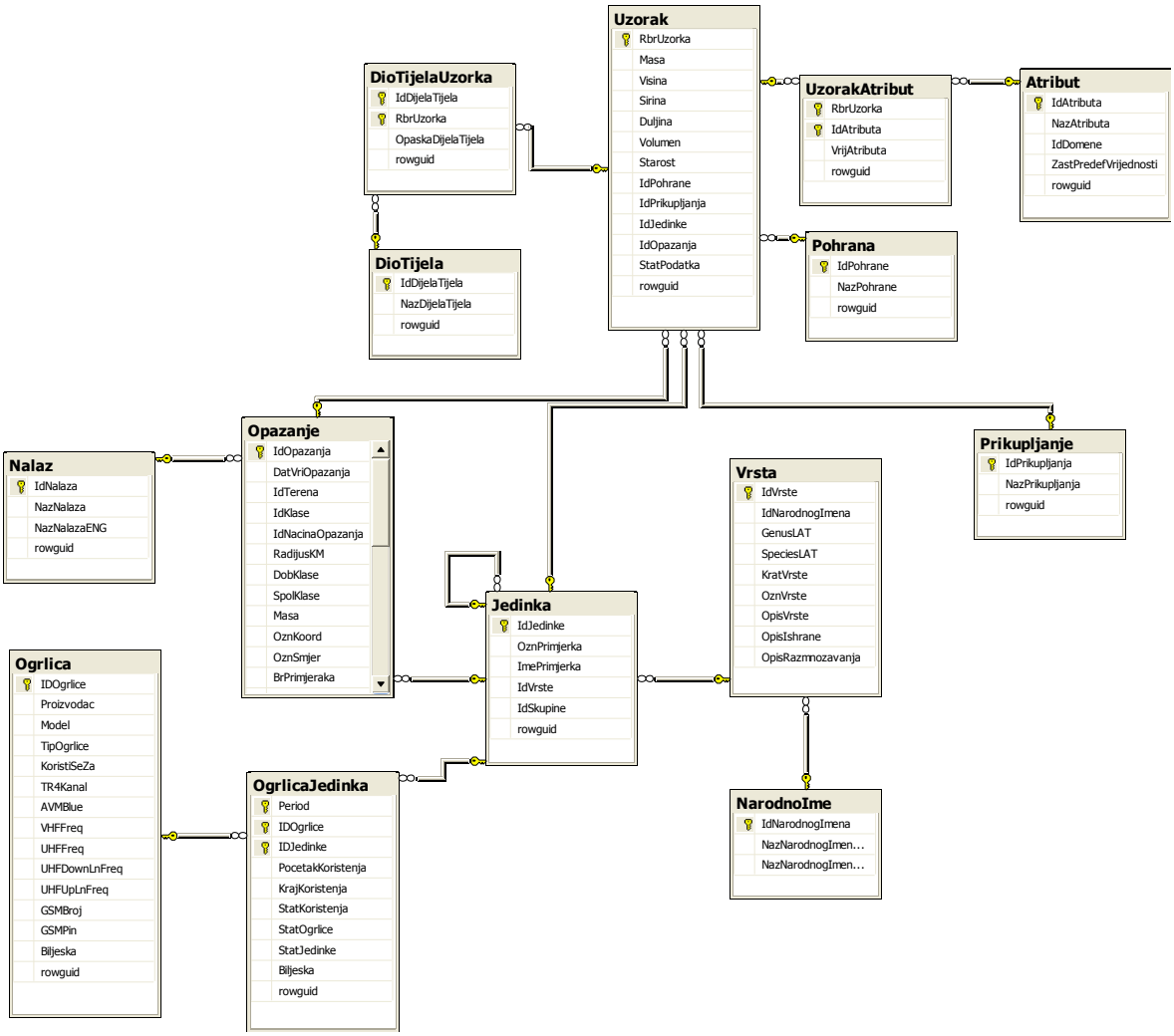
Slika 4-9: Opažanja

Prikazani podmodel baze podataka služi za pohranu podataka o opažanjima. Tablica „Opažanje“ u suštini je centralna tablica baze podataka. U ovu tablicu (i ostale tablice prikazane modelom koje su vezane na nju) pohranjuju se slijedeći podaci:

- Datum i vrijeme određenog opažanja
- Tereni uz koje su vezana opažanja
- Jedinka na koju se opažanje odnosi
- Način opažanja
- Vjerodostojnost opažanja
- Koordinata uz koju je vezano određeno opažanje
- Nalazi
- Uzorci
- Opis opažanja
- Korisnički definirani atributi opažanja (tablice „OpažanjeAtribut“ i „Atribut“)
- Osoba koja je nešto opazila (opažatelj)

Osim ovih podataka naknadno je dodana mogućnost unosa podataka o smjerovima i signalima koji se koriste pri algoritmu triangulacije, a objašnjeni su na podmodelu „Radiotelemetrija“.

4.5.3.3 Primjerak (jedinka)

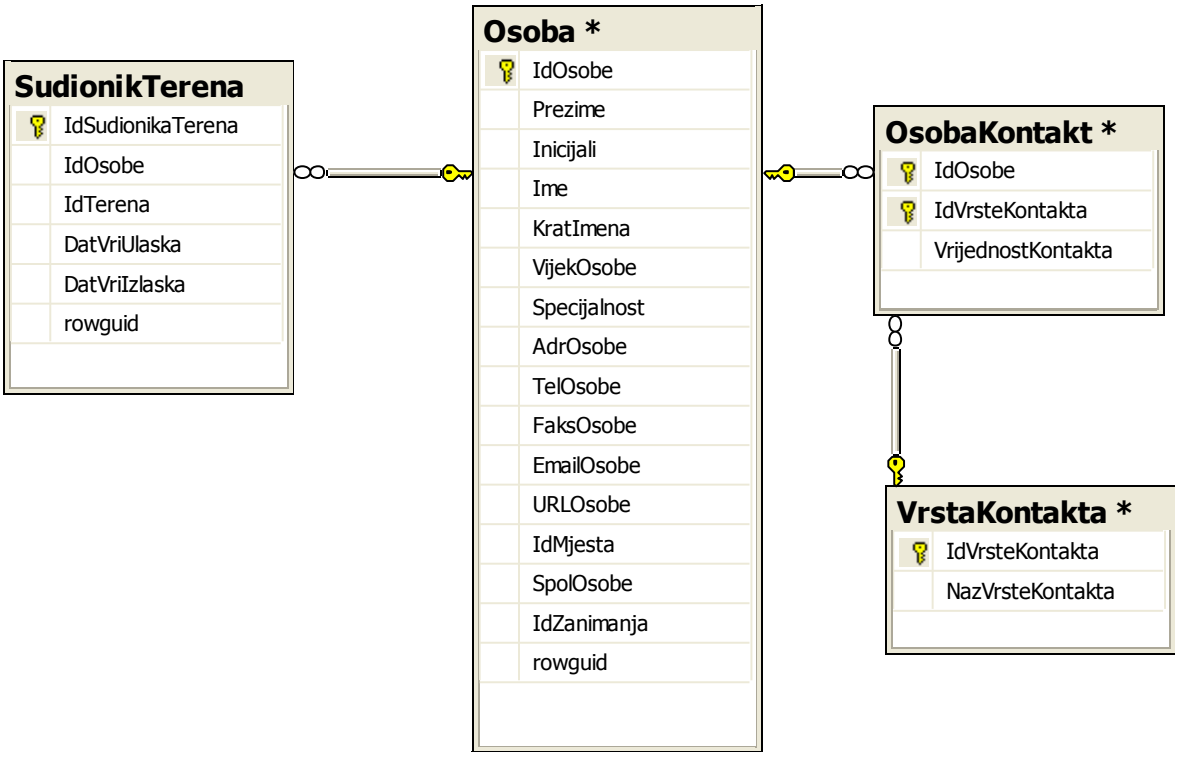


Slika 4-10: Primjerak (Jedinka)

Dio modela prikazan na slici 4-11 omogućuje pohranu podataka o pojedinim jedinkama (životinjama). Osim toga, prikazana je i povezanost tablice „Jedinka“ s ostalim tablicama modela, a podaci koji se pohranjuju su:

- Jedinka – naziv jedinke i pripadnost jedinke odgovarajućoj vrsti
- Narodno ime vrste
- Podaci o ogrlicama i povezanost s jedinkama koje ih nose
- Opažanja, uzorci i nalazi vezani uz konkretnu jedinku

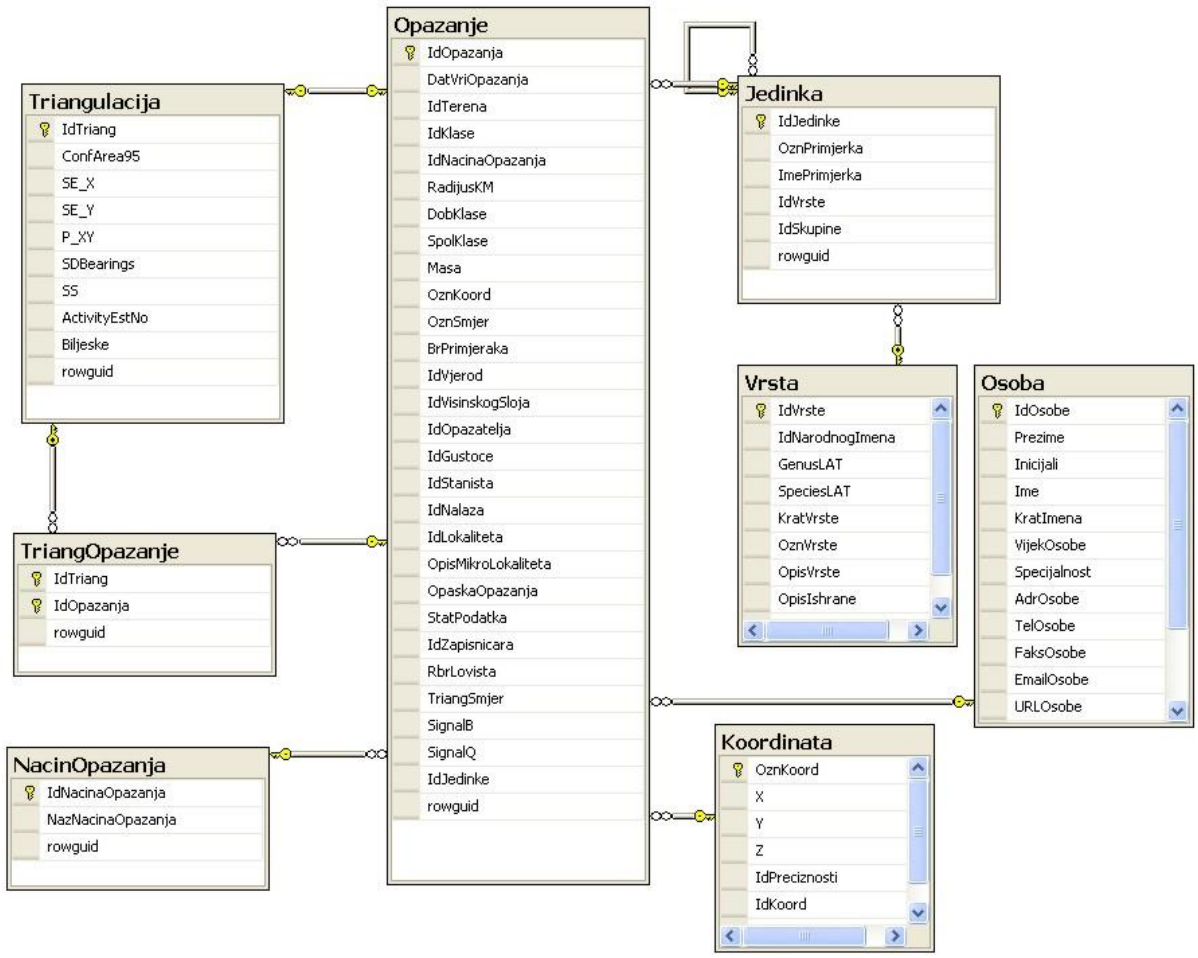
4.5.3.4 Osobe (kontakti) i sudionici terena



Slika 4-11: Osobe (kontakti)

Prikazani dio modela na slici 4-12 omoguće pohranu podataka o svim sudionicima terenskog istraživanja (ali i o drugim osobama). Tablica „Osoba“ sadrži atribute za pohranu osnovnih podataka o pojedinoj osobi, a korisniku je omogućeno definiranje i dodatnih atributa (kontakata) – tablice „OsobaKontakt“ i „Kontakt“. Sudjelovanje pojedine osobe u nekom terenskom istraživanju evidentira se upisivanjem odgovarajućeg zapisa u tablicu „SudionikTerena“.

4.5.3.5 Radiotelemetrija



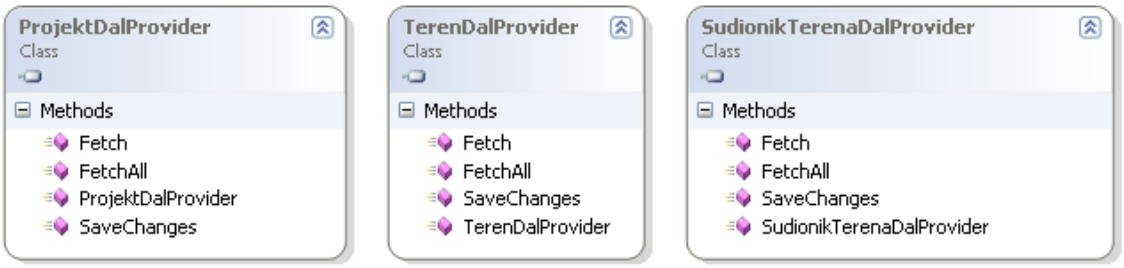
Slika 4-12: Radiotelemetrija

Prikazani dio modela na slici 4-12 omogućuje pohranu podataka vezanih uz radiotelemetriju, odnosno podatke dobivene korištenjem ugrađenog algoritma triangulacije. Pojedina mjerenja signala unose se u tablicu „Opažanje“ uz referenciranje odgovarajućeg zapisa iz tablice „NacinOpazanja“ (zapis „radiotelemetrija“) i pohranu koordinate lokacije s koje se vrši mjerenje u tablicu „Koordinata“. Rezultat algoritma (lokacija jedinke) također se upisuje u tablicu „Opažanje„ uz referenciranje drugog zapisa iz tablice „NacinOpazanja„ (zapis „Algoritam triangulacije“) i pohranu koordinate dobivene lokacije u tablicu „Koordinata“. Ostali parametri algoritma triangulacije pohranjuju se u tablicu „Triangulacija“. Tablica „TriangOpazanje“ povezuje dobivenu lokaciju jedinke s mjerenjima koja su korištena pri izračunu.

4.5.4 Podatkovni sloj

Podatkovni sloj aplikacije realiziran je SQL bazom podataka (opisanom u poglavlju 4.5.2) i implementacijom podsloja pristupa podacima (DAL). U sklopu baze podataka za svaki pojedini entitet (teren, projekt...) ugrađene su pohranjene procedure koje se koriste za manipulaciju nad podacima pohranjenim u bazi (unos, izmjena, brisanje i dohvat).

Opći opis podatkovne komponente (podsloja DAL) za entitete „Teren“, „SudionikTerena“ i „Projekt“ prikazan je na slici 4-13.

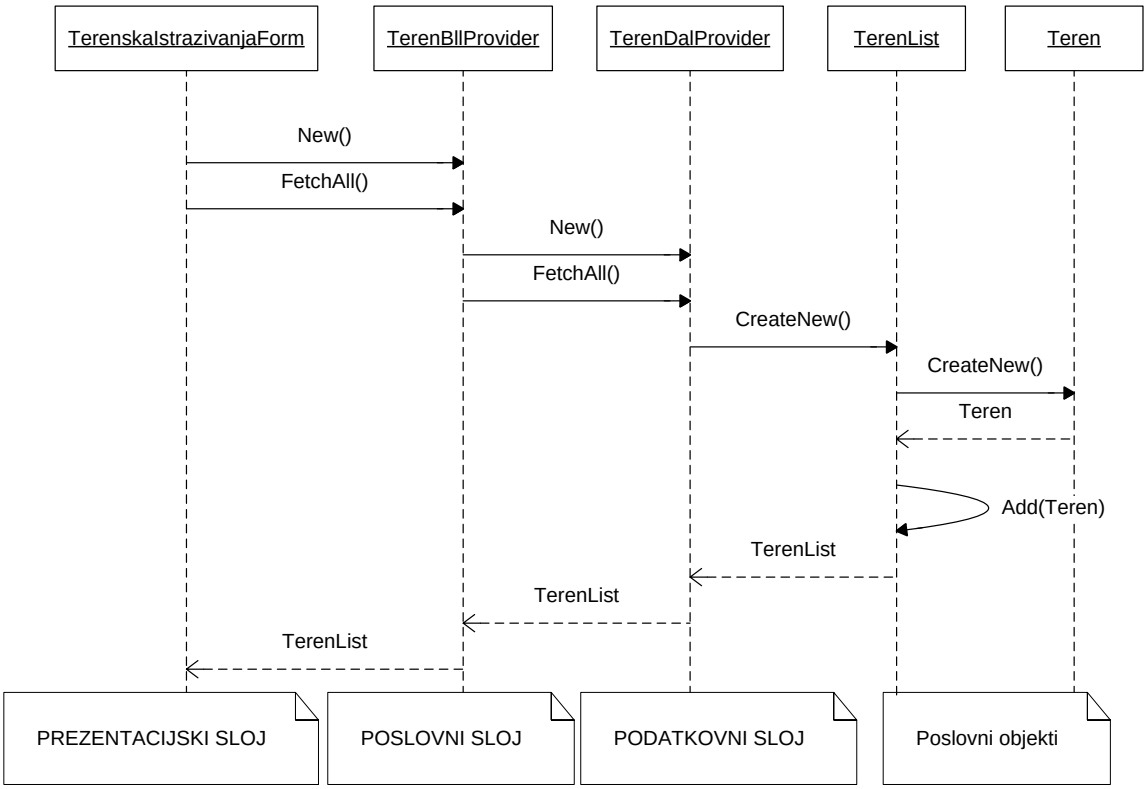


Slika 4-13: Opći opis podatkovne komponente (DalProvider klase)

Podsloj pristupa podacima realiziran je nizom klasa „EntitetDalProvider“ koje koje imaju definirane motode za izravan pristup podacima iz baze podataka pozivanjem prethodno opisanih pohranjenih procedura. Slijedi primjer metode „FetchAll“ klase „TerenDalProvider“ koja omogućuje dohvat svih zapisa iz tablice „Teren“:

```
public TerenList FetchAll() {
      using (SqlConnection dataConnection = new SqlConnection
      (ConfigurationManager.ConnectionStrings["wloDesktop.Properties.Settings.dbWLOC
      onnectionString1"].ConnectionString))
            {
                using (SqlCommand fetchCommand = dataConnection.CreateCommand())
                {
                   fetchCommand.CommandText = "[dbo].[ap_TerenList_R]";
                   fetchCommand.CommandType = CommandType.StoredProcedure;
                   dataConnection.Open();
                    using (SqlDataReader terenPodaci = fetchCommand.ExecuteReader())
                    {
                         return TerenList.CreateNew(terenPodaci);
                    }
                }
            }
```

Na slici 4-14 prikazan je dijagram slijeda koji pokazuje jednostavan dohvat liste poslovnih objekata „Teren“. Na dijagramu se vidi i komunikacija podatkovnog sloja s ostalim slojevima aplikacije.

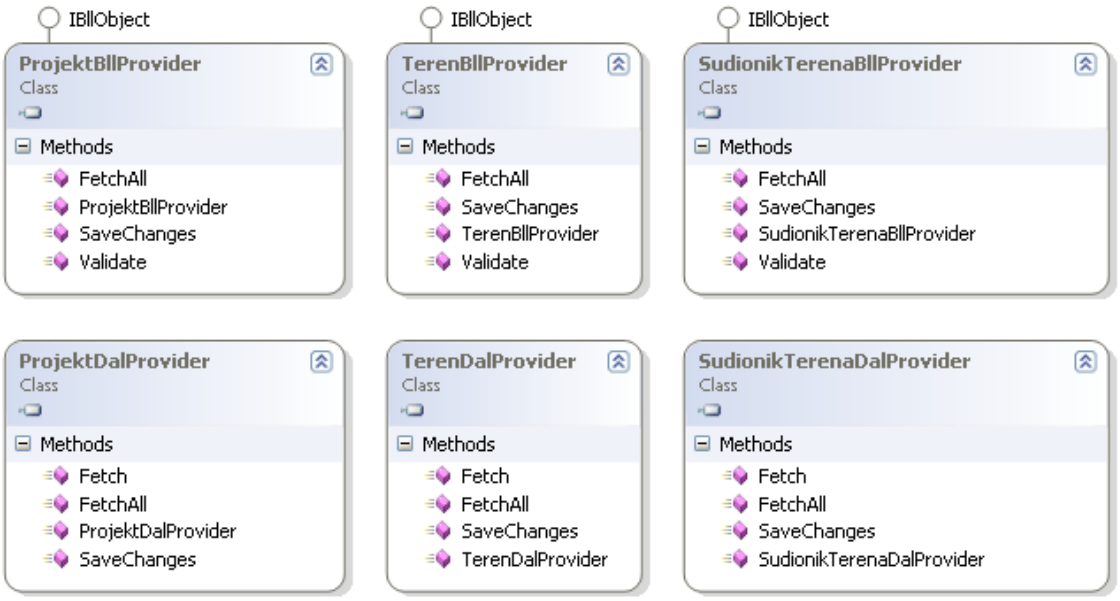


Slika 4-14: Dohvat liste poslovnih objekata „Teren“

4.5.5 Poslovni sloj

Poslovni sloj aplikacije realiziran je kao međusloj između dohvata podataka i njihove prezentacije, a pri tome implementira cjelokupnu logiku aplikacije. Osnovna funkcija koju obavlja ovaj sloj je dohvat podataka iz podatkovnog sloja (komunikacijom s DAL objektima), filtriranje i obrada dohvaćenih podataka prema različitim kriterijima i naposljetku isporuka obrađenih podataka prezentacijskom sloju. Osim toga, poslovni sloj posreduje i pri izmjeni ili unosu novih podataka na način da obavlja validaciju podataka prije njihove konačne pohrane u bazu.

Na slici 4-15 dan je opći opis komponente poslovne logike za entitete „Teren“, „SudionikTerena“ i „Projekt“.

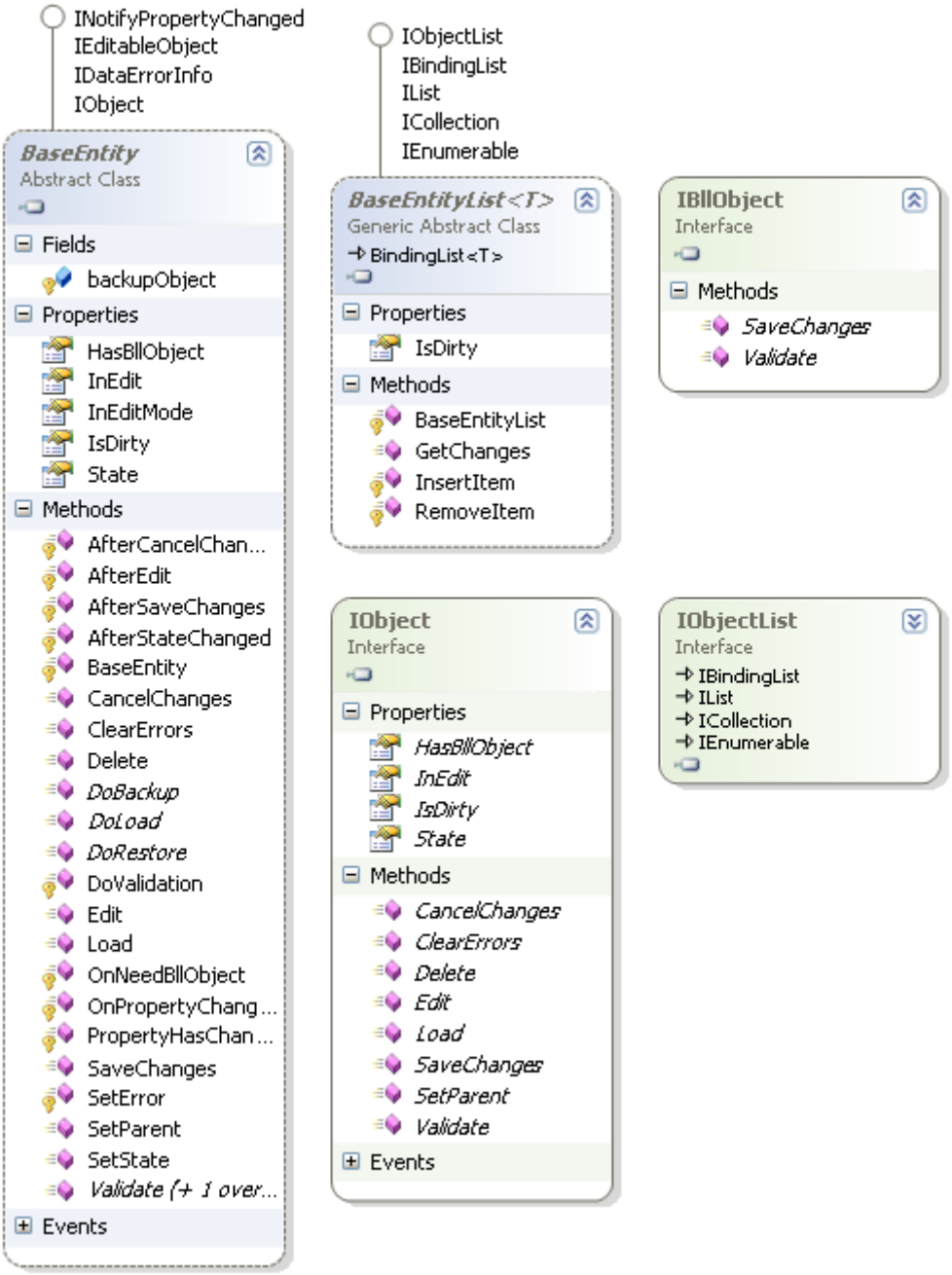


Slika 4-15: opći opis komponente poslovne logike

Osnovne funkcije koje obavljaju svi pojedini BLL objekti (EnitetBllProvider) su:

- dohvat i pohrana podataka o entitetu preko podatkovnog sloja
- validacija podataka
- filtriranje podataka
- sortiranje podataka

U nastavku, prikaz sučelja komponente poslovne logike prema ostalim dijelovima sustava također je prikazan slikom (slika 4-16).



Slika 4-16: Opis sučelja komponente poslovne logike

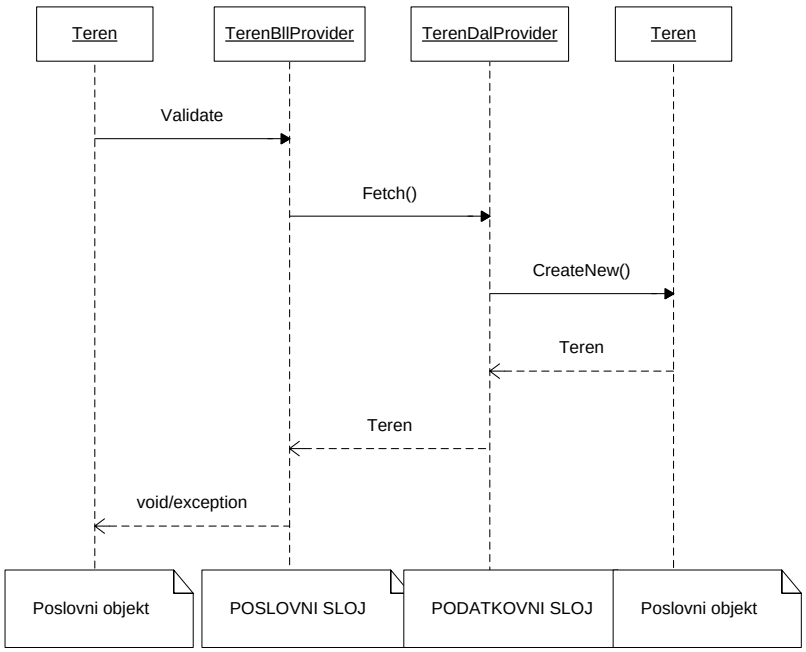
Na početku smo spomenuli da poslovni sloj predstavlja poveznicu između pohranjenih podataka i njihove prezentacije. Poslovni sloj pruža sučelje sloju prezentacije preko kojeg mu predaje potrebne podatke. Akcije koje ovaj sloj obavlja i koje su preko spomenutog sučelja dostupne prezentacijskom sloju mogu se odvijati u suradnji s podatkovnim slojem (primjerice, dohvat podataka) ili se mogu obavljati isključivo unutar poslovnog sloja (poput filtriranja podataka). Slijedi primjer metode koja se izvršava isključivo unutar poslovnog sloja, a služi za filtriranje liste zapisa iz tablice

„SudionikTerena“ prema terenu na kojem su vršili istraživanje:

```
public SudionikTerenaList FilterByTeren(SudionikTerenaList sudionikTerenaAll, int
IdTerena)
        {
            if (sudionikTerenaAll != null && IdTerena != null)
            {
                SudionikTerenaList filteredList = new SudionikTerenaList();
                foreach (SudionikTerena sudionikTerenaItem in sudionikTerenaAll)
                {
                    if (sudionikTerenaItem.IdTerena == IdTerena)
                    {
                        filteredList.Add(sudionikTerenaItem);
                    }
                }
                return filteredList;
            }
            else
            {
                return new SudionikTerenaList();
            }

        }
```

Radi boljeg shvaćanje uloge poslovnog sloja i njegove komunikacije s ostalim slojevima aplikacije, na slici 4-17 prikazan je dijagram slijeda kojim je opisan postupak validacije primarnog ključa pri promjeni svojstva poslovnog objekta „Teren“.

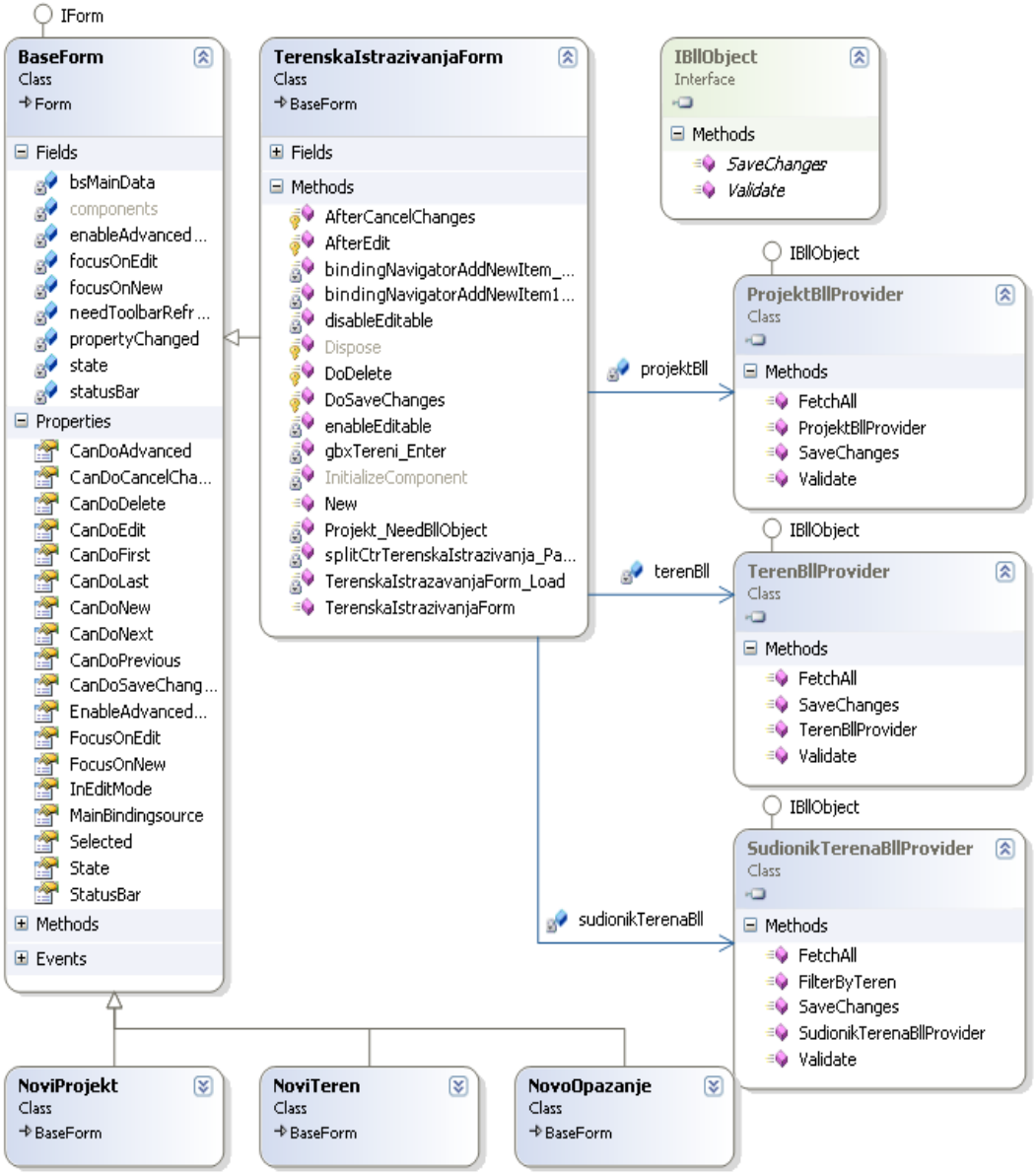


Slika 4-17: Validacija primarnog ključa entiteta „Teren“

4.5.6 Prezentacijski sloj

Prezentacijski sloj aplikacije „wloDesktop“ predstavlja grafičko korisničko sučelje (engl. *Graphical User Interface*). Preko korisničkog sučelja korisnik pristupa sloju poslovne logike preko kojeg dohvaća podatke iz podatkovnog sloja. Osnovni zadatak prezentacijskog sloja je da korisniku prikazuje različite zaslonske maske za brzu i efikasnu manipulaciju podacima pohranjenim u bazi podataka.

Opći opis prezentacijske komponente aplikacije „wloDesktop“ prikazan je na slici 4-18.



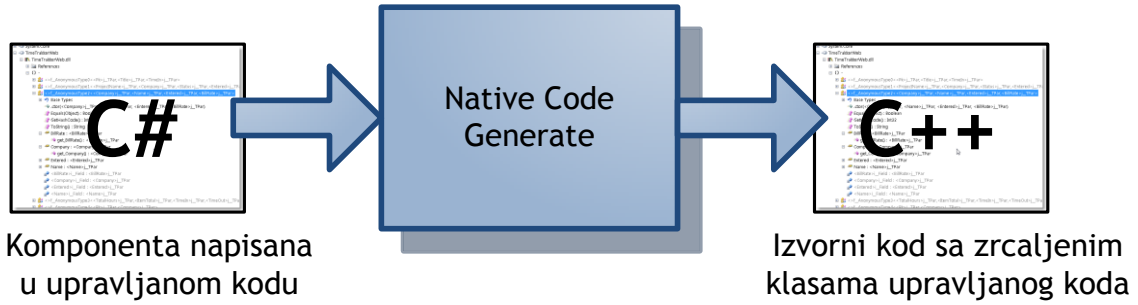
Slika 4-18: Opći opis prezentacijske komponente

Detaljno objašnjenje prezentacijskog sloja aplikacije, odnosno grafičkog korisničkog sučelja izloženo je u poglavlju 5.2.

4.6 Arhitektura izvornog i upravljanog koda na džepnom računalu

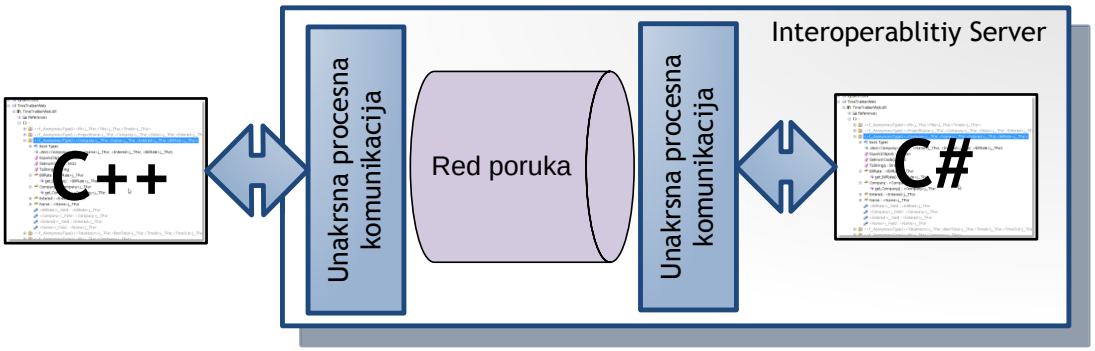
Mirror je paket koji omogućuje poziv *upravljanog koda* (engl. managed code) iz izvornog (engl. native) procesa na džepnom računalu. Službeno ne postoji **interoperabilnost** između komponenti napisanih u izvornom i upravljanom kodu i ako ne bi koristili pomoćni alat kao što je Mirror ne bi mogli ostvariti vezu između njih. *Mirror* se sastoji od dva programa: *Native Code Generate* i *Interoperability Server*.[3]

Native Code Generate je program koji kao ulaz prima asemblij (engl. assembly) upravljanog koda i generira na izlazu C++ kod koji će sadržavati zrcaljena sučelja klase napisane u upravljanom kodu . Time nam je omogućeno da izvorni proces ne zove metode u upravljanom kodu jer je to nemoguće već će zvati metode u zrcaljenom sučelju (Fuller, 2005).



Slika 4-19: Proces rada analizatora izvornog koda

Interoperability Server služi kao domaćin (engl. host) za komponentu napisanu u upravljanom kodu. Interoperability Server služi kao most između izvornog koda kojeg je generirao Native Code Generate i upravljanog koda u komponenti. Komunikacija se ostvaruje uz pomoć reda poruka (engl. message queue) i *poprečne procesne komunikacije* (engl. cross-process communication). Za poprečnu procesnu komunikaciju se koristi mehanizam koji je sadržan u Windows Mobile operacijskom sustavu. [3]



Slika 4-20: Izgled komunikacije između izvornog i upravljanog koda

Komponentu koju pišemo upravljanim kodom i koju želimo pozivati iz izvornog procesa mora sadržavati neka ograničenja za razliku od komponente koja ne bi trebala biti pozivana iz izvornog procesa. Mogu se koristiti konstruktori, metode, svojstva i indekseri, a kao tipovi podataka mogu se koristiti svi primitivni tipovi podataka kao i znakovni nizovi i enumeratori. Ako je potrebno mogu se prebacivati i instance klasa napisanih u upravljanom kodu.

Postupak kreiranja interoperabilnosti između izvornog i upravljanog koda počinje pisanjem komponente u upravljanom kodu. Dobar savjet je da se pokuša kreirati klase koje će biti podijeljene po načelu jedinstvene odgovornosti sa što manjim međuovisnostima. Razlog je u izgradnji dualnih klasa koje gradi parser. Analizator se nalazi u datoteci NativeCodeGenerate.exe, a pokreće se iz komadne linije. Najbolje je pokrenuti „command prompt“ i pozicionirati se u direktorij gdje se nalazi analizator. Komponente napisane u upravljanom kodu potrebno je staviti u direktorij sa analizatorom kako bi smanjili ulaznu naredbu kod pokretanja analizatora i tako si olakšali analiziranje komponente. Kao izlaz dobivaju se dualne klase napisane u C++ izvornom kodu koje je potrebno koristiti u komponenti napisanoj u izvornom kodu, preporuča sa razvijati *Active Template Library* (skraćeno ATL) komponentu.

Naredba za pokretanje NativeCodeGenerate.exe datoteke u kojoj želimo analizirati komponentu koja je razvijana u završnom radu glasi:

```
NativeCodeGenerate.exe /queueName:wloCE /directory:C:\BUILD
/assembly: C:\BUILD\wloCE.dll wloCE.data wloCE.callerFormAddBearing
wloCE.callerFormBearingList wloCE.callerFormPoint
wloCE.callerFormPointList
```

4.7 Replikacija podataka između poslužitelja i mobilnog klijenta

4.7.1 Pojam replikacije

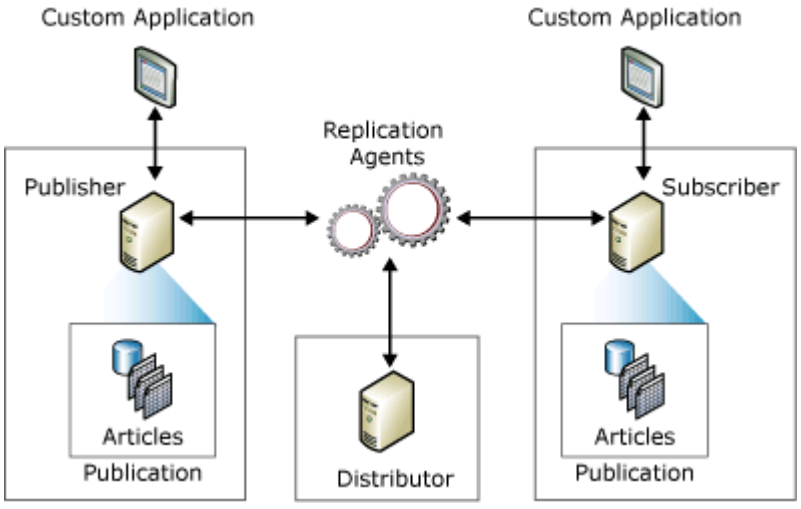
Replikaciju definiramo kao skup tehnologija koje omogućuju držanje instanci istih podataka na više različitih lokacija.

Replikacija koristi publiciraj-pretplati model za distribuiranje podataka:

- **Izdavač** je poslužitelj koji predstavlja izvor svih podataka koji sudjeluju u samom procesu replikacije. Izdavač definira članke za svaku tablicu ili neki drugi objekt baze podataka koji će biti korišteni kao izvor replikacije. Jedan ili više povezanih članaka iz iste baze podataka su organizirane u publikaciju. Publikacije su najbolji način da se grupiraju istovrsni podaci i objekti koje se žele replicirati zajedno.

- **Pretplatnik** je poslužitelj koji prima podatke replicirane od strane Izdavača. Pretplatnik definira pretplatu na određenu publikaciju. Pretplata specificira kada Pretplatnik prima publikaciju od Izdavača i mapira članke u tablice i ostale objekte baze podataka u Pretplatniku.

- **Distributer** je poslužitelj koji izvršava različite zadatke kada prebacuje članke od Izdavača do Pretplatnika. Stvarni zadaci koji su izvršeni ovise o vrsti izvršene replikacije.



Slika 4-21: Arhitektura replikacije [13]

Postoje tri vrste modela replikacije koji se razlikuju po samoj arhitekturi pojedinog modela:

- **Spojna replikacija** *(engl. Merge Replication)* – Vrsta replikacije koja pruža mogućnost i Pretplatnicima da mijenjaju izvorne podatke koji su replicirani sa Izdavača. Sama sinkronizacija između Pretplatnika i Izdavača se odvija preko snimke stanja. Unutar samog modela ove vrste replikacije, karakteristično je to što su Izdavač i Distributer isti poslužitelj. Najveći problem kod replikacije je očuvanje konzistentnosti podataka budući da više Pretplatnika može mijenjati izvorne podatke. Spojna replikacija posjeduje unutar sebe ugrađene mehanizme koji omogućuju automatsko rješavanje konflikata, no postoji mogućnost i samostalnih pravila za njihovo rješavanje.

- **Replikacija snimke stanja** (*engl. Snapshot Replication*) – Kod ove vrste replikacije karakteristična je snimka stanja. Snimka stanja predstavlja trenutno stanje podataka na Izdavaču. Svi pretplatnici koji repliciraju podatke od Izdavača posjeduju samo kopije podataka čija je vrijednost jednaka vrijednostima podataka na Izdavaču u trenutku stvaranja replikacije. Na temelju snimke stanja, koja se obično stvara u periodičnim vremenskim trenucima, stvaraju se sve publikacije od strane Izdavača. Budući da je ova vrsta zbog ovakvog načina korištenja dosta statična, koristi se samo u slučajevima kada je potrebno prenositi samo manju količinu podataka. Za razliku od spojne replikacije, nije moguće da Pretplatnici mijenjaju izvorne podatke replicirane od Izdavača.

- **Transakcijska replikacija** (*engl. Transactional Replication*) – Za razliku od replikacije snimke stanja koja predstavlja statičnu vrstu replikacije, kod transakcijske replikacije prvo dolazi do sinkronizacije između Pretplatnika i Izdavača, a zatim, kako se podaci mijenjaju naIzdavaču i vrše transakcije, promijenjeni podaci se repliciraju na Pretplatnike. Ova vrsta replikacije koristi se u slučajevima kada je pretplatnicima bitno dobiti podatke u trenucima čim se promjena na podacima dogodi. [12]

4.7.2 Spojna replikacija

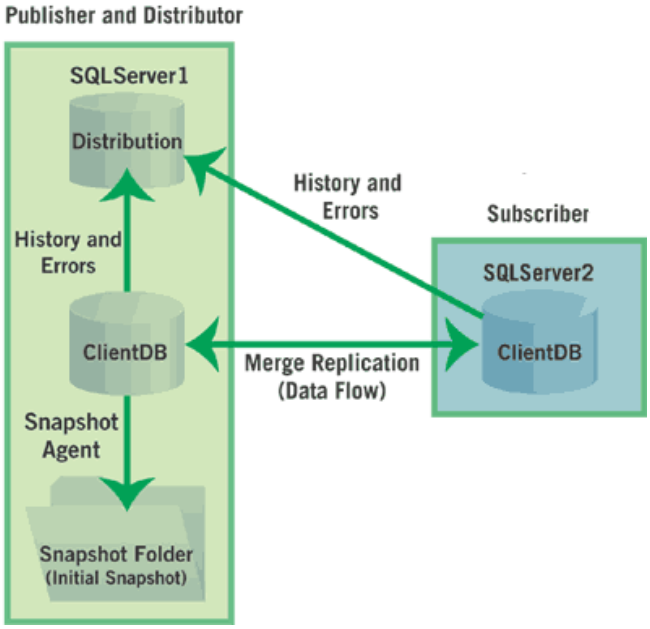
Posebnost spojne replikacije je u tome što su i Distributer i Izdavač jedan entitet. Sam proces spojne replikacije označava prijenos podataka od Izdavača prema Pretplatnicima, ali na takav način da i Pretplatnici mogu mijenjati te podatke i dodavati nove te na kraju rada uspješno izvršiti proces sinkronizacije. Spojna replikacija uključuje već ugrađene mehanizme za rješavanje konflikata koji mogu doći pri procesu sinkronizacije.

Upravo zbog gore navedenih osobina ove vrste replikacije odabrana je spojna replikacija za implementaciju u praktičnom dijelu projekta budući da je neophodno da promatrači divljih životinja mogu modificirati replicirane podatke prilikom terenskog rada. Bitno je omogućiti unos i izmjenu podataka neovisno o spajanju na Izdavača, a tek naknadnim spajanjem da se izvrši proces sinkronizacije.

Proces sinkronizacije je omogućen na zahtjev, za razliku od drugog tipa replikacije, sam Pretplatnik može u bilo kojem trenutku tražiti sinkronizaciju sa Izdavačem te nije ovisan o unaprijed definiranim vremenskim periodima sinkroniziranja.

Spojna replikacija podržava filtriranje podataka koji se repliciraju. Budući da se replikacija često koristi na mobilnim uređajima i za implementaciju terenskog rada, potrebno je prilikom projektiranja sustava voditi računa o resursima sa kojima korisnik raspolaže. Prilikom komunikacije mobilnog uređaja i centralnog sustava cilj je smanjiti mrežni promet prilikom sinkronizacije podataka te je u mehanizam replikacije ugrađena mogućnost filtriranja podataka kako bi se prenosili samo oni podaci koji su uistinu potrebni pretplatniku za njegov rad. Spojna replikacija omogućuje filtriranje i stupaca i redaka unutar neke tablice koja je označena kao objekt nad kojim se vrši repliciranje podataka.

Korisnik može definirati, ukoliko mu je to potrebno pri izvršavanju procesa replikacije, logiku koja će se izvršiti pri sinkroniziranju podataka ukoliko mu sam postupak sinkroniziranja nije dovoljan. Sam kod koji korisnik definira ima pristup svim podacima koji se sinkroniziraju te uglavnom reagira na okidače koji su uzrokovani modifikacijom podataka ili samim procesom sinkronizacije.



Slika 4-22: Arhitektura spojne replikacije [13]

Spojna replikacija je implementirana sa Agentom za snimku stanja (engl. Snapshot Agent) i Spojnim Agentom (engl. Merge Agent). Agent za snimku stanja pripremi datoteke snimke stanja, dok Spojni Agent aplicira poslove inicijalne snimke stanja sadržane u tablicama publicirane baze podataka na Pretplatnika.

Prije nego Pretplatnik može primiti podatke od Distributera, mora podržavati podatke sa istom shemom tablica kao i Distributer. Proces koji kopira cijelu trenutnu publikaciju od Izdavača prema Pretplatniku naziva se inicijalni odsječak. Kada se odsječci prenesu Pretplatnicima, utječu samo na one Pretplatnike koji očekuju te odsječke. Spojni Agent tada podrazumijeva da su Pretplatnik i Izdavač sinkronizirani i trenutno kreće slati nova umetanja, brisanja i modificiranje objavljenih podataka. [11]

Kada je red promijenjen u spojnom članku, ugrađeni okidač promijeni generirani stupac tog retka na vrijednost 0. Kada se Spojni Agent izvodi, on prikuplja sve retke sa vrijednošću generiranog stupca 0 u jednu ili više grupa i dodjeljuje im posebno generirane vrijednosti u ovisnosti sa prethodnim vrijednostima. Spojni agent na svakoj strani (mobilna i centralna) vodi brigu o vrijednostima podataka koje se međusobno šalju. Za vrijeme sinkronizacije Spojni agent šalje sve one podatke koji su modificirani tokom prethodnog rada. Blokovi promijenjenih podataka idu od strane na kojoj su podaci izmijenjeni prema kopiji podataka koja treba biti osvježena sa tim novim podacima.

Na odredišnoj bazi podataka, pristigle vrijednosti se spajaju sa starim vrijednostima

prema unaprijed definiranim pravilima replikacije. Spojni agent uspoređuje prethodne i izmijenjene podatke te se na pojavu bilo kojeg konflikta poziva rješavanje istog. Pri tome se koristi prioritetno rangiranje. Alternativno, mogu se definirati vlastiti načini rješavanja konflikata na članku koji se replicira kroz pohranjene procedure. Podaci se promijene na drugim mjestima tek u slučaju da se pozove proces sinkronizacije. Sam proces sinkronizacije može trajati nekoliko minuta,dana pa čak i tjedana, ovisno o količini i kompleksnosti podataka koji se repliciraju. Na kraju procesa sve strane završe sa ujednačenim podacima.
Autonomija pojedine strane koja vrši repliciranje je unutar spojne replikacije dosta velika. Sve strane mogu vršiti unos podataka, brisanje ili modificiranje već postojećih strana, neovisno o operacijama koje se vrše na drugim poslužiteljima. No spojna replikacija ne garantira transakcijski integritet nego promovira podatkovnu konvergenciju. Sve promjene koje se dogode na različitim stranama u konačnici se svedu na istu vrijednost premda ta vrijednost nije ona garantirana koja bi bila posljedica toga da su sve promjene bile obavljene samo na jednoj strani. [11]

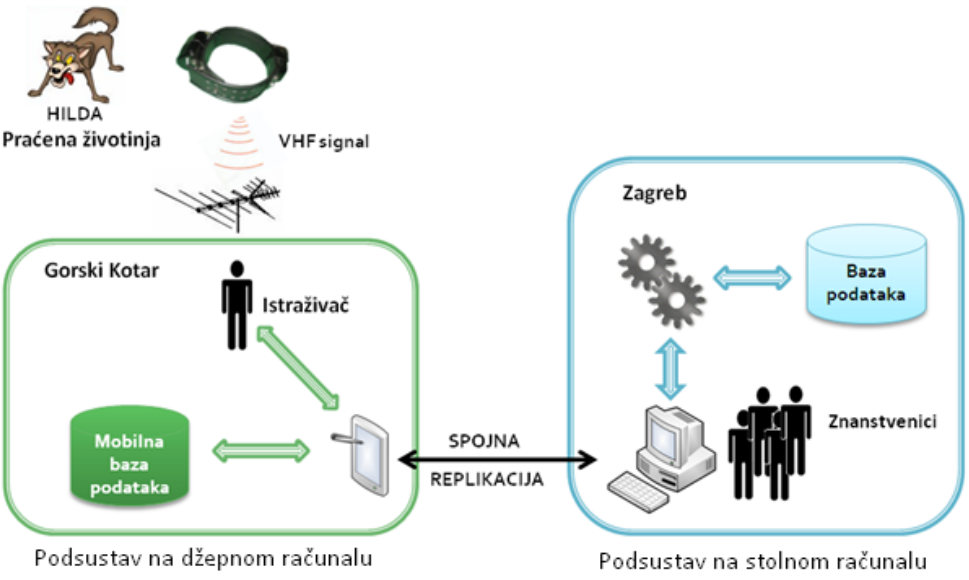
Neke od glavnih ugrađenih komponenti koje se koriste tijekom procesa replikacije:

- SQL Server Mobile Database Engine – kontrolira lokalnu bazu
- SQL Server Agent – pokreće replikacijske agente i prati operacije izvan same replikacije
- Snapshot Agent – priprema početne podatke, sprema zapise, prati informacije o sinkronizaciji
- Merge Agent – priprema početnu sliku na Pretplatniku, sinkronizira svaku promjenu
- SQL Server Mobile Client Agent – primarna komponenta, implementira objektno sučelje
- SQL Server Mobile Server Agent – kontrolira komunikaciju između poslužitelja I mobilnog uređaja (HTTP zahtjevi)
- SQL Server Reconciler
- SQL Server Replication Provider [12]

5. Rezultati

5.1. Pregled cjelokupnog sustava

WildLife Observer (WLO) je programski sustav namijenjen praćenju životinja na moderan način korištenjem džepnog računala, stolnog računala i algoritama na osnovu kojih se određuje položaj životinje u prostoru. Sustav omogućuje evidenciju svih relevantnih podataka vezanih uz terenska istraživanja divljih životinja. Podaci su korisniku prikazani u tabličnom obliku, ali i grafički na karti koja je dio GIS sustava. Između centralnog mjesta za pohranu podataka i mobilnih uređaja uspostavljen je mehanizam spojne replikacije koji omogućuje efikasnu sinkronizaciju podataka između svih radnih stanica. WLO sustav oslanja se na VHF radio ogrlice (na životinji), te antene za mjerenje radio signala. Nakon izvršenih mjerenja, korisnik može jednostavno locirati životinju korištenjem implementiranog matematičkog algoritma *triangulacije* koji je pokrenut na podsustavu za džepno računalo. Ovaj algoritam na osnovu izvršenih mjerenja određuje lokaciju životinje, a sam rezultat se grafički prikazuje na karti. Također putem podsustava za džepno računalo se mogu upsisati komentari i opažanja. Više istraživača na terenu (npr. Gorski Kotar) prikupljaju podatke i u svakom trenutku kada imaju pristup Internetu mogu spojnom replikacijom sinkronizirati podatke sa centralnom bazom podataka koja se nalazi u znanstvenom laboratoriju gdje znanstvenici na podsustavu za stlono računalo vrše analize i donose zaključke o populacijama životinja, i koriste podatke u znanstvene svrhe kao i svrhe zaštite populacije.



Slika 5-1: Konceptualni model sustava

5.2. Sustav na stolnom računalu

U poglavlju 4.5 detaljno smo objasnili arhitekturu troslojne aplikacije wloDesktop. U ovom poglavlju opisano je korisničko sučelje aplikacije i način njegova korištenja.

5.2.1. Pokretanje aplikacije – početna zaslonska maska

Prilikom pokretanja aplikacije, korisniku se otvara početna zaslonska maska prikazana na slici 5-2. Ovo je osnovna, roditeljska zaslonska maska unutar koje su sadržane sve ostale zaslonske maske koje korisnik može pozvati.

Slika 5-2: Osnovna (roditeljska) zaslonska maska aplikacije „wloDesktop“

Na slici su brojevima označena dva osnovna dijela roditeljske zaslonske maske:

1. **Glavni izbornik –** omogućuje brzu navigaciju kroz različite mogućnosti aplikacije
   - **Program –** pokretanje akcija koje se odnose na cjelokupnu aplikaciju (primjerice izlaz iz programa)

- **Podaci –** pokretanje različitih formi za unos i pregled podataka
    - *Terenska istraživanja* – poziv glavne zaslonske maske za rad s podacima o terenskim istraživanjima
    - *Novi unos* – pozivi zaslonskih maski za unos novih zapisa u bazu podataka
- **Šifrarnici –** pokretanje generičke forme za ažuriranje podataka šifrarnika
- **Prozori –** popis trenutno otvorenih zaslonskih maski uz različite mogućnosti njihovog razmještaja unutar glavne zaslonske maske (kaskadno, horizontalno i vertikalno poravnanje)
- **Pomoć –** sadrži stavku za poziv zaslonske maske za prikaz informacija o programu

2. **Statusna traka –** prikaz statusa obrade podataka (unos, izmjena, brisanje)

Za početak rada s podacima potrebno je pokrenuti jedan od dva raspoloživa modula aplikacije:

1. **Terenska istraživanja –** modul za rad s podacima vezanim uz terenska istraživanja
2. **Triangulacija –** modul za rad s telemetrijskim podacima – algoritam triangulacije i prikaz geokodiranih informacija na karti

Odgovarajući modul moguće je pokrenuti odabirom odgovarajuće stavke iz izbornika „Podaci“ prikazanim na slici 5-3.

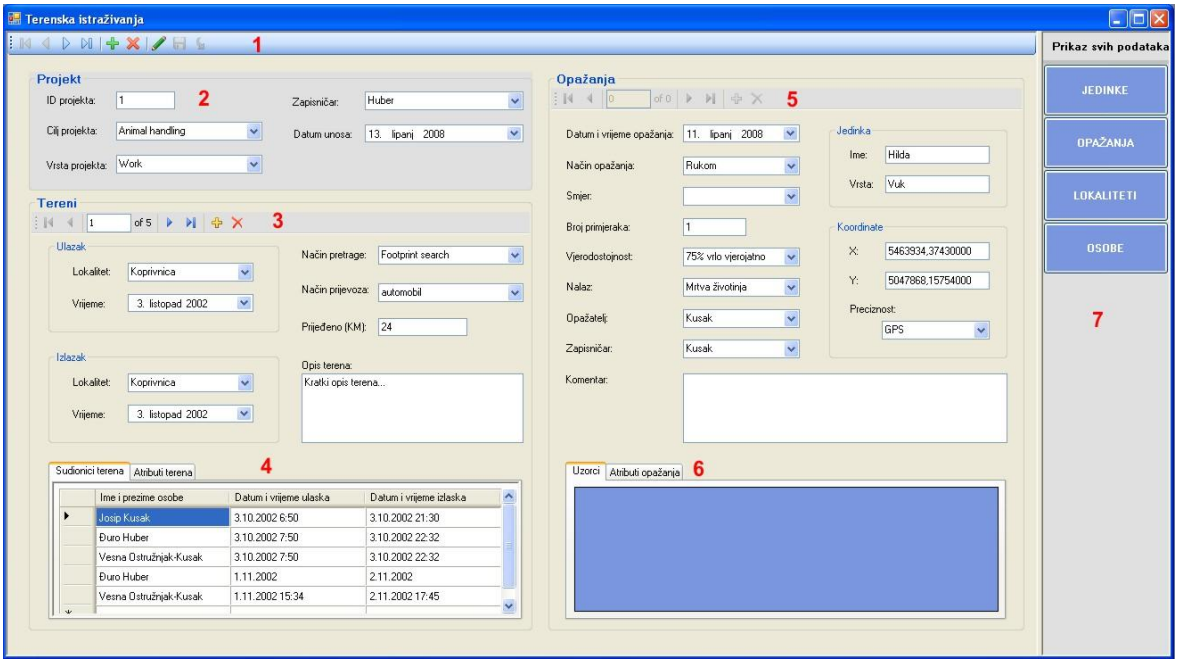
Slika 5-3: Izbornik za pokretanje modula

Odabirom odgovarajuće stavke, pokreće se odabrani modul i otvara nova zaslonska maska unutar početne zaslonske maske. Korištenje svakog od modula objašnjeno je u nastavku.

5.2.2. Modul „Terenska istraživanja“

Rad s podacima vezanim uz terenska istraživanja započinje otvaranjem zaslonske maske „TerenskaIstrazivanjaForm“. Ova zaslonska maska prikazuje tri glavne skupine podataka i njihovu međuovisnost. Podaci o pojedinim entitetima (projekt, teren, opažanje) dobavljaju se preko objekata tipa „EntitetBllProvider“, a podaci iz šifrarnika preko DataSet objekta „dsWLO“. Glavni izvor podataka na zaslonskoj masci je lista svih projekata. Ostali podaci se prikazuju u ovisnosti o trenutno prikazanom zapisu o projektu.

Na slici 5-4 prikazana je zaslonska maska „TerenskaIstrazivanjaForm“.



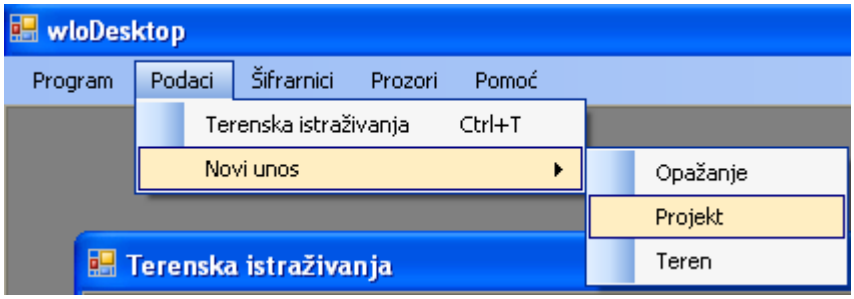
Slika 5-4: Zaslonska maska „TerenskaIstrazivanjaForm“

U nastavku su ukratko objašnjeni pojedini dijelovi zaslonske maske koji su na slici označeni brojevima.

1. **Glavni navigator** – glavni navigator zaslonske maske omogućuje kretanje kroz pojedine zapise o projektima. U ovisnosti o trenutnom projektu, prikazani su odgovarajući podaci o terenima i opažanjima vezanim uz taj projekt. Osim kretanja kroz zapise, navigator omogućuje i pozivanje zaslonske maske za unos novog projekta te brisanje i izmjenu trenutnog projekta (uz mogućnost poništenja načinjenih izmjena).
2. Dio zaslonske maske namijenjen prikazu podataka o projektima.

3. Dio zaslonske maske namijenjen prikazu podataka o terenima povezanim s trenutnim projektom. Na vrhu ovog dijela zaslonske maske nalazi se novi navigator koji omogućuje manipulaciju zapisima o terenima (na analogan način kao i kod navigatora za projekte). U ovisnosti o trenutnom terenu u donjem dijelu ovog dijela zaslonske maske prikazuju se podaci o sudionicima i atributima terena (podaci se prikazuju pomoću objekta DataGridView).
4. Dio zaslonske maske za prikaz podataka o sudionicima i atributima terena.
5. Dio zaslonske maske za prikaz podataka o opažanjima. Slično kao i kod stavke 3, na vrhu se nalazi novi navigator koji omogućuje manipulaciju zapisima o opažanjima.
6. Dio zaslonske maske za prikaz podataka o uzorcima i atributima opažanja povezanim s trenutnim zapisom o opažanju.
7. **Brzi izbornik** – omogućuje brzo pozivanje zaslonskih maski za pregled i ažuriranje svih dostupnih podataka o pojedinim entitetima iz baze podataka (jedinke, lokaliteti, osobe...).

Za unos novih podataka korisnik može kliknuti na gumb za dodavanje novog zapisa (znak '+') na odgovarajućem navigatoru ili odabrati stavku za unos željenog zapisa iz glavnog izbornika. Odabir stavke za unos novog projekta iz glavnog izbornika prikazan je na slici 5-5.



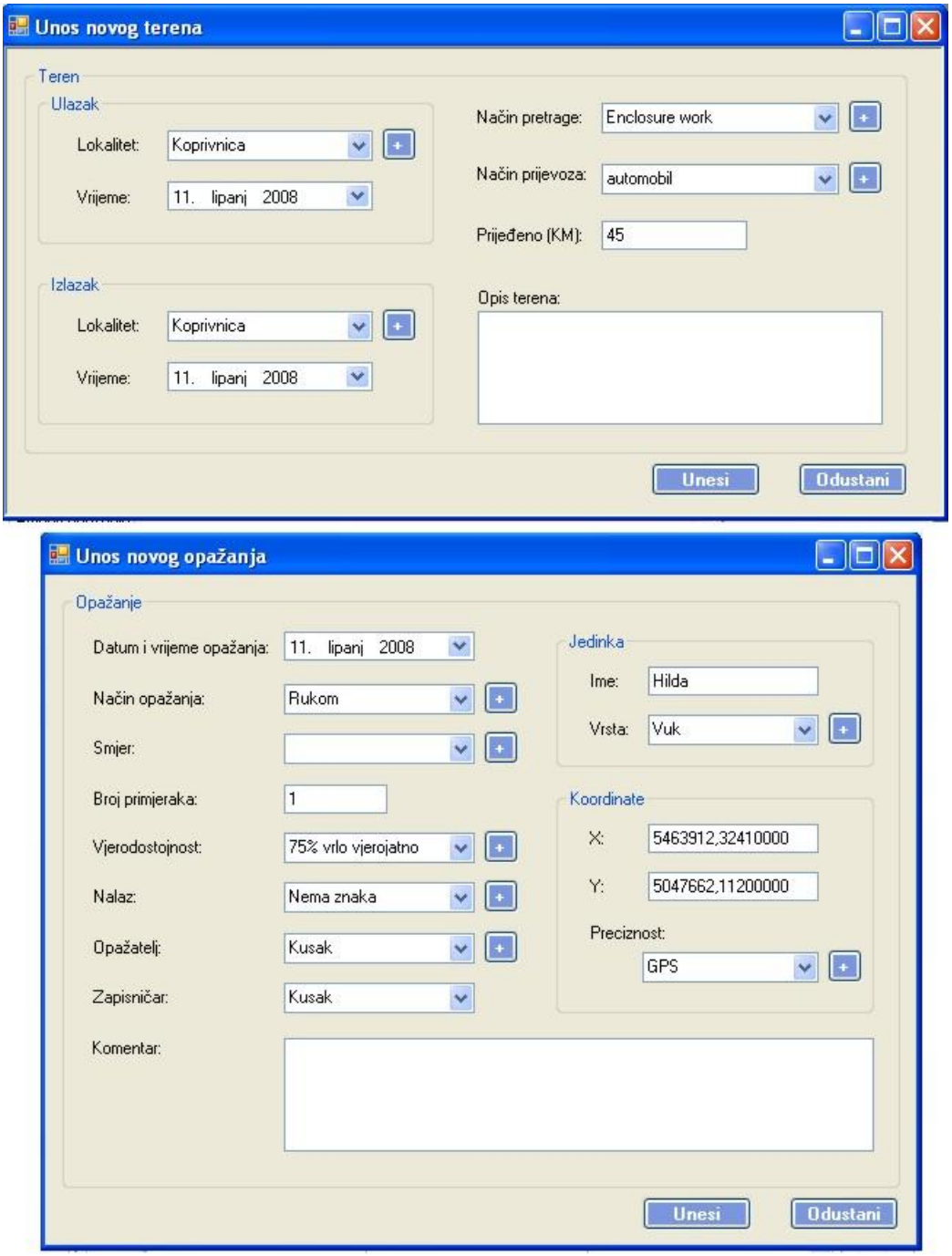
Slika 5-5: Odabir stavke za unos novog projekta

Odabirom stavke za unos novog projekta, korisniku se otvara zaslonska maska „NoviProjektForm“ prikazana na slici 5-6.

Slika 5-6: zaslonska maska „NoviProjektForm“

Možemo vidjeti da je s desne strane svakog padajućeg menija koji prikazuje podatke iz šifrarnika mali gumb sa znakom '+'. Klikom na gumb koji se nalazi pokraj odgovarajućeg padajućeg menija, otvara se generička zaslonska maska namijenjena ažuriranju podataka za šifrarnik čije podatke dotični padajući meni sadrži. Na ovaj način korisniku je omogućeno brzo dodavanje zapisa u šifrarnik u slučaju da željeni zapis nije ponuđen u padajućem meniju. Klikom na gumb „Unesi“ željeni zapis će se pohraniti u bazu podataka.

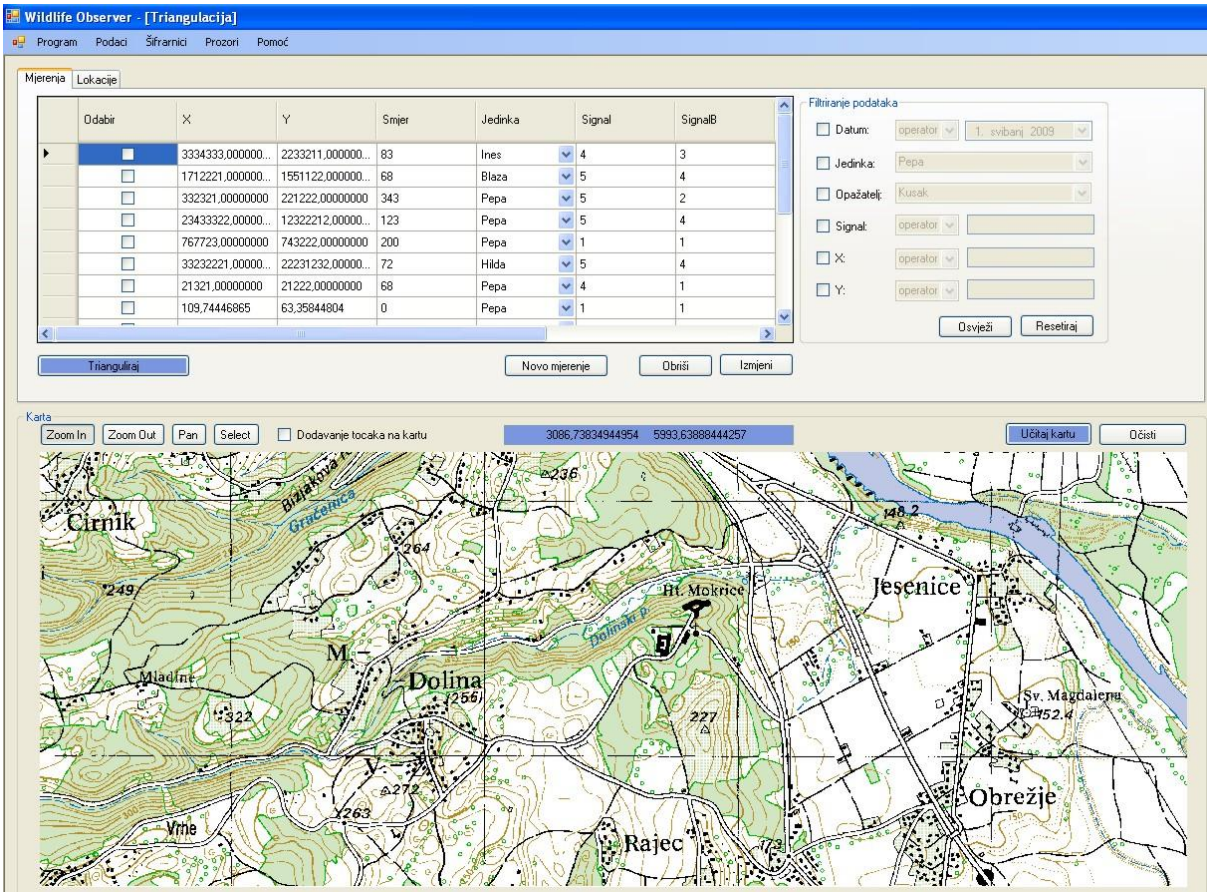
Osim za unos novog projekta, postoje i zaslonske maske za unose ostalih vrsta podataka. Na slici 3.15 prikazane su zaslonske maske za unos novog opažanja (NovoOpazanjeForm) i novog terena (NoviTerenForm).



Slika 5-7: Zaslonske maske „NoviTerenForm“ i „NovoOpažanjeForm“

5.2.3. Modul „Triangulacija“

Odabirom stavke „Triangulacija“ iz izbornika „Podaci“ pokreće se modul za rad s telemetrijskim podacima. Početni izgled zaslonske maske ovog modula prikazan je na slici 5-8.



Slika 5-8: Zaslonska maska „Triangulacija“

Početna zaslonska maska sastoji se od dva osnovna dijela:

- Podatkovni dio – prikaz izvršenih mjerenja radio signala i filtriranje podataka
- Karta – prikaz geokodiranih informacija (mjerenja i lokacija)

U podatkovnom dijelu postoje dvije kartice, ovisno o vrsti podataka koje korisnik želi pregledavati i prikazivati na karti:

- Mjerenja – podaci vezani uz mjerenja radio signala
- Lokacije – prikaz lokacije dobivenih korištenjem algoritma triangulacije

U dijelu za rad s mjerenjima, omogućeno je i jednostavno filtriranje podataka. Dio zaslonske maske koji omogućuje filtriranje prikazan je na slici 5-9.

Slika 5-9: Filtriranje podataka o mjerenjima

Korisniku je omogućeno filtriranje mjerenja prema slijedećim atributima:

- Datum
- Jedinka
- Opažatelj
- Signal
- Koordinata X
- Koordinata Y

Prilikom filtriranja, moguće je koristiti kombinaciju jednog ili više atributa prema kojima korisnik želi filtrirati podatke. Uključivanje pojedinog atributa prema kojem se žele filtrirati podaci postiže se odabirom odgovarajućeg okvira za izbor. Osim toga, za svaki pojedini atribut moguće je odabrati željeni operator usporedbe. Nakon odabira željenih atributa za filtriranje, korisnik osvježava prikazane podatke o mjerenjima klikom na gumb „Osvježi“.

Donji dio forme predstavlja interaktivnu kartu uz pomoć koje je omogućen prikaz geokodiranih informacija korisniku radi lakše analize podataka. Samo korisničko sučelje karte je izgrađeno tako da se znanstvenicima omogući što lakše i intuitivnije korištenje ovog dijela aplikacije.

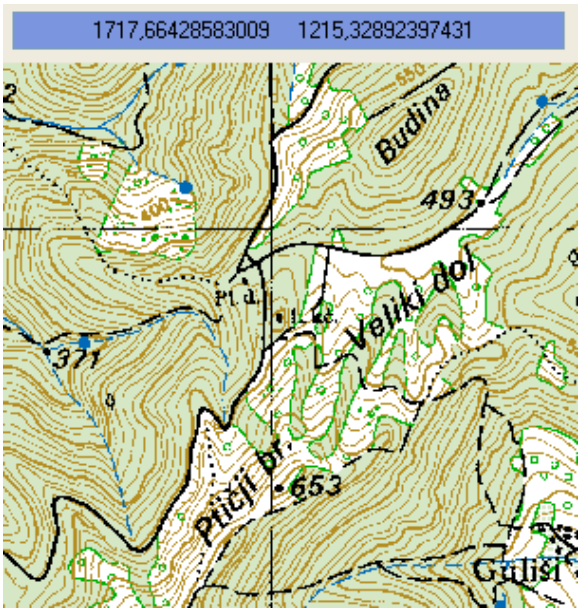
Za rad sa geokodiranim informacijama potrebno je učitati određeni geografski medij

na kojem će se promatrati ti podaci. Učitavanje medija postiže se tako da korisnik odabere gumb „Učitaj kartu“ te iz izbornika odabere lokaciju na kojoj je pohranjen dotični medij. Nakon što je medij učitan, korisnik ima mogućnost daljnjeg rada sa kartom.

Slika 5-10: Kontrole za upravljanje kartom

U gornjem lijevom kutu nalaze se kontrole pomoću kojih se upravlja samom kartom. Tri osnovne mogućnosti su *zoom* i *zoom out* određenog područja na karti te pomicanje po karti (*pan*) dok se četvrta mogućnost odnosi na odabir određene lokacije na karti. Prilikom promjene načina upravljanjem kartom korisnik može vidjeti i promjenu pokazivača miša koji mu signalizira u kojem načinu rada se trenutno nalazi.

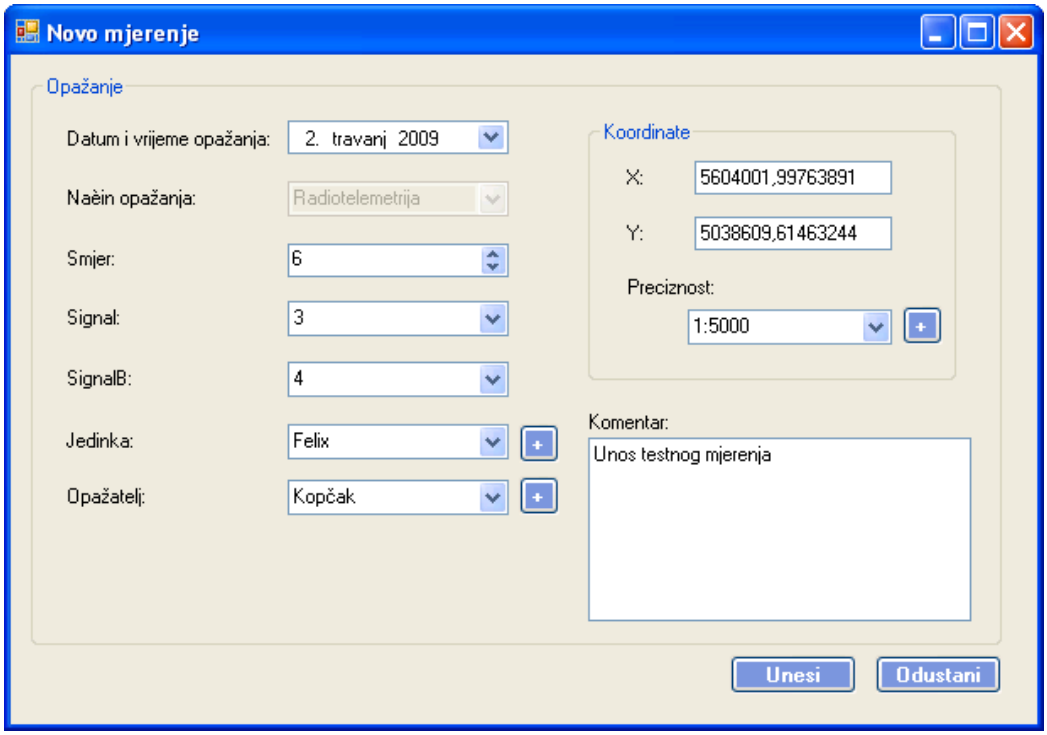
Prilikom rada sa kartom, omogućena je opcija pregleda koordinata lokacija koje se promatraju tako da se pri prevlačenju miša preko određene lokacije njene koordinate prikazuju pri vrhu karte u posebnom, plavo obojanom, dijelu predviđenom za taj prikaz.



Slika 5-11: Pregled koordinata lokacija na karti

Ukoliko korisnik odabere opciju „Select“, potrebno je odabrati klikom miša određenu lokaciju na karti kako bi se izvršio unos novog mjerenja. Nakon toga pojavljuje se nova

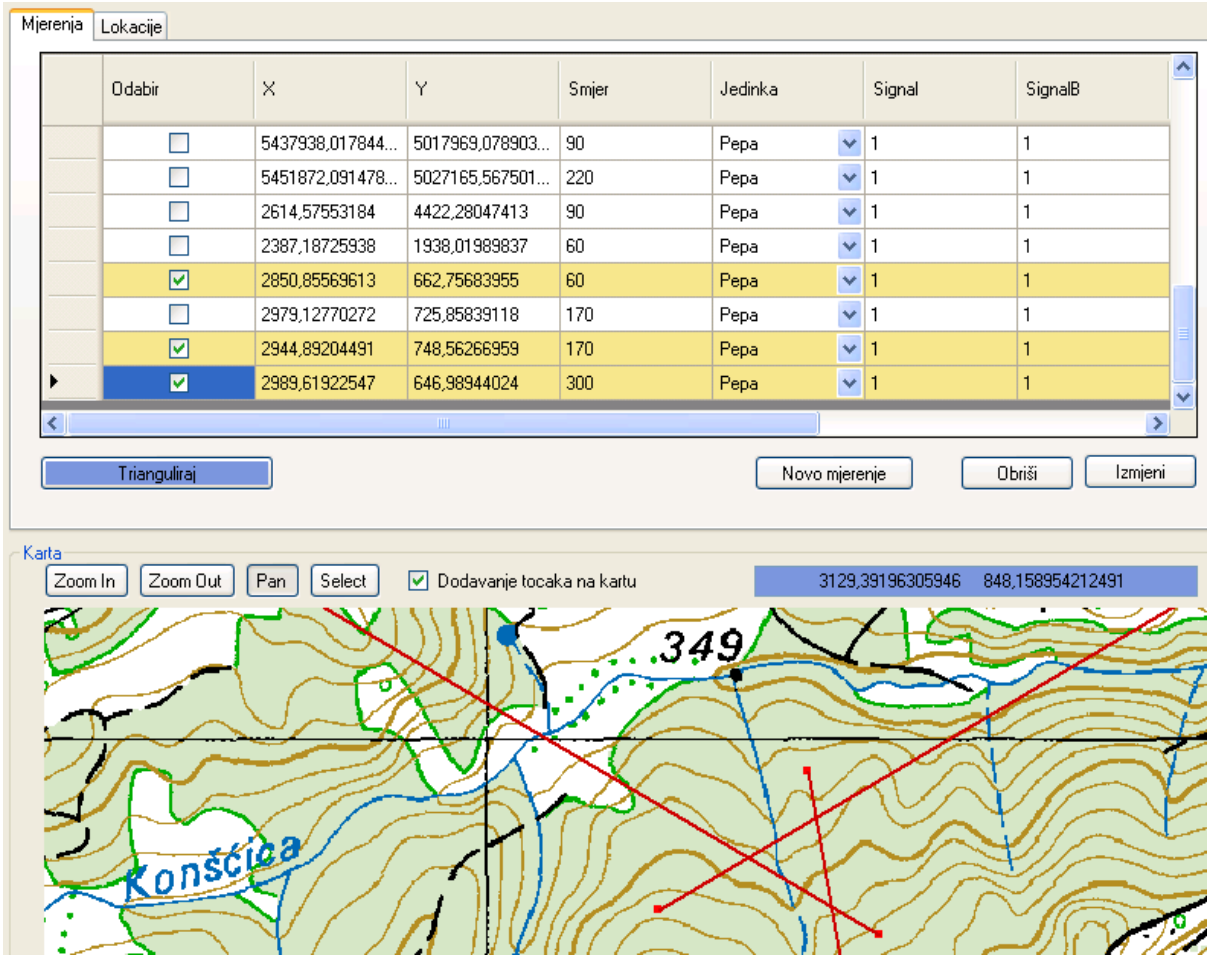
forma za unos novog mjerenja prikazana na slici 5-12.



Slika 5-12. Otvaranje forme za unos mjerenja nakon klika mišem na kartu

Na slici sa prikazom forme može se primijetiti da su polja za koordinate već ispunjena sa koordinatama one lokacije na koju je korisnik kliknuo mišem. Ova funkcionalnost izgrađena je kako bi korisniku bio omogućen još jedan način unosa podataka osim onog standardnog kada korisnik već poznaje koordinate lokacije. Uz koordinate, korisnik ima mogućnost unosa podataka karakterističnih za svako mjerenje. Opis tih podataka detaljnije je definiran u poglavlju koje opisuje model podataka korišten pri izgradnji ove aplikacije. Na kraju unosa svih podataka korisnik ima mogućnost unosa ili odustanka od unosa podataka u bazu podataka.

Kroz analizu podataka korisniku je potrebno prezentirati podatke koje je već prikupio. Stoga postoji mogućnost pregleda svih podataka o mjerenjima s kojim korisnik raspolaže u bazi podataka. Za prikaz podataka potrebno je odabrati opciju „Dodavanje točaka na kartu“. Zatim je nužno odabrati iz gornjeg popisa svih mjerenja samo ona mjerenja koja se želi prikazati na karti. Korisnik jednostavnim odabirom mjerenja dobije prikaz tih točaka na kartu što je prikazano na slici 5-13.

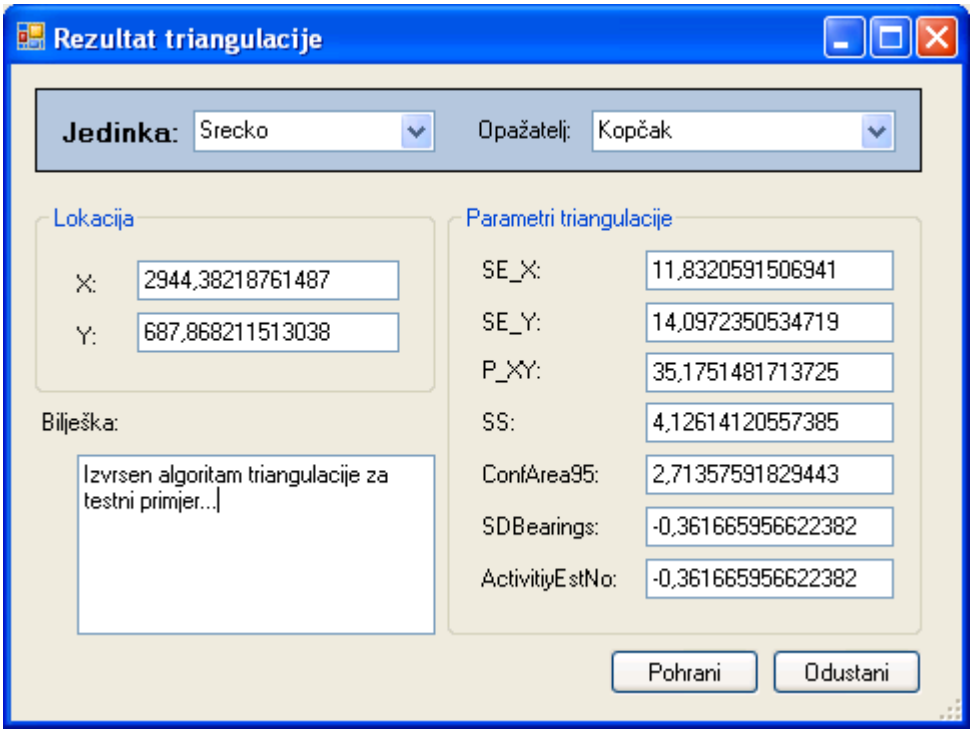


Slika 5-13. Dodavanje točaka na kartu

Implementirana je mogućnost upravljanja sa prikazanim točkama tako da se dodane točke mogu i ukloniti sa karte na analogan način kako su i dodane na nju, jednostavnim odabirom točke koja je već prikazana na karti. Na taj način omogućeno je znanstvenicima da procjenjuju ona mjerenja koja su zadovoljavaju uvjete za izvršenje algoritma triangulacije. Prikaz točke određen je sa samom koordinatom točke koja se prikazuje i na slici je prikazan sa malim kvadratićem dok se iscrtava još i smjer signala koji je određen prilikom samog mjerenja, a prikazan je linijom koja izlazi iz točke lokacije.

Prilikom odabira točaka željenih mjerenja korisniku su još uvijek dostupne sve kontrole za upravljanje kartom ukoliko je potrebna dodatna analiza područja na karti.

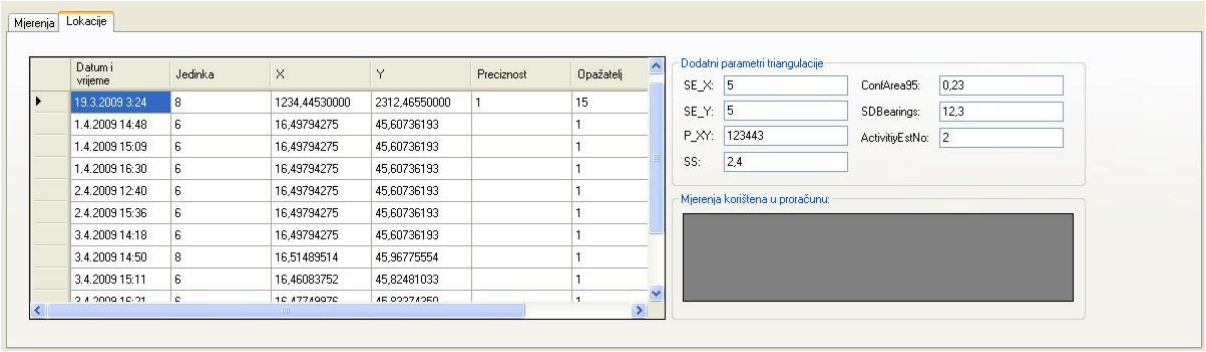
Nakon što korisnik odabere ona mjerenja koja ga zanimaju pruža mu se mogućnost izvršavanja algoritma triangulacije za točno ta mjerenja. Potrebno je samo odabrati opciju „Trianguliraj“ i pojaviti će se nova forma sa rezultatima algoritma.



.**Slika 5-14:** Rezultat algoritma

Rezultat algoritma koji je predočen korisniku sadrži koordinate izračunate točke za dobivenu lokaciju te dodatne parametre algoritma triangulacije koji je opisan u prethodnom poglavlju. Nakon prikaza rezultata korisnik ima mogućnost pohrane ili odbacivanja rezultata. Klikom na „Pohrani“ dobivena lokacija pohranjuje se u bazu podataka.

Odabirom kartice „Lokacije“ prebacujemo se na dio zaslonske maske namijenjen ažuriranju podataka o lokacijama, prikazanom na slici 5-15.



Slika 5-15: Dio zaslonske maske namijenjen ažuriranju podataka o lokacijama

Slično prethodno objašnjenom dijelu za mjerenja, dio za prikaz lokacija s lijeve strane sadrži popis svih lokacija dobivenih algoritmom triangulacije. Odabirom pojedine lokacije, s

desne strane korisniku su prikazani dodatni parametri triangulacije, kao i popis svih mjerenja korištenih pri izračunu dotične lokacije. Osim toga, odabirom pojedine lokacije ona se prikazuje na karti zajedno sa svim mjerenjima.

5.1 Sustav na džepnom računalu

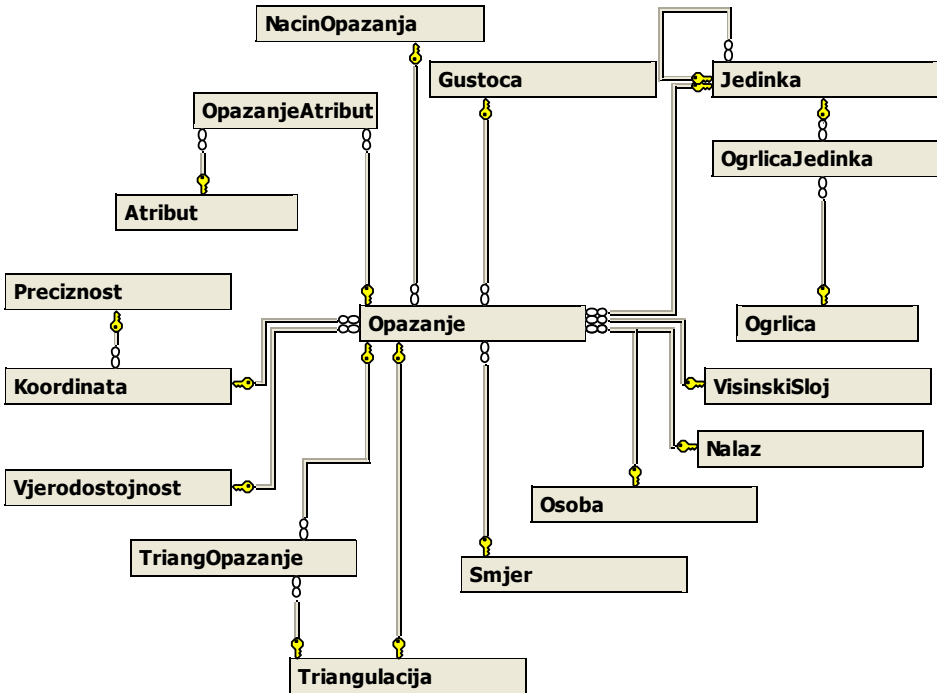
WloPad je dio „WildLife Observer“ (WLO) sustava koje je razvijeno kako bi omogućilo praćenje divljih životinja u pokretu. Temelji se na aplikaciji ArcPad te je uz pomoć ArcPad Application Builder alata razvijena dodatna funkcionalnost koja je potrebna za efikasno praćenje životinja. Cilj sustava je pružiti terenskom istraživaču jedno mjesto putem kojeg će moći obavljati sve svoje dužnosti. Korištenjem WloPad aplikacije istraživač može koristiti ArcPad za svoju orijentaciju na terenu i imati mogućnost telemetrijskog praćenja životinje, uvid u postojeće podatke i unos novih podataka.

U ovom poglavlju će biti objašnjeno kako izgleda model podataka, kako komuniciraju pojedine klase i objekti, izgled i uporabu korisničkog sučelja te izgled arhitekture.

5.1.1 Opis modela podataka

Aplikacija koristi dio baze koja je izgrađena na temelju složenijeg modela podataka koji se koristi na Windows platformi te ćemo u ovom poglavlju prikazati samo model podataka vezan za funkcionalnost praćenja životinje u pokretu odnosno telemetriju. Cjelovita baza podataka projektirana je u projektu „Fauna Croatica Database“ [6], a WloPad koristi jedan podmodel te baze podataka uz manje izmjene koje su izvršene zbog specifičnosti praćenja životinja. Baza je mobilnoga tipa jer se nalazi lokalno na džepnom računalu te se spojnom replikacijom sa središnjim sustavom podaci razmjenjuju i osvježavaju. Time je omogućeno da korisnik može posjedovati podatke od drugih istraživača koji prate životinje ili da može podatke obrađivati na stolnom računalu.

5.1.2 Konceptualni izgled baze podataka WLO



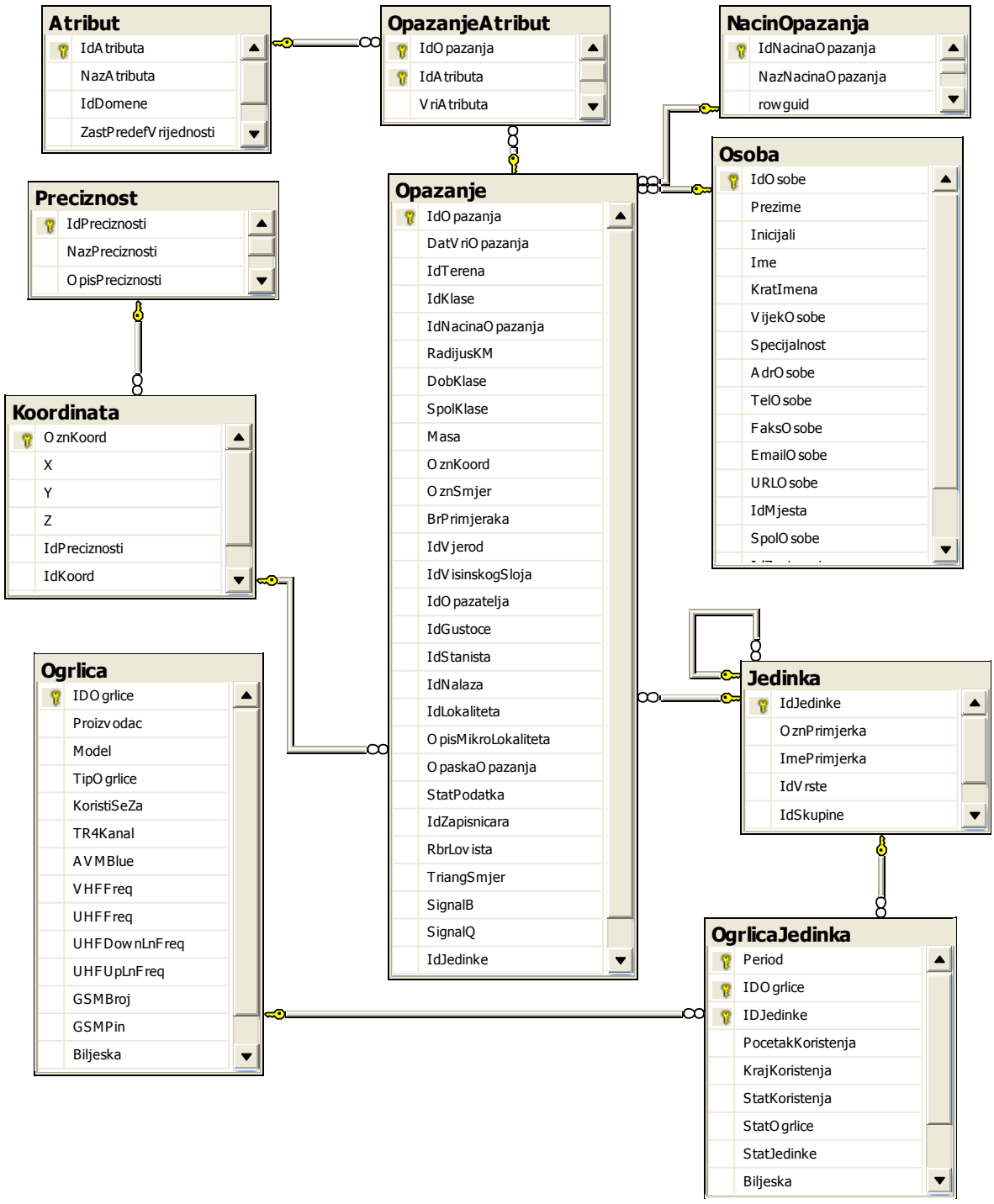
Slika 5-16: Prikaz izgleda konceptualne baze podataka

Slikom 5-16 dan je prikaz konceptualne baze podataka koja se sastoji od različitih tablica. Centralni dio baze vezan za praćenje životinje je tablica Opažanje u koju se upisuju svi podaci vezani za bilo kakvu pojavu koja se mogla primijetiti u domeni praćenja.

Tablica 5-1: Popis tablica u konceptualnoj bazi podataka

Naziv tablice	Značenje
Gustoca	Gustoća pojavljivanja opažene skupine
Jedinka	Jedinka ili skupina koju možemo razlikovati od drugih
Koordinata	Geografska koordinata, točka
NacinOpazanja	Način na koji je neka jedinka opažena (npr. fotografiranjem)
Nalaz	Kategorija opažanja (npr. trag, ulov ...)
Odredio	Osoba koja je odredila neki uzorak
Ogrlica	Popis svih ogrlica i podataka o njima
OgrlicaJedinka	Povezuje pojedine ogrlice sa jedinkama koje ih koriste
Opazanje	Opažanje jedinke, pohrana podataka o smjerovima za triangulaciju
OpazanjeAtribut	Dodatni korisnički atributi opažanja
Osoba	Bilo koja osoba u sustavu
Preciznost	Preciznost koordinate
Smjer	Smjer na kompasu
Triangulacija	Rezultati triangulacije
TriangOpazanje	Spojna tablica koja povezuje mjerenja sa izračunatom pozicijom
Vjerodostojnost	Vjerodostojnost (pouzdanost) opažanja

5.1.3 Telemetrijska mjerenja



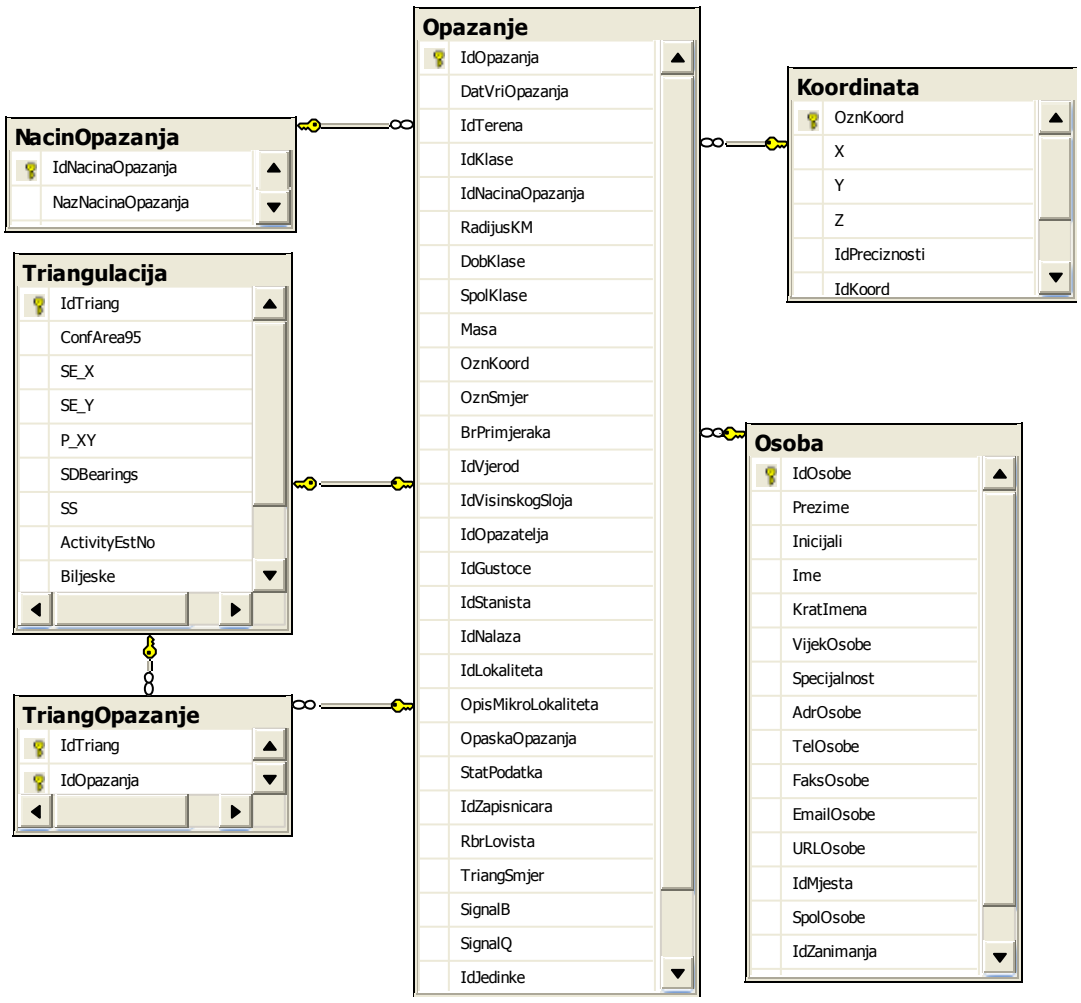
Slika 5-17: Prikaz tablica vezanih za podatke mjerenja u telemetriji

Pogled na bazu podataka koji je vezan za mjerenja u telemetriji odnosi se na unos, obradu i analizu mjerenja koja istraživač vrši kako bi mogao dati ulazni skup algoritmu triangulacije i na taj način odrediti položaj životinje. Mjerenje je definirano položajem i smjerom iz kojeg dolazi najjači signal, no to nisu svi podaci. Podaci su podijeljeni po

tablicama na sljedeći način:

- **Opažanje –** Središnja tablica u bazi, omogućava pristup podacima vezano za telemetrijska mjerenja kao što su vrijeme mjerenja, bilješka vezana za mjerenje, smjer iz kojeg dolazi radio signal, kvaliteta signala i kvaliteta signala prijašnjih mjerenja, a osim toga sadrži veze na druge tablice.
- **Koordinata –** Tablica u kojoj su pohranjeni svi položaji, sadrži atribute položaja mjerenja.
- **Jedinka –** Tablica sadrži popis životinja koje se prate što nam je važno kada želimo vidjeti mjerenja za svaku životinju posebno.
- **Osoba –** Podaci o osobama koje koriste sustav ili rade u sustavu.
- **Ogrlica –** Podaci o ogrlici koji omogućuju uvid u ispravnost ogrlice.
- **OgrlicaJedinka –** Tablica koja daje informaciju o tome koja jedinka nosi koju ogrlicu.
- **NacinOpazanja –** Tablica koja je povezana na Opazanje i omogucava da odredimo koju vrstu opazanja smo vršili.

5.1.4 Položaj određen triangulacijom



Slika 5-18: Prikaz tablica vezanih za podatke pozicija određenih triangulacijom

Pogled na bazu podataka koji omogućuje pohranu i prikaz položaja životinja određenih telemetrijskim praćenjem. Opažanje je središnje mjesto pogleda.

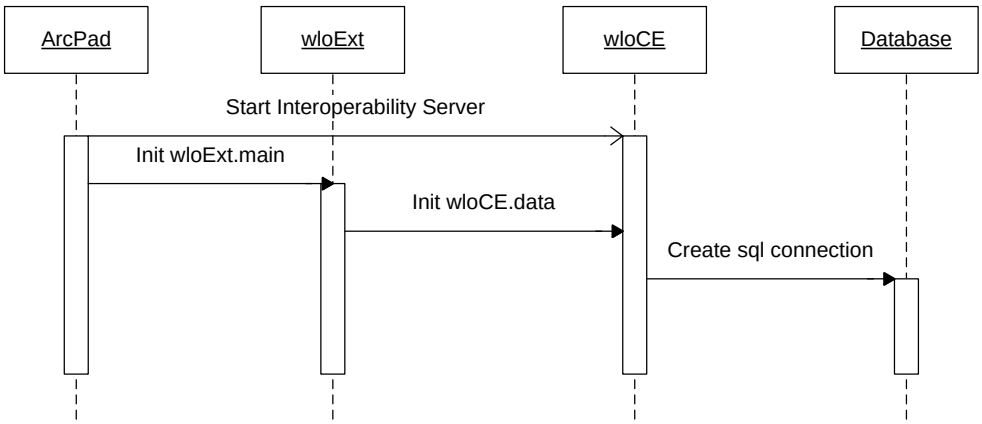
- **Opažanje** – Tablica u koju u ovom pogledu pohranjujemo vrijeme određivanja položaja triangulacijom i sadrži veze na ostale tablice koje sadrže ostale podatke.
- **Triangulacija** – Tablica koja sadrži podatke o izlaznim varijablama algoritma triangulacije (površina elipse pogreške, standardna devijacijska pogreška po x i y varijabli i dr.).
- **TriangOpazanje** – Tablica koja veže mjerenja koja su korištena u određivanju položaja uz pomoć algoritma triangulacije sa izračunatim položajem.
- **Koordinata** – Pohranjujemo koordinate izračunatog položaja životinje.

5.1.5 Slijed izvođenja programskih funkcija

U ovom poglavlju ćemo objasniti slijed izvođenja WloPad programa. Što se događa unutar komponente kada korisnik koristi neke funkcionalne dijelove komponente. Osnovni tijekovi funkcija se baziraju na komunikaciji ArcPad mobilne GIS aplikacije i komponente napisane u .NET tehnologiji pod imenom wloCE. Situacija je specifična jer koristimo funkcionalnosti napisane u upravljanom kodu iz aplikacije koja omogućuje proširenje funkcionalnosti u skriptnom jeziku. U dijagramima se nalaze tri sloja, ArcPad kao aplikacija koja komunicira s korisnikom preko karte i alatnih traka, wloExt komponente koju kreira Native Code Generate opisan u poglavlju 4.6, i koja omogućuje komunikaciju izvornog i upravljanog koda, komponenta wloCE koja sadrži sve funkcionalnosti vezane za unos i obradu podataka te mobilna baza podataka. Prikazat ćemo kako se sustav pri pokretanju podiže i priprema za rad, kako se upisuju podaci mjerenja i kako se prikazuju podaci iz mobilne baze podataka.

5.1.6 Pokretanje sustava

Korisnik prilikom pokretanja svoje središnje aplikacije ArcPad želi automatski koristiti i funkcionalnosti vezane za praćenje životinja, a koje se nalaze u komponenti wloCE. Kako bi to bilo moguće potrebno je napraviti određene akcije.



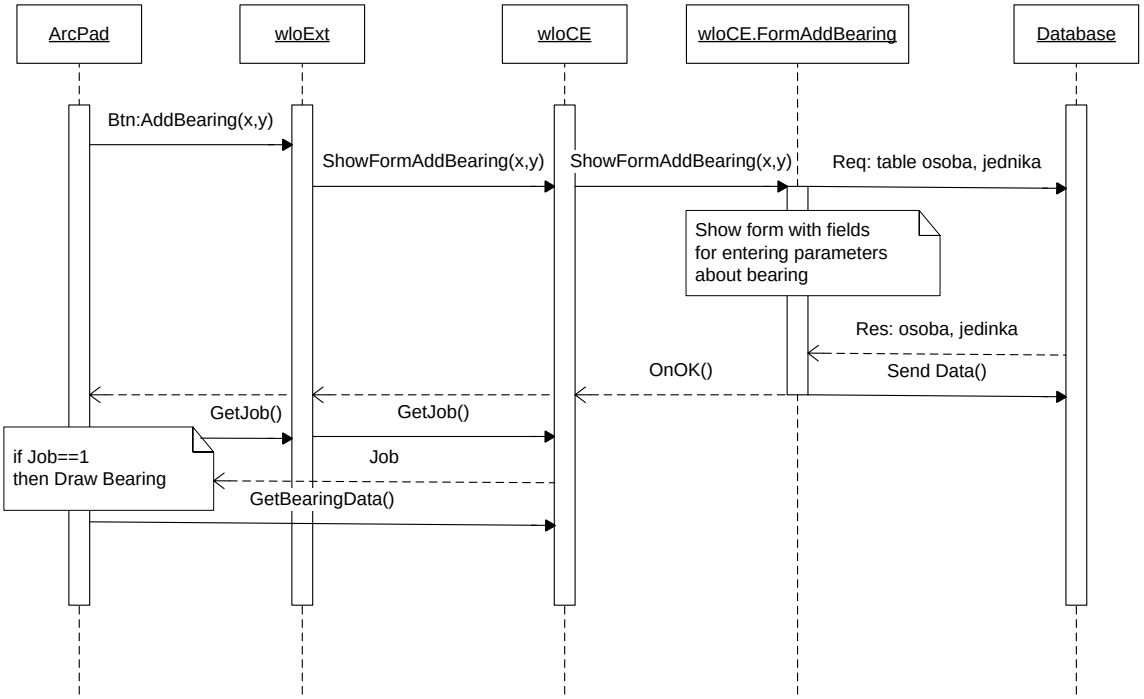
Slika 5-19: Prikaz slijeda izvođenja prilikom pokretanja sustava

Prilikom pokretanja ArcPad programa potrebno je pokrenuti Interoperability Server koji omogućava pozivanje funkcionalnosti komponente wloCE. Nakon što se Interoperability Server podigao ArcPad program inicijalizira wloExt komponentu koja služi zajedno s

Interoperability Server programom kao most koji nam omogućuje da pozovemo iz skriptnog jezika upravljani kod. Pokretanje wloPad sustava završava kreiranjem konekcije prema bazi podataka.

5.1.7 Upis mjerenja telemetrije

WloPad program omogućuje korisniku da koristeći jednu aplikaciju može imati mobilni GIS sustav i pratiti životinje. Kako bi se to svojstvo ostvarilo omogućeno je putem ArcPad programa upis mjerenja telemetrijom.

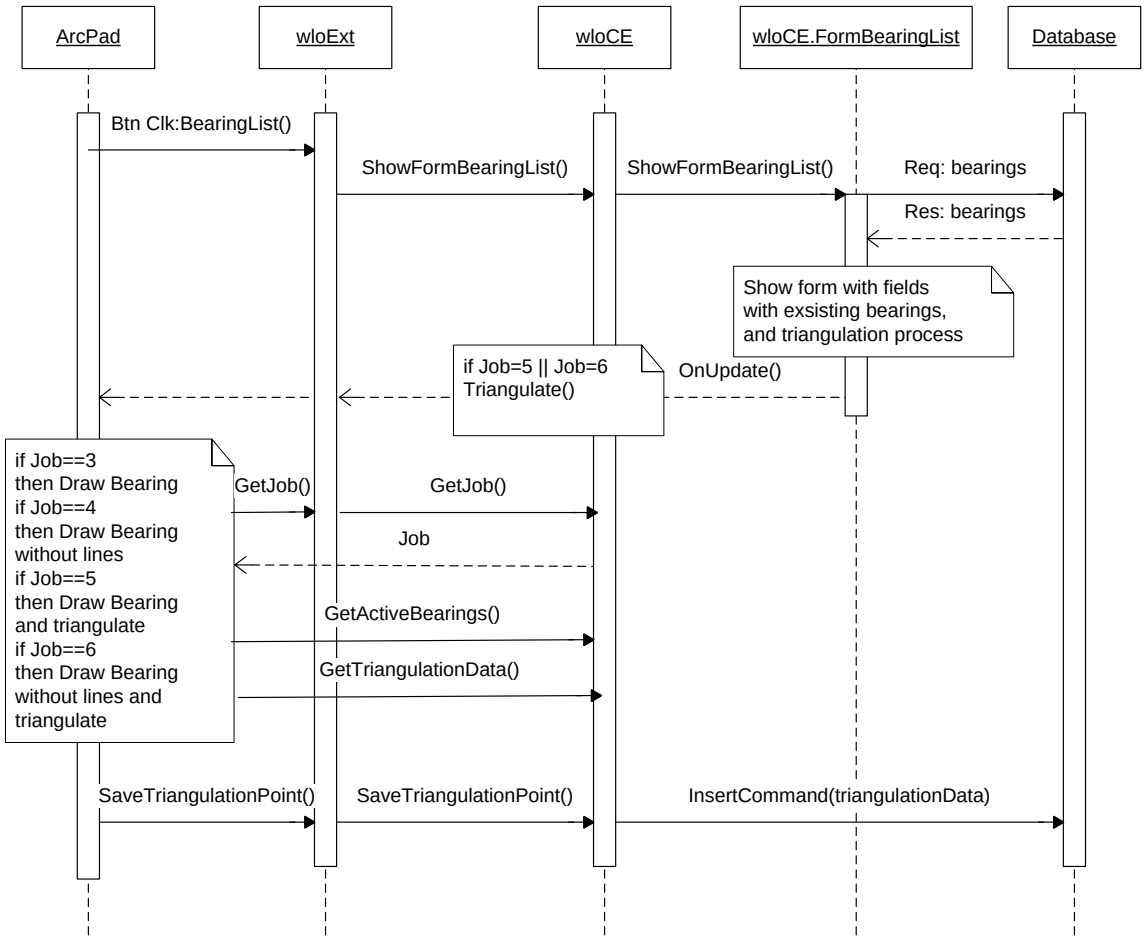


Slika 5-20: Prikaz slijeda izvođenja prilikom upisa mjerenja telemetrije

Na slici 5-20 vidimo kako se odvija komunikacija među komponentama kada korisnik odluči putem ArcPad programa upisati mjerenje koje je na terenu upravo izvršio. Kada korisnik klikne na gumb „Add bearing“ pozove se funkcija u wloExt komponenti i predaje joj se koordinata na kojoj se korisnik trenutno nalazi. WloExt komponenta potom poziva preko Interoperability Server programa wloCE komponentu, odnosno metodu ShowFormAddBearing kojoj se predaju koordinate. Pozivom se aktivira forma za unos podataka mjerenja te se iz mobilne baze podataka pribavljaju podaci o životinjama i osobama. Korisnik potom upisuje podatke u formu i akcijom unosa podatka daje znak

komponenti da podatke validira i spremi u bazu podataka. Kada se podaci spreme u bazu podataka komponenta daje znak ArcPad programu da je forma za unos mjerenja zatvorena koristeći varijablu Job. ArcPad potom mora zatražiti podatke o upisanom mjerenju kako bi mogao iscrtati smjer i točku na karti.

5.1.8 Pregled mjerenja i triangulacija



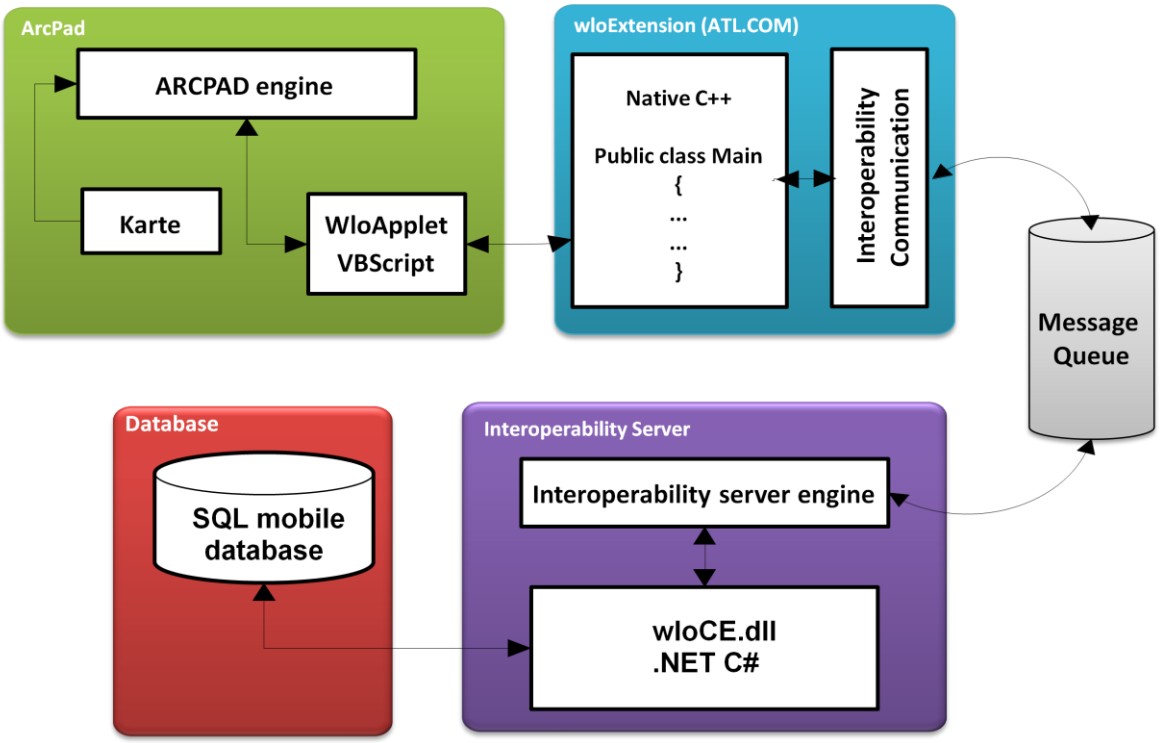
Slika 5-21: Prikaz slijeda izvođenja prilikom pregleda mjerenja i triangulacije

Korisnik pritiskom gumba na alatnoj traci pokreće akciju otvaranja forme koja prikazuje popis svih dosad obavljenih mjerenja u telemetrijskom praćenju životinje. Forma zahtijeva podatke o mjerenjima koji se nalaze u mobilnoj bazi podataka. Korisnik odabire koja mjerenja će biti ulazni skup algoritma triangulacije i prilikom korisničke akcije OnUpdate() ovisno koje su postavke odabrane vrše se određene akcije. Korisnik ne mora

izvršiti triangulaciju već može samo iscrtati mjerenja sa ili bez pravca smjera iz kojeg je dolazio najjači signal. wloCE komponenta izvršava algoritam triangulacije ako je tako odabrano. ArcPad sustav traži podatke ovisno o tome koje su akcije izabrane i iscrtava podatke na kartu. Ako je izračunata pozicija zadovoljavajuća korisnik potvrdnom akcijom pokreće poziv funkcija SaveTriangulationPoint() iz ArcPad sustava do wloCE komponente koja izvršava nad bazom Insert naredbu i upisuje podatke.

5.1.9 Dizajn arhitekture i komponenti

WloPad je aplikacija koja ima doista zanimljivu arhitekturu i izgled te namjenu pojedinih komponenti. Arhitektura je oblikovana kako bi dala rješenje na tehnološke probleme, zahtjeve korisnika i uvjete rada aplikacije. Praćenje životinja se događa na nepristupačnom terenu što stvara potrebu za džepnim računalom koje koristi Microsoft Compact Framework. Microsoft Compact Framework je slabijih mogućnosti od Microsoft Frameworka na računalima što otežava razvoj i funkcionalnosti. Kako je teren nepristupačan nemoguće je osloniti se na web servise kao što su Google Maps ili Microsoft Virtual Earth jer ne postoji pristup Internetu, a ako i postoji propusnost je mala i skupa. Zato je odlučeno da se koriste GIS aplikacije koje imaju mogućnost rada sa kartama koje se mogu lokalno pohraniti na džepno računalo. Takve aplikacije bi trebale imati mogućnost proširenja funkcionalnosti kako bi se mogla dodati funkcija praćenja životinja. Izborom je odlučeno da se koristi ESRI ArcPad koji omogućava proširenje putem ekstenzija ili dodataka (engl. applet) pisanih u skriptnom jeziku. Problem tehnološke prirode nastaje u uočavanju nedostatka Microsoft Compact Frameworka 2.0 koji ne podržava interoperabilnost između izvornog i upravljanog koda što predstavlja izuzetan problem i prepreku. Potreba da se ArcPad poveže sa mobilnom bazom podataka je ključna za projekt. Danas je interoperabilnost izuzetno cijenjena jer omogućava povezivanje i suradnju već izgrađenih aplikacija da zajedno rade i dijele podatke što povećava kvalitetu ukupne aplikacije te brzinu razvoja. Problem je riješen korištenjem aplikacije Mirror objašnjenje u poglavlju 4.6 iako s puno rada i testiranja jer je aplikacija samo pomoć za riješiti nedostatak interoperabilnosti i sadržava izuzetno malo dokumentacije.



Slika 5-22: Model arhitekture aplikacije WloPad

Na slici se 5-22 se nalazi izgled arhitekture sustava WloPad. Sustav se sastoji od ArcPad aplikacije, WloApplet dodatka, wloExtension komponente, Interoperability Server aplikacije, wloCE komponente i mobilne baze podataka.

- **ArcPad** – Mobilni GIS sustav
- **WloApplet** – Dodatak razvijen u ArcPad application Builder razvojnom okruženju, napisan je u VBScript skriptnom jeziku i upravlja WLO alatnom trakom i sadrži procedure koje iscrtavaju mjerenja u telemetriji, rezultate triangulacije te komunicira sa wloExtension komponentom. Podatke koje iscrtava potječu iz mobilne baze podataka .
- **wloExtension** – *Active Template Library* (ATL) komponenta napisana u C++ programskom jeziku. Sadrži dualne datoteke i metode iz wloCE komponente i na taj način uz pomoć Interoperability Server aplikacije i reda poruka omogućuje da pozivi iz WloApplet dodatka budu proslijeđeni u wloCE komponentu.
- **Interoperability Server** – Interoperabilni most između izvornog i upravljanog koda objašnjen u poglavlju 4.6.
- **wloCE** – Komponenta napisana upravljanim C# kodom. Koristi Microsoft Framework 2.0 čime je omogućen brz i kvalitetan razvoj putem Visual Studio

razvojnog okruženja. Sadrži forme i metode putem kojih korisnik unosi, analizira i obrađuje podatke o praćenju životinja telemetrijom.

- **Mobilna baza podataka** – Baza podataka u SQL mobile tehnologiji koja spojnom replikacijom razmjenjuje podatke sa glavnom bazom i sadrži podatke važne za praćenje životinja.

Pogledom na sliku 5-22 možemo uočiti da je ovim arhitekturalnim rješenjem omogućeno da skriptni jezici mogu komunicirati s bazom podataka na način da vrše upite i obradu podataka te da koriste Microsoft Framework, osim toga to nije samo moguće na platformi Microsoft Windows već i na Microsoft Windows Mobile platformi. Ovo arhitektonsko rješenje se može primijeniti i na druge aplikacije osim ArcPad sustava, a koje omogućavaju proširivanje svojih funkcionalnosti nekim programskim jezikom.

5.1.10 Funkcije sustava

Glavna funkcija sustava je praćenje životinja u pokretu telemetrijom. Funkcija se sastoji od podfunkcija koje omogućuju otvaranje karte, unos mjerenja telemetrijom, prikaz mjerenja na karti i tablično, korištenje mjerenja u algoritmu triangulacije, prikaz i spremanje rezultata triangulacije.

- **Otvaranje karte** – Korisnik može odabrati koju kartu želi otvoriti.
- **Unos mjerenja telemetrijom** – Unos podataka o poziciji mjerenja, smjeru najjačeg signala i dodatne informacije o osobi koja je vršila mjerenje, vremenu mjerenja te koju smo jedinku mjerili. Podaci se automatski spremaju u bazu podataka.
- **Prikaz mjerenja tablično i na karti** – Korisnik može dobiti uvid u mjerenja koje je obavio te ih na formi odabrati i prikazati na karti sa ili bez smjera iz kojeg je došao najjači signal. Funkcija služi da istraživač ima uvid gdje treba obaviti sljedeća mjerenja.
- **Algoritam triangulacije** – Funkcija koja omogućuje korisniku da na jednom mjestu ima podatke mjerenja i da radi određivanje položaja životinje na osnovu tih mjerenja.
- **Prikaz rezultata triangulacije** – Rezultati triangulacije se mogu prikazati tablično sa svim atributima iz pogleda baze podataka u poglavlju 5.1.1. ili grafički na karti gdje se vizualno može utvrditi položaj životinje i mogućnost pogrešne procjene.

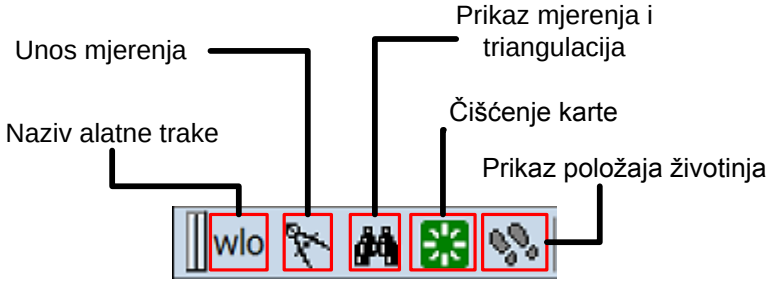
- **Spremanje rezultata triangulacije** – Korisnik može odlučiti da li želi pohraniti izračunati položaj životinje ili ne, ova mogućnost sprečava unošenje pogrešnih pozicija životinja u bazu podataka.

Nabrojane podfunkcije u potpunosti omogućuju praćenje životinja telemetrijom, a ovako izgledaju korisnička sučelja koje korisnik koristi. Sučelja su korisnički prijateljska i dizajnirana kako bi se što bolje uklopila u sučelje aplikacije ArcPad.

Slika 5-23: Pokretanje sustava i odabir karte

Pokretanje WloPad aplikacije počinje iz Start izbornika pokretanjem ArcPad 7.0.1. aplikacije na slici 5-23. Nakon toga se izabire karta s kojoj korisnik želi raditi. I pritisne se gumb OK.

Kada se otvori ArcPad i karta prikaže se i alatna traka koja je dio WloPad aplikacije.



Slika 5-24: Izgled alatne trake i popis funkcija

Alatna traka sadrži sve potrebne funkcionalnosti na jednom mjestu u obliku tipki. Traka se sastoji od tipki uz pomoć kojih korisnik pristupa ostalim komponentama te može unositi nova mjerenja, prikazivati podatke i određivati položaj životinje triangulacijom. Ako ima previše elemenata na karti može ih sa tipkom na slici 5-24 koja se zove „Čišćenje karte“ ukloniti. Pritiskom na „Unos mjerenja“ korisnik pokreće formu za unos novog telemetrijskog mjerenja.

Slika 5-25: Izgled forme za unos mjerenja telemetrijom i izgled mjerenja na karti

Na slici 5-25 vidimo obrazac za unos mjerenja i grafički prikaz unesenog mjerenja. Obrazac sadrži dodatna polja osim pozicije i smjera signala kako bi se korisniku kasnije omogućilo bolje upravljanje podacima. Unos se unosi pritiskom na tipku OK.
Objašnjenja polja za unos:

- **X,Y** – Koordinate pozicije mjerenja koje je korisnik obavio.
- **Bearing** – Unos smjera iz kojeg dolazi najjači signal izaženo u stupnjevima.
- **Animal ID** – Jedinstveni broj jednike koju istraživač prati.
- **Signal** – Kvaliteta signala (1-6).

- **Activity** – Aktivnost životinje za vrijeme mjerenja (pasive, active, unknown, mortality).
- **Signal Before** – Prijašnja mjerenja s te pozicije.
- **Date Time** – Vrijeme mjerenja.
- **Tracker** – Prezime i ime mjeritelja.

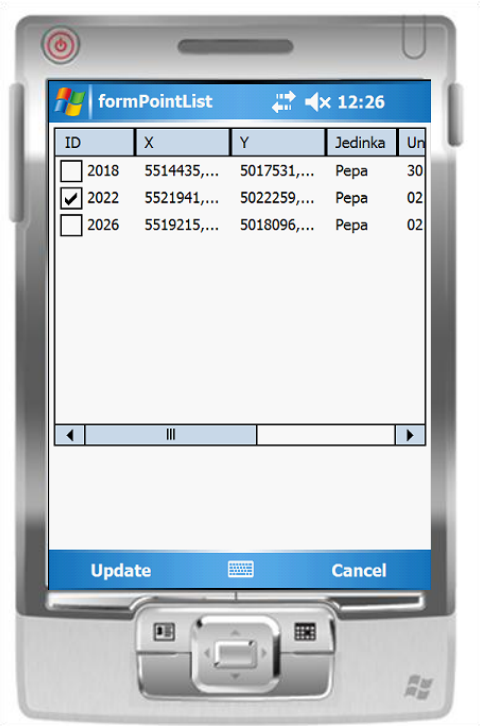
Istraživač prati životinju i izvršio je više mjerenja u kratkom roku, sada želi prikazati te podatke na karti i izračunati gdje se životinja nalazi. Koristiti će tipku „Prikaz mjerenja i triangulacija“.

Slika 5-26: Izgled forme za odabir mjerenja i prikaz rezultata triangulacije na karti

Na slici 5-26 prikazana je forma koja se prikaže na ekranu kada korisnik stisne tipku „Prikaz mjerenja i triangulacija“. Forma sadrži sva mjerenja koja su izmjerena i korisnik može gledajući vrijednosti atributa kao što su ime jedinke, vrijeme mjerenja, koordinate i dr. odabrati koja mjerenja želi prikazati na karti. Ako želi da se pokrene algoritam triangulacije onda mora odabrati opciju „Triangulate“ i ako želi da mu se iscrtaju samo točke bez pravaca koji označavaju smjerove radio signala onda mora potvrditi opciju „Show only points“. Pritiskom na tipku „Update“ pokreće se izračun pozicije životinje i iscrtavanje podataka na

karti. Rješenje vidimo na slici 5-26 gdje su iscrtana mjerenja i određena pozicija životinje te elipsa pogreške čija površina koja je zapisana u statusnoj traci daje objektivnu ocjenu da li je određena pozicija dovoljno realna i točna. Ako je korisnik zadovoljan točnošću izmjerenog treba pritisnuti u gornjem desnom uglu tipku sa kvačicom koja označava da je pozicija dobra i da se može spremiti u bazu podataka. U slučaju da mjerenje nije dobro pritiskom na tipku sa križićem sa karte će se očistiti samo objekti vezani za rezultat neuspješne triangulacije.

Ako korisnik želi može pritiskom na tipku „Prikaz položaja životinja“ vidjeti gdje su se životinje nalazile i kada te prikazati te položaje na karti.



Slika 5-27: Forma za prikaz izračunatih pozicija triangulacijom

5.3. Sinkronizacija i replikacija podataka

5.3.1. WloReplication

WloReplication je podsustav WLO sustava koji je zadužen za implementaciju replikacije između mobilnog korisnika koji vrši terensko istraživanje i centralnog sustava. Sustav se oslanja na spojnu replikaciju koja je podrobnije objašnjena u poglavlju „Metode i materijali“. WloReplication je zamišljen kao mobilna aplikacija budući da su mobilne aplikacije ergonomičnije u pogledu terenskog istraživanja, a još uvijek pružaju dovoljno velik opseg mogućnosti za korisnika što se tiče njegovog rada.

Slika 5-28: Glavna forma wloReplication komponente

Prije samog čina replikacije potrebno je pripremiti Izdavača kako bi mogao pružiti članke Pretplatnicima. Sam postupak pripremanja Izdavača biti će obrađen u narednom poglavlju. Nakon što Izdavač pripremi podatke za replikaciju potrebno je da se korisnik pretplati na određenog Izdavača od kojeg želi primati članke.

Sam čin pretplate na određenog Izdavača se vrši samo jednom, prilikom početnog postavljanja replikacije i potrebno je kako bi Pretplatnik omogućio mehanizmu replikacije informacije o Izdavaču i podacima koje želi preuzeti sa centralnog sustava.

WloReplication aplikacija ima dvije glavne opcije koje omogućuju repliciranje

podataka. Prva opcija koju korisnik mora koristiti je pretplata na Izdavača. Pri pokretanju aplikacije prikazat će se glavna forma prikazana na slici 5-28. Potrebno je odabrati opciju Subscribe.

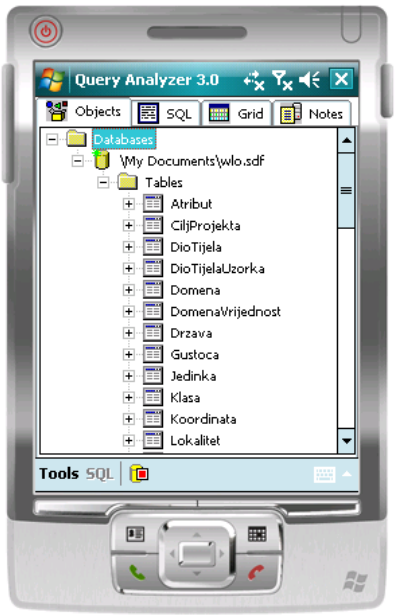
Slika 5-29: Forma za unos podataka o pretplati

Nakon što se odabere opcija Subscribe pojavi se forma prikazana na slici 5-29. Potrebno je ispuniti podatke koji su neophodni za uspješnu pretplatu na nekog Izdavača. Potrebni podaci su:

- **Device name** – Ime mobilnog uređaja koji će preuzeti ulogu Pretplatnika
- **Server name** – Ime poslužitelja koji će preuzeti ulogu Izdavača
- **IIS Folder Name** – Naziv mape koja je kreirana na poslužitelju Izdavaču te služi za pristup Poslužitelja prema Izdavaču u cilju sinkronizacije podataka. Detalji kreiranja mape će također biti objašnjeni naknadno.
- **Publication** – Naziv publikacije koja je kreirana na strani Izdavača. Predstavlja skup tablica koje se žele prenijeti na Pretplatnike radi pregleda, modifikacije ili dodavanja novih podataka.
- **Database** – Ime baze podataka nad kojom se vrši repliciranje podataka. Baza podataka na Izdavaču služi kao izvor podataka iz kojeg se onda vade objekti koji sudjeluju u stvaranju nove publikacije.

- **User** – Predstavlja korisničko ime administratorskog korisnika (sa) za pristup SQL Serveru budući da je prilikom implementacije wloReplication sustava odabran taj način autentifikacije korisnika.
- **Password** – Predstavlja lozinku administratorskog korisničkog imena koja je postavljena pri samoj instalaciji SQL Server komponente.

Nakon što su svi podaci pravilno ispunjeni, potrebno je odabrati opciju Subscribe kako bi se pokrenuo sam proces pretplate na Izdavača te je tako prvi korak replikacije uspješno obavljen. Potrebno je ponovno naglasiti da je ovaj korak potrebno raditi samo u slučaju da Pretplatnik nije već prethodno stvorio pretplatu na tog Izdavača, u ostalim slučajevima ovaj korak nije potreban.



Slika 5-30: Replicirana baza podataka

Nakon što se izvrši pretplata na Izdavača, na Pretplatnika se replicira baza podataka ili samo onaj dio baze za koji je stvorena publikacija. Na slici 5-30 prikazana je baza podataka koja se replicirala uz pomoć publikacije čiji naziv je upisan u formu za pretplatu na Izdavača. Početna publikacija je obuhvatila čitavu bazu podataka sa svim tablicama i drugim objektima koji se nalaze u njoj.

Kada se baza pripremi za replikaciju, sam proces replikacije napravi neke izmjene strukturom baze podataka i to u svrhu praćenja promjena nad njom. Tablicama baze dodijele se posebni atributi koji označavaju jedinstvenost zapisa i identificiraju ga što se dalje koristi

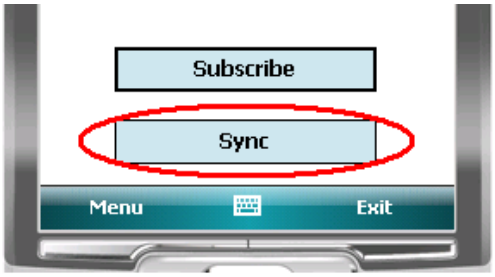
prilikom sinkronizacije podataka i mogućem rješavanju konflikta.

Baza koja se replicirala na mobilni uređaj ima istu strukturu kao i baza na centralnom sustavu, samo se razlikuje po formatu koji je prilagođen mobilnim uređajima. Također, postoje već implementirani Microsoft alati koji omogućavaju rad sa mobilnim bazama i njihov pregled.

Daljni rad nad bazom podataka i modificiranja njenih zapisa predviđen je za wloPad komponentu WLO sustava. WloReplication komponenta samo pripremi podatke za mobilni uređaj kako bi ga druga komponenta mogla koristiti, a da bi pritom bila očuvana konzistentnost podataka te mogućnost sinkronizacije sa glavnim sustavom.

Nakon modifikacija koje Pretplatnik učini nad bazom podataka tijekom terenskog istraživanja dolazi se do koraka koji i centralni dio replikacije. Podatke je sada potrebno sinkronizirati sa podacima na centralnom sustavu. Za sinkroniziranje podataka potrebno je pokrenuti wloReplication aplikaciju te odabrati drugu opciju, opciju Sync. Nakon odabira te opcije dolazi do sinkronizacije podataka između mobilnog uređaja i centralnog poslužitelja koji ima ulogu Izdavača. Sve promjene koje je Pretplatnik obavio na bazi podataka spojit će se sa centralnom bazom i mogućim promjenama koje su se u međuvremenu mogle obaviti na centralnom sustavu.

Svi konflikti do kojih je moglo doći, primjerice zbog promjene istih podataka i na poslužitelju i na mobilnom sustavu rješava sam proces replikacije sa svojim ugrađenim mehanizmima.



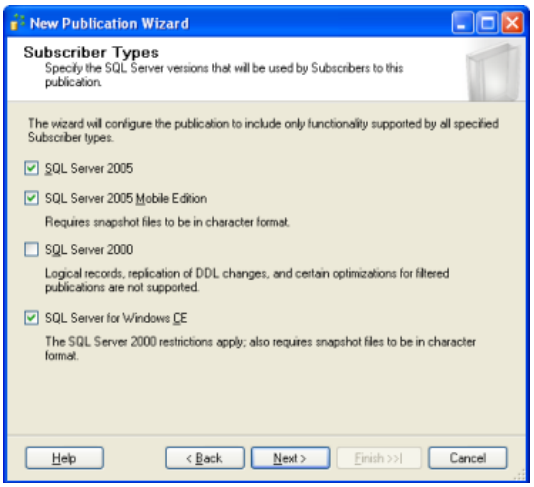
Slika 5-31: Opcija Sync

5.3.2. Postavljanje replikacije

Aplikacija WloReplication je razvijena u programskom jeziku C# koristeći Microsoft Visual Studio 2005 Professional Edition , a sama baza se nalazi na SQL Serveru 2005.
Na poslužiteljskom računalu instalirani su Microsoft Internet Information Services (IIS), Microsoft SQL Server 2005 Developer Edition i Microsoft SQL Server 2005 Mobile Server

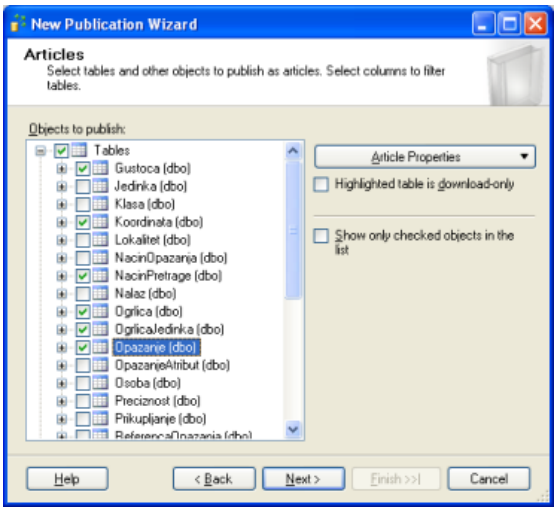
Tools. Da bi korisnici mogli pristupiti publikaciji na poslužitelju koristeći ručna računala treba podesiti NTFS ovlasti na virtualnoj mapi i mapi snimke stanja te IIS i Microsoft SQL Server 2000 Developer Edition.

Za stvaranje nove publikacije koristi se Publication Wizard kojeg otvorimo nakon pokretanja Microsoft SQL Server Management Studio-a. Nakon što smo odabrali na kojoj bazi želimo stvoriti publikaciju i odabrali spojnu replikaciju kao tip replikacije, potrebno je odabrati verziju SQL Servera koju koristi Pretplatnik.



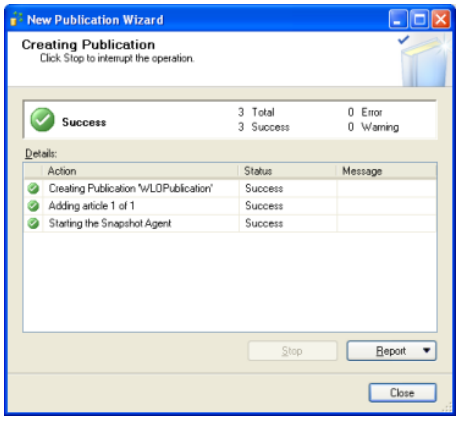
Slika 5-32: Odabir vrste Pretplatnika

Nakon toga, potrebno je definirati koje tablice želite obuhvatiti publikacijom, odnosno definirati podatke koji će se replicirati. Uvijek je preporučljivo odabrati srodne podatke za zasebnu publikaciju. Po potrebi se mogu dodati razni filteri ukoliko se želi selektirati podatke u ovisnosti o nekim njihovim atributima.



Slika 5-33: Filtriranje podataka

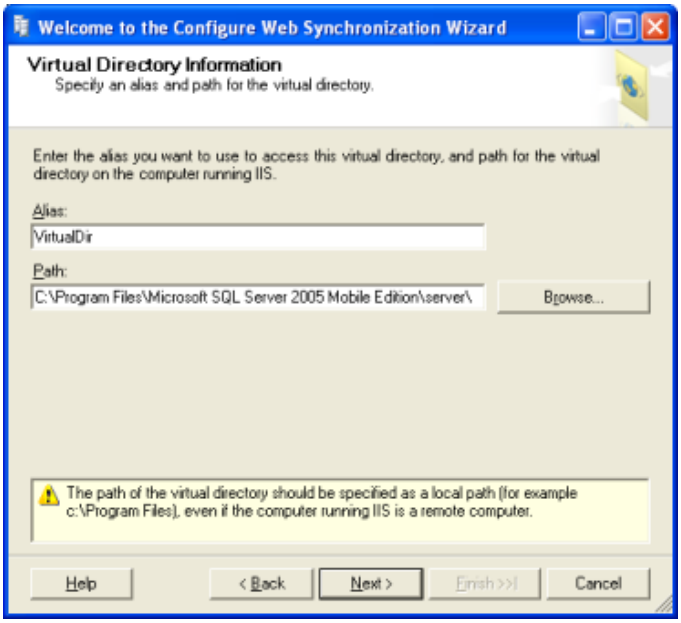
Zatim se definiraju korisničke postavke (korisničko ime , password) korisnika koji će koristiti publikaciju. Na kraju se upiše ime publikacije koju se želi kreirati i dobije se obavijest o uspješno stvorenoj publikaciji.



Slika 5-34: Uspješno kreirana publikacija

Nakon uspješnog stvaranja publikacije dodati sve korisnike koji će imati pristup publikaciji u listu pristupa publikaciji (engl. Publication Access List).

Za podešavanje IIS-a i NTFS ovlasti potrebno je instalirati Microsoft SQL Server 2005 Mobile Server Tools sa odabranom opcijom "Synchronize with SQL Server 2005". Staviti mapu snimke stanja (engl. Snapshot folder) na dijeljenje (engl. sharing) te dozvoliti korisniku IUSR_MACHINENAME čitanje mape. Za mapu mora biti uključeno i Web dijeljenje (engl. Web sharing). IUSR_MACHINENAME mora se nalaziti kao korisnik u postavkama sigurnosti (engl. Security tab). U IIS-u postaviti anonimni pristup virtualnoj mapi snimke stanja. Pokrenuti "Configure Web Synchronization Wizard" te za pretplatnika odabrati SQL Server Mobile Edition. Stvoriti novu virtualnu mapu u "Default Web Site" te joj dati alias ime kao što je prikazano slikom.

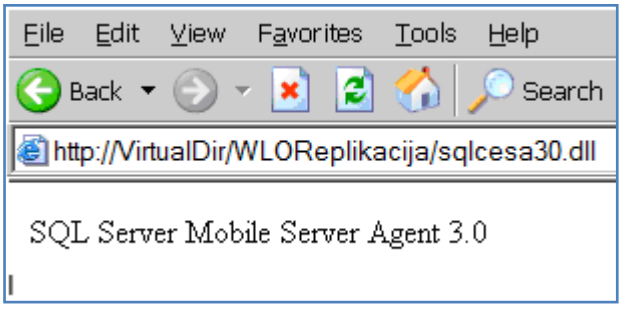


Slika 5-35: Kreiranje virtualnog direktorija

Prihvatiti stvaranje nove mape i kopiranja te registriranja SQL Mobile Server Agent-a. Klijenti će se spajati anonimno te je potrebno odabrati anonimno prijavljivanje za pristup virtualnoj mapi. Virtualna mapa će biti korištena za spojnu replikaciju pa je potrebno odabrati opciju "The virtual directory will be used for SQL Server merge replication with a UNC snapshot share.". Potrebno je upisati putanju snimke stanja u obliku \\MACHINENAME\SNAPSHOT. [16]

Nakon uspješne konfiguracije maknuti mapu snimke stanja sa dijeljenja i Web dijeljenja te ostaviti IUSR_MACHINENAME kao korisnika u postavkama sigurnosti. U IIS-u dozvoliti anonimni pristup novostvorenoj virtualnoj mapi a u Windows Explorer-u otvoriti novostvorenu mapu te pronaći datoteku __MACHINENAME_VIRTUALDIRECTORY i otvoriti ju. Promijeniti postojeći tekst u \\MACHINENAME\SNAPSHOT_PATH.

Otvaranjem Microsoft Internet Explorer-a i upisivanjem "http://MACHINENAME/VIRTUALDIRECTORY/sqlcesa30.dll" dobije se ekran prikazan slijedećom slikom:



Slika 5-36: Provjera uspješnosti postavljanja replikacije

6. Rasprava

6.1. Ostvarena modularnost sustava

Tijekom razvoja sustava jedan od postavljenih prioriteta koji je potrebno ostvariti bila je raspodjeljenost sustava. Svrha toga može se vidjeti u posebnosti obavljanja istraživačkog posla. Jedan dio posla odnosi se na terenska istraživanja prilikom kojeg se vrši prikupljanje podataka te izvršenje algoritma triangulacije u svrhu određivanja položaja divlje životinje. Drugi dio posla odnosi se na analizu prikupljenih podataka kako bi akademska zajednica mogla donositi zaključke na temelju prikupljenih podataka u cilju zaštite okoliša i životinja.

Wild Life Observer je sustav koji je omogućio povezanost ta dva načina rada. Povezanost je ostvarena na taj način da je svaki podsustav implementiran zasebno,jedan kao mobilna i drugi kao stolna aplikacija, a zatim je osmišljen način komuniciranja i sinkroniziranja podataka između te dvije komponente sustava.

Budući da je svaki podsustav razvijan zasebno možemo ih definirati kao jedinstvene module koji komuniciraju preko posebno dizajniranog načina komunikacije.

Znanstvenicima su najbitniji podaci tokom njihovog istraživačkog rada. Stoga su konzistentnost i jednoznačnost podataka bili nešto na što je posebno obraćena pažnja.

Podaci na dva odvojena modula imaju istu strukturu samo se nalaze u dva različita formata prilagođenih medijima na kojima se nalaze, čime je omogućeno jednostavno preslikavanje podataka sa jednog podsustava na drugi.

Prenošenje i sinkronizacija podataka između modula ostvareni su preko replikacije podataka čime se željela postići očuvanost podataka koji su neophodni za istraživački rad. Metodom replikacije omogućena je sinkronizacija podataka koji su prikupljeni na terenu sa onim već postojećim u centralnom sustavu ostvarenom kroz stolnu aplikaciju. Detalji ovog mehanizma opisani su u poglavlju „Spojna replikacija“.

6.2. Postignuta funkcionalnost i primjenjivost

Wild Life Observer predstavlja zaokruženu cjelinu koja pruža većinu funkcionalnosti koje su bile definirane pri začetku rada na projektu. Kroz razvoj sustava pojavili su se dodatni zahtjevi koji su u planu za implementaciju tijekom budućnosti.

Trenutno WLO sustav pruža znanstvenicima mogućnost prikupljanja podataka pomoću mobilne aplikacije koja služi kao potpora terenskom istraživanju. Kroz mobilnu aplikaciju ostvaren je jednostavan i interaktivan način unosa podataka koji su prikupljeni promatranjem divljih životinja. Unutar same mobilne aplikacije implementiran je i algoritam triangulacije uz pomoć kojeg je moguće odrediti poziciju divlje životinje na temelju signala koji odašilje ogrlica na životinji.

Kao što je ranije bilo napomenuto, podaci su od ključne važnosti za istraživački rad te je u tu svrhu osigurana sigurna i postojana pohrana podataka unutar sustava kako bi se znanstvenicima omogućila analiza tih prikupljenih zapisa o aktivnostima životinja.

Kroz implementaciju stolne aplikacije omogućeno je znanstvenicima da održavaju zapise i podatke o projektima kojima se bave, istraživanjima u kojima sudjeluju te svim popratnim aktivnostima koje su vezane uz njihov primarni rad. Drugi dio aplikacije implementiran je kako bi osigurao funkcionalnost analize prikupljenih podataka. Analiza podataka omogućena je uz pomoć interaktivne geografske karte koja je implementirana na taj način da omogućuje prikaz prikupljenih podataka kako bi se znanstvenicima na efikasan način predočili rezultati njihova rada. Uz to, omogućeno je izvođenje algoritma triangulacije na stolnoj aplikaciji te priprema za prikaz rezultata na karti.

Kako bi se omogućila što efikasnija analiza podataka unutar stolne aplikacije su ugrađeni mnogobrojni načini pretrage podataka kako bi znanstvenici mogli pretraživati podatke prema više parametara te na temelju toga donositi zaključke o aktivnostima životinja. Jedan od glavnih problema pri razvoju sustava je bila sinkronizacija podataka. Ova funkcionalnost je u potpunosti ostvarena i testirana čime je omogućena modularnost sustava i prikupljanje podataka iz više izvora što je neophodno za kvalitetan istraživački rad.

Budući da je sustav razvijan prema potrebama stvarne znanstvene zajednice njegova primjena je već jasno definirana. Sustav će nakon dodavanja dodatnih funkcionalnosti biti pušten u primjenu i služiti će se u postojećim,ali i budućim projektima znanstvenika koji se bave promatranjem divljih životinja u Hrvatskoj. Primjenjivost ovakvog sustava je ogromna budući da trenutno ne postoji sličan sustav koji bi se mogao koristiti. Do sada je znanstvenicima njihov istraživački rad bio bitno usporen i otežan budući da nisu imali

razvijen sustav koji bi služio kao potpora za istraživački rad. Izgradnjom ovog sustava omogućena je jedinstvena pohrana svih podataka, njihova analiza te upotreba u daljnjim istraživačkim aktivnostima.

6.3. Verifikacija rješenja temeljena na stvarnim podacima krajnjeg korisnika

U poglavlju 3.3. objasnili smo neke od glavnih problema postojeće infrastrukture korištene pri terenskim istraživanjima, koji su uglavnom vezani uz loše dizajniran model podataka. Kod razvoja novog modela podataka primarni je cilj bio razviti jedinstveni model koji će omogućiti jednostavnu integraciju različitih komponenti sustava te sinkronizaciju i konzistentnost podataka. Osim toga, bitno je da novi model podataka sadržajno odgovara postojećem modelu, odnosno da omogućuje pohranu istog skupa informacija.

Cjelokupni sustav do sada je testiran na naknadno prikupljenim podacima s terena dok su svi povijesni podaci i dalje pohranjeni u okviru starog modela podataka. Budući da je razvijen novi, poboljšani model podataka, slijedeća faza razvoja WLO sustava predstavljat će prebacivanje svih postojećih povijesnih podataka o terenskim istraživanjima iz postojećih baza podataka u novu jedinstvenu bazu podataka razvijenu u sklopu WLO sustava, te puštanje samog sustava u primjenu.

7. Zaključci

Kroz „WildLife Observer“ ostvarena je programska potpora za istraživanje životnih aktivnosti divljih životinja. Sustav predstavlja značajnu pomoć znanstvenicima budući da prije ovog sustava nije postojalo cjelovito rješenje koje bi služilo kao potpora za njihov rad. Upravo nepostojanje sličnog sustava je predstavljalo najveći izazov prilikom izgradnje WildLife Observera budući da su sve komponente morale biti izgrađene iz samog početka i bez mogućnosti usporedbe sa nekim drugim sustavom.

Tijekom razvoja, posebna pažnja posvetila se vjerodostojnom oblikovanju sustava koji odgovara primjeni u stvarnom znanstvenom radu te je ostvaren kontakt sa profesorom Veterinarskog fakulteta, dr. sc. Josipom Kuskom, kako bi se prikupili stvarni podaci proizašli iz promatranja divljih životinja. Pomoću njegovih zahtjevima izgrađen je vjerodostojan sustav na koji se znanstvenici mogu osloniti u svom budućem radu.

Cjelovitost WLO sustava vidi se i u mogućnostima višekorisničkog znanstvenog rada budući da je implementira podsustav za rad u laboratorijima te zasebno onaj koji služi kao potpora terenskom istraživanju. Još jedna od velikih prednosti Wild Life Observera je mogućnost sinkronizacije podataka tih dviju, naizgled zasebnih, komponenti sustava. Na taj način omogućeno je svim korisnicima da u svakom trenutku mogu pristupiti i koristiti najnovije prikupljene podatke. Tijekom terenskog istraživanja koristi se džepno računalo koje ne smanjuje mobilnost istraživača, a na kojem se izvršava podsustav za džepno računalo za kojeg je osmišljena specifična arhitektura koja omogućuje da putem GIS sustava istraživač u isto vrijeme prati svoje kretanje i pregledava i upisuje podatke o životinji. Pritom je bilo potrebno uložiti veliki trud prilikom dizajniranja arhitekture koja je primjenjiva i na druge sustave u budućnosti. Cijeli sustav podržava podršku za praćenje radiosignala iz ogrlice, a rezultati koji se dobivaju procesom triangulacije i kvantitativna mjera pogreške se temelje na standardiziranim i globalno prihvaćenim matematičkim modelima.

Budući da je „WildLife Observer“ projekt koji se razvio iz realnog problema i stvarne potrebe za takvim sustavom, postoji još nekoliko zahtjeva koji nisu u potpunosti implementirani do ove razvojne faze. No bitno je naglasiti da je već stvoren dobar temelj za nadogradnjom tih zahtjeva. Krajnji cilj je da projekt započne s primjenom i ostvari svoj puni potencijal u potpori očuvanju prirode i okoliša.

8. Zahvale

Posebno se zahvaljujemo našem mentoru prof.dr.sc. Krešimiru Fertalju na mnogobrojnim stručnim, praktičnim ali i životnim savjetima koji su bili od presudne važnosti za početak, razvoj i dovršetak WLO projekta. Tijekom cijelog razvojnog ciklusa projekta pokazao je ogroman angažman i uložio veliku količinu vremena i truda koji daleko nadilaze fakultetske obaveze mentora. Bez njegovog predanog angažmana ovaj rad nikada ne bi ugledao svjetlo dana te mu se stoga ovim putem od srca zahvaljujemo!

Zahvaljujemo se dr. Josipu Kusku s Veterinarskog fakulteta u Zagrebu, na stručnim i praktičnim savjetima tijekom cijelog procesa razvoja WLO sustava koji su bili od velike pomoći pri realizaciji samog projekta. Njegov angažman na projektu uvelike je doprinjeo tome da WLO sustav postane primjenjiv u praksi.

Zahvaljujemo se dipl.ing. Ivani Nižetić i mr.sc. Borisu Milašinoviću, asistentima Zavoda za primjenjeno računarstvo Fakulteta elektrotehnike i računarstva u Zagrebu, na velikodušnoj pomoći prilikom razvoja modela podataka i implementacije algoritma triangulacije.

Zahvaljujemo se kolegi studentu i prijatelju Anti Barišiću s Fakulteta elektrotehnike i računarstva na pomoći pri dizajniranju modela podataka i općenito na sudjelovanju u radu WLO tima.

HVALA!

WLO tim

9. Popis literature

[1] Lenth, R. On finding the source of a signal. Technometrics. Vol.23 No.2, 1981.

[2] Paradowski, L. Uncertainty Ellipses and Their Application to Interval Estimation of Emitter Position. IEEE transactions on aerospace and electronic systems. Vol. 33, No. 1, 1997.

[3] Jason Fuller, Calling managed code from native, 06.09.2005., *Calling managed code from native*, http://blogs.msdn.com/windowsmobile/archive/2005/09/06/461805.aspx, 05.04.2009.

[5] Radiotelemetry Triangulation Program, Travanj 2008. *Radiotelemetry Triangulation Program*, http://locateiii.com/, 07.06.2008.

[6] Fertalj, Krešimir; Milašinović, Boris; Nižetić, Ivana. Specifikacija sustava Cro-fauna, Državni zavod za zaštitu prirode, 2007. (elaborat)

[7] K. Fertalj, Razvoj primijenjene programske potpore – predavanja, Zagreb, FER, 2008.

[8] Bivariate Normal Distribution and Error Ellipses, 19.04.2009. http://www.scribd.com/doc/2341150/Lecture-5-Bivariate-ND-Error-Ellipses, 22.04.2009.

[9] Multivariate normal distribution, http://en.wikipedia.org/wiki/Multivariate_normal_distribution, *Multivariate normal distribution*, 7.04.2009.

[10] Désiré Luc Massart. *Handbook of chemometrics and qualimetrics*: *Hypotesis tests and confidence limits.* Elsevier, 1998.

[11] Knez M., *Općenita aplikacija za evidenciju na ručnom računalu*, diplomski rad, Fakultet elektrotehnike i računarstva, 2007.

[12] Ožura V., *Poslovna evidencija na ručnom računalu,* diplomski rad, Fakultet elektrotehnike i računarstva, 2006.

[13] Kopčak G., *Grafički prikaz geokodiranih informacija*, završni rad, Fakultet elektrotehnike i računarstva, 2008.

[14] Michaelis C., *MapWindow GIS ActiveX Control – Sample project: Data Visualisation Tool,* www.mapwindow.org/tutorials/MapWinGISActiveX%20DevelopmentTutorial.pdf , 13.4.2008.

[15] MapWindow GIS Overview,

http://emrg.usu.edu/software/mapwindow/MapWindowOverview.pdf, 10.2.2008.

[16] Programming Merge Replication with the Microsoft .NET Compact Framework, http://msdn.microsoft.com/en-us/library/aa446539.aspx, 12.2.2008.

[17] Darleen Sadoski, Santiago Cormella-Dorda, *Three Tier Software Arhitectures*, 16.2.2000., http://www.sei.cmu.edu/str/descriptions/threetier.html, 10.5.2008.

[18] Brian Mains, *Introduction to 3-tier arhitecture*, 28.4.2008., http://dotnetslackers.com/articles/net/IntroductionTo3TierArchitecture.aspx, 10.5.2008.

[19] *Multitier arhitecture, http://en.wikipedia.org/wiki/Multitier_architecture,* 12.5.2008.

[20] Grant, E. *On three-layer arhitectures,* Jet Propulsion Laboratory - California Institute of Technology

[21] Karim Hyatt, What is N-tier arhitecture?, *N-tier Application Development with Microsoft.NET,* 13.5.2008.

[22] Kusak J., Desnica S., Štrbenac A., *Akcija praćenja populacija velikih zvijeri po tragovima u snijegu – izvješće o rezultatima*, Zagreb, lipanj, 2007.g.

[23] Katanić N., *Programska potpora terenskom istraživanju*, završni rad, Fakultet elektrotehnike i računarstva, 2008.

10. Sažetak

Wild Life Observer (WLO) je moderan programski sustav za praćenje divljih životinja i njihovih aktivnosti. Ovaj sustav omogućuje znanstvenicima lakše i učinkovitije praćenje životinja u prirodi te razumijevanje njihovih životnih procesa kroz prikupljanje, analizu i obradu podataka. Na temelju tih podataka pruža se mogućnost donošenja relevantnih zaključaka o životnim procesima divljih životinja koje se promatraju prilikom terenskog i znanstvenog istraživanja.

U sklopu rada proveli smo analizu postojeće infrastrukture korištene za istraživanje hrvatske faune te zaključili da je takva infrastruktura neprikladna i nepotpuna za takvu vrstu istraživanja. Štoviše, za većinu znanstvenih potreba takva infrastruktura zapravo ne postoji. Stoga ovaj rad predstavlja projektno zasnovano, računalom podržano rješenje za praćenje divljih životinja.

Programski sustav sastoji se od dva glavna podsustava prilagođenih različitim načinima rada: podsustava za džepno računalo i podsustava za stolno računalo.

Podsustav za džepno računalo pruža podršku za praćenje lokacije životinja putem radiotelemetrije. Proces praćenja oslanja se na VHF radio ogrlice na životinji, te antene uz pomoć kojih se vrše mjerenja radio signala s različitih pozicija. Nakon izvršenih mjerenja, korisnik može jednostavno locirati životinju korištenjem programski ugrađenog matematičkog algoritma triangulacije. Ovaj algoritam na osnovu izvršenih mjerenja određuje lokaciju životinje i pogrešku mjerenja, a rezultat se grafički prikazuje na karti i pohranjuje u mobilnu bazu podataka. Prilikom razvoja ovog podsustava uspjeli smo uspostaviti interoperabilnost između izvornog i upravljanog koda, koja inače nije podržana standardnim razvojnim bibliotekama na džepnom računalu. Rješenje smo ugradili temeljem dizajna arhitekture koji omogućuje povezivanje GIS korisničkog sučelja s aplikacijskom logikom i mobilnom bazom podataka.

Podsustav za stolno računalo je višeslojna aplikacija koja služi za pregled, obradu i analizu svih podataka prikupljenih na terenu, s pomoću mobilnog računala ili ručno. Omogućena je evidencija podataka o projektima, terenskim istraživanjima, sudionicima istraživanja, jedinkama, opažanjima na terenu itd. Lokacije dobivene radiotelemetrijom, odnosno triangulacijom na terenu, moguće je analizirati s pomoću GIS modula stolne aplikacije. Ovaj modul obuhvaća grafički prikaz geokodiranih informacija o mjerenjima te lokacija dobivenih temeljem tih mjerenja.

Između podsustava za džepno računalo i podsustava za stolno računalo uspostavljen je mehanizam spojne replikacije koji omogućuje efikasnu sinkronizaciju podataka. Na taj način osigurana je konzistentnost podataka cjelokupnog sustava koja je temelj za kolaboraciju između istraživača na terenu i znanstvenika u laboratoriju.

Ključne riječi: divlje životinje, praćenje, programska potpora, mobilna aplikacija, spojna replikacija, geografski informacijski sustav, višeslojna aplikacija

11. Summary

Wild Life Observer (WLO) is a modern software system aimed on tracking wild animals and observing their activities. For scientists, this system provides an easy and more efficient way to track animals in nature and to understand their life processes through data gathering, processing and analysis. This gives you the ability to make relevant data-based conclusions about wild animal life processes monitored within the limits of field and scientific research.

In this work, we have performed analysis of an existing infrastructure used for croatian fauna research and found such infrastructure as unsuitable and incomplete for this kind of research. Moreover, for the most of the scientific needs such infrastructure doesn't even exist. Therefore, this work represents a project-founded computer-aided solution for wild animal tracking.

Software system is composed of two main subsystems suitable to different work modes: mobile device subsystem and desktop computer subsystem.

Mobile device subsystem provides the support for tracking the location of wild animals, based on radiotelemetry method. Tracking process relies on VHF radio necklaces worn by animals, as well as on radio antennas used for radio signal measurement in different locations. Upon the measurements are performed, user can simply locate the animal using the implemented mathematical triangulation algorithm. Triangulation algorithm calculates the animal's position and gives us the estimated deviation of this position compared to actual position of an animal, using the performed measurements as a ground basis for calculation. Given result, that is calculated position, is graphically represented to user on a map and stored in mobile database. During the development process of this subsystem, we managed to establish the interoperability between the native and managed code which is usually not supported by standard mobile device frameworks. We implemented this solution using such architectural design that provides us the ability to interconnect the GIS user interface with the application logic and mobile database.

Desktop computer subsystem is a multitier application used for editing, processing and analyzing the data collected within the field research, using the mobile device or by hand. User has the ability to register all relevant data about projects, field researches, research participants, animals, observations on the field etc. Furthermore, using the application GIS module, user can analyze the radiotelemetry data: signal measurements and animal locations triangulated in the field. This module also provides the graphical representation of geocoded

data on the map.

We managed to set up a merge replication mechanism between all mobile device subsystems and desktop computer subsystems, which provide an efficient data synchronization mechanism. This mechanism is a basis for data consistency within the boundaries of the system as a whole, as well as for an easy collaboration between researchers in the field and scientists in the laboratory.

Keywords: wils animals, tracking, software support, mobile application, merge replication, geographic information system, multilayered application

12. Životopisi autora

Vedran Ivanac

Rođen je 7. listopada 1986. godine u Zagrebu. Godine 2001. upisao je srednju tehničku školu „Ruđer Bošković“, smjer Računarstvo. Maturirao je s odličnim uspjehom 2005. godine. Iste godine upisuje Fakultet elektrotehnike i računarstva u Zagrebu. Godine 2008. završava dodiplomski studij na Fakultetu elektrotehnike i računarstva, smjer Računarstvo sa odličnim uspjehom. Trenutno je student prve godine diplomskog studija smjer Programsko inženjerstvo i informacijski sustavi. Bavi se svime što je vezano za programiranje i inovacije.

Nenad Katanić

Rođen 16.4.1986. godine u Slavonskom Brodu. Pohađao prirodoslovno-matematičku gimnaziju „Matija Mesić“ završivši sve razrede s odličnim uspjehom. 2005. godine upisao Fakultet elektrotehnike i računarstva u Zagrebu. 2008. godine završio preddiplomski studij na Fakultetu elektrotehnike i računarstva, smjer Računarstvo, modul Programsko inženjerstvo i informacijski sustavi, s odličnim uspjehom, te stekao titulu sveučilišnog prvostupnika inženjera računarstva (univ.bacc.ing.comp.). Iste godine, upisao Diplomski studij na Fakultetu elektrotehnike i računarstva, smjer Računarstvo, modul Programsko inženjerstvo i informacijski sustavi. Od 2009. godine stipendist kompanije Ericsson Nikola Tesla.

Goran Kopčak

Rođen 14. rujna 1986. godine u Slavonskom Brodu. Pohađao prirodoslovno-matematičku gimnaziju „Matija Mesić“ te maturirao sa odličnim uspjehom i izabran za učenika sa pravom izravnog upisa na Fakultet elektrotehnike i računarstva od strane učiteljskog vijeća. Završio preddiplomski studij na FER-u, modul Računalno inženjerstvo 2008. godine. Iste godine upisuje diplomski studij na istoimenom fakultetu, profil Programsko inženjerstvo i informacijski sustavi. Od 2008. godine stipendist kompanije „Ericsson Nikola Tesla“.