Rakenduse koostajad:
Marek Kirillov
Rando Laisaar
Siim Treilmann
Tiit Tallermaa

Meeskond „Leffe“
Jarmo Poolak
jpoolak@itcollege.ee
5. jaanuar 2013
Tallinn

Taksobaasi lõpptoote retsensioon ehk meenutusi veebiprogrammeerimisest

Esimene ettekujutus Taksobaasist oli, et tegu on tarkvaraga, mis on mõeldud taksodele või dispetšeritele. Peale lühikest tutvumist sai selgeks, et tegelikult on tegu üldiselt kõikki taksosid ja -juhtide ning nende litsentse hõlmava andmepangaga. Praegusel juhul jääb arusaamatuks, kellel täpselt on vajadus selliste andmete järgi. Minuteada taksode järgi teostab Tallinnas kontrolli MUPO ehk siis nemad oleksid potensiaalsed kliendid.

Rakenduse graafiline pool jätab väga hea mulje. Vormid ja tabelid on hästi läbi mõeldud hõlpsasti kasutatavad.

Mõned kasutatavuse puudused (usability)

- Puuduvad lühendite seletused nagu näiteks VL Omanik, VL Nr. SK Nr. jne;
- Inimese andmeid ei saa taaskasutada vaid tuleb iga kord käsitsi uuesti sisestada;
- Andete taaskasutamise võimaluse puudumine käib ka auto margi, mudeli ja ettevõtte kohta;
- Kuna andmeid sisestamisel ei saa taaskasutada, siis on vormil ka veidi palju välju korraga täitmiseks;
- Kohati "Sulge" nupp sulgeb akna, kord väljub hoopis programmist. Need nupud võiksid olla eraldi tähistatud;
- Põhiaken pole liigutatav ega lohistatav, vaid on naelutatud akna keskele. Õnneks laseb ennast minimiseerida;
- Login ja kasutajate halduse aken on alati pealmine (always-on-top), mis häirib muud tööd;
- Vilkuv otsingunupp hakkab lõpuks häirima.

Mõned programmilised vead (bugs)

Olulisi vigu programmi käitumises ei leidnud, vaid mõned kosmeetilised.

- Andmete sisestamisel tabulaatoriga liikumine ei tööta alati korrektselt;
- Andmete järjestamine tulpade järgi ei tööta alati eelduste kohaselt. Nt Auto margid järjestab: VW, BMW jne. Eeldaks, et BMW on eespool, kui VW

Mis on läinud hästi (well done)

- Hea kujundus ja ülesehitus
- Olemas on hädavajalik funktsionaalsus ja veidi rohkemgi
- Põhitegevused lihtsasti mõistetavad ja õpitavad
- Otsing töötab hästi
- Hea kasutajate haldus
- Logi pidamine kasutajate tegevustest

Andmebaas

Andmebaas koosneb kolmest tabelist: Andmed, Kasutajad ja Logi.

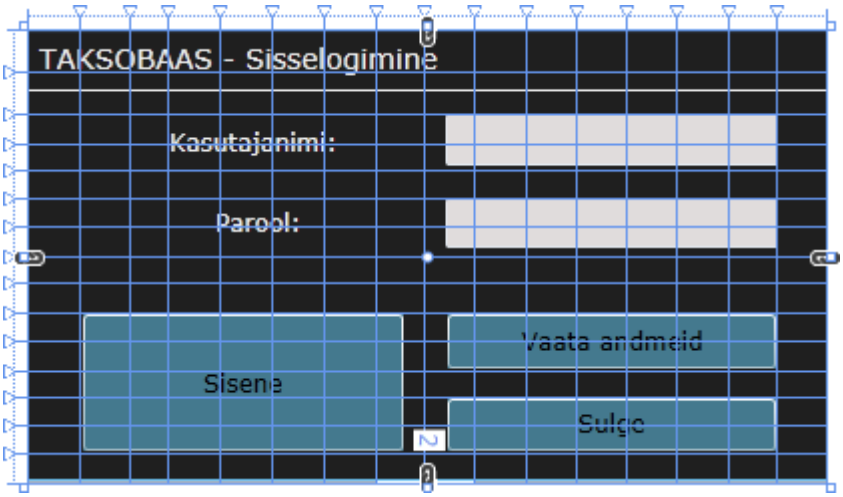
- Tabelid pole omavahel seotud ehk foreign keyd (FKd) puuduvad ehkki vähemalt Logi tabel võiks olla seotud tabeliga Kasutajad. Paraku on see seos loodud vaid lähtekoodis ja kasutades mitte ID-d vaid hoopis kasutaja nime (stringi tüüpi).
- Andmebaasi väljade nimed on veidi krüptilised: j_eesnimi, o_isikukood, VL_nr ja SK_nr. Nimetused võiksid olla pikalt välja kirjutatud, et ei peaks mõistatama millega tegemist. Tähemärgid ei maksa midagi, küll aga arendajate aeg, kes selle rakendusega kokku võivad puutuda.
- Andmebaas võiks olla normaliseeritud ehk ühest suurest „Andmed“ tabelist võiks saada mitu väikest: Autod, Isikud, Litsentsid, Ettevõtted ja võibolla ka Aadressid, kui nende järgi tekib nõudlus.

Lähtekood (code review)

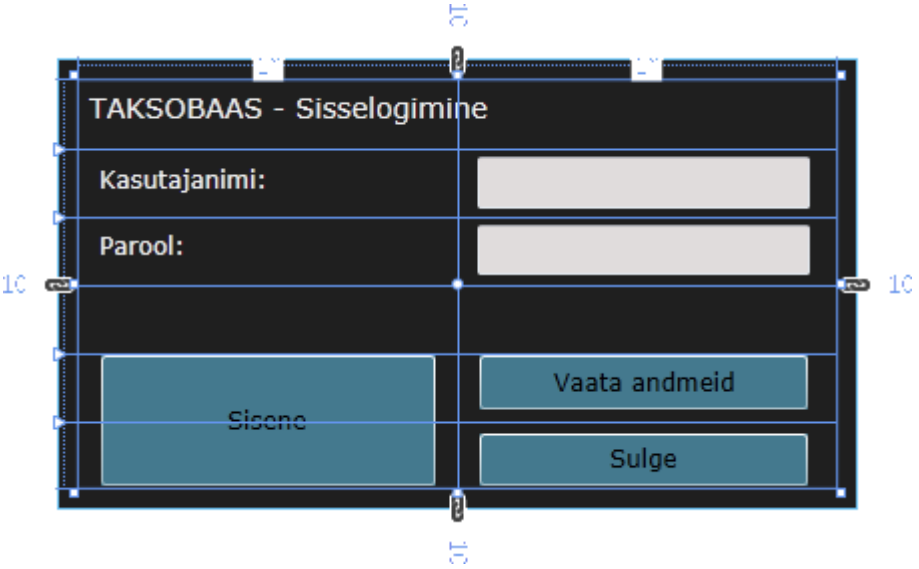
Veidi tähe- ehk koodinärimist.

- Visual Studio lahenduse (Solution) nimi on Taksobaas1.2, mis tähendab, et ei kasutata versioonihaldustarkava (revision control) nagu nt. GIT, SVN jt. See on väga oluline, kui projektis on mitu inimest, et kõigil oleks pidevalt värske kood. Samuti on see hea viis koodi muutumise jälgimiseks.
- Projekti failid on kenasti organiseeritud kahte kataloogi: Mudel ja Vaatemudel. Ülejäänud (põhiliselt XAML) failid on juurkataloogis.
- Faili- aga ka klassinimedes on kasutatud alakriipsu, nt. Logi_vaatamine.xaml. Targemad mehed soovitavad kirjutada sõnad suur tähega (CamelCase), nt. LogiVaatamine.xaml.
- Mõne faili- ja klassinimes kasutatakse esmapilgul arusaamatut lühendit "VM": "Kasutaja_haldusVM.cs". Selgub, et kuna see fail asub "Vaatemudel" kataloogis tähistab too VM, et tegu on vaatemudeliga. Niiviisi peaks ju järjepidevusest tingituna Krüpteerimine klass Mudel kataloogis tähistatama KrüpteerimineMudel.cs.
  Seega, tüübi või laadi kirjeldus failil ja klassinimes pole vajalik. Selle funktsiooni täitab .NET namespace, nt: "Taksobaas1._2.Vaatemudel.LogVaatamine".
- Kõik klassifaillid sisaldavad regioone (#region), mis tuleb koodi nägemiseks lahti klõpsutada. Tihti on regiooni sisuks vaid üks field või property rida. Klassi kiire ülevaate saamiseks on soovitatav regioone mitte kasutada.
- Meetodite nimed on tihti väikese tähega.
- Propertied võiks olla kirjutatud kasutades lühikest nn. „Auto property“ viisi:
  ```
  public string Eesnimi { get; set; }
  ```

- Sisselogimise akna (MainWindow.xaml) UI elementide paigutuseks on kasutatud vaid Gridi väga rohkete veergude ridadega:



Sarnandeb veebikujundusele tabeli abil. Lihtsam oleks asju paigutada, kui Gridis oleks vaid kaks veergu ning kuus rida. Tulemus Design views võiks olla selline:



- XAML koodis pole defineeritud ühtset stiili UI elementide kauppa eraldi failis vaid iga elemendi välimus on defineeritud üha uuesti ja uuesti. See muudab ühtse stiili muutmise võimatuks.
- Globaalmuutujad.cs kasutab sisselogimise viga tähistamiseks krüptilisi numbreid 1 ja 2. Õigem oleks kasutada Enumeid, millega saab defineerida võimalikud ja arusaadavad väärtused.
- Kasutaja andmete olekute meelespidamiseks mõeldud `Taksobaas1._2.Mudel.global` klassi asemel tuleks need olekud panna oma klassi(de) konteksti. Kasutaja vaheklass võiks olla selline:

```
public class Kasutaja
{
    public int Id { get; set; }
    public string Login { get; set; }
    public KasutajaTüüp Tüüp { get; set; }
    public string Eesnimi { get; set; }
    public string Perenimi { get; set; }
    public string Telefon { get; set; }
    public string Email { get; set; }
    public DateTime? BlokeeritudKuni { get; set; }
    public DateTime Lisatud { get; set; }

    public bool Kustutav()
    {
        return Login.ToLower() != "admin";
    }
}

public enum KasutajaTüüp
{
    Anonüümne = 0,
    Administraator = 1,
    Tavakasutaja = 2
}
```

See klass asendaks klassi `KasutajaKuvamine` ning seda tuleks kasutada ListBoxides kasutajate kuvamiseks.

- Kasutaja_haldus.xaml.cs võiks toimetada valitud ListView eelpoolmainitud Kasutaja objektiga, mille küljest saaks küsida andmebaasi rea ID:

```
private void Blokeeri(object sender, RoutedEventArgs e)
{
    Kasutaja kasutaja = lboxkasutajad.SelectedItem as Kasutaja;
    if (kasutaja == null)
        return;

    var blokeerimine = new Blokeerimine(kasutaja.Id);
    blokeerimine.Show();

    Close();
}
```

Sellisel viisil Blokeerimine klass saab toimetada omaette kasutaja ID-ga ja ei pea andmeid küsima väljaspoolt nn. globaalmuutujatest nagu PHP veebiprogrammeerimisel.

- Märkuseks, et C# klassidel pole (parameetriteta) konstruktori defineerimine kohustuslik.

Kokkuvõte

Vaatamata ehk mõnedele probleemsetele kohtadele nagu ebakonventsionaalne lähtekood ja andmebaasi normaliseerituse puudusele, programm ise töötab suuremate tõrgeteta, on hea välimusega ja on mugav kasutada ehk täidab oma ülesannet hästi.