IEGULDĪJUMS TAVĀ NĀKOTNĒ!

Eiropas Reģionālās attīstības fonds

Prioritāte: 2.1. Zinātne un inovācijas

Pasākums: 2.1.1. Zinātne, pētniecība un attīstība

Aktivitāte: 2.1.1.1. Atbalsts zinātnei un pētniecībai

Projekts: ”Multi - modeļu izstrādes tehnoloģija .NET pielietojumu projektiem”

Projekta sākuma datums: 2014.gada 1.janvāris.

Projekta beigu datums: 2015.gada 30.jūnijs.

Līguma Nr. 2013/0031/2DP/2.1.1.1.0/13/APIA/VIAA/010

ESF finansējuma saņēmējs: SIA, SWH SETS

Sadarbības partneris: Elektronikas un datorzinātņu institūts (EDI)

Projekta aktivitātes Nr. 3.9.2 "MVC koncepta iestrāde daudz modeļu vidē" progresa pārskats

Pārskats Nr. 34 par periodu no 2015.gada 1.janvāra līdz 2015.gada 30.jūnijam.

SATURS

1. Kopsavilkums

Pārskata periodā (2015-01-01 – 2015-06-30) projekta „Multi - modeļu izstrādes tehnoloģija .NET pielietojumu projektiem” aktivitātes "MVC koncepta iestrāde daudz modeļu vidē" ietvaros veikti šādi darbi:

1. MVC tehnoloģiju izpēte.
2. MVC aplikāciju izstrādes izpēte daudzmodeļu vidē, kas ietver saskarnes kontroļu arhitektūras principu izstrādi, kā arī saskarnes kontroļu ģenerācijas izstrādi.
3. Alternatīvās pieejas lietotāju saskarnes uzbūvē izpēti, kas ietver Angular JS, Backbone un Marionette izpēti.
4. Aktivitātes pētnieciskā darbība apspriesta ik nedēļas projekta semināros.

2. Ievads

Microsoft .NET izstrādes vide piedāvā infrastruktūru vairākiem programmatūras realizācijas tipiem. Viens no tiem ir MVC tehnoloģija, kas ir paredzēta web-aplikāciju izstrādei.

Šīs aktivitātes mērķis ir izskatīt izstrādājamās sistēmas iespējamo realizāciju daudzmodeļu vidē ar MVC tehnoloģijas palīdzību - tai skaitā pārbaudīt, vai izstrādātie tehnoloģiskie modeļi ir pietiekami, lai saturētu detaļas modeļa specifisko objektu implementācijai.

3. MVC tehnoloģija

3.1. Web-aplikācija

Web-aplikācija ir klienta-servera tipa aplikācija, kuras klienta daļu izpilda pārlūkprogramma. Šī prasība uzliek stingrus ierobežojumus gan klienta, gan servera daļu uzbūvei. Klienta daļa ir web-lapas, kas satur informāciju, norādes uz citām lapām vai formas, ko aizpilda lietotājs. Jebkura darbība, ko piedāvā lietotāja saskarne, ir pieprasījums, ko pārlūkprogramma nosūta uz servera, saņemot kā atbildi jaunu web-lapu. Attiecīgi, servera daļa apkalpo pieprasījumus no pārlūkprogrammām un atbildes noformē kā web-lapas.
Tādējādi sanāk, ka iespējas iebūvēt klientā uzvedību, kas būtu neatkarīga no servera, ir ļoti ierobežotas. Tomēr pastāv dažāda veida paplašinājumi, kas to ļauj darīt:

- **JavaScript** - speciālā programmēšanas valoda, kuras konstrukcijas spēj interpetēt pārlūkprogrammas. Tā tieši ir radīta, lai ļautu iebūvēt web-lapās darbošanās, kas neiesaista servera daļu. Piemēram, tā ļauj veikt elementāru aizpildāmās formas datu verifikāciju un tamlīdzīgas lietas.
- **JQuery** - virsbūve virs JavaScript, kas ļauj ērtāk strādāt ar lapas objektiem.
- **Ajax** - (Asynchronous JavaScript and XML) tehnoloģija[1], kas ļauj nosūtīt uz serveri asinhronus pieprasījumus (ļaujot lietotājam tikmēr strādāt tālāk ar redzamo lapu) un kā atbildes saņemt (un attiecīgi mainīt lietotāja saskarnē) tikai daļas no web-lapas. Tas ļoti samazina datu plūsmu starp klientu un serveri un padara pašu lietotāja saskarni "gludāku".

3.2. MVC tehnoloģija

MVC (Model-View-Controller) ir Microsoft ieviestā web-aplikāciju arhitektūras tehnoloģija[2], kas apraksta lietotāja saskarnes līmeņa uzbūvi. Tā paredz, ka sistēmas arhitektūrā tiks stingri nodalīti dati (Model), to prezentācija (View) un funkcionālā daļa (Controller), kas realizē aplikācijas biznesa loģiku, tai skaitā - lietotāja saskarnes daļu sasaisti:

- **Model** - modelis, MVC kontekstā - datu objekti, ar kuriem strādā izstrādājamās aplikācijas biznesa loģika.
- **View** - skats, MVC konekstā - web-lapa vai lapas daļa, kas satur lietotāja saskarnes elementus un domāta datu objektu attēlošanai.
- **Controller** - kontrolieris, MVC kontekstā - saņem pieprasījumus no klienta daļas, apstrādā datus, saformē datu objektus un izvēlas/saliek web-lapu, ko nosūtīs uz klienta daļu kā atbildi.

Microsoft Visual Studio piedāvā ietvaru, kas ir izstrādāts speciāli, lai palīdzētu stingrāk kontrolēt MVC arhitektūras principus.

4. MVC aplikācijas izstrāde daudzmodeļu vidē

Izstrādātie tehnoloģiskie modeļi, kas apraksta datu objektus, sagatavo datu plūsmu, kas tiek novesta līdz lietotāja saskarnes līmenim. Tagad ir jāsaprot, kādā veidā organizēt šo līmeni, lai tas vienlaikus atbilstu MVC principiem un izmantotu ģenerāciju no skatu [3] un darba vietas [4] tehnoloģiskiem modeļiem.

4.1. Saskarnes kontroļu arhitektūras principi

Piedāvātā lietotāja saskarnes arhitektūra ir sekojošā.
Lietotāja saskarne sastāv no lapām, katra no kurām sastāv no saskarnes kontroļiem. Tie varētu būt manuāli izstrādātie vai noģenerētie no modeļa. Katrs saskarnes kontrolis sastāv no trim daļām: modeļa, skata un kontroliera (MVC nozīmē). Ieviešam dažus noteikumus:

- Jebkurš saskarnes kontroļa datu objekts (modelis MVC nozīmē) ir objektu kopa ar atslēgām.
- Jebkurš saskarnes kontroļa skats ir lapas sagatave, kas rēķinās ar to, ka tā datu objektā būs elementi zem konkrētām atslēgām, un operēs ar tām.
- Jebkurš saskarnes kontroļa kontrolieris satur funkciju *makeModel* (ar parametriem, kas ir atkarīgi no saskarnes kontroļa uzbūves un uzdevuma), kas uztaisa šim saskarnes kontrolim atbilstošo modeli.
- Saskarnes kontroļa skats un kontroliera klase saucās vienādi.

Katrs saskarnes kontrolis pats nodrošina kaut kādu informāciju un izskatu, kā arī iekļauj apakškontroļus zem kaut kādām rolēm (piemēram, biznesa objektu tipa pārvaldības forma varētu iekļaut šī tipa objektu tabulu). Darbības šablons ir tāds:

1. galvenā kontroļa kontrolieris (tas, kurš pieņēma lietotāju pieprasījumu) izsauc savu funkciju *makeModel*, kas savāc viņam nepieciešamo informāciju, ka arī izsauc apakškontroļu *makeModel* funkcijas un pieglabā rezultātus zem kaut kādām atslēgām savā modelī;
2. aizpildot savu modeli, tas sagatavo atbildi: pielieto sev atbilstošajam skatam sagatavoto modeli;
3. skatā, savukārt, apakšskati tiek iekļauti kā daļējie skati (*Partial*).

Šī pieeja ļauj dinamiski sastādīt iekļaujamās daļas un iekapsulēt katra saskarnes kontroļa funkcionalitāti.

4.2. Saskarnes kontroļu ģenerācija

Skatu tehnoloģiskais modelis[3] satur saskarnes kontroļus (*GuiControl*), kas varētu būt nepieciešami. Tie ir apvienoti tipu hierarhijā (*GuiControlType*). Līdz ar to, ģenerācijai ir jāuztaisa no tehnoloģiskā modeļa datiem:

- kontrolieru hierarhiju atbilstoši *GuiControlType*, kur pati augstākā virsklase ir *Controller* (no Microsoft bibliotēkas, kas tiek izmantota MVC aplikācijā),
- skatu sagataves atbilstoši nodefinētiem *GuiControl*,
- kontrolieru klases atbilstoši nodefinētiem *GuiControl*, kas manto atbilstošajai *GuiControlType* klasei.

Piemērs ģenerācijai ir pielikumā 1. un 2.

Darba vietas tehnoloģiskais modelis uzdod augšējās lapas izskatu, bet ģenerācija ir līdzīga: uztaisa atbilstošo skatu un kontrolieri, kas satur biznesa funkciju izsaukumus un rezultātu vietu apzīmējumus.

5. Alternatīvās pieejas lietotāja saskarnes uzbūvē.

Izskatītā MVC aplikācijas arhitektūra veido web-aplikācijas klienta daļu, kas ir stingri atkarīga no servera: jebkura lietotāja darbība klienta galā prasa sadarbību ar serveri. Šo web-aplikācijas klienta veidu sauc par "plāno" klientu: tas izraisa lielu komunikācijas plūsmu ar serveri, bet pats klients ir vienkāršs un neliels pēc apjoma.

Var būvēt klientu otrādi (to sauc par "bagātu" klientu): izmantojot JavaScript/JQuery un Ajax tehnoloģijas, klienta daļa tiek padarīta maksimāli patstāvīga un neatkarīga no servera, līdz pat tam, ka klients reāli prasa tikai biznesa objektu datus. Tas ļauj minimizēt komunikācijas plūsmu ar serveri, bet klients šādā gadījumā ir sarežģīts un liels. Tik sarežģīts un tik liels, ka viņā pašā ir jāizstrādā sava arhitektūra.

Šim gadījumam ir izstrādātas vairākas tehnoloģijas un ietvari, kas ļauj būvēt šo "bagātu" klientu MVC tehnoloģijā. Respektīvi, paša klienta JavaScript/JQuery kods arī tiek uzturēts modeļu-skatu-kontrolieru sadalījumā.

Projekta ietvaros tika izpētīti daži šādi JQuery papildinājumi-ietvari.

5.1. Angular JS

Speciāls JQuery papildinājums [5]. Centrālie jēdzieni ir: *controller* (kontroliers) un *scope*.
Kontrolieris ir JQuery funkcija, kurā ir aprakstīts veids, kā pārvaldīt vienu bloku ielādētā lapā.
Scope - ir dati (vai modelis, vai mainīgo telpa) vienā kontrolierī. Tajā var definēt mainīgos un funkcijas. Tos var izmantot blokos, lai parādītu lapā. Izmaiņas *scope* mainīgajos uzreiz attēlojas blokā, nekādas īpašas darbības nav jāveic.
Ir spēkā daži likumi:

- Kontrolieri ir neatkarīgi: nav iespējams vienam nolasīt kaut ko no cita *scope*'a, vai izpildīt funkciju.
- Vienu kontroliera funkciju var pielietot vairākiem blokiem (rodas vairāki eksemplāri)
- *Scope*'i var pārklāties, ja bloks ar vienu kontrolieri satur bloku ar citu kontrolieri (tad ārējais skaitās vecāks, un iekšējais manto ārējā *scope*. Ja grib, lai mainīgie nepārklātos, tie ir jāinicializē kontroliera ķermenī - tad katram būs savs neatkarīgs komplekts).
- Ir pieejami papildus līdzekļi:
    - *service* - globāli redzami objekti (viens eksemplārs ir redzams visiem kontrolieriem - *data sharing*)
    - *factory* - viena funkcija-konstruktors, kas redzama visiem kontrolieriem (*code sharing*)

Iespējamais risinājums web-aplikācijas realizācijai ir sekojošs:

- Katrs *GuiControl* (GC) ir .cshtml ar
  - kontroliera funkcijas aprakstu
  - HTML bloku:

    * *
    * *
- Kontrolieru sadarbību organizē caur servici "*apiStore*" - globālo objektu kopu ar atslegām, kur katrs kontroliera eksemplārs noliek objektu (ar propertijām un funkcijām), kas ir pieejami no ārpuses. Atslēga ir - kontroliera eksemplāra nosaukums. Ja vajag izveidot notikumus, tad tos veido visaugstākā līmenī un ar nosaukumu, atvasināto no kontroliera eksemplāra nosaukuma.
- Lai nosaukumi būtu unikāli (bet pazīstami), katrs kontrolis saņem nosaukumu kā .cshtml modeli un pielieto *div* blokā kā inicializācijas parametru.
- Ja kontrolis satur apakškontroļus, tad tas saformē priekš viņiem nosaukumus no savējā plus apakšdaļas apzīmējums. Kontroli ieliek, iekļaujot *div* blokā ar *@Html.Partial*

5.2. Backbone

Speciāls JQuery papildinājums [6]. Centrālie jēdzieni ir *View* un *Model* (klienta pusē).
Model ir dati (loģiskajā nozīmē; tehniski - JQuery klase, kura var saturēt arī funkcijas). *Model*'im ir iebūvētā funkcija datu nolasīšanai no servera.
View - klase, kura, pielietojot datus, producē lapā rādāmu bloku (*DOM* apakškoku). Galvenie elementi:

- ***el*** - *View* "saknes" virsotne - *DOM*'a elements, zem kura *View* taisīs savu apakškoku. Jāiedod vai nu JQuery selektors, vai atlasītā *DOM* virsotne
- ***template*** - HTML paraugs, kurā ar speciālu sintaksi ir paredzēta vieta datiem; var dinamiski nolasīt no servera caur kontrolieri/akciju
- ***render*** - funkcija, kura no *template* un iedotā modeļa uztaisa *DOM* apakškoku un pieliek zem *el*; jāapraksta un jāizsauc manuāli.

Ir spēkā daži likumi:

- *View* definīcijas var mantot viena no otras.
- *View* var saturēt citus *View* eksemplārus: *render* funkcijai pa priekšu jāizveido HTML no savas *template*, kurā ar *id* atribūtu ir iezīmetas vietas citiem *View*. Tad jāsameklē jaunizveidotajā apakskokā šīs vietas un jāuztaisa attiecīgi *View* eksemplāri, piešķirot tiem šīs vietas kā saknes elementus. Un jāpalaiž to *render* funkcijas.

Iespējamais risinājums web-aplikācijas realizācijai ir sekojošs:

- Katrs vizuālais *GuiControl* (GC) ir
  - *View* ar tadu pašu vārdu, kurš satur arī dotā kontroļa API propertijas un funkcijas. Dati tiek piegādāti, izveidojot *Model*, piepildot ar datiem no servera (automātiski tiek izveidoti visi lauki)
  - *GC*.cshtml ar *template* + vienotā servera kontroliera akcija.

- Katrs funkcionālais kontrolis ir *Model* ar nodefinētu interfeisu un funkcijām (*Model* īpašības neizmanto, tas ir izvēlēts kā *Object* klases analogs)

5.3. Marionette

Speciāls *Backbone* papildinājums [7], kas ļauj automatizēt vairākas lietas no *Backbone* attiecībā uz *View*. Centrālie jēdzieni ir *View, Model* un *Region.*
Model ir dati (loģiskajā nozīmē; tehniski - JQuery klase, kura var saturēt arī funkcijas). *Model*'im ir iebūvētā funkcija datu nolasīšanai no servera.
View - klase, kura, pielietojot datus, producē lapā rādāmu gabalu (*DOM* apakškoku). Galvenie elementi:

- *template* - HTML paraugs, kurā ar speciālu sintaksi ir paredzēta vieta datiem; var dinamiski nolasīt no servera caur kontrolieri/akciju
- *render* - funkcija, kura no *template* un iedotā modeļa uztaisa *DOM* apakškoku un pieliek zem *el*. Izmanto dažādas *View* apakšklases ar iebūvētām funkcijām. Ir iespējams pārrakstīt.

Region - klase, kura apraksta *DOM* virsotnes, kas ir paredzētas *View* ievietošanai. Tie ir izmantoti, lai izsauktu *View render* funkcijas un izvietotu rezultātu. Reģionus var nodefinēt pašus vai *View* sastāvā.

Ir spēkā daži likumi:

- *View* definīcijas var mantot viena no otras.
- *View* var saturēt citus *View* eksemplārus kā sastāvdaļas. Priekš tā :
    - iekš *View template* iezīmē, kur likt apakš-*View*; *View* definīcijā aizpilda propertiju "*regions*"
    - pārraksta *render* funkciju: vispirms apstrādā savu *template* (ar iebūvēto funkciju), pēc tam uztaisa apakš-*View* eksemplāru un iedod reģionam

Iespējamais risinājums web-aplikācijas realizācijai ir tāds pats, kā Backbone gadījumā.

6. Secinājumi

Šīs aktivitātes ietvaros tika izstrādāta web-aplikācijas izstrādes arhitektūras koncepcija, kas ietver sevī gan MVC principus, gan izstrādāto tehnoloģisko modeļu izmantošanu ģenerācijā. Tika izpētītas alternatīvās pieejas lietotāja saskarnes uzbūves MVC tehnoloģijā un tām atbilstošās iespējas lietot saskarnes līmeņu tehnoloģiskos modeļus.

7. Literatūras saraksts

[1] AJAX Introduction (http://www.w3schools.com/ajax/ajax_intro.asp)
[2] ASP.NET MVC Overview (https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx)
[3] Nr. 3.9.1. "Skatu meta modelis"
[4] Nr. 3.9.3. "Darba vietas meta modelis"
[5] AngularJS (https://angularjs.org/)
[6] Backbone (http://backbonejs.org/)
[7] Marionette (http://marionettejs.com/)

8. Pielikumi

8.1. Piemērs skata sagatavei

"Index" tipa saskarnes kontroļu skatu sagataves ģenerācija

```
<#+
partial class GuiBase_Perfomer_GuiControl_index_MVC {
       public override void makeView(GuiObject me, Dictionary<string, object>
parms){
              var x=me as GuiControl;
              if (x.preparationKind != "index") return;
              makeView_FlatGui(x,parms);
       }
       protected void makeView_FlatGui(GuiControl x, Dictionary<string, object>
parms){
              var prefix="FlatGui";
              StartNewFile(prefix+x.name+".cshtml");
              var fields =
x.GuiFieldS.Where(o=>o.isHidden==false).OrderBy(o=>o.position).ToList();
#>
@using System.Collections.Generic;
@model MEDUSGUI.Models.M_index
@{
       var titles = Model.Titles;
       string q;
}
<h3>@Model.Message</h3>
<table cellpadding="3" border="1" cellspacing="4">
    @{
       <tr>
<#+
              foreach (var f in fields) {
#>
        @{
                if(titles.TryGetValue("<#= f.MDSvisualISN #>", out q)==false){
                  q="<#= f.name #>";
                     }
                }
              <th>@q</th>
<#+
         }
#>
       </tr>
    foreach (var item0 in Model.data)
    {
         var item=item0 as MEDUSBL.<#= x.basedOn.name #>;
    <tr>
<#+
              foreach (var f in fields) {
                     var fname = guiFieldName(f);
#>
              <td>
              @{
                     if (item.<#= fname #>==null) {
```

```
                        @:null
                    } else {
                        @Html.DisplayFor(modelItem=>item.<#= fname #>)
                    }
                }
                </td>
<#+
        }
#>
                <td>
                @{
                <form method="post" action="@Url.Action("actions","<#=
prefix+x.basedOn.name #>")">
<#+
                foreach (var c in x.basedOn.DVcolumnS) {
#>
                    @Html.HiddenFor(modelItem=>item.<#= c.name #>)
<#+
        }
 #>
                    <input type="submit" name="submitType" value="Details" />
                    <input type="submit" name="submitType" value="Edit" />
                </form>
                 }
                </td>
        </tr>
    }
  }
</table>

<#+
    }
}
#>
```

"Index" tipa saskarnes kontroļu noģenerētā skatu sagatave (*M_index*: speciāls interfeiss ar objektu kopu ar atslēgām priekš "Index" tipa skatiem)

```
@using System.Collections.Generic;
@model MEDUSGUI.Models.M_index
@{
        var titles = Model.Titles;
        string q;
}
<h3>@Model.Message</h3>
<table cellpadding="3" border="1" cellspacing="4">
    @{
        <tr>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.UserName", out
q)==false){
                    q="UserName";
                        }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.Email", out
q)==false){
```

```
                    q="Email";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.SecurityStamp", out
q)==false){
                    q="SecurityStamp";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.EmailConfirmed", out
q)==false){
                    q="EmailConfirmed";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.PasswordHash", out
q)==false){
                    q="PasswordHash";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.PhoneNumber", out
q)==false){
                    q="PhoneNumber";
                       }
                  }
                <th>@q</th>
@{

if(titles.TryGetValue("AspNetUsers_N_R_index.PhoneNumberConfirmed", out
q)==false){
                    q="PhoneNumberConfirmed";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.TwoFactorEnabled",
out q)==false){
                    q="TwoFactorEnabled";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.LockoutEndDateUtc",
out q)==false){
                    q="LockoutEndDateUtc";
                       }
                  }
                <th>@q</th>
@{
                  if(titles.TryGetValue("AspNetUsers_N_R_index.LockoutEnabled", out
q)==false){
                    q="LockoutEnabled";
```

```
                    }
                }
                <th>@q</th>
@{
                if(titles.TryGetValue("AspNetUsers_N_R_index.AccessFailedCount",
out q)==false){
                  q="AccessFailedCount";
                    }
                }
                <th>@q</th>
@{
                if(titles.TryGetValue("AspNetUsers_N_R_index.Id", out q)==false){
                  q="Id";
                    }
                }
                <th>@q</th>
@{

if(titles.TryGetValue("AspNetUsers_N_R_index.ForceToChangePassword", out
q)==false){
                  q="ForceToChangePassword";
                    }
                }
                <th>@q</th>
        </tr>
    foreach (var item0 in Model.data)
    {
         var item=item0 as MEDUSBL.AspNetUsers_N_R;
    <tr>
                <td>
                @{
                        if (item.UserName==null) {
                                @:null
                        } else {
                                @Html.DisplayFor(modelItem=>item.UserName)
                        }
                }
                </td>
                <td>
                @{
                        if (item.Email==null) {
                                @:null
                        } else {
                                @Html.DisplayFor(modelItem=>item.Email)
                        }
                }
                </td>
                <td>
                @{
                        if (item.SecurityStamp==null) {
                                @:null
                        } else {
                                @Html.DisplayFor(modelItem=>item.SecurityStamp)
                        }
                }
                </td>
                <td>
                @{
```

```
            if (item.EmailConfirmed==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.EmailConfirmed)
            }
      }
      </td>
      <td>
      @{
            if (item.PasswordHash==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.PasswordHash)
            }
      }
      </td>
      <td>
      @{
            if (item.PhoneNumber==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.PhoneNumber)
            }
      }
      </td>
      <td>
      @{
            if (item.PhoneNumberConfirmed==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.PhoneNumberConfirmed)
            }
      }
      </td>
      <td>
      @{
            if (item.TwoFactorEnabled==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.TwoFactorEnabled)
            }
      }
      </td>
      <td>
      @{
            if (item.LockoutEndDateUtc==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.LockoutEndDateUtc)
            }
      }
      </td>
      <td>
      @{
            if (item.LockoutEnabled==null) {
                  @:null
            } else {
                  @Html.DisplayFor(modelItem=>item.LockoutEnabled)
```

```
                }
            }
            </td>
            <td>
            @{
                if (item.AccessFailedCount==null) {
                    @:null
                } else {
                    @Html.DisplayFor(modelItem=>item.AccessFailedCount)
                }
            }
            </td>
            <td>
            @{
                if (item.Id==null) {
                    @:null
                } else {
                    @Html.DisplayFor(modelItem=>item.Id)
                }
            }
            </td>
            <td>
            @{
                if (item.ForceToChangePassword==null) {
                    @:null
                } else {
                    @Html.DisplayFor(modelItem=>item.ForceToChangePassword)
                }
            }
            </td>
            <td>
            @{
            <form method="post"
action="@Url.Action("actions","FlatGuiAspNetUsers_N_R")">
                @Html.HiddenFor(modelItem=>item.MDSrefName)
                @Html.HiddenFor(modelItem=>item.UserName)
                @Html.HiddenFor(modelItem=>item.Email)
                @Html.HiddenFor(modelItem=>item.SecurityStamp)
                @Html.HiddenFor(modelItem=>item.EmailConfirmed)
                @Html.HiddenFor(modelItem=>item.PasswordHash)
                @Html.HiddenFor(modelItem=>item.PhoneNumber)
                @Html.HiddenFor(modelItem=>item.PhoneNumberConfirmed)
                @Html.HiddenFor(modelItem=>item.TwoFactorEnabled)
                @Html.HiddenFor(modelItem=>item.LockoutEndDateUtc)
                @Html.HiddenFor(modelItem=>item.LockoutEnabled)
                @Html.HiddenFor(modelItem=>item.AccessFailedCount)
                @Html.HiddenFor(modelItem=>item.Id)
                @Html.HiddenFor(modelItem=>item.ForceToChangePassword)
                <input type="submit" name="submitType" value="Details" />
                <input type="submit" name="submitType" value="Edit" />
            </form>
             }
            </td>
        </tr>
    }
  }
</table>
```

8.2. Ģenerācija kontrolierim

"Index" tipa saskarnes kontroļu kontrolieru ģenerācija

```
<#+
partial class GuiBase_Perfomer_GuiControl_index_MVC {
  public override void makeCode(GuiObject me, Dictionary<string, object> parms){
        if (me.preparationKind != "index") return;
        var ctrl = me as GuiControl;
        makeCode_FlatGui(ctrl,parms);
  }
  protected void makeCode_FlatGui(GuiControl ctrl, Dictionary<string, object>
parms) {
        var fields =
ctrl.GuiFieldS.Where(o=>o.isHidden==false).OrderBy(o=>o.position).ToList();
        var dvn=ctrl.basedOn.name;
        var suff=dvn+"_index";
#>
public partial class FlatGui<#= suff #> : GuiControlClass_<#= suff #> {
  public override string dataViewName { get {  return "<#= dvn #>"; } }
  public override Dictionary<string, string> titles { get {
   return new Dictionary<string,string>() {
<#+
        var dictValues = fields.Select(o=>string.Format("{{\"{0}\",\"{1}\"}}",
o.MDSvisualISN, o.name)).ToList();
#>
             <#= string.Join(",", dictValues) #>
        };
       } }

  public ActionResult <#= ctrl.name #>(List<DataViewObject> list) {
       var t =titles;
       ViewBag.Titles = t;
       return PartialView("FlatGui<#= ctrl.name #>", list.Select(o=>o as <#=
ctrl.basedOn.name #>).ToList());}
}
<#+
  }
}
#>
```

"Index" tipa saskarnes kontroļu kontrolieru noģenerētais piemērs (izgriezums)

```
using System.Collections.Generic;
using System.Web.Mvc;
using System;
using System.Linq;
using MEDUSR;
using MEDUSP;
using MEDUSBL;
using Terra.Controllers;
namespace MEDUSGUI {
  public partial class GuiControlClass_index : V2_Root {}
  public partial class GuiControlClass_details : V2_Root {}
  public partial class GuiControlClass_edit : GuiControlClass_editable {}
  public partial class GuiControlClass_new : GuiControlClass_editable {}
  public partial class GuiControlClass_editable : V2_Root {}
```

```
  public partial class GuiControlClass_newd : V2_Root {}
…
public partial class FlatGuiAspNetUsers_N_R_index :
GuiControlClass_AspNetUsers_N_R_index {
  public override string dataViewName { get {  return "AspNetUsers_N_R"; } }
  public override Dictionary<string, string> titles { get {
   return new Dictionary<string,string>() {

       {"AspNetUsers_N_R_index.UserName","UserName"},{"AspNetUsers_N_R_index.Email
","Email"},{"AspNetUsers_N_R_index.SecurityStamp","SecurityStamp"},{"AspNetUsers_N
_R_index.EmailConfirmed","EmailConfirmed"},{"AspNetUsers_N_R_index.PasswordHash","
PasswordHash"},{"AspNetUsers_N_R_index.PhoneNumber","PhoneNumber"},{"AspNetUsers_N
_R_index.PhoneNumberConfirmed","PhoneNumberConfirmed"},{"AspNetUsers_N_R_index.Two
FactorEnabled","TwoFactorEnabled"},{"AspNetUsers_N_R_index.LockoutEndDateUtc","Loc
koutEndDateUtc"},{"AspNetUsers_N_R_index.LockoutEnabled","LockoutEnabled"},{"AspNe
tUsers_N_R_index.AccessFailedCount","AccessFailedCount"},{"AspNetUsers_N_R_index.I
d","Id"},{"AspNetUsers_N_R_index.ForceToChangePassword","ForceToChangePassword"}
         };
       } }

  public ActionResult AspNetUsers_N_R_index(List<DataViewObject> list) {
       var t =titles;
       ViewBag.Titles = t;
       return PartialView("FlatGuiAspNetUsers_N_R_index", list.Select(o=>o as
AspNetUsers_N_R).ToList());}
}


…
}
```

Manuāli sagatavota virsklase "Index" tipa saskarnes kontroļu kontrolieriem

```
using MEDUSGUI.Models;
using MEDUSP;
using MEDUSR;

namespace MEDUSGUI {
  partial class GuiControlClass_index {
    public override void fillModel(UniversalPlug model, MEDUSR.SelectionDescriptor
selis = null) {
      model["Titles"]=titles;
      var sd=selis==null?new SelectionDescriptor():selis;
      model["data"]=MEDUSR.Service_MAINbl.api.select(dataViewName, sd).data;
      }
    public override MEDUSR.Models.M_Root makeModel(UniversalPlug parameters) {
      var r=new M_index();
      r.Titles=titles;
      var selis=parameters.get("selector") as SelectionDescriptor;
      if(selis==null)
        selis=new SelectionDescriptor();
      r.data=MEDUSR.Service_MAINbl.api.select(dataViewName, selis).data;
      r.guiView=this.GetType().Name;
      return r;
      }
    }
  }
```