eOskrba

Spletna aplikacija (uporabniški vmesnik)

dokumentacija za razvijalce

Zadnja sprememba: 18. 7. 2011

Avtor:

- Aleksander Bešir
  alex.besir@gmail.com

V dokumentu je razloženo delovanje uporabniškega vmesnika (spletne aplikacije) kot dela projekta eOskrba.

Vsebina dokumentacije

1 Prijava v sistem

1.1 Potek v ozadju

Prijava v sistem deluje tako, da se za izbran par uporabniškega imena in gesla, ki ju uporabnik vpiše v login obrazec, naredi poizvedba na Actitviti engine. Če uporabnik obstaja in je avtentikacija uspela, se instanca REST client-a, preko katerega je poizvedba stekla, shrani in se referenca do instance shrani v uporabnikovo sejo.

Vse nadaljnje poizvedbe na Activiti engine-u, ki se za uporabnika zgodijo, potekajo preko v ta namen napisanih funkcij REST client-a, ki so del shranjene instance.

Shranjena (vzdrževana) instanca REST client-a za uporabnika se izniči, ko se uporabnik odjavi. Če se uporabnik ne odjavi s pritiskom na gumb “Odjava”, ostane instanca shranjena vse dokler je seja še veljavna. V trenutku, ko preteče uporabnikova seja, Java (oz. njen garbage collector) sama izniči instanco REST clienta.

Poleg instance REST clienta se v uporabnikovi seji hranijo še različni podatki (polno ime, e-mail naslov, role, group).

1.2 Kategorizacija uporabnikov

Za kategorizacijo uporabnikov obstajata v vsaki uporabniški seji 2 spremenljivki, od katerih je odvisno obnašanje ostalih delov aplikacije. Glede na ti 2 spremenljiivki se določa, kateri deli aplikacije so dostopni uporabnikom (varnost), ter katera vsebina se uporabniku prikaže (bodisi vsebina za bolnike z astmo, ali shizofrenijo ali sladkorno boleznijo). Ti spremenljivki sta:

- session.role Zavzame lahko vrednost “patient”, “caremanager” ali “doctor”,
- session.group Zavzame lahko vrednost “asthma”, “diabetes” ali “schizophrenia”

2 Navigacija

2.1 Osnovna postavitev

Aplikacija je na videz podobna aplikaciji iGoogle. Če povzamem - sestavljena je iz enega glavnega meni-ja, ki daje videz “zavihkov”, vsak zavihek (oz. stran, ki se za izbran zavihek prikaže) pa je sestavljena iz samostojnih manjših aplikacij. Le-te bom v nadeljevanju poimenoval “widget”-i. Vsak widget je premičen, ter se ga da skriti.

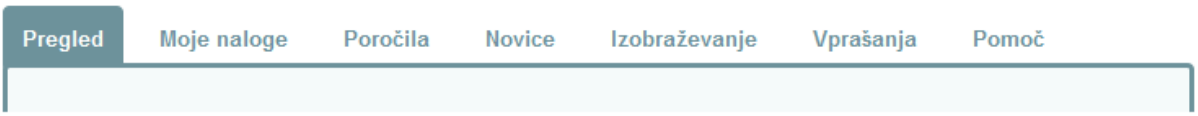
Widgeti so večinoma narejeni tako, da se po naloženi spletni strani še sami posebej naložijo, in sicer preko AJAX-a. AJAX poizvedbe se vršijo z uporabo jQuery javascript API-ja.

2.2 Zavihki

Za vsako vlogo (session.role) obstaja svoj controller, v katerem so definirani zavihki:

- patientTabsController
- caremanagerTabsController
- doctorTabsController

Iz zavihkov se sestavi glavni menu na vrhu strani aplikacije:



Vsak zavihek ima vsebino definirano v svoji GSP datoteki (tj. v svojem pogledu). V osnovi je vsak zavihek sestavljen iz levega in desnega okvirja za widgete (widget-holders), v katerih se potem naložijo posamezni widgeti. Na vrhu vsebina vsakega zavihka naj bi bil tudi en fiksenen (nepremičen) okvir (definiran z `<eo:widget intro=”yes” >`), v katerem je na kratko uporabniku razloženo, kako uporabljati posamezne widget-e.

Dostop do vsakega controllerja z jezički je omejen na eno vlogo (session.role). Filtriranje se vrši v roleTabsFilters filtru.

3 Widgeti

Vsak widget je definiran z unikatnim imenom, 1 grails controllerjem, 1 pravilom znotraj grails filter classa widgetsFilter ter svojim direktorijem (istoimenskim) v grails direktoriju "Views".

Vsak controller widgeta mora imeti akcijo "show", ki poskrbi za prikaz glavne vsebine widgeta. Akcija "show" lahko zahteva različne parametre, ki se podajajo kot GET ali POST parametri..

3.1 Dostopnost

Vsak widget je dostopen samo uporabnikom ki so neprijavljeni. Če do widgeta želi dostopati neprijavljen uporabnik, se namesto vsebine widgeta vrne besedilo, ki to pojasni.

Vsak widget je lahko dostopen ali nedostopen posamezni vlogi uporabnika (vloga je shranjena v spremenljivki session.role, lahko pa zavzame vrednost "patient", "caremanager" ali "doctor").

Preverjanje, ali posamezni uporabnik sme dostopati do widgeta, se izvaja v grails filtru widgetsFilter.

3.2 Sestava widgetov, nalaganje in reagiranje na dogodke

3.2.1 Deklariranje widgeta v GSP

V GSP (Grails) dodamo widget tako, da uporabimo poseben tag, ki je deklariran v lastnem taglib-u eoskrbe (eoskrbaTagLib):

```
<eo:widget2 title="" load="" action="" color="" id="" params="">
    ...
</eo:widget2>
```

Pomen atributov eo:widget2 tag-a:

- **title**
  Naslovni napis, ki se pojavi v okvirčku widgeta
- **load**
  Ime widgeta, ki ga želimo naložiti (primer: "myTasksWidget")
- **action**
  Ime akcije, ki jo želimo prikazati (te so za vsak widget naštete v nadeljevanju. Primer: »show«)
- **color**
  Barva okvirčka widgeta. Lahko je: "normal", "red", "yellow"
- **id** (neobvezno)
  Parameter ID, ki se akciji posreduje. Smiselno samo pri akcijah, ki sprejmejo parameter z imenom "id". (Primer: "1894")
- **params** (neobvezno)
  Seznam ostalih parametrov, ki se posredujejo akciji. Seznam parametrov, ki jih akcije sprejmejo, je v nadeljevanju tega dokumenta.

Vsebina zapisana med tag-oma <eo:widget2> in </eo:widget2> se vključi v widgetov “footer”. Sem naj se definirajo JavaScript funkcije, ki jih sprožijo različni dogodki v widgetih. Le-ti so našteti za vsako akcijo v nadeljevanju dokumenta.

3.2.2 Sestava widgeta

Widget, ki je deklariran na zgoraj opisan način je v HTML zapisu razdeljen z natanko določeno poimenovanimi razdelki (ki so realizirani z tag-om).

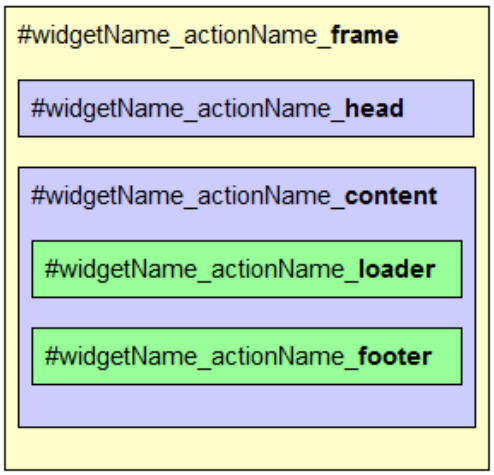
Razdelki so poimenovani (id) avtomatsko, in sicer so oblike:

```
#imeWidgeta_imeAkcije_imeRazdelka
```

Tako na primer “header” razdelek “myTasksWidgeta” pri akciji “show” dobi ime (id):

```
#myTasksWidget_show_header
```

Za vsak widget velja naslednja oblika:



Razlaga pomena in vsebine posameznih razdelkov:

- **frame**
  Osnovni okvir celotnega widgeta
- **head**
  Vsebuje naslov widgeta in kontrole za premik widgeta
- **content**
  Vsebuje loader in footer
- **loader**
  To je razdelek, v katerega se naloži akcija widgeta. Vse kar uporabnik vpiše, klikne in vidi, se nahaja v tem razdelku
- **footer**
  Tu je JavaScript koda, ki avtomatsko naloži akcijo v loader razdelek. Tu se nahaja tudi vsa dodatna vsebina, ki jo vpišemo med <eo:widget2> in </eo:widget2> tagoma. Ta razdelek je predviden za definicije funkcij, ki jih prožijo dogodki v widgetih.

3.2.3 Nalaganje

S kreiranjem widgeta tako, kot je to vpisano na začetku tega poglavja, se widget naloži sam, kadarkoli se GSP koda izvede (prikaže v brskalniku). Poleg tega avtomatskega nalaganja, lahko z JavaScript-om obstoječ widget ponovno naložimo še na dva načina. To je uporabno recimo, če se zaradi uporabnikove interakcije vsebina v bazi spremeni in bi bilo zato treba vsebino widgeta osvežiti oziroma ponovno naložiti:

- widgetName_actionName**_refresh()**
  Ob klicu te predefinirane JavaScript funkcije se v loader razdelek widgeta ponovno naloži akcija. Pri tem ostaneta "id" in "params" atributa, ki sta bila definirana v GSP kodi, enaka.
- widgetName_actionName**_reload(params)**
  Ob klicu te predefinirane JavaScript funkcije se v loader razdelek widgeta ponovno naloži akcija, vendar se pri tem ne uporabita "id" in "params" atributa, ki sta bila bila definirana v GSP kodi, vendar se uporabijo novi parametri, ki so definirani v params params objektu. Primer params objekta: `{ id : 1001, quantity: 5 }`

3.2.3.1 Nalaganje widgeta brez <eo:widget2> tag-a

Za posebne primere obstaja tudi JavaScript funkcija, ki naloži vsebino widgeta v poljubni HTML element (običajno div) brez da bi widget pred tem naložili z uporabo <eo:widget2> tag-a:

```
loadWidgetToElement(widgetPath, elementSelector, data)
```

Parameter widgetPath je niz z url-jem widgeta, parameter elementSelector je niz z id atributom HTML elementa, v katerega želimo naložiti widget, parameter data pa je objekt parametrov za izbran widget.

Če bi tako na primer želeli ustvariti funkcijo primerNalaganjaRocno(), ki naloži v HTML div element z id atributom "primerDiv-a" akcijo "showConversation" widgeta "myTicketsWidget" s parametrom "id" 100, bi to zapisali takole:

```
<g:javascript>
    function primerNalaganjaRocno() {
        loadWidgetToElement(
            "${g.createLink(
                controller:'myTicketsWidget',
                action:'showConversation'
             )}",
            "#primerDiv-a",
            { id: 100 }
        );
    }
</g:javascript>
```

Opozorilo: Tako naloženim widget-om se **_refresh()** in **_reload(params)** funkciji ne generirata avtomatsko. Za njuni funkcionalnosti mora programer poskrbeti sam, če ju potrebuje. Prav tako na ta način naložen widget nima ogrodja, ki je predstavljeno v poglavju 3.2.2. Funkcija loadWidgetToElement naloži v HTML element, ki je opredeljen s parametrom elementSelector, samo to, kar <eo:widget2> naloži v #widgetName_actionName_loader odsek avtomatsko. Tako naložen widget pa ima še zmeraj vse ostale funkcionalnosti, ki so zapisane v nadeljevanju v poglavju 3.3.

3.2.4 Reagiranje na dogodke

Vsak klik na gumb v widgetu (oziroma widgetovi akciji) še ne naredi ničesar. Vse kar se ob pritisku na gumb "Pokaži obrazec" v "show" akciji "myTasksWidget" widgeta zgodi, je JavaScript klic funckije myTasksWidget_show_onShowForm(id). Ta funkcija še ni definirana in jo mora uporabnik widgeta definirati sam. Definira jo tako, da jo implementira med <eo:widget2> in </eo:widget2> tagoma, kot že rečeno.

Če bi na primer imeli widget "sampleWidget" z akcijo "show", ki vsebuje gumb "Sporoči" in bi želeli, da se ob kliku na gumb prikaže sporočilno okno s kratkim sporočilom, bi najprej morali pogledati v to dokumentacijo in ugotoviti, katera funkcija se proži ob kliku na gumb. Za vsak gumb vsake akcije je to zapisano v poglavju 3.3, pod Gumbi in JavaScript klici. V našem primeru bi ime funkcije lahko bilo sampleWidget_show_onShowMessage(). Ko poznamo ime funkcije, lahko v GSP kodi ustvarimo widget:

```
<eo:widget2 title="Primer" load="sampleWidget" action="show"
color="normal">
   <g:javascript>
      function sampleWidget_show_onShowMessage() {
         alert("Primer kratkega sporočila");
      }
   </g:javascript>
</eo:widget2>
```

Opozorilo: Ne pozabite na <g:javascript> tag, saj <eo:widget2> dopušča vstavljanje tudi drugih elementov v footer in ne samo JavaScript kode.

S takšnim načinom "lovljenja" dogodkov v widgetih lahko kadarkoli brez poseganja v kodo widgetov, ampak samo s posegom v GSP kodo, spremenimo JavaScript kodo, ki določa, kaj se dogodi pri posameznem dogodku.

3.2.5 Vidne in ne-vidne akcije widgetov

Vsak widget ima takoimenovane "vidne" in "ne-vidne" akcije. Vidne akcije, so tiste, ki vrnejo GSP kodo, ki jo lahko vstavimo v loader razdelek widgeta (to se zgodi avtomatsko pri kreaciji widgeta). Tako recimo ne moremo ustvariti widgeta z:

```
<eo:widget2 title="Primer" load="sampleWidget" action="delete"
color="normal">
```

Nevidne akcije so namenjene asinhronim (ajax) zahtevkom. Čeprav so te akcije "nevidne" vseeno vrnejo nekaj GSP kode, v kateri je zgolj sporočilo, ali je bila akcija uspešna ali ne. Za prikaz teh sporočil v layout-u spletne aplikacije obstaja prazen tag z id-jem #ajax-response-holder.

Da nam ne bi bilo treba asinhronih klicev nevidnih akcij izvajati sami in vrnjeno sporočilo o uspešnosti prav tako sami naložiti v #ajax-response-holder, obstaja JavaScript funckija (v eoskrba.js), ki to naredi avtomatsko:

```
postWithFlash(url,data,afterComplete)
```

Klic funkcije izvede akcijo, ki je definirana z url in ji posreduje parametre, ki so v objektu data, po tem pa sporočilo o uspehu avtomatsko naloži v #ajax-response-holder. V afterComplete lahko dopišemo še anonimno funkcijo (ali ime obstoječe funkcije), ki se bo izvedla natanko tedaj, ko se v #ajax-response-holder zapiše sporočilo o uspehu.

Če bi tako na primer želeli ob kliku na nek gumb neke akcije nekega widgeta asinhrono izvesti nevidno akcijo nekega drugega widgeta in sporočilo o uspehu prikazati uporabniku, bi to lahko v GSP zapisali takole:

```
<eo:widget2 title=”Primer” load=”widget1” action=”action1”
color=”normal”>
   <g:javascript>
      function widget1_action1_onSomeEvent() {
         postWithFlash(
            "${g.createLink(
               controller:’widget2’,
               action:’action2’
            )}", {
               param1: “blabla”,
               param2: 1024,
               param3: true
            },
            null
         );
      }
   </g:javascript>
</eo:widget2>
```

3.3 Posebni widgeti

Poleg standardnih widget-ov, ki jih ponavadi naložimo z uporabo `<eo:widget2>` tag-a, imamo še nekaj posebnih widgetov.

3.3.1 panelWidget

Poleg standardnega `<eo:widget2>` tag-a za nalaganje widget-ov imamo poseben tag, za nalaganje panelWidget-a:

```
<eo:panelWidget title="" readyText="" color="" params="">
   ...
</eo:panelWidget>
```

`<eo:panelWidget>` tag je podoben `<eo:widget2>` tag-u, s to razliko, da programer ne izbere widgeta in akcije, ki naj se naložita, pač pa se zmeraj avtomatsko naloži panelWidget widget.

panelWidget je poseben widget, ki služi kot okvir, v katerega lahko potem dinamično / asinhrono nalagamo poljubne druge widgete. Sam po sebi nima panelWidget nobene funkcionalnosti in posledično nima niti svojih dogodkovnih akcij.

Ko v GSP kodi deklariramo widget z `<eo:widget2>` tagom , se na zaslonu izriše okvirček, ki vsebuje vsebino izbranega widgeta. Kasneje lahko vsebino spreminjamo le z _reload() in refresh() vgrajenima funkcijama, še zmeraj pa v okvirčku ostaja naložen isti widget. Namen panelWidget-a pa je, da v svoj okvirček dovoli kasnejše nalaganje katerih-koli drugih widgetov. (v istem okvirčku je lahko tako na primer najprej widget koledar, kasneje se vanj naloži widget z nalogami...)

Za dinamično nalaganje poljubnih widgetov v panelWidget okvirček imamo že definirano JS funkcijo:

```
panelWidget_load(title, widget, action, paramsToLoad);
```

Klic te fukcije v panelWidget (nekje v GSP kodi smo morali prej uporabiti `<eo:panelWidget>` tag) naloži drug widget, ki ga določimo z parametroma widget in action (oboje sta niza (string-a)). Poleg tega lahko podabo tudi objekt paramsToLoad, v katerem so dodatni parametri za widget, ki ga želimo dinamično naložiti. S parametrom title lahko zamenjamo naslov okvirčka.

Primer uporabe (imena so namišljena):

```
panelWidget_load(
   "Nov naslov okvirčka",
   "sampleWidget",
   "someAction",
   {
      id: 1,
      showDetails: true
   }
);
```

Poleg `panelWidget_load` funckije, obstaja še ena funkcija, ki je že definirana:

```
panelWidget_unload(title);
```

Ta funckija samo izprazni okvirček, ponastavi naslov na `title` in v okvirček vstavi besedilo, ki je bilo v `<eo:panelWidget>` tagu določeno z atributom `readyText`.

Obe zgoraj omenjeni predefinirani funkciji ostaneta po uporabi še zmeraj prisotni. Dinamično naloženi widgeti ne uničijo panelWidget-a in njegovih funkcionalnosti.

Ker widgeti, ki jih dinamično nalagamo v panelWidget zelo pogosto prožijo dogodke, moramo imeti, tako kot pri `<eo:widget2>` tag-u nek sistem, kako dogodke uloviti in reagirati nanje. Pri `<eo:panelWidget>` tagu to naredimo tako, da med začetnim in zaključnim tagom vstavimo JavaScript kodo s katero definiramo funkcije, ki naj se odvijejo pri posameznem dogodku. Edina razlika je v tem, da moramo v naprej vedeti, kateri widget-i bodo morda dinamično naloženi v panelWidget. Če imamo namen v panelWidget dinamično naložiti widget primerWidgetA (akcijo show) in primerWidgetB (akcijo form), ob widgeta pa sprožita vsaj en dogodek, bi naša GSP koda zgledala nekako takole:

```
<eo:panelWidget>
   <g:javascript>
      function primerWidgetA_show_onButtonClick() {
         // reagiraj
      }
      function primerWidgetB_form_onSomething() {
         // reagiraj
      }
   </g:javascript>
</eo:panelWidget>
```

3.4 Imena in opisi

Tu so našteti vsi widget-i, ki sestavljajo eOskrba spletno aplikacijo, njihovi opisi, dostopnost, akcije, možni parametri akcij ter javascript dogodki, ki jih akcije povzročijo.

Ne-vidne akcije imajo za imenom zvezdico (*).

3.4.1 panelWidget

Opomba: **panelWidget widget-a naj ne bi uporabljali ročno. Ta widget je posredno uporabljen z uporabo tag-a <eo:panelWidget>. Spodnje informacije o widgetu dajejo zgolj vpogled o delovanju v ozadju. Navodila za uporabo <eo:panelWidget> tag-a so v poglavju 3.3.1.**

Ne-pravi widget, ki služi kot poseben vsebovalnik za widgete, ki se naložijo dinamično/asinhrono (npr. ob kliku na določen gumb). Ta widget ima samo eno akcijo, `show`, ki zahteva 3 vhodne parametre in poljubno število dodatnih:

- `readyText`
- `widgetToLoad`
- `actionToLoad`
- *dodatni parametri...*

Kar widget ob nalaganju naredi, je preprosto to, da vase naloži drug widget, ki je opredeljen z vsemi parametr. Če parameter widgetToLoad nastavimo na null ali pa ni opredeljen, potem panelWidget ne naloži vase drugega widgeta, vendar namesto njega izpiše besedilo, ki ga podamo s parametrom `readyText`.

panelWidget ima posebno vlogo – uporablja se v primeru, ko želimo da celotno desno stran uporabniškega vmesnika zavzema en sam okvir, v katerega potem glede na uporabnikove akcije dinamično nalagamo ustrezne widgete. Zato tega widgeta naj ne bi nikoli naložili z <eo:widget2> tagom, ampak z <eo:widget2p> tagom.

Tudi ta widget ima (tako kot ostali) predefinirani funkciji `panelWidget_show_reload()` in `panelWidget_show_refresh()`, vendar je njuna uporaba zaradi specifičnosti uporabe odsvetovana. Za dinamično nalaganje widgetov v panelWidget imamo že predefinirano JS funkcijo panelWidget_load():

```
panelWidget_load(
   title,
   widgetPath,
   params
);
```

Uporaba zgornje JS funkcije ponovno naloži panelWidget tako, da le-ta takoj vase naloži izbran drug widget (določen z widgetPath in params). Poleg tega nastavi naslov okvirčka na `title`.

Če imamo tako na primer na strani nekaj navadnih widgetov, med njimi preprost widget z gumbom, ki ob kliku povzroči dogodek `sampleWidget_someAction_onSomeEvent()`, ter panelWidget na desni, lahko z JS zapišemo tako funckijo:

```
function sampleWidget_someAction_onSomeEvent() {
   panelWidget_load(
      "Novica 12",
      "${g.createLink(controller:'newsWidget',action:show)}",
      {
         id: 12,
         textSize: 14
      }
```

```
    );
}
```

Zgornja koda bi povzročila to, da bi ob kliku na gumb nekega navadnega widgeta z imenom `sampleWidget` (in akcijo `someAction`) v panelWidget dinamično naložili widget `newsWidget` z akcijo `show` in parametri `id` in `textSize`. Tak widget v tem projektu v resnici ne obstaja, tu smo si ga izmislili samo za primer.

Kar je še potrebno opozoriti, je, da tako naloženi widget-i v panelWidget nimajo avtomatsko predefiniranih funckij _refresh() in reload(), saj niso bili naloženi z <eo:widget2> tag-om. Vendar jih prav tako lahko osvežimo z uporabo predefinirane panelWidget_show_refresh(), čeprav to ni priporočljivo. Priporočljivo je, da pri menjavi vsebine v panelWidgetu zmeraj uporabimo funckijo `panelWidget_load()`.

Pri widgetih, ki jih dinamično naložimo v panelWidget je nekoliko drugače tudi z lovljenjem dogodkov (eventov). Ker tako naloženi widget-i niso obdani s svojim <eo:widget2> tag-om, moramo dogodkovne funckije definirati nekje drugje (pri navadnih widgetih te funkcije namreč definiramo med pripadajočima <eo:widget2> in <eo:widget2 /> tagoma.). Priporočljivo je, da te funkcije definiramo med <eo:widget2p> in <eo:widget2p /> tagoma, ki ju uporabimo za nalaganje panelWidget-a.

show(readyText, widgetToLoad, actionToLoad, ...) *

opis	Edina funkcija panelWidget-a.	
params	**readyText**	Besedilo, ki se prikaže znotraj panelWidgeta, če je parameter widgetToLoad enak null ali ni definiran
	widgetToLoad	Niz (string), ki predstavlja ime widget-a, za katerega želimo da se naloži v panelWidget
	actionToLoad	Niz (string), ki predstavlja ime akcije widgeta, ki jo želimo uporabiti
	...	Poljubno število dodatnih parametrov, ki se posredujejo naprej widgetu, ki je opredeljen z widgetToLoad in actionToLoad.

3.4.2 myTasksWidget

Omogoča delo z nalogami iz Activiti engine-a.

Dostopnost: P, CM, D.

show()

opis	Prikaže vse Activiti naloge dodeljene uporabniku	
dogodki	**_onShowForm(int id_taska)**	klik na gumb »Prikaži obtazec« pri določeni nalogi, ki zahteva obrazec

	`_onShowTask( int id_taska)`	klik na gumb »Prikaži obtazec« pri določeni nalogi, ki zahteva potrditev

form(id, redirectToController, redirectToAction)

opis	Prikaže informacije o nalogi, obrazec (če ga naloga ima) ter potrditveni gumb	
params	`id`	id task-a, za katerega želimo prikazati obrazec
	`returnToController`	controller, kamor naj se uporabnik vrne po zaključku
	`returnToAction`	action, kamor naj se uporabnik vrne po zaključku

completeTask(id) *

opis	Ne-vidna akcija, ki zaključi task, ki ne zahteva nobenih vnosnih podatkov (nima form-a ali ima form, ki nima vnosnih polj)	
params	`id`	Id številka Activiti taska, ki naj se zaključi

3.4.3 myTicketsWidget

Omogoča delo z vprašanji tipa »ticket«.

Dostopnost: P, CM, D

showOpen()

opis	Prikaže vsa uporabnikova odprta vprašanja	
dogodki	`_onShowConversation( ticket_id, topic )`	klik na naslov vprašanja ali izbira »prikaži celoten pogovor«
	`_onCloseTicket( ticket_id )`	izbira »zapri vprašanje«

showClosed()

opis	Prikaže vsa uporabnikova zaprta vprašanja	
dogodki	`_onShowConversation( ticket_id, topic )`	klik na naslov vprašanja

newTicketForm()

opis	Prikaže obrazec za vpis novega vprašanja	
dogodki	`_onSubmitNewTicket( topic, question )`	Oddaja novega vprašanja. Topic je naslov vprašanja, question pa vsebina.

newTicketSubmit(topic, question) *

opis	Postavljeno vprašanje doda v bazo.	
params	`topic`	Naslov (tema) vprašanja
	`question`	Vsebina (besedilo) vprašanja

showConversation([id])

opis	Prikaže celoten pogovor za izbrano vprašanje in ponudi možnost oddaje odgovora, če je vprašanje odprto	
params	`id`	Id številka vprašanja
dogodki	`_onCloseTicket( ticket_id )`	Klik na gumb »Zapri vprašanje«
	`_onSubmitReply( ticket_id, content )`	Oddaja odgovora na vprašanje

replySubmit(id, content) *

opis	Doda odgovor z vsebino content vprašanju	
params	`id`	Id številka vprašanja, kateremu je namenjen odgovor
	`content`	Vsebina odgovora

closeSubmit(id) *

opis	Zapre vprašanje	
params	`id`	Id številka vprašanja, ki naj se zapre

3.4.4 liveChatWidget

Omogoča pogovor v živo med pacienti in oskrbovalci oz. zdravniki.

patientButton()

opis	Prikaže gumb, s katerim lahko pacient zahteva pogovor v živo	
dogodki	`_onStartChat()`	Klik na gumb »Začni pogovor v živo«

patientWait()

opis	Ustvari nov zahtevek za pogovor v živo čaka na prevzem. Ko oskrbovalec ali zdravnik prevzame pogovor, se sproži dogodek _onAccept(chatSession_id)	
dogodki	`_onAccept( chatSession_id )`	Zahteva za pogovor v živo je bila sprejeta. chatSession_id je id številka pogovora.
	`_onCancel()`	Uporabnik zahteva prekinitev zahteve za pogovor.

showPending()

opis	Prikaže seznam vseh pacientov, ki čakajo na prevzem njihove zahteve za pogovor v živo. Vsebina widgeta se ne osveži avtomatsko – za to mora poskrbeti programer, ki si lahko pomaga z dogodkom _onTick(), ki se sproži 3 selunde po naložitvi widget-a. Če za uporabnika še zmeraj obstaja pogovor, ki še ni zaključen, widget avtomatsko blokira (ne dovoli) prevzem novih zahtevkov.	
dogodki	`_onAccept( chatSession_id )`	Uporabnik prevzame pogovor.
	`_onTick()`	Od zadnje osvežitve so potekle 3 sekunde.

startChat(id)

opis	Prične pogovor s sejno številko id. Če to akcijo widgeta naloži oskrbovalec ali zdravnik, se pogovor pred tem še avtomatkso prevzame. V widgetu se prikaže prostor, kamor se avtomatsko dodajaja novo besedilo, ter vnosno polje, ki omogoča uporabnikom pisanje besedila. Ob kliku na gumb »Pošlji« startChat poskrbi sam za dodajanje odgovora v bazo (ne sproži dogodka, ki bi ga moral programer uloviti in odgovor dodati sam). Edini dogodek, ki ga ta akcija proži, je _onEndChat(), ki se zgodi, če uporabnik klikne na gumb za konec pogovora. Tudi v tem primeru se pogovor zaključi avtomatsko. Programer lahko dogodek uporabi za preusmeritev uporabnika na drugo lokacijo po kliku na gumb.	
params	`id`	Id številka seje pogovora
dogodki	`_onEndChat()`	Uporabnik klikne na gumb za konec pogovora

3.4.5 newsReaderWidget

Omogoča branje novic.

Dostopnost: P, CM, D.

show(page, itemsPerPage)

opis	Prikaže seznam novic (naslov). Omogoča ostranjevanje s parametroma page in items per page.	
params	`page`	Številka strani za prikaz (privzeto: 1)
	`itemsPerPage`	Število novic na stran (privzeto: 5)
dogodki	`_onPage( page_num )`	Zamenjava strani
	`_onReadMore( newsitem_id, title );`	Zahteva za prikaz celotne novice z določeno id številko in naslovom

readMore(id)

opis	Prikaže vsebino novice.	
params	`id`	Id številka novice za prikaz

3.4.6 newsWriterWidget

Omogoča objavo, urejanje in brisanje novic.

Dostopnost: CM, D.

show()

opis	Prikaže vse novice in ponudi možnost brisanja in urejanja lastnih novic.	
dogodki	`_onEdit( newsitem_id )`	Zahteva za urejanje novice
	`_onRemove( newsitem_id )`	Zahteva za brisanje novice

form(id)

opis	Prikaže obrazec za dodajanje nove novice	
dogodki	`_onSubmit ( title, content )`	Potrditev obrazca za ustvarjanje nove novice

edit(id)

opis	Prikaže obrazec za spreminjanje novice (če je id določen)	
params	`id`	Id številka novice, ki jo želimo spremeniti.
	`_onSubmit ( id, title, content )`	Potrditev obrazca za urejanje novice

addNewsitem(title, content) *

opis	Ne-vidna akcija, ki v sistem doda novico.	
params	`title`	Naslov nove novice
	`content`	Besedilo nove novice

updateNewsitem(id, title, content) *

opis	Ne-vidna akcija, ki popravi vsebino novice	
params	`id`	Id številka novice
	`title`	Novi naslov novice
	`content`	Novo besedilo novice

removeNewsitem(id) *

opis	Ne-vidna akcija, ki odstrani novico	
params	`id`	Id številka novice

3.4.7 adviceReaderWidget

Omogoča branje priporočil.

Dostopnost: P, CM, D.

show()	
opis	Prikaže seznam priporočil (naslov).

3.4.8 adviceWriterWidget

Omogoča objavo, urejanje in brisanje priporočil.

Dostopnost: CM, D.

show()

opis	Prikaže vse novice in ponudi možnost brisanja in urejanja priporočil.	
dogodki	`_onEdit( adviceitem_id )`	Zahteva za urejanje priporočila
	`_onRemove( adviceitem_id )`	Izbira brisanja priporočila

form()

opis	Prikaže obrazec za dodajanje novega priporočila.	
dogodki	`_onSubmit( title, content, dateStart, dateEnd, priority )`	Potrditev obrazca

addAdviceitem(title, content, dateStart, dateEnd, priority) *

opis	Ne-vidna akcija, ki v sistem doda priporočilo.	
params	`title`	Naslov novega priporočila
	`content`	Besedilo novega priporočila
	`dateStart`	Datum začetka veljavnosti priporočila
	`dateEnd`	Datum konca veljavnosti priporočila
	`priority`	Pomembnost priporočila (med 1 in 100)

updateAdviceitem(id, title, content, dateStart, dateEnd, priority) *

opis	Ne-vidna akcija, ki popravi vsebino priporočila	
params	`id`	Id številka novice
	`title`	Novi naslov novice
	`content`	Novo besedilo novice
	`dateStart`	Nov datum začetka veljavnosti priporočila
	`dateEnd`	Nov datum konca veljavnosti priporočila
	`priority`	Pomembnost priporočila

removeAdvceitem(id) *

opis	Ne-vidna akcija, ki odstrani priporočilo	
params	`id`	Id številka priporočila

3.4.9 myCalendarWidget

Omogoča uporabo osebnega koledarja, v katerem se lahko označi osebne dogodke.

Dostopnost: P, CM, D.

show(year, month)

opis	Prikaže koledar za mesec month leta year za prijavljenega uporabnika	
params	`year`	Letnica koledarja (npr. 2011)
	`month`	Številka izbranega meseca (npr 1 za januar)
dogodki	`_onDayChosen( year, month, day )`	Klik na določen dan v koledarju. Parametri funckije so izraženi kot cela števila (od 1 naprej)
	`_onPrevious( year, month )`	Klik na gumb za prikaz prejšnjega meseca
	`_onNext( year, month )`	Klik na gumb za prikaz naslednjega meseca

Poleg zgornjih dogodkovnih funkcij, ki jih mora uporabnik definirati, obstajata še dve že definirani funkciji, s katerima si uporabnik lahko pomaga. Ti dve funckiji lahko uporabnik kliče kadarkoli je ta akcija naložena. Obe vračata vrednost tipa int:

- `_getSelectedYear()` - Vrne številko leta, ki je trenutno prikazano
- `_getSelectedMonth()` - Vrne številko meseca, ki je trenutno prikazan (1 do 12)

details(year, month, day)

opis	Prikaže podrobnosti koledarja za datum, ki ga določajo parametri.	
params	`year`	Letnica izbranega datuma (celo število)
	`month`	Mesec izbranega datuma (celo število med 1 in 12)
	`day`	Dan v mesecu izbranega datuma (celo število od 1 naprej)
dogodki	`_onAddNotice( year, month, day, text )`	Uporanik doda opombo za datum določen s prvimi tremi parametri in besedilom, določenim s četrtim parametrom
	`_onRemoveNotice( notice_id )`	Izbira odstranitve določene opombe

addNotice(year, month, day, text) *

opis	Nevidna akcija, ki doda opombo z vsebino text in datumom, ki je opredeljen s parametri.	
params	**year**	Letnica izbranega datuma (celo število)
	month	Mesec izbranega datuma (celo število med 1 in 12)
	day	Dan v mesecu izbranega datuma (celo število od 1 naprej)
	text	Besedilo opombe

removeNotice(id) *

opis	Nevidna akcija, ki odstrani opombo z id številko id.	
params	**id**	Id številka opombe, ki naj se odstrani

3.4.10 pefInputWidget

Widget, ki ovija Activiti proces ProcesOpomnikVnosPEF

Dostopnost: P

show()

opis	Prikaže widget. Vsebina widgeta je generirana avtomatsko. Če za danega uporabnika ne obstaja instanca procesa ProcesOpomnikVnosPEF, widget preprosto prikaže besedilo, ki pravi, da uporabniku ni treba vnesti nove vrednosti PEF. Če instanca procesa obstaja, se bodisi prikaže obrazec za vnos, bodisi pa obvestilo o dobri, slabi ali kritični PEF vrednosti in z navodili, kaj storiti v teh primerih.	
dogodki	`_onPefGoodSubmit( task_id )`	Pacient klikne na gumb »V redu« pod obvestilom o dobri vrednosti PEF.
	`_onPefBadSubmit( task_id )`	Pacient klikne na gumb »V redu« pod obvestilom o slabi vrednosti PEF.
	`_onPefCritSubmit( task_id )`	Pacient klikne na gumb »V redu« pod obvestilom o kritični vrednosti PEF.
	`_onPefInputSubmit( task_id, pef_value )`	Pacient vnese izmerjeno PEF vrednost v obrazec in obrazec potrdi.

inputSubmit(id, pef) *

opis	Ne-vidna akcija, ki zaključi Activiti user task vnašanja PEF vrednosti tako, da odda obrazec.	
params	`id`	Id številka Activiti user task-a
	`pef`	Številska vrednost izmerjene PEF vrednosti.

3.4.11 learningContentReaderWidget

Omogoča branje izobraževalnih vsebin.

Dostopnost: P, CM, D.

show(page, itemsPerPage)

opis	Prikaže seznam izobraževalnih vsebin (naslovov). Omogoča ostranjevanje s parametroma page in items per page.	
params	`page`	Številka strani za prikaz (privzeto: 1)
	`itemsPerPage`	Število novic na stran (privzeto: 5)
dogodki	`_onPage( page_num )`	Zamenjava strani
	`_onReadMore(item_id, title );`	Zahteva za prikaz celotne vsebine z določeno id številko in naslovom

readMore(id)

opis	Prikaže izobraževalno vsebino.	
params	`id`	Id številka vsebine za prikaz

3.4.12 learningContentWriterWidget

Omogoča objavo, urejanje in brisanje izobraževalnih vsebin.

Dostopnost: CM, D.

show()

opis	Prikaže vse izobraževalne vsebine in ponudi možnost brisanja in urejanja.	
dogodki	`_onEdit( item_id )`	Zahteva za urejanje vsebine
	`_onRemove( item_id )`	Zahteva za brisanje vsebine

form(id)

opis	Prikaže obrazec za dodajanje nove vsebine	
dogodki	`_onSubmit ( title, content )`	Potrditev obrazca za ustvarjanje nove vsebine

edit(id)

opis	Prikaže obrazec za spreminjanje vsebine	
params	`id`	Id številka vsebine, ki jo želimo spremeniti.
	`_onSubmit ( id, title, content )`	Potrditev obrazca za urejanje vsebine

addLearningContent(title, content) *

opis	Ne-vidna akcija, ki v sistem doda vsebino.	
params	`title`	Naslov nove vsebine
	`content`	Besedilo nove vsebine

updateLearningContent(id, title, content) *

opis	Ne-vidna akcija, ki popravi vsebino	
params	`id`	Id številka vsebine
	`title`	Novi naslov vsebine
	`content`	Novo besedilo vsebine

removeLearningContent (id) *

opis	Ne-vidna akcija, ki odstrani vsebino	
params	`id`	Id številka vsebine

3.4.13 ticketsWidget

Widget za delo z vprašanji, vendar je za razliko od myTicketsWidgeta namenjen oskrbovalcem oz. zdravnikom. Nekatere akcije so zelo podobne. Razlika je v tem, da oskrbovalci in zdravniki vprašanj ne morejo postaviti. Imajo pregled nad vsemi odprtimi vprašanji pacientov, ki niso še dobila odgovorov. Takim vprašanjem lahko oskrbovalec ali zdravnik da odgovor, s čimer vprašanje »prevzame«. To pomeni, da se odgovorjeno vprašanje drugim oskrbovalcem in zdravnikom ne prikaže več. Vprašanje se premakne k oskrbovalčevim oz. zdravnikovim »Mojim odprtim vprašanjem«, kjer ima možnost dodatnega odgovarjanja (oziroma pogovora s pacientom). Vprašanje ostane vidno vse dokler ni pacient z odgovorom zadovoljen in vprašanje zapre. Če pacient odgovorjenega vprašanja iz različnih razlogov ne zapre, a je bilo vprašanje s strani oskrbovalca oz. zrdavnika odgovorjeno in nadaljna razprava ni smiselna, lahko oskrbovalec oz. zdravnik vprašanje prisilno zapre.

Dostopnos: CM, D.

showPending()

opis	Prikaže seznam vprašanj, na katera ni še nihče odgovoril	
dogodki	`_onShowConversation( id, title )`	Zahteva za prikaz celotnega vprašanja

showPendingConversation(id)

opis	Prikaže celotno vprašanje, ki še ni dobilo nobenega odgovora	
params	`id`	Id številka vprašanja
dogodki	`_onSubmitReply( id, content )`	Potrditev obrazca za dodajanje odgovora
	`_onForceClose( id, reason )`	Zahteva za prisilno zaprtje vprašanja

forceCloseTicket(id, reason) *

opis	Ne-vidna akcija, ki prisilno zapre vprašanje	
params	`id`	Id številka vprašanja, ki naj se prisilno zapre
	`reason`	Razlog (besedilo, html) prisilnega zaprtja vprašanja.

addReply(id, content) *

opis	Ne-vidna akcija, ki vprašanju doda odgovor. Če vprašanje čaka na prvi odgovor, postane vprašanje last avtorju odgovora (drugi oskrbovalci in zdravniki tega vprašanja ne vidijo več).	
params	`id`	Id številka vprašanja, kateremu naj se doda odgovor
	`content`	Vsebina odgovora

showMyOpened()

opis	Prikaže seznam vprašanj, ki jih je prijavljen uporabnik »prevzel«.	
dogodki	`_onShowConversation( id, title )`	Zahteva za prikaz celotnega vprašanja

showConversation(id)

opis	Prikaže celotno vprašanje	
params	`id`	Id številka vprašanja
dogodki	`_onSubmitReply( id, content )`	Potrditev obrazca za dodajanje odgovora
	`_onForceClose( id, reason )`	Zahteva za prisilno zaprtje vprašanja

3.4.14 reportingWidget

Widget za prikaz zajetih meritev pacientov tako pacientom, kot oskrbovalcem in zdravnikom. Widget zna izrisati grafe, prikazati podatke v tabeli ipd.

Dostopnost: P, CM, D

patientPicker()

opis	Prikaže obrazec s katerim lahko pacient izbere, kaj želi videti. Dogodek _onPick se sproži po kliku na gumb za prikaz poročila. Po tem se pa dogodek sproži ob vsaki spremembi v obrazcu. ***Ta widget je namenjen pacientom.***	

dogodki	`_onPick(`
	`  what,  <-- array[string]`
	`  from,  <-- long`
	`  till,  <-- long`
	`  func,  <-- string`
	`  scope  <-- string`
	`)`
		Pacient izpolni obrazec in klikne na gumb za prikaz.
		V what se zapišejo imena vrednosti, ki jih želi uporabnik prikazati.
		V from in till se zapišeta timestamp-a obdobja, ki ga želimo prikazati.
		V func se vpiše ime funckije, ki naj se izvede nad grupami meritev. Lahko je 'avg', 'min' ali 'max'.
		V scope se vpiše ime grupirne funkcije. Lahko je 'daily', 'weekly', 'monthly'.

patientShow (what, from, till, func, scope)

opis	Prikaže poročilo glede na izbrane argumente za pacienta. ***Ta widget je namenjen pacientom.***	
params	`what  <-- string`	Niz z imeni vrednosti, ki jih želimo prikazati. Imena vrednosti morajo biti ločena z vejico, brez presledkov. Niz se lahko konča z vejico. *Primer »abc,def,ghi,«*
	`from  <-- long`	Začetek obdobja, ki ga želimo prikazati. Zapis je v JS long predstavitvi časa.
	`till  <-- long`	Konec obdobja, ki ga želimo prikazati. Zapis je v JS long predstavitvi časa.
	`func  <-- string`	Ime funckije, ki naj se izvede nad grupami meritev. Lahko je 'avg', 'min' ali 'max'.
	`scope  <-- string`	Ime funckije, ki naj se izvede nad grupami meritev. Lahko je 'avg', 'min' ali 'max'.
dogodki		

picker()

opis	Prikaže obrazec s katerim lahko CM ali D izbere, kaj želi videti. Dogodek _onPick se sproži po kliku na gumb za prikaz poročila. Po tem se pa dogodek sproži ob vsaki spremembi v obrazcu. ***Ta widget je namenjen D in CM.***	

dogodki	`_onPick(`
	`  who,  <-- string`
	`  what,  <-- array[string]`
	`  from,  <-- long`
	`  till,  <-- long`
	`  func,  <-- string`
	`  scope  <-- string`
	`)`
		D ali CM izpolni obrazec in klikne na gumb za prikaz.
		V who se vpiše uporabniško ime pacienta.
		V what se zapišejo imena vrednosti, ki jih želi uporabnik prikazati.
		V from in till se zapišeta timestamp-a obdobja, ki ga želimo prikazati.
		V func se vpiše ime funckije, ki naj se izvede nad grupami meritev. Lahko je 'avg', 'min' ali 'max'.
		V scope se vpiše ime grupirne funkcije. Lahko je 'daily', 'weekly', 'monthly'.

3.4.15 helpWidget

Widget za prikaz pomoči za uporabo portala (namenjeno pacientom).

contents()

opis	Prikaže seznam vsebine pomoči	
dogodki	`_onShowContent( id )`	Uporabnik klikne na eno izmed vsebin

show(id)

opis	Prikaže besedilo vsebine	
params	`id`	Id številka vsebine pomoči

3.4.16 adminWidget

Widget namenjen D in CM, ki omogoča vključevanje pacientov v sistem, izbris iz sistema, ipd.

asthmaRegForm ()

opis	Prikaže obrazec za vključitev astmatika v sistem	
dogodki	`_onSubmit()`	Oddaja obrazca

3.4.17 patientStatusWidget

Widget, s katerim imajo D in CM pregled nad trenutnim stanjem pacientov.