



ИНСТИТУТ ЗА МАТЕМАТИКУ И ИНФОРМАТИКУ  
ПРИРОДНО-МАТЕМАТИЧКИ ФАКУЛТЕТ  
УНИВЕРЗИТЕТ У КРАГУЈЕВЦУ

ДИПЛОМСКИ РАД

---

**Синхронизација календара за *ownCloud*  
платформу - *OwnCloud* календар конектор**

---

*Студент:*  
Владимир Варагић

*Професор:*  
др Милош Ивановић

Септембар 2016.

# Садржај

<b>1</b>	<b>Увод</b>	<b>3</b>
<b>2</b>	<b>Преглед коришћених технологија</b>	<b>5</b>
<b>3</b>	<b>Радно окружење</b>	<b>7</b>
3.1	XWT платформа . . . . .	7
3.2	WebDAV . . . . .	9
3.3	CalDAV . . . . .	11
<b>4</b>	<b><i>OwnCloud</i> пројекат</b>	<b>13</b>
4.1	<i>Calendar</i> подапликација . . . . .	14
<b>5</b>	<b><i>OwnCloud</i> календар конектор</b>	<b>16</b>
5.1	Жељене функционалности . . . . .	16
5.1.1	Аутентификација . . . . .	18
5.1.2	Синхронизација догађаја на захтев . . . . .	19
5.1.3	Аутоматска синхронизација догађаја . . . . .	20
5.1.4	Преглед преузетих догађаја . . . . .	21
5.1.5	Приказ нотификација . . . . .	21
5.2	Идеје за даљи развој . . . . .	24

# Листа скраћеница

- *API - Application Programming Interface,*
- *BSD - Berkeley Software Distribution,*
- *CalDAV - Calendaring Extensions to WebDAV,*
- *GPL - GNU Public Licence,*
- *IT - Information Technology,*
- *MIT - Massachusetts Institute of Technology,*
- *SVN - Subversion,*
- *URL - Uniform Resource Locator,*
- *WebDAV - Web-based Distributed Authoring and Versioning.*

# Глава 1

## Увод

Технолошки развој, а посебно развој интернета, је довео до тога да је интернет постао саставни и готово неизоставни део свакодневног живота, а постојање и широка употреба мобилних уређаја (паметних телефона, нетбук рачунара, таблет рачунара,...) временом је развила потребу за сталним приступом приватним подацима и документима. Самим тим складиштење приватних података и докумената на кућним стоним рачунарима временом је постало превазиђено, а као алтернатива појавило се рачунарство у облаку.

Коришћењем рачунарства у облаку могуће је складиштити личне податке на приватном удаљеном серверу, при том имајући могућност приступа тим подацима са било које локације на интернету, употребом било ког мобилног уређаја, што се у великој мери преклапа са наведеним тенденцијама. Поред великог броја комерцијалних решења, попут *Dropbox-a*, развијена су и многобројна "отворена" решења која корисницима на једноставан и интуитиван начин обезбеђују већу контролу над подацима. Једно од таквих "отворених" решења је и *OwnCloud*.

Поред основне функционалности, складиштења приватних података, *OwnCloud* нуди и низ других погодности, између осталих и вођење календара догађаја. Основни алат за коришћење свих сервиса које нуди *OwnCloud* је одговарајући веб портал. Поред дате апликације, постоји још доста других апликација (десктоп клијенти, мобилне апликације,...) које *OwnCloud* нуди. Такође, у понуди је и велики број софтвера са "отвореним" лиценцама, иза којих не стоји *OwnCloud* тим, а који нуде услуге које у основи користе *OwnCloud* сервисе. Сва та решења, било да су званичне *OwnCloud* апликације, било да су *3rd party* апликације, суочавају се са одређеним ограничењима. На пример, званични *OwnCloud* десктоп клијент има раздвојене, независне, верзије апликације за водеће оперативне системе (*Windows*, *Mac*, *Linux*) и нуди искључиво могућност синхронизације докумената, док, на пример, синхронизација календара догађаја није подржана. С тим у вези, развој мултиплатформског десктоп клијента који би омогућио синхронизацију календара догађаја, који је и тема овог рада, требало би да понуди решење за превазилажење наведених ограничења. Са друге стране, потенцијална употребна вредност таквог решења је још један од мотива за развој датог решења:

- евиденција планираних активности корисника са приказом одговарајућих обавештења,
- употреба у настави, нпр. приказ термина полагања испита,
- и било које друге активности које су у основи засноване на дељењу информација о догађајима/активностима између групе корисника система.

У наставку ће бити укратко описан садржај поглавља овог рада.

Поглавље *Преглед коришћених технологија* представља кратак опис технологија које су коришћене приликом развоја решења које је тема овог рада, док је нешто шири опис дат у поглављу *Радно окружење*.

Поглавље *ownCloud* укратко описује пројекат и апликацију чије сервисе дати десктоп клијент треба да користи, док је опис десктоп клијента и приказ кључних делова програмског кода представљен у поглављу *ownCloudCalendar*.

## Глава 2

# Преглед коришћених технологија

У глобалу, сва софтверска решења можемо поделити у две категорије. Једну групу чине комерцијална решења, која су заштићена власничким лиценцама, док другу групу чине софтверска решења са, у већој или мањој мери, "отвореним" лиценцама, при чему је програмски код обично отворен и доступан. У свету софтвера "отвореног" кода постоји више различитих типова лиценци, а неке од познатијих су BSD, GPL и MIT[1] лиценце. Када се говори о "отвореним" лиценцама мора се бити веома обазрив у смислу отворености и слободе коју лиценца као таква пружа. У складу са тим, потребно је напоменути да је решење које је тема овог рада у потпуности отворено и да је код доступан у целости.

Софтверски производи су временом постали све сложенији и све компликованији, што је довело до тога да један човек углавном не може сам да се бави имплементацијом неког софтверског решења у прихватљивом временском периоду, већ су на развоју софтвера најчешће ангажовани тимови људи. Из тог разлога, као нов изазов појавила се потреба за решењем које би омогућило да сви чланови тима могу паралелно да раде на развоју софтвера, не угрожавајући активности осталих чланова тима. Као одговор на наведени проблем, појавили су се различити алати за контролу изворног кода. Дужи временски период *Subversion* је био најзаступљенији алат за контролу изворног кода, али у последње време *Git*[2] преузима примат, јер је заснован на другачијим принципима, тако да више задовољава потребе корисника. Као такав *Git* је био погодан за коришћење и у овом раду заједно са слободним и бесплатним *Git* репозиторијумом *GitHub*[3], који поред простора који пружа, даје и неопходну статистику везану за број учесника на пројекту, њихову активност итд.

У суштини, подела софтверских решења се може вршити по различитим карактеристикама: оперативним системима за које су развијани, пословним процесима које покривају, технологијама које су коришћене у току развоја итд. Једна од битних подела, која се огледа у потпуно различитим концептима на којима су апликације засноване, јесте подела на десктоп и веб апликације. У смислу наведене поделе, решење које је тема датог рада представља десктоп апликацију. Постојање више различитих корисничких интерфејса или више верзија апликације за различите оперативне системе може се сматрати стандардом. С тим у вези, у процесу развоја софтвера, појавила се и потреба за поделом логике и архитектуре уређења самог софтверског решења. На овај начин обезбеђена је оптимизација програмског кода, као и могућност поновне употребе постојећег кода. У складу са наведеном поделом дато решење можемо посматрати као двослојну апликацију, где један слој представља сам кориснички интерфејс, а други слој је задужен за "комуникацију" са *OwnCloud* платформом.

Један од водећих изазова у развоју десктоп апликација је да се нађе начин за превазилажење ограничења која су изазавана оперативним системима на којима те апликације треба да раде. Разлике у концептима и техничким специфичностима које постоје међу водећим оперативним системима утицале су на то да десктоп апликације развијене за један оперативни систем не могу да раде на другим оперативним системима без одговарајућег

прилагођавања. Како би се ова ограничења превазишла, јавила се потреба за развојем платформи које би омогућиле да десктоп апликације без проблема раде на свим оперативним системима. Једна од таквих платфотми је и *XWT*[4], о којој ће бити више речено у поглављу 3. *Радно окружење*.

У развоју решења, које је тема датог рада, коришћене су и следеће готове компоненте са "отвореним" лиценцама:

- *Log4Net* - библиотека класа за логовање грешака,
- *DDay.iCal*[5] - библиотека класа за рад са календаром за окружење *.NET-a 2.0* и новије верзије,
- *CalDAV*[6] - протокол за синхронизацију календара, који је такође детаљније описан у поглављу 3. *Радно окружење*.

## Глава 3

# Радно окружење

Један од водећих принципа у процесу развоја софтверских решења јесте тежња да се програмски код подели у што више независних функционалних целина, које се касније на разне начине могу интегрисати. На овај начин креира се скуп библиотека које је могуће поново користити. Такође, овакав приступ олакшава развој и одржавање самог софтвера. Пре него што започне имплементацију неког софтверског решења програмер се суочава са низом избора које мора да направи:

- избор програмског језика - програм се обично реализује у једном програмском језику, бар на нивоу једне библиотеке,
- избор технологије у којој ће бити имплементиран кориснички интерфејс,
- избор платформе на којој ће се радити равној итд.

Као што је већ поменуто у секцији *Преглед коришћених технологија* *OwnCloud календар конектор* је десктоп апликација, писана у програмском језику *C sharp*, развијана на *XWT* платформи, и у делу који се односи на рад са календаром користи готове компоненте *DDay.iCal* и *CalDAV*.

### 3.1 XWT платформа

*XWT* је *.NET* мултиплатформски алат са "отвореном" лиценцом за развој апликација које могу да раде на различитим платформама, а да у основи користе заједнички код. Користи се најчешће за развој десктоп апликација које на овај начин раде на свим подржаним платформама, без потребе да се код прилагођава специфичностима сваке од њих. Разлика у односу на традиционални приступ у развоју десктоп апликација је у томе што се контроле исцртавају динамички, у самом коду, а сам *XWT API* има способност да у зависности од платформе изабере одговарајуће контроле. У наставку је дат пример кода где је приказано на који начин је креирана форма за пријаву на апликацију (Слика 3.1), коришћењем *XWT* алата:

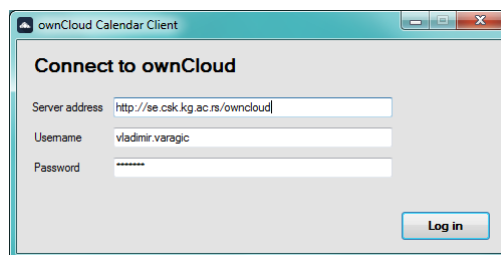
Листинг 3.1: Исцртавање форме за пријаву коришћењем *XWT* алата

```
1 using Xwt;  
2 using Xwt.Drawing;  
3  
4 public class LogIn : BaseForm  
5 {  
6     #region Private fields  
7  
8     Label lblTitle = new Label("Connect to ownCloud")  
9     {
```



```
10     Font = Font.SystemFont.WithWeight(Xwt.Drawing.FontWeight.Bold).WithSize
      (14)
11 };
12 Label lblServerAddress = new Label("Server address");
13 TextEntry txtServerAddress = new TextEntry();
14 Label lblUserName = new Label("Username");
15 TextEntry txtUsername = new TextEntry();
16 Label lblPassword = new Label("Password");
17 PasswordEntry txtPassword = new PasswordEntry();
18 Button btnLogIn = new Button("Log in");
19
20 #endregion Private fields
21
22 #region Constructors
23
24 public Login()
25 {
26     try
27     {
28         DrawControls();
29
30         btnLogIn.Clicked += delegate
31         {
32             btnLogIn_Click();
33         };
34     }
35     catch (Exception ex)
36     {
37         HandleException(ex);
38     }
39 }
40
41 #endregion Constructors
42
43 #region Nonvirtual methods
44
45 private void DrawControls()
46 {
47     lblTitle.MinWidth = 220;
48     AddChild(lblTitle, 13, 13);
49     AddChild(lblServerAddress, 13, 60);
50     txtServerAddress.WidthRequest = 278;
51     AddChild(txtServerAddress, 96, 57);
52
53     AddChild(lblUserName, 13, 90);
54     txtUsername.WidthRequest = 278;
55     AddChild(txtUsername, 96, 87);
56
57     AddChild(lblPassword, 13, 120);
58     txtPassword.WidthRequest = 278;
59     AddChild(txtPassword, 96, 117);
60
61     btnLogIn.MinWidth = 90;
62     btnLogIn.MinHeight = 30;
63     btnLogIn.BackgroundColor = Xwt.Drawing.Color.FromBytes(68, 187, 238);
64     AddChild(btnLogIn, 382, 170);
65 }
66
67 #endregion Nonvirtual methods
68 }
```

---



Слика 3.1: Форма за пријаву на систем

## 3.2 WebDAV

*WebDAV* представља проширење постојећег *HTTP* протокола. *WebDAV* протокол обезбеђује окружење које корисницима пружа могућност креирања, ажурирања, преузимања документа са удаљеног сервера. Значајно унапређење које је *WebDAV* протокол донео огледа се у количини података који могу да се преносе путем мреже. Раније је број датотека био ограничен на једну датотеку по упиту, а овим протоколом се омогућава преношење више датотека. Битно својство овог протокола је и то да нуди и контролу верзије података (нпр. одржавање информација о аутору документа, датуму и времену измене документа итд.).

За разлику од других протокола за пренос података (*FTP*, *SSH*) за које је потребно додатно отварање портова, *WebDAV* протокол користи стандардан *HTTP*-порт (обично 80), па додатне конфигурације на нивоу мреже нису потребне.

Што се тиче техничке позадине *WebDAV* протокола, он се састоји из скупа нових метода и заглавља постојећег *HTTP* протокола и највероватније је први протокол који користи *XML* језик. Методе које користи су:

- *PROPFIND* - користи се за читање особина ресурса као и евентуалне структуре истих,
- *PROPPATCH* - мења и брише више особина ресурса у једном кораку,
- *MKCOL* - креира нову колекцију,
- *COPY* - копира ресурс са једне на другу адресу (URI),
- *MOVE* - пребацује ресурс са једне на другу адресу (URI),
- *LOCK* - штити ("закључава") ресурс,
- *UNLOCK* - уклања заштиту ресурса.

Готови сви оперативни ситеми имају уграђену подршку за *WebDAV* протокол. У следећем примеру приказано је како се помоћу *WebDAV* протокола, коришћењем *PROPFIND* методе преузма вредност поља *displayName* из документа *test.eml*:

Листинг 3.2: Пример коришћења *PROPFIND* методе *WebDAV* протокола

```

1 using System;
2 using System.Net;
3 using System.IO;
4 using System.Text;
5 using System.Xml;
6
7 namespace ExchangeSDK.Snippets.CSharp
8 {
9     class GettingItemPropertyValuesWebDAV
10    {

```

```

11     [STAThread]
12     static void Main(string[] args)
13     {
14         // Variables.
15         System.Net.HttpWebRequest Request;
16         System.Net.WebResponse Response;
17         System.Net.CredentialCache MyCredentialCache;
18         string strSrcURI = "http://Server/public/TestFolder1/test.eml";
19         string strUserName = "UserName";
20         string strPassword = "!Password";
21         string strDomain = "Domain";
22         string strBody = "";
23         byte[] bytes = null;
24         System.IO.Stream RequestStream = null;
25         System.IO.Stream ResponseStream = null;
26         XmlDocument ResponseXmlDoc = null;
27         XmlNodeList DisplayNameNodes = null;
28
29         try
30         {
31             // Build the PROPFIND request body.
32             strBody = "<?xml version='1.0'>"
33                 + "<d:propfind xmlns:d='DAV:'><d:prop>"
34                 + "<d:displayname/></d:prop></d:propfind>";
35
36             // Create a new CredentialCache object and fill it with the
37             // credentials required to access the server.
38             MyCredentialCache = new System.Net.CredentialCache();
39             MyCredentialCache.Add( new System.Uri(strSrcURI),
40                 "NTLM",
41                 new System.Net.NetworkCredential(strUserName, strPassword,
42                     strDomain)
43             );
44
45             // Create the HttpWebRequest object.
46             Request = (System.Net.HttpWebRequest)HttpWebRequest.Create(
47                 strSrcURI);
48
49             // Add the network credentials to the request.
50             Request.Credentials = MyCredentialCache;
51
52             // Specify the method.
53             Request.Method = "PROPFIND";
54
55             // Encode the body using UTF-8.
56             bytes = Encoding.UTF8.GetBytes((string)strBody);
57
58             // Set the content header length. This must be
59             // done before writing data to the request stream.
60             Request.ContentLength = bytes.Length;
61
62             // Get a reference to the request stream.
63             RequestStream = Request.GetRequestStream();
64
65             // Write the request body to the request stream.
66             RequestStream.Write(bytes, 0, bytes.Length);
67
68             // Close the Stream object to release the connection
69             // for further use.
70             RequestStream.Close();
71
72             // Set the content type header.

```

```

71         Request.ContentType = "text/xml";
72
73         // Send the PROPFIND method request and get the
74         // response from the server.
75         Response = (HttpWebResponse)Request.GetResponse();
76
77         // Get the XML response stream.
78         ResponseStream = Response.GetResponseStream();
79
80         // Create the XmlDocument object from the XML response stream.
81         ResponseXmlDoc = new XmlDocument();
82         ResponseXmlDoc.Load(ResponseStream);
83
84         // Build a list of the DAV:href XML nodes, corresponding to the
85         // folders
86         // in the mailbox. The DAV: namespace is typically assigned
87         // the a:
88         // prefix in the XML response body.
89         DisplayNameNodes = ResponseXmlDoc.GetElementsByTagName("a:
90             displayname");
91
92         if(DisplayNameNodes.Count > 0)
93         {
94             // Display the item's display name.
95             Console.WriteLine("DAV:displayname property...");
96             Console.WriteLine(DisplayNameNodes[0].InnerText);
97         }
98         else
99         {
100             Console.WriteLine("DAV:displayname property not found...");
101         }
102
103         // Clean up.
104         ResponseStream.Close();
105         Response.Close();
106     }
107     catch(Exception ex)
108     {
109         // Catch any exceptions. Any error codes from the PROPFIND
110         // method request on the server will be caught here, also.
111         Console.WriteLine(ex.Message);
112     }
113 }

```

### 3.3 CalDAV

*CalDAV* протокол је проширење *WebDAV* протокола и представља интернет стандард који омогућава клијенту да приступи информацијама о планираним догађајима на удаљеном серверу. Користи *ICalendar*[7] формат података. Дозвољава истовремени приступ истим подацима од стране више клијената, чиме се омогућава кооперативно планирање и дељење информација. Дакле, *CalDAV* је клијент/сервер протокол за календар и планирање који омогућава корисницима приступ до календара на серверу и могућност планирања догађаја са другим корисницима сервера. У наставку је дат приказ кода за преузимање информација о догађајима са *OwnCloud* календара, коришћењем *CalDAV* протокола:

Листинг 3.3: Преузимање информација о догађајима са *OwnCloud* календара

```
1 public IICalendarCollection ownCloudCalendar_GetEvents(Uri server, string
   username, string password)
2 {
3     IICalendarCollection iCalCollection = null;
4     try
5     {
6         iCalCollection = iCalendar.LoadFromUri(server, username, password);
7     }
8     catch (Exception ex)
9     {
10        Logging.LogError(ex, LoggingCategories.Controller);
11        throw new LoggedException(ex);
12    }
13    return iCalCollection;
14 }
```

## Глава 4

# *OwnCloud* пројекат

Рачунарство у облаку представља скуп ресурса, чији је задатак да омогући складиштење велике количине података или извршавање великог броја процеса. Основна карактеристика рачунарства у облаку је та да корисницима омогућава коришћење удаљених ресурса, при чему им није дозвољен физички приступ датим ресурсима. Пораст броја корисника са оваквим захтевима утицао је и на појаву великог броја комерцијалних платформи које нуде услугу рачунарства у облаку, међу којима су *Amazon*, *Microsoft*, *Dropbox* и многе друге. Једно од таквих решења је и *ownCloud*[8] пројекат.

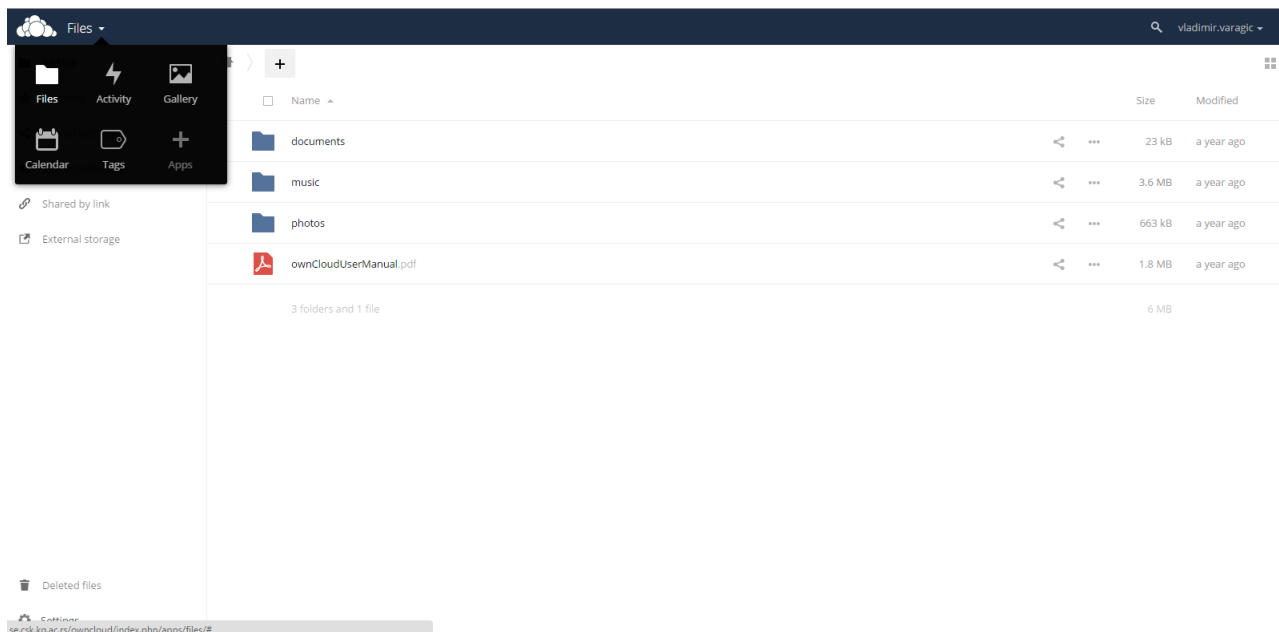
На почетку основна идеја *OwnCloud* пројекта је била да се обичном кориснику омогући да има приватно складиште на којем ће моћи да складишти своје податке. У међувремену, овај пројекат је добио много унапређења која нису директно везана за само складиштење података. Творац пројекта Франк Карличек је идеју о потреби решења са "отвореним" лиценцама изнео на скупу програмера и успео је да обезбеди неопходан број учесника који ће допринети развоју и популаризацији овог пројекта.



Слика 4.1: *OwnCloud* лого

Постоје три могућности за приступ подацима на *ownCloud-u*:

- Десктоп апликација - омогућава кориснику да складишти и/или преузима податке са удаљених ресурса,
- *WebDAV* технологија - погоднија од десктоп апликације, која се мора инсталирати на сваком рачунару, али са друге стране ограничава корисницима приступ само до података,
- Веб апликација - нема ограничења као друге две опције. Дакле, омогућен је приступ свим погодностима које *ownCloud* портал нуди и које ће бити представљене у наставку.

Слика 4.2: Кориснички интерфејс *ownCloud* веб апликације

Као што се може видети на Слици 4.2, неке од основних функционалности су:

- Приказ листе фајлова и директоријума тренутно пријављеног корисника,
- Могућност додавања нових садржаја,
- Могућност брзе претраге садржаја,
- Листа доступних апликација,
- Могућност приступа приватним подацима корисника, као и могућност одјаве.

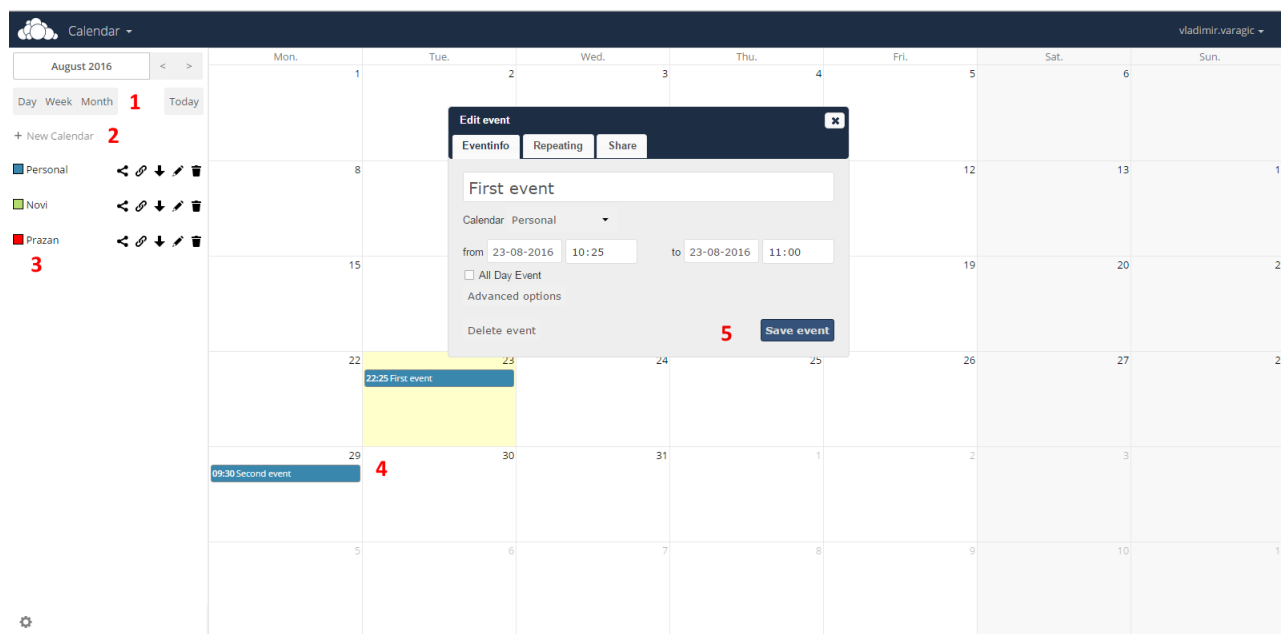
Поред могућности складиштења и приступа подацима *ownCloud* веб апликација нуди могућност коришћења њених подапликација:

- Вођење листе контаката,
- Дељење корисничких података између корисника истог складишта,
- Праћење активности корисника,
- Листа доступних апликација,
- Праћење календара, итд.

Посебна пажња биће посвећена *Calendar* подапликацији.

## 4.1 *Calendar* подапликација

Као што је већ раније поменуто, основна идеја и функција *ownCloud* апликације је била да омогући складиштење података корисника, што није било довољно атрактивно. Увођењем платформе за креирање нових сервиса (подапликација) овај проблем је превазиђен. Један од њих је и *Calendar* подапликација.

Слика 4.3: Кориснички интерфејс *ownCloud Calendar* подапликације

На Слици 4.3 су приказане све њене функционалности:

1. Начин приказа календара (дневни, недељни, месечни),
2. Креирање новог календара,
3. Приказ постојећих календара,
4. Приказ постојећих догађаја,
5. Администрација (унос, измена, брисање) догађаја.

Подапликација *Calendar* је прилично једноставна и интуитивна за коришћење.



## Глава 5

# *OwnCloud* календар конектор

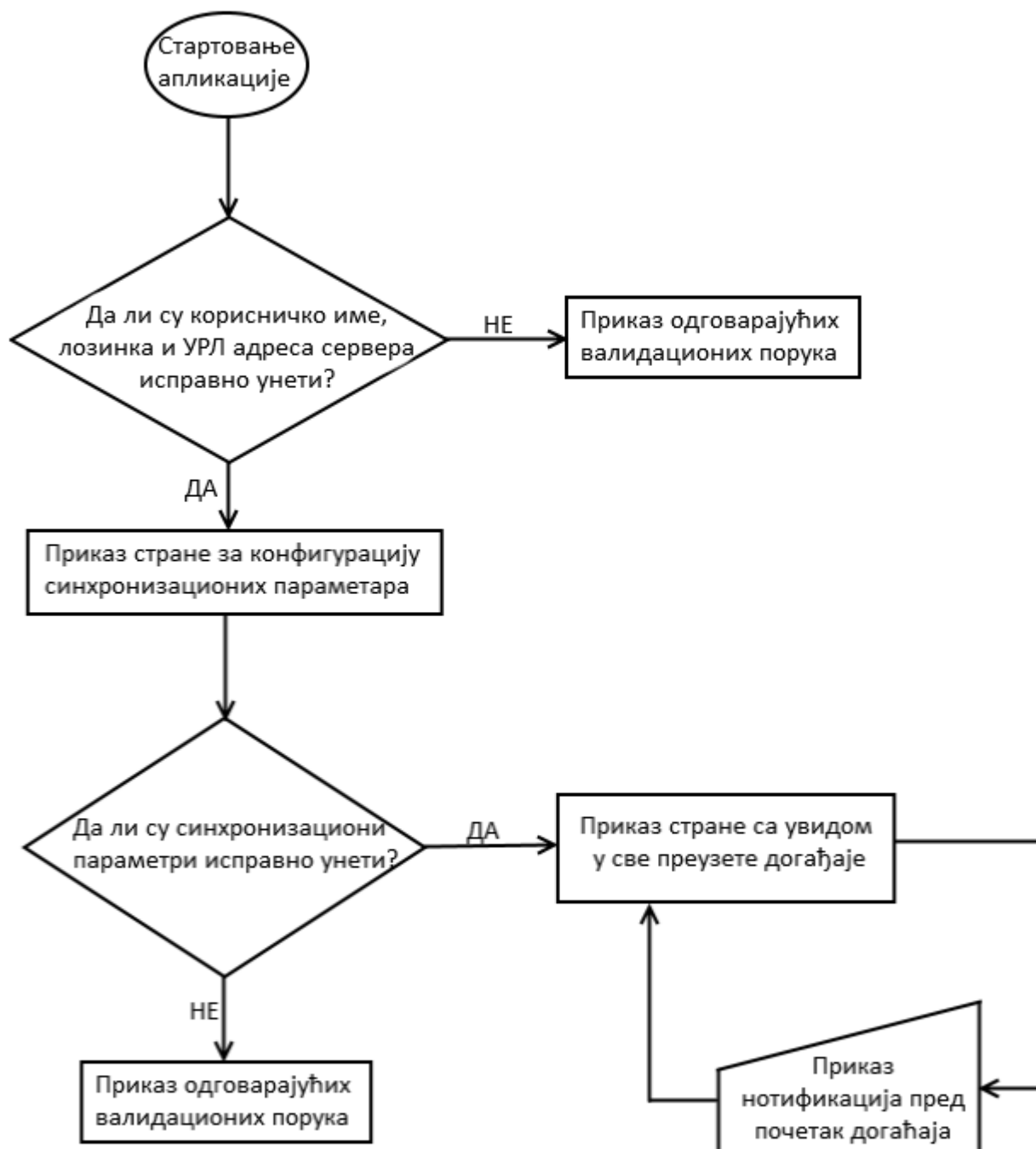
У претходним поглављима описани су основни концепти технологија и окружења који су коришћени у развоју датог пројекта, са циљем да се читаоцу омогући да формира слику комплетног, заокруженог решења. Сам пројекат, који је тема овог рада, може се посматрати као део тог решења. У овом поглављу фокус ће бити постављен на појашњења неких делова његове имплементације.

### 5.1 Жељене функционалности

Актуелна, званична верзија *ownCloud* десктоп клијента обезбеђује само синхронизацију докумената који се налазе на *ownCloud* платформи. Основни циљ овог пројекта јесте да се развије решење, у виду мултиплатформског десктоп клијента, које би омогућило преузимање информација о креираним догађајима на *ownCloud* календару и приказ одговарајућих обавештења. Апликација има следећи скуп функционалности:

- синхронизација догађаја на захтев,
- аутоматска синхронизација догађаја,
- могућност управљања аутоматском синхронизацијом (потребна/није потребна, дефинисање временског интервала након којег ће се стартовати,...),
- преглед преузетих догађаја,
- приступ делу за администрацију догађаја на веб порталу *ownCloud* платформе,
- приказ одговарајућег обавештења, непосредно пре почетка неког догађаја.

Ток активности које треба да обезбеде ове функционалности описан је на дијаграму 5.1.

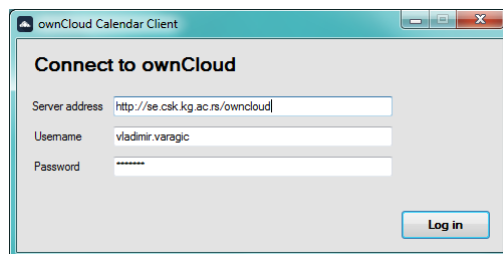


Слика 5.1: Дијаграм тока активности

На основу приказаног алгоритма може се стећи јасна и потпуна слика о начину рада саме апликације. У наставку ће бити детаљније објашњене неке интересантније функционалности и биће приказани делови програмског кода, док се комплетан код пројекта може погледати на одговарајућем репозиторијуму[9].

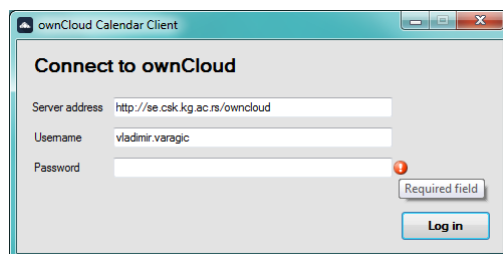
### 5.1.1 Аутентификација

Аутентификација корисника на веб портал *ownCloud* платформе одрађена је коришћењем класа *WebClient*, *NetworkCredential* које су саставни део *.NET Framework-a*. Подаци унети на форми за пријаву на систем (Слика 5.2), која се приказује након стартовања апликације, се прослеђују на верификацију.



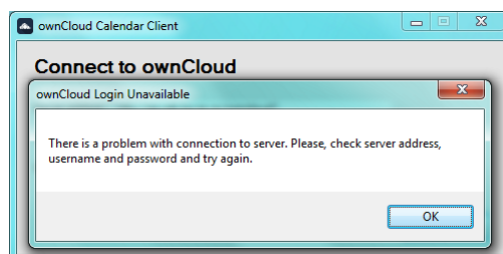
Слика 5.2: Форма за пријаву на систем

Сви подаци на форми за пријаву су обавезни, па се у случају да неки податак није унет, прикаже одговарајући индикатор (Слика 5.3).



Слика 5.3: Форма за пријаву на систем

Такође, у случају да неки од података који се уносе приликом пријаве на апликацију (адреса сервера, корисничко име или лозинка) није исправан приказује се одговарајућа порука (Слика 5.4).

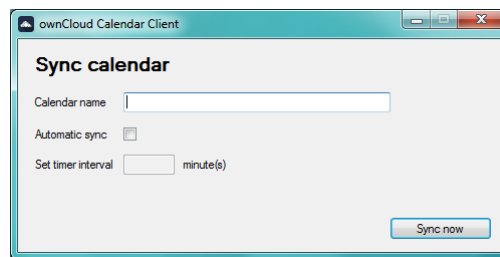


Слика 5.4: Форма за пријаву на систем

У супротном, ако су сви подаци исправни, корисник успешно приступа апликацији и приказује му се форма за синхронизацију догађаја са *ownCloud* календара.

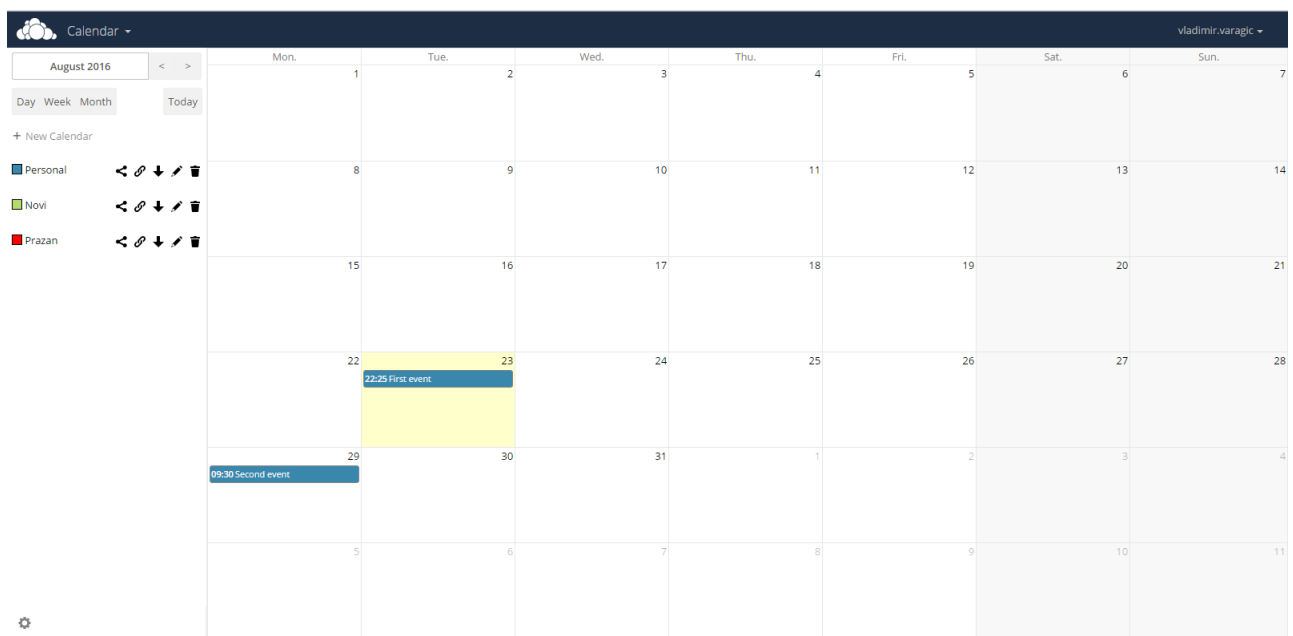
### 5.1.2 Синхронизација догађаја на захтев

Као што је већ наведено у поглављу 5.1.1 *Аутентификација*, након успешног приступа апликацији кориснику се приказује форма за конфигурацију синхронизације (Слика 5.5).



Слика 5.5: Синхронизација догађаја са *ownCloud* календара

*OwnCloud* платформа омогућава кориснику да на порталу води више различитих календара тј. да календар дели у различите категорије.



Слика 5.6: *OwnCloud* календар

Са друге стране, синхронизацијом се у једном тренутку могу преузети само догађаји који су везани за једну категорију, тако да је назив календара обавезан податак приликом синхронизације. Такође, приликом покретања синхронизације ради се валидација исправности назива календара и уколико не постоји календар са унетим називом кориснику се прикаже одговарајућа порука. У супротном, ако је унет исправан назив календара, кориснику се приказују догађаји који постоје на наведеном календару. Сам приказ података о догађају биће детаљно описан у секцији 5.1.4 *Преглед перузетих догађаја*.

У наставку је дат приказ метода за синхронизују података:

Листинг 5.1: Методе за синхронизацију догађаја са *ownCloud* календара

```
1 private void btnSyncCalendar_Click(object sender, EventArgs e)
2 {
3     try
4     {
5         if (ValidateControls())
6         {
```

```

7         IICalendarCollection iCalCollection = GetCalendarEventsData();
8
9         if (iCalCollection == null)
10        {
11            ShowMessage("There is no calendar with the name " + txtCalendarName.
12                        Text.Trim(), "ownCloud Calendar Sync Unavailable");
13        }
14        else
15        {
16            Hide();
17            int? syncTimerInterval = null;
18            if (!String.IsNullOrEmpty(txtTimerInterval.Text))
19            {
20                syncTimerInterval = Convert.ToInt32(txtTimerInterval.Text);
21            }
22            EventsList eventsList =
23                new EventsList(iCalCollection, cbAutomaticSync.Checked,
24                              syncTimerInterval, txtCalendarName.Text, serverUrl, username,
25                              password, serverAddress);
26            eventsList.ShowDialog();
27            if (eventsList.IsHidden)
28            {
29                HideForm();
30            }
31            else
32            {
33                Show();
34            }
35        }
36    }
37    }
38    }
39    }
40
41    private IICalendarCollection GetCalendarEventsData()
42    {
43        owdCloudCalendarConnector connector = new owdCloudCalendarConnector();
44
45        string url = serverAddress
46                    + cCalDavUrlExtension
47                    + username
48                    + cCalDavUrlExtensionSlash
49                    + txtCalendarName.Text.Trim().ToLower()
50                    + cCalDavUrlExtensionExport;
51        serverUrl = new Uri(url);
52
53        return connector.ownCloudCalendar_GetEvents(serverUrl, username, password)
54            ;
55    }

```

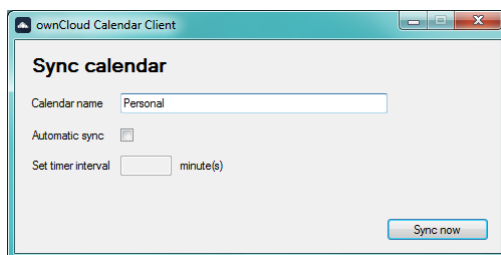
### 5.1.3 Аутоматска синхронизација догађаја

Поред наведене функционалности за синхронизацију догађаја на захтев, омогућена је и функционалност аутоматске синхронизације догађаја. Уколико корисник жели да користи дату функционалност, потребно је да чекира опцију *Automatic sync* на форми за синхронизацију догађаја (Слика 5.5). Када је опција *Automatic sync* чекирана, податак *Sync time interval* је обавезан. Дакле, након дефинисања наведених података, апликација ће аутоматски синхронизовати догађаје са одговарајућег календара у наведеном временском

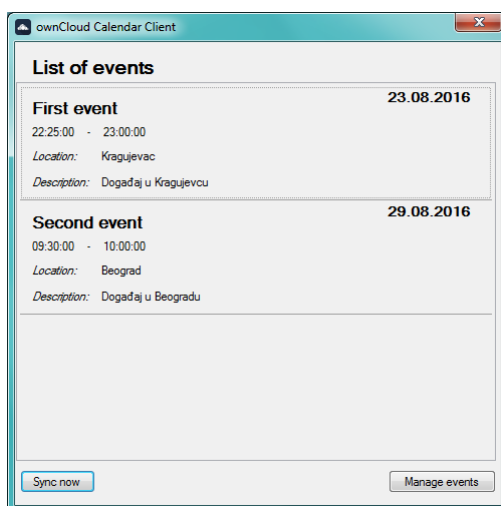
интервалу. Временски интервал се дефинише у минутима и одговарајућом валидацијом онемогућено је да вредност овог податка буде било шта што није цео позитиван број.

#### 5.1.4 Преглед преузетих догађаја

Када су сви обавезни подаци исправно унети, било да је у питању синхронизација догађаја на захтев, било да је у питању аутоматска синхронизација догађаја, кликом на дугме *Sync now* (Слика 5.7) пружају се догађаји са одговарајућег календара и прикаже се форма са листом догађаја (Слика 5.8).



Слика 5.7: Синхронизација догађаја са *ownCloud* календара



Слика 5.8: Приказ преузетих догађаја

Са форме за Преглед преузетих догађаја (Слика 5.8) корисник у сваком тренутку може поново да покрене синхронизацију догађаја (кликом на дугме *Sync now*), без обзира на то да ли је функционалност аутоматске синхронизације изабрана или не. Корисник, такође, има могућност да са форме за Преглед преузетих догађаја (Слика 5.8), кликом на дугме *Manage events* приступи календару на веб порталу *ownCloud* платформе (Слика 5.6) и администрира (креира нове, ажурира постојеће, брише) догађаје.

#### 5.1.5 Приказ нотификација

Поред описаних функционалности апликација има још једну, вероватно најзанимљивију функционалност, а то је приказ одговарајућих нотификација у вези са преузетим догађајима. Нотификације се приказују према унапред дефинисаним параметрима:

- први параметар (*notificationMessageTimerInMinutes*) представља временски период (у минутима) којим се дефинише колико минута пре стартовања догађаја приказати одговарајућу нотификацију,

- други параметар (*notificationPingTimeInterval*) представља vremeski период (у милисекундама) којим се дефинише колико често ће се проверавати да ли је први параметар достигао дефинисану вредност.

Ови параметри су конфигурабилни и део су конфигурационог фајла:

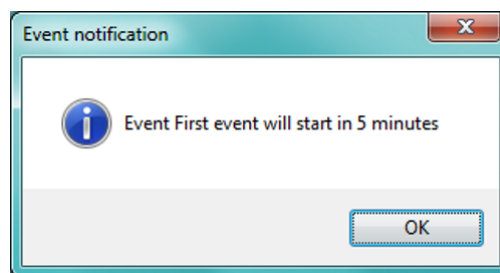
Листинг 5.2: Конфигурациони фајл

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <configuration>
3   <appSettings>
4     <add key="notificationPingTimeInterval" value="60000" />
5     <add key="notificationMessageTimerInMinutes" value="5" />
6   </appSettings>
7   <startup>
8     <supportedRuntime version="v4.0" sku=".NETFramework,
9       Version=v4.5" />
10  </startup>
11 <runtime>
12   <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
13     <dependentAssembly>
14       <assemblyIdentity name="System.Web.Mvc" publicKeyToken="31
15         bf3856ad364e35" culture="neutral" />
16       <bindingRedirect oldVersion="0.0.0.0-5.1.0.0" newVersion="
17         5.1.0.0" />
18     </dependentAssembly>
19   </assemblyBinding>
20 </runtime>
21 </configuration>

```

Дакле, у складу са дефинисаним вредностима наведених параметара, апликација сваког минута проверава да ли постоји догађај који ће стартовати за 5 минута и у случају да такав догађај постоји, кориснику се прикаже одговарајућа нотификација (Слика 5.9).



Слика 5.9: Приказ нотификације

У наставку је дат приказ методе којом је дата функционалност имплементирана:

Листинг 5.3: Метода за приказ нотификација

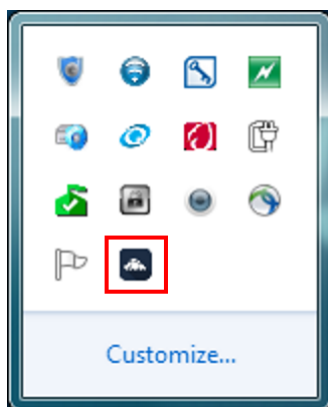
```

1 private void SetNotificationTimer()
2 {
3   Timer timer = new Timer();
4   int pingTimeInterval;
5
6   pingTimeInterval = Convert.ToInt32(ConfigurationManager.AppSettings["
7     notificationPingTimeInterval"].ToString());
8   timer.Tick += new EventHandler(CheckEventStartTime);
9 }

```

```
8
9   timer.Interval = pingTimeInterval;
10  timer.Start();
11  CheckEventStartTime(this, null);
12 }
```

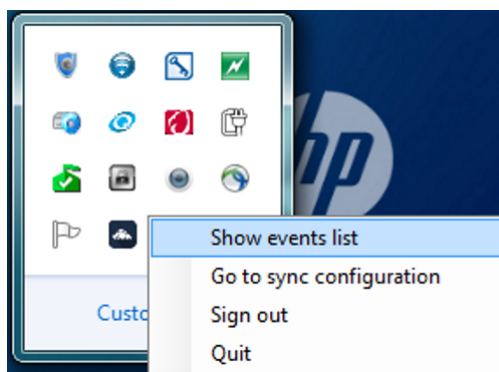
Поред функционалности описаних у претходним секцијама, споменућемо још нека својства апликације. Најпре, треба нагласити да је апликација *Single instance*, односно у једном тренутку могуће је покренути само једну инстанцу апликације. У случају да корисник покуша да покрене више инстанци апликације, то му неће бити дозвољено и приказаће се одговарајућа порука. Такође, требало би напоменути да се кликом на дугме *Close* на форми за Приказ преузетих догађаја апликација не затвара, већ се само минимизује, тј. апликација је и даље покренута и иконица апликације налази се у таскбару (Слика 5.10).



Слика 5.10: Таскбар - приказ иконице

Десни клик на иконицу у таскбару нуди следеће опције (Слика 5.11):

- отварање форме за приказ преузетих догађаја,
- отварање форме за унос параметара синхронизације,
- одјава са апликације и приказ форме за пријаву,
- затварање апликације.



Слика 5.11: Додатне опције



## 5.2 Идеје за даљи развој

Иако је функционално исправна, постојећу верзију апликације треба посматрати само као полазни корак у развоју коначног производа. Актуелна верзија апликације има својеврсна ограничења условљена коришћеним API-има (нпр. немогућност синхронизације више календара истовремено). Унапређења датих API-а или појава нових утицали би на то да се појави потреба за имплементацијом додатних или изменом постојећих функционалности. Са друге стране, постојећа верзија се такође може унапредити на више начина:

- побољшање корисничког интерфејса,
- предефинисани прикази догађаја (за разлику од актуелног приказа свих догађаја),
- ...

# Библиографија

- [1] BSD, GPL и MIT лиценце, <http://producingoss.com/en/license-choosing.html>
- [2] GIT, <https://git-scm.com/>
- [3] Github, <https://github.com/>
- [4] XWT платформа, <https://github.com/mono/xwt/tree/master/Xwt/Xwt>
- [5] DDay.iCal - an iCalendar class library, <https://sourceforge.net/p/dday-ical/wiki/Home/>
- [6] CalDay, <http://caldav.calconnect.org/index.html>
- [7] ICalendar, <https://www.calconnect.org/resources/calendaring-standards#iCalendar>
- [8] Званична страница *OwnCloud* пројекта, <http://owncloud.org/>
- [9] Репозиторијум *ownCloud Calendar Synchronization* апликације, <https://own-cloud-calendar.googlecode.com/svn>