



Ministerul Educației Naționale și Cercetării Științifice
Universitatea OVIDIUS Constanța
Facultatea de Matematică și Informatică Specializarea Informatică

Aplicație desktop pentru managementul proiectelor

Lucrare de licență

Coordonator științific

Lect. univ. Dr. Alexandrescu Adrian

Absolvent

Ababei Andrei

Constanta 2016

Aplicație desktop pentru managementul proiectelor

Coordonator științific

Lect. univ. Dr. Alexandrescu Adrian

Absolvent

Ababei Andrei

Cuprins

Lista figurilor	1
Capitolul 1 - Introducere	2
Motivație	2
Introducere	3
Agile	3
Waterfall	3
Tema proiectului	4
Utilizatori	4
Sistemul de task-uri	5
Ansamblul soluțiilor cunoscute	6
Team Foundation Server	6
Asana	8
Capitolul 2 – Tehnologii utilizate	9
Tehnologii utilizate	9
Prezentare generală	12
Capitolul 3 – Utilizarea sistemului	13
Instalarea aplicației	13
Instalarea back-end-ului	13
Instalarea front-end-ului	16
Intrarea în aplicație	16
Crearea utilizatorilor	17
Crearea proiectelor	17
Configurarea aplicației	17
Utilizare aplicației	17
Dashboard	18
Timeline	18
Formularul de modificare task	19
Linia de stare	22
Administrare	22
Capitolul 4 – Dezvoltarea sistemului	26
Dezvoltarea aplicației	26
Back-end	26
Front-end	30
Concluzii	36
Bibliografie	37

Lista figurilor

Figură 1 - Model Waterfall și Agile.....	4
Figură 2 - Structurarea proiectelor Agile.....	6
Figură 3 - Pagina Task-uri Team Foundation Server.....	7
Figură 4 - Interfață Asana	8
Figură 5 - Diagrama librăriilor și a aplicațiilor.....	12
Figură 6 - Instalare IIS.....	14
Figură 7 - Adăugare site în IIS Manager.....	15
Figură 8 - Stare server	16
Figură 9 - Selector proiecte	17
Figură 10 - Sistemul de task-uri în mod timeline.....	18
Figură 11 - Task marcat cu un check ce arată că a fost terminat	19
Figură 12 - Buton copiere nume task	20
Figură 13 - Fereastră modificare task	21
Figură 14 - Bară de stare	22
Figură 15 - Clase cerere și răspuns	26
Figură 16 - Clasă de răspuns pentru preluarea proiectelor.....	27
Figură 17 - Diagramă clase generate	28
Figură 18 - Controlul de antet.....	30
Figură 19 - Model controale panou principal	31
Figură 20 - Clasele cerere și răspuns pentru preluarea backlog-urilor.....	32
Figură 21 - Control-ul de Timeline	33

Capitolul 1 - Introducere

Motivație

Management-ul sau “arta de înfăptui ceva împreună cu alți oameni” [1], desemnează funcțional o activitate, ca de exemplu, management-ul timpului sau management-ul comenzilor. In alte cuvinte, management-ul asigura desfășurarea eficientă a activităților si urmărește atingerea unui nivel maxim de rezultate prin folosirea optimă a resurselor.

Un sistem de management reprezintă un ansamblu de procese, echipamente si date, destinate sa furnizeze informații si sa facă legătura între sistemul de conducere si sistemul condus.

Tema lucrării mele de licență este denumita “Aplicație desktop pentru managementul proiectelor” și reprezintă rezultatul perseverenței mele atât în ceea ce privește munca depusă la birou cât și în ceea ce privește gestionarea optima a task-urilor. Facultatea mi-a clădit un bagaj de cunoștințe destul de puternic. Planul managerial al domnului decan ce sugera implicarea studenților în proiecte extra curriculare și inserarea lor în domeniul muncii, m-a determinat sa mă angajez la o firma de programare numita Solution-Center. Mi-a fost puțin greu la început să mă obișnuiesc cu programul, cu task-urile cu termene limita, cu munca în echipa. Astfel, am fost inspirat să creez o aplicație care sa „controleze” task-urile atribuite fiecărui membru al echipei astfel încât să se prevină depășirea termenului limită impus, dar și utilizarea resurselor într-un mod inteligent.

Prezenta lucrare cuprinde în detaliu aspecte atât teoretice cât și practice privind modalitățile de concepere si implementare a sistemului împreună cu diagrame de clase și de model, snippet-uri¹ de cod chiar și imagini sugestive în locurile în care am crezut că este necesar. Lucrarea este structura în 4 capitole, fiecare conținând subcapitole, aranjate într-o ordine cât mai sugestive ce urmează un flux logic fiind ușor de urmărit, ce enunța într-un mod academic proiectarea, tehnologiile utilizate, implementarea aplicației și utilizarea ei.

Țin să le mulțumesc profesorilor pentru răbdarea de care au dat dovada pe parcursul timpului, pentru îndrumarea profesionala și pentru susținerea morala ce m-au ajutat să trec cu succes peste toate greutățile întâmpinate.

¹ Snippet – scurtă secvență de cod reutilizabilă

Introducere

Managementul proiectului cuprinde și înglobează toate sarcinile ce țin de analiza, planificarea, organizarea și monitorizarea tuturor sarcinilor ce trebuie îndeplinite pentru a duce un proiect la bun sfârșit într-un timp optim și folosind resurse minime. Cele mai comune faze de proiectare, numite și ciclul de viață al proiectelor, sunt:

- Inițializarea proiectului
- Planificarea proiectului
- Execuția proiectului
- Monitorizarea proiectului
- Finalizarea proiectului.

Cele mai cunoscute maniere de management al proiectelor sunt **Agile** și **Waterfall model**, modele ce stau la baza funcționării aplicației. În continuare voi face o scurtă descriere pentru fiecare model.

Agile

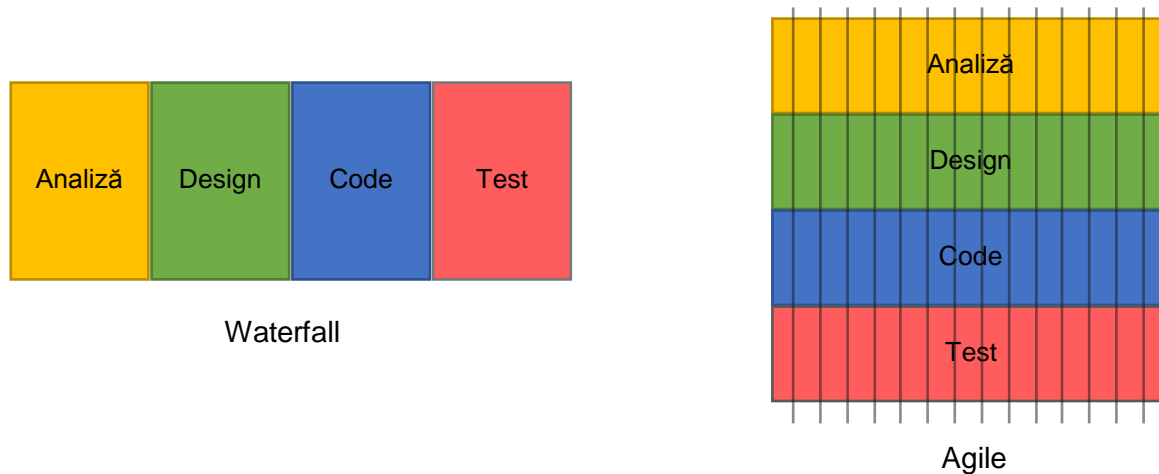
Agile este un set de principii, în general, pentru proiecte software, bazată pe dezvoltarea progresivă și promovează organizare pe modelul *Divide et impera*, în sensul în care impune divizarea unei probleme în subprobleme mici și planificarea lor pe durate scurte mai departe problemele vor fi împărțite în iterații (*sprints*). Datorită acestei caracteristici, la sfârșit de zi fiecare participant la proiect poate să prezinte progresul făcut cu toate problemele ce le-a preluat în ziua respectivă, iar la sfârșitul fiecărei iterații i se poate prezenta clientul-ului progresul făcut, împreună cu o versiune stabilă (*dar neterminată*) a proiectului, și poate fi informat de costurile și schimbările ce vor avea loc în următoarele iterații. [2]

Waterfall

În contrast cu **Agile**, “*Modelul Cascadă*” este o secvență de procese ce implică opt stagii :

- Conceptul
- Inițierea
- Analiza
- Designul
- Construcția
- Testarea
- Implementarea
- Mentenanța

Principalul dezavantaj, față de agile, este imposibilitatea întoarcerii la un pas anterior fără a fi nevoie de a se aplica modificare stagiilor superioare. Mai jos putem observa diferența dintre cele două modele (Figură 1).



Figură 1 - Model Waterfall și Agile

Tema proiectului

În această lucrare voi prezenta demersul meu pentru realizarea unei interfețe și a unui sistem independent pentru managementul proiectelor folosind modelul **Agile**. Aplicația este numită **WProject** este de fapt împărțită în două aplicații :

- **Front-End**
Aplicația desktop practic se comportă ca un client, se conectează și comunica cu aplicația de back-end.
- **Back-End**
În aplicația de back-end (*denumită și **Dispecer***) se procesează toate cererile trimise de pe toate front-end-urile. În dispecer se află toată logica de business și se fac majoritatea operațiilor de comunicare (notificări, chat, acțiuni la nivel de front-end, etc..)

Aplicația WProject este o soluție ideală pentru proiecte mici și medii deoarece este o aplicație ușor de instalat și utilizat suportând mai multe proiecte și gruparea utilizatorilor pe proiecte. Utilizatorii având posibilitatea comunicării în privat sau o într-un grup folosind o cameră de chat.

Utilizatori

Utilizatorii sunt împărțiți în două grupe principale

- **Administratorii**
Administratorii sunt practic arhitecții de proiect ce au toate drepturile de acces peste proiectele ce le aparțin. Avem ca exemplu drepturi:
 - Poate crea proiecte noi
 - Poate crea grupuri noi (modifică/șterge cele existente)

- Poate crea categorii și iterații noi
- Poate altera/șterge categoriile și iterațiile din proiectele ce îi aparțin
- Are acces peste toate task-urile și backlog-urile din proiectele ce îi aparțin
- Poate crea și suspenda utilizatori
- Poate include și exclude utilizatori din proiectele sale
- Poate transfera proiectele sale altor utilizatori
- **Utilizatorii**
 Utilizatorii au drepturi în funcție de grupurile în se află sau ce drepturi explicite au primit de către administratori.
 - Poate crea backlog-uri și task-uri noi (modifică/șterge task-urile și backlog-urile ce îi aparțin)
 - Își poate modifica propriile preferințe
 Pot exista și grupe secundare create de administratori pentru a ușura atribuirea de drepturi:
 - Developers
 - Testers
 - Support

Sistemul de task-uri

Task-ul este entitatea principală și unitatea de măsură ce arată ce și cât a lucrat utilizatorul într-o zi. El conține informațiile principale ce indică utilizatorului ce are de făcut. Câteva din aceste informații sunt :

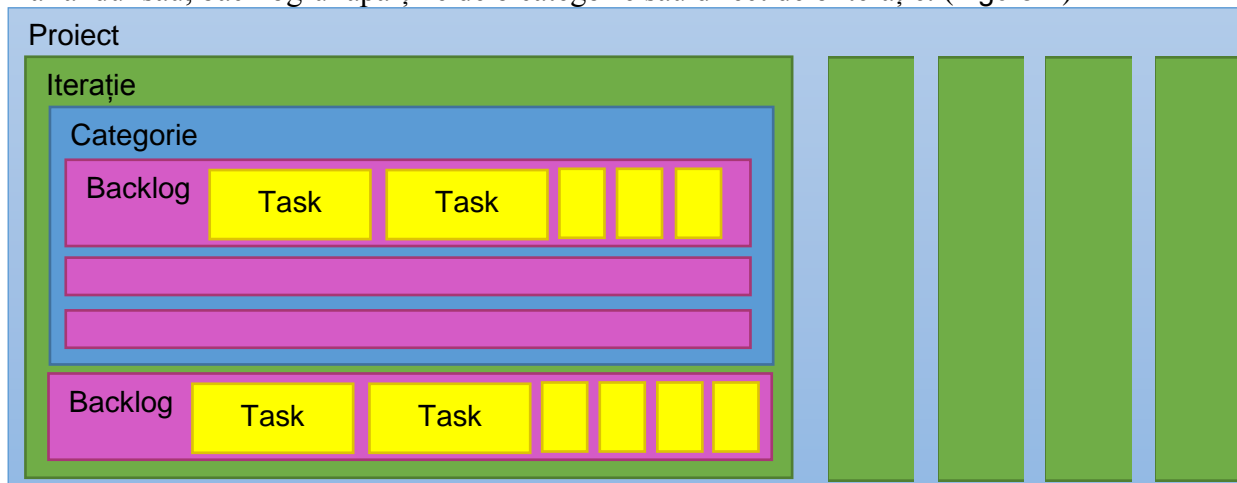
- Titlul – care de obicei se referă, într-un mod foarte generic, ce trebuie făcut.
- Descrierea – informațiile esențiale explicate în detaliu necesare pentru a închide task-ul²
- Fișiere atașate – la nivelul task-ului se pot atașa și fișiere ce pot ajuta la completarea task-ului (*ex. Documente, Imagini, etc...*)
- Starea task-ului – un task poate avea diferite stări :
 - To do – task-ul încă nu a fost început
 - In progress – task-ul a fost început și se lucrează la el
 - Done – task-ul a fost terminat cu succes
 - Removed – task-ul a fost anulat din anumite motive
- Owner (*proprietarul task-ului*) – utilizatorul care lucrează la task
- Prioritate – prioritatea task-ului (între 1 și 10, 1 fiind cel mai urgent)
- Timp rămas estimat (ETA)– timpul rămas estimat, în minute, până la terminarea task-ului.

Fiecare task aparține de un **backlog**, un backlog este o colecție de task-uri ce descrie o funcționalitate a produsului el poate fi de două feluri :

- Funcționalitate
- Bug (*problemă*)

² A închide un task – a termina un task

De obicei are aproximativ 3-4 task-uri: analiză, implementare, testare, publicare (*opțional*)
La rândul său, backlog-ul aparține de o categorie sau direct de o iterație. (Figură 2)



Figură 2 - Structurarea proiectelor Agile

Ansamblul soluțiilor cunoscute

Tema de licență propusă se numește „Aplicație desktop pentru managementul proiectelor”, aplicație ce înglobează un sistem complet pentru managementul task-urilor și comunicarea efectivă între membri proiectului. Aplicația a fost proiectată pentru proiecte de tip dezvoltare software și web, dar aceasta poate fi ușor extinsă și pentru alte tipuri de proiecte din alte domenii.

După alegerea temei de licență, înaintea începerii proiectării, am început să fac o analiză asupra aplicațiilor din același domeniu, urmărind punctele slabe ale altor aplicații, modul de utilizare, manierele de lucru cu proiectele și ce aş putea aduce ca element inovator.

Team Foundation Server

Team Foundation Server este un produs dezvoltat de Microsoft, inclus în Visual Studio, ce oferă pe lângă managementul proiectelor oferă și controlul versiunilor³ proiectului, raportare, managementul resurselor, testare și managementul release-urilor⁴. Principalele avantaje în utilizarea TFS-ului (Team Foundation Server) sunt :

- Înglobarea tuturor aplicațiilor într-o singură soluție
- Raportare
- Dashboard pentru întâlniri de tip SCRUM⁵

³ Controlul versiunilor – aplicație ce se ocupă cu controlul versiunilor (numite revizii) al codului sursă al unui program sau altor tipuri de fișiere.

⁴ Managementul release-urilor – proces de planificare și programare a lansărilor de noi versiune, ce include testarea pe diferite sisteme [8]

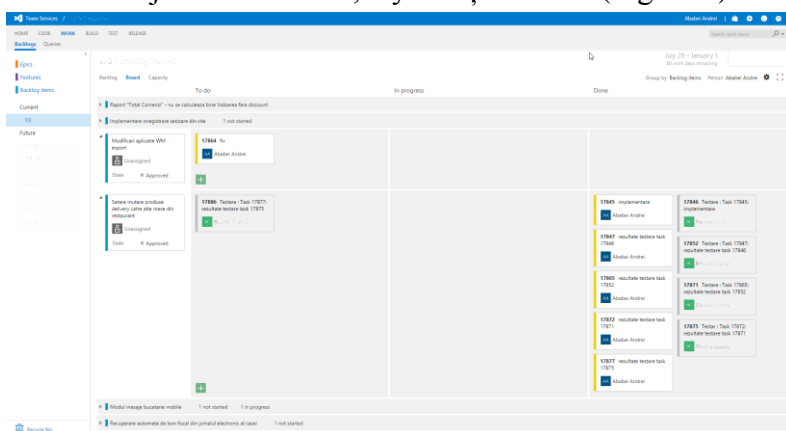
⁵ SCRUM – întâlnire a membrilor unui proiect o un interval propus pentru revizia task-urilor de la ultimul scrum și discuție pentru următoarele etape de dezvoltare.

- Unelte pentru testare
- Extensii de tip plug-in⁶
- Posibilitate utilizării serviciului în cloud
- Gratuit (până în 5 utilizatori)

Pe lângă avantajele impresionante, suita Team Foundation Server prezintă și o serie de dezavantaje :

- Urmărirea greoaie în cazul proiectelor mari
În cazul unui proiect mare, unui manager de proiect îi va îngreuna puțin urmărirea proiectelor de deoarece backlog-urile nu pot fi împărțite pe categorii, ci doar în iterații.
- Imposibilitatea setării zilei și interval orar pentru task-uri [3]
Unui task nu i se poate seta un interval orar ce determină timpul de completare (estimat) al acestuia. Acest lucru este de foarte mulți utilizatori ai aplicației și ar fi foarte util pentru urmărirea progresului pentru ziua curentă.
- Limitarea setării priorității taskurilor la 5 (1~5)
- Lipsa notificărilor și al mesajelor private
Deși aplicația suportă chat la nivel global și notificări prin email, aceasta nu oferă mesaje private (sau de grup) și nici notificări la nivelul aplicației.
- Prețul
În cazul în care într-un proiect sunt mai mult de 5 membri, utilizarea suitei va impune costuri începând de la 30 \$ pe lună cu doar 10 membri și servicii basic, pentru o echipa mai mare, și servicii în plus prețul poate ajunge ușor și la 500 \$ pe lună.

Aplicația prezintă un design ușor și simplist, fapt pentru care am urmat modelul și în aplicația mea cu mici ajustări de culori, layout⁷ și fonturi. (Figură 3)



Figură 3 - Pagina Task-uri Team Foundation Server

⁶ Plug-in (sau extensie) – componentă software ce extinde o aplicație cu noi funcționalități sau chiar reparând anumite defecțiuni și probleme de securitate.

⁷ Layout – Mod de aranjare a elementelor grafice și a elementelor vizuale într-o pagină

Asana

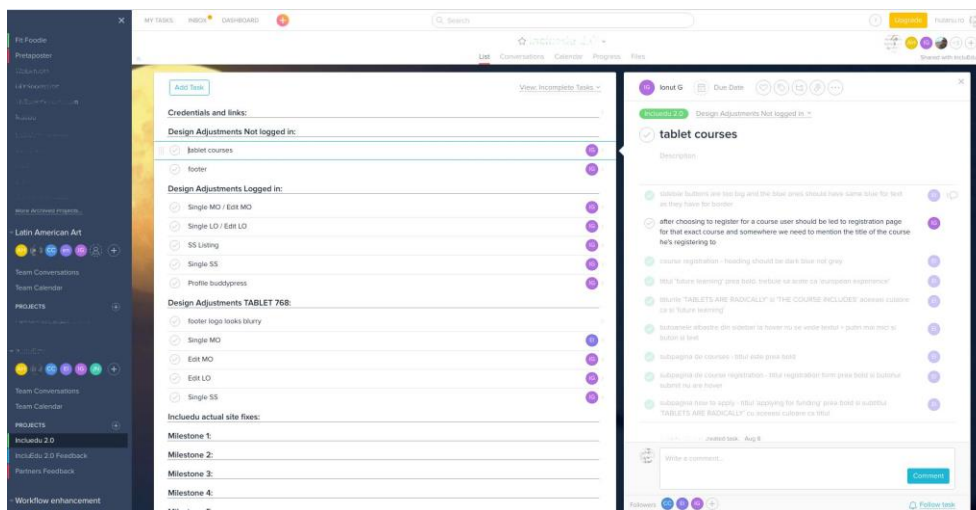
Asana este o aplicație web (Figură 4) și mobile pentru managementul task-urilor ce permite vizualizarea facilă, crearea, modificarea sau ștergerea task-urilor. Planificarea taskurilor se face utilizând conceptul de milestone ce cuprinde o colecție de task-uri. Avantajele utilizării platformei sunt :

- Aplicație nativă pentru Android și iOS
- Toate modificările asupra unui task se notifică printr-un email către utilizatorul asignat
- Posibilitatea atribuirii unui task o dată de deadline, iar la apropierea deadline-ului utilizatorul asignat task-ului va primi periodic email la o săptămână, la 3 zile și în ziua respectivă.
- Atașamente media de tip imagini și video-uri încorporate
- Utilizatorii menționați în comentariile task-urilor folosind caracterul '@' primesc email anunțându-i că au fost menționați în task-ul respectiv

Dezavantaje

- La ștergerea unui task nu se primește notificare
- Nu există chat în aplicație.

În altă ordine de idei, Asana este o aplicație utilizată la nivel mondial și include API⁸ pentru aplicații externe în același timp integrează și servicii precum Dropbox, Evernote, Google Drive, Harvest, Zendesk, etc...



Figură 4 - Interfață Asana

⁸ API – Application programming interface (interfață pentru programarea de aplicații) – o colecție de protocoale și unelte pentru integrarea de caracteristici din alte aplicații sau servicii.

Capitolul 2 – Tehnologii utilizate

Tehnologii utilizate

Toată aplicația a fost construită pe platforma **.NET** de la Microsoft. .NET (dotNET) este o colecție de framework-uri și componente dezvoltate de Microsoft, ce oferă un mediu pentru dezvoltarea și execuția de aplicații și interoperabilitate pentru alte limbaje de programare. La baza .NET stă Common Language Runtime (CLR) ce reprezintă mediul pentru execuția codului. Acesta oferă servicii precum : compilarea, alocarea și realocarea memoriei, managementul firelor de execuție, securitate și tratarea excepțiilor.

Împreună cu CLR, în pachetul .NET se află și o colecție imensă de librării (Framework Class Library – FCL) ce oferă :

- Interfață cu utilizatorul (WinForms, WPF)
- Acces de date (ADO.NET)
- Conectare cu baze de date (Entity Framework)
- Aplicații Web(ASP.NET)
- Comunicare în rețea (folosind Sockeți)
- Fire multiple de execuție (folosind Task-uri și Thread-uri)
- Limbaj de interogare structurat (SQL – folosind LINQ)

C# (CSharp) este unul dintre limbajele de programare realizate pentru a funcționa peste CLR, dezvoltate de Microsoft și inclus în platforma .NET. C# vrea să fie un limbaj simplu, modern, orientat pe obiecte și inovativ [4]. Am ales C# deoarece este foarte ușor de folosit și mi-a permis să îl folosesc atât pentru dezvoltarea front-end cât și pentru dezvoltarea back-end.

După cum am spus mai sus, întreaga aplicație a fost construită peste platforma .NET după cum urmează :

Pentru partea de front-end am ales să lucrez pentru UI⁹ cu WinForms, am ales WinForms deoarece este inclusă în .NetFramework încă din 3.0 și pentru că am o experiență de lucru mai mare față de celelalte librării UI din .NET (WPF sau XAML). Comunicarea cu back-end-ul se face în două moduri :

1. Pentru apeluri mici (*verificări, conectare, jurnalizare, preluări mici de date, notificări*)
Folosesc o librărie de comunicație, din aceeași platformă, numită SignalR.
2. Pentru apeluri mari (*încărcări/descărcări de fișiere, preluare cache, volum mare de date*)
Folosesc un serviciu REST peste HTTP.

Back-end-ul este o aplicație web făcută în ASP ce folosește aceeași librărie de comunicare ca pe front-end, SignalR, pentru apeluri mici, iar pentru servicii mai mari un serviciu REST.

⁹ UI – Interfață cu Utilizatorul (User Interface)

Baza de date este ținută pe un server de MySQL, care este mai mult decât necesară pentru această aplicație.





Conectarea se face folosind un ORM¹⁰, gratuit, de la Telerik numit Data Access. ORM-ul la început generează toate clasele corespunzătoare tabelelor și view-urilor aplicând tuturor proprietăților atribute de tip **Column** iar claselor atribute de tip **Table**:

```
[Table("user")]
[KeyGenerator(KeyGenerator.Autoinc)]
public partial class User : GenericEntity
{
    [Column("id", OpenAccessType = OpenAccessType.Int32, IsPrimaryKey = true)]
    public int Id { get; set; }

    [Column("name", OpenAccessType = OpenAccessType.UnicodeString)]
    public string Name { get; set; }

    [Column("email", OpenAccessType = OpenAccessType.UnicodeString)]
    public string Email { get; set; }

    [Column("suspended", OpenAccessType = OpenAccessType.Boolean, IsNullable = true)]
    public bool Suspended { get; set; }
}
```

user	
	id: BIGINT
	name: VARCHAR(128)
	email: VARCHAR(256)
	suspended: TINYINT

```
using (var context = WpContext.CreateContext)
{
    User user = context.Users.FirstOrDefault(u => u.Name == "Andrei");
    if (user != null)
    {
        user.Suspended = true;
        context.SaveChanges();
    }
}
```

După cum se observă clasele generate vor avea ca atribut numele tabelului ce îl mapează iar fiecare proprietate vor avea ca atribut numele coloanei mapată plus informații esențiale precum Tipul de dată, dacă este cheie primară, dacă este nulabilă, ș.a....

Pentru a modifica un rând se folosește un context ce face conectarea la baza de date, acesta va crea automat și o tranzacție, iar modificările se vor aplica atunci când dorim.

După cum putem observa, folosind LINQ, se trimite un predicat iar metoda **FirstOrDefault** ne va returna primul rezultat ce îndeplinește condiția din predicat, în cazul nostru primul utilizator cu numele „Andrei”. Mai departe, dacă utilizatorul a fost găsit, acesta se marchează ca fiind suspendat și se comită¹¹ modificările.

Comunicația dintre back-end și front-end se face printr-o librărie din ASP.NET numită SignalR.

¹⁰ ORM – Object-relational mapping – Tehnică de convertirea a modelului bazei de date în obiecte compatibile pentru manipularea programatică

¹¹ Comit – a aplica modificări, a salva

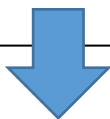
Această librărie oferă un canal de comunicare bi-direcțional între client și server folosind Socketi Web [5] oferind de asemenea evenimente de conexiune (*conectare*, *deconectare*, *reconectare*) și securitatea conexiunii, astfel încât apelurile de metode web se face foarte ușor atât de pe client cât și de pe server.

```
namespace WProject.DataAccess
{
    [Table("user")]
    [KeyGenerator(KeyGenerator.Autoinc)]
    public partial class User : GenericEntity
    {
        [Column("id", OpenAccessType = OpenAccessType.Int32, IsPrimaryKey = true)]
        public int Id { get; set; }

        [Column("name", OpenAccessType = OpenAccessType.UnicodeString)]
        public string Name { get; set; }

        [Column("email", OpenAccessType = OpenAccessType.UnicodeString)]
        public string Email { get; set; }

        [Column("suspended", OpenAccessType = OpenAccessType.Boolean, IsNullable = true)]
        public bool Suspended { get; set; }
    }
}
```



```
namespace WProject.WebApiClasses
{
    public partial class User
    {
        public int Id { get; set; }

        public string Name { get; set; }

        public string Email { get; set; }

        public bool Suspended { get; set; }
    }
}
```

Mai sus am descris procedura prin care ORM-ul va genera clase pentru manipularea bazei de date, acestea fiind folosite doar pe back-end pe front-end și în comunicare se vor folosi clase de manevră, cu aceleași proprietăți dar fără atribute, astfel încât ne-manageruite de ORM, și într-un namespace separat.

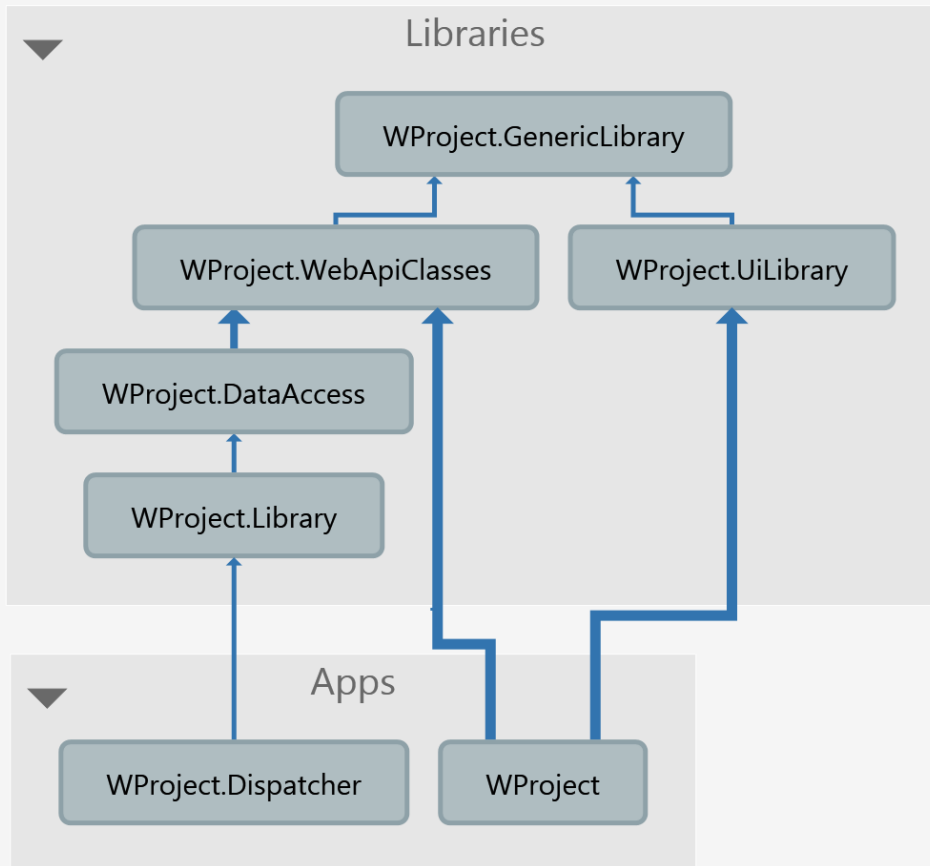
După cum se poate observa este o clasă mult mai „light” ocupând mult mai puțină memorie pe front-end, serializarea se face mult mai repede iar transferul durează mult mai puțin. Acest obiect al clasei de manevră se serializează și se trimite mai departe.

Serializarea datelor, ce se face la transfer-ul de pe client pe server și invers, se face folosind formatul **JSON** prin framework-ul **Json.NET** de la Newtonsoft, ce face serializarea și

deserializarea folosind reflexie. Am ales acest framework în primul rând deoarece este open-source, iar în al doilea rând este cel mai rapid framework JSON pentru .NET.

Prezentare generală

După cum am specificat, aplicația este împărțită în două, front-end și back-end. La nivel de soluție, proiectul este împărțit în două aplicații și cinci librării. (Figură 5)



Figură 5 - Diagrama librărilor și a aplicațiilor

- WProject – Este aplicația de front-end
- WProject.Dispatcher – Este aplicația web de back-end
- WProject.GenericLibrary – Este o librărie în care se țin
 - Constante
 - Excepții
 - Coduri de eroare
 - Extensii de metode
 - Funcții util

- WProject.WebApiClasses – Librăria în care se țin clasele de manevră și clasele de comunicare
- WProject.DataAccess – Librăria în care se află clasele generate de ORM
- WProject.Library – Librăria ce conține
 - Metode CRUD pentru obiectele din ORM
 - Metode de conversie din obiecte ORM în obiecte WebApi
 - Metode pentru jurnalizare în baza de date
- WProject.UiLibrary – Librărie cu controale personalizate dar si multe metode ajutătoare pentru UI

Capitolul 3 – Utilizarea sistemului

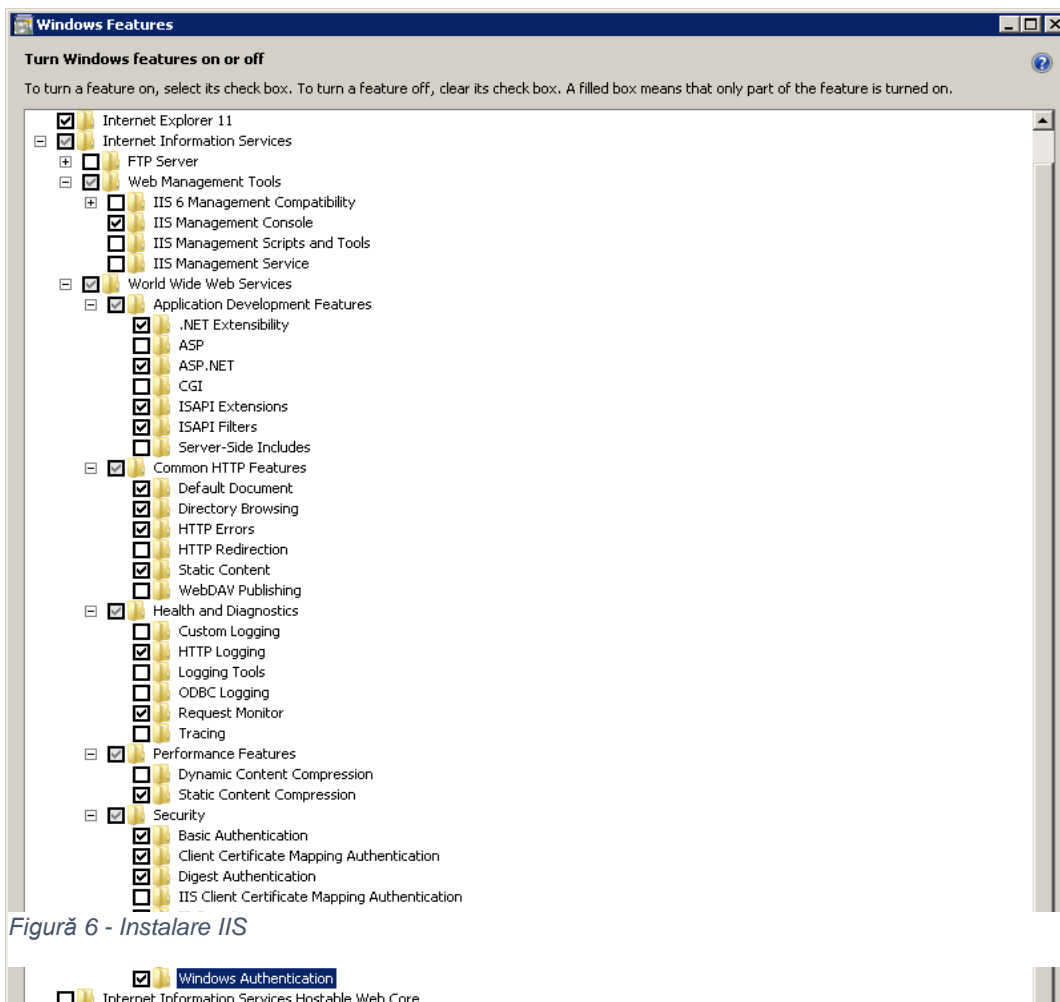
Instalarea aplicației

Instalarea back-end-ului

Pentru a instala aplicația de back-end este nevoie de un PC sau server cu cel puțin următoarea configurație :

- **OS** : Windows 7 Professional / Windows Server 2008
- **Procesor** : Dual Core 1.4 GHz 64bit (Recomandat 2.0 GHz)
- **RAM** : 1GB (Recomandat 2 GB)
- **HDD** : 2GB (Recomandat 5GB)

Deoarece este o back-end-ul este o aplicație ASP.NET este necesar instalarea IIS (Internet Information Services) minim versiunea 7.0. Pentru instalarea IIS-ului se intră în Control Panel > Programs and Features > Turn Windows features on or off, și se selectează ca în figura următoare. (Figură 6). După salvare este recomandat de a executa o repornire a sistemului pentru a porni toate serviciile necesare (HTTP, HTTPS, FTP, FTPS, SMTP și NNTP) și pentru a încărca toate resursele necesare rulării serverului de Windows. Pentru utilizarea protocolului HTTP/2 este necesar Internet Information Services 10, inclus în Windows 10 sau Windows Server 2016.



Figură 6 - Instalare IIS

Mai departe se descarcă MySQL Server (care se găsește în directorul MySQL_Server), iar la configurarea lui, pe lângă utilizatorul root, se mai adaugă un utilizator, cu drepturi globale, cu numele wproject și o parolă aleasă.

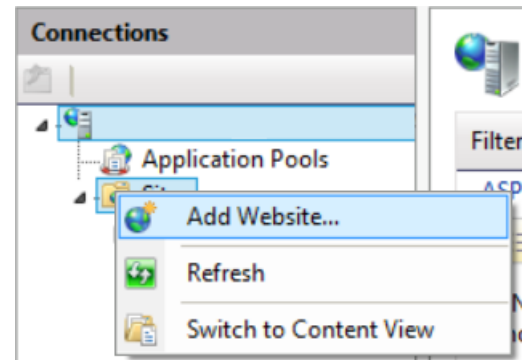
Mai departe, se intră în MySQL Command Line Client (se poate găsi în Start, sau în C:\Program Files\MySQL\MySQL Server 5.x\bin\mysql.exe). Odată pornită linia de comandă se va introduce parola de root, apoi se va crea o bază de date folosind comanda

```
CREATE DATABASE wproject
USE wproject
```

După crearea bazei de date se va folosi dump-ul inițial pentru crearea tabelelor folosind comanda

```
SOURCE c:\Users\User\Desktop\Exemplu_cale\WProject\Install\dump.sql
```

După execuția dump-ului putem trece la configurarea server-ului, se intră în Start și se caută **Internet Information Services Manager** sau se apasă WIN + R iar în fereastra apărută scriem **inetmgr**. În fereastra apărută deschidem arborele din stânga până vedem un elementul **Sites**, dăm click dreapta pe el, iar din meniul contextual apărut selectăm „Add Website...” (Figură 7). În fereastra apărută la Site Name completăm **WProject.Dispatcher**, la Physical path selectăm c:\ iar în c creăm un nou director numit **wproject_dispatcher**, după crearea lui îl selectăm și apăsăm Ok.



Figură 7 - Adăugare site în IIS Manager

La port vom pune 8002 și apăsăm Ok. Apoi vom copia conținutul directorului **Server Files** în directorul creat adineaori (c:\wproject_dispatcher). Intrăm în directorul c:\wproject_dispatcher și vom deschide cu un editor text fișierul **Web.config**.

Urmând structura configuration > applicationSettings vom găsi următoarele chei :

Pentru cheia <setting name="MysqlServer" ...> vom pune la value

```
| localhost
```

Așa și pentru restul cheilor folosind modelul :

```
<setting name="MysqlServer" serializeAs="String">
    <value>localhost</value>
</setting>
<setting name="MysqlPort" serializeAs="String">
    <value>3306</value>
</setting>
<setting name="MysqlDatabase" serializeAs="String">
    <value>wproject</value>
</setting>
<setting name="MysqlUser" serializeAs="String">
    <value>wproject</value>
</setting>
<setting name="MysqlPassword" serializeAs="String">
    <value>parola_aleasă</value>
</setting>
```

Salvăm fișierul, putem intra într-un browser și introducem la adresă

| `http://localhost:8002`

iar în pagina apărută putem vedea starea conexiunilor, și eventual în caz de probleme ce putem face să le remediem.(Figură 8)

Dacă toate bulinele sunt verzi înseamnă că serverul a fost instalat cu succes și putem trece la etapa următoare.

WProject.Dispatcher

● Server

● MySQL

● SignalR

Figură 8 - Stare server

Instalarea front-end-ului

Instalarea front-end-ului este mult mai simplă, din directorul Front End se deschide fișierul WProject_Install.exe iar acesta va instala aplicația, și va face și o comandă rapidă de desktop. După instalare se intră în WProject (din icoana de pe desktop sau din meniul de Start) și va apărea o fereastră care va spune că clientul nu este configurat, vi se va cere să introduceți adresa dispecerului. Dacă aplicația rulează pe același PC cu server-ul este îndeajuns să puneți

| `localhost`

În caz că aplicația rulează pe un alt calculator este necesar să puneți adresa/IP-ul server-ului, ex:

| `172.16.1.103`

După ce ați completat adresa apăsați Ok și așteptați până se verifică conexiunea, iar dacă verificarea s-a încheiat cu succes vă puteți conecta cu:

Nume	admin
Parola	wproject

Intrarea în aplicație

În cazul în care se intră în aplicație și nu există proiecte disponibile, dacă utilizatorul conectat este administrator, acesta va fi redirecționat către o secțiune de inițiere de aplicație ce conține 3 pași, pe toată perioada configurării, aplicația va fi inutilizabilă de orice fel de utilizator. O configurare completă durează aproximativ 10-15 de minute în funcție de numărul de utilizatori, de numărul proiectelor, a iterațiilor și categoriilor din fiecare proiect și bineînțeles de experiența utilizatorului în utilizarea aplicațiilor similare.

Crearea utilizatorilor

În primul rând va fi configurat utilizatorul **admin**. El va fi obligat să își introducă un e-mail, să își introducă o parolă (diferită de wproject), opțional să își schimbe numele.

Apoi se vor introduce utilizatorii – nume și email – aceștia primind pe mail un link de setare a parolei. Opțional le va putea seta drepturile de acces. Mai departe vor fi create grupurile (acesta fiind un pas opțional) cu drepturile de acces pe fiecare grup și utilizatorii ce fac parte din grupurile respective.

Crearea proiectelor

Cel de al doilea pas este configurarea proiectelor. Fiecare utilizator ca să poată folosi aplicația trebuie să fie inclus în cel puțin un proiect. Astfel, administratorul va crea proiectele, le va asigna celorlalți administratori și va include utilizatorii în ele. Iterațiile și categoriile se vor face mai târziu de fiecare administrator din modulul de administrare.

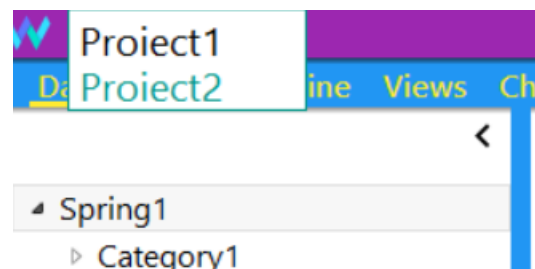
Configurarea aplicației

Ultimele setări sunt pentru personalizarea sistemului de task-uri, spre exemplu vom putea configura culorile task-urilor, ale backlog-urilor. Ce module din aplicație se vor folosi (Dashboard, Timeline, Chat). Numărul maxim de utilizatori ce pot fi conectați în același timp, opțiunea dacă un utilizator își poate utiliza contul în două aplicații separate și orele de program (ce vor fi afișate în modulul de Timeline).

Aceste setări sunt pre completate cu modelul cel mai uzual, astfel încât un administrator să nu fie obligat să introducă toate aceste setări.

Utilizare aplicației

Putem observa în partea de sus a aplicației proiectul curent în care lucrăm. Dacă ne-am conectat cu un utilizator ce are acces la mai multe proiecte, sau un administrator ce deține mai multe proiecte putem chiar modifica proiectul curent folosindu-ne de control-ul de tip ComboBox. (Figură 9). Odată proiectul schimbat, acesta va reîncărca conținutul paginii curente.



Figură 9 - Selector proiecte

Dashboard

Dashboard-ul reprezintă metoda principală și cea mai ușoară și uzuală metodă de lucru cu task-uri și backlog-uri. În dreapta se află iterațiile și categoriile ce pot fi ascunse, iar în partea dreapta este tabelul cu task-uri, grupate pe backlog-uri și aliniate în funcție de starea lor (To Do, In Progress, Done), cele Removed nu mai apar în Dashboard, dar pot fi urmărite în Admin. Culoarea de fundal al task-urilor sunt de două tipuri, gri sau galbene, cele galbene sunt cele atribuite utilizatorului conectat, iar cele gri altor utilizatori sau task-urile fără utilizator.

Starea task-urilor se poate seta doar trăgând de el în starea dorită (drag-n-drop) și i se poate modifica utilizatorul din controlul de tip ComboBox din partea stângă jos a fiecărui task (bineînțeles dacă utilizatorul are drept). Atunci când se dă click pe un task apare un formular de modificare avansată a task-ului respectiv. Dacă se dorește adăugarea unui noi task, trebuie deschis backlog-ul în care dorim să adăugăm task-ul și atunci când punem mouse-ul în zona de to-do a backlog-ului va apărea jos un buton „plus”, atunci când îl vom apăsa va apărea fereastra de adăugare de task.

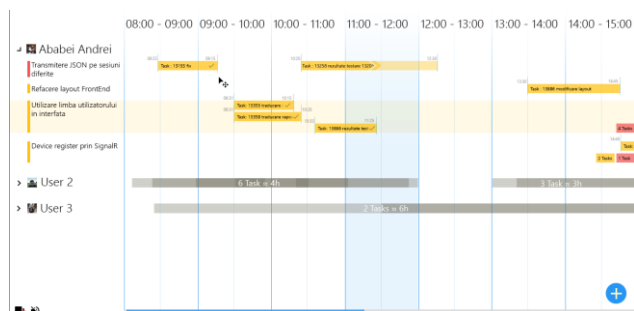
La fel și în cazul backlog-urilor putem de asemenea seta rapid utilizatorul ce deține backlog-ul și starea acestuia din controlul de tip ComboBox din fiecare backlog, atunci când acesta este deschis. Când este închis se poate observa sub titlul cui aparține backlog-ul, iar în coloanele To do, In Progress și respectiv Done câte task-uri sunt în starea respectivă, de asemenea și timpul estimat în total al task-urilor din starea respectivă. Pentru a adăuga un backlog se folosește butonul de „plus” din antetul task-urilor.

De asemenea deasupra antetului se mai afla o serie de butoane și texte, de la stânga la dreapta :

- Butonul de filtrare/căutare – filtrează task-urile după utilizator/stare sau caută după titlu și conținut.
- Timpul limită pentru iterația respectivă împreună cu un graf al task-urilor.
- Butonul de ajutor, afișează manualul aplicației cu căutare.
- Butonul de setări.
- Butonul de full-screen – folosește întreaga suprafață a ferestrei doar pentru task-uri.

Timeline

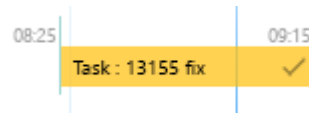
O altă posibilitate de a vizualiza task-urile este în modul Timeline (Figură 10), care practic reprezintă aranjarea task-urilor după ore și durată. După cum se poate observa mai jos, se afișează toate task-urile programate în ziua curentă pe perioadă de lucru definită.



Figură 10 - Sistemul de task-uri în mod timeline

Intervalele orare pot fi mutate, pur si simplu trăgând de task la fel și perioada estimată trăgând de marginea din dreapta. În momentul în care un task este trecut pe Done acesta va apărea în Timeline cu ora și minutul exact când a fost închis, în caz că taskul este încă In Progress sau în To do intervalul va fi calculat în funcție de timpul estimat introdus.

Task-urile fără timp de început sau fără timp estimativ vor apărea totuși în Timeline, dar în marginea dreapta cu roșu, iar cele din dreapta cu galben sunt task-urile ce ies din dimensiunea controlului, făcând scroll vor apărea. Task-urile Done vor fi marcate și cu un „check” în marginea dreaptă pentru o observare mai ușoară(Figură 11).



Figură 11 - Task marcat cu un check ce arată că a fost terminat

La fel ca la Dashboard se poate face filtrarea pe utilizatori dar în plus se pot afișa și task-urile programate pe zilele următoare. Pentru adăuga un task se folosește butonul din colțul de stânga-jos, și exact ca la Dashboard va apărea un formular de adăugare de task. Mai jos, după task-urile atribuite utilizatorului curent se pot vedea task-urile și altor utilizatori. În caz că pe panoul de task-uri pe linie vor apărea un model cu toate task-urile și numărul lor împreună cu timpul estimat rămas.

Acțiunile rapide din Timeline se fac dintr-un meniu contextual pe fiecare task. Din meniul contextual avem posibilitatea să :

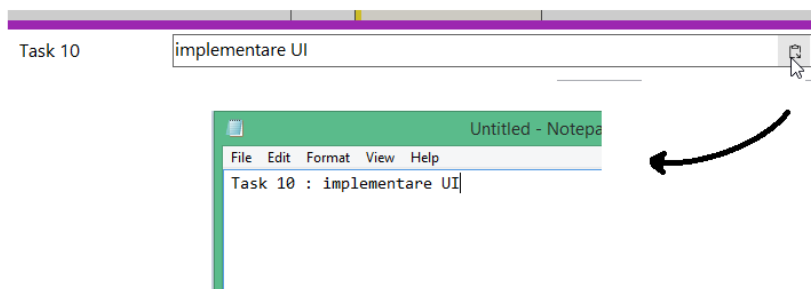
- Modificăm starea task-ului
- Modificăm utilizatorul
- Modificăm timpul estimat rămas
- Copiem id-ul și titlul task-ului în clipboard

Formularul de modificare task

În momentul în care dăm click pe un task din Dashboard sau din Timeline, va apărea un formular ce ne va permite sa facem modificări asupra unui task, dar să și vedem informațiile structurate la nivel de tab-uri plus informațiile generale plasate în partea de sus a formularului. Formularul este structurat astfel :

- Partea de sus :
 - **Codul task-ului**
În colțul de sus se află codul unic al task-ului cu care va putea fi identificat, acesta este unic la nivelul întregii soluții
 - **Numele task-ului**
După cod, urmează un câmp de text cu numele task-ului (de obicei o scurtă descriere, de cel mult o fraza) ce poate fi modificat, iar în partea dreaptă a

câmpului se află un buton de copiere rapidă a numelui task-ului (Figură 12)

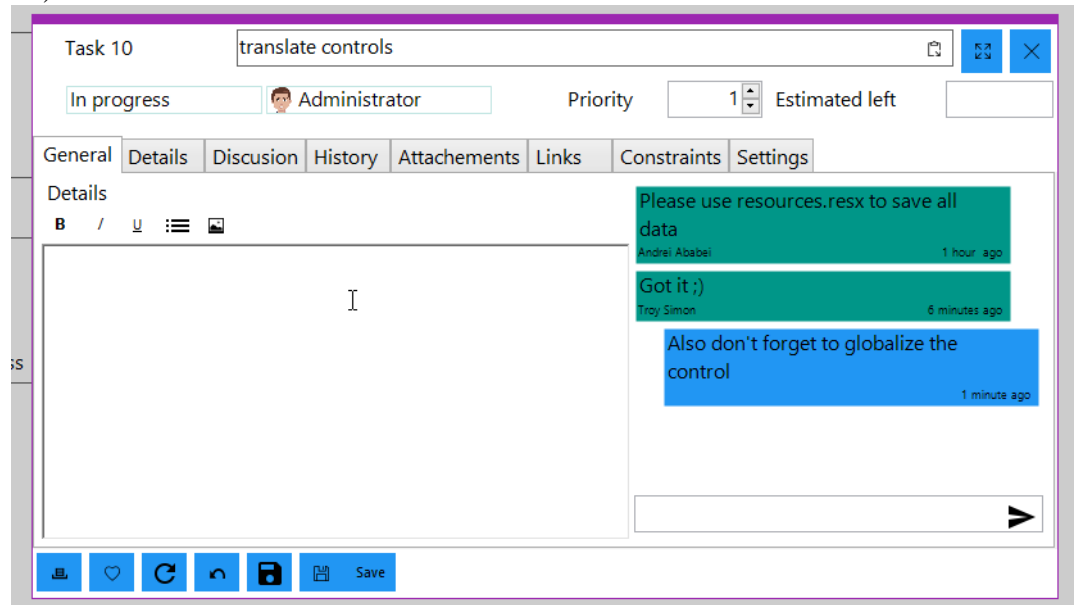


Figură 12 - Buton copiere nume task

- **Butoanele de maximizare și închidere**
Ultimele controale de pe primul rând se află butoanele de maximizare fereastră și închidere fereastră
- **Selectorul de stare**
Rândul doi începe cu selectorul de stare, una dintre cele mai importante proprietăți ale unui task. La crearea unui task noi starea implicită este “To Do”. Stările unui task pot fi :
 - To Do - Task-ul nu a fost încă preluat
 - In Progress - Task-ul a fost preluat de un utilizator și se lucrează la el
 - Done - Task-ul a fost terminat cu succes
 - Removed - Task-ul a fost șters
- **Selectorul de utilizator**
O altă proprietate importantă a unui task este utilizatorul asignat task-ului respectiv, utilizatorul asignat este cel ce se ocupă cu rezolvarea task-ului. Un task nu se poate atribui mai multor persoane, ci acesta va fi “spart” un sub-task-uri mai mici.
- **Prioritate**
Pentru o urmărire mai ușoară și o prioritizare a muncii se poate completa acest câmp pentru a sorta task-urile în ordinea priorității.
- **Timp estimat rămas**
Un aspect important în terminarea unui proiect este timpul limită și prioritizarea lucrului. Pentru ca un manager de echipă să poată controla proiectul și să îl ducă la bun sfârșit în termenul propus, acesta trebuie să știe fiecare task cât mai durează.
- **Tab-ul General**
 - **Detalii**
O descriere mai amplă a task-ului incluzând toate punctele ce trebuie atinse pentru a termina task-ul.

- **Mesaje**

Folosită pentru comunicarea mai simplă între membri proiectului (Figură 13)



Figură 13 - Fereastră modificare task

- **Tab-ul Detalii**
Aici se găsesc informații avansate precum data de lucru în care trebuie realizat task-ul și intervalul orara în care a fost programat task-ul.
- **Tab-ul Discuție**
Este un modul ce va fi implementat în viitor oferind posibilitatea comunicării direct cu clientul, oferindu-i un simplu link cu care acesta poate participa la conversația task-ului
- **Tab-ul Istoric**
În tab-ul istoric se regăsesc toate modificările cu făcute asupra task-ului grupate pe câmpuri.
- **Tab-ul Link-uri**
Tab-ul încă nu este implementat, acesta pe viitor va oferi posibilitatea ca clienții să urmărească în timp real stadiul task-urilor.
- **Tab-ul Constrângeri**
În tab-ul constrângeri vor fi declarate constrângerile, de exemplu : *un task nu poate fi început doar după ce un task este marcat ca fiind "Done"*
- **Tab-ul Setări**
În care se află setările avansate ale task-ului
- **Acțiunile task-ului**
 - **Butonul de imprimare**
Exportă task-ul într-un pdf
 - **Butonul "favorit"**
Odată marcat task-ul ca fiind "favorit", utilizatorul care si-a adăugat la favorite,

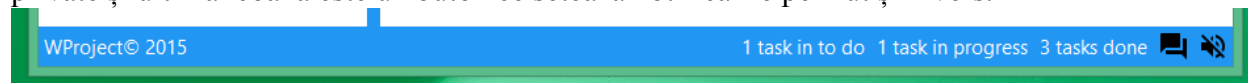
acesta va primi notificări și email-uri în momentul în care se fac modificări asupra task-ului.

- **Butonul reîncărcare**
Odată apăsat se reiau toate informațiile de pe server făcute de la ultima salvare și le introduce în formular, anulând modificările nesalvate.
- **Butonul anulare**
Similar cu butonul de reîncărcare, acesta anulează toate modificările nesalvate fără a mai lua informațiile de pe server, ci folosindu-le pe cele din memorie.
- **Butonul salvează**
Trimite toate modificările pe server și anunță toți clienții asupra modificărilor făcute, reîncărcând controlul.
- **Butonul salvează și închide**
Ultimul buton, execută exact același comportament ca și cel de salvare cu excepția că după execuția procesului de salvare, fereastra se închide.

La fiecare modificare de task/backlog serverul trimite tuturor clienților modificările pentru a modifica datele din caseta task-ului chiar și poziția lui în tabelul din Dashboard sau vizibilitate controlului din modulul de Timeline.

Linia de stare

În dashboard apare în partea de jos un control de tip “statusbar” Figură 14 în care apare în partea stângă numele aplicației, iar în partea dreaptă apar numărul de task-uri pentru fiecare stare : To Do (doar ziua curentă, sau fără zi curentă), In Progress și Done (terminate în ziua curentă), iar la sfârșit apar două icoane, prima de afișare a conversației de grup și conversațiilor private și ultima icoană este un buton ce setează notificările pe mut și invers.



Figură 14 - Bară de stare

Administrare

Modulul de administrare este accesibil doar pentru managerii de proiecte și administratorilor, el este folosit gestionarea utilizatorilor, grupurilor, a proiectelor și setărilor generale.

Gestionarea utilizatorilor

În pagina de gestiune a utilizatorilor, în funcție de tipul de utilizator ce accesează pagina, sunt afișați utilizatorii. În caz că utilizatorul este de tip **administrator**, vor fi afișați toți utilizatorii, în caz că utilizatorul este **manager**, vor fi afișați utilizatorii ce sunt alocați proiectelor ce le dețin. În antetul paginii se află un câmp de filtrare după numele utilizatorilor. După câmpul de căutare se află lista de utilizatori ordonați alfabetic, fiecare coloană având informații generale precum nume, proiectele de care aparține, grupurile de care aparține, starea lui (dacă este conectat sau nu,

în caz că nu este conectat este afișată ultima conectare), și un set de butoane de acțiuni precum **Modifică**, **Mesaj** și **Altele** ce deschide un meniu contextual cu acțiuni rapide (**Setare grupuri**, **Setare proiecte**, **Deconectare** și **Blochează utilizator**). În josul paginii se află controlul de paginație, vizibilă doar dacă în listă sunt mai mult de 20 de utilizatori, fiind grupați în perechi de câte 20 pe fiecare pagină. Acțiunile ce pot fi făcute pe un utilizator sunt :

- **Modificare**

Ce afișează un formular cu datele utilizatorului dispuse în 3 tab-uri:

- **General**

Tab-ul general conține informații de bază precum Nume, Email, Data de expirare a contului și drepturile de acces

- **Grupuri**

În tab-ul de grupuri se regăsesc toate grupurile plasate într-o listă, fiecare linie având câte un control de tip checkbox pentru a marca apartenența la respectivul grup cu posibilitatea de a schimba selecția, acțiune ce va rezulta cu setarea participării utilizatorului în grup.

- **Proiecte**

Conținutul ultimului tab constă într-un control, similar cu cel al grupurilor, cu o coloană de nume și una de tip checkbox ce arată participarea în proiect.

- **Mesaj**

Odată apăsat butonul, va apărea o fereastră de mesaj ce va avea câmpul destinat precompletat cu utilizatorul respectiv.

- **Altele**

Butonul are ca eveniment afișarea unui meniu contextual cu 4 opțiuni:

- **Setare grupuri**

Ce va afișa fereastra de modificare utilizator cu tab-ul **Grupuri** activ.

- **Setare proiecte**

Exact ca la opțiunea **Setare grupuri**, acesta va afișa fereastra de modificare utilizator cu tab-ul **Proiecte** activ.

- **Deconectare**

Odată apăsată opțiunea va avea ca rezultat deconectarea utilizatorului, fiind obligat ca acesta să se reconecteze dacă dorește să refolească aplicația. Aceasta opțiune poate fi folosită în cazul în care o conexiune a fost blocată din cauza unei erori sau unui acces nepermis.

- **Blochează**

Opțiunea este folosită pentru a bloca accesul unui utilizator la aplicație. La apăsarea opțiunii va apărea un formular prin care ne va anunța că acțiunea pe care urmărim să o facem este ireversibilă și va cere să confirmăm blocarea utilizatorului prin scrierea numelui într-o casetă de text pentru a confirma blocarea. După marcarea utilizatorului ca fiind blocat, acesta va fi deconectat automat, acesta nemaiputând să se conecteze.

Gestionarea grupurilor

În pagina de grupuri sunt încărcate toate grupurile de utilizatori într-o listă ordonată alfabetic incluzând informațiile esențiale, precum nume, număr de utilizatori aparținând grupului și acțiuni. Grupurile se folosesc de obicei pentru gruparea utilizatorilor pe anumite nivele de acces. De cele mai multe grupurile sunt create ca funcțiile fiecărui utilizator (ex: Manager, Dezvoltator, Tester, Suport, etc...) iar aceste pot fi foarte utile pentru a trimite un mesaj tuturor utilizatorilor dintr-un grup, acest lucru făcându-se și din modul de chat. Doar utilizatorii de tip **manager** și **administrator** pot adăuga grupuri noi apăsând butonul din dreapta jos alt tabelului. Butoanele din coloana de acțiuni sunt următoarele :

- **Modifică**

La apăsarea butonul va apărea un formular cu un câmp de text în care se introduce numele grupului și încă o listă cu utilizatori ce conține următoarele coloane un control de tip checkbox ce marchează apartenența la grup, numele utilizatorului și încă o coloană ce va fi completată, dacă utilizatorul aparține de grupul respectiv, cu numele managerului/administratorului ce a adăugat utilizatorul în grup împreună cu data și ora când a fost adăugat în grup. Două grupuri nu pot avea același nume pentru a nu se face confuzie între ele.

- **Masaj**

Ce va afișa o fereastră de mesaj ce are ca destinatar precompletat numele grupului și câmpul de mesaj ce va fi trimis toți utilizatorilor din grup.

- **Șterge**

Odată apăsând butonul de ștergere va apărea o fereastră de confirmare ce ne informează că acțiunea este ireversibilă și pentru confirmarea operațiunii va trebui să introducem într-o casetă de text numele grupului respectiv.

Gestionarea proiectelor

Pagina de gestionare a proiectelor este cea mai complexă, aceasta fiind compusă din două panouri principale, primul fiind lista de proiecte și cel de-al doilea este arborele de iterații și categorii.

Panoul de proiecte este populat cu lista ce aparțin proiectelor utilizatorului curent, cu excepția administratorilor, acești utilizatori văd toate proiectele, ordonate alfabetic. Fiecare proiect având în partea stângă un buton de modificare atunci când se pune mouse-ul peste obiectul din listă. Deasupra listei se află un buton de adăugare proiect. La apăsarea butonului de modificare va apărea un formular cu informații despre proiect editabile precum :

- **Numele proiectului**

Odată modificat numele proiectului, toți utilizatorii vor primi o notificare ce îi anunță că numele proiectului a fost modificat.

- **Utilizatorul ce deține proiectul**

Odată modificat deținătorul, proiectul va dispărea din listă (dacă utilizatorul nu este de tip administrator).

- **Perioada proiectului (deadline)**

Perioada în care proiectul trebuie început respectiv terminat.

- **Utilizatorii ce fac parte din proiect**

Similar cu controlul de grupuri, listă cu 3 coloane, prima coloana este un control de tip checkbox ce marchează dacă utilizatorul este inclus în proiect, a doua coloană este numele utilizatorului, iar cea de-a treia coloană este completată cu numele managerului/administratorului care a inclus respectivul utilizator în proiect plus data și ora când a fost adăugat în proiect. În momentul în care, un utilizator, este adăugat sau scos din proiect, utilizatorul respectiv va primi o notificare ce îl informează despre acțiunea ce a fost făcută

- **Afișează setări avansate**

La bifarea checkbox-ului apare un buton pentru ștergerea proiectului, care, odată apăsat apare controlul ce ne anunță că ștergerea este ireversibilă și pentru confirmare trebuie să introducem numele proiectului respectiv.

Panoul de iterații conține iterațiile și categoriile dispuse în mod arborescent și în mod tabelar. Pe primul nivel aflându-se iterațiile, apoi pe nivelele mai inferioare se află categoriile. Tabelul având două coloane, prima fiind numele iterației sau a categoriei, depinzând de nivel, iar cea de-a doua este o coloană cu acțiuni

- **Adaugă**

La apăsarea butonului de adăugare va apărea un formular pentru crearea unei categorii (sau subcategorii) ce va include un câmp de text pentru nume și două controale de dată opționale ce semnifică data limită a categoriei.

- **Modifică**

Butonul de modificare deschide un formular pentru modificarea iterației sau a categoriei, pentru ambele tipuri, formularul include un câmp de text folosit pentru modificarea numelui și două câmpuri de tip dată ce marchează limita iterației sau a categoriei.

- **Ștergere**

La fel ca la celelalte ferestre de confirmare, ne va anunța că ștergerea este ireversibilă și un câmp de text în care trebuie introdus numele iterației/categoriei pentru a finaliza ștergerea.

Deasupra listei se află încă un buton de adăugare iterație ce, odată apăsat, va apărea un formular de adăugare iterație similar cu cel de modificare iterație sau categorie.

Capitolul 4 – Dezvoltarea sistemului

Dezvoltarea aplicației

Back-end

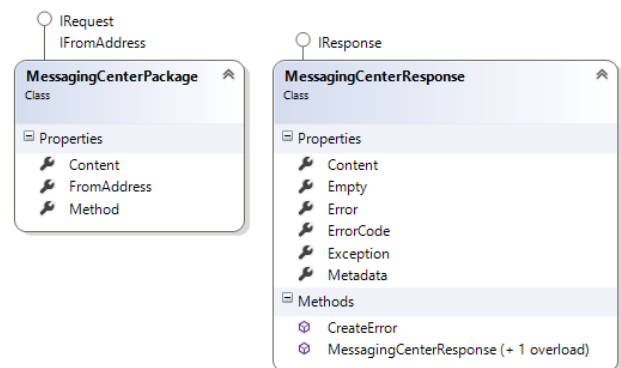
Aplicația ce rulează pe server este construită pe modelul WebApi ce implementează un standard OWIN ce creează o legătură între aplicații și servere. Pentru utilizarea mai ușoară se folosește și librăria SignalR, dezvoltată de Microsoft, ce ușurează crearea de metode web ce pot fi apelate de pe client sub forma de metode asincrone sau de pe server pe client transmise ca evenimente crescând în acest fel scalabilitatea și a minimiză resursele. SignalR este o librărie C# open-source și gratuită ce folosește WebSocket [5] și implementează HTML5 API ce conferă posibilitatea creării unui canal de comunicație bi-direcțional. Pentru inițierea conexiunii pe server am creat clasa **Startup** implementata astfel :

```
internal class Startup
{
    public void Configuration(IApplicationBuilder app)
    {
        app.UseCors(CorsOptions.AllowAll);
        var hubConfiguration = new HubConfiguration();
        app.MapSignalR(hubConfiguration);
    }
}
```

Iar clasa de comunicație a fost concepută extinzând clasa **Hub** din SignalR numită **MessagingCenter**, aceasta fiind o clasă parțială împărțită în două fișiere :

- **MessagingCenter.cs**

Conține o listă statică de clienți conectați numită **WpClients** și suprascrierea metodelor **OnConnected():Task** și **OnDisconnected():Task** ce sunt folosite pentru popularea listei **WpClients**, la conectare se adaugă în listă clientul conectat, iar la deconectare se șterge clientul din listă. Fișierul mai conține o metoda numită **CallServiceMethod** ce primește un parametru de tip **MessagingCenterPackage** și returnează răspunsul încapsulat în tip-ul **MessagingCenterResponse**. Acestea sunt clase folosite pentru comunicare între client – server / server – client. (Figură 15). Metoda primește cererea ce conține numele metodei ce va fi apelată, adresa de unde se face cerea și parametri, ce vin de obicei serializați. Folosind reflexie se verifică



Figură 15 - Clase cerere și răspuns

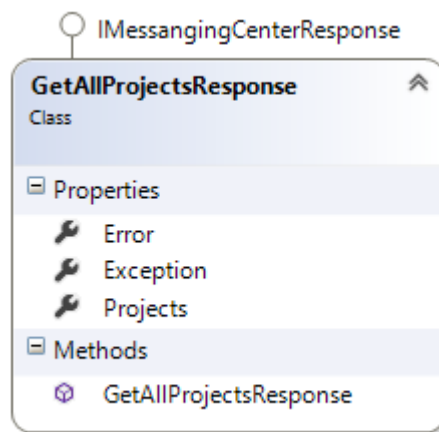
dacă există metoda în clasa MessagingCenter, dacă da atunci o invocă, în caz contrar se returnează un răspuns de tip eroare ce conține codul de eroare 4

```
| public const int ERROR_MESSAGING_CENTERE_METHOD_NOT_FOUND = 4;
```

- MessagingCenter.Server.cs

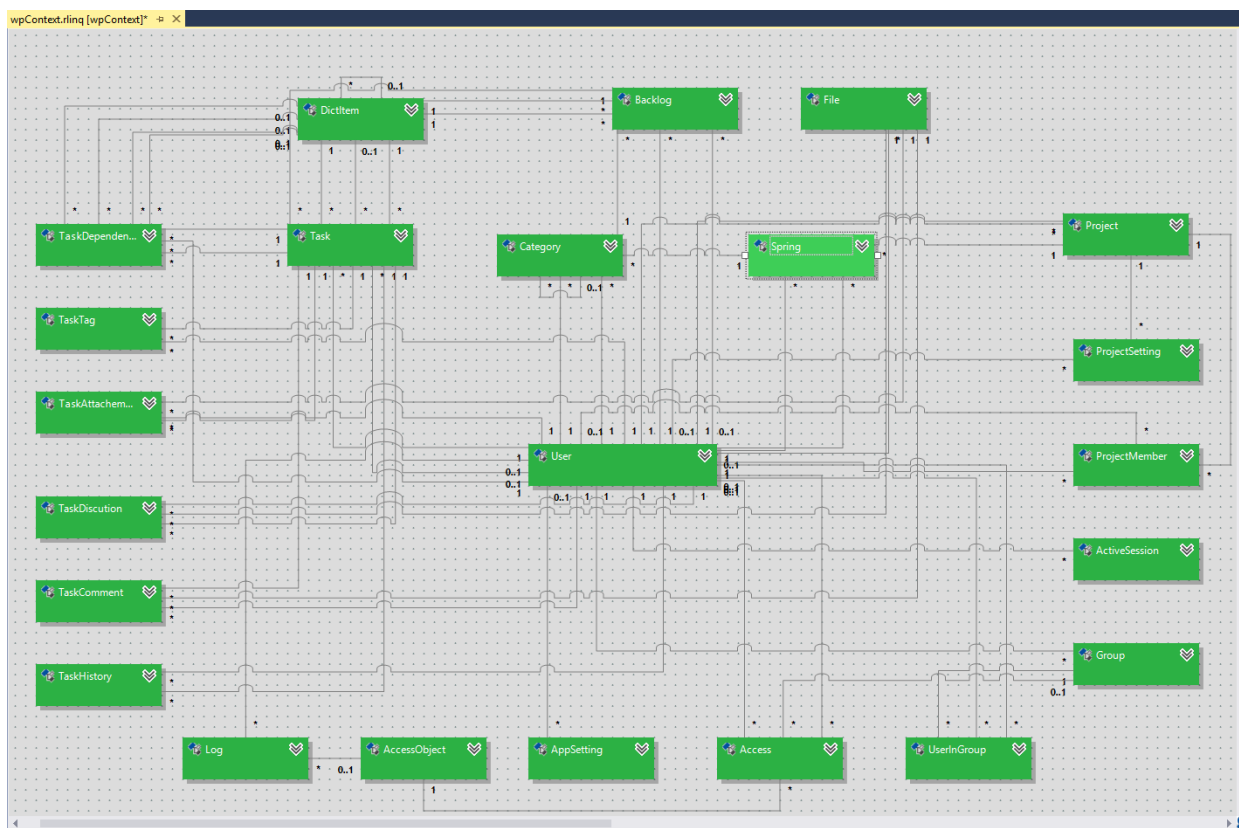
În acest fișier se află toate metodele ce sunt invocate din CallServiceMethod. Metodele ce nu depind de lista de utilizatori sunt declarate statice, iar restul sunt normale. Câteva dintre metode sunt :

- GetTask – Metodă ce returnează toate informațiile despre un task
- ChangeTaskState – Metodă ce schimbă starea unui task cu starea primită în cerere
- GetAllProjects – Ce serializează un obiect de tip GetAllProjectsResponse, acesta la rândul său conține o colecție de proiecte (Figură 16)



Figură 16 - Clasă de răspuns pentru preluarea proiectelor

Proiectul de back-end are referință cu WProject.DataAccess ce conține toate clasele „mappate” la baza de date generate de Telerik Data Access. Aceste au fost generate automat de către extensie, acesta oferă și un designer grafic (Figură 17) cu posibilitatea recreării claselor după modelul bazei de date, și chiar creării bazei de date după modelul din designer. Acesta oferă un suport primar pentru mutarea elementelor, redimensionarea lor, rearanjarea legăturilor și chiar suport pentru export ca imagine png.



Figură 17 - Diagramă clase generate

Clasele, împreună cu proprietățile și câmpurile, sunt generate automat, fiecare element fiind-ui setat atribute pentru relaționarea cu baza de date, iar fiecare clasă are un atribut `Table` ce primește ca parametru numele tabelului cu care relaționează, iar fiecare proprietate deține un atribut `Column` ce primește ca parametri : numele coloanei asociată în baza de date, tipul coloanei (int, float, datetime, varchar, text) și încă o proprietate ce marchează dacă câmpul asociat în baza de date este cheie primară. Toate proprietățile vor fi populate folosind reflexie pe clasa respectivă. De exemplu, pentru popularea unei entități de tip `User`, librăria va încărca în memorie toate proprietățile și va căuta în baza de date tabelul din atributul `Table`, declarat la începutul clasei, dacă tabelul nu este găsit librăria va arunca o excepție¹² de tipul `TelerikTableNotFoundException`, în caz contrar librăria va continua cu procesarea proprietăților și a atributelor fiecăruia. Pentru fiecare proprietate ce deține un atribut `Column`, se adaugă ca câmp, numele coloanei din atribut, în construcția interogării. La fel ca în cazul tabelului, dacă coloana nu există în tabel, librăria va arunca o excepție de `TelerikColumnNotFoundException`.

¹² A arunca o excepție – a întrerupe execuția codului până în momentul în care este „tratată”

Dacă toate coloanele au fost găsite și interogarea a fost construită, aceasta este executată apoi folosind reflexia se creează obiectele și se setează valorile pentru fiecare câmp. Clasele generate fiind parțiale, le-am putut extinde adăugând metodele statice `ToWebApi` ce primești ca parametru un obiect din `DataAccess` și returnează un obiect cu aceleași proprietăți dar delegate de contextul bazei de date și metoda statică `FromWebApi` ce primește un obiect din librăria `WebApiClasses` și returnează un obiect din librăria `DataAccess`.

```
public static Wproject.DataAccess.Task FromWebApi(WebApiClasses.Classes.Task task)
{
    try
    {
        if (task == null)
            return null;

        return new Wproject.DataAccess.Task
        {
            Id = task.Id,
            Name = task.Name,
            CreatedAt = task.CreatedAt,
            CreatedById = task.CreatedById,
            AssignedToId = task.AssignedToId,
            StateId = task.StateId,
            TypeId = task.TypeId,
            StageId = task.StageId,
            BacklogId = task.BacklogId,
            Priority = task.Priority,
            PeriodFrom = task.From,
            PeriodTo = task.To,
            RemainingWork = task.RemainingWork,
            WorkDate = task.WorkDate,
            StartHour = Utils.GetDateFromTimeStamp(task.StartHour),
            EndHour = Utils.GetDateFromTimeStamp(task.EndHour),
            Description = task.Description,
            Deleted = task.Deleted.If((short?)1, null)
        };
    }
    catch
    {
        return null;
    }
}
```

Pe lângă aceste metode existente în toate clasele generate de Telerik, unele clase au și metode ce primesc încă un parametru de tip `wpContext` supraîncărcând metoda

```
ToWebApi(WProject.DataAccess.Task):WebApiClasses.Classes.Task
ToWebApi(WProject.DataAccess.Task,wpContext):WebApiClasses.Classes.Task
```

Acestea populează și obiectele legate în baza de date legate prin chei străine, spre exemplu Utilizator, Backlog, Comentarii, etc...

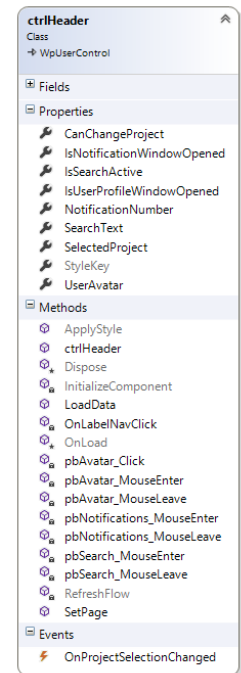
Front-end

Fereastra principală conține 3 controale și o fereastră copil ce este folosită ca fereastră de overlay¹³ în momentul afișării unui formular, aceasta să acopere fereastra principală, totuși fereastra de overlay poate fi afișată oricând folosind metoda ShowOverlay(true), din clasa statică UIHelper. Controalele principale din fereastră sunt :

- **ctrlHeader**

Reprezintă antetul aplicației ce conține pe primul rând se află un control de tip comboBox ce servește drept selector pentru proiect cu posibilitatea adăugării evenimente ce sunt declanșate la schimbarea selecției proiectului folosind evenimentul OnProjectSelectionChanged (Figură 18)

```
public event SelectedItemChangeHandler OnProjectSelectionChanged
{
    add
    {
        if (ddProjects != null)
            ddProjects.SelectedItemChanged += value;
    }
    remove
    {
        if (ddProjects != null)
            ddProjects.SelectedItemChanged -= value;
    }
}
```



Figură 18 - Controlul de antet

În partea stângă se află butonul de căutare, notificările și imaginea de profil, care odată apăsată deschide un pop-up¹⁴ cu imaginea utilizatorului, numele lui, adresa de email și butoane de tip link pentru setări de profil și deconectare. Pe următorul rând se află lista modulelor : Dashboard, Timeline, Chat și Administrare. Acționarea unui buton din lista modulelor invocă metoda ShowPage(Pages):void din ctrlMainPanel.

- **ctrlLoginControl**

Este controlul de conectare în care utilizatorul își introduce numele și parola, cu posibilitatea de ași salva numele și parola local fără a fi nevoie introducerea datelor de conectare la următoarea deschidere a aplicației. La apăsarea butonului de conectare se execută în mod asincron, pentru a nu bloca interfața utilizatorului [6], conectarea pe server folosind metoda ExecuteLogin(string, string):Task<LoginResponse> ce face cererea pe server și actualizează lista clienților conectați cu utilizatorul curent. Dacă datele de conectare sunt incorecte, se afișează un mesaj corespunzător.

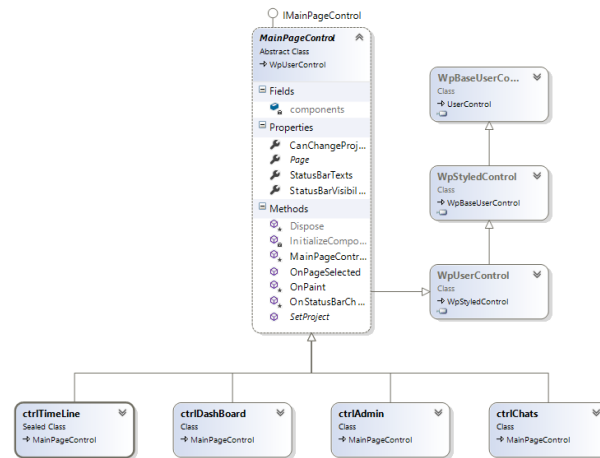
¹³ Fereastră overlay – fereastră gri transparentă ce marchează inactivitate ferestrei părinte și setează atenția asupra altui element de deasupra ferestrei overlay

¹⁴ Pup-up sau popup – control similar cu cel de meniu contextual ce conține informații și controale ce au legătură cu butonul declanșator

- **ctrlMainPanel**

Acesta este control-ul principal ce conține toate celelalte module și controlul de status. Controalele ce conțin modulele sunt organizate într-o colecție de tip dicționar cheie valoare, având cheia enumerarea de tip Pages iar ca valoare reține valori de tipul MainPageControl. Controalele se instanțiază în constructorul controlului. Pagina implicită ce se afișează este Dashboard-ul încă din constructor folosind metoda ShowPage(Pages):void. Clasa abstractă MainPageControl este folosit și extins de toate modulele. (Figură 19)

```
public enum Pages
{
    DashBoard,
    TimeLine,
    Chats,
    Admin
}
```



Figură 19 - Model controale panou principal

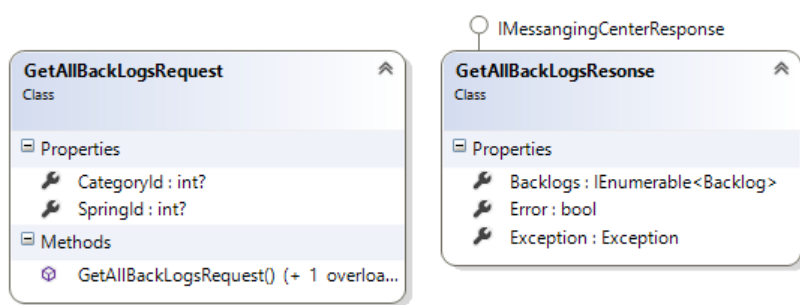
Dashboard

Controlul de Dashboard suprascrie următoarele metode moștenite de la MainPageControl:

- **SetProject(Project):void**
Ce implementează comportamentul modului atunci când proiectul este schimbat din selectorul din ctrlHeader.
- **OnStatusBarChatClick():void**
Ce suprascrie acțiunea ce se execută la apăsarea butonului de chat din bara de stare.
- Proprietatea **StatusBarText:string[]**
Ce setează cele 3 câmpuri de text din partea dreaptă a controlului de stare.

Controlul de dashboard are în componența sa două controale, unul ce conține iterațiile și categoriile, și controlul ce conține backlog-urile și task-urile. Controalele fiind separate printr-un Panel ce, odată dat click pe el, controlul de iterații și categorii se ascunde, iar cel de task-uri va folosi întreaga suprafață astfel întreaga suprafață utilă va fi folosită în întregime pentru urmărirea task-urilor. La schimbarea selecției unei iterații sau categorii, este memorată în sistem selecția, astfel încât la următoarea pornire a programului se va face selecția automată bazându-se pe ultima salvare făcută, și se reîncarcă panoul cu task-uri ținând cont de selecția curentă. Controlul de iterații

este populat la prima deschidere a controlului și la schimbarea proiectului din controlul antet folosind metoda `SetProject(Project):System.Threading.Task` ce va curăța arborele cu iterații apoi apelând metoda `GetSprings(int, int):System.Threading.Task<GetSpringResponse>` de pe server, aceasta executându-se în mod asincron, aceasta returnând un răspuns cu toate iterațiile și categoriile într-un mod arborescent. După primirea răspunsului se începe popularea controlului de tip arbore în mod recursiv. În momentul în care se face click pe un element din control se execută un evenimentul `SelectedSpringOrCategoryChaged(Spring, Category):void`. Odată încărcat controlul de iterații, se începe popularea controlului de task-uri. Folosind metoda `LoadTasks(Spring, Category):System.Threading.Task<WProject.WebApiClasses.Task>` ce curăță toate task-urile existente apelează în mod asincron metoda `GetAllBackLogs(int?,int?):System.Threading.Task<GetAllBacklogsResponse>` ce returnează un obiect de tipul `GetAllBacklogsResponse`, acesta conținând o colecție de backlog-uri (Figură 20), fiecare backlog având la rândul lui o colecție de task-uri, pe lângă informațiile proprii.



Figură 20 - Clasele cerere și răspuns pentru preluarea backlog-urilor

Odată primit răspunsul, în panoul principal, ce se redimensionează automat în funcție de conținut, se vor adăuga controalele de tip `ctrlDashboardItem` folosind metoda `AddRange(Control[]):void` a panoului cu task-uri, la sfârșit apelându-și metoda statică din `UIHelper UpdateStatusBarTexts():void` ce reîncarcă textele din bara de stare.

Timeline

Controlul de vizualizare a task-urilor în mod Timeline suprascrive proprietatea `StatusBarVisibility:bool` returnând valoarea false pentru a nu afișa bara de stare în modulul Timeline. Controlul conține antetul de rânduri și tabelul de task-uri pe ore. Încărcarea task-urilor se face folosind metoda `SetTasks(IEnumerable<Task>,ITaskAddableControl):void` din controlul `ctrlTimeLineRowHeader` mai departe parametru `tasks` se va grupa pe utilizatorul asignat task-ului. Pentru fiecare utilizator se trimite prin parametri id-ul utilizatorului, lista de task-uri ale utilizatorului, și obiectul ce implementează interfața `ITaskAddableControl` ce creează un obiect de tipul `ctrlTimeLineRowHeaderUser` apoi apelându-se mai departe `SetBacklogs(IEnumerable<Task>,ITaskAddableControl):void` această metodă grupează task-urile după backlog și le adaugă sub controlul de „userRow” și se apelează metoda `AddTasks(IEnumerable<Task>,int):int` obiectului `taskAddable`, trimis prin parametru, ce

implementează interfața `ITaskAddableControl` aceasta returnând înălțimea controlului creat. Acest lucru întâmplându-se pentru fiecare utilizator. La sfârșitul adăugării controalelor se execută metoda `ResizeControl():void` ce calculează totalul necesar pentru toate controalele și îl atribuie înălțimii controlului părinte, iar dacă controlul părinte este mai mare decât containerul în care se află, containerului i se afișează controlul de tip scroll. În Figură 21 putem observa plasarea controalelor în Modulul de Timeline.



Figură 21 - Control-ul de Timeline

Fiecare `ctrlTimeLineTaskItem` dintr-un `ctrlTimeLineRow` poate fi redimensionat pe orizontală trăgând din marginea dreaptă pentru a mări / micșora timpul stimat al task-ului. Pe același principiu un task poate fi tras cu mouse-ul și mutat pe direcția orizontală pentru a modifica ora de început a task-ului. Fiecare task având `ToolTip-uri`¹⁵ ce indică ora de început și ora de sfârșit, calculată pe baza timpului estimat introdus. Task-urile ce nu au timp estimat vor fi afișate în grafic cu un timp estimat de 60 de minute (o oră) și un semn al întrebării în dreptul numelui. Toate controalele (excepție făcând `ctrlTimeLineRowHeader` și `ctrlTimeLineTasks`) se generează automat în momentul reîncărcării task-urilor în controlul de Timeline. Toate pozițiile controalelor sunt calculate automat în momentul adăugării lor în controlul părinte respectiv task-ului, acest lucru optimizând complexitatea, aceasta devenind doar $O(N)$. Performanța este o caracteristică cheie în modulul de Dashboard și Timeline deoarece în aceste module se fac operațiuni frecvente, iar utilizatorul nu trebuie să întâmpine nici un fel de întârziere în încărcarea datelor.

¹⁵ `ToolTip` – control ce afișează informații de obicei atunci când cursorul se află peste un alt control.

WebCallFactory

Clasa WebCallFactory este o clasa statică ce conține metode pentru apeluri pe server. Pentru a putea folosi clasa WebCallFactory, la pornirea aplicației se execută metoda InitConnection():System.Threading.Task ce instanțiază câmpul _connection și creează un hub pentru clasa MessagingCenter. După crearea hub-ului se încearcă pornirea conexiunii folosind metoda Start():System.Threading.Task.

```
public static async Task InitConnection()
{
    _connection = new HubConnection(Settings.Default.DispatcherUrl);
    _hubProxy = _connection.CreateHubProxy("MessagingCenter");
    try
    {
        State = ConnectionState.Connecting;

        await _connection.Start();
        ConnectionIsAlive = true;
        State = ConnectionState.Connected;
    }
    catch (Exception mex)
    {
        State = ConnectionState.NotConected;
        ConnectionIsAlive = false;
        Logger.Log(mex);
    }
}
```

Metodele din WebCallFactory depind de inițializarea conexiunii deoarece în metoda de cerere la baza de date se folosește proprietatea statică Hub din clasa Connection ce încapsulează câmpul _connection. Toate metodele pentru apelarea serviciilor web folosesc metoda de bază ExecuteMethod(string, object):System.Threading.Task<MessagingCenterResponse> ce face un apel asincron la server invocând metoda CallServiceMethod de pe server, incluzând un pachet MessagingCenterPackage ce incluzând numele metodei, informațiile despre client (cel care a făcut cererea) și conținutul cererii serializat în JSON.

```
public static async Task<MessagingCenterResponse> ExecuteMethod(string method, object content)
{
    try
    {
        if (!Connection.ConnectionIsAlive)
            throw new ExecuteServerMethodException("Connection is not alive");

        MessagingCenterPackage mpackage = new MessagingCenterPackage(Connection.FromAddress, method)
        {
            Content = content != null ? JsonConvert.SerializeObject(content) : string.Empty
        };
        Connection.NetworkTransferInProgress = true;
        return await Connection.Hub.Invoke<MessagingCenterResponse>("CallServiceMethod", mpackage);
    }
    finally
    {
        Connection.NetworkTransferInProgress = false;
    }
}
```

Clasa `MessagingCenterResponse` conține o proprietate numită `Content` (Figură 15) ce conține răspunsul serializat dat de server, acesta fiind deserializată în clasa dorită. Totuși pentru a optimiza procesul de dezvoltare a funcțiilor similare, am supraîncărcat metoda `ExecuteMethod` făcând-o generică returnând rezultatul deserializat în mod automat.

```

/// <summary>
/// Execute a method from server using SignalR Hub
/// </summary>
/// <typeparam name="T">Type of response expected</typeparam>
/// <param name="method">Method name to call</param>
/// <param name="content">Request parameters requested in WebMethod</param>
/// <param name="continueThrow">In any error case the exception is caught and error is return within
<value>IMessagingCenterResponse</value> object, if flag is set on true, the exception will throw on
the exception</param>
/// <param name="checkForEmpty">In case of response <value>Content</value> property is null or empty a
n exception will throw, if flag is set on false the method will return a new instance of <value>T</val
ue></param>
/// <returns>Deserialized content of response into type <value>T</value></returns>
public static async Task<T> ExecuteMethod<T>(string method, object content, bool continueThrow = false
, bool checkForEmpty = true)
    where T : IMessagingCenterResponse, new()
{
    try
    {
        var mres = await ExecuteMethod(method, content);

        if (mres == null)
            throw new Exception($"[{method}]Response is empty");

        if (mres.ErrorCode != MessagingCenterErrors.NO_ERROR)
            throw new WpException(mres.ErrorCode, mres.Error, mres.Exception);

        if (string.IsNullOrEmpty(mres.Content))
            if (checkForEmpty)
                throw new Exception($"[{method}]Content is empty");
            else
                return new T();

        return JsonConvert.DeserializeObject<T>(mres.Content);
    }
    catch (Exception mex)
    {
        if (continueThrow)
            throw;

        Logger.Log(mex);
        return new T
        {
            Error = true,
            Exception = mex
        };
    }
}

```

Concluzii

Ca un proiect să fie dus la bun sfârșit și să se poată interveni cu modificări și îmbunătățiri în orice moment, acesta trebuie proiectat foarte bine și necesită o arhitectură puternică pentru a rezista pe viitor. De aceea un manager de echipă trebuie să coordoneze membri proiectului într-un mod strategic și cu un plan bine structurat pentru un consum minim de resurse și într-un timp cât mai scurt posibil. Principala provocare a unui manager de proiect este să îndeplinească toate sarcinile sub condițiile impuse de client [7], principalele constrângeri fiind timpul, bugetul și calitatea.

Sistemul a fost dezvoltată în limbajul C#, acesta fiind împărțit în două aplicații ce reprezintă front-end-ul și back-end-ul. Acest lucru facilitează posibilitatea extinderii front-end-ului și pe alte platforme, nefiind necesar aducerea de actualizări back-end-ului, acesta urmând într-o oarecare măsură modelul aplicațiilor în cloud. În zilele noastre tehnologia evoluează într-un mod foarte rapid iar modelele și uneltele de dezvoltare se îmbunătățesc și apar altele noi, astfel încât o aplicație trebuie să urmeze anumite reguli și modele de dezvoltare pentru a ușura eventuale modificări de cod.

Pe viitor doresc să extind aplicația și mai mult adăugând cel puțin două module

Queries – ce oferă posibilitatea ca utilizatorul să își creeze filtre și criterii de ordinare personale.

Views – modul pentru comunicarea cu clientul într-un mod mai eficient prin modul de chat și vizualizare stării proiectului cu iterații și task-uri.

Pe lângă adăugarea celor două module doresc să implementez aplicația pe dispozitive mobile și web. Proiectul este open-source disponibil pe github accesând link-ul <https://github.com/AbabeiAndrei/WProject> și este disponibil sub licența MIT License¹⁶ astfel încât oricine poate folosi aplicați. Aceasta poate fi găsită în directorul Release, dar se poate utiliza codul sursă în egală măsură.

¹⁶ MIT License – Tip de licență pentru aplicațiile gratuite ce permite ca aplicația să fie utilizată, copiată, modificată, distribuită, vândută și utilizare în scop comercial. Toate acestea cu condiția de a fi inclus fișierul LICENCE în distribuție, drepturile de autor fiind păstrate [9].

Bibliografie

- [1] M. P. Follett, „infed,” 2002. [Interactiv]. Available: <http://infed.org/mobi/mary-parker-follett-community-creative-experience-and-education/>. [Accesat 09 08 2016].
- [2] K. Beck, Embracing Change with Extreme Programming, Computer, 1999.
- [3] J. Qiao, „Mircosoft Forums,” 08 07 2013. [Interactiv]. Available: <https://social.msdn.microsoft.com/Forums/vstudio/en-US/68b5d67b-8769-4fdc-b056-bf51e421b0b8/howwhere-to-set-workdays-for-team-members-in-tfs-what-about-public-holidays?forum=TFService>. [Accesat 10 08 2016].
- [4] E. International, „C# Language Specification,” 06 2006. [Interactiv]. Available: <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf>. [Accesat 20 07 2016].
- [5] D. Mohl, Building Web, Cloud, and Mobile Solutions With F#, O'Reilly Media, Inc., 2012.
- [6] Microsoft, „async (C# Reference),” [Interactiv]. Available: <https://msdn.microsoft.com/en-us/library/hh156513.aspx?f=255&MSPPErr=-2147217396>. [Accesat 20 08 2016].
- [7] J. Phillips, în *PMP Project Management Professional Study Guide*, McGraw-Hill Professional, 2003, p. 354.
- [8] D. F. Jez Humble, în *Continuous Delivery: reliable software releases through build, test, and deployment automation*, Pearson Education Inc, 2010, p. 110.
- [9] K. W. (kevin), „TLDRLegal,” 2014. [Interactiv]. Available: <https://tldrlegal.com/license/mit-license>. [Accesat 1 09 2016].