

Ministerul Educației Naționale și Cercetării Științifice

Universitatea OVIDIUS Constanța

Facultatea de Matematică și Informatică Specializarea Informatică

Aplicație web pentru managementul proiectelor

Lucrare de licența

Coordonator științific

**Lect. univ. Dr. Alexandrescu Adrian**

Absolvent

**Ababei Andrei**

Constanta 2016

# Capitolul 1

## Motivație

.

## 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 dar si o scurtă introducere al conceptului **DevOps**.

### 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 (*springs*). 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.

### 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.

Analiză

Design

Code

Test

Analiză

Design

Code

Test

Waterfall

Agile

### DevOps

DevOps este un set de practici ce asigură colaborarea și comunizarea dintre dezvoltatori și departamentul IT (testare, suport și tehnic), astfel încât din partea departamentului de dezvoltare

## Tema proiectului

In aceasta lucrare voi prezenta demersul meu pentru realizarea unei interfețe si a unui sistem independent pentru managementul proiectelor folosind modelul **Agile**. Aplicația este numită **WProject** este de fapt împărțită în doua 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 si* ***Dispecer****)* se procesează toate cererile trimise de pe toate front-end-urile. In 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 si medii deoarece este ...

TODO DE CONTINUAT

### 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. TODO DE MODIFICAT
  + 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 si grupe secundare create de administratori pentru a ușura atribuirea de drepturi, de exemplu :

* Developers
* Testers
* Support

### Sistemul de task-uri

Task-ul este entitatea principala și unitatea de măsura 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 în de obicei se referă, într-un mod foarte generic, ce trebuie făcut. TODO DE MODIFICAT
* Descrierea – informațiile esențiale explicate în detaliu necesare pentru a închide task-ul[[1]](#footnote-1)
* 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 doua feluri :

* Funcționalitate
* Bug (*problemă*)

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.

Backlog

Task

Task

Task

Task

Backlog

Categorie

Iterație

Proiect

# Capitolul 2

## 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 si interportabilitate 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 sa fie un limbaj simplu, modern, orientat pe obiecte și inovativ.

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. [TODO DE CONTINUAT]

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 UI[[2]](#footnote-2) cu WinForms, am ales WinForms deoarece este inclusă in .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.

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

Conectarea se face folosind un ORM[[3]](#footnote-3), 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; }

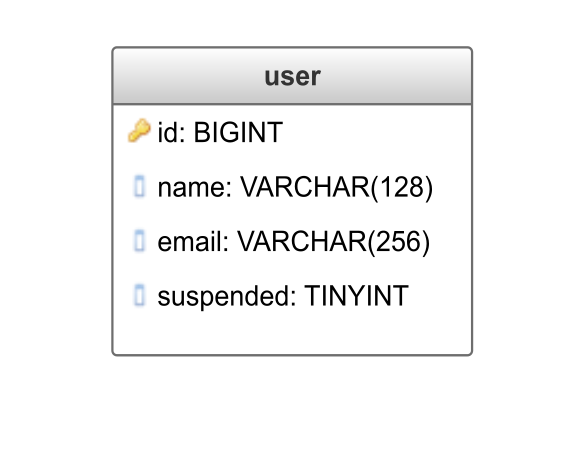
    [Column("email", OpenAccessType = OpenAccessType.UnicodeString)]

    public string Email { get; set; }

[Column("suspended", OpenAccessType = OpenAccessType.Boolean, IsNullable = true)]

    public bool Suspended { get; set; }

}



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, ex:

using (var context = WpContext.CreateContext)

{

    User user = context.Users.FirstOrDefault(u => u.Name == "Andrei");

    if (user != null)

{

user.Suspended = true;

context.SaveChanges();

}

}

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 comit[[4]](#footnote-4) modificările.

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

Această librărie oferă un canal de comunicare bi-direcțional între client și server folosind Sockeți Web 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.

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.

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; }

}

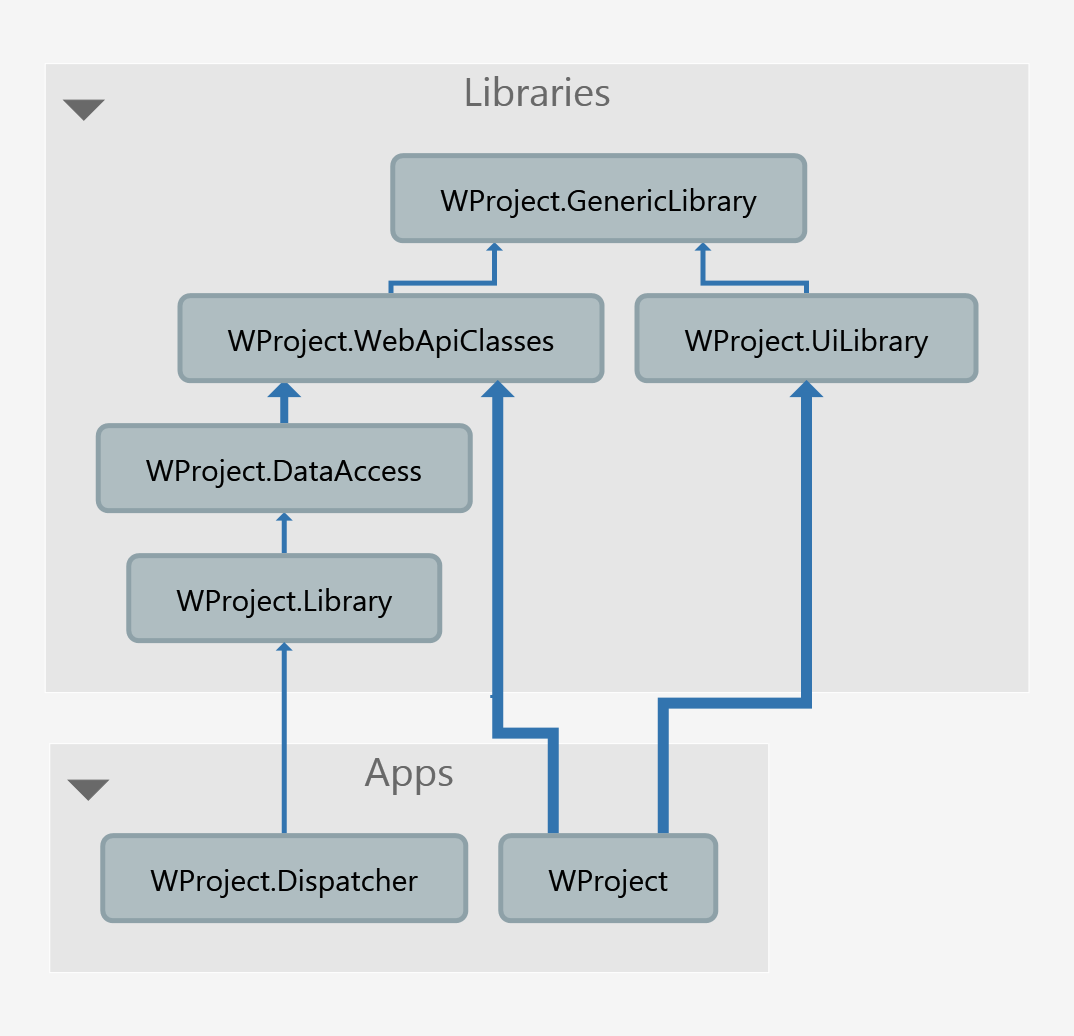
}

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.



* 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

[TODO CONTINUARE]

# Capitolul 3

## 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)

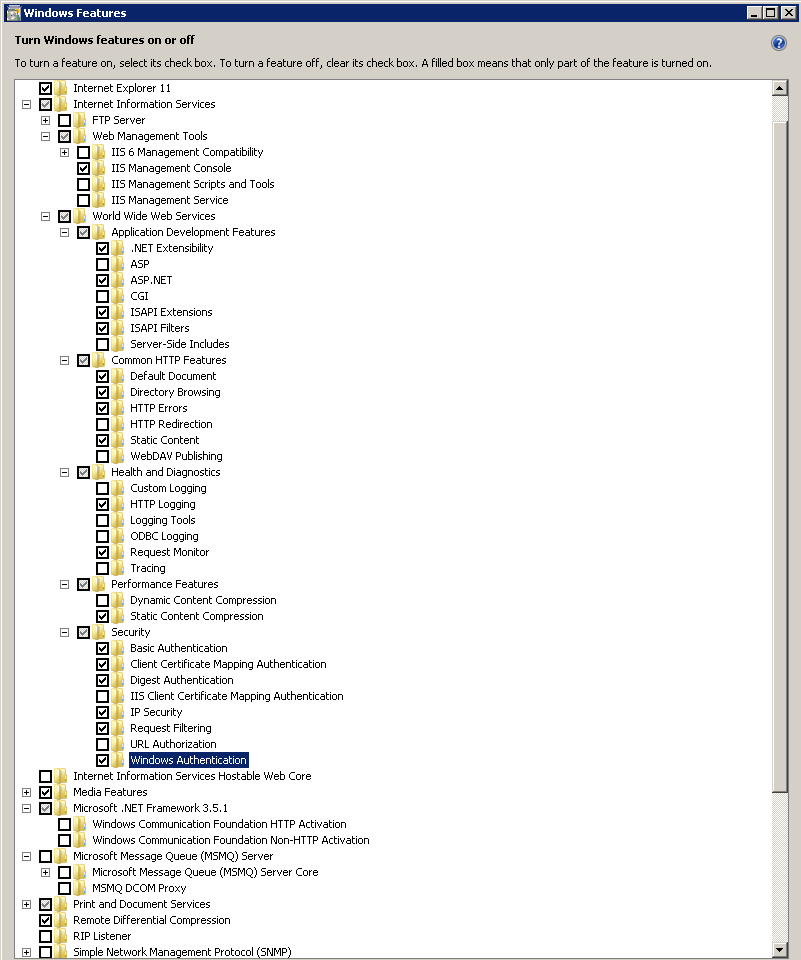
Deoarece este o back-end-ul este o aplicație ASP.NET este necesar instalarea IIS (Internet Information Services) minim versiunea 7.0.

Figura 1 – Instalare IIS

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. (*Figura 1*)

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.

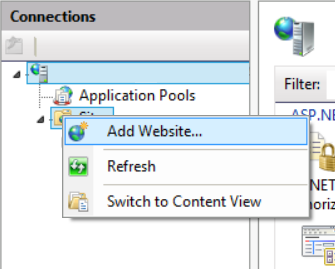
Mai departe, se intră în MySQL Command Line Client (se poate găsi în Start, sau în C:\Program Files\MySQL\MySQL Server 5.x\bin\mysql.exe). Odată pornită linia de comandă se va introduce parola de root, apoi se va crea o bază de date folosind comanda

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 apăruta 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...” (*figura 2*). 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**.**

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**.

Figura 2 – Adăugare site în IIS

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>

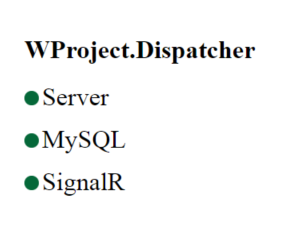
</setting>

<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.

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

### 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 fereastra 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.

### Crearea utilizatorilor

În primul rând va fi configurat utilizatorul **admin**. El va fi obligat să își introducă un e-mail, să își introducă o parola (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.

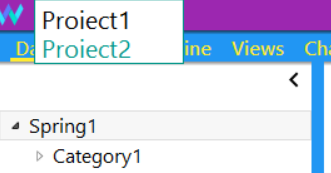
### Crearea proiectelor

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 [TODO DE COMPLETAT]

## 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.

### Dashboard

Dashboard-ul reprezintă metoda principală și cea mai ușoară 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 dorita (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.

..Imagine..

Când este închis se poate observa sub titlul cui aparține backlog-ul, iar in 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.

...Imagine

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 respectiva 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, 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.

.. imagine..

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.

...imagine..

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.

...imagine...

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

1. A închide un task – a termina un task [↑](#footnote-ref-1)
2. UI – Interfață cu Utilizatorul (User Interface) [↑](#footnote-ref-2)
3. ORM – Object-relational mapping – Tehnică de convertirea a modelului bazei de date in obiecte compatibile pentru manipularea programatică [↑](#footnote-ref-3)
4. Comit – a aplica modificări, a salva [↑](#footnote-ref-4)