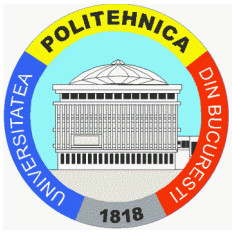


UNIVERSITATEA POLITEHNICA BUCURESTI

FACULTATEA DE AUTOMATICA SI CALCULATOARE

TASKZILLA

Documentul de Proiectare arhitecturala

CHITU MARIA
ENE DANIELA
FRINCU AMALIA
GHIDARCEA ANAMARIA

Indrumator:
ALEXANDRU GRADINARU

Aplicatia Taskzilla

Proiectare arhitecturala

Introducere

Scopul sistemului

“Taskzilla” reprezinta o aplicatie de gestiune a task-urilor si a sarcinilor in cadrul unei firme/ familii/ organizatii. In aplicatie exista doua tipuri de utilizatori: worker si team leader, iar principiul de functionare este urmatorul: team leaderii propun task-uri worker-ilor si urmaresc activitatea acestora, iar workerii lucreaza la task-urile asignate lor si actualizeaza evolutia acestora prin schimbarea statusului si prin adaugarea de comentarii. Atunci cand un team leader adauga un nou task, acesta se afla in starea „propus‟, iar cand worker-ul incepe sa lucreze la el va trece in starea „activ‟. Odata cu finalizarea sa, worker-ul va schimba statusul task-ului in „rezolvat‟, iar dupa ce are loc verificarea si validarea task-ului de catre team leader, task-ul va trece in stadiul „inchis‟.

Documente referinte

- Documentul de specificare a cerintelor

Obiective de proiectare

Criterii de proiectare

- **Criterii de performanţă**

  Timpul de raspuns al sistemului va fi determinat de timpul de executie al procedurilor stocate din baza de date. Astfel query-urile se vor termina proportional cu dimensiunea tabelelor interogate. Cum aplicatia este una locala, serverul ce pastreaza baza de date va contine doar informatiile companiei care foloseste aplicatia, dimensiunea datelor nefiind o problema. Throughput-ul si memoria sunt de asemenea importante pentru serverul ce contine baza de date. Fiind un sistem modelat folosind o arhitectura simpla si eficienta, nu exista cerinte de sistem speciale privind *memoria*.

- **Criterii de încredere**

  Robustetea sistemului este data de verificarea viabilitatii si integritatii datelor, atat in aplicatia client de C#, cat si in procedurile stocate din baza de date. *Fiabilitatea* este asigurata de fluxul predefinit prin ecrane WindowsForms al aplicatiei client. *Disponibilitatea* sistemului este influentata de disponibilitatea bazei de date care este

nucleul sistemului. *Capacitatea sistemului de a functiona in situatii de cadere* poate fi asigurata prin montarea unor dispozitive specializate ce asigura protectia echipamentelor si datelor in astfel de situatii (Network, Server & Storage UPS). *Securitatea* este asigurata de caracteristica sistemului de a functiona doar in interiorul unei retele si de masurile de protectie impotriva atacurilor de tipul SQL Injection.

- **Criterii de cost**

  Costul dezvoltarii sistemului initial este cel mai semnificativ din intreg procesul. *Costul tranzitiei la utilizator* este unul redus deoarece implica doar instalarea aplicatiei pe statiile client, configurarea in retea a unui server ce va contine baza de date si replicarea acesteia folosind custom scripts. *Actualizarea si mentenanta* se vor face prin update-uri periodice ce vor contine repararea defectelor sau eventuale imbunatatiri. Aceste se vor face de asemenea printr-un sistem automat de update, folosind pachete ClickOnce si pachete obtinute in urma build-urilor din TFS.

- **Criterii de mentenanţă**

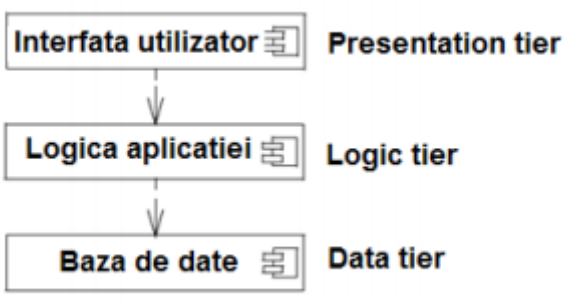
  Sistemul poate fi extins cu usurinta prin dezvoltari ulterioare si update-ul versiunii aplicatiei aflata deja in productie. Sistemul poate fi portat doar pe platforme Windows, nu exista suport pentru alte sisteme de operare. *Claritatea codului* este data de organizarea specifica .NET, folosind C# pentru backend si WindowsForms pentru frontend. *Trasabilitatea cerintelor* este data de modularizarea codului si organizarea functionalitatilor in frunctie de ecranele aplicatiei.

Arhitectura propusa

Prezentare generala a arhitecturii sistemului

Sistemul este modelat folosind arhitectura Three-tier, fiind o arhitectura inchisa structurata pe 3 niveluri.

- *Nivelul interfaţă utilizator – Presentation tier* este reprezentat de utilizarea form-urilor WindowsForms si controalelor Infragistics.
- N*ivelul aplicaţie (application logic) – Logic tier* este modelat folosind limbajul C#, organizarea aplicatiei in clase si interfete functionale.
- *Nivelul stocare (storage) - Data tier este* implementat folosind Transact-SQL.



Decompozitia in subsisteme si responsabilitatile fiecarui subsistem

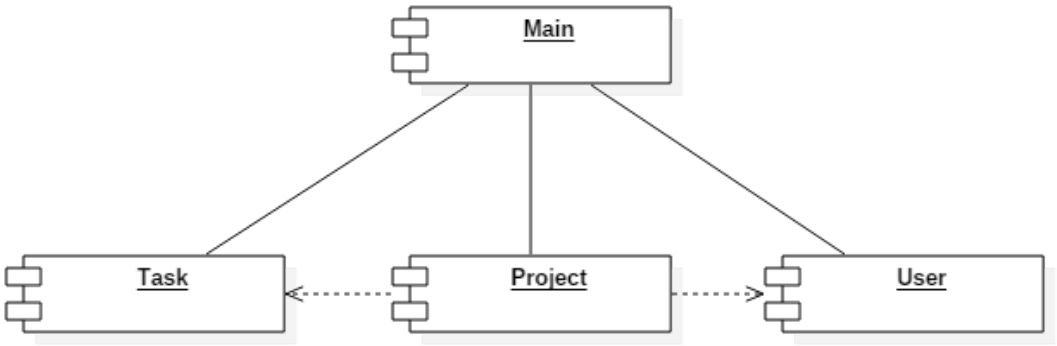
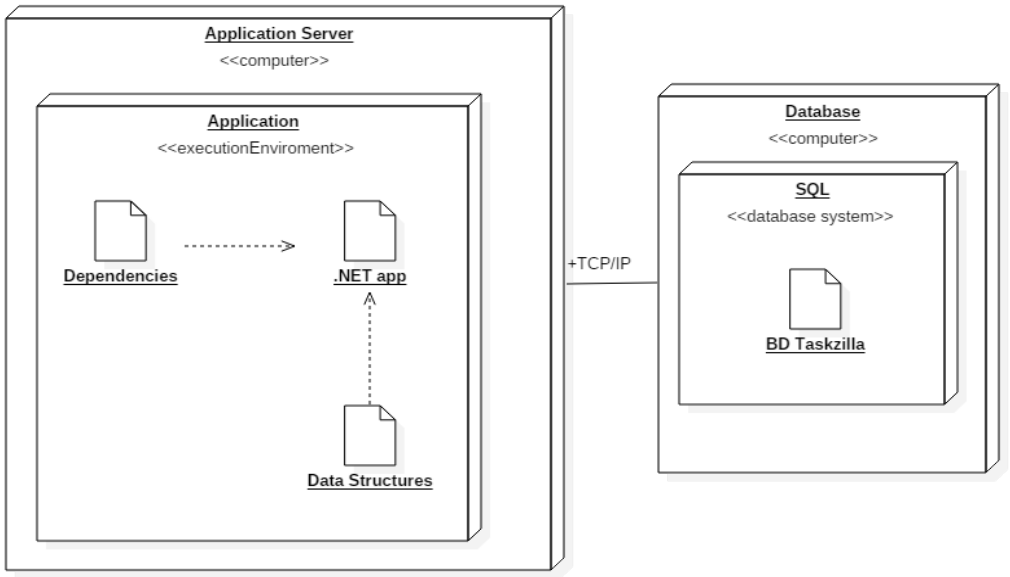


Diagrama de componente ilustrata mai sus prezinta in linii mari structura logica a aplicatiei conturata in jurul componentelor principale.

Cele 2 subsisteme care formeaza aplicatia sunt:

- componenta client, reprezentata printr-o aplicatie C#, care va prezenta datele intr-o interfata grafica si va face o parte din prelucrarile necesare sistemului
- componenta server, reprezentata de baza de date SQL, care va stoca persistent datele si, de asemenea, va face o parte din prelucrari prin intermediul procedurilor socate.

Distributia subsistemelor pe platforme hardware/software (diagrama de distributie)



Managementul datelor persistente

Diagrama UML a relatiilor dintre tabelele bazei de date

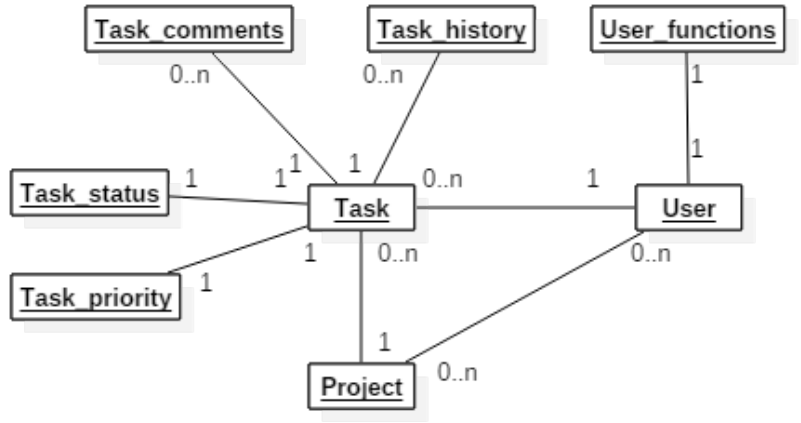


Tabela principala a aplicatiei este tabela Taskuri se afla in relatie one-to-many cu tabelele nomenclator Prioritati si Stausuri si in aceasi relatie cu tabelele Comentarii-Taskuri si Istoric-Taskuri in care se inregistreaza informatii din aplicatie. Tabela Utilizatori se afla in relatie one-to-many cu tabela Taskuri si in relatie many-to-many cu tabela Proiecte. De asemenea tabela Proiecte se afla in relatie one-to-many cu tabela Taskuri.

Diagrama incipienta a structurii bazei de date

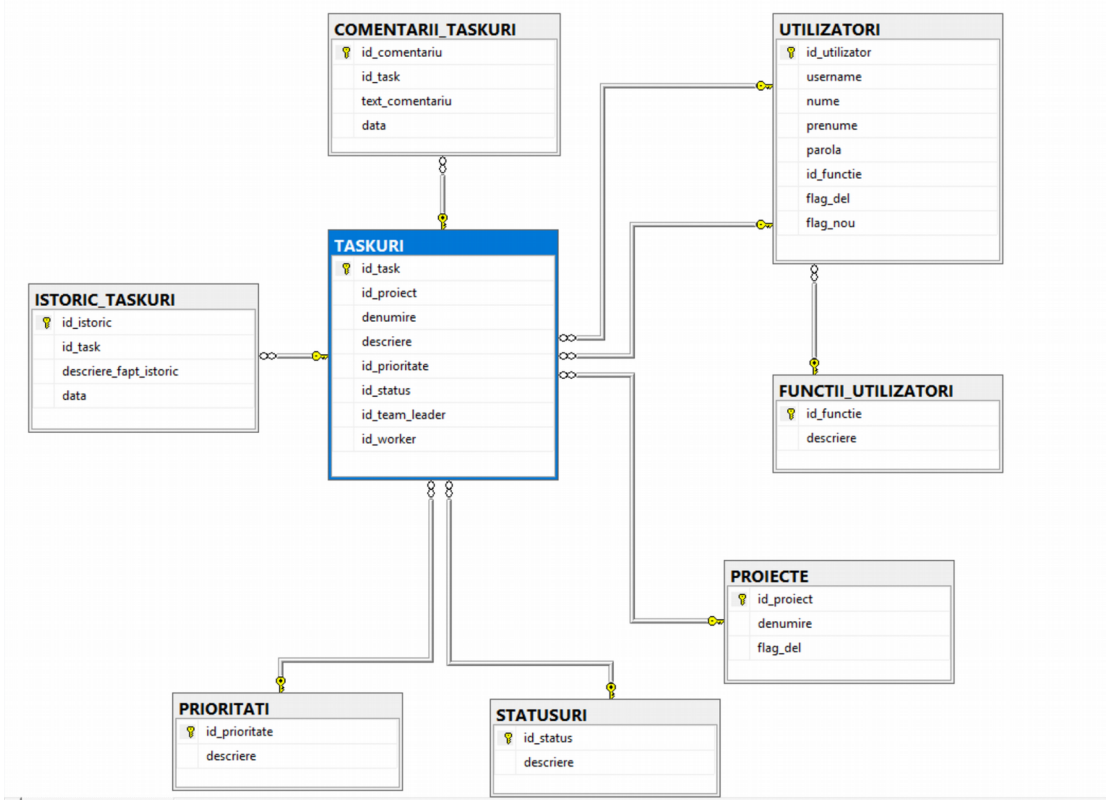
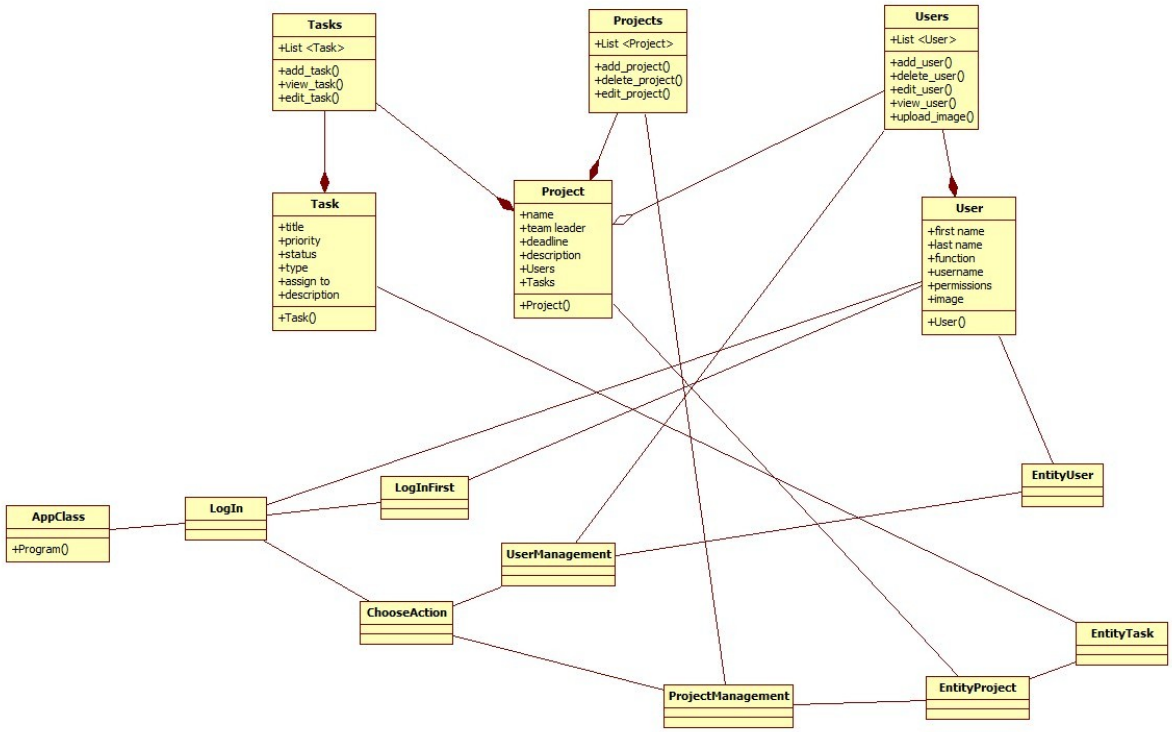


Diagrama de clase



Clasele in care este structurata aplicatia foloseste un model bazat pe colectie si entitate. Cele 3 componente principale au fiecare o clasa entitate care descrie obiectul in sine, atributele sale si metodele specifice, iar clasele colectie descriu atributele si metodele colective. De asemenea, folosind WindowsForms, fiecare ecran, conturat sub forma unui form are propria clasa autogenerata de Designer care la nivel logic, va comunica cu clasele din partea de backend.

Controlul accesului utilizatorilor la sistem

Accesul utilizatorilor la sistem se face printr-un sistem de login pe baza de username si parola.

Utilizatorul, la adaugarea in sistem, are asociata o parola implicita. In baza de date, in tabela Users exista o coloana care determina daca utilizatorul si-a schimbat parola implicita. Astfel, la prima logare, daca utilizatorul, identificat prin username, exista in baza de date avand coloana FirstLogin = 0, trebuie sa isi schimbe parola. Initial in baza de date exista un singur utilizator cu rolul de administrator care poate adauga noi utilizatori. Parolele sunt pastrate in baza folosind un hash.

Comunicatia cu baza se face prin intermendiul procedurilor si parametrilor pentru a preveni SQL Injection.

Fluxul global al controlului

Un flux complet al sistemului este diferentiat in functie de rolul pe care il are utilizatorul.

Daca utilizatorul are rolul de **manager**, rolul ii ofera urmatoarele permisiuni

- creare, stergere, editare a unui proiect
- creare, stergere, editare a unui utilizator
- creare, stergere, editare a unui task

Astfel, dupa logare un manager are dreptul de a alege intre cele 2 functionalitati ale aplicatiei: user management si task management (acest ecran nu este vizibil pentru utilizatorii cu alte roluri). Daca user-ul alege “User management”, poate vizualiza toti utilizatorii din sistem intr-un grid ce prezinta numele, functia si data angajarii. Coloanele prezentate de control pot fi sortate prin click pe header-ul de tabel. Managerul poate edita sau vizualiza un utilizator prin selectia acestuia din grid si apasare unuia din butoanele “View”, “Edit”. Detaliile despre utilizator vor fi prezentate intr-un ecran de entitate utilizator. Procedeul de adaugare va deschide acelasi ecran avand campurie necompletate.

Alegerea functionalitatii de “Task management” deschide un ecran principal care prezinta in partea stanga jos proiectele cu posibilitatea de a adauga, sterge si edita un proiect. La selectia unui proiect, in partea dreapta se deschide un grid ce prezinta toate task-urile asociate unui proiect. Pentru a vizualiza, edita un task, se selecteaza din grid si se apasa butonul corespunzator din partea de sus a ecranului ce va deschide un ecran entitate pentru task. Adaugarea unui task nou deschide acelasi ecran in care sunt prezentate campurile pentru a fi completate.

Daca utilizatorul are rolul de **worker**, rolul ii ofera urmatoarele permisiuni

- creare, stergere, editare a unui task

La primul login, utilizatorul trebuie sa isi schimbe parola predefinita de sistem. Dupa schimbare, utilizatorul este dus in ecranul principal unde poate vizualiza proiectele si taskurile asociate.

Daca utilizatorul are rol de **observer** poate vizualiza aceleasi ecrane ca si utilizatorul cu rol de worker doar ca sunt prezentate in mod read-only.

Conditiile limita (cazurile de utilizare limită)

Configurarea sistemului se imparte in 2 etape. Instalarea aplicatiei pe calculatoarele client prin rularea unui executabil obtinut in urma build-ului aplicatiei si adaugarii pachetelor ClickOnce pentru dependente. Replicarea bazei de date pe statia server se va face prin instalarea tool-urilor necesare (SQL Server, Management Studio) si replicarea bazei prin custom script-uri.

Serverul foloseste politica de transact SQL, astfel in cazul unei *opriri neasteptate* se garanteaza integritatea sistemului si datelor.

Tratarea exceptiilor se face atat in aplicatia C#, prin folosirea blocurilor try, catch, finally si modelului de Exception Handling specific limbajului C#, cat si in procedurile stocate folosind transact SQL si functiei RISEERROR.