Project Finder

Arhitectura sistemului

|  |  |
| --- | --- |
| **Nume proiect** | Project Finder |
| **Nume document** | Arhitectura sistemului / Architectural Design Document |
| **Autori** | Baciu H. Alexandru, Corduneanu Vlad, Simion Emilian |
| **Reviewed by** | - |
| **Status** | În lucru |

***Versiuni***

|  |  |  |
| --- | --- | --- |
| Dată | Versiune | Modificări |
| 19.10.2020 | 0.1 | Structura documentului |
|  |  |  |
|  |  |  |

# Introducere

## Scopul documentului

Documentul de design arhitectural contine detalii despre arhitectura proiectului Project Finder, aplicație ce va fi implementată de echipa de dezvoltare Pawer-Team. Documentul descrie descompunerea întregului sistem în componente separate, interfețele și dependențele fiecărei componente și cerințele pe care acestea le îndeplinesc.

## Scurtă descriere a proiectului

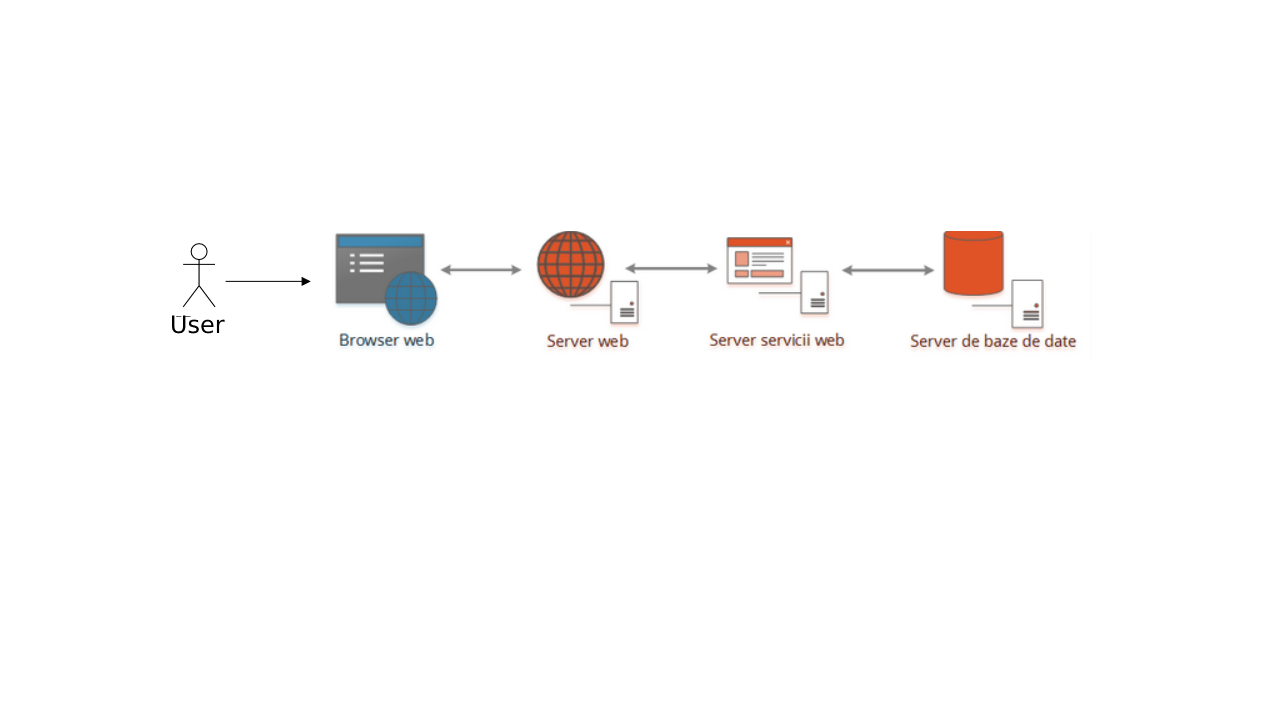
Platforma pune la dispoziție posibilitatea aplicării/creării unui proiect in echipa. Aceasta are ca scop o ușoară rezolvare a problemei găsirii unei echipe pentru realizarea de diferite proiecte.

## Definiții și abrevieri

* Project Finder – platforma pentru găsirea unui proiect in echipa

# Prezentarea sistemului

Project Finder va putea fi accesat prin intermediul browserului care va face cereri către serverul web(serverul Angular) care mai departe se folosește de serverul de servicii web(aplicația realizata cu spring boot) care la randul sau comunica cu un server de baza de date(MySQL) pentru a satisface cererile clienților.



# Decizii de design

## Frameworks

Pentru partea de back-end framework-ul folosit este Spring Boot, iar ca limbaj este folosit Java. Pentru partea de front-end vom folosit Angular, iar ca limbaj typescript si Bootstrap pentru design.

## MySql

Datele vor fi stocate in MySql, baza de date relationala aleasa pentru aces proiect deoarece structura tuturor datelor este fixa, iar intre date exista relatii. S-a prefera MySql pentru usurinta si multitudinea informatiilor pe care le putem gasi despre aceastra pe internet.

## Modularization

Aplicatia web fiind una complexa, partea de back-end se imparte in mai multe module care ne ajuta sa tinem si o cuplare slaba intre componente, comunicarea facandu-se prin intermediul interfetelor. Pe partea de front-end separabilitatea este mentinuta la nivelul modulelor si componentelor din Angular (Angular component pattern).

## Back-end

Pe partea de back-end, partea de granita contine endpoint-urile reprezentate de controllere si datele de tranfer folosite in comunicarea cu clientul. In controllere sunt injectate serviciile care sunt separate pentru fiecare resursa, arhitectura folosita find REST. Partea de comunicare live este realizata prin web sockets si este folosita in aplicatie pentru partea de chat. Serviciile contin un depozit(repository) de functii care permit folosirea entitatilor care vor reprezenta clasele pentru datele din baza de date.

## Front-end

Aplicatia se imparte in doua module. Un modul care contine componenta de logare si header-ul aplicatiei, si un modul care contine toate componentele necesare administrarii proiectelor si profilului utilizatorului. Aceste module se folosesc de diverse servicii pentru comunicare, autentificare si comunicarea intre doua componente.

# Arhitectura top-level

## Identificarea componentelor

Project finder este un proiect destinat utilizatorilor pentru administrarea/ cautarea proiectelor din facultate. Prin urmare, componentele principale sunt reprezentate de **componenete de user** si **componente de proiect**.

Datele despre users si proiecte trebuie sa fie stocate persistent, deci o alta componenta este data de **baza de date mysql**. Comunicarea intre baza de date si servicii este realizata prin **Java Persistence API**. Proiectele stocate vor fi regasite pe partea de client intr un **Feed** prin care user-ul poate actiona(edita, sterge, actualiza, alatura) asupra proiectelor. Feed-ul si **componenta de chat** este accesibil doar utilizatorilor autentificati prin **componenta de autentificare**.

## Relațiile și comunicarea dintre componente

Serviciile din api comunica cu componenta de baze de date prin intermediul middleware-ului JPA care este realizat prin Hibernate(ORM). Componentele de user, chat si proiect se folosesc de servicii pentru a fi stocate persistent si pentru a transmite catre utilizator datele necesare. Componenta de autentificare permite accesul asupra datelor doar utilizatorilor aplicatiei. Componenta de chat depinde de componenta de user, aceasta reprezentand comunicarea intre doi useri.

# Descrierea componentelor

## Back-end

**Baza de date**

Este o componeneta independenta care salveaza datele pentru proiecte, useri si chat. Componenta va avea ca input entitatile transformate in date pentru tabele, iar ca output va oferi informatii despre useri, chat si proiecte. Asupra bazei de date actioneaza comenzile pentru gestiunea tabelelor: update, delete, insert, select, etc.

**Java persistence api**

Aceasta componenta contine toate interfetele care ajuta accesarea bazei de date, comenzi ce vor fi transformate in instructiuni sql asupra entitatilor care vor da campurile de input. Componentele care depind de aceasta sunt serviciile pentru chat, user si proiect care apeleaza metode pentru accesul si modificarea datelor. Ca input primeste informatii pentru a fi scrise in baza de date sau cititite din baza de date, iar ca output ofera obiecte cu informatii din baza de date care pot fi utilizate in aplicatie sau instructiuni pentru stocarea persistenta a datelor.

**Autentificarea**

Autentificarea ofera utilizatorilor posibilitatea de log-in, sing-up log-out. Aceasta componenta depinde de middleware-ul folosit pentru a verifica sau stoca credentialele user-ului. La input primeste credentialele iar la output mesajul corespunzator(eroare, succes sau token).

**Chat**

Componenta de chat ofera utilizatorilor posibilitatea de a comunica. Ca input primeste mesajele scrise de utilizatori pentru a fi salvate in baza de date prin comenzile oferite de JPA si ca output ofera conversatia preluata din baza de date dintre acestia.

**User**

Componenta de user ofera posibilitarea crearii unei identitati in cadrul aplicatiei pentru un user. Aceasta faciliteaza editarea profilului si insusirea de proiecte. Ca input primeste date despre utilizator si ca output ofera identitatea acestuia.

**Proiect**

Componenta de proiecte ofera posibilitatea de a actiona asupra proiectelor(edita, actualiza, lista, sterge, apartenenta). Ca input primeste date despre proiect, iar ca output ofera o detalii despre acesta.

## Front-end

**Autentificare**

Permite posibilitatea crearii unui cont sau accesarii aplicatiei pentru un user existent. Ca input primesteste credentialele, iar ca output transmite obiectul create din credentiale catre server.

**Chat**

Componenta de chat ofera utilizatorilor un view prin care acestia pot face conversatie sau pot urmari o conversatie mai veche. Ca input primeste mesaje iar ca output va transmite aceste mesaje catre server.

**User**

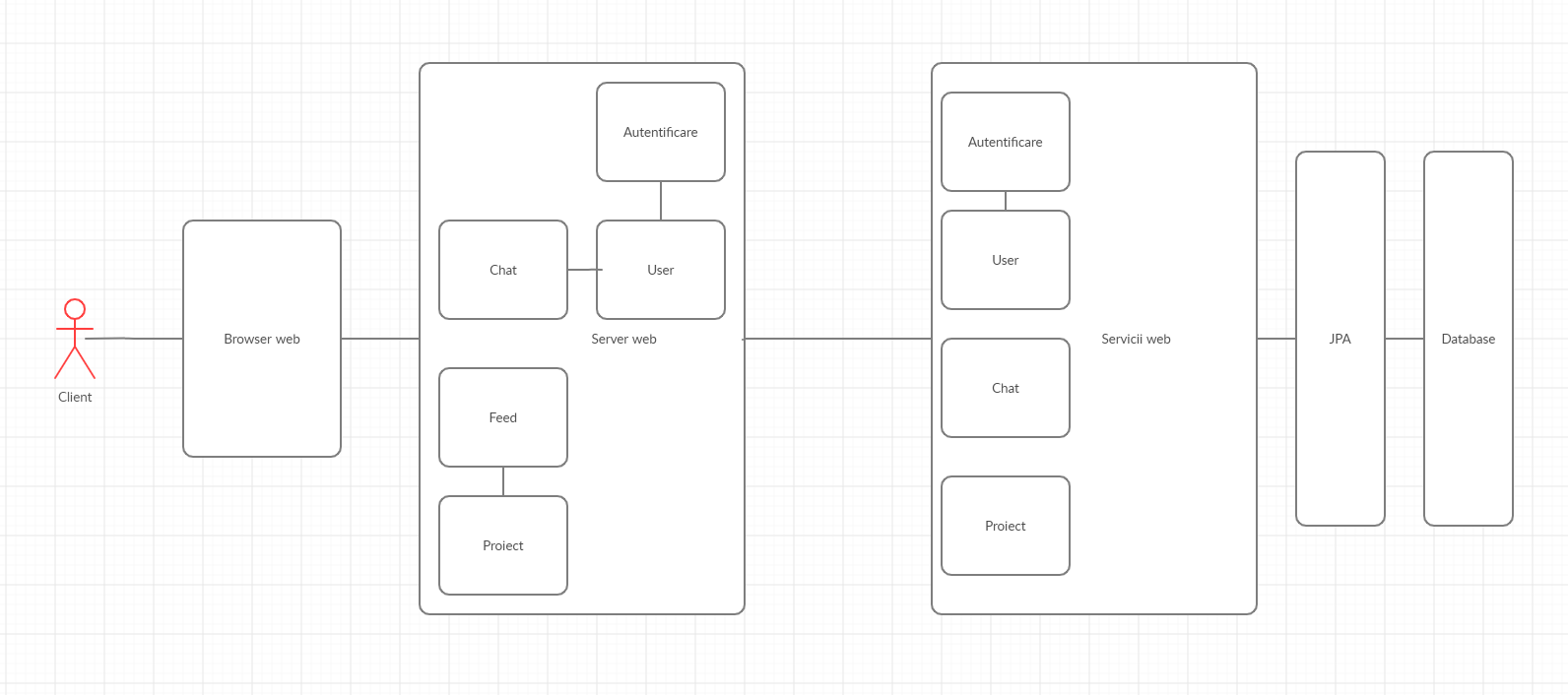
Componenta de user ofera posibilitarea vizualizarii si editarii profilului. Ca input primeste detalii despre utilizator, iar ca output transmite comenzi de editare, actualizare utilizator stergere catre server.

**Proiect**

Componenta de proiect ofera posibilitatea administrarii proiectelor. Ca input primeste o operatie asupra proiectului: actualizare, ediatre, stergere, apartenenta, iar ca output transmite aceste comenzi catre server cu obiectele necesare.

**Feed**

Componenta de feed ofera un view asupra proiectelor din aplicatie, dar si un view asupra proiectelor proprii. Din aceste view-uri utilizatorii pot actiona asupra proiectelor. Ca input primeste o lista de componenta proiect iar ca output genereaza o colectie de proiecte.



# Concluzii