****

**UNIVERSITATEA DE VEST DIN TIMIȘOARA**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**PROGRAMUL DE STUDII DE LICENȚĂ:**

**INFORMATICĂ**

**LUCRARE DE LICENȚĂ**

**COORDONATOR ȘTIINȚIFIC: ABSOLVENT:**

Asist. Dr. Gabriel Iuhasz Roșian Mihai

**TIMIȘOARA**

**2019**

**UNIVERSITATEA DE VEST DIN TIMIȘOARA**

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**PROGRAMUL DE STUDII DE LICENȚĂ:**

**INFORMATICĂ**

**Aplicație fitness EHealth**

**COORDONATOR ȘTIINȚIFIC: ABSOLVENT:**

Asist. Dr. Gabriel Iuhasz Roșian Mihai

**TIMIȘOARA**

**2019**

**Abstract**

Today most people who work in an office sit between 4-8 hours in a chair. Stand in the same position for more than 2 hours and repeatedly every day it make the risk of problems with the back, blood circulation, cardiovascular pressure and more problems to appear.

The EHealth application comes to the aid of people who neglect the need for daily physical activity, also the application comes to the aid of balanced diets. Working in an office the chances are very high that the company offers employees sweets, juices and other snacks for their employees, consumed repeatedly and in large quantities can lead to serious health problems such as: diabetes, obesity, loss of energy, and so on This application tries to merge both the need for physical movement of the human being and the need for healthy eating. This help comes from the fact that the application keeps track of your workouts and consumed foods and it generates a statistic to show you at what level of balance between movement, nutrition and state it is. The application also comes with a range of exercises divided by muscle groups to suggest exclusive training of the weakest part of the users body.

Keywords: Food, Exercises, Movement, Office Work

**Rezumat**

În zilele noastre majoritatea populației care lucrează într-un birou petrec între 4 și 8 ore pe scaun. A petrece timp într-o poziție timp de mai mult de 2 ore și în mod repetat în fiecare zi încep să apară probleme medicale la nivelul spatelui, circulației sângelui, presiunii cardio-vasculare, etc. De asemenea, lucrând într-un

Scopul aplicației EHealth este de a oferi un ajutor persoanelor care neglijează nevoia de activitate fizică zilnică, aceasta venind și cu un ghid pentru o dietă, respectiv alimentație sănătoasă și echilibrată.Lucrând într-un birou șansele sunt foarte mari ca firma respectivă să ofere angajaților dulciuri, sucuri și alte snacksuri pentru angajații săi. Acestea consumate fără măsură pot duce la probleme de sănătate grave precum ar fi: diabetul, obezitate morbidă, pierderea nivelului de energie, etc.

Aplicația EHealth îmbină cele două nevoi principale prin care omul va duce o viață sănătoasă: nevoia de mișcare fizică a omului, respectiv cea de alimentație sănătoasă.

Acest ajutor vine prin faptul că aplicația îți ține evidența antrenamentelor și mâncărurilor consumate și îți generează o statistică prin care să te avertizeze la ce nivel de echilibru între mișcare, alimentatie și stat se află acesta. De asemenea, aplicația vine și cu o gamă de exerciții împărțite pe grupuri de mușchi pentru a sugerat

antrenarea exclusivă a părții cea mai slabe a utilizatorului.

Cuvinte Cheie: Alimentație, Exerciții, Mișcare, Stat la birou

Cuprins

[**1. Introducere**](#_3l18frh) **[7](#_3l18frh)**

[1.1. Prezentarea generală a proiectului](#_3znysh7) [7](#_3znysh7)

[1.2. Obiective și țeluri](#_206ipza) [8](#_206ipza)

[**2. Modelul Afacerii**](#_2et92p0) **[8](#_2et92p0)**

[2.1. Studiul aplicațiilor din zona de sănătate și fitness](#_tyjcwt) [8](#_tyjcwt)

[2.2. Aplicații existente](#_3dy6vkm) [8](#_3dy6vkm)

[2.3. Aplicația propusă](#_2s8eyo1) [8](#_2s8eyo1)

[**3. Descrierea aplicației**](#_17dp8vu) **[9](#_17dp8vu)**

[3.1. Tehnologii folosite](#_3rdcrjn) [9](#_3rdcrjn)

[3.2. Partea de server](#_lnxbz9) [10](#_lnxbz9)

[3.3. Partea de client](#_4k668n3) [13](#_4k668n3)

[3.4. Caracteristici](#_4i7ojhp) [17](#_4i7ojhp)

[**4. Implementarea sistemului**](#_qsh70q) **[20](#_qsh70q)**

[4.1. Metodologia implementării](#_3as4poj) [20](#_3as4poj)

[4.2. Diagrama fluxului de date](#_2p2csry) [22](#_2p2csry)

[4.3. Diagrama relațiilor dintre entități](#_3o7alnk) [24](#_3o7alnk)

[4.3. Diagrame de secvență](#_ihv636) [25](#_ihv636)

[**5. Procesul de dezvoltare al sistemului**](#_vx1227) **[26](#_vx1227)**

[5.1. Proiectarea interfeței utilizator](#_1v1yuxt) [26](#_1v1yuxt)

[5.2. Proiectarea funcționalității](#_19c6y18) [28](#_19c6y18)

[**6. Concluzii**](#_3tbugp1) **[29](#_3tbugp1)**

[6.1 Îmbunătățiri și idei viitoare](#_28h4qwu) [29](#_28h4qwu)

[6.2 Concluzii](#_2lwamvv) [29](#_2lwamvv)

[**Bibliografie**](#_2zbgiuw) **30**

Listă de figuri

[Figura 1: Aplicatia MyFitnessPal 9](#_1t3h5sf)

[Figura 2: Aplicația EatThisMuch 9](#_4d34og8)

[Figura 3: Analiza diferențelor dintre locurile de munca bazate pe limbajul c# și cele bazate pe PHP 11](#_26in1rg)

[Figura 4: Legatura dintr obiectele din baza de date și un sistem de tip ORM 12](#_35nkun2)

[Figura 5: Legătura dintre Entity Framework și restul sistemului 13](#_1ksv4uv)

[Figura 6: Reprezentare vizuala a arhitecturii de tip MVC și componentele din care aceasta este compusă 14](#_44sinio)

[Figura 7: Structura celor de la Azure în cadrul aplicației Azure Storage 14](#_2jxsxqh)

[Figura 8: Model de utilizare a framework-ului knockout.js 16](#_z337ya)

[Figura 9: Exemplu de legatura dintre partea de HTML și viewmodelul declarat în exemplul anterior 16](#_3j2qqm3)

[Figura 10: Modulul implementat de mine peste functionalitatea de baza a JQuery Ajax 18](#_1y810tw)

[Figura 11: Prezentarea paginii principale din aplicația EHealth cu funcționalitățile descrise în acest subcapitol 21](#_2xcytpi)

[Figura 12: Prezentarea paginii de meniuri alimentare din cadrul aplicației EHealth 21](#_1ci93xb)

[Figura 13: Prezentarea paginii de alimente din cadrul aplicației EHealth 22](#_3whwml4)

[Figura 14: Prezentarea paginii de antrenamente din cadrul aplicației EHealth 23](#_2bn6wsx)

[Figura 15: Vizualizarea comparării dintre un sistem ce implementează Repository Pattern vs Unit of Work Pattern și un sistem ce folosește DbContext 24](#_1pxezwc)

[Figura 16: Integrarea Repository Pattern într-o aplicație cu conexiune la o baza de date 25](#_49x2ik5)

[Figura 17: Diagrama fluxului de date în cadrul aplicației EHealth 26](#_147n2zr)

[Figura 18: Diagrama relațiilor dintre entitățile prezente în cadrul aplicației EHealth 28](#_23ckvvd)

[Figura 19: Diagrama secvențială a sistemului de antrenamente 29](#_32hioqz)

[Figura 20: Diagrama secvențială a sistemului de generat meniuri 29](#_1hmsyys)

[Figura 21: Diagrama secvențială a procesului de autentificare 30](#_41mghml)

[Figura 22: Diagrama secvențială a sistemului de administrare 30](#_2grqrue)

[Figura 23: Vizualizare a unui moment din cadrul procesului de dezvoltarea al aplicației (campurile colorate cu galben, reprezinta task-uri ce trebuiesc implementate în cadrul aplicației EHealth) 31](#_3fwokq0)

[Figura 24: Captura de ecran a designului aplicației, realizat în aplicația Zeplin 32](#_4f1mdlm)

[Figura 25: Logo-ul aplicației EHealth 32](#_2u6wntf)

[Figura 26: Exemplu de posibila arhitectura a unei aplicații web folosind Docker 35](#_nmf14n)

[Figura 27: Posibila arhitectura a unei aplicații web ce foloseste ASP.NET si MVC, folosind Kubernetes. 36](#_37m2jsg)

[Figura 28: Arhitectura flux a celor de la facebook 37](#_1mrcu09)

[Figura 29: Diagrama arhitecturii Redux 38](#_46r0co2)

# 

# **1. Introducere**

Am ales această lucrare deoarece, mulți oameni din ziua de astăzi, mai ales cei dintr-un domeniu unde li se solicită statul mult la un birou pe scaun, încep să-și neglijeze sănătatea fizică și din această cauză apar boli ce puteau fii prevenite din timp. Din diverse motive mulți oameni aleg să nu facă deloc mișcare în timpul lor liber. Tema aleasă de mine cât și aplicația dezvoltată pe baza temei, caută să aducă un argument motivelor acestea din cauza cărora unele persoane refuză să facă mișcare.

Câteva din aceste motive sunt:

* Motive financiare. Nu toată lumea își permite în ziua de astăzi un instructor de fitness sau expertiză unui profesionist în domeniu, pentru a avea rezultatele dorite într-un timp relativ scurt.
* Motivul lipsei de timp. Nu au timp să se documenteze ce anume ar trebuii să facă, când, în ce ordine sau dacă exercițiile făcute de acestea sunt făcut în mod corect
* Motivul nesigurantei de sine. Unele persoane nu se simt simt bine din punct de vedere emoțional în momentul în care cineva se uită la ei în timp ce fac mișcare, acest lucru se întâmplă atât la sala cât și atunci când mișcare are loc într-un loc public. Etc.

Aplicatia Ehealth cauta sa rezolve aceste motive astfel incat, toata lumea, sa poată avea parte de beneficiile ce vin o data cu efortul fizic repetat și o alimentatie cat de cat echilibrata.

## **1.1. Prezentarea generală a proiectului**

Aplicația EHealth este o aplicație web de tip client-server monolita. Acest tip de aplicație aduce o modalitate mai bună de a partaja volumul de muncă. Clientul inițializează întotdeauna o conexiune cu server-ul în timp ce server-ul așteaptă întotdeauna solicitări de la unul sau mai mulți clienți.

Proiectul caută sa ajute utilizatorii cu 2 mari aspecte ale vieții lor, și anume: Activitatea fizica si Alimentatie în relație cu activitatea lor fizica.

Aplicația primește date legate de starea fizica a utilizatorului (greutate, inaltime, nivel de activitate saptamanala, etc.) cât și scopurile și dorințele acestuia ( greutatea dorită, scopul exercițiilor, etc.) pentru a monitoriza progresul acestuia și ai sugera utilizatorului un interval caloric în care acesta ar trebuii sa se situeze pentru as atinge țelurile.

Primind date legate de starea fizica actuala si cea dorita, sistemul sugereaza, un plan de alimentatie cat si un plan de activitate fizica, ambele în stransa legatura una cu cealalta, menite sa aducă utilizatorul în starea fizica dorită. În același timp aplicația permite modificarea de către utilizator a acestor sugestii atat de alimentatie cat si de activitate fizica cu variante existente în sistem sau cu variante adaugate in sistem de el însuși.

**1.2. Obiective și țeluri**

Obiectivele aplicației sunt expuse pe scurt mai jos:

* Obiectivul principal este de a proiecta si construii o aplicație pentru toate persoanele ce doresc sa facă mișcare dar din diferite motive nu o pot face, o aplicație cu un aspect prietenos cu utilizatorul
* Aplicația să poată fii folosită și în cadrul unei săli pentru a ajuta la organizarea sălii.
* O interfata grafica buna și ușor de înțeles

Aplicația EHealth a fost dezvoltat cu aceste obiective la baza, astfel la final, utilizatorul va avea partea de o experienta placuta și adevărata folosind aplicația pentru nevoile sale.

# **2. Modelul Afacerii**

## **2.1. Studiul aplicațiilor din zona de sănătate și fitness**

Pe piața se regasesc un număr mare de aplicații ce abordează aceeași tematică și încearcă să rezolve aceste probleme.

Dintre aceste aplicații majoritatea încearca sa separe alimentatia de activitatea fizica, astfel incat, majoritatea aplicațiilor de pe piața abordează fie activitatea fizica necesara unui individ, fie alimentatia. Acest lucru ajută într-o masura, dar nu vei avea un ajutor complet și un rezultat multumitor, deoarece activitatea fizica este strans legată de alimentatie.

## **2.2. Aplicații existente**

Cateva dintre aplicațiile care abordează aceeași tematică ca și tema aleasă de mine sunt următoarele:

* MyFitnessPal este o aplicație pentru smartphone și un site web care urmărește dieta și exercițiile fizice pentru a determina aportul caloric și nutrienții optime pentru obiectivele utilizatorilor și folosește elemente de gamificare pentru a motiva utilizatorii.
* Eat This Much este o aplicație web cat si o aplicație mobile ce abordează mai mult aspectul de alimentatie decât cel de activitate fizica si îți sugerează un meniu de alimentatie lunar în funcție de nevoile tale. Aceasta aboredeaza oarecum si activitatea ta fizica, tinand cont de nivelul de activitate atunci cand iti genereaza meniul lunar cu retete gastronomice.

MyFitnessPal oferă suport atat pe partea de activitate fizica cat si pe partea de alimentatie asemanator aplicației EHealth. Consider ca aplicația EHealth se diferențiază de aceasta prin interfața grafica mai prietenoasă cu utilizatorul, cei de la MyFitnessPal îți oferă multă informație incarcata în același loc cu foarte puțină vizualizare grafica a acestor date.

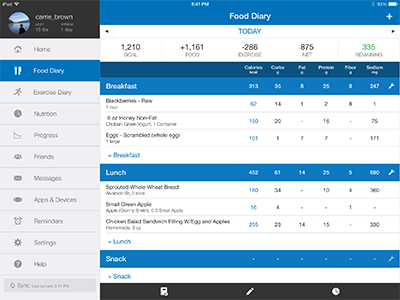


Figura 1: Aplicatia MyFitnessPal

Aplicația EatThisMuch ofera vizualizare grafica foarte multa si foarte intuitiva, cat si foarte multe opțiuni de customizare a meniurilor alimentare și a ingredientelor din care aceste sunt compuse, dar aceasta aplicatie îți oferă suport doar pe partea de alimentatie, nu și pe partea de activitate fizica, ceea ce înseamnă ca nu îți oferă un sistem complet fitness.

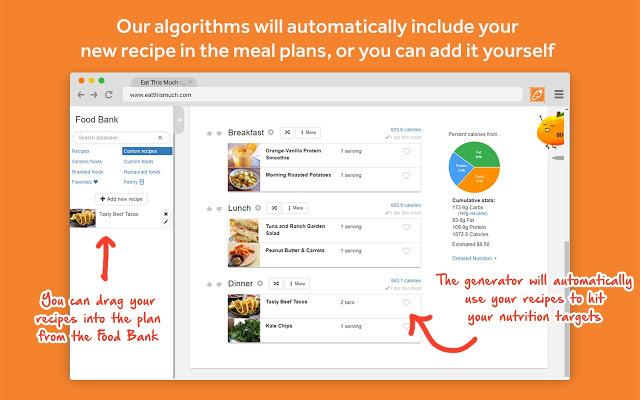


Figura 2: Aplicația EatThisMuch

## **2.3. Aplicația propusă**

Aplicatia propusa de tema lucrarii va fii proiectata în vederea aducerii părților cele mai bune din aplicațiile menționate la punctul anterior. Aplicația EHealth v-a îmbina atat elemente de gamificare, cât și ambele elemente ale unei aplicații de fitness complete, activitatea fizica si alimentatia echilibrata.

Partea de alimentatie va contine generarea automată de meniuri alimentare cat si posibilitatea de creare a acestora, calculul consumului de calorii, vizualizare grafica a tuturor datelor și progresul acestora.

Partea de activitate fizica va contine generarea automată de antrenamente cat si posibilitatea de creare a acestora, calcul de timp petrecut în cadrul acestor activități, exerciții categorisite pe nivele, de la exercitii complicate la unele ușoare, astfel atat un amator de mișcare fizica cat si un profesionist se vor simți în dificultate în urma practicării acestor exerciții.

# **3. Descrierea aplicației**

## **3.1. Tehnologii folosite**

Pentru realizarea lucrării am folosit ca limbaj de programare pentru partea de server C# și framework-ul celor de la Microsoft numit .NET FRAMEWORK. Alegerea acestor tehnologii s-a făcut în comparație cu un alt limbaj de programare popular și folosit în cadrul dezvoltării de aplicații web, numit PHP. După o perioada de studii am concluzionat faptul ca C# este o alegere mai buna fata de PHP, în continuare voi prezenta cateva detalii si diferente cheie ce m-au încurajat sa fac aceasta decizie.

Php-ul reprezinta un limbaj de scriptare folosit pe partea de server ce a fost proiectat pentru realizarea paginilor web.

C# reprezinta un limbaj de programare orientat obiect dezvoltat de către cei de la Microsoft, proiectat cu scopul de a combina puterea de calcul adusă de C ++ cu ușurința de programare adusă de Visual Basic. Standardul Ecma listează aceste obiective de proiectare pentru C #:

* Limbajul se dorește a fi un limbaj de programare simplu, modern, cu scop general, orientat pe obiecte.
* Limbajul și implementarea acestora ar trebui să ofere suport pentru principiile de inginerie software, cum ar fi verificarea puternică a tipului, verificarea limitelor, detectarea încercărilor de utilizare a variabilelor neinitializate și colectarea automată a gunoiului. Robustă, durabilitatea și productivitatea programatorului sunt importante.
* Limba este destinată utilizării în dezvoltarea de componente software adecvate pentru implementare în medii distribuite.
* Portabilitatea este foarte importantă pentru codul sursă și programatorii, în special cei deja familiarizați cu C și C ++.
* Sprijinul pentru internaționalizare este foarte important.
* C # este destinat să fie potrivit pentru scrierea aplicațiilor atât pentru sisteme găzduite, cât și pentru sisteme încorporate, de la cele mari care folosesc sisteme de operare sofisticate, până la cele foarte mici având funcții dedicate.
* Deși aplicațiile C # sunt destinate să fie economice în ceea ce privește cerințele de memorie și puterea de procesare, limbajul nu a fost destinat să concureze direct la performanță și dimensiune cu C sau limbaj de asamblare.

După câte se poate observa din definițiile celor 2 limbaje C# este un limbaj orientat obiect pe cand PHP-ul nu este ( se pot folosii practici de POO folosind PHP, cum se pot folosii în orice limbaj dar acesta nu a fost gandit pentru POO).

Deși PHP la baza este mult mai performant decat C#, îmbunătățirile aduse de C# pentru a ușura munca programatoriilor cât și faptul ca PHP-ul nu permite programarea pe mai multe fire de execuție, pe când C# nu numai ca permite acest lucru dar este și ușor de implementat aceste lucruri, m-au făcut sa aleg C# și arhitectura .NET pentru realizarea lucrării.

Pe partea de client am ales sa folosesc pe langa limbajele de baza HTML, CSS, Javascript, să folosești o arhitectura de javascript numita Knockout.js. Am ales aceasta arhitectura deoarece, asemanator părții de server am avut de ales din mai multe opțiuni, dintre aceste cateva optiuni au fost: Vue.js, React.js, Angular 8. În cele din urma alegerea am făcut-o din punct de vedere al dimensiunii și complexității proiectului și faptul ca arhitectura Knockout.js iti ofera minimul necesar pentru realizarea unei astfel de aplicații web, cat si faptul ca Knockout folosește arhitectura MVVM prezenta și în aplicațiile desktop de tip WPF.



Figura 3: Analiza diferențelor dintre locurile de munca bazate pe limbajul c# și cele bazate pe PHP

## **3.2. Partea de server**

După cum am menționat și la punctul anterior partea de server a fost realizata folosind limbajul dezvoltat de cei de la Microsoft numit C# și folosind arhitectura .NET. Pentru a face legatura la baza de date într-un mod cat mai eficient am decis sa folosesc Entity Framework.

Entity Framework este realizata folosind tehnica ORM (Object Relational Mapping). În domeniul informaticii, ORM-ul este o tehnică de programare pentru conversia datelor între sisteme de tip incompatibile folosind limbaje de programare orientate pe obiecte. Aceasta creează, de fapt, o „bază de obiecte virtuale” care poate fi folosită din limbajul de programare. Există atât pachete gratuite, cât și comerciale, care realizeaza maparea relațională cu obiecte, deși unii programatori optează pentru a construi propriile lor instrumente ORM.

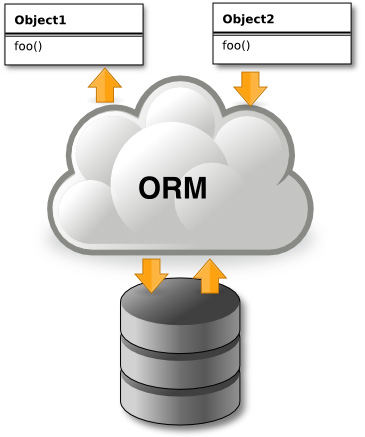


Figura 4: Legatura dintr obiectele din baza de date și un sistem de tip ORM

În cazul acestei lucrări, entity framework este folosită pentru transpunerea entităților din baza de date, în clasele din cadrul aplicației. Aceasta abordare de a transpune entități din baza de date în clasele din cadrul aplicației, poarta denumirea de ,,Database First”. Denumirea provine din faptul ca prima oara este creata arhitectura bazei de date si mai apoi aceasta este transpusa în mod automat în clase.

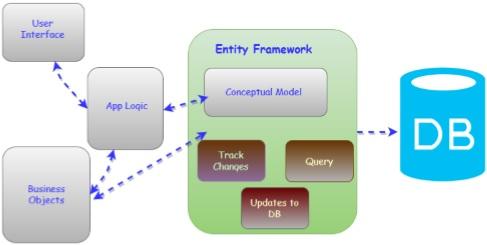


Figura 5: Legătura dintre Entity Framework și restul sistemului

Pentru baza de date am folosit Sql Server, acesta folosește ca limbaj de interogare a bazelor de date Transact SQL sau prescurtat T-SQL.

Pentru a putea accesa aplicația printr-un browser folosind un url specific(ehealth\), a fost nevoie de urcarea aplicație pe un server de tip web. Pentru acest lucru s-a folosit IIS-ul (Internet Information Service). IIS este un web server dezvoltat de cei de la microsoft și poate fii folosit de către toate aplicațiile compatibile cu sistemele de operare din familia Windows NT. Acest server suporta cereri de tipul: HTTP, HTTP/2, HTTPS, FTP, FTPS, SMTP si NNTP.

Aplicația EHealth a fost proiectată folosind arhitectura MVC (Model View Controller). Aceasta arhitectura de aplicație separa aplicația în trei mari entități, și anume:

* Model: În cadrul acestei entități se realizeaza forma datelor ce sunt procesate de aplicație și trimise mai apoi spre a fii stocate în baza de date cat si functionalitatea legata de afacerea pentru care este dezvoltata aplicatia.
* View: Aceasta entitate reprezinta interfata utilizator care, in cadrul MVC, aceasta prezinta datele primite de la model și permite utilizatorului sa vizualizeze aceste date cat si posibilitatea sa le modifice.
* Controller: Acesta se ocupă cu cerințele provenite de la utilizator. În mod normal utilizatorul interacționează cu interfața grafica, modificand date acesta genereaza o solicitare ce v-a fii gestionată de către controller, la randul lui, controller-ul, v-a răspunde cu un nou view format pe baza noilor date din model.

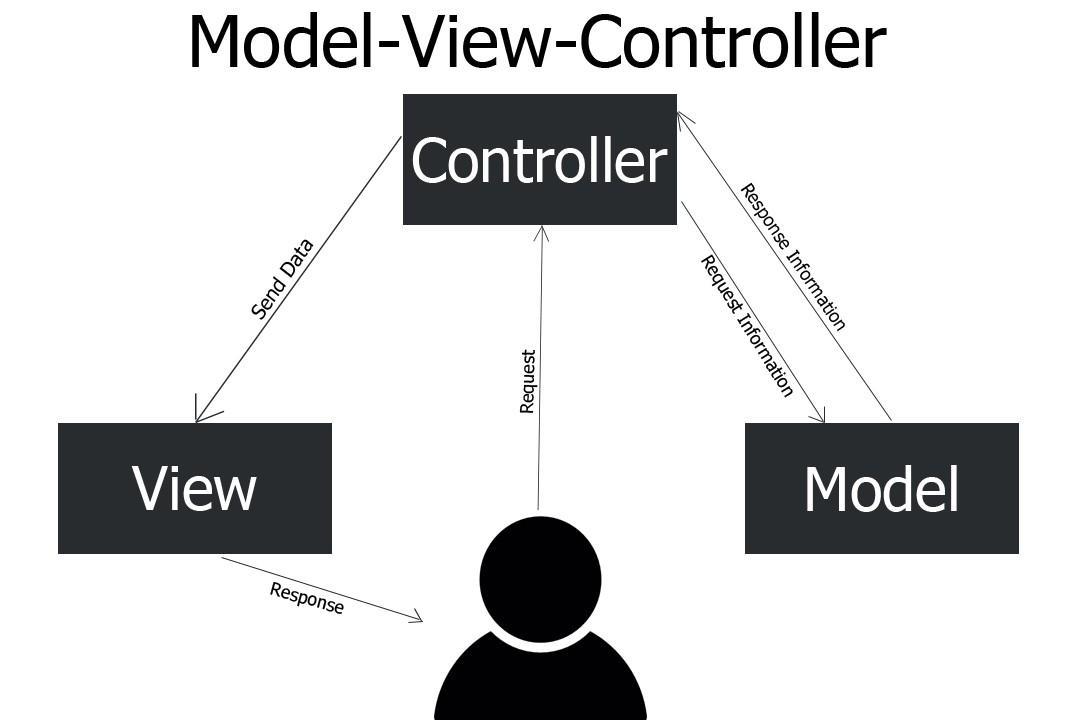


Figura 6: Reprezentare vizuala a arhitecturii de tip MVC și componentele din care aceasta este compusă

Salvarea pozelor și video-urilor prezente în cadrul aplicației se realizeaza folosind o aplicație din cadrul unui serviciu dezvoltat de cei de la microsoft pentru developeri numit Azure DevOps, iar serviciul specific se numește Azure Blob Storage. Azure DevOps este un serviciu oferit de către cei de la microsoft. ce conține multe unelte care ajuta la dezvoltarea de software. Printre aceste unele de se regasesc Azure Blob Storage. Azure Blob Storage este un serviciu în Microsoft Azure, care permite dezvoltatorilor să stocheze date nestructurate pe platforma de cloud Microsoft. Aceste date pot fi accesate de oriunde în lume și pot include audio, video și text. Globurile sunt grupate în „containere” care sunt legate de conturile utilizatorilor.

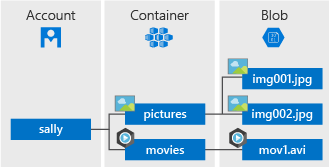


Figura 7: Structura celor de la Azure în cadrul aplicației Azure Storage

**Cont stocare**

Un cont de stocare oferă un spațiu de nume unic în Azure pentru datele dvs. Fiecare obiect pe care îl stocați în Azure Storage are o adresă care include numele dvs. de cont unic. Combinația dintre numele contului și punctul final Blob de stocare Azure formează adresa de bază pentru obiectele din contul de stocare.

**Container**

Un container organizează un set de blocuri, similar cu un director dintr-un sistem de fișiere. Un cont de stocare poate include un număr nelimitat de containere, iar un container poate stoca un număr nelimitat de blocuri.

**Blob**

Azure Storage acceptă trei tipuri de blobs:

Blobs-urile stochează text și date binare, până la aproximativ 4,7 TB. Blocurile bloc sunt alcătuite din blocuri de date care pot fi gestionate individual.

Blocurile de completare sunt alcătuite din blocuri precum blocurile de bloc, dar sunt optimizate pentru operațiunile de anexare. Blocurile de completare sunt ideale pentru scenarii cum ar fi înregistrarea datelor de la mașinile virtuale.

Globurile de pagină stochează fișierele de acces aleator cu dimensiunea de până la 8 TB. Globurile de pagină stochează fișierele hard disk-ului virtual (VHD) și servesc ca discuri pentru mașinile virtuale Azure. Pentru mai multe informații despre blocurile de pagini, consultați Prezentarea generală a blocurilor de pagini Azure

**3.3. Partea de client**

Partea de client a fost realizata utilizand o combinatie de mai multe limbaje și arhitecturi, și anume:

Pentru partea de functionalitate am folosit framework-ul de javascript numit Knockout combinat cu anume elemente din Jquery. Knockout este o implementare JavaScript de sine stătătoare a modelului Model-View-ViewModel cu șabloane, model folosit în mare parte în dezvoltarea de aplicații desktop folosind C# și XAML. Datorită faptului ca knockout implementează MVVM, permite tranziția mai ușoară a unui programator de aplicații desktop la unu de web și invers. Principiile de bază sunt:

o separare clară între datele de domeniu, componentele de vizualizare și datele care trebuie afișate

prezența unui strat de cod specializat clar definit pentru a gestiona relațiile dintre componentele de vizualizare

Acesta din urmă folosește caracteristicile native de gestionare a evenimentelor din limba JavaScript.

Aceste caracteristici simplifică specificarea relațiilor complexe între componentele de vizualizare, care la rândul lor fac ecranul mai receptiv și experiența utilizatorului mai bogată.

Knockout a fost dezvoltat și este menținut ca un proiect open source de Steve Sanderson.

Exemplu de utilizarea a Knockout.js:

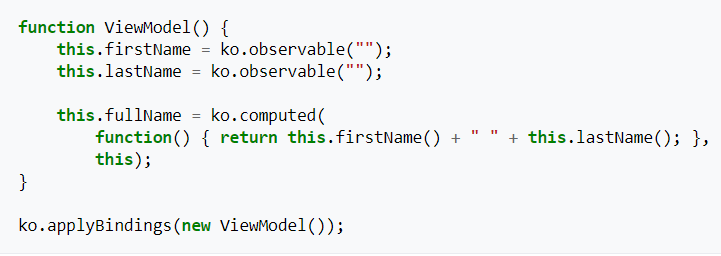


Figura 8: Model de utilizare a framework-ului knockout.js

În acest exemplu, două căsuțe de text sunt legate de variabile observabile pe un model de date. Afișajul „nume complet” este legat de un observabil dependent, a cărui valoare este calculată în termeni de observabili. Când este modificată oricare dintre casetele de text, afișajul „nume complet” este actualizat automat, fără o gestionare explicită a evenimentelor.

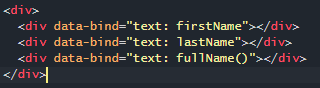


Figura 9: Exemplu de legatura dintre partea de HTML și viewmodelul declarat în exemplul anterior

Pentru selecția elementelor din DOM (Document Object Model) cât și pentru a utiliza anumite functionalitati din aceasta librărie, am folosit JQuery. jQuery este o librarie javascript concepută pentru a simplifica traversarea și manipularea arborelui HTML DOM, precum și gestionarea evenimentelor, animațiilor CSS și cererilor de tip Ajax. Este un software gratuit, open-source, care utilizează permisul MIT permisiv. Din mai 2019, jQuery este utilizat de 73% din cele 10 milioane de site-uri web cele mai populare. Analiza web indică faptul că este cea mai des extinsă bibliotecă JavaScript cu o marjă mare, având de 3 până la 4 ori mai multă utilizare decât orice altă bibliotecă JavaScript.

Realizarea elementelor vizuale din cadrul aplicației și aspectul acestora a fost realizat folosind framework-ul de CSS numit Semantic UI versiunea 2.4. Potrivit site-ului web a celor de la Semantic, obiectivul framework-ului este de a împuternici designerii și dezvoltatorii “prin crearea unui limbaj pentru partajarea interfeței de utilizator”. Aceștia fac acest lucru folosind un limbaj semantic, descriptiv pentru clasele sale și convențiile de denumire. În loc să folosească prescurtări, așa cum fac alte cadre, acesta folosește cuvinte reale într-o manieră mai apropiată de engleza simplă. Cateva din caracteristicile framework-ului sunt:

* Un element UI este un bloc de bază de bază. Poate apărea singur sau în grupuri uniforme. De exemplu, un buton poate fi independent sau introdus într-un grup de butoane.
* O colecție ÎI este un grup de diferite tipuri de elemente care sunt interdependente. De exemplu, un formular web poate avea butoane, intrări, casete de selectare, pictograme și așa mai departe.
* O vizualizare UI reprezintă o bucată comună a conținutului site-ului web. De exemplu, o secțiune de feed-uri sau comentarii.
* Un modul UI este o componentă cu funcționalitate interactivă bazată pe JavaScript. Exemple includ un acordeon, un dimmer, modal, etc.
* Comportamentul UI este o componentă care nu poate exista independent, dar este utilizată în schimb pentru a injecta funcționalități în alte componente. De exemplu, comportamentul de validare a formularului oferă funcționalitate de validare pentru componenta Form.

Pentru a face apeluri către server am folosit tehnologia AJAX, ( prescurtarea pentru JavaScript asincron și XML ) este un set de tehnici de dezvoltare web care utilizează multe tehnologii web din partea clientului pentru a crea aplicații web asincrone. Cu Ajax, aplicațiile web pot trimite și recupera date de pe un server asincron (în fundal) fără a interfera cu afișarea și comportamentul paginii existente. Prin decuplarea stratului de schimb de date de stratul de prezentare, Ajax permite paginilor web și, prin extensie, aplicațiilor web, să modifice conținut dinamic, fără a fi nevoie să reîncărcați întreaga pagină. În practică, implementările moderne folosesc în mod obișnuit JSON în loc de XML.

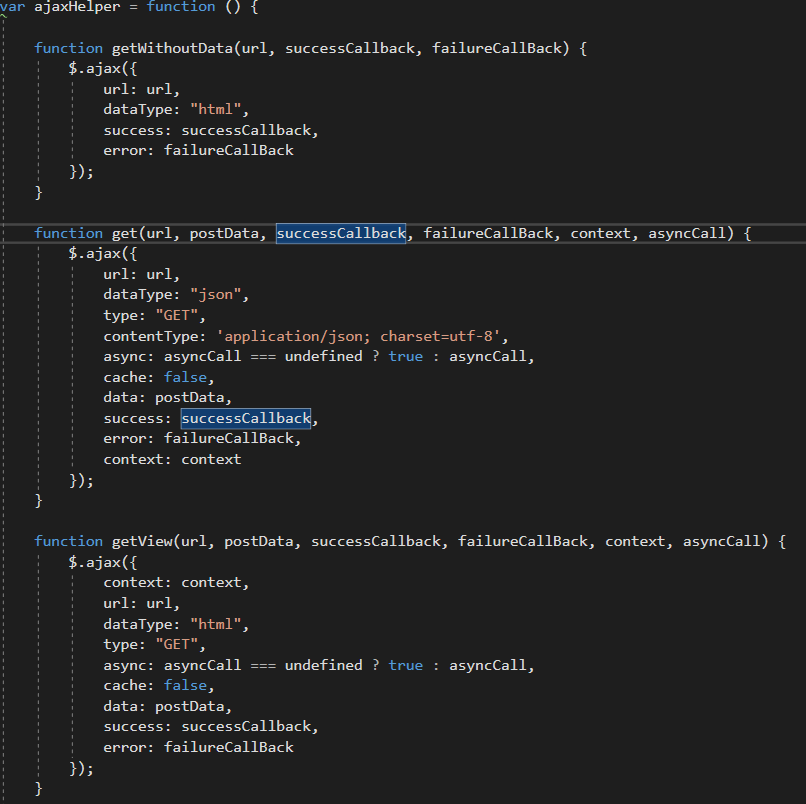


Figura 10: Modulul implementat de mine peste functionalitatea de baza a JQuery Ajax

Ajax nu este o singură tehnologie, ci mai degrabă un grup de tehnologii. HTML și CSS pot fi utilizate în combinație pentru marcarea și stilul informațiilor. Pagina web poate fi apoi modificată folosind JavaScript pentru a afișa dinamic - și permite utilizatorului să interacționeze cu - noile informații. Obiectul XMLHttpRequest încorporat sau din 2017 nouă funcție "fetch ()" din JavaScript este folosită în mod obișnuit pentru a executa Ajax pe paginile web, permițând site-urilor să încarce conținut pe ecran fără a reîncărca pagina.

Ajax nu este o tehnologie nouă sau un limbaj diferit, ci doar tehnologiile existente utilizate în moduri noi. Peste AJAX-ul din Jquery am intervenit cu o modalitate de a ușura și mai mult modul de scriere al sintaxei, și anume am definit metode ce apelează metode din cadrul Jquery AJAX, aceste metode se afla în modulul de javascript numit AjaxHelper.

Pentru partea vizuala am folosit HTML5 si CSS, folosind media query și calcul de procente în funcție de dimensiunea paginii, pentru a proiecta o interfata vizuala scalabila in functie de dimensiunea ecranului de pe care aceasta este vizualizata.

HTML5 este cea mai recentă evoluție a standardului care definește HTML. Termenul reprezintă două concepte diferite. Este o nouă versiune a limbajului HTML, cu noi elemente, atribute și comportamente și un set mai mare de tehnologii care permite construirea de site-uri și aplicații Web mai diverse și mai puternice.

Foile de stil în cascadă (CSS) este un limbaj al foilor de stil folosit pentru a descrie prezentarea unui document scris în HTML sau XML (incluzând dialecte XML precum SVG, MathML sau XHTML). CSS descrie modul în care elementele trebuie redate pe ecran, pe hârtie, în vorbire sau pe alte suporturi.

CSS este unul dintre limbajele de bază ale Web-ului deschis și este standardizat în browserele Web conform specificației W3C. Dezvoltat pe niveluri, CSS1 este acum învechit, CSS2.1 este o recomandare, iar CSS3, acum împărțit în module mai mici, avansează pe pista de standardizare.

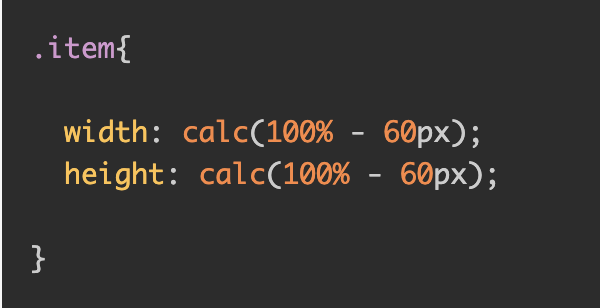
Funcția “calc()” din CSS

Web-ul modern este o experiență flexibilă cu atât de multe dimensiuni de ecran și dispozitive pentru a fi acceptate. Aceasta înseamnă că sunt necesare o proiectare fluidă și sensibilă. CSS calc () face mult mai ușor să construiți machete sensibile fluide. Acest lucru poate duce la o situație lipicioasă dacă doriți ca un element să redimensioneze în mod natural, pe măsură ce elementul părinte sau dimensiunile ferestrei se modifică. În trecut, acest lucru a fost destul de greu de implementat și, în general, ar necesita un fel de JavaScript pentru a asculta evenimentul de redimensionare al browserului. Astăzi browserele moderne acceptă funcția “calc()” din CSS.

Funcția “calc()” primește ca parametru o ecuație matematică, iar aceasta va calcula automat o valoare a proprietății actualizate pe măsură ce dimensiunea ecranului se modifică.

Funcția “calc()” permite ca expresii matematice cu adunare (+), scădere (-), înmulțire (\*) și diviziune (/) să fie utilizate ca valori componente.

De asemenea, puteți amesteca unități, cum ar fi pixeli cu procente, ems, rems, vw, vh etc.



*Figura 9: Exemplu de utilizare a funcției “calc” din CSS*

În momentul scrierii acestei documentații arhitectura părții vizuale se scaleaza într-un mod rezonabil pană la o dimensiune a ecranului de 1300px x 800px. Pe viitor as dorii sa pun mai mult accent pe partea de “responsiveness” pentru a permite aplicației să fie folosită și pe dispozitive cu ecrane mai mici de rezolutia actuala.

## **3.4. Caracteristici**

Pentru dezvoltarea aplicației EHealth, în urma studiului asupra aplicațiilor existente din zona fitness-ului, am alcatuit o lista de caracteristici ce îmbina părțile cele mai bune găsite în aplicațiile existente, cat si contribuții proprii în vederea eficientizarii procesului de activitate fizica, și anume:

* Se monitorizează procesul utilizatorului, către obiectivul ales, din punct de vedere al: caloriilor arse și consumate, nivelului de activitate pe diferite perioade de timp, obiectivul setat de utilizator pe o perioada respectivă, etc. Prezentandu-i acestuia diagrame și grafice vizuale.
* Oferă posibilitatea utilizatorului să își creeze meniuri bazate pe alimente, cat si instrucțiuni din urma cărora utilizatorul să-și poată prepara singur acel meniu. Deasemenea pentru convenienta utilizatorului, aplicația are deja incarcata un număr de meniuri cu instrucțiuni.

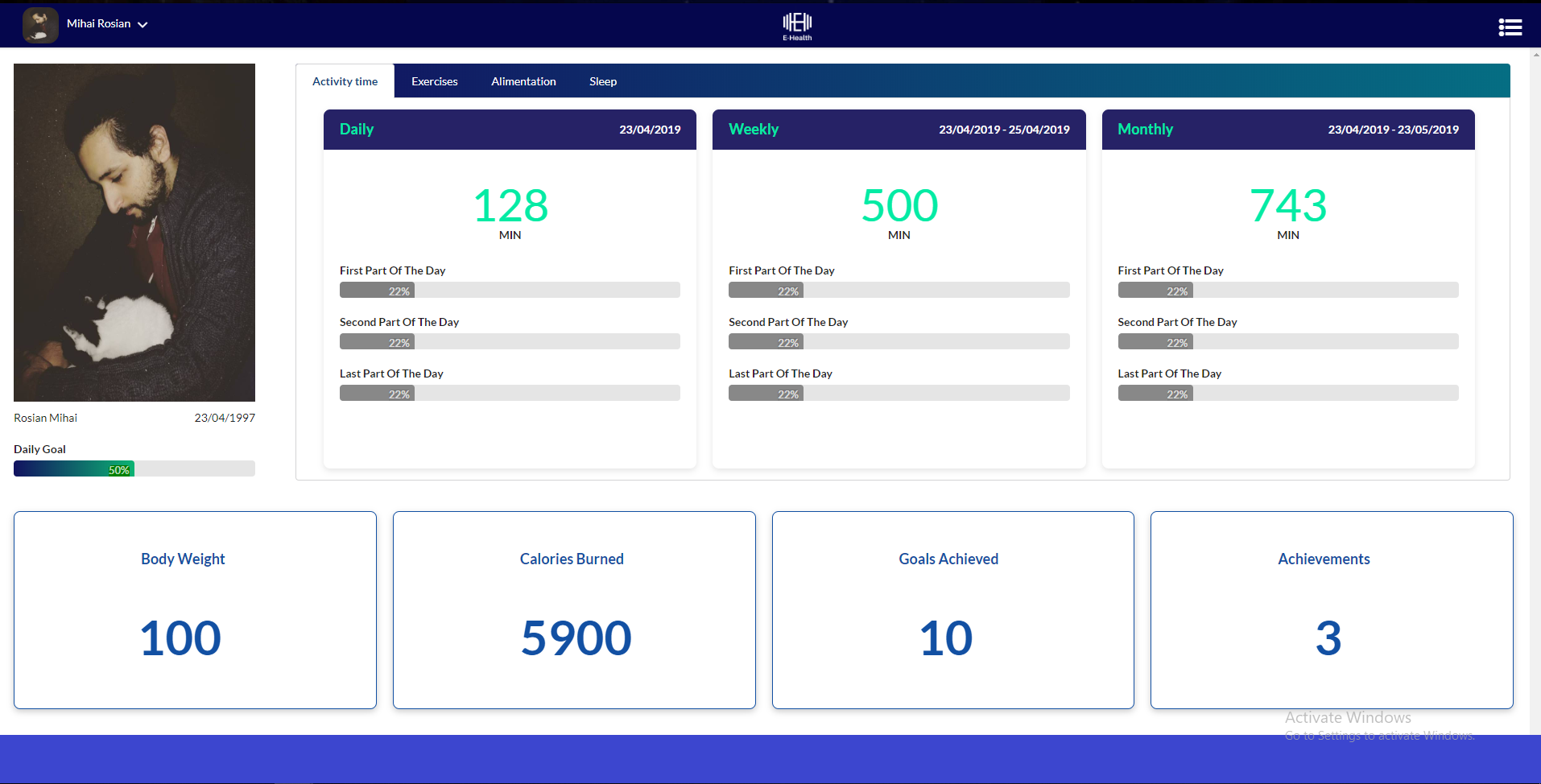


Figura 11: Prezentarea paginii principale din aplicația EHealth cu funcționalitățile descrise în acest subcapitol

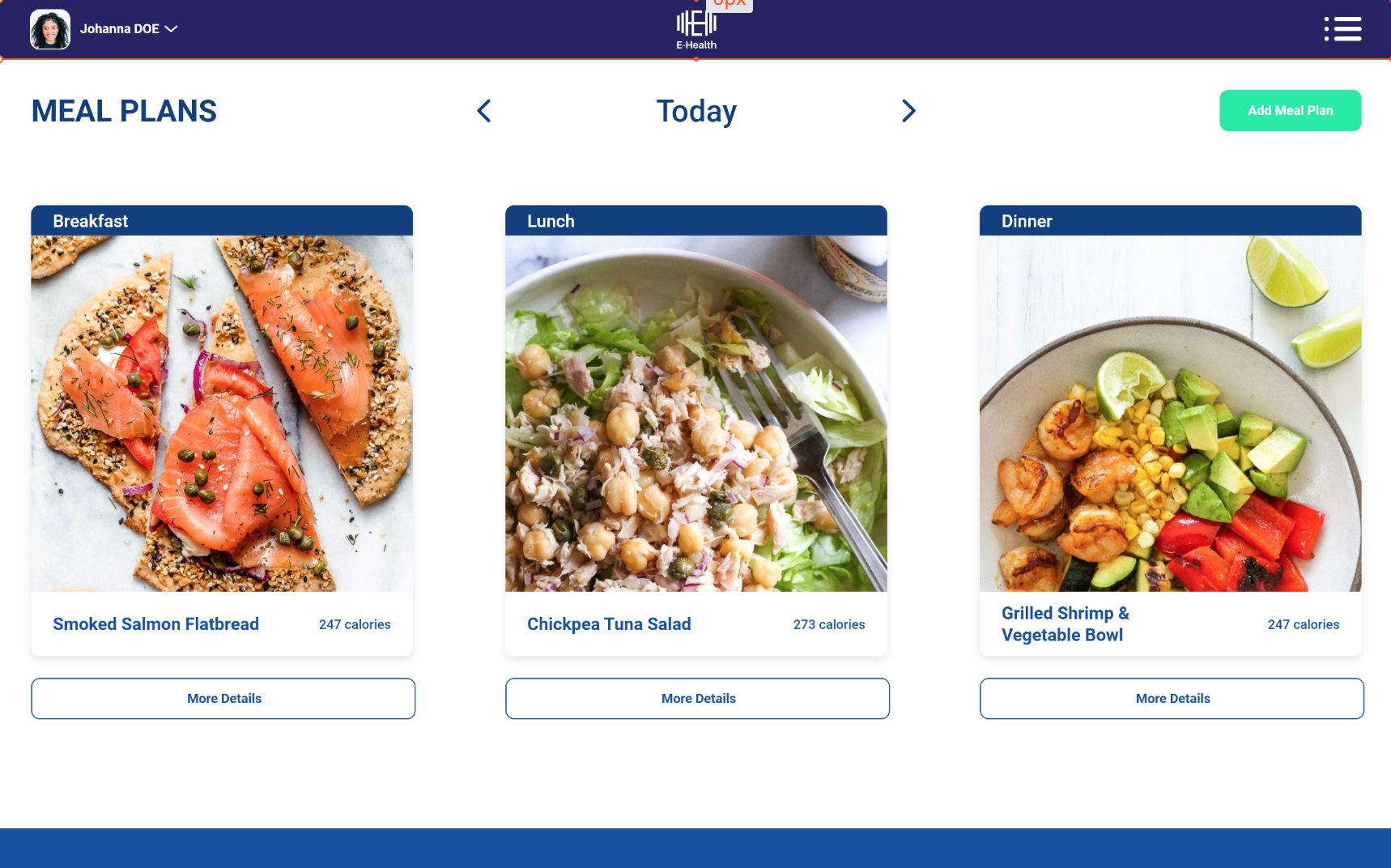


Figura 12: Prezentarea paginii de meniuri alimentare din cadrul aplicației EHealth

* Aplicația oferă utilizatorilor posibilitatea de a adăuga alimente în aplicație, alimente care vor fii folosite în crearea meniurilor.
* Pentru a fii o aplicație de fitness completa, EHealth oferă utilizatorilor sprijin și în partea de activitate fizica, permitand utilizatorului sa adauge tipuri de exerciții în cadrul aplicației, cat și să folosească aceste exerciții pentru a creea antrenamente. Bineinteles un număr de antrenamente și exerciții se vor regasii în cadrul aplicației fără ca utilizatorul sa adauge neapărat unele noi.

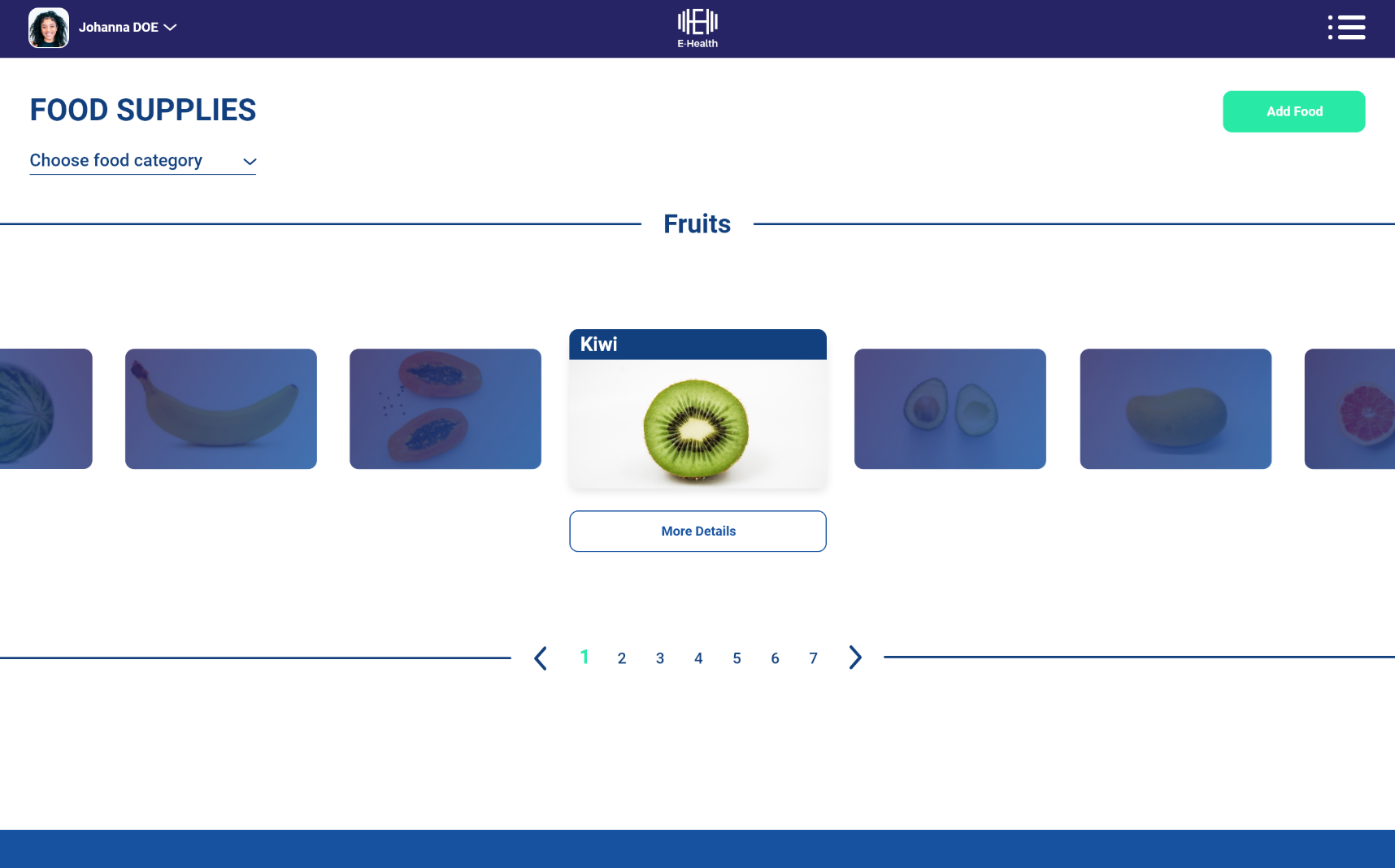


Figura 13: Prezentarea paginii de alimente din cadrul aplicației EHealth

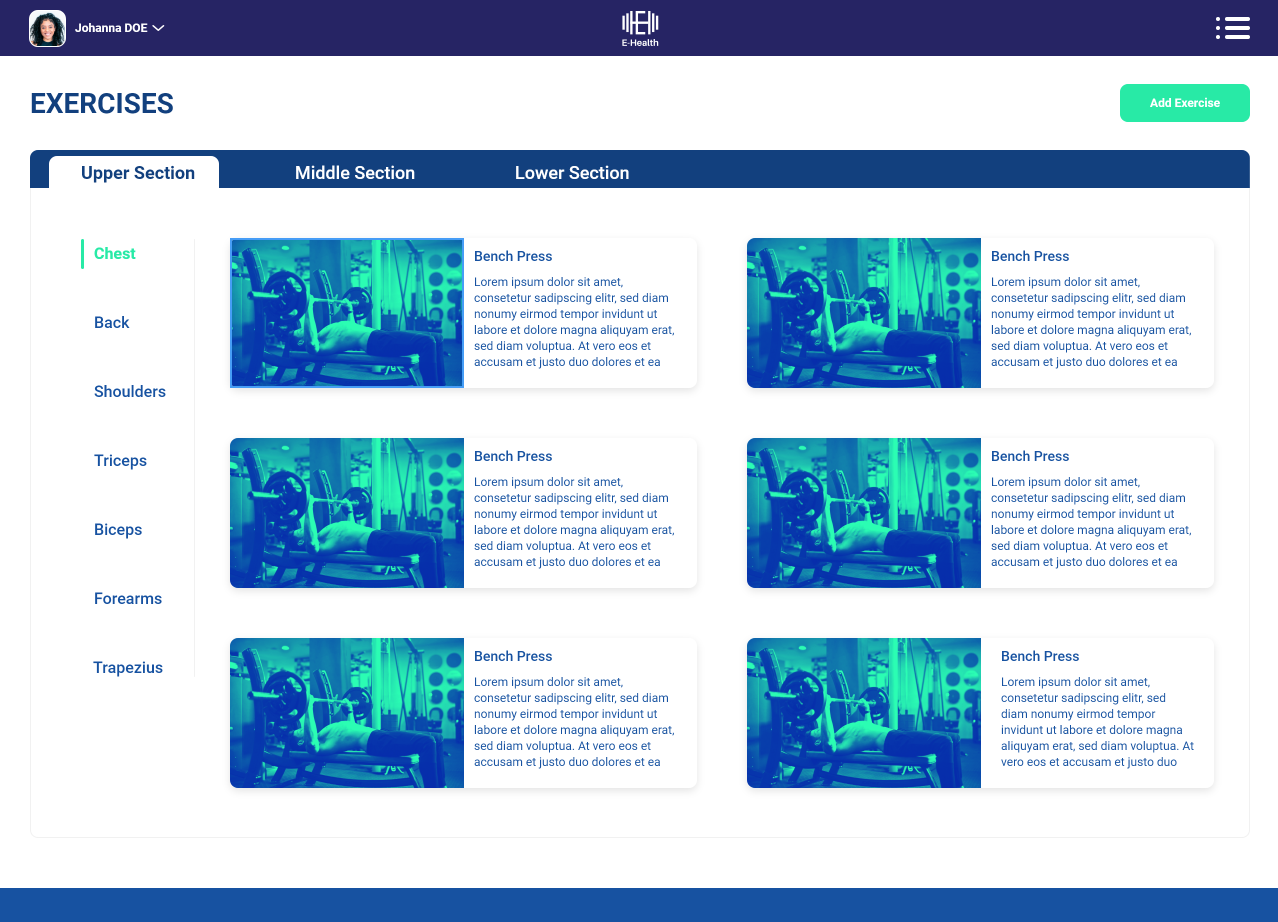


Figura 14: Prezentarea paginii de antrenamente din cadrul aplicației EHealth

* Pentru a tine userii într-o continua întrecere cu ei însăși cât și cu apropiații acestora, aplicația oferă un sistem de tip facebook, ce contine, posibilitatea de a adăuga alți utilizatori dintr-o lista de prieteni și de a comunica cu aceștia prin intermediul unui chat.

Pe langa aceste caracteristici alea aplicației, văzute de catre utilizatorii obisnuiti, aplicatia mai prezinta si un portal pentru admini, acesta a fost creat pentru a asigura o buna utilizare a aplicației de către utilizatori, cat si pentru a putea impune un sistem de reguli, astfel adminul, în urma incalcarii regulilor poate acționa asupra utilizatorilor necorespunzători. Utilizatorul acestui portal are posibilitatea de a creea, șterge, restricționa accesul la anumite caracteristici ale aplicației, unui user, cât și caracteristicele unui user normal.

# **4. Implementarea sistemului**

## **4.1. Metodologia implementării**

Pentru implementarea cat mai eficienta și pentru a permite o mai mare reutilizare cat și extindere a aplicației, ca și arhitectura aplicația, a fost împărțită pe mai multe nivele.

BusinessLogic: La acest nivel se regasesc functionalitati de baza a fiecărei entități prezente în aplicație, în general, logica ce ține de domeniul pentru care a fost creeata aplicația se regaseste aici. Fiecărei entități prezente în baza de date i se regaseste cate un “Core” la acest nivel. În cadrul acestor “Cores” se realizeaza logica specifica entitatii de care aparține. Toate aceste clase numite “Core” mostenesc din clasa de baza “BaseCore”, în aceasta se realizeaza lucrurile cele mai de baza ce se pot realiza în cadrul unei entități, și anume, comunicarea cu baza de date. Pentru a eficientiza comunicare cu baza de date și a lăsa loc de extindere, am implementat conceptul de Repository și Unit of Work Pattern.

Repository Urile și Unit of Work sunt destinate să creeze un strat de abstractizare între stratul de acces la date și stratul de logică de afaceri al unei aplicații. Implementarea acestor tipare ajuta la izolarea aplicației de modificările din depozitul de date și poate facilita testarea automată a unităților sau dezvoltarea bazată pe test (TDD - Test Driven Development).

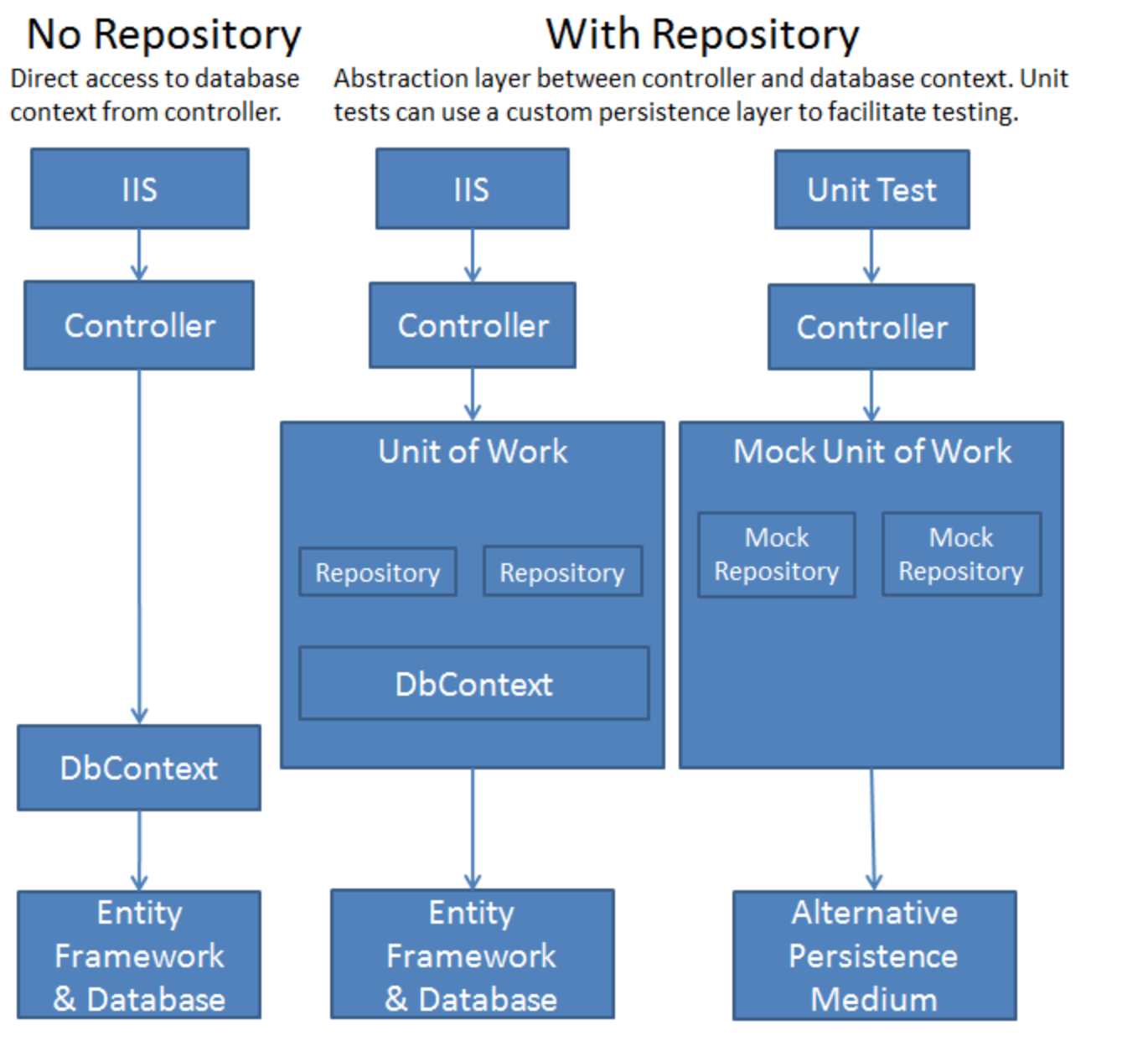


Figura 15: Vizualizarea comparării dintre un sistem ce implementează Repository Pattern vs Unit of Work Pattern și un sistem ce folosește DbContext

Repository pattern, este unul dintre cele mai populare modele pentru a crea o aplicație la nivel de firma. Implementarea acestui model nu restricționează să lucrăm direct cu datele din aplicație și creează noi straturi pentru operațiunile bazei de date, logica de afaceri și interfața de utilizare a aplicației. Dacă o aplicație nu respectă Repository Pattern, aceasta poate avea următoarele probleme:

* Duplică codurile de operații ale bazei de date
* Necesitatea interfeței de utilizare pentru unitatea operațiunilor bazei de date de testare și logica de afaceri
* Necesitatea dependențelor externe pentru unitatea de testare a logicii de afaceri
* Dificil de implementat caching a bazei de date etc.

Implementarea Repository Pattern-ului în cadrul aplicației are următoarele avantaje:

* Logica de afaceri poate fi testată unitar fără logica de acces la date;
* Codul de acces la baza de date poate fi utilizat;
* Codul de acces la baza de date este gestionat central (“Core”) și permite foarte ușor de implementat orice politici de acces la baze de date, cum ar fi memoria caching;
* Este ușor de implementat logica domeniului.

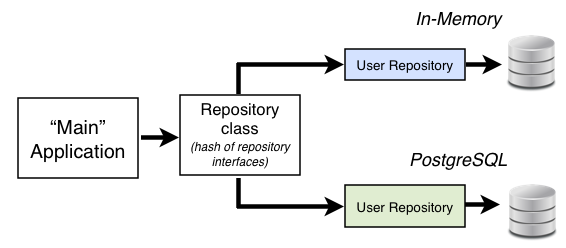


Figura 16: Integrarea Repository Pattern într-o aplicație cu conexiune la o baza de date

DataLayer: În cadrul acestui nivel se regasesc obiectele realizate de către ORM, obiecte ce oglindesc tabelele prezente în baza de date. Tot aici se regasesc repository urile aferente fiecărei entități din aplicația, cat și clasa “UnitOfWork”. Un repository of data instantiat, creeaza o conexiune la baza de date, cu ajutorul careia putem folosi query-urile de comunicare cu aceasta. În caz ca aveam de modificat sau creeat mai multe entități diferite, pentru a nu creea cate o conexiune la baza de date pentru fiecare entitate ( acest lucru nu este deloc eficient și poate duce la probleme ), vom instantia un “UnitOfWork”, acesta o dată fiind instantiat creeaza o singura legatura la baza de date, indiferent cate repository uri sunt create cu ajutorul acestuia.

EHealth: La acest nivel se regaseste partea de client, fișierele de configurare, fișierele de css cât și fișierele javascript. Tot în cadrul acestui nivel găsim controller-ele, aceste facilitează legătura dintre client și server. Clientul trimite o cerere server-ului prin intermediul apelurilor de tip Ajax, către metodele publice din interiorul controller ielelor.

Custom Membership: La acest nivel se adaugă un nivel peste sistemul de autentificare a celor de la Microsoft pentru a facilita o autentificare bazată și pe alte criterii, nu doar pe username și parola, cât și posibilitatea de a extinde functionalitatile de baza aduse de cei de la Microsoft.

## **4.2. Diagrama fluxului de date**

În următorul subcapitol vom prezenta diagrama fluxului de date, atat pentru un user obișnuit cât și pentru un user de tip admin.

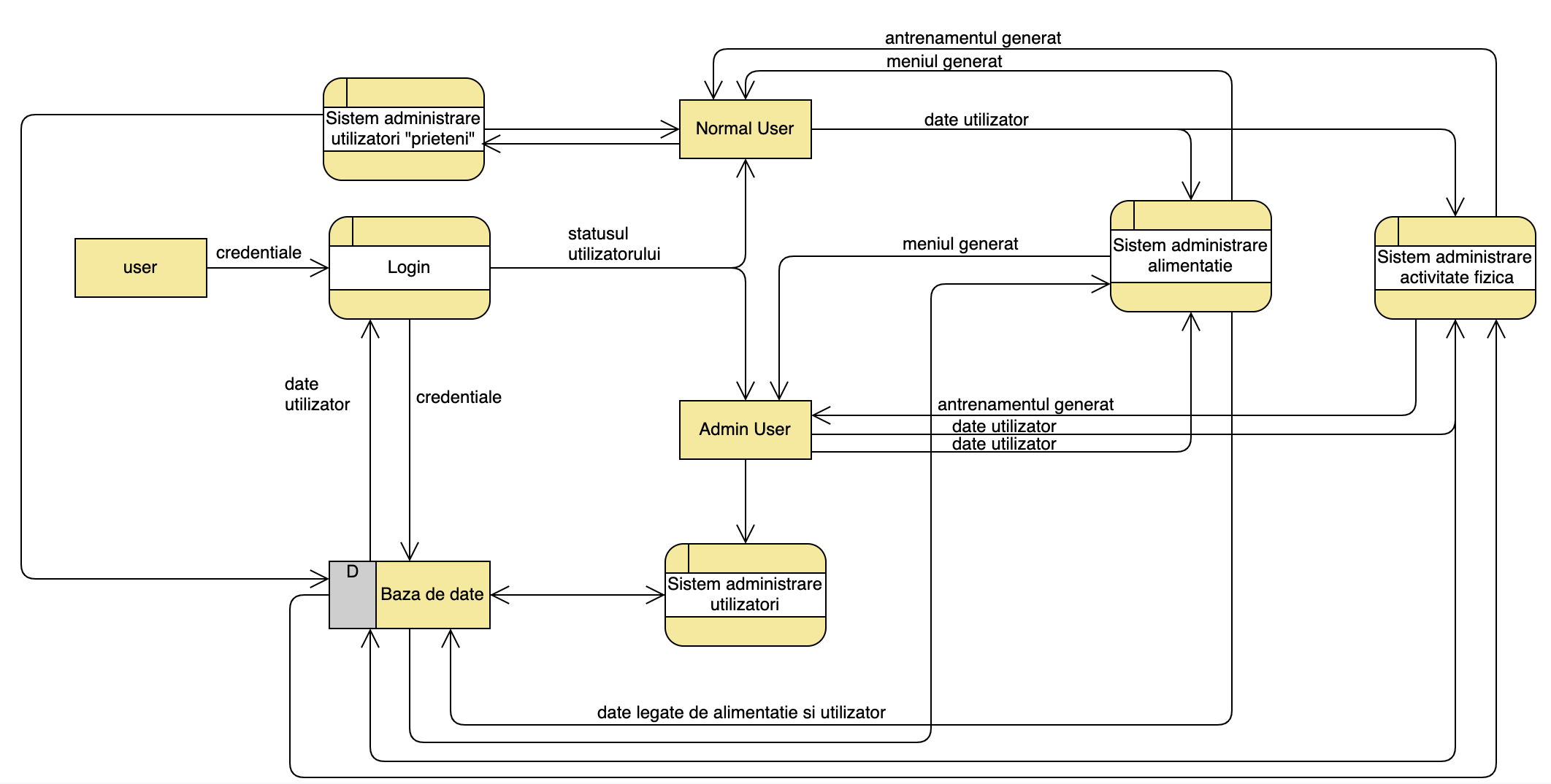


Figura 17: Diagrama fluxului de date în cadrul aplicației EHealth

Sistemul Login: În cadrul acestui sistem, datele sunt preluate de la user, sunt encriptate și sunt verificate cu cele salvate în baza de date. Dacă în urma verificării s-a constatat ca utilizatorul căutat după username și parola, nu exista, se returnează un cod de eroare, care mai apoi vine interpretat de partea de client pentru a atenționa utilizatorul, ca exista posibilitatea ca datele introduse de acesta sa fie eronate sau contul acestuia nu exista. În cazul în care utilizatorul nu are cont în baza de date, acesta are opțiune sa își creeze unul, accesand pagina de registrare. Dacă în schimb autentificarea utilizatorului a fost cu succes, atunci sistemul de Login verifica statusul acestuia, fiecare utilizator are un status în urma căruia se realizeaza diferențierea între un utilizator normal și unul admin. În funcție de statusul utilizatorului anumite functionalitati prezente în cadrul aplicației ar putea sa nu fie prezente ( nu am dorii ca un utilizator normal sa aibă acces la sistemul de administrare de utilizatori ).

Sistem administrare alimentatie: Acest sistem primește ca date de intrare, detalii legate de utilizator, introduse de aceștia în aplicație, cât și date generate de aplicație pentru acest utilizator. În urma datelor primite acest sistem le va salva în baza de date iar mai apoi aceste date vor fii prelucrate cu scopul de a genera un meniu de alimentatie specific nevoilor acestui utilizator. Tot în cadrul acestui sistem utilizatorul are posibilitatea sa creeze alimente și meniuri care nu se regasesc în baza de date a aplicației și care, acesta, considera ca îi sunt mai potrivite lui.

Sistem administrare activități fizice: La fel ca și sistemul de alimentatie, acest sistem primește datele specifice utilizatorului și datele generate de aplicație pentru a genera mai multe sesiuni de antrenamente împărțite pe o durata de timp variabila. Tot aici utilizatorul are posibilitatea de as introduce exercițiile sale proprii cât și să formeze antrenamente cu ajutorul acestora.

Sistem administrare utilizatori și utilizatori “prieteni”: În cadrul acestor 2 sisteme se procesează relațiile dintre utilizatori ai posibilitatea acestora de a comunica între ei.

În cazul adminului, acesta are posibilitatea de a vizualiza toti utilizatorii prezenti in aplicatie și de a aplica acțiuni asupra acestora. Utilizatorul de tip administrator are posibilitatea sa stearga, creeze, sa opreasca accesul la aplicație, utilizatori care nu au statusul de admin.

În cazul utilizatorului normal, acesta are posibilitatea de a vizualiza lista cu toți utilizatorii și de a trimite cerere de prietenia acestora. În urma acceptării cererii de către utilizatorii care au primit, aceștia vor putea comunica între ei prin intermediul unui chat, cât și să vadă detalii legate de acestea.

Baza de date: Aceasta este împărțită în două, o parte se găsește local pe computerul personal și cea de-a doua partea se regaseste in Azure Storage. Azure Storage este soluția de stocare cloud creata de cei de la Microsoft pentru scenarii moderne de stocare a datelor. Azure Storage oferă un magazin de obiecte scalabile masiv pentru obiecte de date, un serviciu de sistem de fișiere pentru cloud, un magazin de mesagerie pentru mesagerie de încredere și un magazin NoSQL.

Accesul din cadrul aplicației la Azure Storage se face cu ajutorul clasei “Azure Helper”. Aceasta clasa conține metodele de baza pentru a realiza operații de scriere, citire, ștergere în cadrul Azure Storage.

## **4.3. Diagrama relațiilor dintre entități**

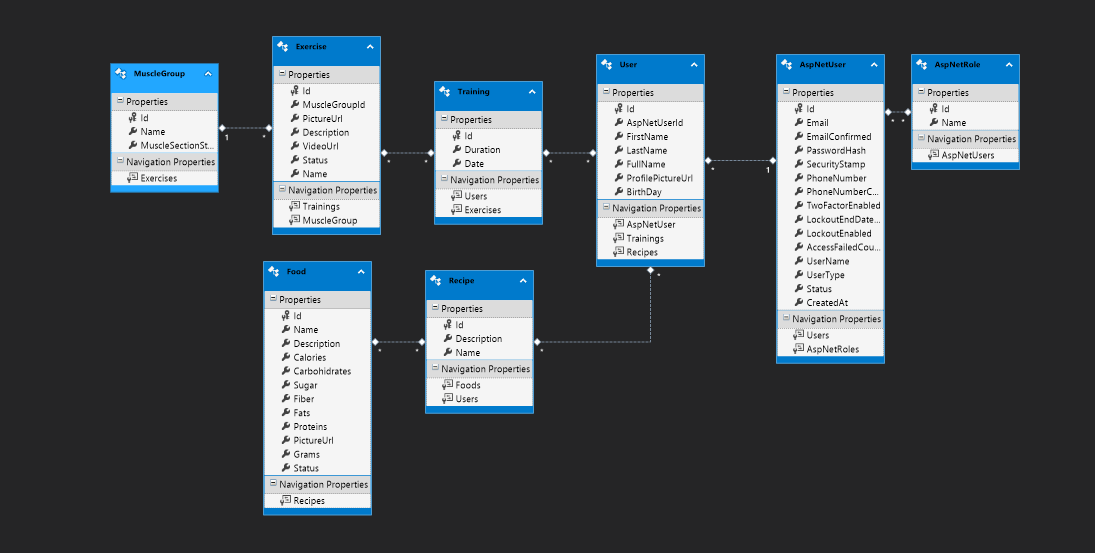


Figura 18: Diagrama relațiilor dintre entitățile prezente în cadrul aplicației EHealth

Lista entităților: Muscle Group, Exercise, Training, User, AspNetUser, AspNetRoles, Recipe, Food.

Fiecare entitate are un rol bine definit în cadrul arhitecturii, astfel avem următoarele roluri:

* Muscle Group - Reprezinta grupa musculara, o grupa musculara are mai multe Exerciții asociate acesteia.
* Exercițiile - Au o grupa de muschi, date specifice unei grupe de muschi în contextul antrenarii acesteia (descriere, nume, videoUri), un status care ne spune dacă grupa de muschi a fost sau dezactivata din aplicație (în cadrul aplicației, operațiile de ștergere schimba campul status a entităților). In momentul in care aceste entitati vor fii afisate in interfata grafica sau se vor folosii pentru anumite calcule de logica, se v-a tine cont de status ca acesta sa fie “ACTIVE”. Pentru a tine evidenta statusurilor de stare am folosit enumul “EntityStatus”.
* Training - Reprezinta o sesiune de antrenament, o sesiune de antrenament are o lista de exerciții și de user, odată ( data calendaristică la care a fost programat sesiunea ) și o durata de timp ( cat dureaza completarea antrenamentului ).
* User - acesta reprezinta utilizatorul aplicatie si contine date specifice unei persoane ( first name, last name, picture url, birth dat ), de asemenea mai contine o lista de antrenamente și o lista de meniuri alimentare asociate acestuia. Propietatea AspNetUserId, implicita AspNetUser reprezinta entitatea generata de .NET pentru acțiunile de autentificare și permisiunile de rol asociate acestuia.
* AspNetUserRoles - Aceasta entitate este una din cele generate de .NET, în cadrul acestei entități sunt specificate rolurile utilizatorilor din cadrul aplicației. În momentul de fata contine 2 tipuri de roluri: utilizator obișnuit, admin.
* Recipe - Entitatea aceasta reprezinta meniurile alimentare prezente in aplicație, acestea sunt formate din mai multe entități de tip alimente.
* Food - Entitatea aceasta reprezinta un aliment.

Admin-ul fata de utilizatorii obișnuiți are acces la toate caracteristicile aplicație plus un sistem de administrare a tuturor utilizatorilor care nu sunt admin. Un utilizator de tip admin nu poate interactiona cu un alt admin.

## **4.3. Diagrame de secvență**

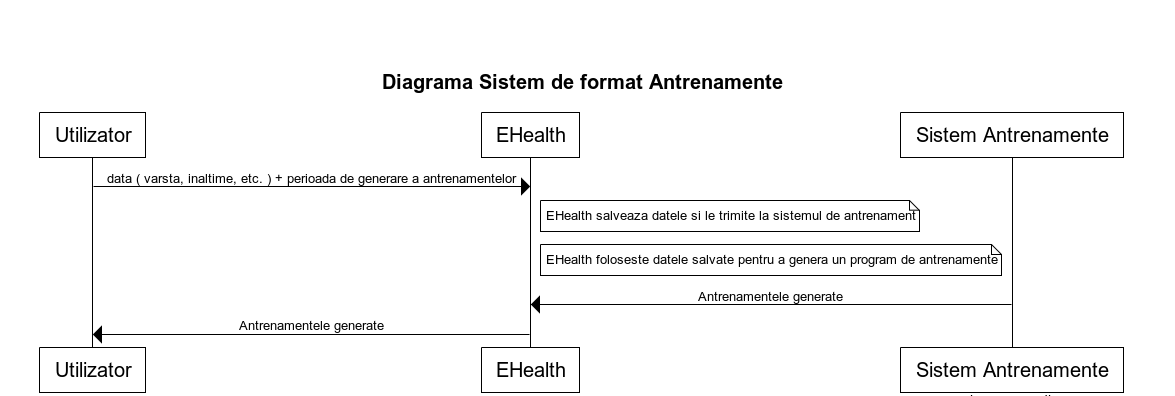
****

Figura 19: Diagrama secvențială a sistemului de antrenamente

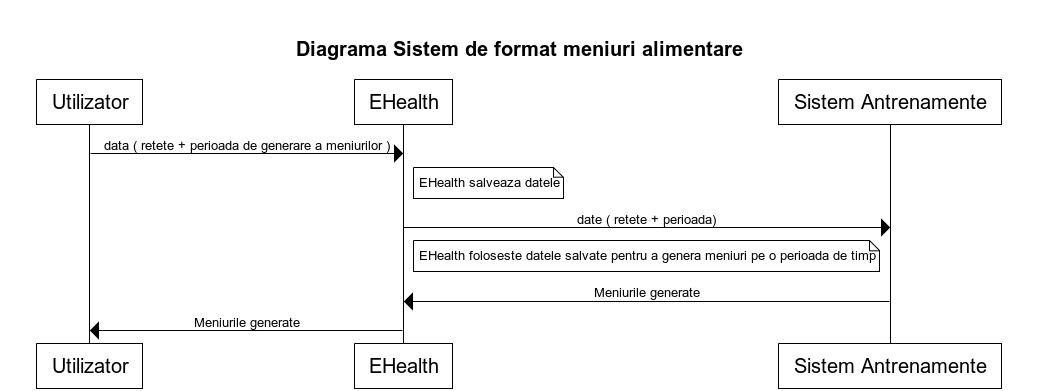
****

Figura 20: Diagrama secvențială a sistemului de generat meniuri

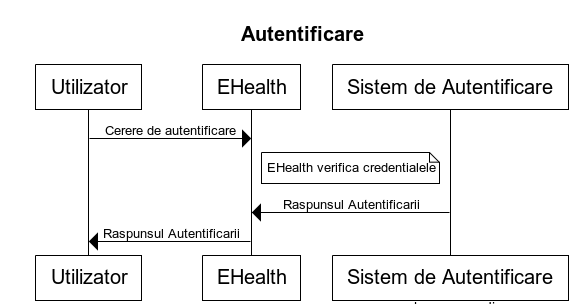


Figura 21: Diagrama secvențială a procesului de autentificare

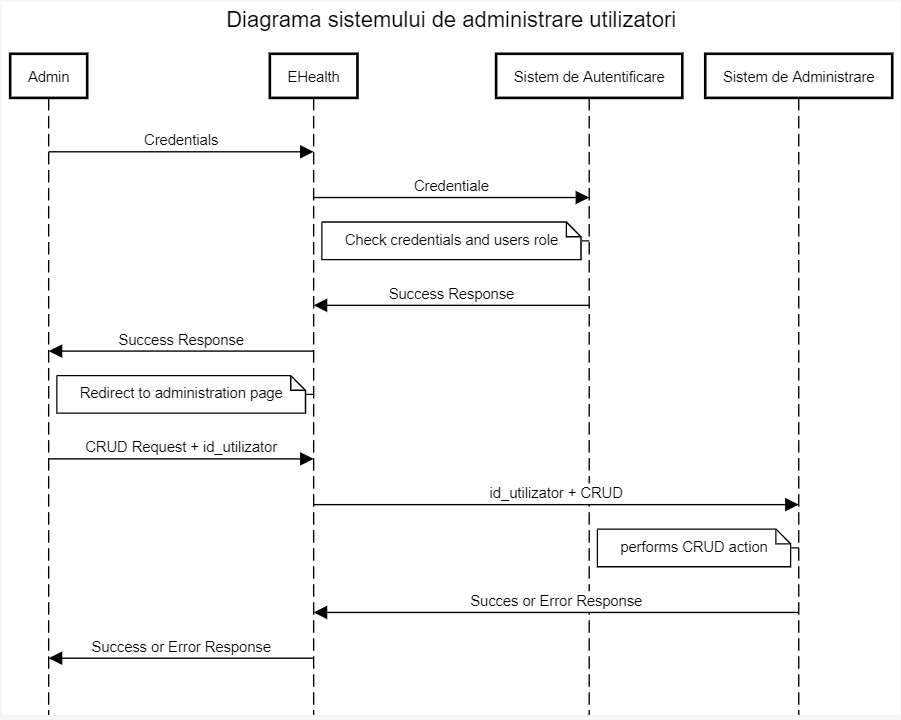
**

Figura 22: Diagrama secvențială a sistemului de administrare

# **5. Procesul de dezvoltare al sistemului**

Pentru a facilita eficienta în dezvoltarea sistemului a fost nevoie de o forma de organizare a caracteristicilor aplicației și starea lor de finalizare. Pentru aceasta am folosit serviciul celor de la Microsoft numit Team Foundation Server. Team Foundation Server (prescurtat TFS) este un produs Microsoft care oferă gestionarea codului sursă (fie cu Team Foundation Version Control sau Git), raportare, gestionarea cerințelor, managementul proiectelor (atât pentru dezvoltarea de software software, cât și pentru echipele de cascadă), construcții automate, capabilități de laborator, testare și gestionare a eliberării Acoperă întreg ciclul de viață al aplicației și permite funcțiile DevOps. TFS poate fi utilizat ca o bază pentru numeroase medii de dezvoltare integrate (IDE), dar este adaptat pentru Microsoft Visual Studio și Eclipse pe toate platformele.

Mai jos avem o captura de ecran a unui moment din dezvoltarea aplicație folosind TFS. Putem observa din imagine caracteristicile ce trebuiesc implementate, stadiul de finalizare la care se afla și tipul de caracteristică (bug, feature, etc.).

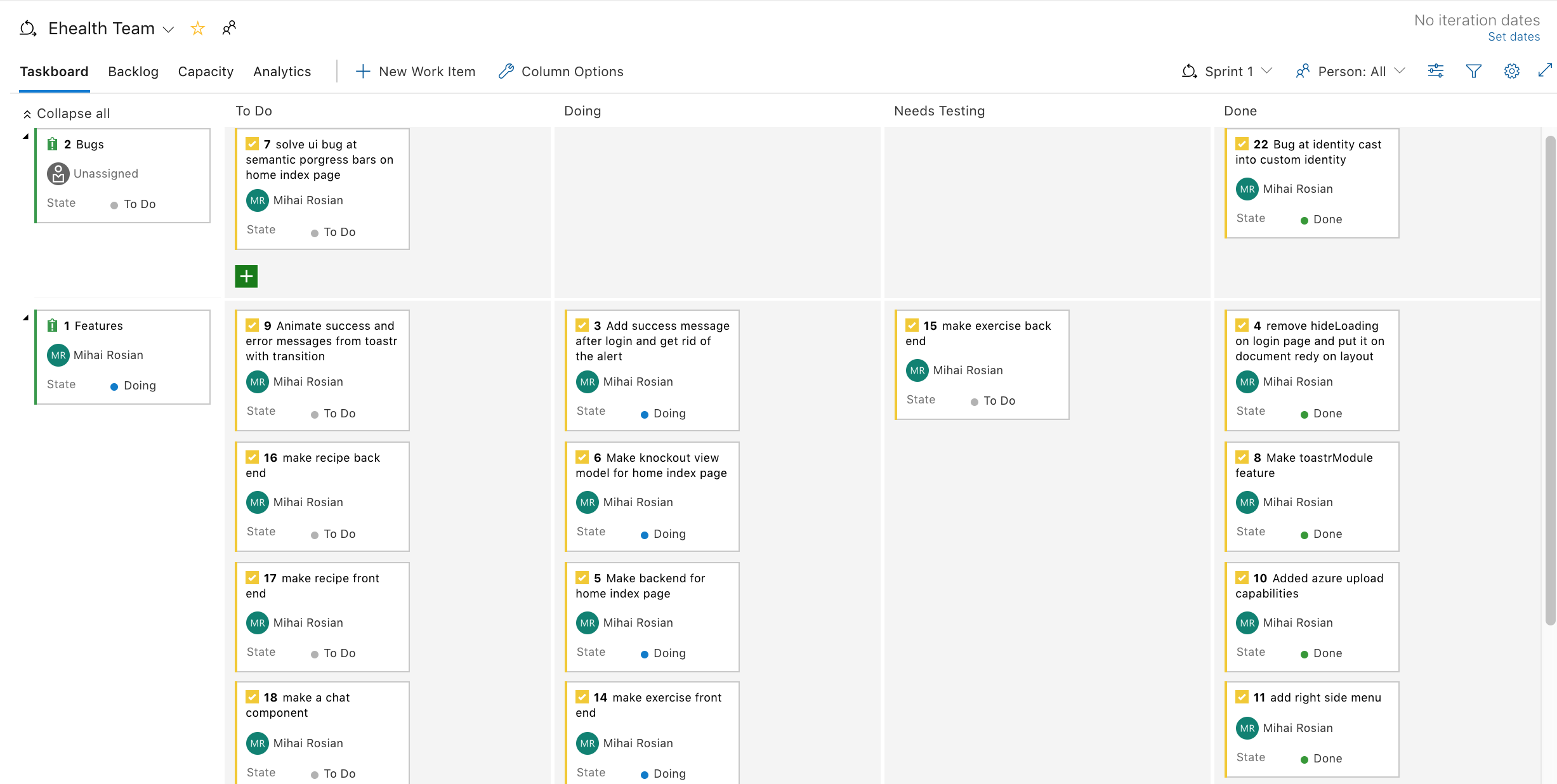


Figura 23: Vizualizare a unui moment din cadrul procesului de dezvoltarea al aplicației (campurile colorate cu galben, reprezinta task-uri ce trebuiesc implementate în cadrul aplicației EHealth)

După stabilirea temei de licenta, am început sa structurez lucrarea în mai multe caracteristici și să le adaug în TFS, după adaugare le-am stabilit un grad de prioritate și dificultate.

## **5.1. Proiectarea interfeței utilizator**

Deoarece am vrut ca aplicația să fie concepută aproape în întregime de mine, am decis ca atat logo-ul aplicației cât si aspectul vor fii concepute de mine. Pentru acest lucru am folosit aplicația numita Zeppelin. Zeplin este un instrument de colaborare, conceput pentru designeri de UI și dezvoltatori frontend. Cu Zeppelin, colaborarea dintre proiectanți și dezvoltatori devine ușoară, eficientă și economisește timp. Designerii pot transforma designurile lor în specificații și orientări, iar dezvoltatorii pot genera fragmente de cod legate de platformă. Am folosit aceasta aplicatie cu scopul de a dezvolta aplicația pentru licenta în continuare și după terminarea acesteia.

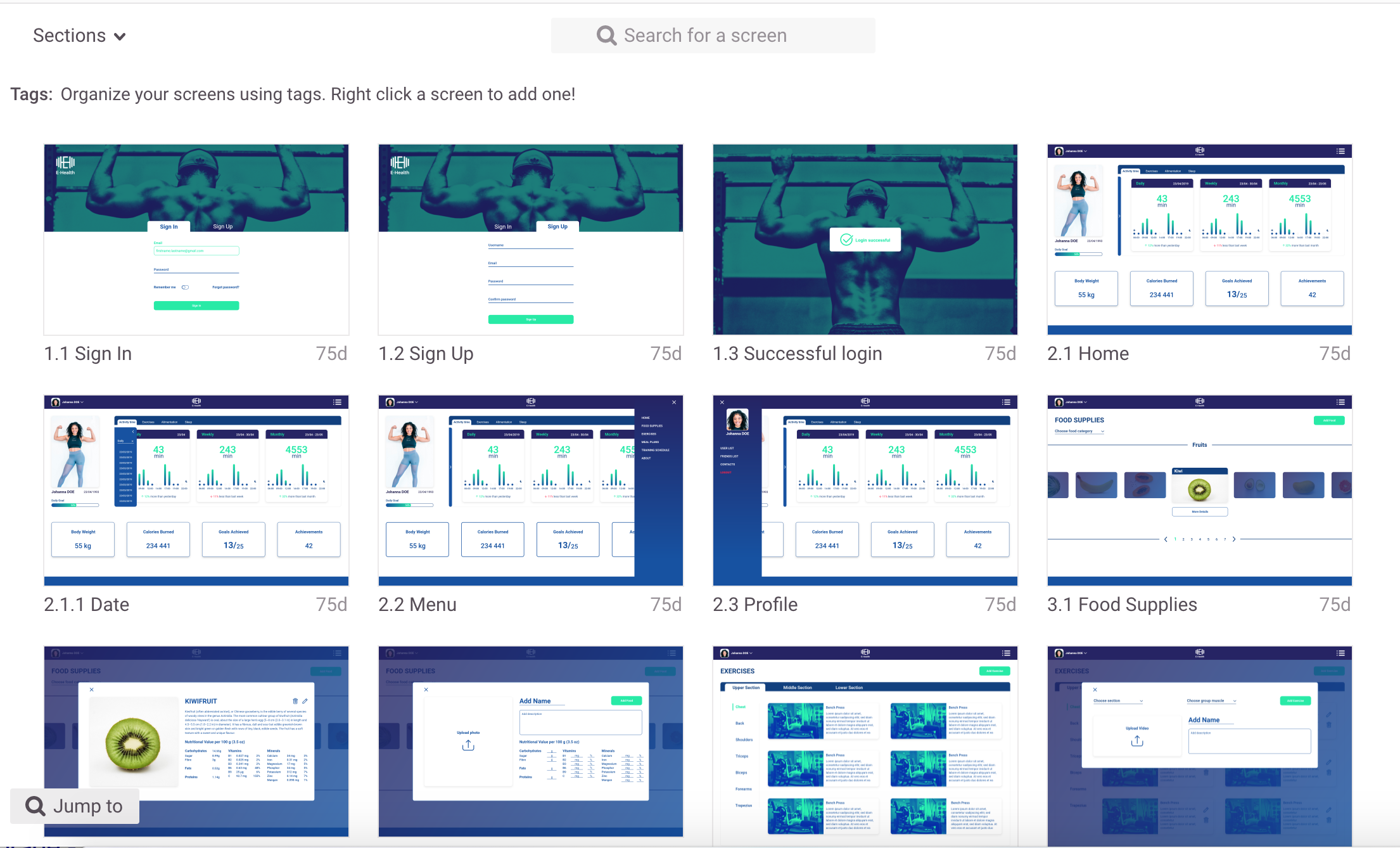
Figura 24: Captura de ecran a designului aplicației, realizat în aplicația Zeplin



Figura 25: Logo-ul aplicației EHealth

Paleta de culori a fost formata din culorile: #ffffff, #262266, #1250a3, #04eba4.

Fiecare pagina prezenta în aplicație a fost proiectată folosind un proces de UI/UX. Proiectarea experienței utilizatorului (UXD) este procesul de îmbunătățire a satisfacției utilizatorului față de un produs, prin îmbunătățirea capacității de utilizare, accesibilitate și dezirabilitate oferite în interacțiunea cu un produs. Proiectarea experienței utilizatorului cuprinde designul tradițional de interacțiune om-calculator (HCI) și îl extinde prin abordarea tuturor aspectelor unui produs sau serviciu percepute de utilizatori.

Pasii proiectarii au fost următorii:

* Înțelegerea: înțelegerea cerințelor, creearea personalității utilizatorului, (“Cine folosește aplicația”, “Cu ce scop o v-a folosii?”, etc.), definirea cazurilor de folosinta;
* Documentare: analiza competiției, analiza trendurilor de UI/UX, respectarea instrucțiunilor;
* Prototip: adunare de idei, alcătuire de scheme și diagrame, evaluare și proiectare;
* Proiectare: proiectarea imaginilor, crearea de prototipuri, definirea de instrucțiuni de UX;
* Implementarea: implementarea functionalitatiilor, implementarea experienței;

## **5.2. Proiectarea funcționalității**

Proiectarea funcționalității a început de la organizarea în Visual Studio Online a caracteristicilor aplicației și a proiectului în sine. O data realizate acest lucru, pentru aplicația în sine, a fost nevoie de un sistem de versionare a aceștia, pentru a se evita probleme de compatibilitatea a componentelor noi introduse în cod și pentru a păstra o istorie a modificărilor făcute asupra proiectului. Pentru acest lucru am folosit Git și interfața utilizator gratuită de Git a celor de la Atlassian numita “Source Tree”.

Git-ul este un sistem distribuit de versionare pentru urmărirea modificărilor codului sursă în timpul dezvoltării de acestuia. Este conceput pentru coordonarea lucrărilor între programatori, dar poate fi utilizat pentru a urmări modificările din orice set de fișiere. Obiectivele sale includ viteza, integritatea datelor și asistență pentru fluxuri de lucru distribuite, neliniare.

Redactarea codului sursă pentru partea de software al aplicației a fost realizată în mediul de dezvoltare software numit Visual Studio 2019. Microsoft Visual Studio este un mediu de dezvoltare integrat (IDE) de la Microsoft. Este utilizat pentru a dezvolta programe de calculator, precum și site-uri web, aplicații web, servicii web și aplicații mobile. Visual Studio folosește platforme de dezvoltare software Microsoft, cum ar fi Windows API, Windows Forms, Windows Presentation Foundation, Windows Store și Microsoft Silverlight. Poate produce atât codul nativ, cât și codul gestionat.

# **6. Concluzii**

## **6.1 Îmbunătățiri și idei viitoare**

În cazul în care pe viitor aplicația va scala la un număr mare de utilizator, vor fii nevoie de imbunatatiri pe partea de securitate a aplicație. Cateva din aceste îmbunătățiri ar fii:

* Înregistrarea utilizatorilor pe baza de confirmare prin email
* Implementarea unui sistem de autentificare folosind aplicații a altor organizații (third party authentication)
* Implementarea unui generator de coduri, pe care utilizatorii ar trebuii sa il adaugă în momentul în care vor sa se conecteze la aplicație, pe langa username și parola.

Pe partea de client as dorii ca pe viitor sa pun accentul mai mult pe partea de css, astfel incat aplicație sa fie responsive și pe ecrane de telefon. Momentan aplicația suporta majoritatea ecranelor de laptop și tabletă. Datorită acestui lucru am putea evita crearea unei aplicații de telefon în viitorul apropiat. Deasemenea as dorii sa extind funcționalitățile aplicației în așa fel incat aceasta sa poata fii folosită și ca un sistem de administrare a clienților unei săli de fitness. Acest lucru ar aduce un bonus destul de mare clienților sălii, deoarece în momentul în care aceștia își vor achita abonamentul și vor primii aplicația din cadrul sălii, aceștia vor avea și un sistem de antrenamente și meniuri alimentare oferite de sala pe care le pot urma.

În momentul dezvoltării unui sistem de administrare a unei săli de fitness as putea lua în considerare proiectarea unei aplicații pentru telefon pentru a separa sistemul de administrare de aplicația fitness.

O alta îmbunătățirea ce as dorii sa o fac pe viitor este sa mut implementarea aplicației din framework-ul .NET în cel de .NET Core. .NET Core este un framework gratuit și open-source, gestionat pentru computer pentru sistemele de operare Windows, Linux și macOS. Este o sursă deschisă, multiplă platformă , succesoare la .NET Framework. Proiectul este dezvoltat în principal de Microsoft și lansat sub Licența MIT.

.NET Core acceptă integral C # și F # și acceptă parțial Visual Basic .NET.

In prezent, VB.NET compilează și rulează .NET Core, dar Visual Runtime separat nu este implementat. Microsoft a anunțat că .NET Core 3 va include Visual Basic Runtime. Din octombrie 2018, C ++ / CLI nu este încă acceptat, deși asistența este planificată pe Windows.

Folosind framework-ul de .NET Cores se poate scădea costul de întreținere al aplicației cat și costul de gazduire. Un server ce rulează sistemul de operare Linux este mai puțin costisitor decat unul ce rulează Windows Server. Astfel dacă aplicația poate rula pe mai multe platforme opțiunile devin mai vaste.

De asemenea pentru modularizarea aplicatiei as dorii pe viitor o modificare a arhitecturii de baza în așa fel incat aplicația sa nu mai fie un întreg (monolita) ci sa fie alcatuita din mai multe servicii (microservices). Pentru acestea as putea folosii tehnologii cum ar fii Kubernetes, Docker, etc.

Docker este un set de produse platform-as-a-service (PaaS) care utilizează virtualizarea la nivel de sistem de operare pentru a livra software în pachete numite containere. Containerele sunt izolate unul de celălalt și își grupează propriul software, biblioteci și fișiere de configurare; ei pot comunica între ele prin canale bine definite. Toate containerele sunt conduse de un singur nucleu al sistemului de operare și sunt astfel mai ușoare decât mașinile virtuale.

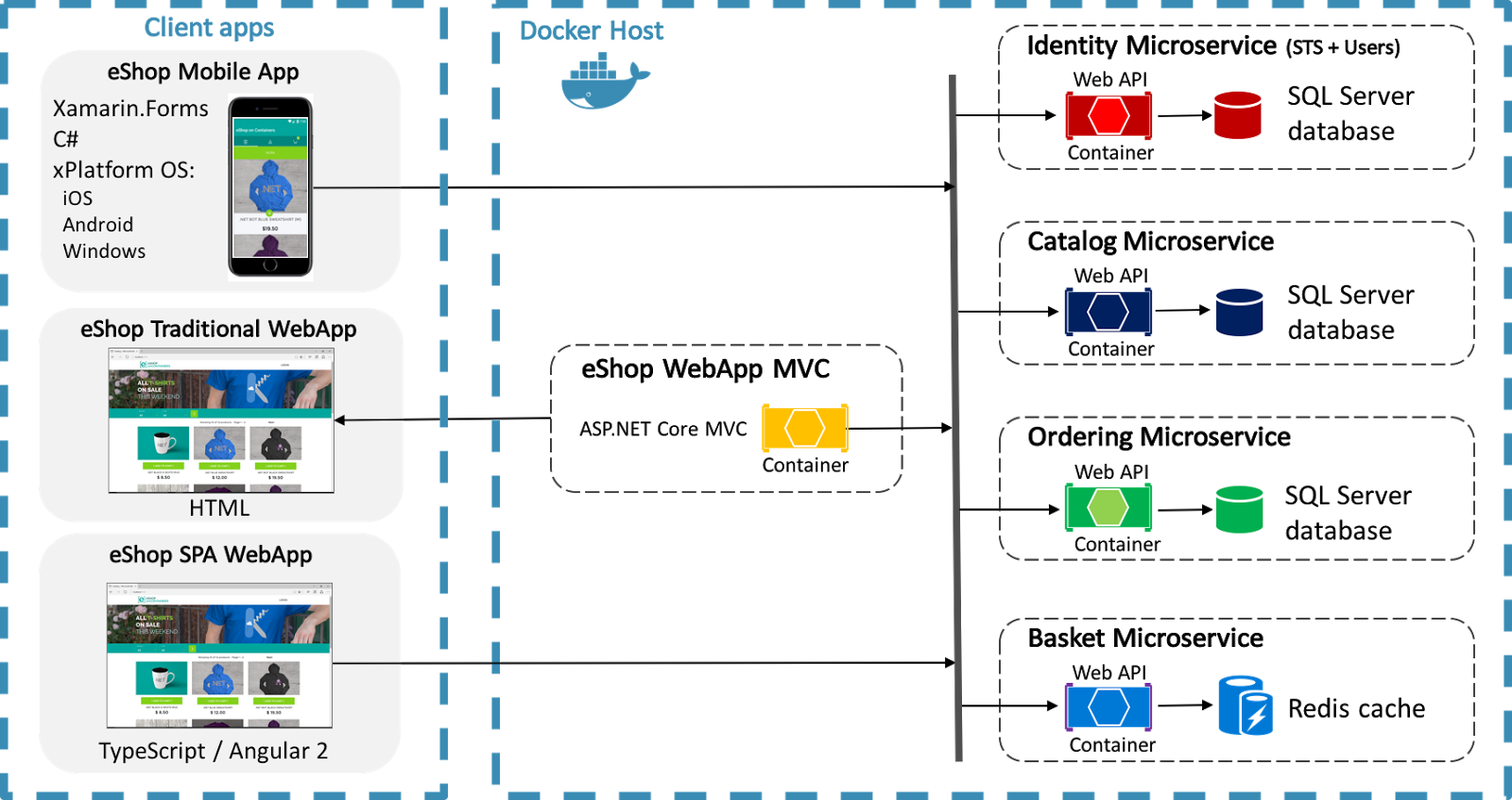


Figura 26: Exemplu de posibila arhitectura a unei aplicații web folosind Docker

Kubernetes (prescurtat ca k8s) este un sistem open-source container orchestration pentru automatizarea implementării, scalarii și gestionării aplicațiilor. A fost inițial proiectat de Google, iar acum este întreținut de Cloud Native Computing Foundation. Acesta își propune să ofere o „platformă pentru automatizarea operațiunilor de desfășurare, scalare și aplicare a containerelor în clusterele gazdă”. Funcționează cu o serie de unelte pentru containere, inclusiv Docker. Multe servicii cloud oferă o platformă sau infrastructură bazată pe Kubernetes ca serviciu (PaaS vs IaaS) pe care Kubernetes poate fi implementat ca un serviciu care furnizează platformă. Mulți furnizori oferă, de asemenea, propriile distribuții Kubernetes.

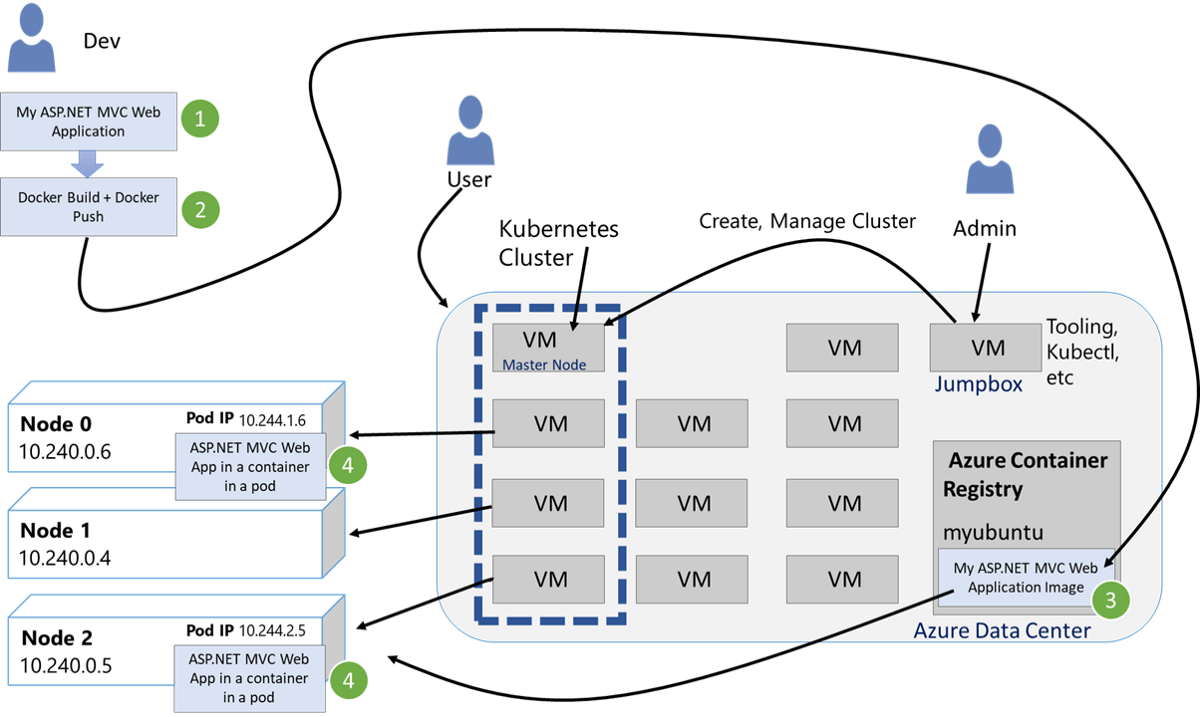


Figura 27: Posibila arhitectura a unei aplicații web ce foloseste ASP.NET si MVC, folosind Kubernetes.

Pentru a imbunatatii viteza de navigare în cadrul aplicației as dorii ca aplicația să fie realizata într-o singura pagina (single page application).

Și ultima îmbunătățirea pe care o pot vedea în momentul de fata ar fii pe partea de client. As putea pe viitor sa folosesc un framework de javascript mai popular. Acest lucru mi-ar permite sa am un suport mai mare din partea utilizatorilor în număr mare, mi-ar permite să îmi dezvolt expertiză în acest framework, ar duce la o mai buna modernizare a părții de client de cat cea existenta acum, ar crește viteza la care se mișca lucrurile. Cateva dintre aceste framework-uri ar putea fii: React ( React Hooks ), Vue.Js, Angular 8.

Dintre aceste framework-uri as tinde spre a folosii React Hooks, acesta vine la pachet cu un DOM virtual care este mai rapid. React (cunoscut și sub numele de Reactjs sau ReactJS) este o bibliotecă JavaScript pentru construirea de interfețe de utilizator. Acesta este întreținut de Facebook și de o comunitate de dezvoltatori și companii individuale.

React poate fi utilizat ca bază pentru dezvoltarea de aplicații web cu o singură pagină (single page application) sau mobile, deoarece este optim pentru a obține date cu schimbare rapidă care trebuie înregistrate. Cu toate acestea, preluarea datelor este doar începutul a ceea ce se întâmplă pe o pagină web, motiv pentru care aplicațiile complexe React necesită, de obicei, utilizarea bibliotecilor suplimentare pentru gestionarea de stat, rutarea și interacțiunea cu o API: Extjs și Gatsbyjs sunt exemple de astfel de biblioteci.

În procesul de studiu al framework-urilor existente, am intampinat probleme în cadrul Actului cu modul în care acesta pasează date (one way flow). Datele sunt trimise de la părinte către copil, singura modalitate prin care copilul poate comunica cu părintele este prin “callback uri” de la apelul părintelui. Acest lucru face foarte dificila pasarea de date între componentele care formează aplicația.

React Hooks vine cu o soluție pentru aceasta problema printr-un set de metode prin care copii pot ridica (rise) metode la părinți, iar aceștia pot sa expedieze funcții la aceștia(dispatch). Hooks sunt funcții care îi permit dezvoltatorilor să „conecteze” la reacția caracteristicilor de stare și ciclu de viață de la componentele funcției. Hooks nu funcționează în cadrul claselor - te lasă să folosești React fără clase. React oferă câteva cârlige încorporate precum useState. Puteți, de asemenea, să vă creați propriile hooks pentru a reutiliza comportamentul statistic între diferite componente.

Hooks nu este singura arhitectura construită folosind React, alte arhitecturi ar fii: React Flux, React Redux.

Flux este o arhitectură pe care Facebook o folosește intern atunci când lucrează cu React. Nu este un cadru sau o bibliotecă. Este pur și simplu un nou tip de arhitectură care completează React și conceptul de Unidirecțional Data Flow.

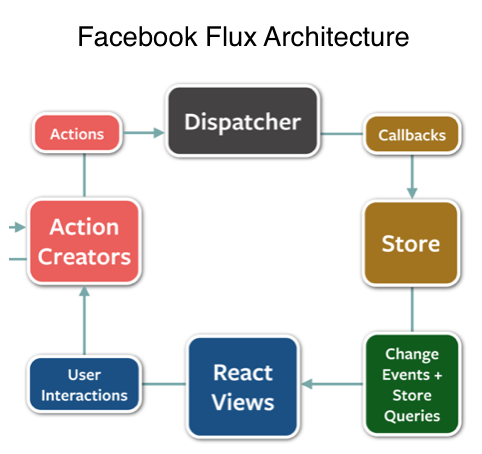


Figura 28: Arhitectura flux a celor de la facebook

Redux este o bibliotecă JavaScript open-source pentru gestionarea stării aplicației. Este cel mai frecvent utilizat cu biblioteci, cum ar fi React sau Angular, pentru construirea interfețelor utilizatorului. Asemănător arhitecturii Facebook Flux (și inspirat de), a fost creat de Dan Abramov și Andrew Clark.

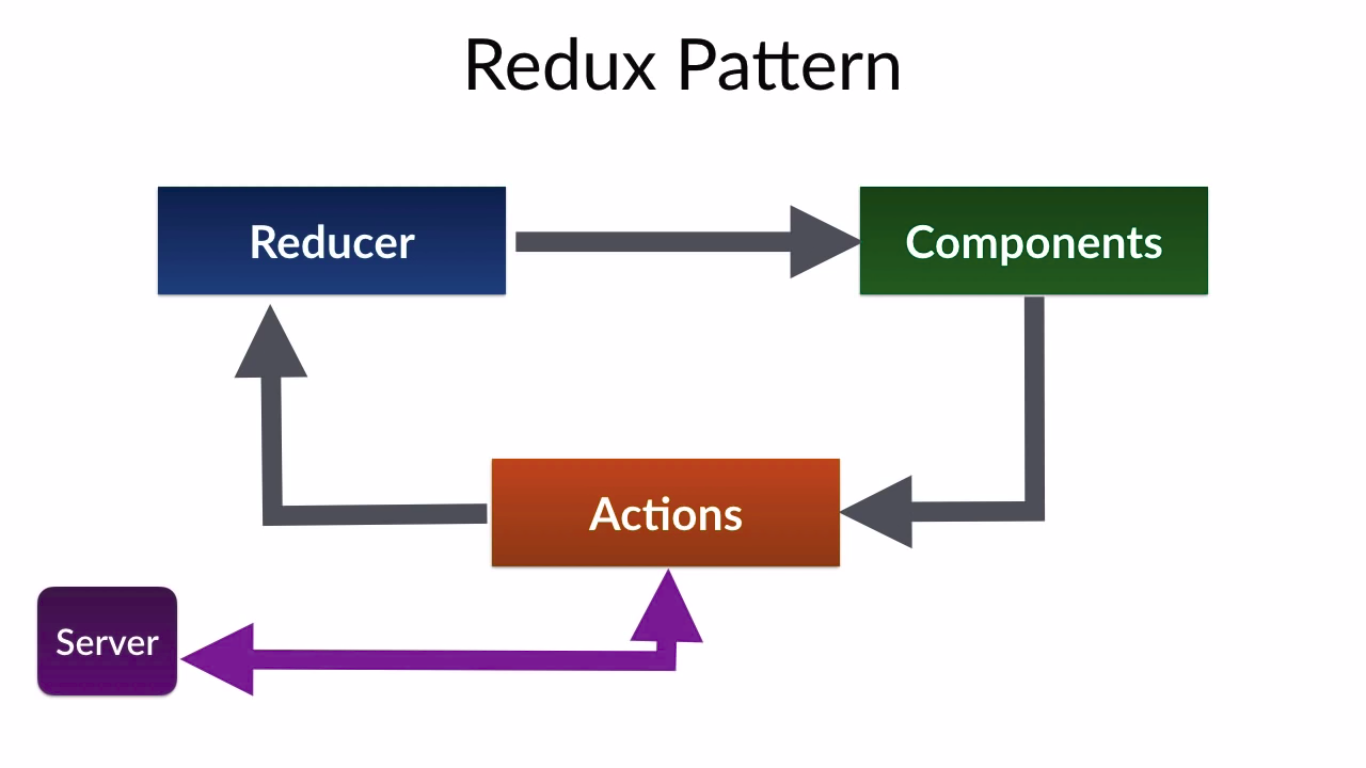


Figura 29: Diagrama arhitecturii Redux

## **6.2 Concluzii**

În concluzie aplicația mai poate sa fie imbunatatia mult, atat pentru a crește scalabilitatea cât și a imbunatatii caracteristici existente. Chiar fără toate aceste îmbunătățiri scopul aplicației este respectat, aceasta ridicandu-se la așteptările oricărui utilizator.

Aplicația a fost proiectată pentru a ajuta oamenii care muncesc într-un loc de munca ce implica statul la birou o perioada mai lungă de timp, aceasta aplicatie încearca sa ușureze partea de documentare și de studiu a unei persoane dornice sa inceapa un stil de viata sanatos, prin generarea activităților fizice, meniurilor alimentare și urmărirea progresului utilizatorului.

Un utilizator obișnuit, în urma achiziționării unui abonament la sala, ar folosii aplicatia pentru a-si genera o lista de antrenamente pe saptamana, o lista cu meniuri alimentare pe o saptamana. Dacă utilizatorul dorește poate sa modifice antrenamentele și sau sa adauge exerciții noi, create de acesta în sistem și folosind toate aceste caracteristici generate de aplicație, să-și atingă scopurile. Aplicația îl încurajează mereu pe utilizator și întreține dorința de a se dezvolta mereu, prin multe caracteristici extrase din jocurile din ziua de azi. Folosind aplicația zilnic, utilizatorul va avea o experienta asemănătoare unui joc, acesta v-a primii punctaje în urma îndeplinirii sarcinilor zilnice, iar în urma punctajului va primii premii în cadrul aplicației. Acest sistem de premiere poate fii folosit și în cadrul unei săli, pentru a încuraja clienții și pentru ai atrage și pe alții.

# **Bibliografie**

[1] Informatii generale despre proiecte implementate cu Asp.net si arhitectura MVC - Pro ASP.NET MVC 5 Book by Adam Freeman

[2] Arhitectura MVC - <https://www.tutorialsteacher.com/mvc/mvc-architecture>

[3] Integrarea aplicatiei cu IIS - Professional IIS 7 and ASP.NET Integrated Programming by Shahram Khosravi

[4] JavaScript and JQuery: Interactive Front-End Web Development Book by David Sawyer McFarland

[5] CSS responsive folosind functia calc - <https://dev.to/tyzia/using-calc-in-css-responsive-design-5l1>

[6] Web Design with HTML, CSS, JavaScript and JQuery Set Book by Jon Duckett

[7] Limbajul de interogarea folosit de bazele de date din cadrul Microsoft Sql Server - T-SQL Fundamentals Book by Itzik Ben-Gan

[8] Integrarea aplicatiei cu Azure Storage - [https://azure.microsoft.com/](https://azure.microsoft.com/en-us/free/storage/search/?&ef_id=Cj0KCQjw2K3rBRDiARIsAOFSW_7NGx-KFo6VP0BwAAAx5PXktvegcY5Dc9OmRSLpfnMzEHm2S69ETPwaAiL7EALw_wcB:G:s&OCID=AID2000602_SEM_IgqxUS57&MarinID=IgqxUS57_363179200039_azure%20storage_e_c__75821091266_kwd-296465859739&lnkd=Google_Azure_Brand&dclid=CjgKEAjw2K3rBRCahJ7lrfTkhQsSJAB-TTRebOZXsEMFI-rSQW_az2BgHw9T5vDFkRkrarr4XXsLuPD_BwE)

[9] Aplicatia folosita pentru administrarea caracteristicilor aplicatiei - <https://app.vssps.visualstudio.com>

[10] Aplicatie folosita pentru proiectarea interfetei utilizator - <https://zeplin.io/>

[11] Informatii despre framework-ul de javascript folosit - <https://knockoutjs.com/>

[12] Studiu asupra UI/UX aplicațiilor din domeniul fitness:

* <https://tubikstudio.com/case-study-manuva-uiux-design-gym-fitness-app/>
* <https://blog.prototypr.io/ui-ux-case-study-of-gym-app-%EF%B8%8F-a71bd5155a89>
* <https://uxplanet.org/ux-in-health-and-fitness-apps-what-it-takes-to-do-it-right-bc503b363c0>
* <https://www.researchgate.net/publication/320284487_Usability_Testing_of_Fitness_Mobile_Application_Methodology_and_Quantitative_Results>

[13] Generalități despre UI/UX:

* Don’t Make Me Think book by Steve Krug
* <https://www.udemy.com/course/the-essential-uxui-designer-course/>
* <https://uxdesign.cc/ultimate-start-guide-for-beginner-ux-ui-designer-b848be089589>

[14] Detalii despre framework-ul de CSS - <https://semantic-ui.com/>

[15] Interfata utilizator pentru Git - <https://www.sourcetreeapp.com/>

[16] Informații Semantic UI - <https://www.sitepoint.com/introducing-semantic-ui-component-library/>

[17] Informații bază de date și server - <https://www.microsoft.com/en-us/sql-server/sql-server-2017>

[18] Informatii generale despre React - <https://reactjs.org/>

[19] Informare in legatura cu framework-uri populare de javascript -

* <https://raygun.com/blog/popular-javascript-frameworks/>
* <https://en.wikipedia.org/wiki/JavaScript_framework>
* <https://www.fusioncharts.com/resources/developers/js-frontend-frameworks-comparison>
* <https://www.youtube.com/watch?v=f8CzCX4XaFQ>

[20] Informații generale Docker -

* <https://www.docker.com/>
* <https://en.wikipedia.org/wiki/Docker_(software)>
* <https://www.youtube.com/watch?v=SC7lLm6QAb8>

[21] Informații generale Kubernetes -

* <https://kubernetes.io/>
* <https://en.wikipedia.org/wiki/Kubernetes>
* <https://www.youtube.com/watch?v=SC7lLm6QAb8>