



Universitatea POLITEHNICA București  
Facultatea Automatică și Calculatoare  
Departamentul Automatică și Informatică Industrială

## **LUCRARE DE DIPLOMĂ**

### **Sistem de monitorizare a persoanelor de vârstă a treia**

Coordonator

Prof. Dr. Ing

Valentin SGÂRCIU

Absolvent

Andreea-Iulia

CONCEA-PRISĂCARU

**2019**

# Cuprins

Cuprins .....	2
1. Introducere.....	3
1.1 Prezentarea domeniului în care se încadrează lucrarea .....	3
1.2 Justificarea alegerii temei.....	3
1.3 Obiectivele lucrării.....	4
1.4 Descrierea succintă a capitolelor lucrării .....	4
2. Problema abordată și soluții pentru rezolvarea acesteia.....	6
2.1 Problema abordată.....	6
2.2 Soluții existente.....	6
2.3 Soluția propusă.....	7
2.4 Avantajele soluției propuse .....	8
3. Scopul lucrării și grupul țintă .....	9
4. Etapele dezvoltării proiectului.....	11
4.1 Identificarea cerințelor funcționale .....	12
4.2 Proiectarea arhitecturii sistemului.....	13
4.3 Proiectare hardware.....	14
4.4 Proiectare software.....	17
4.5 Testarea funcționalităților .....	21
5. Componente utilizate.....	22
5.1 Node MCU .....	22
5.2 Senzor de puls XD58C.....	24
5.3 Senzor de temperatură DS18B20 .....	25
5.4 Modul accelerometru și giroscop cu 3 axe MPU6050 .....	26
5.5 Componente auxiliare .....	27
6. Tehnologii utilizate.....	28
6.1 Arduino IDE.....	28
6.2 Wi-Fi .....	29
6.3 IoT.....	30
6.4 Firebase .....	31
6.5 Android.....	33
6.6 Java & POO.....	36
6.7 XML.....	38
6.8 One Signal.....	40
7. Scenarii de utilizare .....	42
8. Dezvoltări ulterioare .....	47
9. Concluzii.....	48
10. Bibliografie .....	49

# 1. Introducere

## 1.1 Prezentarea domeniului în care se încadrează lucrarea

Termenul de Internet of Things (IoT) este un concept de actualitate și este răspândit din ce în ce mai mult în ultima perioadă în industria IT. IoT reprezintă conectarea mai multor dispozitive, de cele mai multe ori wireless, acestea comunicând între ele fie prin Internet, fie prin intermediul propriei lor rețele. Pentru a putea fi posibilă această comunicare, dispozitivele trebuie să poată interacționa cu alte dispozitive. De asemenea, ele trebuie să fie capabile să efectueze anumite operații cu datele (achiziție, prelucrare și controlare). Așadar, putem defini termenul de IoT ca fiind o rețea de dispozitive capabile să achiziționeze, controleze și să comunice mai departe datele cu care acestea acționează. Diferența între un dispozitiv conectat la Internet și un dispozitiv dintr-o rețea IoT este reprezentată atât de inteligența controlului, cât și de autonomie. Senzorii, dispozitivele de control, și în general toate “lucrurile” dintr-o rețea IoT sunt autonome și au elemente de inteligență a controlului. Așadar, ele pot fi programate să acționeze periodic, fără a mai fi necesară intervenția umană, de aici și necesitatea de autonomie a dispozitivelor IoT. Perspectivele oferite de IoT nu se referă strict la anumite îmbunătățiri ale domeniilor existente, ci mai degrabă la transformarea acestora în ceva nou, inteligent, ce poate în viitor chiar să suplinească forța de muncă umană.<sup>1</sup>

## 1.2 Justificarea alegerii temei

IoT reprezintă o rețea de obiecte ce încorporează circuite electronice ce permit comunicarea prin infrastructura existentă wireless/cablu, în vederea monitorizării sau controlului de la distanță. IoT se poate adresa unei varietăți foarte mari de soluții inteligente de monitorizare și control, cum ar fi: Smart Home, control inteligent al agriculturii, mașini inteligente, monitorizare inteligentă în domeniul medical etc. Această lucrare se adresează în special ramurii de achiziție și monitorizare a datelor a unui sistem de senzori pentru anumiți parametri vitali ai corpului uman (puls, temperatură). Am ales abordarea acestui domeniu deoarece oferă o perspectivă modernă în ceea ce privește industria medicală.

---

<sup>1</sup> [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things)

Deoarece monitorizarea mai multor pacienți în cazul unui deficit de personal poate reprezenta o problemă, IoT reprezintă o soluție de viitor pentru a o rezolva. Folosind funcționalități precum: notificări inteligente în funcție de nivelul anumitor parametri, achiziția și centralizarea datelor legate de anumiți parametri vitali într-o platformă accesibilă, se pot dezvolta o multitudine de soluții inteligente care pot revoluționa industria medicală.

### **1.3 Obiectivele lucrării**

Obiectivul principal al acestei lucrări este construirea unui sistem inteligent de monitorizare a câtorva parametri vitali pentru un pacient aflat fie într-o instituție medicală, fie la domiciliu. De asemenea, se urmărește dezvoltarea unei aplicații Android, aplicație ce va comunica cu dispozitivul prin intermediul unei baze de date în Cloud. Sistemul va monitoriza pulsul, temperatura și va detecta eventualele fluctuații în mișcarea utilizatorului (pacientului). Datele achiziționate de la senzori vor fi transmise prin intermediul Wi-Fi într-o bază de date, iar din baza de date acestea vor fi preluate mai departe de către aplicație. Sistemul va fi capabil să trimită notificări personalului medical în cazuri de urgență (valori scăzute/ridicate ale parametrilor sub/peste un anumit prag). În cazul fluctuațiilor mișcării, sistemul va fi capabil să detecteze eventualele căderi ale pacientului, folosind un senzor specializat. Un alt obiectiv al acestei lucrări este reprezentat de autonomia sistemului, deoarece se dorește ca acesta să fie cât mai independent de acțiunile umane asupra sa. De asemenea, se urmărește obținerea unui sistem portabil, pentru a facilita ușurința în utilizare (mobilitatea utilizatorului). Pentru acest lucru, se vor folosi conexiuni wireless.

### **1.4 Descrierea succintă a capitolelor lucrării**

Primul capitol al lucrării este unul introductiv, în care se descrie domeniul unde se încadrează lucrarea, mai precis domeniul Internet of Things. De asemenea, este prezentată și o scurtă justificare a alegerii temei, precum și obiectivele urmărite în realizarea acestui proiect.

În al doilea capitol este prezentată problema pe care această lucrare o abordează, soluții existente, soluția propusă în vederea rezolvării acesteia și avantaje.

Al treilea capitol descrie scopul lucrării: ce urmărește acest proiect și rezultatele așteptate de la acesta. Tot în acest capitol este prezentat grupul țintă pe care acest proiect îl vizează (viitori utilizatori).

Al patrulea capitol prezintă etapele de dezvoltare ale proiectului: identificarea cerințelor funcționale (ce ar trebui să facă acest sistem), proiectarea arhitecturii sistemului (design hardware), proiectarea hardware (realizarea propriu-zisă a arhitecturii identificate la pasul anterior și algoritmi pentru senzori), proiectarea software (dezvoltarea aplicației ce va comunica cu partea hardware) și testarea funcționalităților, pentru a determina dacă sistemul funcționează conform așteptărilor (achiziții de date corecte, prelucrarea și transmiterea datelor, sistemul de alarmare).

În cel de-al cincilea capitol sunt descrise componentele utilizate pentru proiectarea hardware a sistemului (microprocesorul, senzorii și componentele auxiliare).

Cel de-al șaselea capitol descrie tehnologiile utilizate pentru realizarea funcționalităților propuse ale sistemului.

Capitolul șapte prezintă scenariile de utilizare ale sistemului.

Capitolul opt prezintă dezvoltări ulterioare ce pot fi adăugate proiectului (ce îmbunătățiri i se pot aduce, ce funcționalități noi ar putea fi adăugate etc.)

Capitolul nouă sunt prezentate concluziile la care am ajuns în urma realizării acestui sistem.

Ultimul capitol al lucrării reprezintă bibliografia.

## **2. Problema abordată și soluții pentru rezolvarea acesteia**

### **2.1 Problema abordată**

Monitorizarea mai multor pacienți în cazul deficitului de personal medical poate reprezenta o adevărată problemă, ce poate duce la neglijarea pacienților și la apariția unor evenimente nedorite. Principala problemă abordată de această lucrare este necesitatea monitorizării permanente, inteligente și continue a unor parametri vitali ai corpului uman, precum și alertarea personalului medical în cazul detectării unor valori care nu respectă anumite praguri de normalitate ale acestora.

### **2.2 Soluții existente**

IoMT reprezintă un domeniu în plin proces de extindere, care revoluționează medicina prin numeroasele soluții pe care le cuprinde. Aceste soluții sunt concepute pentru a se axa pe monitorizarea anumitor seturi de parametri, având funcționalități diferite, în funcție de caz. Soluția propusă de mine este utilă în special persoanelor de vârstă a treia, în cazul cărora apar cel mai des problemele pe care aceasta le urmărește. RPM (Remote Patient Monitoring) este o metodă de monitorizare a sănătății pacienților de la distanță. În acest sens există o mulțime de soluții: glucometre pentru pacienții cu diabet zaharat, dispozitive de monitorizare a pulsului și a tensiunii arteriale, dispozitive de localizare a pacienților cu boli psihice ș.a.m.d.

O soluție existentă de tip RPM ce vizează monitorizarea pacienților la distanță este Current Health. Sistemul de monitorizare abordează același principiu ca al soluției dezvoltate de mine, acesta fiind format dintr-un dispozitiv de monitorizare ce va fi atașat pacientului și o aplicație destinată personalului medical. Dispozitivul va monitoriza mai mulți parametri vitali, printre care se numără și pulsul.

mHealth (mobile health) este un termen utilizat pentru dispozitivele mobile ce se folosesc de tehnologia wireless pentru monitorizarea sănătății. Acest concept ajută utilizatorii în prevenirea apariției unor afecțiuni, aceștia putând monitoriza de acasă evoluția anumitor parametri. Un avantaj pe care acest concept îl oferă este faptul că permite utilizatorilor monitorizarea sănătății de acasă, fără a mai fi nevoiți să vadă un doctor.

Una dintre soluțiile existente de acest tip ce permite monitorizarea anumitor parametri de acasă este aplicația Instant Heart Rate. Această aplicație oferă date precise despre rata pulsului, funcționând similar cu pulsometrele, bazându-se pe detectarea schimbărilor din deget pentru a furniza valori corecte ale ritmului cardiac. Aplicația se folosește de senzorii din telefon, pentru a măsura pulsul utilizatorul trebuind să își pună degetul pe camera telefonului. O altă soluție similară este Samsung Health. Această aplicație folosește senzorii integrați în telefonul mobil pentru măsurarea stresului, pulsului și a saturației oxigenului. Cele două aplicații nu sunt destinate utilizării profesionale, permițând însă monitorizarea de acasă a frecvenței cardiace.

## **2.3 Soluția propusă**

Soluția propusă este reprezentată de realizarea unui sistem inteligent de monitorizare a anumitor parametri vitali (puls, temperatură). Acest sistem va veni însoțit de o aplicație Android de monitorizare și alertare destinată personalului medical. De asemenea, se dorește detectarea unor deviații ale mișcării pacientului, pentru a putea determina eventualele situații în care acesta a căzut. În vederea acestui lucru, se va aborda domeniul IoT și se vor folosi anumiți senzori capabili să achiziționeze datele de interes pentru monitorizarea parametrilor specificați mai sus. Datele vor fi achiziționate în permanență de senzorii din care sistemul va fi compus și vor fi transmise mai departe folosind tehnologii de tip wireless (Wi-Fi) într-o bază de date în Cloud. Senzorii vor fi conectați în rețea prin intermediul unei plăci de dezvoltare care permite comunicația Wi-Fi. Din baza de date, informația legată de parametri monitorizați va fi prelucrată și transmisă aplicației Android. Aplicația va fi capabilă să trimită notificări în cazuri de urgență, alertând personalul medical, astfel putându-se evita agravarea anumitor situații.

De asemenea, se urmărește și problema autonomiei acestui sistem. În această privință, sistemul fiind unul demonstrativ (un prototip), se vor folosi componente (baterii) care să asigure autonomia acestuia în vederea realizării câtorva scenarii de utilizare.

Se dorește ca acest sistem să fie unul portabil, pentru a putea permite mobilitatea utilizatorului. Pentru acest lucru, se vor căuta componente de dimensiuni cât mai mici capabile să realizeze funcționalitățile propuse. Așadar, se va alege o plăcuță de dezvoltare de dimensiuni mici, senzori de dimensiuni reduse, rezultând un sistem inteligent de monitorizare cu o greutate mică și de mici dimensiuni.

## **2.4 Avantajele soluției propuse**

Un avantaj pe care soluția propusă de mine îl are în plus față de Health Care este detectarea căderilor pacienților. Sistemul monitorizează mișcarea pacientului, iar în cazul unei căderi alertează personalul medical. Un alt avantaj este dimensiunea redusă a dispozitivului, dar și costul scăzut.

În comparație cu Instant Heart Rate și Samsung Health un avantaj principal este reprezentat de faptul că pacientul trebuie să utilizeze doar sistemul de monitorizare, personalul medical fiind cel care se va ocupa de monitorizarea din aplicație. Un alt avantaj este integrarea mai multor senzori și a sistemului de alarmare în caz de urgență, spre deosebire de aplicațiile și sistemele ce vizează strict un parametru. De asemenea, sistemul de alarmare va fi personalizat special pentru persoanele de vârstă a treia, prin setarea pragurilor în mod special pentru aceștia.



### 3. Scopul lucrării și grupul țintă

Această lucrare urmărește realizarea unui sistem de monitorizare continuă și inteligentă a câtorva parametri ai corpului uman, folosind tehnologii din domeniul IoT. Sistemul va comunica cu o aplicație Android prin intermediul unei baze de date în timp real. Datele de interes vor fi transmise prin comunicație Wi-Fi către baza de date, de unde vor fi preluate de aplicație. Pentru a se realiza acest lucru este necesar ca dispozitivul utilizatorului să aibă o conexiune activă și funcțională la internet.

În continuare, se urmărește ca prototipul final să poată înlocui parțial (sau chiar total, în unele cazuri) prezența permanentă a unui personal medical de specialitate care să monitorizeze evoluția acestor parametri pe un ecran fix, aflat în același loc cu pacientul. Singurele acțiuni pe care personalul medical trebuie să le facă în monitorizarea acestor parametri ai pacientului sunt cele necesare în cadrul primirii notificărilor de urgență. Așadar, proiectul urmărește flexibilitatea în ceea ce privește scenariile de utilizare ale acestuia, precum și realizarea unui sistem de alertare în timp real în anumite cazuri speciale, ce vor fi predefinite în cadrul cerințelor funcționale.

Deoarece acest proiect va reprezenta un prototip, cu rol demonstrativ, scopul de a asigura autonomia acestui sistem se concentrează pe oferirea unei perioade de funcționare suficient de lungi pentru a realiza câteva scenarii de utilizare posibile. Monitorizarea în timp real a parametrilor achiziționați de senzori presupune accesul permanent la date, pentru a permite stabilirea unor praguri de alarmare în funcție de evoluția acestora în timp.

În vederea monitorizării, se va utiliza o aplicație Android care va permite vizualizarea grafică a datelor. Totodată se asigură flexibilitatea și accesibilitatea în ceea ce privește achiziția datelor, aceasta realizându-se în timp real. Evoluția parametrilor va fi constant actualizată prin intermediul bazei de date Cloud. Aplicația va fi capabilă să trimită notificări de alarmă la sesizarea unor parametri anormali (ce depășesc anumite praguri).

Proiectul se adresează în special persoanelor de vârstă a treia, în cazul cărora monitorizarea permanentă a anumitor parametri poate fi crucială. De multe ori, necesitatea monitorizării permanente a unui număr mare de astfel de pacienți poate reprezenta o problemă în cazul deficitului de personal. Lucrarea vine cu o soluție demonstrativă de monitorizare a câtorva parametri ce pot fi considerați foarte importanți pentru astfel de persoane.

De asemenea, sistemul poate fi utilizat și în cazul persoanelor mai tinere, chiar și pentru o scurtă consultare a celor doi parametri (puls și temperatură). Așadar, lucrarea propune o soluție flexibilă în ceea ce privește utilitatea acesteia. Monitorizarea folosind

accelerometrul poate fi și aceasta utilă în cazul persoanelor de orice vârstă, însă cea mai mare utilitate o are în cazul persoanelor în vârstă, scenariile în care acestea pot avea devieri bruște ale mișcării normale (căderi cauzate din diverse motive, în general specifice vârstei înaintate) fiind mult mai des întâlnite.

Aplicația se adresează personalului medical din clinicile de monitorizare a pacienților în vârstă. Aceasta va avea o interfață prietenoasă, ce va permite personalului medical următoarele: logarea în cont, adăugarea pacienților, vizualizarea paginii pacienților și monitorizarea acestora.

Din punctul de vedere al pacienților monitorizați, aceștia vor folosi doar dispozitivul de monitorizare, pe care îl vor avea asupra lor în timpul procesului de urmărire al parametrilor.

## 4. Etapele dezvoltării proiectului

Etapele parcurse în vederea realizării acestui proiect sunt următoarele:

- Identificarea cerințelor funcționale: descrierea funcționalităților dorite ale sistemului. În urma acestei etape va rezulta lista de cerințe.
- Proiectarea arhitecturii sistemului: în urma acestei etape va rezulta arhitectura sistemului (design hardware + software).
- Proiectare hardware: realizarea propriu-zisă a arhitecturii descrise la pasul anterior. Tot în cadrul acestei etape se vor scrie algoritmii utilizați de către partea hardware a sistemului în vederea implementării funcționalităților. În urma acestei etape va rezulta prototipul hardware.
- Proiectare software: Dezvoltarea aplicației ce va comunica cu prototipul hardware prin intermediul bazei de date. În urma acestei etape ne va rezulta un prototip intermediar al proiectului.
- Testarea funcționalităților: verificarea funcționării sistemului conform așteptărilor (achiziții de date corecte, prelucrarea și transmiterea datelor, sistemul de alarmare, vizualizarea datelor în aplicație, funcționarea corectă a aplicației). În urma acestei etape va rezulta prototipul final.

Etapele prezentate anterior, împreună cu rezultatele sunt prezentate și în figura 5.1.

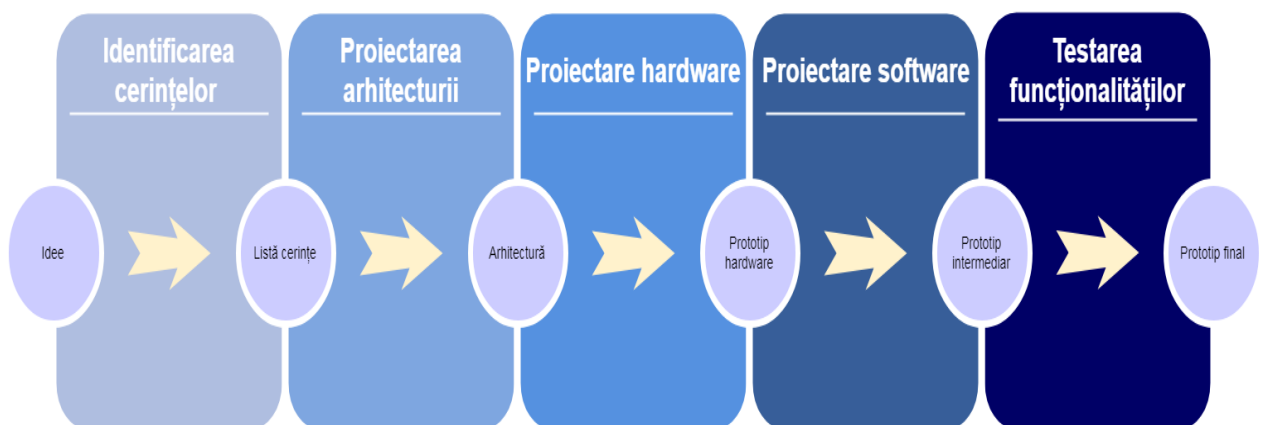


Figura 4.1 Etapele dezvoltării proiectului

## 4.1 Identificarea cerințelor funcționale

Cerințele funcționale sunt cele care definesc funcționalitățile sistemului/aplicației. Pe baza cerințelor funcționale se va realiza ulterior implementarea hardware și software.

- Dezvoltarea hardware a prototipului.
  - Dezvoltarea unui sistem de achiziție a datelor preluate de la senzori.
  - Transmiterea valorilor achiziționate de la senzori într-o bază de date prin intermediul unui modul Wi-Fi.
  - Dezvoltarea unui algoritm de detectare a căderilor.
  - Dezvoltarea unui algoritm de calcul al pulsului provenit de la semnalul senzorului.
- Dezvoltarea unei aplicații Android destinată personalului medical.
  - Crearea paginii principale a aplicației.
    - Implementarea butoanelor de logare și înregistrare pentru doctori.
  - Crearea paginii de login.
  - Crearea paginii de register.
  - Crearea paginii de selectare și adăugare pacient.
    - Implementarea butonului de adăugare pacienți pentru fiecare doctor.
    - Implementarea opțiunii de căutare a pacienților.
  - Implementarea paginii de vizualizare a datelor despre pacienți.
  - Implementarea funcționalității de notificare în caz de urgență.
  - Implementarea paginii pentru vizualizarea grafică a datelor .

## 4.2 Proiectarea arhitecturii sistemului

Arhitectura sistemului poate fi definită în urma analizării cerințelor funcționale. Aceasta definește felul în care componentele sistemului se vor comporta și vor interacționa între ele.

Astfel, în urma analizării cerințelor, am stabilit tehnologiile și componentele ce vor fi utilizate. Pentru partea hardware se va folosi ca mediu de dezvoltare Arduino IDE, iar ca placă de dezvoltare Node MCU, placă ce are un modul Wi-Fi integrat, ceea ce îl face ideal pentru transmiterea datelor. Datele vor fi transmise prin intermediul modulului Wi-Fi într-o bază de date. Pentru stocarea informațiilor se va utiliza o bază de date, Firebase, aceasta fiind o bază de date în timp real, informațiile stocându-se în Cloud. Baza de date oferă o serie de servicii ce vor fi detaliate în capitolul de tehnologii, utile în dezvoltarea aplicației. Mai apoi datele vor fi preluate, prelucrate și transmise aplicației. Pe partea software de dezvoltare a aplicației am ales utilizarea mediului Android Studio IDE, mediu ce oferă o serie de instrumente și funcționalități, ce se pliază pe implementarea cerințelor. Se va implementa și un sistem de alarmare cu ajutorul OneSignal, sistem ce va trimite notificări personalului medical. Interacțiunea dintre toate aceste componente se poate observa în următoarea figură (figura 4.2).

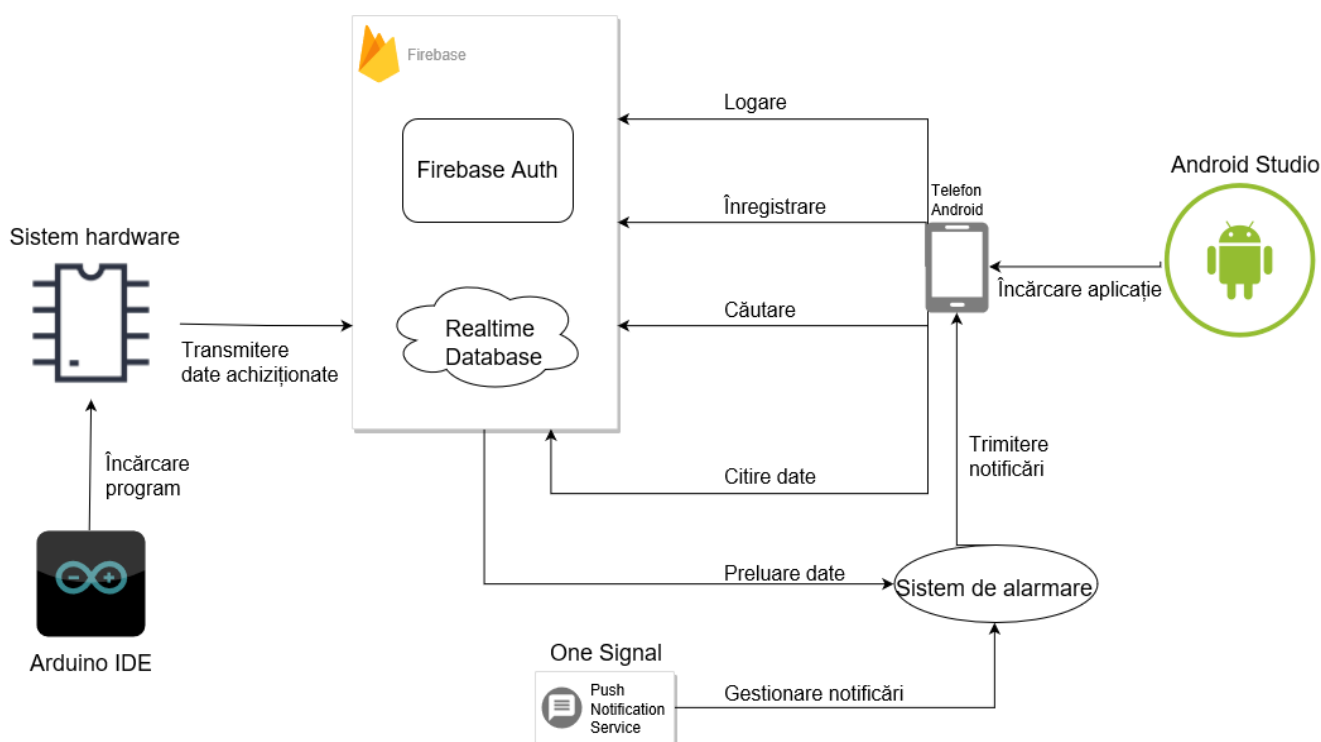


Figura 4.2.1 Arhitectura sistemului

### 4.3 Proiectare hardware

Pe partea de dezvoltare hardware am ales utilizarea unei plăcuțe de dezvoltare Node MCU, plăcuță ce vine cu un modul Wi-Fi integrat (ESP8266). Pentru partea de senzori am ales un accelerometru (MPU6050) pentru detectarea căderilor și un senzor de puls (XD58C) pentru achiziționarea datelor despre bătăile inimii. Aceste componente au dimensiuni și costuri reduse, conexiunea Wi-Fi ajutând în dezvoltarea sistemului IoT.

Am realizat o schemă de conexiune a componentelor, pe care am implementat-o (Figura 4.3.1), după care am testat funcționarea algoritmilor. Senzorii vor fi conectați direct la pinii plăcii de dezvoltare, iar plăcuța va fi în primă parte alimentată de la calculator.

Sistemul final va fi un dispozitiv portabil și autonom. Acest lucru se va realiza prin alimentarea sistemului la o baterie și prin dimensiunile reduse ale prototipului.

Ca program de dezvoltare am ales utilizarea mediului Arduino IDE, care oferă o serie de biblioteci și exemple utile. Am setat conexiunea cu Firebase, setând valorile pentru Host și Auth. De asemenea, am realizat conexiunea Wi-Fi setând ID-ul și parola.

Mai departe, în urma prelucrării datelor de la senzori prin intermediul algoritmilor implementați, acestea au fost colectate și trimise în Firebase prin conexiunea Wi-Fi setată.

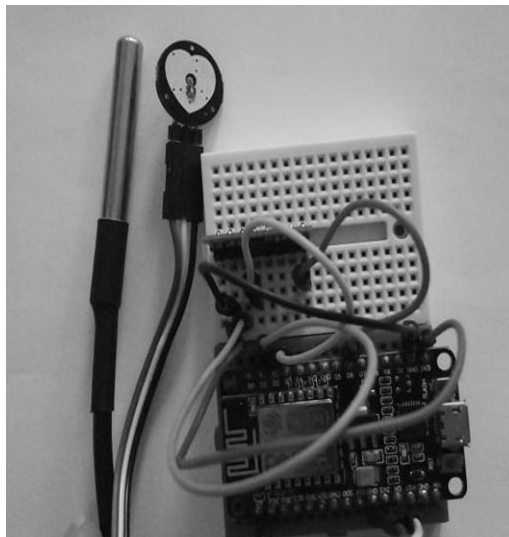


Figura 4.3.1 Schema hardware provizorie

Pentru accelerometru, se vor citi întâi datele acestuia din regiștri, date cu care se va lucra în continuare în cadrul algoritmului de detectare a căderii.

Algoritmul este bazat pe faptul că, în timpul unei căderi, o persoană experimentează o cădere liberă de moment sau o reducere a accelerației, urmat de o creștere bruscă a accelerației acesteia. Așadar, pentru a implementa acest lucru, se va folosi norma vectorului accelerației. După achiziționarea datelor de la senzori, se va face calibrarea axelor, reținând aceste valori în  $ax, ay, az$ . După care am calculat norma vectorului accelerație care are formula:

$Raw\_AM = \sqrt{\sqrt{ax^2 + ay^2 + az^2} \cdot 0.5}$ . Această valoare va fi convertită în  $m/s^2$  prin înmulțirea lui  $Raw\_AM$  cu o aproximație a lui  $g$  (accelerația gravitațională):  $AM = Raw\_AM \cdot 10$ . În timpul unei căderi libere valoarea lui  $AM$  tinde la 0, astfel ne vom seta un prag suficient de bun pentru aproximarea acestui fenomen de  $2 m/s^2$ . Ulterior verificăm dacă valoarea lui  $AM$  scade sub acest prag, dacă acest lucru se întâmplă vom incrementa  $trigger1count$ , trigger folosit pentru a detecta căderea liberă, altfel vom incrementa  $trigger2count$ , trigger ce detectează falsele căderi. Se verifică prin setarea unui prag, ales prin testări succesive, dacă valoarea lui  $trigger1count$  (de câte ori s-a declanșat trigger 1) este mai mare decât acel prag, în caz că acest lucru se întâmplă  $trigger1$  devine true și resetăm counter-ul pentru acesta, declanșarea acestui eveniment indicând detectarea unei căderi. Dacă numărul căderilor false (false positives),  $trigger2count$ , este mai mare decât acel prag vom seta  $trigger1count$  cu 1 și vom reseta  $trigger2count$ . La fiecare declanșare a trigger-ului pentru detectarea căderii se va trimite o valoare în baza de date. De fiecare dată când această valoare va fi primită se va trimite o notificare de alarmare a personalului medical de tipul : “Pacientul X a avut o pierdere a echilibrului ”.

O întrerupere este un semnal trimis către procesor ce indică tratarea urgentă a unui eveniment. Întreruperile îți permit o formă de paralelism atunci când vorbim de sisteme embedded, acestea ajutând la prioritizarea activităților. Pentru a realiza achiziția și transmiterea datelor în același timp către baza de date am utilizat întreruperile. O întrerupere externă setată pe un pin și o întrerupere în cod bazată pe timer.

Rata pulsului reprezintă numărul de bătăi ale inimii percepute într-un minut. Această valoare este influențată de mai mulți factori, cum ar fi condiția fizică și vârsta. Rata medie a pulsului pentru majoritatea adulților, inclusiv pentru persoanele în vârstă este cuprinsă între 60 și 100 bpm. Aceste valori diferă, în cazul copiilor fiind mai ridicate, iar în cazul persoanelor sportive fiind mai scăzute. Atunci când apar valori ale pulsului peste/sub limitele normale, acest lucru semnalează prezența unor afecțiuni (tahicardie/bradicardie). Limita superioară în cazul adulților și a persoanelor în vârstă este de 100 bpm, iar limita inferioară este de 60 bpm. Atât tahicardia, cât și bradicardia pot indica prezența altor

afecțiuni ce necesită evaluare medicală și monitorizare. Dacă sunt netratate pot duce la apariția unor complicații de sănătate.<sup>2</sup>

Pentru prevenirea acestor afecțiuni și apariția altor complicații se va monitoriza rata pulsului pentru pacienți. În acest sens am setat două praguri de valori, unul superior și unul inferior. Pragul superior va fi de 100 bpm, la apariția unor valori repetate ale pulsului peste acest prag, personalul medical va fi alertat printr-o notificare de tipul “Pacientul are o valoare ridicată a pulsului.”. Pragul inferior va fi de 60 bpm, la apariția unor valori repetate sub acest prag, personalul medical va fi alertat printr-o notificare de tipul : “Pacientul are o valoare scăzută a pulsului”.<sup>3</sup>

În vederea obținerii unei rate precise a pulsului am dezvoltat un algoritm de calcul. Acest algoritm va fi implementat în metoda `calculate_bpm()`. Mai întâi vom reține într-un buffer ultimele `N_DATA` valori ale semnalului preluat de la senzor. Parcurgând vectorul de valori vom aduna valorile, după care le vom împărți la câte sunt pentru a obține o medie. În funcție de această medie se va seta o limită inferioară și una superioară (`threshold_value_rise`, `threshold_value_fall`), limitele fiind direct proporționale cu media. Acestea au fost setate prin testări succesive pentru a obține o mai bună acuratețe a valorilor obținute. Ulterior, vom parcurge vectorul de valori pentru a verifica dacă valorile depășesc limitele setate, dacă valoarea depășește limita superioară atunci semnalul este în creștere, altfel dacă depășește limita inferioară semnalul este în scădere. Se vor reține ultimele 2 peak-uri ale semnalului, acestea ajutându-ne în calcularea `avg_bpm`. Formula finală pentru calcularea `avg_bpm` :  $60 / ((\text{peak\_curent} - \text{peak\_anterior}) * \text{nr\_eșantioane})$ . Am ales arbitrar valoarea `nr_eșantioane` ca fiind 0.02. După aceea peak-ul anterior va deveni peak-ul current și se va incrementa numărul de peak-uri. Semnalul final va avea valoarea lui `avg_bpm` împărțit la numărul de peak-uri. Această valoare va fi transmisă în baza de date, de unde va fi preluată și comparată cu valorile de alarmare pentru a trimite notificări în cazuri de urgență. De asemenea, se va putea observa evoluția valorilor prin intermediul unui grafic.

---

<sup>2</sup> <https://www.healthline.com/health/dangerous-heart-rate?fbclid=IwAR3ODFVIMQRpsimPdPCIm5jYe49P2-g6RwoT1caJx2Q3BAzllanINrx3JQ#the-takeaway>

<sup>3</sup> Guyton A. , Hall J., (2007), *Tratat de fiziologie a omului*, Editura Medicala Callisto.



## 4.4 Proiectare software

Pentru proiectarea software am folosit mediul de dezvoltare Android IDE pentru a dezvolta o aplicație mobilă. Pentru stocarea datelor din aplicație am utilizat platforma Firebase, care oferă o bază de date în timp real, iar pentru autentificare serviciul FirebaseAuth oferit de aceeași platformă.

Pentru partea de backend am creat următoarea ierarhie de clase Java (figura 4.4.1). Clasele se află în pachetul `com.example.patientmonitorization`. Aceste clase implementează funcționalitățile din spatele design-ului.

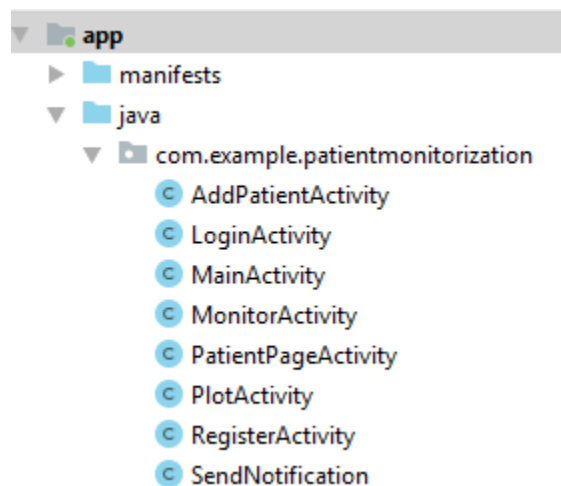


Figura 4.4.1 Ierarhie clase Java

Pentru partea de frontend am creat următoarele fișiere XML (figura 4.4.2), fișiere ce conțin partea de design a aplicației. Fiecărui fișier xml îi corespunde o clasă java.

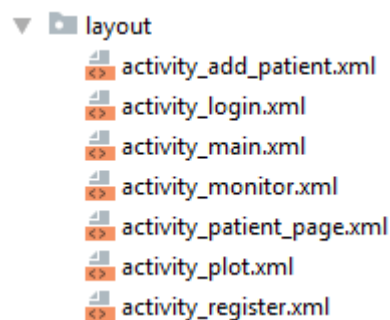


Figura 4.4.2 Fișiere XML

Clasei MainActivity îi corespunde pagina de pornire a aplicației. Această pagină va avea 2 butoane, prin care utilizatorul poate să își aleagă activitatea dorită: logare sau înregistrare. Fișierul activity\_main conține design-ul paginii. Pagina conține două elemente grafice de tip button și un TextView. De asemenea, orientarea va fi de tip RelativeLayout, pentru poziționarea elementelor grafice oriunde în pagină. Suprascrierea metodei onCreate este implicit inclusă în fiecare activitate, iar în cadrul acesteia sunt definite toate acțiunile din pagina respectivă. Legătura între un element vizual și acțiunea din spatele lui se face prin metoda findViewById, apelată pentru o instanță a obiectului creat, ce primește ca parametru id-ul obiectului vizual din layout. Pentru definirea acțiunii unui buton, i se atașează acestuia un listener de tipsetOnClickListener, în cadrul căruia se va suprascrise metoda onClick, în cadrul căreia se definește funcționalitatea dorită. În cazul butoanelor de Login și Register se instanțiază un obiect de tip Intent, ce primește ca parametri activitatea din care pornește și activitatea care se va deschide mai departe. Asupra acestui obiect se va apela metoda startActivity, care face tranziția propriu zisă dintre parametri intent-ului.

Clasei LoginActivity îi corespunde pagina de logare a aplicației. Această pagină va avea un buton, un TextView și două EditText-uri pentru introducerea datelor de logare, acestea fiind definite în activity\_login.xml. În suprascrierea metodei onCreate sunt definite toate acțiunile butoanelor din cadrul acestei pagini. Cele două EditText-uri vor fi instanțiate, utilizând metoda findViewById, ce va avea ca parametru Id-ul elementului definit în fișierul xml. Acțiunea de tipsetOnClickListener pentru Login verifică text-ul introdus în câmpurile EditText. În cazul în care câmpurile sunt goale se afișează un mesaj de tip Toast, care ne spune că datele de logare sunt incomplete. Altfel, dacă câmpurile coincid cu un cont valid din FirebaseAuth se va crea o nouă sesiune de autentificare și se va trece pe pagina MonitorActivity. Dacă datele introduse sunt greșite se va afișa mesajul: "Eroare de logare".

Clasei RegisterActivity îi corespunde pagina de înregistrare din aplicație. Această pagină conține un TextView, cinci elemente de tip EditText și un buton. Definirea acestor elemente grafice se face în fișierul activity\_register.xml. În suprascrierea metodei onCreate sunt instanțiate elementele grafice și este definită metoda pentru apăsarea butonului de Register. Acțiunea de tipsetOnClickListener verifică câmpurile introduse, în cazul în care unul dintre acestea este gol se afișează un mesaj de eroare. Altfel, se crează un utilizator nou în FirebaseAuth cu email-ul și parola introduce și se salvează datele acestuia de înregistrare la o referință către id-ul său în Realtime Database.

Clasa MonitorActivity este pagina în care utilizatorul (personalul medical) va fi redirecționat după logare. În această pagină utilizatorul are posibilitatea de a adăuga sau de a căuta un pacient. În activity\_monitor.xml avem două elemente de tip TextView, două elemente de tip Button și un EditText. În metoda onCreate a clasei java se vor defini funcțiile pentru cele două butoane de căutare, respectiv adăugare. Pentru butonul de adăugare în definirea metodeisetOnClickListener se face trecerea din pagina curentă în

pagina de adăugare pacient (AddPatientActivity). Pentru butonul de căutare a unui pacient, acțiunea acestuia la onClick va prelua CNP-ul introdus în câmpul EditText. Dacă câmpul introdus este gol se va genera un mesaj de eroare, altfel se vor căuta informațiile despre pacient în baza de date prin metoda onDataChange. Dacă pacientul este găsit se va trece la următoarea activitate, prin instanțierea unui obiect de tip intent și se va apela metoda putExtra asupra acelei instanțe. Această metodă se apelează pentru a trimite mai departe valori ce sunt necesare în cadrul activității spre care se face trecerea, de unde acestea vor fi preluate după tag-ul setat pentru fiecare valoare (prin metoda getExtra). Altfel se va afișa mesajul: “Pacientul nu a fost găsit!”. Din această metodă prin apăsarea butonului de căutare se va trece la pagina pacientului (PatientPageActivity).

Clasa AddPatientActivity corespunde paginii de adăugare pacient. Fișierul activity\_add\_patient.xml conține: un TextView, șase EditText-uri, un buton și un spinner. Acțiunea setOnClickListener a butonului de adăugare pacient verifică câmpurile introduse în EditText. Dacă unul dintre acestea este nul se va afișa un mesaj de eroare, altfel se crează un pacient nou în baza de date, corespunzător medicului care l-a adăugat. Pentru spinner am adăugat în fișierul strings.xml elementele sub forma unui array pentru definirea acestuia. Funcționalitatea acestei pagini este asemănătoare cu cea a paginii de Register a unui doctor, cu excepția faptului că pentru un pacient nu se va folosi modulul FirebaseAuth, deoarece un pacient nu este înregistrat ca utilizator, ci doar stocat ca referință în cadrul unui doctor.

Clasa PatientPageActivity corespunde paginii de monitorizare a pacientului. În această pagină vor putea fi vizualizate date despre pacientul căutat. După ce un doctor deschide această pagină, acesta devine abonat la notificările despre pacientul respectiv. Fișierul activity\_patient\_page.xml conține partea vizuală ce stă în spatele paginii. În această pagină avem elemente de tip TextView în care vor fi afișate date despre pacient și un buton care va redirecționa utilizatorul spre pagina de vizualizare grafic. Valorile sunt preluate cu ajutorul metodei getStringExtra. La apăsarea butonului de monitorizare se va trece la pagina PlotActivity prin intermediul unui intent.

Clasa PlotActivity reprezintă pagina de vizualizare grafică a aplicației. Acestei clase îi corespunde fișierul activity\_plot.xml. În acest fișier avem elemente de tip TextView și graphView. În clasa java se instanțiază un obiect de tip graphView și se setează pentru el anumite proprietăți. Graficul va plota pe axa X puncte, în funcție de valorile citite din baza de date în timp real, ce vor fi unite.

La vizualizarea paginii unui pacient, personalul medical este automat abonat la notificările despre acel pacient. Datele sunt trimise prin Wi-Fi în baza de date, de acolo prin apelarea metodei onDataChange asupra referinței, se verifică dacă valorile respectă anumite praguri, în cazul în care acest lucru nu se întâmplă se va trimite o notificare. Notificarea se va trimite folosind metoda SendNotification, folosind serviciul OneSignal.

Structura bazei de date înregistrate în Firebase Realtime Database este următoarea:

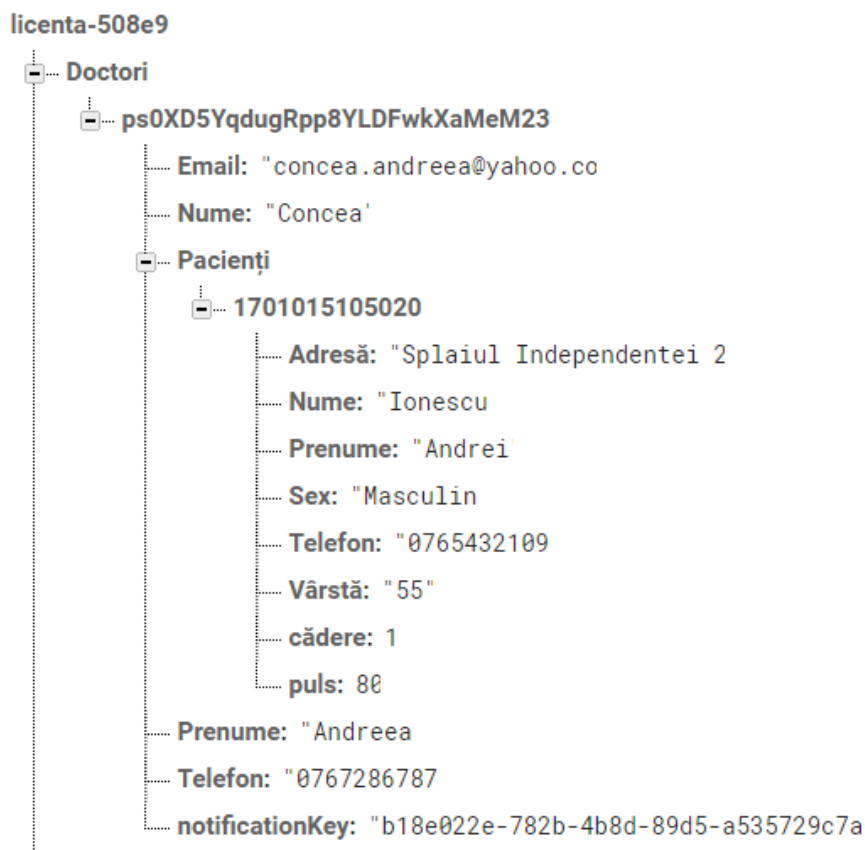


Figura 4.4.3 Structură bază de date

Baza de date va conține înregistrări despre doctori, cum ar fi: nume, prenume, email, număr de telefon și pacienții acestuia. Despre pacienți se vor stoca următoarele date: nume, prenume, adresă, telefon, sex, vârstă și câmpurile transmise din Arduino: puls și cădere. Pe baza acestor două câmpuri se vor transmite notificări de alarmare utilizatorului în cazuri de urgență.

## 4.5 Testarea funcționalităților

În cadrul acestei etape am verificat funcționarea corectă a cerințelor implementate. Testarea a fost realizată pentru a verifica îndeplinirea obiectivelor, dar și pentru a identifica și rezolva eventuale probleme ce pot să apară.

Pentru testarea aplicației mobile am utilizat telefoane mobile cu versiunea 8.0 de Android. Testarea a fost realizată simulând diferite scenarii de utilizare. Au fost accesate toate paginile aplicației și au fost testate funcționalitățile tuturor butoanelor din paginile respective.

Pentru pagina de Login am verificat funcționalitatea corectă pentru conturile deja create. De asemenea, am tratat cazul în care datele sunt incomplete sau incorecte și în acest caz am afișat un mesaj de eroare. Mai departe, pentru înregistrare am verificat crearea conturilor în baza de date și mai apoi logarea cu acestea. Ca și în cazul login-ului am tratat cazul în care datele sunt incomplete și am generat un mesaj de eroare.

Pentru pagina principală a personalului medical am verificat funcționalitățile butoanelor de căutare și adăugare pacient. Pentru adăugare s-a testat, la fel ca și în cazul înregistrării, crearea pacienților în baza de date sub ID-ul doctorului curent. Pentru butonul de căutare s-a verificat funcționarea corectă, în cazul introducerii unui ID care nu se află în baza de date se va genera un mesaj de eroare.

Pentru pagina de monitorizare a pacientului a fost verificat faptul că datele afișate sunt în concordanță cu cele stocate în baza de date. Am verificat funcționalitatea butonului de monitorizare, care redirecționează spre o nouă pagină, unde se face vizualizarea grafică a datelor.

Pentru pagina de vizualizare grafică am testat funcționalitatea graficului prin trimiterea de date, mai întâi de la tastatură, iar mai apoi de la senzori, verificând comunicația sistemului hardware cu baza de date.

Sistemul de alarmare a fost testat prin simularea unor situații nedorite. Aceste situații au fost simulate trimițând valori de la tastatură în baza de date, valori ce depășesc pragurile de normalitate setate.

De asemenea, a fost testată și funcționarea sistemului hardware. Pentru început am verificat funcționarea corectă a senzorilor și a valorilor afișate. Ulterior, am verificat conexiunea Wi-Fi, trimițând datele colectate de la senzori în baza de date.

## 5. Componente utilizate

### 5.1 Node MCU

Ca placă de dezvoltare am utilizat NodeMCU (figura 5.1.1), placă ce vine cu un modul pentru WIFI (ESP8266) deja incorporat. Această placă este un circuit integrat, ușor de utilizat, ce vine la pachet cu module: GPIO, PWM, I2C, 1-Wire și ADC. Kit-ul de dezvoltare conține un modul ESP-12E, un regulator de tensiune și o interfață pentru USB.

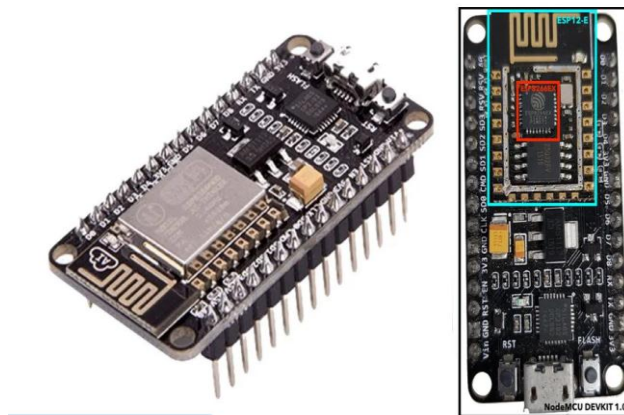


Figura 5.1.1 NodeMCU

Modulul ESP-12E a fost creat de AI-THINKER. Acesta este format dintr-un modul ESP8266, ce este acoperit de o carcasă metalică. Schema modului se regăsește în figura de mai jos (figura 5.1.2).



Figura 5.1.2 ESP-12E

Modulul ESP8266 este produs de Espressif, acesta are Wi-Fi integrat și consumă foarte puțină energie. Este utilizat des în aplicații IoT. Modulul se regăsește în figura de mai jos

Modulul are un pin analaogic A0 și 9 pini digitali (D0, D1, D2, D3, D4, D5, D6, D7, D8, D9). Conține în plus 6 pini GPIO, care pot fi programați ca PWM, I2C, 1-Wire, excepție făcând pinul GPIO16 (D0), care poate fi folosit doar pentru citire/scriere. Suportă alimentare atât la 3V3, cât și la 5V.<sup>4</sup>

Descrierea tuturor pinilor poate fi regăsită în următoarea schemă (figura 5.1.3).

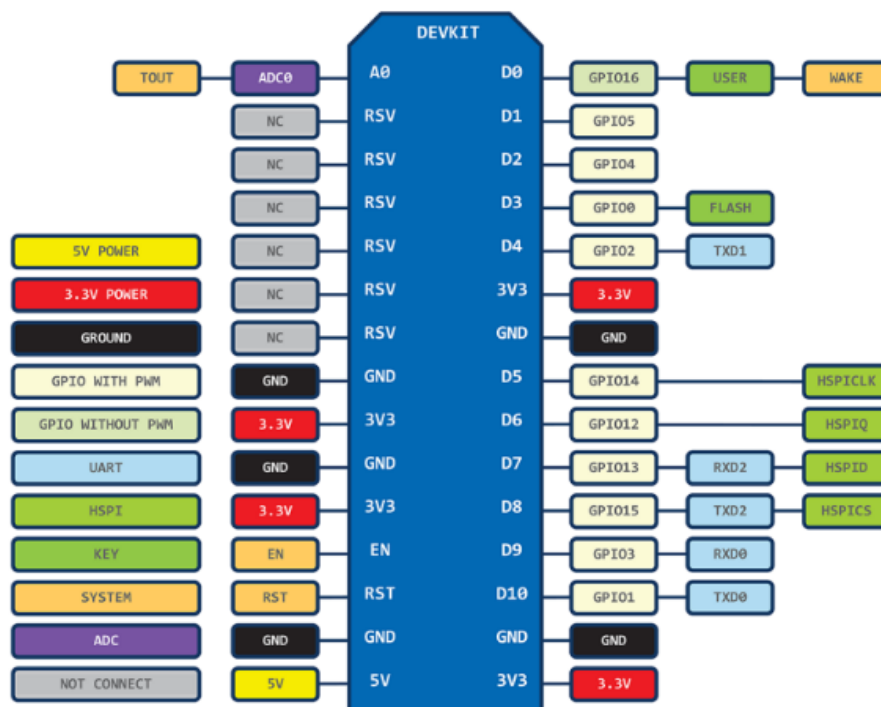


Figura 5.1.3 Descrierea pinilor

Am ales să utilizez acest modul datorită avantajelor pe care le oferă :

- Cost mic;
- Dimensiuni reduse;
- Modul Wi-Fi integrat;
- Consum redus de energie.

<sup>4</sup> [https://www.handsonetec.com/pdf\\_learn/esp8266-V10.pdf](https://www.handsonetec.com/pdf_learn/esp8266-V10.pdf)

## 5.2 Senzor de puls XD58C

Senzorul de puls XD-58C (figura 5.2.1) funcționează împreună cu o placă de dezvoltare ce are cel puțin un pin analog. Acesta senzor ne ajută în colectarea și prelucrarea datelor despre bătăile inimii.

Caracteristici tehnice:

- Tensiune de alimentare: 3V3, 5V;
- Curent: 4mA, 5V;
- Diametru: 16mm;
- Grosime: 3mm ;



Figura 5.2.1 Senzor de puls XD58C

Senzorul poate fi pus pe deget sau pe lobul urechii pentru a măsura pulsul. Acesta se bazează pe reflecția lumini emisă de LED din senzor. Este ușor de utilizat și se găsește în multe echipamente ce măsoară bătăile inimii.

Senzorul are 3 pini, unul pentru Vcc, unul pentru GND și unul pentru achiziționarea datelor. Schema de conectare este prezentată mai jos (figura 6.2.2).<sup>5</sup>



Figura 5.2.2 Schema de conectare XD58C

<sup>5</sup> [https://www.optimusdigital.ro/ro/senzori-altele/1273-senzor-de-puls-xd-58c.html?search\\_query=senzor+de+puls&results=26](https://www.optimusdigital.ro/ro/senzori-altele/1273-senzor-de-puls-xd-58c.html?search_query=senzor+de+puls&results=26)



## 5.3 Senzor de temperatură DS18B20

Senzorul de temperatură DS18B20 este unul ce asigură o performanță ridicată, oferind o precizie de până la 12 biți. De asemenea, acesta este rezistent la apă.

Caracteristici tehnice:

- Tensiune de alimentare: 3V3, 5V;
- Curent: 1.5mA;
- Interfață de comunicație: 1 Wire;
- Interval de temperatură: -10 C - +85 C;
- Rezoluție: de la 9 biți până la 12 biți.



Figura 5.3.1 Senzor DS18B20

În schema anterioară (figura 5.3.1) sunt prezentați pinii senzorului: unul pentru Vcc, unul pentru GND și unul pentru colectarea informației de interes, temperatura. Modul în care se conectează este prezentat în figura 5.3.2.<sup>6</sup>

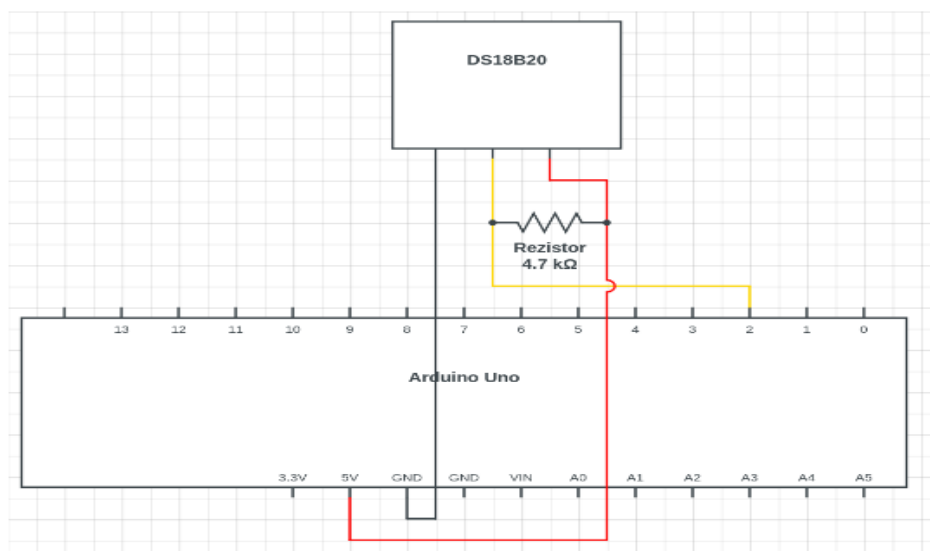


Figura 5.3.2 Schmă de conectare senzor DS18B20

<sup>6</sup> [https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/586-senzor-de-temperatura-rezistent-la-apa.html?search\\_query=senzor+de+temperatura&results=220](https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/586-senzor-de-temperatura-rezistent-la-apa.html?search_query=senzor+de+temperatura&results=220)

## 5.4 Modul accelerometru și giroscop cu 3 axe MPU6050

Acest modul (figura 6.4.1) are un circuit integrat un accelerometru, giroscop și un senzor de temperatură. Oferă o comunicație de tip I2C, conexiunea fiind realizată de obicei pe plăcuța modului.

Caracteristici tehnice

- Tensiune de alimentare: 3V3, 5V;
- Tensiune magistrală I2C: 3V3;
- Curent: 5mA;

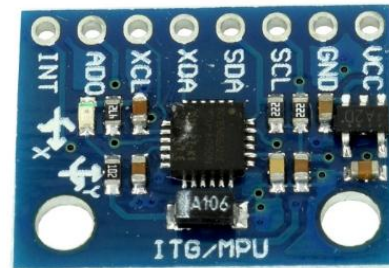


Figura 5.4.1 Modul accelerometru și giroscop

Acest modul este util pentru detectarea mișcării și a intensității ei.

Schema de conectare este prezentată în următoarea figură (figura 5.4.2).<sup>7</sup>

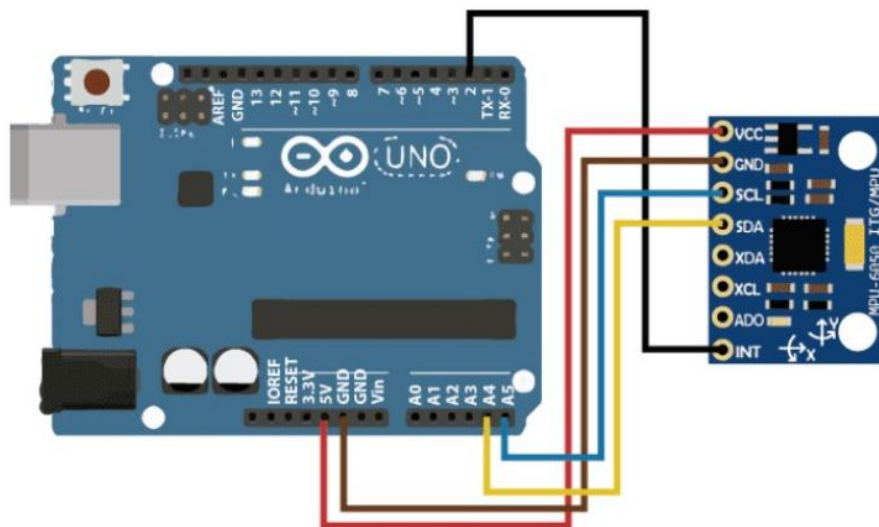


Figura 5.4.2 Schemă de conectare modul accelerometru și giroscop

<sup>7</sup> [https://www.optimusdigital.ro/ro/senzori-senzori-inertiali/96-modul-senzor-triaxial-mpu-6050.html?search\\_query=accelerometru+mpu6050&results=6](https://www.optimusdigital.ro/ro/senzori-senzori-inertiali/96-modul-senzor-triaxial-mpu-6050.html?search_query=accelerometru+mpu6050&results=6)

## 5.5 Componente auxiliare

Pentru conectarea componentelor de bază am utilizat ca și elemente auxiliare următoarele:

- Breadboard;
- Fire;
- Baterie.

## 6. Tehnologii utilizate

### 6.1 Arduino IDE

Arduino este o platformă open-source destinată atât dezvoltării hardware, cât și software, în vederea realizării proiectelor electronice. Programele Arduino pot fi scrise în orice limbaj ce are un compilator capabil să producă cod mașină în binar. Arduino IDE este o platformă folosită pentru a scrie și încărca programe pe plăcuțe compatibile cu Arduino. Acest mediu de dezvoltare este o aplicație cross-platform și este scris în Java. Mediul Arduino IDE oferă suport pentru limbajele de programare C și C++ și folosește reguli speciale de structurare a codului. De asemenea, oferă o bibliotecă software Wiring, în care se regăsesc o multitudine de proceduri de intrare și ieșire. Acest IDE include un editor de cod cu funcționalități de bază, precum: copiere, decupare, lipire de text, căutare și înlocuire, indentare automată etc. De asemenea, include mecanisme simple de compilare și încărcare a codului pe o plăcuță compatibilă cu Arduino, o zonă de mesaje, o consolă text, un toolbar cu butoane pentru funcții de bază ale IDE-ului.

Un program (sketch) este format din două funcții de bază, ce sunt obligatorii în orice sketch scris de către un utilizator: `setup()` și `loop()`, ce sunt integrate în main. Orice program este salvat în computer-ul dezvoltatorului ca un fișier text cu extensia `.ino`. Metoda `setup()` este rulată o singură dată la începutul programului, iar în interiorul acesteia se realizează inițializările. Metoda `loop()` este o funcție apelată de mai multe ori, până la oprirea alimentării plăcuței.

Majoritatea plăcuțelor de dezvoltare compatibile cu Arduino IDE au integrate un LED și o rezistență limitatoare de curent, care este conectată între pinul 13 și ground, ce reprezintă baza pentru multe teste și funcționalități ale programelor. Un program tipic pentru începători, similar cu arhicunoscutul program “Hello, World!”, este “blink”, care are funcția de a clipi în continuu LED-ul integrat pe plăcuța de dezvoltare. Acesta folosește funcțiile `pinMode()`, `digitalWrite()` și `delay()`, care sunt conținute în bibliotecile interne ale IDE-ului Arduino. Toate comenzile din acest IDE sunt case sensitive. Codul principal al unei aplicații, sketch-ul, care este creat pe platforma IDE, va genera un fișier sursă de tip Hex (hexazecimal), care va fi mai departe transferat și încărcat pe microcontrollerul aflat pe placa de dezvoltare.

Fiind o platformă open-source, Arduino IDE are un număr mare de biblioteci software gratuite, ce pot fi folosite de orice utilizator în proiectele sale pentru adăugarea de noi funcționalități. O bibliotecă inclusă va apărea la începutul unui program, și va fi de forma `#include <NUME.h>`. Majoritatea bibliotecilor sunt preinstalate și vin la pachet cu software-ul Arduino.

Pentru a face referință către anumiți pini ai plăcii de dezvoltare, se folosesc comenzile `digitalRead` și `digitalWrite`, cu ajutorul cărora un pin se poate seta ca fiind de tip input, respectiv output.

Tipurile de funcții pentru a controla o placă de dezvoltare compatibilă cu acest IDE și pentru a realiza anumite calcule, sunt: Digital I/O, Analog I/O, Advanced I/O, Zero, Due & MKR Family, Time, Math, Trigonometry, Characters, Random Numbers, Bits and Bytes, External Interrupts, Interrupts, Communication, USB. Tipurile de date cu care se poate lucra în acest IDE sunt: String, array, bool, boolean, byte, char, double, float, int, long, short, size\_t, string, unsigned char, unsigned int, unsigned long, void, word. Structurile de control cu care se poate lucra din IDE-ul Arduino sunt: break, continue, do...while, if, else, for, return, while, switch...case și goto. Se pot utiliza operatori aritmetici, de comparare, binari, pointeri, operatori pe biți și operatori compuși. Variabilele se pot declara ca fiind const, static sau volatile.

Arduino beneficiază de mult suport din partea comunităților online, precum și de documentații bogate. Atunci când se întâlnește o problemă neașteptată în rularea unui program, o soluție foarte eficientă este căutarea online a soluțiilor, deoarece există multe forumuri și sunt șanse mari ca o persoană să fi avut fix aceeași problemă înainte, pe care a rezolvat-o prin intermediul unui forum, cu ajutorul comunității. În cazul în care nu se găsesc rezultate, se pot crea oricând discuții noi pe un forum, în vederea rezolvării unei probleme, cu ajutorul comunității. Așadar, acest IDE poate fi considerat ca fiind “prietenos” pentru utilizatori, beneficiind de mult suport tehnic, precum și de documentații.<sup>8</sup>

## 6.2 Wi-Fi

Wi-Fi reprezintă o familie de tehnologii radio, ce este utilizată în mod frecvent pentru a crea conexiuni de tip WLAN (wireless local area networking) între mai multe dispozitive. Printre dispozitivele care se pot conecta în rețea, utilizând tehnologiile Wi-Fi se numără: smartphone-urile, calculatoarele, laptop-urile, imprimantele, mașinile etc. Printre acestea se află și plăcile de dezvoltare cu module Wi-Fi încorporate, deci și plăcuța de dezvoltare folosită în cadrul acestui proiect, Node MCU, care vine cu modulul Wi-Fi ESP8266 încorporat în construcția de bază a acesteia. În prezent se produc din ce în ce mai multe tipuri de device-uri capabile să se conecteze prin intermediul Wi-Fi, dispozitive ce sunt numite “Smart”, și care folosesc diverse microcontrollere.

---

<sup>8</sup> <https://ro.wikipedia.org/wiki/Arduino>

Dispozitivele compatibile se pot conecta între ele prin intermediul unui punct de acces fără fir. Un astfel de punct de acces (denumit hotspot) are o anumită rază de acoperire în care permite dispozitivelor să se conecteze. Avantajul utilizării Wi-Fi față de Ethernet este faptul că oferă mobilitate și flexibilitate, astfel putându-se dezvolta numeroase dispozitive capabile să se conecteze la Internet fără a genera costuri suplimentare pentru cablarea lor în rețea (prin Ethernet), dispozitive ce pot fi ulterior mutate cu ușurință și utilizate fără a avea acea restricție generată de cablurile de conectare, restricție ce poate fi de multe ori incomodă în utilizarea dispozitivelor de monitorizare de la distanță, cum este în cazul acestui proiect. Orice modul ESP8266 poate opera ca o stație (dispozitiv care se conectează la WiFi) sau ca un access point pentru alte dispozitive. Controlul WiFi prin intermediul NodeMCU se poate face printr-o multitudine de funcții și proceduri predefinite.

Această tehnologie se dovedește a fi utilă în dezvoltarea sistemelor de monitorizare și control, astfel se pot construi rețele de senzori, rețele ce au posibilitatea de a comunica prin intermediul Wi-Fi. De aici rezultă un nou concept, cel de Internet of Things, ce va fi descris în cele ce urmează.<sup>9</sup>

## 6.3 IoT

Internet of Things, abreviat IoT, reprezintă un concept simplu, care se bazează pe conectarea tuturor “lucrurilor” (senzori, dispozitive electronice, obiecte din viața de zi cu zi) din lume la Internet. Aceste dispozitive, care în conceptul de IoT sunt numite lucruri, pot comunica și interacționa cu alte dispozitive, prin Internet, acestea putând fi monitorizate și controlate de la distanță. Definiția IoT devine din ce în ce mai stufoasă atunci când ne gândim la integrarea tehnologiilor precum: analiza în timp real, machine learning, sisteme embedded ș.a.m.d. Ariile tehnologice pe care le acoperă sistemele embedded, și anume: rețelele de senzori wireless, sistemele de control și automatizare, contribuie la răspândirea conceptului de Internet of Things. Pe piața actuală, tehnologia IoT este aproape sinonimă cu capacitatea unui produs de a fi “smart”.

O mare parte a lucrurilor din IoT sunt concepute pentru utilizatorii de zi cu zi, cum ar fi mașinile inteligente, casele inteligente, grădinile automatizate etc. Prin intermediul unei rețele de senzori se pot achiziționa date, pe baza cărora ulterior se poate face controlul parametrilor de interes, ceea ce conferă acestor lucruri capacitatea de a fi inteligente.

Internet of Medical Things (sau Internet of Health Things) reprezintă o ramură a IoT ce se adresează medicinei, ce constă în colectarea datelor pentru analiză, cercetare și monitorizare. Dispozitivele IoT pot fi folosite pentru a permite monitorizarea de la distanță

---

<sup>9</sup> <https://ro.wikipedia.org/wiki/Wi-Fi>

a stării de sănătate a unui pacient, precum și pentru un sistem de alarmare în caz de urgență (în cazul în care parametrii monitorizați depășesc sau scad sub valorile limită impuse în cadrul aceluși sistem de monitorizare). Așadar, IoT oferă o perspectivă nouă în ceea ce privește monitorizarea unui pacient, perspectivă ce urmărește suplinirea personalului medical din această arie. Dispozitivele de monitorizare a sănătății pot urmări de la anumiți parametri ai corpului uman, până la anumite implanturi speciale (de la un dispozitiv care monitorizează tensiunea sau pulsul unui pacient s-a ajuns până la un dispozitiv care monitorizează un implant de stimulator cardiac). Anumiți senzori specializați pot fi utilizați pentru a monitoriza sănătatea persoanelor de o vârstă înaintată (urmărirea în timp real a parametrilor acestora, precum puls, temperatură, tensiune etc.). De asemenea, se poate monitoriza și evoluția mișcării acestor pacienți, pentru a determina dacă aceștia se deplasează într-o manieră normală (se pot detecta eventualele căderi ale acestor pacienți, care pot fi utilizate mai departe ca semnale de alarmă pentru personalul care se ocupă de aceștia). Acești senzori creează o rețea inteligentă, care este capabilă să colecteze, proceseze, transfere și analizeze informațiile achiziționate (parametrii monitorizați) în diferite medii (se pot conecta astfel de dispozitive de monitorizare de acasă la un sistem al unui spital). De asemenea, există și dispozitive destinate utilizării zilnice, pentru a încuraja un regim sănătos de viață (brățări care monitorizează pulsul). Datorită evoluției tehnologiilor de fabricație a dispozitivelor electronice, se pot crea sisteme IoMT cu costuri reduse semnificativ. IoMT permite doctorilor, pacienților și tuturor persoanelor implicate să facă parte dintr-un sistem, unde istoricul pacienților este salvat într-o bază de date, permițând astfel accesul personalului medical la informații legate de pacienți. Monitorizarea de la distanță este posibilă utilizând conexiuni ce folosesc soluții wireless performante. Această conexiune permite personalului care monitorizează anumiți parametri să aplice algoritmi complecși pentru analiza stării de sănătate a unui pacient.<sup>10</sup>

## 6.4 Firebase

Firebase este o platformă ce oferă o varietate mare de servicii și tool-uri, ce este utilizată în dezvoltarea aplicațiilor mobile și web. Serviciile oferite se împart în două mari categorii:

- De dezvoltare și testare a aplicației: Real Time Database, Auth, Test Lab, Crashlytics, Cloud Functions, Firestore, Cloud Storage, Performance Monitoring, Crash Reporting și Hosting.
- De păstrare și creștere a audienței: Firebase Analytics, Invites, Cloud Messaging, Predictions, AdMob, Adwords, Dynamic Links, Remote Config și App Indexing.

---

<sup>10</sup> [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things)

Serviciul Firebase Real Time Database oferă o bază de date NoSQL, stocată în Cloud, ce permite salvarea și sincronizarea datelor în timp real. Datele sunt stocate într-un obiect JSON, care permite prelucrarea și gestionarea datelor în timp real. Conexiunea se realizează prin intermediul unui API, în baza de date fiind stocate valoarea curentă a datelor de interes și orice modificare apare ulterior. Avantajul utilizării acestui serviciu constă în ușurința cu care utilizatorii își pot accesa datele și pot comunica cu alți utilizatori. O altă facilitate oferită de acest serviciu este faptul că vine împreună cu SDK-uri (System Development Kit) mobile și web, ce permit construirea unei aplicații, fără necesitatea unor servere suplimentare. Atunci când utilizatorul este offline, SDK-ul folosește un cache local pentru a stoca modificările ce apar, modificări ce vor fi sincronizate automat când acesta redevine online.

Serviciul Firebase Authentication facilitează o manieră simplă și intuitivă de autentificare. Acesta oferă SDK-uri și biblioteci pentru autentificarea utilizatorilor în aplicație. Autentificarea se poate face pe baza următoarelor criterii: email și parolă, număr de telefon, google, facebook etc. Utilizarea acestui serviciu crește nivelul de securitate al procesului de autentificare, îmbunătățind procesul de logare.

Serviciul Firebase Cloud Messaging (FCM) permite trimiterea de mesaje și notificări, oferind o bună conexiune între server și dispozitive. Mesajele pot fi trimise pe iOS, Android și web, doar pe baza conexiunii realizate, fără nici un cost. Trimiterea mesajelor se realizează instant sau la momente ulterioare de timp, pe care le dorim. Se pot trimite mesaje fie unui singur dispozitiv, fie unui grup de dispozitive, ce sunt abonate la anumite subiecte.

Serviciul Firebase Database Query permite preluarea datelor prin intermediul unor interogări, aceste interogări se bazează pe următoarele metode: `orderByKey()`, `orderByChild('child')`, `orderByValue()`, `orderByPriority()`.

Serviciul Firebase Storage permite stocarea conținutului generat de utilizator, asigurând securitatea datelor și flexibilitatea rețelei. Acesta folosește un sistem de fișiere pentru a structura datele.

Serviciul Firebase Test Labs pune la dispoziție o varietate de dispozitive mobile de test, destinate testării aplicației. Acesta vine cu 3 moduri de testare: Instrumentation Test, Robo Test și Game Loop Test. Testele de tip Instrumentation Test sunt scrise de dezvoltatorul aplicației, pentru a testa aplicația, în general sunt utilizate framework-uri precum: Espresso și UI Automator. Testele Robo Test utilizează Firebase Test Labs pentru a simula click-uri în aplicație și pentru a observa cum se comportă fiecare funcționalitate. Testele Game Loop Test sunt folosite în general pentru testarea jocurilor și vin cu o versiune demonstrativă în care aplicația rulează, simulând acțiunile jucătorului.



Serviciul Remote Config ne ajută să transmitem schimbările utilizatorului instantaneu. Acesta permite modificarea aplicației fără a mai fi necesar să încărcăm o nouă versiune a aplicației în Play Store și să observăm cum se comportă o anumită parte sau cum arată aplicația în funcție de dispozitiv.

Serviciul Firebase App Indexing îți oferă posibilitatea de a indexa un anumit conținut, folosind în aplicație același URL ca al website-ului. Astfel, utilizatorii vor fi redirecționați către aplicație la apăsarea link-ului.

Serviciul Firebase Dynamic Links se bazează pe conținutul link-urilor dinamice. Aceste link-uri dinamice te redirecționează către o anumită zonă din aplicație. Sunt folosite pentru: transformarea utilizatorilor web în utilizatori de aplicații native, creșterea partajării între utilizatori, setarea de mesaje personalizate.

Serviciul Firestore folosește Cloud Firestore, care este o bază de date destinată documentelor, de tip NoSQL, ce permite stocarea, sincronizarea și căutarea datelor la o scară largă. Se aseamănă cu Real Time Database, aducând în plus alte funcționalități.

Avantajele utilizării Firebase sunt numeroase, iar printre ele se numără faptul că datele stocate sunt copiate și salvate în mai multe regiuni, astfel datele sunt în siguranță. Un alt avantaj este consistența ridicată, orice schimbare asupra datelor se va reflecta asupra fiecărei copii din baza de date.<sup>11</sup>

## 6.5 Android

Android este un sistem de operare open-source dezvoltat de Google, ce se bazează pe kernel-ul Linux. Acesta este dezvoltat în special pentru dispozitivele cu touchscreen, oferind posibilitatea dezvoltării de aplicații pentru aceste dispozitive. Fiind open-source, dezvoltatorii se pot folosi de framework pentru a-și contrui propriile aplicații și mai apoi să le facă disponibile pentru alți utilizatori.

Android Studio este un IDE (Integrated Development Environment) special pentru dezvoltarea aplicațiilor Android. Acest mediu de dezvoltare vine la pachet cu un SDK (Software Development Kit) și cu toate bibliotecile necesare. Fiecare versiune de Android are propriul SDK, cu ajutorul căruia putem dezvolta aplicații ce suportă cele mai noi funcționalități. Este important să ne actualizăm versiunea de Android pentru a beneficia de funcționalitățile noi apărute și pentru a nu pierde consistența versiunii anterioare. SDK-ul ne permite să creăm AVD-uri (Android Virtual Devices), acestea fiind utile în testarea aplicației.

---

<sup>11</sup> <https://firebase.google.com/docs>.

Android oferă o scalabilitate crescută, poate rula pe o mare diversitate de dispozitive, cu diferite configurații ale ecranului. Pentru a optimiza aplicația dezvoltatorii au posibilitatea de a redimensiona aplicațiile pentru diferite dimensiuni și densități ale ecranului. Sistemul pune la dispoziție un API care vine în ajutorul dezvoltatorilor, acesta permițând controlul interfeței grafice.

O aplicație Android este formată din următoarele componente: Activities, Layout, Values și Drawables. Activitățile sunt clasele Java în care se găsește codul de dezvoltare al aplicației (backend). Layout-ul conține mai multe fișiere XML, în acestea se găsește codul pentru design-ul aplicației (frontend). Valorile pot fi: Animation Resources, Color State

List Resource, Drawable Resource, Layout Resource, Menu Resource, String Resource și Style Resource. Drawable conține resursele grafice folosite în aplicație.

În Android Studio fiecare aplicație conține un proiect de nivel superior cu unul sau mai multe module ale aplicației, fiecare modul conținând la rândul lui echivalentul unui proiect Eclipse, setările necesare, inclusiv src/main, src/androidTest, resurse și Android Manifest. Eclipse oferă un workspace comun pentru gruparea proiectelor, configurațiilor și setărilor legate de acestea. Fișierul src/main se modifică pentru actualizarea și rularea aplicației, fișierul gradle.build conține specificațiile de rulare, iar fișierele din src/androidTest conțin scenariile de test.

Gradle reprezintă un instrument de construire, care ajută în automatizarea, testarea și desfășurarea sarcinilor. Android Studio utilizează Gradle pentru construirea diferitelor versiuni și generarea de APK-uri (Android Package) multiple. Fișierele gradle (build.gradle) sunt fișiere de construcție, utilizate în configurare, ce utilizează sintaxa Groovy. În cadrul fișierului build.gradle (app) pot fi adăugate implementări, ce vor fi descărcate automat, aceste implementări fiind biblioteci ce aduc funcționalități suplimentare aplicației.<sup>12</sup>

O aplicație Android poate avea mai multe activități, scopul acestora fiind de a interacționa cu utilizatorul. O activitate are un ciclu de viață (figura 6.5.1), de la momentul în care apare pe ecran, până la momentul în care aceasta dispare. Din categoria funcționalităților oferite fac parte fragmentele și intenturile. Fragmentele sunt niște activități mai mici, care grupate pot forma o activitate. Un intent este un conector, care asigură executarea corectă a task-urilor între aplicații, făcând conexiunea între activitățile acestora. Pentru a face trecerea de la o activitate la alta, este necesară crearea unui nou obiect de tip Intent, care primește ca parametri numele activității curente, de la care se va face trecerea, și numele clasei corespunzătoare activității spre care se face trecerea. Trecerea propriu-zisă se va face atunci când este apelată metoda startActivity(intent), unde intent reprezintă obiectul creat mai sus. De asemenea, se pot transmite valori ce pot fi

---

<sup>12</sup> Aliferi C., (2016), *Android Programming Cookbook*, Exelixis Media.

necesare altor activități tot prin intermediul unui Intent. După crearea obiectului de tip Intent, se va apela metoda `intent.putExtra("tag", valoare)`, unde tag reprezintă o referință după care va fi identificată valoarea transmisă, în cadrul celeilalte activități. Valoarea se va recepționa în cadrul celeilalte activități prin intermediul metodei `getIntent().getStringExtra("tag")`, care va returna String-ul asociat tag-ului respectiv.

Pentru a crea o activitate se contruiește o nouă clasă, care va extinde clasa de bază Activity. Fiecare activitate este declarată în fișierul Manifest.xml. Clasa Activity are definite următoarele metode: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`, `onDestroy()` și `onRestart()`. Metoda `onCreate()` este apelată atunci când activitatea este creată. Metoda `onStart()` este apelată abia atunci când activitatea devine vizibilă de către utilizator. Metoda `onResume()` este apelată atunci când utilizatorul începe interacțiunea cu activitatea. Metoda `onPause()` este apelată atunci când este pusă pe pauză activitatea curentă, iar cea anterioară este preluată. Metoda `onStop()` este apelată atunci când utilizatorul nu mai vede activitatea. Metoda `onDestroy()` este apelată la distrugerea activității de către sistem. Metoda `onRestart()` este apelată la restartarea activității. Orice activitate nouă creată va implementa metodele menționate anterior. Unei activități îi pot fi aplicate diverse stiluri și teme.

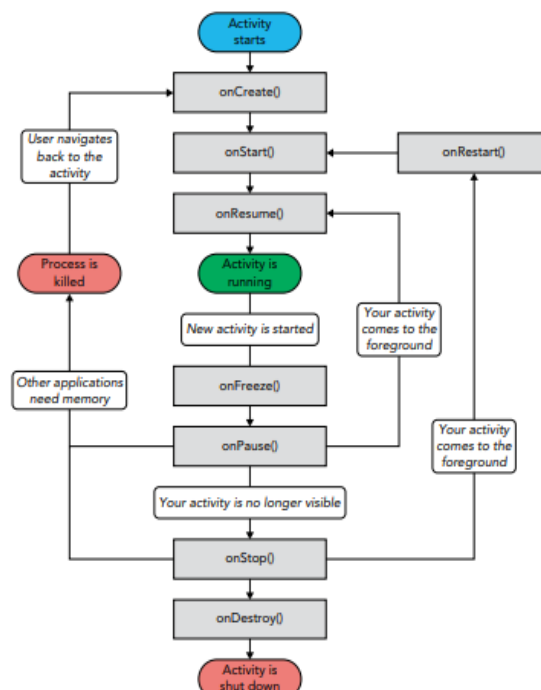


Figura 6.5.1 Ciclul de viață al unei activități

Atunci când există mai mult de o activitate în aplicație, deseori este necesară trecerea de la una la alta. Intent-urile sunt cele care realizează această trecere între activități. O activitate este formată dintr-o clasă (java) și o componentă UI (xml). Legătura între aceste două componente se face prin intermediul funcției `setContentView(R.layout.nume_fișier_xml)`, funcție ce va fi apelată la început, în cadrul metodei `onCreate()`, pentru fiecare activitate.

Orice aplicație trebuie obligatoriu să conțină un fișier numit `AndroidManifest.xml` în directorul rădăcină al proiectului. Acest fișier conține informații esențiale pentru sistemul Android, legate de aplicație, și orice sistem trebuie să aibă acest fișier înainte să ruleze orice cod al aplicației. Printre multe alte funcționalități, acest fișier prezintă următoarele caracteristici: denumește pachetul Java al aplicației, ce servește ca identificator unic pentru aceasta; descrie componentele aplicației: activitățile și serviciile din care este compusă aplicația, numește clasele care implementează fiecare componentă și le descrie capabilitățile (de exemplu, ce mesaje de tip `Intent` pot suporta); acesta definește componentele și condițiile în care pot fi ele lansate de către sistemul Android. De asemenea, declară permisiunile pe care aplicația trebuie să le aibă pentru a accesa părți protejate ale API-ului și pentru a interacționa cu alte aplicații și permisiunile necesare pentru a putea interacționa cu componentele aplicației. Acest fișier mai declară și nivelul minim de Android API pe care îl necesită aplicația pentru a rula. De asemenea, în cadrul acestui fișier se pot defini iconița și numele aplicației, ce se vor vizualiza pe smartphone-ul pe care aceasta va rula.

În cadrul fiecărei activități este definită componenta grafică (UI) corespunzătoare prin apelarea metodei `setContentView(R.layout.activity)`. Legătura între elementele grafice și activitate se face prin `findViewById(R.id.id_elementgrafic)`. Id-ul pentru fiecare element grafic se definește în cadrul fișierului xml. O activitate conține views și ViewGroups. Exemple de view-uri ce se pot regăsi ca elemente grafice : butoane, label-uri, textbox-uri etc. Un ViewGroup este format din unul sau mai multe view-uri. Acesta oferă un layout, care stabilește ordinea apariției view-urilor. Layout-urile pot fi : `FrameLayout`, `LinearLayout` (Horizontal/Vertical), `TableLayout`, `TableRow`, `GridLayout` și `RelativeLayout`. `LinearLayout` aranjează view-urile pe o singură coloană sau pe un singur rând, pe când `RelativeLayout` îți permite modificarea relativă în pagină. Printre view-urile elementare necesare unei interfețe se numără : `TextView`, `EditText`, `Button`, `ImageButton`, `CheckBox`, `ToggleButton`, `ProgressBar`, `Spinner`, `RadioButton` și `RadioGroup`. Aceste elemente grafice pot avea următoarele atribute : `width`, `height`, `text`, `orientation`.<sup>13</sup>

## 6.6 Java & POO

Java este un limbaj de programare ce se bazează în principiu pe OOP (Object Oriented Programming). Acest limbaj împrumută o parte din sintaxa limbajelor C și C++, aducând o serie de noi facilități. Pentru a dezvolta și executa aplicații în Java se utilizează platforma Java. Aceasta este formată din 2 mari componente: : JRE (Java Runtime Environment) și JDK (Java Development Kit). JRE-ul este necesar pentru rularea

---

<sup>13</sup> DiMarzio J. F., (2017), *Beginning Android Programming with Android Studio*, by John Wiley & Sons, Inc.

aplicațiilor Java, iar JDK-ul pentru dezvoltarea aplicațiilor Java. La instalarea JDK-ului acesta vine cu echipat cu un JRE.

JRE-ul este cel care la execuția unei aplicații face legătura între platformă și aplicație. Acesta interpretează aplicația, pentru a verifica că rulează în conformitate cu platforma. Deci, JRE-ul asigură rularea codului pe diverse platforme.

JDK-ul vine la pachet cu diverse tool-uri de dezvoltare. Unul dintre aceste tool-uri este compilatorul Java, acesta transformă codul java în cod binar.

JVM-ul (Java Virtual Machine) este una din componentele de bază ale JRE-ului. Acesta se comportă ca un procesor virtual, permitând aplicațiilor să ruleze pe sistem. Rolul JVM-ului este de a interpreta codul și a-l transforma în cod nativ.

JCL(Java Class Libraries) vine cu o serie de clase și biblioteci, ce conțin o mulțime de metode reutilizabile.

Modularitatea este foarte importantă atunci când vorbim despre programarea orientată pe obiecte (POO). Împărțirea programelor în mai multe module și rezolvarea fiecărui modul în parte ușurează rezolvarea problemei de la care pornim. Avantajele folosirii acestui concect sunt : ușurința în rezolvarea modulelor, re folosirea modulelor, testarea individuală a modulelor și faptul că oameni diferiți pot lucra simultan cu diferite module. .

În general POO se bazează pe lucrul cu obiecte și clase. Modulul unitate în programarea orientată pe obiecte este clasa. O clasă conține atribute și metode. Atributele sunt proprietățile ce descriu clasa, iar metodele sunt acțiunile/funcționalitățile clasei. Un obiect este o instanță a unei clase. Atributele și metodele pot fi : public, private și protected. Modificatorul de acces public îți permite vizibilitatea atributelor și metodelor din orice altă clasă, pe când private îți restricționează accesul la acestea, vizibilitatea lor fiind posibilă doar în clasa părinte. Modificatorul protected se utilizează pentru a oferi vizibilitate și subclaselor ce moștenesc clasa părinte. Conceptele de bază ale programării orientate pe obiecte sunt : incapsularea, moștenirea, polimorfismul și abstractizarea.

Conceptul de incapsulare se referă la ascunderea informației dintr-o clasă. Acest concept definește felul în care se pot accesa datele dintr-o anumită clasă. Definirea datelor de tip private/public/protected crește nivelul de securitate, asigurând utilizarea corectă a acestora.

Conceptul de moștenire permite unei clase să moștenească atributele și metodele altei clase prin derivarea acesteia. Prin derivarea unei clase de bază se pot obține mai multe clase derivate. O clasă derivată poate utiliza metodele clasei de bază. Pentru a asigura că accesul la date se face în mod corect, declararea atributelor și a metodelor se face de tip protected. Astfel datele pot fi utilizate de clasa de bază și de clasa derivată. Acest concept oferă posibilitatea utilizării funcționalităților existente, dar și adăugarea altora.

Conceptul de polimorfism oferă posibilitatea procesării diferite a obiectelor în funcție de tip/clasă. Acest concept permite claselor derivate redefinirea metodelor moștenite.

Conceptul de abstractizare se referă la construirea unui tipar, care prin derivare duce la obținerea altor clase. O clasă abstractă conține metode pur virtuale. Orice clasă care moștenește o clasă abstractă este obligată să implementeze metodele clasei abstracte. Pe lângă acest concept mai există și conceptul de interfață. O interfață este o clasă pur abstractă, mai precis constituie un șablon ce definește structura atributelor și a metodelor, fără o implementare propriu zisă. Câmpurile unei interfețe sunt static și final, iar metodele sunt public în mod implicit. Interfețele ne ajută atunci când ne dorim să avem o structură fără o implementare propriu-zisă, iar clasele abstracte atunci când ne dorim să implementăm doar anumite metode din clasă sau atunci când ne dorim să nu putem instanția clasa.

Alte concepte ce sunt utilizate în programarea orientată pe obiecte sunt overloading și overriding. Overloading-ul permite apariția unei metode cu aceeași denumire, dar cu funcționalități diferite. Aceasta are loc la compilare. Overriding-ul permite redefinirea metodelor din clasa părinte în clasa copil. Se pot suprascrie doar acele metode care au ca modificador de acces public sau protected. Overriding-ul are loc la runtime.<sup>14</sup>

## 6.7 XML

XML (eXtensible Markup Language) constituie un meta-limbaj folosit în activitatea de marcare structurată a documentelor. Acest meta-limbaj este o variantă simplificată a limbajului SGML, fiind proiectat în mod special pentru transferul de date între aplicații. XML-ul reprezintă un model de stocare a datelor semi-structurate și nestructurate în baze de date. În comparație cu HTML, XML-ul este mai bun, oferind o dinamicitate a datelor, diferența fiind tag-urile ce se pot defini la libera alegere a programatorului.

Avantajele oferite de acest limbaj sunt multiple : extensibilitatea, validitatea, accesibilitatea. Extensibilitatea se referă la definirea de noi indicatori/tag-uri atunci când este nevoie. Validitatea se referă la asigurarea corectitudinii structurii datelor. Accesibilitatea se referă la ușurința în stocare, modificare și acces la date. Documentele XML au structura unui arbore. Acestea conțin noduri (tipul elementului și atributele) și date. Un tag este un șir de caractere delimitat de “< >” , datele caracter reprezentând conținutul tag-urilor.

Limbajul XML oferă o metodă de organizare ierarhică a informației, prin împărțirea datelor în elemente, elemente ce pot fi descrise de anumite atribute. Pentru ca un

---

<sup>14</sup> <http://elf.cs.pub.ro/poo/laboratoare/poo-java>

document XML să fie bine format trebuie ca acesta să conțină următoarele secțiuni : prologul, definiția documentului și elementul rădăcină cu nodurile copil. Prologul asigură faptul că documentul respectă versiunea specificată. Acesta se găsește în partea de început a documentului și conține declarații XML, comentarii, instrucțiuni de procesare, spații de nume și declarația DOCTYPE.

Definiția documentului conține declarația unui document de tip DTD, care este format dintr-un set de reguli pentru definirea structurii fișierului XML. Elementul rădăcină conține nodul părinte și nodurile copil aferente. Pe lângă acestea un document XML mai poate conține următoarele marcaje: elemente, attribute, comentarii, entități, secțiuni CDATA ș.a.md.

Elementele reprezintă nodurile unui document XML. Un element este definit între un tag de început și unul de sfârșit. Un element poate conține informații sau poate fi vid. Acestea sunt folosite pentru stocarea informației, dar și pentru definirea structurii. Numele elementelor nu pot conține spații și nu pot începe cu xml, o cifră, “\_” sau alte caractere speciale. Se recomandă ca numele elementelor să fie formate dintr-o combinație de litere, cifre și “\_”.

Atributele se pot atribui unui element și au rol descriptive. Acestea se poziționează în tag-ul de început al elementului, după ce a fost specificat numele tag-ului. Atributele sunt urmate de semnul “=”, urmat de o valoare. Valorile atributelor trebuie plasate între ghilimele și pot conține orice tip de text, însă nu pot fi vide.

Entitățile sunt caractere speciale și nu se pot utiliza în XML. Pentru a le putea utiliza se folosesc referințe la entități. Acestea au nume unice, spre exemplu: &nume (&lt, &gt, &amp, &quot, &apos). Comentariile pot fi definite oriunde în document, acestea sunt utilizate pentru descrierea anumitor detalii legate de structură sau conținut. Comentariile se definesc astfel :  
<!-- comentariu -->, și nu sunt limitate în lungime.

Alte marcaje speciale sunt instrucțiunile de procesare. Acestea conțin informații despre aplicații ce urmează a fi executate. Forma lor generală este următoarea: <? ?>.

Secțiunile CDATA se folosesc pentru includerea blocurilor text ce conțin caractere, care altfel ar fi cunoscute ca fiind marcaje. Se folosesc atunci când ne dorim ca datele să fie considerate caractere, pentru a împiedica interpretarea lor de către parser.

Fișierele XML pot fi:

- Bine-formate: care respectă regulile;
- Valide: care sunt bine-formate și respectă declarația de structură;
- Invalide: care sunt bine formate, dar nu respectă declarația de structură;

Pentru ca un document XML să fie bine format trebuie ca acesta să îndeplinească următoarele condiții :

- Să aibă un singur element rădăcină;
- Documentul trebuie să conțină declarația XML : `<?xml version="1.0"?>`;
- Orice tag deschis să fie închis;
- Tag-urile sunt case-sensitive;
- Valorile atributelor trebuie să fie plasate între ghilimele;
- Reprezentarea caracterelor speciale se face prin entități;
- Imbricarea corectă a tag-urilor;
- Atributele trebuie să aibă nume unice;<sup>15</sup>

## 6.8 One Signal

One Signal este o platformă ce oferă servicii de notificare de tip push pentru aplicații mobile și site-uri web. Acest serviciu oferă suport pentru diverse platforme, oferind SDK-uri speciale pentru fiecare.

Acest serviciu vine la pachet cu un set de metode predefinite.

Metoda `init()` realizează inițializarea serviciului pentru a înregistra dispozitivul, aceasta se apelează în metoda `onCreate()` a clasei respective. Forma de apel este următoarea: `OneSignal.startInit(this).init()`. Metoda `startInit()` se apelează de asemenea în metoda `onCreate()` și are același rol ca `init()` de inițializare a serviciului.

Metoda `setNotificationReceivedHandler()` setează o notificare ce va fi primită pe dispozitivul configurat la declanșarea unui eveniment. Serviciul va funcționa atunci când aplicația este în funcțiune sau când aceasta este în bară. Structura metodei este următoarea: .

`setNotificationReceivedHandler(new ExampleNotificationReceivedHandler())`.

Metoda `setNotificationOpenedHandler()` se apelează la apăsarea notificării, închizând notificarea de avertizare apărută în aplicație. Structura metodei este: `.setNotificationOpenedHandler(new ExampleNotificationOpenedHandler())`.

---

<sup>15</sup> [https://acs.curs.pub.ro/2018/pluginfile.php/48439/mod\\_resource/content/3/CURS%202017.pdf](https://acs.curs.pub.ro/2018/pluginfile.php/48439/mod_resource/content/3/CURS%202017.pdf)



Metoda `setInFocusDisplaying()` organizează felul în care notificările vor fi primite. Structura de apel este următoarea:

`OneSignal.setInFocusDisplaying(OneSignal.OSInFocusDisplayOption.Notification);`

Unde Notificarea este notificarea ce va fi primită,

Metoda `provideUserContent()` îți permite partajarea conținutului privat. Structura apelului de metodă : `OneSignal.provideUserConsent(true)`. SDK-ul se va inițializa și va începe să trimită date abia la apelul `provideUserConsent(true)`.

Metoda `userProvidedPrivacyConsent()` returnează o valoare de tip `bool`, care ne spune dacă utilizatorul îți permite accesul la conținut.

Metoda `disableGmsMissingPrompt()` oferă utilizatorului posibilitatea de a activa/actualiza serviciile Google Play la apelarea ei. Structura apelului: `.disableGmsMissingPrompt(true)`.

Metoda `unsubscribeWhenNotificationsAreDisabled()` are următoarea structură a apelului: `unsubscribeWhenNotificationsAreDisabled(valoare)`, dacă valoarea este `false` nu dezabonează utilizatorii, iar dacă este `true` dezabonează utilizatorii când notificările sunt dezactivate.

Metoda `filterOtherGCMReceivers()` are următoarea structură a apelului : `.filterOtherGCMReceivers(true)`. Această metodă împiedică alte receptoare de broadcast să primească mesaje de la OneSignal. De asemenea, previne dublarea notificărilor ce provin de la alte biblioteci/SDK-uri care implementează notificări.

Metoda `postNotification()` permite trimiterea notificărilor de la un utilizator la altul, sau stabilirea trimiterii notificărilor pe un dispozitiv la un moment viitor de timp.

Metoda `cancelNotification()` anulează trimiterea unei notificări pe bază de id. Structura apelului metodei este următoarea: `OneSignal.cancelNotification(id);`

Metoda `sendTag()` permite identificarea unui utilizator pe baza unui eveniment, lucru ce poate fi folosit ulterior în crearea segmentelor. Pentru trimiterea mai multor tag-uri simultan se apelează metoda `sendTags()`.<sup>16</sup>

---

<sup>16</sup> <https://documentation.onesignal.com/docs/android-native-sdk>.

## 7. Scenarii de utilizare

În acest capitol se va prezenta aplicația din perspectiva utilizatorului. Astfel, dispozitivul va fi pus în funcțiune pentru a comunica cu aplicația, care va prelua valorile transmise de acesta. Se presupune că utilizatorul va avea conectat asupra lui dispozitivul portabil de monitorizare, cu o conexiune stabilă la Wi-Fi, pentru a permite transmiterea datelor.

Personalul medical va fi utilizatorul, acesta va folosi aplicația pentru a monitoriza pacienții săi.

Aplicația se va deschide pe pagina principală (Figura 7.1), de unde utilizatorul poate să selecteze una dintre opțiuni: logare sau înregistrare. La selectarea unei opțiuni acesta va fi redirecționat către pagina corespunzătoare.

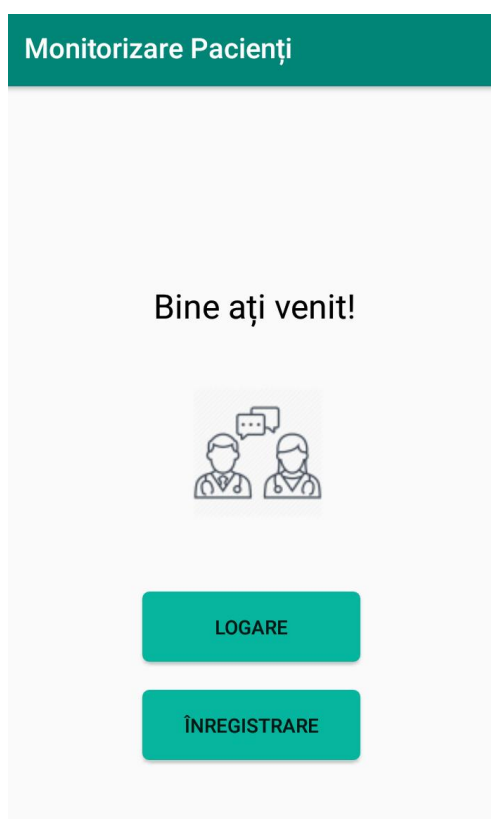


Figura 7.1 Pagina principală a aplicației

În pagina de înregistrare (figura 7.2) utilizatorul își poate crea un nou cont, ce se va salva în baza de date. De asemenea, la crearea noului cont se va adăuga în FirebaseAuth email-ul contului, generându-se un id unic pentru noul cont creat. Din această pagină se va

face redirecționarea înapoi spre pagina principală. În cazul necompletării unor câmpuri se va genera un mesaj de eroare: “Date incomplete”.

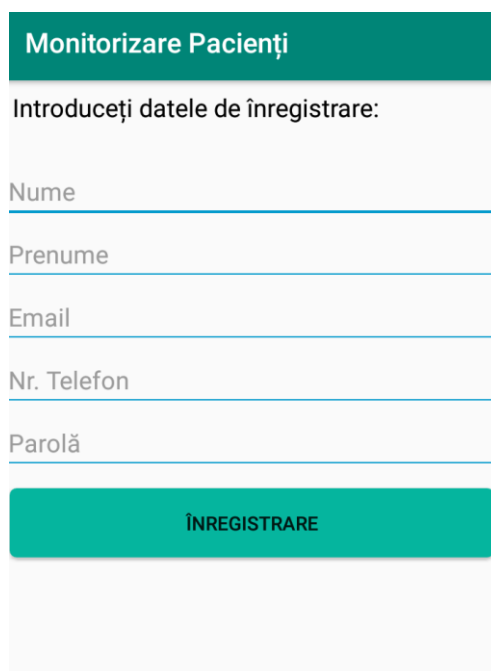


Figura 7.2 Pagina de înregistrare

În pagina de logare (figura 7.3), utilizatorul trebuie să introducă credențialele unui cont valid, altfel se va genera un mesaj de eroare. În cazul introducerii corecte a credențialelor de logare, utilizatorul va fi redirecționat pe pagina personalului medical.

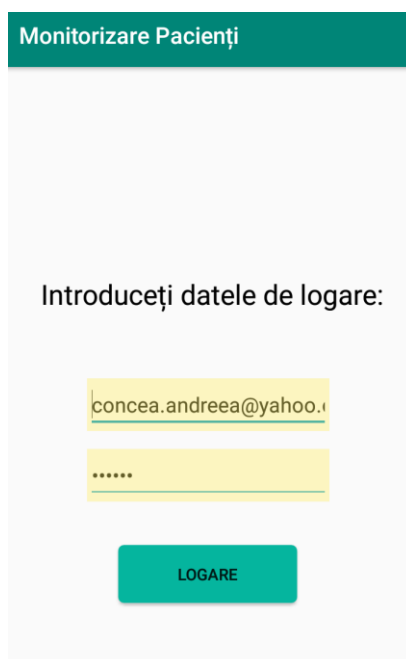


Figura 7.3 Pagina de logare

Pagina personalului medical (figura 7.4) oferă două posibilități: căutare pacient și adăugare pacient. La introducerea unui CNP se va căuta pacientul în baza de date, în cazul în care acesta a fost găsit, la apăsarea butonului se va deschide pagina de monitorizare a pacientului respectiv. În caz contrar se va genera un mesaj de eroare: “Pacientul căutat nu a fost găsit”.

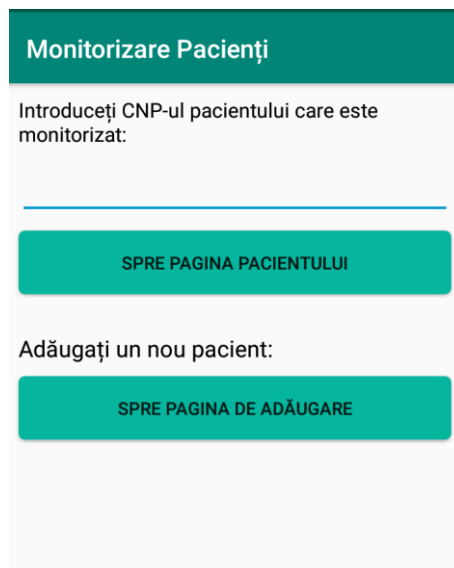


Figura 7.4 Pagina personalului medical

Pagina de adăugare (figura 7.5) este similară cu cea de înregistrare pacienți, datele fiind salvate în Firebase Realtime Database. La necompletarea câmpurilor se va genera un mesaj de eroare: “Date incomplete”. Pacientul va fi înregistrat în cadrul doctorului care l-a introdus în baza de date.

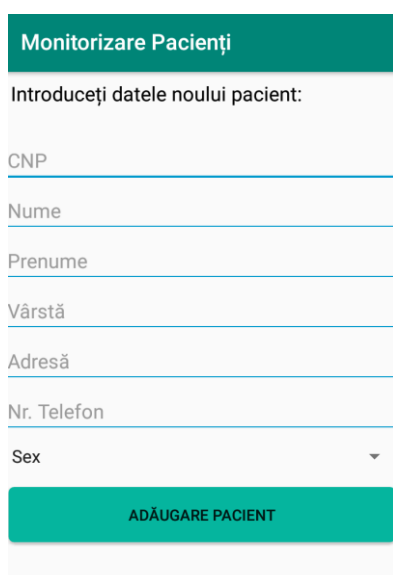



Figura 7.5 Pagina de adăugare pacient

În pagina de monitorizare a pacientului (Figura 7.6) vor fi afișate datele personale ale acestuia. De asemenea, doctorul are opțiunea de vizualizare grafică a datelor. La apăsarea acelui buton se va deschide o nouă pagină, în care vor putea fi vizualizate grafic valorile trimise de la dispozitivul pacientului (Figura 7.7).



The screenshot shows a web interface titled "Monitorizare Pacienți". It contains a form with the following fields and values:

Monitorizare Pacienți	
Adresă:	Str Lalelelor 20
Nume:	Popescu
Prenume:	Ion
Sex:	Masculin
Telefon:	0756453218
Vârstă:	60

At the bottom right of the form is a green button labeled "PAGINA DE MONITORIZARE".

Figura 7.6 Pagina pacientului

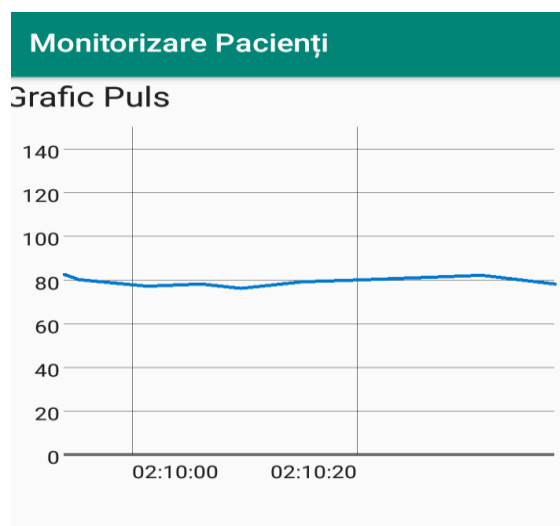


Figura 7.7 Pagina de vizualizare grafică a datelor

Odată intrat pe pagina pacientului, utilizatorul va fi abonat la notificări asupra parametrilor de interes ai acestuia. În cazul apariției unor valori anormale la rând (prea mari/prea mici) ale pulsului, se va trimite instant o notificare de alarmare personalului medical (Figura 7.8).

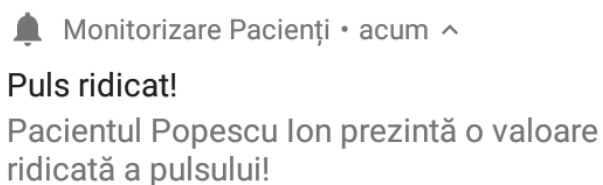


Figura 7.8 Notificare puls

De asemenea, în cazul unor pierderi de echilibru, detectate în urma implementării algoritmului de cădere, ce diferențiază o cădere gravă de o cădere liberă, se va trimite o notificare (Figura 7.9).

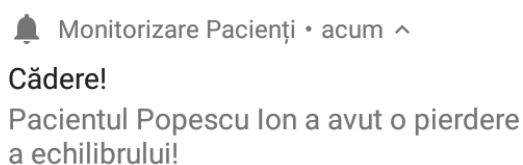


Figura 7.9 Notificare cădere

## 8. Dezvoltări ulterioare

În această secțiune vor fi prezentate dezvoltări ulterioare ce pot aduce îmbunătățiri sistemului și totodată experienței utilizatorului.

Pe partea hardware o posibilă dezvoltare este extinderea sistemului cu un număr mai mare de senzori, senzori ce vor permite o monitorizare complexă a pacientului. De asemenea, se vor dezvolta algoritmi pentru calcularea valorilor achiziționate de la aceștia și se vor adăuga la sistemul de alarmare.

Pe partea software, o dezvoltare adusă aplicației este adăugarea suportului pentru mai multe limbi. Utilizatorul va putea selecta dintr-un meniu limba dorită. Totodată, se dorește implementarea aplicației și pe dispozitive mobile ce suportă sistem de operare iOS. Aceste dezvoltări ajutând la extinderea ariei de utilizatori.

O altă dezvoltare ce poate fi adusă aplicației este integrarea unui modul GPS, care va localiza pacientul pe o hartă ce va fi introdusă în cadrul aplicației. Astfel, la detectarea unei posibile situații de urgență, se va putea face localizarea pentru pacient.

## 9. Concluzii

În urma parcurgerii etapelor de dezvoltare ale proiectului, am ajuns la obținerea unui sistem de monitorizare în timp real a pacienților, sistem ce oferă atât servicii de vizualizare a evoluției parametrilor de interes, cât și servicii de alarmare în caz de urgență pentru personalul medical.

Sistemul obținut este format din două părți: un dispozitiv destinat pacienților și o aplicație IoT de monitorizare destinată doctorilor.

Sistemul hardware rezultat în urma acestui proiect este un dispozitiv portabil, autonom, fiind ușor de utilizat datorită dimensiunilor relativ mici. Proiectul se adresează în special persoanelor de vârstă a treia deoarece parametri monitorizați sunt mai sensibili pentru această categorie de vârstă.

Aplicația este destinată personalului medical, oferind diverse funcționalități în scopul monitorizării pacienților. Aceasta oferă suport pentru toate dispozitivele mobile ce utilizează sistemul de operare Android. Aplicația are o interfață prietenoasă și intuitivă, denumirile utilizate în cadrul acesteia fiind sugestive.

Pentru a putea fi utilizat, întregul sistem necesită o conexiune stabilă la internet, pentru a se realiza comunicația dintre cele două părți prin intermediul serviciilor oferite de Firebase. Datele achiziționate de la dispozitivul hardware sunt transmise prin intermediul Wi-Fi în baza de date în timp real. De acolo sunt preluate și transmise mai departe pentru a putea fi utilizate în aplicație.

Funcționalitatea de alarmare se bazează pe algoritmi de detectare a căderii și de calcul al pulsului. Notificările vor fi generate prin intermediul OneSignal pe baza valorilor preluate din baza de date.

Soluția obținută este un prototip, oferind monitorizare pentru un număr restrâns de parametri și suport doar în limba română, aspecte ce vor putea fi dezvoltate ulterior.



## 10. Bibliografie

1. [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things), accesat: 2019
2. <https://www.healthline.com/health/dangerous-heart-rate?fbclid=IwAR3ODFVIMQRpsimPdPCIm5jYe49P2-g6RwoT1caJx2Q3BAzllanINrx3JQ#the-takeaway>, accesat: 2019
3. Guyton A. , Hall J., (2007), *Tratat de fiziologie a omului*, Editura Medicala Callisto.
4. [https://www.handsontec.com/pdf\\_learn/esp8266-V10.pdf](https://www.handsontec.com/pdf_learn/esp8266-V10.pdf), accesat: 2019
5. [https://www.optimusdigital.ro/ro/senzori-altele/1273-senzor-de-puls-xd-58c.html?search\\_query=senzor+de+puls&results=26](https://www.optimusdigital.ro/ro/senzori-altele/1273-senzor-de-puls-xd-58c.html?search_query=senzor+de+puls&results=26), accesat: 2019
6. [https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/586-senzor-de-temperatura-rezistent-la-apa.html?search\\_query=senzor+de+temperatura&results=220](https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/586-senzor-de-temperatura-rezistent-la-apa.html?search_query=senzor+de+temperatura&results=220), accesat: 2019
7. [https://www.optimusdigital.ro/ro/senzori-senzori-inertiali/96-modul-senzor-triaxial-mpu-6050.html?search\\_query=accelerometru+mpu6050&results=6](https://www.optimusdigital.ro/ro/senzori-senzori-inertiali/96-modul-senzor-triaxial-mpu-6050.html?search_query=accelerometru+mpu6050&results=6), accesat: 2019
8. <https://ro.wikipedia.org/wiki/Arduino>, accesat: 2019
9. <https://ro.wikipedia.org/wiki/Wi-Fi>, accesat: 2019
10. [https://en.wikipedia.org/wiki/Internet\\_of\\_things](https://en.wikipedia.org/wiki/Internet_of_things), accesat: 2019
11. Documentație Firebase - <https://firebase.google.com/docs>, accesat: 2019
12. Aliferi C., (2016), *Android Programming Cookbook*, Exelixis Media.
13. DiMarzio J. F., (2017), *Beginning Android Programming with Android Studio*, by John Wiley & Sons, Inc.
14. Documentație POO – <http://elf.cs.pub.ro/poo/laboratoare/poo-java>, accesat: 2019
15. [https://acs.curs.pub.ro/2018/pluginfile.php/48439/mod\\_resource/content/3/CURS%202017.pdf](https://acs.curs.pub.ro/2018/pluginfile.php/48439/mod_resource/content/3/CURS%202017.pdf), accesat: 2019
16. Documentație OneSignal - <https://documentation.onesignal.com/docs/android-native-sdk>, accesat: 2019
17. Cuno Pfister, (2011), *Getting Started with the Internet of Things*, O'Reilly.
18. Deepak Uttamchandani , (2013), *Handbook of MEMS for wireless and mobile applications*, Woodhead Publishing.
19. Downey Allen B., (2011), *Think Java: How to think like a computer scientist*, O'Reilly.
20. Erik T. Ray, (2001), *Learning XML*, O'Reilly.
21. Mark Geddes, (2016), *Arduino Project Handbook*, O'Reilly.
22. Matt A. Weisfeld, (2000), *The Object-Oriented Thought Process*, Developer's Library.
23. Stoyan Nihtianov and Antonio Luque, (2014), *Smart sensors and MEMS*, Woodhead Publishing.
24. Syed V. Ahmed, (2013), *Intelligent Networks Recent Approaches and Applications in Medical Systems*, Elsevier.
25. Documentație Android Studio IDE - <https://developer.android.com/docs>, accesat: 2019