Calculatoare și Tehnologia Informației
2021-2022

Universitatea Politehnica din Timișoara

APLICAȚIE MOBILĂ PENTRU MONITORIZAREA TRATAMENTULUI MEDICAMENTOS AMBULATORIU

Candidat: Ștefania ȘTEABURDEA

Coordonator științific: Prof. Dr. Ing. Mihai UDRESCU

S.L. Dr. Ing. Alexandru IOVANOVICI

Sesiunea: Iunie 2022

REZUMAT

Aplicația *Time For Pills* are scopul de a monitoriza tratamentul medicamentos și de a le aminti utilizatorilor să își ia medicamentele la ora corespunzătoare prin intermediul unei alarme sau a unei notificări. Am ales această temă deoarece luarea la timp a medicamentelor este esențială, pentru ameliorarea, încetinirea evoluției sau vindecarea bolii. Această aplicație a fost realizată pentru sistemele Android.

În cadrul aplicației, medicația poate să fie gestionată prin adăugarea, ștergerea și editarea medicamentelor luate. Pot fi adăugate diferite informații despre medicament, precum numele, culoarea pastilei, ora de referință, intervalul dintre două doze, data începerii tratamentului și optional data finalizării acestuia. Toate aceste informații vor putea fi apoi editate.

În pagina principală a aplicației este posibilă vizualizarea medicamentelor ce vor trebui luate în ziua respectivă. Utilizatorul poate vedea numele medicamentelor, ora la care vor fi luate, dar și culoarea acestora. În partea inferioară a paginii este afișat istoricul medicației pe zile, sub forma unui calendar.

Pentru a anunța utilizatorul, în momentul în care trebuie luat un medicament, se va afișa o notificare și, opțional, dacă acesta dorește se va porni și o alarmă.

În pagina de setări, utilizatorii vor putea realiza conexiunea cu un ceas, conexiunea cu un dozator de medicamente și vor putea adăuga supraveghetori.

Ceasul verifică dacă utilizatorul a dus mâna la gură pentru luarea pastilei, conexiunea cu acesta realizându-se prin Bluetooth.

Dozatorul de medicamente va contoriza numărul de pastile disponibile și va alege pastila pe care utilizatorul trebuie să o ia. Pentru a putea distribui pastila corespunzătoare, utilizatorul va trebui să adauge compartimentul în care se află medicamentul. În momentul în care un utilizator este conectat la un dozator va apărea în pagina de adăugare și editare un câmp suplimentar pentru a menționa compartimentul. De asemenea în pagina în care sunt afișate medicamentele va apărea și numărul de pastile rămase, informație obținută de la dozator.

În secțiunea dedicată supraveghetorilor, un utilizator poate să adauge una sau mai multe persoane de contact. Se va adăuga numele supraveghetorului și numărul său de telefon, pentru ca în cazul în care un medicament nu a fost luat să poată fi trimise mesaje în acest sens. De asemenea, vor fi trimise mesaje și în cazul în care dozatorul de medicamente are cel mult 3 pastile de un anumit fel.

Această aplicație poate să fie utilă pentru persoanele de orice vârstă, oferind o monitorizare amănunțită a tratamentului medicamentos. De asemenea, supraveghetorii au posibilitatea de a fi înștiințați de fiecare dată când este semnalată vreo problemă, putând să monitoriza cu ușurință utilizatorii.

CUPRINS

1. Introducere 4
2. Specificații 6
2.1. Aplicații similare 6
2.2. Specificațiile sistemului 10
3. Arhitectura sistemului 12
3.1. Diagrama UML a cazurilor de utilizare 12
3.2. Proiectarea bazei de date 14
4. Implementarea funcționalităților 17
4.1. Autentificare și înregistrare 17
4.2. Navigarea folosind fragmente 18
4.3. Medicație 18
4.3.1. Adăugarea unui medicament nou 18
4.3.2. Afișarea medicamentelor 21
4.3.3. Editarea medicamentelor 23
4.4. Istoricul medicației 24
4.5. Alarme 26
4.5.1. Setarea alarmelor 26
4.5.2. Oprirea alarmelor 28
4.6. Trimiterea SMS-urilor 29
4.7. Interacțiunea cu ceasul 31
4.7.1. Conexiunea Bluetooth 31
4.7.2. Luarea medicamentului și oprirea alarmei 34
4.8. Interacțiunea cu dozatorul de medicamente 35
5. Testarea 41
6. Concluzii 44
6.1. Obiective realizate 44
6.2. Dezvoltări ulterioare 44
Bibliografie 46

1. INTRODUCERE

Eficiența unui tratament medicamentos ambulatoriu (la domiciliul pacientului) este influențată de o serie de factori, iar dintre aceștia, aderența pacientului la tratamentul prescris deține un rol foarte important. Pornind de la această iee, am decis să dezvolt un sistem ce monitorizează tratamentul medicamentos și anunță pacientul când ar trebui luat medicamentul, crescând aderența pacientului la tratament și, implicit, eficiența acestuia.

Complianța la tratamentul medicamentos, de asemenea cunoscută ca și aderență, reprezintă măsura în care sunt respectate recomandările medicului cu privire la tratamentul ce trebuie urmat [1]. Aceasta este foarte importantă pentru îmbunătățirea stării de sănătate și vindecarea bolilor.

Prin aderența la tratament este posibilă reducerea costurilor suplimentare generate de creșterea numărului spitalizărilor și de creșterea numărului de consultații în regim de urgență. din cauza non-aderenței la tratament, în Uniunea Europeană sunt estimate 194.500 de decese pe an, iar costurile cauzate de non-aderența la tratament sunt estimate la 125 de miliarde de euro anual [3].

Non-aderența la tratamentul medicamentos este o problemă frecvent întâlnită. În medie, aproximativ 50% din pacienți nu respectă cu exactitate orele la care un anumit medicament trebuie luat sau, în unele cazuri, pot uita complet de administrarea acestuia [2].

În cazul bolilor cronice precum hipertensiunea arterială, afecțiuni cardio-vasculare, diabet sau epilepsie, nerespectarea tratamentului poate să aibă consecințe grave [4]. Neluarea la timp a medicamentelor poate duce la agravarea bolii și apariția complicațiilor, având ca efect scăderea calității vieții și crescând riscul de deces. Organizația Mondială a Sănătății a raportat în anul 2012 că peste 86% din decesele din Europa au fost datorate bolilor cronice [3]. Conform unui studiu realizat de această organizație în anul 2019, problemele cardicace constituie principala cauză a deceselor, numărul acestora crescând de la 2 milioane în anul 2000, la aproape 9 milioane in anul 2019 [5].

În tratamentul bolilor cronice este foarte important ca medicul curant să știe dacă eșecul tratamentului este o consecință a nonadeziunii pacientului la tratamentul prescris (nerespectarea orarului medicamentos) sau acel medicament nu are efect în cazul pacientului respectiv [3]. Pentru tratarea bolilor cronice există o paletă largă de medicamente, medicul putând astfel selecta prin monitorizarea tratamentului medicamentul potrivit pentru pacient [3].

În cadrul bolilor infecțioase, omiterea unor doze medicamentoase și nerespectarea orarului pot duce la apariția unor tulpini microbiene multirezistente la antibiotice. Un caz concret este Tuberculoza, cea mai întâlnită boală infecțioasă la nivel mondial, fiind estimat că o treime din populația globului este infectată [6]. Incidența Tuberculozei cu tulpini multidrog-rezistente și extrem de rezistente la medicamente este în creștere, ceea ce determină o povară sanitară imensă și mortalitate înaltă [6].

Așadar, respectarea programului medicamentos conform prescripției doctorului este esențială. Aplicația Android dezvoltată de mine oferă o soluție în vederea monitorizării tratamentului.

În cadrul aplicației dezvoltate, utilizatorul poate să își adauge medicamentele pe care trebuie să le ia. Odată cu adăugarea medicamentelor va fi posibilă și setarea programului de medicație, fiind specificată prima oră din zi la care medicamentul va trebui luat, după care va fi specificată frecvența (intervalul dintre două doze consecutive). De fiecare dată când un medicament va trebui luat, utilizatorul va fi înștiințat, fie printr-o notificare, fie prin intermediul unei alarme.

Pentru a realiza o verificare mai detaliată cu privire la luarea medicamentelor, utilizatorul va putea să conecteze aplicația la un ceas și la un dozator de medicamente. Dozatorul va elibera pastila ce trebuie luată și va gestiona numărul de medicamente de fiecare tip. Ceasul va verifica dacă utilizatorul a dus mâna la gură pentru luarea pastilei.

Tot prin intermediul aplicației, pot fi adăugați supraveghetori ce primesc SMS-uri pentru a fi înștiințați dacă medicamentul nu a fost luat sau în cazul în care utilizatorul deține și un dozator, dacă au rămas cel mult 3 pastile într-un anumit compartiment al dozatorului.

Aplicația conține și o secțiune în care se poate observa ce medicamente au fost luate în fiecare zi și orele la care acestea au fost luate.

Această aplicație a fost creeată pentru persoanele ce își doresc îmbunătățirea stării de sănătate, conștientizând că urmarea tratamentului poate să le ajute în acest sens și sunt dispuse să coopereze.

2. SPECIFICAȚII

Specificațiile unui sistem descriu serviciile oferite de acesta și constrângerile legate de funcționare [7]. Astfel, sunt prezentate cerințele legate de un anumit sistem ce reflectă necesitățile clienților. Cerințele sistemului trebuie să fie clare, lipsite de ambiguitate, ușor de înțeles și complete. Lipsa preciziei în specificarea cerințelor unui sistem este unul dintre principalele motive ce duc la eșecul unui proiect software. Neînțelegerea corectă a cerințelor poate duce la întârzieri în livrarea sistemului și creșterea costurilor.

Cerințele unui sistem software se pot clasifica în cerințe funcționale si non-funcționale [7].

Cerințele funcționale ale unui sistem descriu ce trebuie să ofere sistemul și cum trebuie să reacționeze acesta în anumite situații [7]. Aceste cerințe depind de modul în care sistemul software este dezvoltat și de posibilii utilizatori. În principiu, cerințele funcționale ale unui sistem trebuie să fie complete (este necesară definirea tuturor serviciilor sistemului) și consistente (cerințele nu trebuie să aibă definiții contradictorii).

Cerințele non-funcționale se referă la constrângeri ale funcțiilor și ale serviciilor sistemului [7]. Aceste constrângeri pot fi legate de performanță, securitate sau disponibilitate. De asemenea, pot fi definite constrângeri legate de capacitățile dispozitivelor de intrare/ieșire. Cerințele non-funcționale specifică constrângerile legate de întreg sistemul. Neîndeplinirea acestora poate face ca întregul sistem să nu poată să fie folosit.

Specificațiile unui sistem presupun descrierea cerințelor funcționale și non-funcționale astfel încât acestea să fie înțelese de utilizatorii sistemului care nu au cunoștințe tehnice. Va fi specificat doar comportamentul exterior al sistemului, fără a include elemente de arhitectură sau design. Așadar, nu se vor folosi notații formale, ci se va folosi limbajul natural.

2.1. APLICAȚII SIMILARE

Pentru a realiza analiza de piață am descris trei aplicații asemănătoare, dintre cele cu cel mai mare numar de descarcari pe platofrmele de tip app-store ale sistemelor de operare mobile corespunzatoare: *Medisafe Pill & Med Reminder*, My Therapy: Medication Reminder și Pills Time Tracker & Reminder, după care le-am comparat cu aplicația dezvoltată de mine.

Medisafe Pill & Med Reminder este o aplicație prin intermediul căreia utilizatorul are posibilitatea urmăririi medicației. Acesta poate să își adauge noi medicamente în listă, putând să seteze ora la care să ia medicamentul și intervalul dintre două doze. De asemenea, utilizatorul trebuie să specifice data la care va începe tratamentul și poate să adauge și durata acestuia. Există posibilitatea de a personaliza pictograma tratamentului, alegând atât forma, cât și culoarea, după cum putem observa în figura 2.1.

După ce medicamentul a fost adăugat, utilizatorul poate să adauge și numărul de pastile pe care le deține. Cu fiecare doză luată, acest număr scade. Ora la care trebuie luat un anumit medicament poate să fie modificată manual de către utilizator.

În momentul în care este primită notificarea, utilizatorul are trei optiuni: poate să apese pe butonul de luare medicament, poate să amâne tratamentul pentru o perioadă de maxim 30 de minute, sau poate să sară peste doza respectivă, conform figurii 2.2. Dacă este aleasă ultima opțiune, va trebui specificat motivul pentru care medicamentul nu a fost luat. Utilizatorul poate să aleagă dintr-o listă de motive deja existente.

Aplicația oferă și posibilitatea de a gestiona datele de contact ale doctorilor. Utilizatorii pot să salveze informații precum numele doctorului, specialtiatea, numărul de telefon, adresa de email și adresa cabinetului. Se poate realiza asocierea medicamentelor înregistrate cu doctorul ce le-a prescris. Pe lângă medicație și doctori, utilizatorul poate să adauge și notițe cu simptomele sau pentru a-și aminti alte informații medicale.

Aplicația poate fi utilizată fără a crea un cont, dar există și o versiune premium în care utilizatorul beneficiază de mai multe opțiuni.

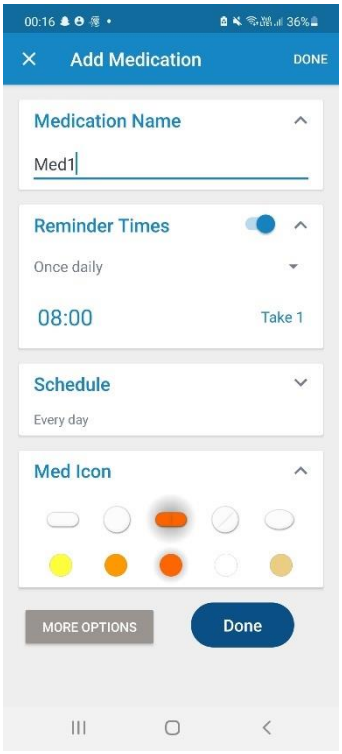


Figura 2.1 Adăugarea unui medicament în cadrul aplicației *Medisafe Pill & Med Reminder*

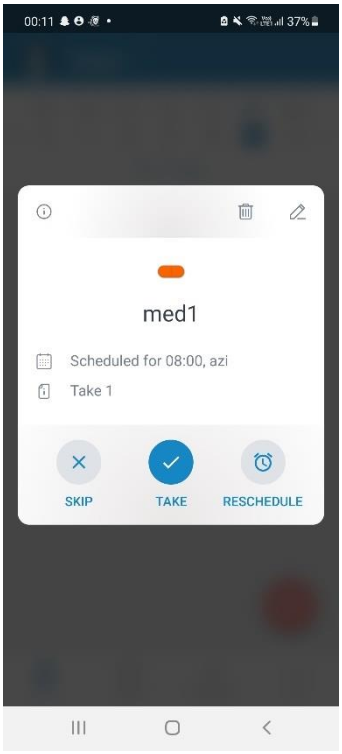


Figura 2.2 Confirmarea luării medicamentului din cadrul aplicației Medisafe Pill & Med Reminder

My Therapy: Medication Reminder, avand interfața principală a aplicației prezentată în figura 2.3**,** este utilizată pentru a reaminti luarea medicației, pentru a gestiona programările la doctor și pentru a vizualiza obiectivele zilei curente. După adăugarea numelui medicamentului, se alege tipul medicamentului medicației (tablete,

capsule, tratament injectabil, sirop etc.). Apoi se poate alege frecvența (o dată pe zi / de două ori pe zi / altă frecvență), orele și dozajul. Înainte de salvarea medicamentului, aplicația cere și cantitatea de pastile deținută de utilizator.

Cu fiecare medicament luat, scade numărul de pastile disponibil. Când numărul de pastile scade sub o anumită cantitate setată de utilizator, aplicația va trimite o notificare. Orice informație legată de medicament poate să fie editată.

În cadrul aplicației există și opțiunea de adăugare a unui supraveghetor, după cum se poate observa în figura 2.4. Acesta poate să urmărească medicația, numărul de pastile rămase și tratamentul. Conexiunea cu supraveghetorul se realizează prin transmiterea unui cod. Aplicația poate fi utilizată și fără a crea un cont.

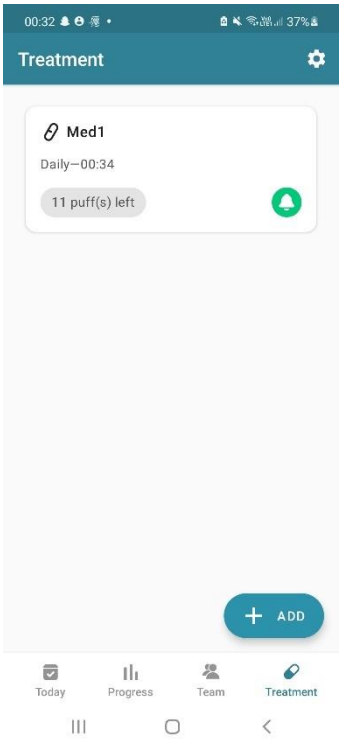


Figura 2.3 Pagina principală a aplicației *My Therapy: Medication Reminder*

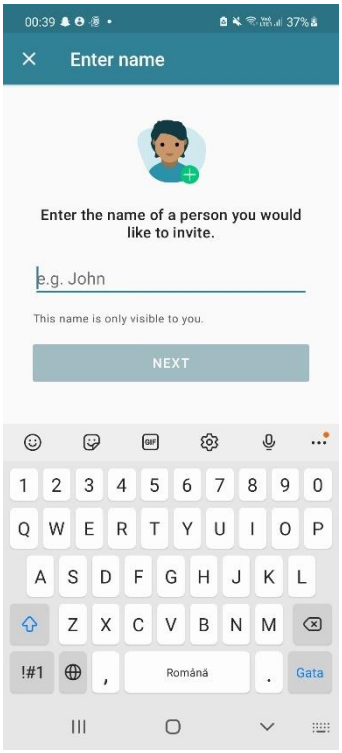


Figura 2.4 Adăugarea unui supraveghetor în cadrul aplicației *My Therapy: Medication Reminder*

Pills Time Tracker & Reminder este o aplicație cu ajutorul căreia utilizatorul are posibilitatea să adauge notificări pentru urmărirea medicației, a programului de mese și a programărilor la doctor. La adăugarea unui nou medicament, se cere numele și pentru ce este folosit, după care se alege o pictogramă cu tipul medicației (tablete, capsule, tratament injectabil, sirop etc.), culoarea medicamentului și opțional o poză, conform figurii 2.5. De asemenea, trebuie menționată data începerii tratamentului, data încheierii tratamentului și detalii cu privire la intervalul de timp.

În pagina principală a aplicației se pot vedea medicamentele luate în ziua curentă, iar în partea superioară se află o pictogramă ce redirecționează utilizatorul către un calendar, pentru a-și putea urmări istoricul. Utilizatorul are posibilitatea de a-și adăuga orele de masă, acestea fiind un posibil punct de reper pentru luarea medicamentelor și, respectiv, setarea notificărilor.

Prin intermediul aplicației pot fi monitorizate și mesele zilei. Pe lângă adăugarea unui program de mese, in cadrul versiunii premium pot fi adăugați și doctori, după cum se poate observa în figura 2.6.

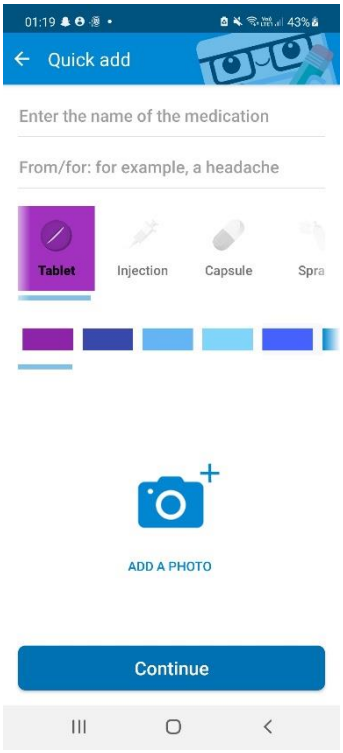


Figura 2.5 Adăugarea unui medicament nou în cadrul aplicației *Pills Time Tracker & Reminder*

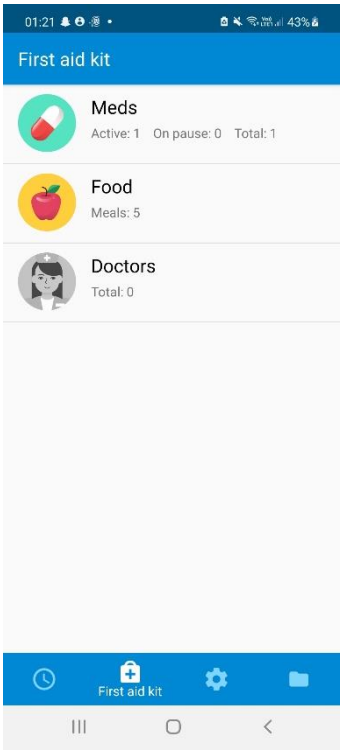


Figura 2.5 Elementele ce pot fi monitorizate în cadrul aplicației *Pills Time Tracker & Reminder*

În tabelul 2.1. Putem observa că aplicația *Time For Pills* dezvoltată de mine conține majoritatea funcționalităților oferite de celelalte aplicații existente pe piață. Similar cu aplicațiile *Medisafe Pill & Med Reminder, My Therapy: Medication Reminder* și *Pills Time Tracker & Reminder,* utilizatorii se pot autentifica, pot urmări tratamentul medicamentos și pot primii notificări în momentul în care un medicament trebuie luat. De asemenea, există posibilitatea de a seta și alarme, opțiune ce poate fi întâlinită și la aplicația *My Therapy: Medication Reminder.* Spre deosebire de aplicațiile *Medisafe Pill & Med Reminder* și *Pills Time Tracker & Reminder* aplicația mea oferă și posibilitatea adăugării unui supraveghetor.

Aplicația mea aduce în plus posibilitatea interacțiunii cu un ceas și cu un dozator de medicamente. Astfel, se poate verifica cu ușurință dacă medicamentul a fost luat, îmbunătățind urmărirea tratamentului medicamentos. Prin intermediul dozatorului este posibilă și contorizarea numărului de pastile disponibile.

Caracteristici	**Medisafe Pill & Med Reminder**	**My Therapy: Medication Reminder**	**Pills Time Tracker & Reminder**	***Time For Pills***
Link store	Magazin Play	Magazin Play	Magazin Play	-
Notă store	4.7 / 5	4.7 / 5	4.8/ 5	-
Nr. Instalări	1mil.+	5mil.+	100K+	-
Nr. Ratinguri	219K	120K	4K	-
Autentificare utilizatori	✓	✓	✓	✓
Utilizare aplicație fără a crea cont	✓	✓	✓	-
Notificări aplicație	✓	✓	✓	✓
Alarme	-	✓	-	✓
Urmărirea tratamentului medicamentos	✓	✓	✓	✓
Adăugarea unui supraveghetor	-	✓	-	✓
Interacțiunea cu un ceas	-	-	-	✓
Interacțiunea cu un dozator de medicamente	-	-	-	✓

Tabelul 2.1. Analiză comparativă între aplicația *Time For Pills* ce a fost dezvoltată de mine și alte aplicații deja existente pe piață

2.2. SPECIFICAȚIILE SISTEMULUI

Sistemul are următoarele cerințe funcționale:

- Pentru folosirea aplicației un utilizator are posibilitatea să își creeze un cont și să se autentifice.
- Utilizatorul poate să urmărească ce medicamente mai are de luat în ziua respectivă, fiind afișate numele acestora, ora la care vor fi luate și culoarea pastilei.
- Se poate urmări istoricul tratamentului medicamentos pe zile, în fiecare zi fiind prezentate numele medicamentelor din ziua respectivă, ora la care trebuiau luate și dacă au fost luate.

- Un utilizator are posibilitatea să adauge un nou medicament prin completarea unui formular în care trebuie să specifice numele medicamentului, culoarea acestuia, ora de referință, intervalul dintre două doze, tipul alarmei, data începerii tratamentului și, opțional, data finalizării acestuia.
- Un utilizator va putea edita datele medicamentelor.
- Un utilizator poate să șteargă medicamentele.
- Un utilizator poate să primească notificări și alarme prin intermediul aplicației, pentru a-i aminti să ia un anumit medicament.
- Urmărirea tratamentului medicamentos este posibilă prin confirmarea luării medicamentului după pornirea alarmei sau după apariția notificării.
- Utilizatorul are posibilitatea să conecteze aplicația la un ceas ce va detecta dacă medicamentul a fost luat verificând dacă acesta a dus mâna la gură.
- Utilizatorul are posibilitatea să realizeze conexiunea cu un dozator ce va elibera medicamentele ce trebuiesc luate la o anumită oră și va verifica numărul de pastile rămase.
- Va fi afișată o poză cu pastilele rămase în dozator.
- Se vor putea adăuga supraveghetori care vor primi mesaje în cazul in care utilizatorul nu a luat un medicament sau dacă au rămas cel mult 3 pastile într-un anumit compartiment al dozatorului de medicamente.

Sistemul are următoarele cerințe non-funcționale:

- Pentru ca un utilizator să poată utiliza sistemul acesta trebuie să dispună de conexiune la internet.
- Aplicația poate fi instalată pe un dispozitiv Android cu versiunea minimă API 23 (Android 6)

3. ARHITECTURA SISTEMULUI

Arhitectura unui sistem este necesară pentru a înțelege modul în care un sistem este organizat, prezentând structura generală a acestuia [7]. Proiectarea arhitecturii unui sistem reprezintă legătura principală dintre implementare și specificații, prin definirea diferitelor componente ale sistemului și relațiile dintre ele.

În practică, există o suprapunere seminificativă între specificațiile sistemului și arhitectură. Specificațiile nu includ niciun detaliu de implementare. Descompunerea arhitecturală este necesară pentru a structura și organiza specificațiile.

Arhitectura software poate fi folosită ca un plan de design pentru negocierea cerințelor sistemului și pentru a structura discuțiile cu clienții și dezvoltatorii [8]. De asemenea, arhitectura sistemelor reprezintă un instrument esențial pentru managementul complexității [8].

3.1. DIAGRAMA UML A CAZURILOR DE UTILIZARE

Diagrama cazurilor de utilizare modelează comportamentul sistemului. Această diagramă descrie interacțiunea actorilor cu diferite cazuri de utilizare, oferind o imagine de ansamblu a aplicației [9]. Actorii sunt reprezentați de utilizatorii aplicației. Cazurile de utilizare sunt reprezentate prin elipse sau cercuri, descriind funcțiile pe care sistemul le îndeplinește.

Între cazurile de utilizare pot sa fie stabilite diverse relații, precum relația de incluziune și cea de extindere.

Relația de incluziune, specificată prin cuvântul cheie <<include>>, presupune ca un caz de utilizare de bază să încorporeze obligatoriu cazul inclus.

Relația de extindere, specificată prin cuvântul cheie <<extend>>, presupune ca un caz de bază să încorporeze opțional cazul extins.

Cu ajutorul diagramei UML a cazurilor de utilizare, prezentată în Figura 3.1, am detaliat aspecte legate de cerințele sistemului și pe baza ei am stabilit care va fi structura aplicației.

În pagina principală, utilizatorul va putea să vadă ce medicamente mai are de luat în ziua respectivă și de asemenea, va putea să își urmărească istoricul medicației.

Apilcația va avea și o pagină prin intermediul căreia va putea fi gestionată medicația, prin adăugarea, ștergerea și editarea medicamentelor.

În momentul în care un medicament va trebui să fie luat, se va trimite o notificare. Utilizatorul poate alege dacă își dorește să primească, pe lângă notificare, și o alarmă. Se poate verifica dacă utilizatorul a luat medicamentul prin conexiunea cu un ceas, cu un dozator de medicamente sau cu ambele dispozitive. Luarea medicamentului poate să fie confirmată și manual de utilizator, prin intermediul unui buton aflat în structura notificării, sau prin fereastra generată de alarmă în aplicație.

Un utilizator va putea să își adauge supraveghetori ce vor primi mesaje în cazul în care utilizatorul nu a luat un medicament. Mesajele vor putea fi trimise la 5 minute după sunarea alarmei, moment în care medicamentul este considerat neluat.

Aplicația va putea să fie utilizată numai dacă utilizatorul este autentificat, fapt ce presupune crearea unui cont.

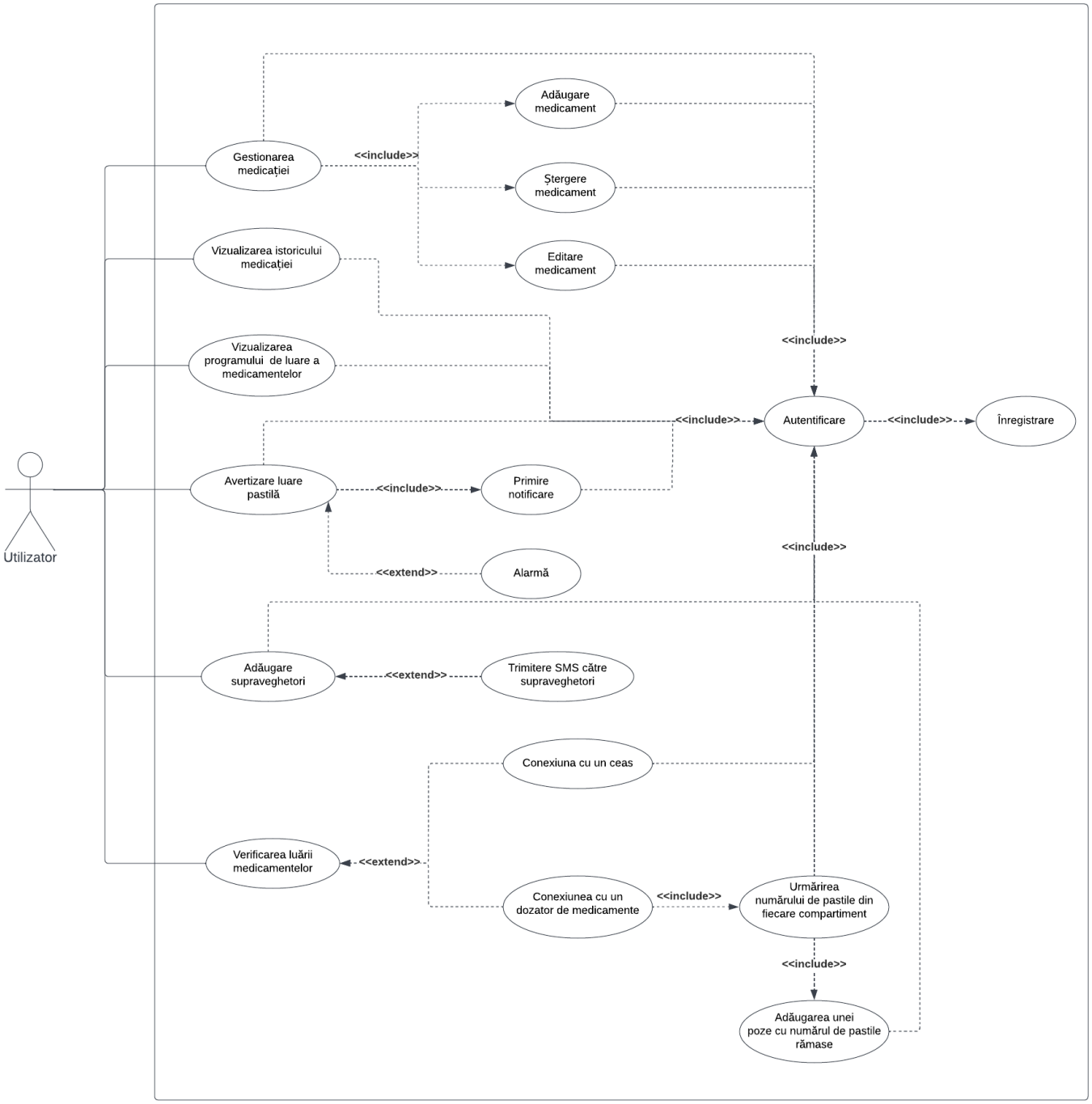


Figura 3.1 Diagrama UML a cazurilor de utilizare

3.2. PROIECTAREA BAZEI DE DATE

Pentru proiectarea bazei de date am identificat tipurile de obiecte ce vor trebui stocate: utilizatori, medicamente, supraveghetori, istoric și dozator. Structura și relațiile dintre aceste obiecte sunt prezentate prin intermediul diagramei bloc prezentatăă in figura 3.2.

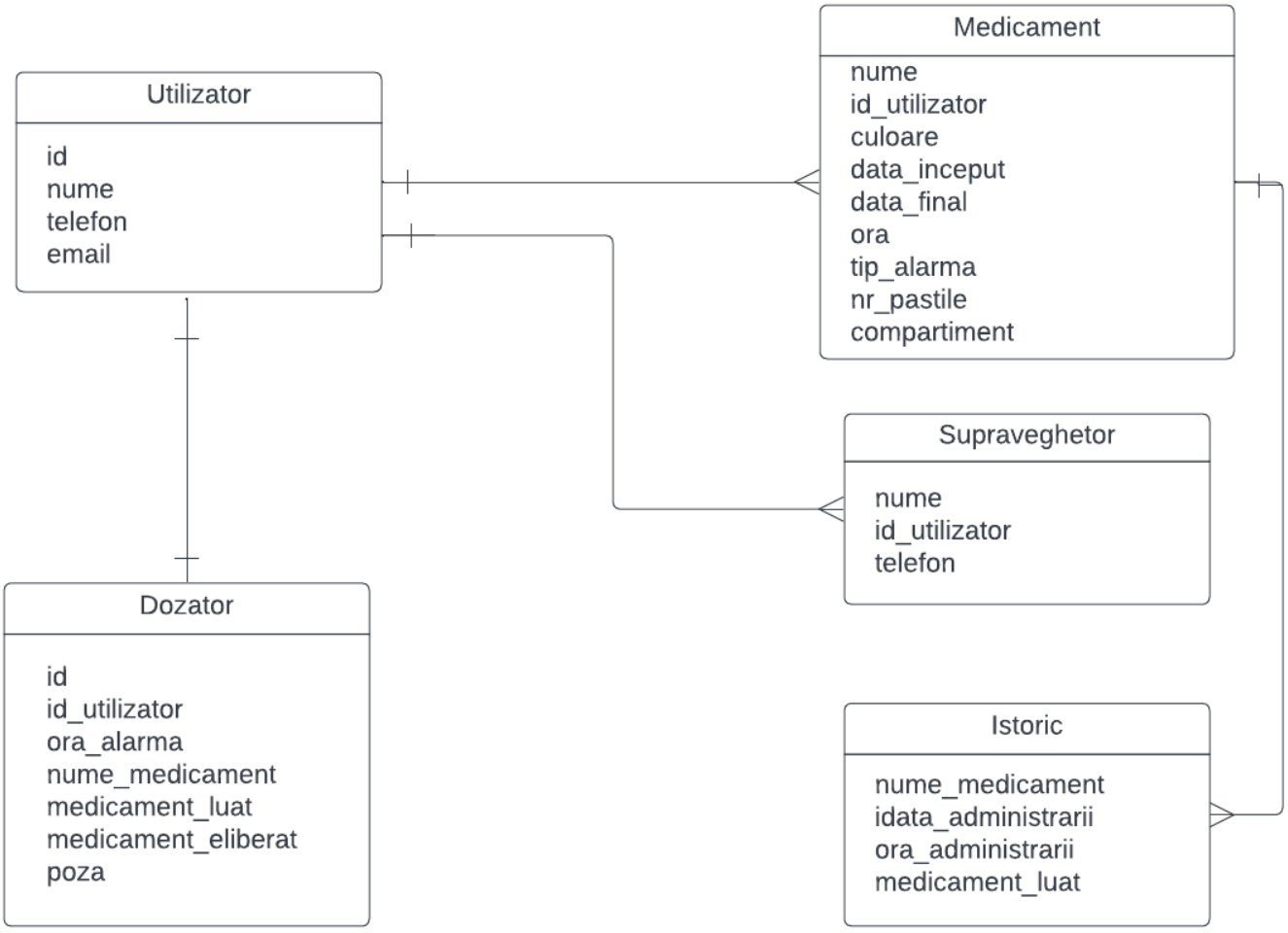


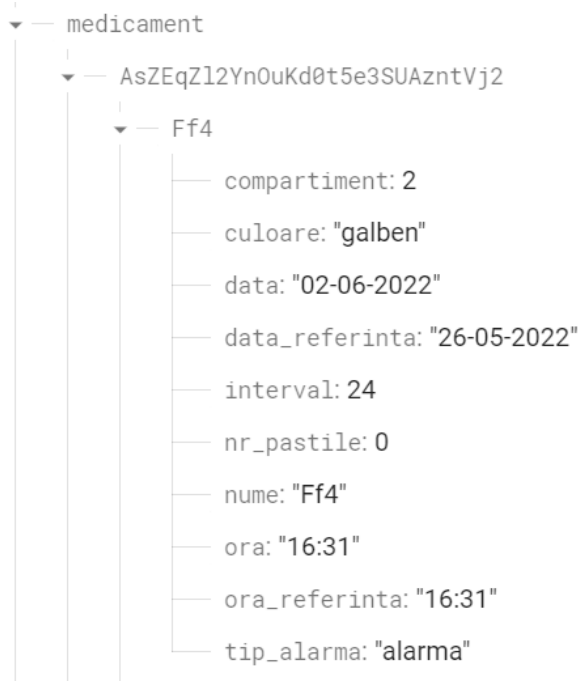
Figura 3.2 Schema bloc a bazei de date

Aceste informații vor fi stocate într-o bază de date de tip NoSQL. Sistemele NoSQL sunt baze de date non-relaționale, create pentru a putea fi stocate cantități foarte mari de date și pentru a se putea realiza o multitudine de procesări de date în paralel. Acest tip de baze de date au crescut în popularitate datorită companiilor mari precum Google, Facebook și Amazon, ce lucrează cu cantități foarte mari de date [10].

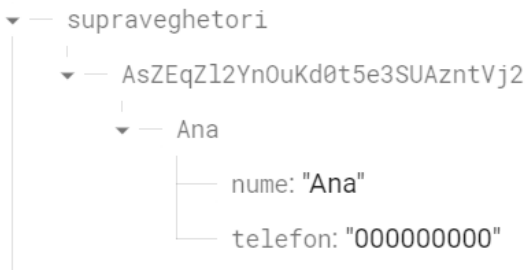
Bazele de date NoSQL oferă posibilitatea de a stoca și de a extrage datele mai rapid și oferă o flexibilitate mai mare [11].

Salvarea obiectelor de tip utilizator se va realiza într-un subarbore, users. Fiecare utilizator salvat are un identificator unic de tip String ca și cheie, iar datele acestuia sunt salvate sub următoarea formă:

Medicamentele unui utilizator vor fi stocate în subarborele medicament, în funcție de codul unic al utilizatorului:



Supraveghetorii vor fi stocați astfel:



În cazul istoricului, datele vor fi stocate in funcție de dată, oră și numele medicamentului pentru fiecare utilizator în parte:

Datele ce vor fi transmise către dozator au fost salvate sub următoarea formă:

4. IMPLEMENTAREA FUNCȚIONALITĂȚILOR

Aplicația a fost implementată pentru Android, acesta fiind cel mai răspândit sistem de operare la momentul de față. Aplicațiile Android pot rula și pe Chrome OS și Windows 11 [12]. Mediul de dezvoltare folosit este Android Studio, iar pentru stocarea datelor am ales baza de date Firebase.

4.1. AUTENTIFICARE ȘI ÎNREGISTRARE

Pentru a utiliza aplicația, utilizatorul trebuie prima dată să își creeze un cont. Pentru a creea contul, utilizatorul va completa un formular de înregistrare, după cum putem observa în figura 4.2.

Înainte ca utilizatorul să fie salvat în baza de date, se vor valida datele introduse de acesta. Se va verifica dacă toate câmpurile sunt completate. Dacă toate câmpurile sunt completate, se va verifica dacă adresa de email este una validă, dacă parola are mai mult de 6 caractere și dacă cele două parole introduse, parola și confirmarea parolei corespund.

În cazul în care utilizatorul are deja un cont, acesta se va putea autentifica din pagina prezentată în figura 4.1.

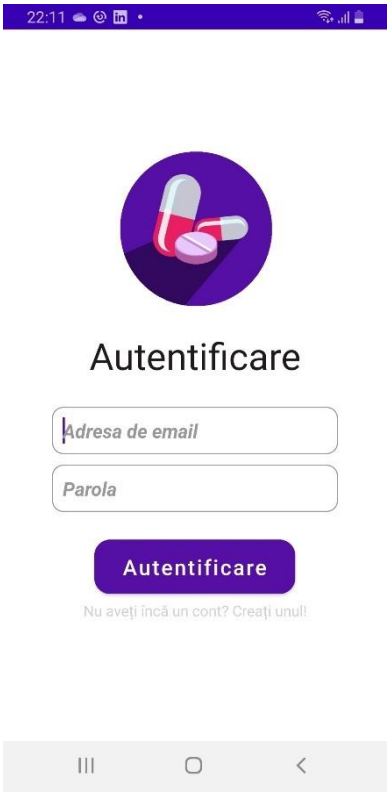


Figura 4.1 Autentificarea

Figura 4.2 Înregistrarea

4.2. NAVIGAREA FOLOSIND FRAGMENTE

Pentru navigarea între pagina principală, pagina de medicație și pagina de setări am folosit BottomNavigationView, oferită de com.google.android.material. Pentru a putea folosi această componentă, în fișierul build.gradle am adăugat următoarea dependință: 'com.google.android.material:material:1.7.0-alpha01'.

Pentru a realizeaza navigarea între cele 3 pagini ale aplicației am creeat meniul menu.xml. Acesta conține elementele ce vor fi afișate în bara de navigare.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item
            android:id="@+id/program"
            android:icon="@drawable/ic_baseline_calendar_month_24"
            android:title="Program"
            />
    <item
        android:id="@+id/medicatie"
        android:icon="@drawable/ic_add"
        android:title="Medicatie"
        />
    <item
        android:id="@+id/setari"
        android:icon="@drawable/ic_baseline_settings_24"
        android:title="Setari"
        />
</menu>
```

Secvența de cod 4.1: Meniul realizat pentru bara de navigare între pagina ce conține programul, pagina pentru gestionarea medicației și pagina de setări.

Fiecărui element al bării de navigare îi corespunde câte un fragment. Un fragment reprezintă o porțiune reutilizabilă a interfeței cu utilizatorul. Un fragment își poate gestiona și defini propriul layout. Un fragment nu poate exista pe cont propriu, depinzând de o activitate sau de un alt fragment.

Activitatea este responsabilă pentru afișarea fragmentului corespunzător în procesul de navigare între pagini. Activitatea ce se ocupă cu navigarea între fragmente în cadrul aplicației mele este PaginaPrincipalaUtilizator.

4.3. MEDICAȚIE

4.3.1. ADĂUGAREA UNUI MEDICAMENT NOU

Pentru a adăuga un medicament nou am creat un formular în care vor fi specificate numele medicamentului, culoarea acestuia, prima oră din zi la care utilizatorul dorește să ia medicamentul, intervalul dintre două doze, tipul alarmei, data începerii tratamentului și, opțional, data finalizării acestuia, după cum putem observa în figura 4.3. Dacă utilizatorul deține și un dozator de medicamente, va apărea un câmp suplimentar

unde utilizatorul va specifica compartimentul în care se află medicamentul înregistrat. Pentru a facilita gestionarea datelor am creat o clasă Medicamente ce conține toate aceste câmpuri.

Pentru a selecta ora am folosit un obiect de tipul TimePickerDialog, ce afișează prin metoda show() un ceas pentru a putea selecta ora (Figura 4.4). Preluarea orei este realizată prin intermediul metodei onTimeSet() a interfeței TimePickerDialog.OnTimeSetListener. Cel de-al doilea parametru al metodei reprezintă ora (int), iar cel de-al treilea minutul (lint).

```
TimePickerDialog.OnTimeSetListener onTimeSetListener=new
TimePickerDialog.OnTimeSetListener() {
    @Override
    public void onTimeSet(TimePicker timePicker, int i, int i1) {
        ora=i;
        minut=i1;
        String oraStr=String.format(Locale.getDefault(),"%02d:%02d",ora,minut);
        timeButton.setText(oraStr);
        medicament.setOra_referinta(oraStr);
        medicament.setOra(oraStr);
    }
};
TimePickerDialog timePickerDialog =new
TimePickerDialog(this,onTimeSetListener,ora,minut,true);
timePickerDialog.setTitle("Setati ora");
timePickerDialog.show();
```

Secvența de cod 4.2: Afișarea unei ferestre cu un ceas pentru a selecta ora la care va fi luat medicamentul.

Pentru culoarea medicamentului, tipul alarmei și intervalul dintre două doze utilizatorul poate să aleagă dintr-o listă. Pentru aceasta am folosit un obiect Spinner. Pentru adăugarea elementelor în listă este creeat un ArrayAdapter<String>. Metoda setDropDownViewResource(int) este folosită pentru setarea aspectului pe care adaptorul îl folosește pentru a afișa lista. Pentru a realiza legătura dintre ArrayAdapter și Spinner este utilizată metoda setAdapter().

```
spinner=findViewById(R.id.dropdown);
adapter=new ArrayAdapter<String>(this,R.layout.item_dropdown,items);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
spinner.setAdapter(adapter);
```

Secvența de cod 4.3: Crearea unei liste de tip dropdown pentru selectarea culorii medicamentului

Este implementată interfața AdapterView.OnItemSelectedListener pentru a suprascrie metoda onItemSelected(). În această metodă se preia elementul selectat, se verifică de ce Spinner aparține și se setează câmpul corespunzător al medicamentului. Dacă utilizatorul alege ca intervalul dintre două doze să fie o dată la x ore, va apărea un câmp suplimentar unde utilizatorul poate să scrie numărul de ore.

```
@Override
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l)
{
    String text=adapterView.getItemAtPosition(i).toString();
    if(adapterView.equals(spinner))
        medicament.setCuloare(text);
    if(adapterView.equals(spinner2))
    {
        medicament.setTip_alarma(text);
    }
    if(adapterView.equals(spinner3)){

        medicament.setInterv(text);
        if(text.equals("la x ore")){
            xh.setVisibility(View.VISIBLE);
            getX=true;
        }
        else{
            xh.setVisibility(View.GONE);
            getX=false;
        }
    }
}
```

Secvența de cod 4.4: Preluarea datelor de la meniurile de tip dropdown

Pentru a selecta data a fost folosit un obiect de tipul DatePickerDialog pentru a afișa fereastra în care se poate selecta data dorită (Figura 4.5).

```
DatePickerDialog datePickerDialog=new DatePickerDialog(AdaugareMedicament.this,
android.R.style.Theme_Holo_Light_Dialog_MinWidth,setListener2,an,luna,zi);
datePickerDialog.getWindow().setBackgroundDrawable(new
ColorDrawable((Color.TRANSPARENT)));
datePickerDialog.show();
```

Secvența de cod 4.5: Afișarea unei ferestre pentru a putea selecta data

Pentru a putea prelua data este folosit DatePickerDialog.OnDateSetListener(). Se va suprascrie metoda onDateSet() ce are ca parametrii un obiect de tipul DatePicker, anul (int), luna (int) și ziua(int). Luna selectată poate să ia valori de la 0 la 11. De aceea trebuie adăugat un 1 pentru a prelua luna calendaristică.

```
setListener=new DatePickerDialog.OnDateSetListener() {
    @Override
    public void onDateSet(DatePicker datePicker, int i, int i1, int i2) {
        i1=i1+1;
        String data=String.format(Locale.getDefault(),"%02d-%02d-%04d",i2,i1,i);
        dateButton.setText(data);
        medicament.setData_referinta(data);

    }
};
```

Secvența de cod 4.6: Preluarea datei

Înainte de a adăuga medicamentul în baza de date am realizat o verificare ca toate câmpurile să fie completate corect. În cazul în care unul din câmpurile obligatorii nu a fost completat, va apărea un mesaj în acest sens. De asemenea, data începerii tratamentului nu poate să fie o dată din trecut, iar în cazul în care data începerii tratamentului este data curentă, se va verifica ora. Dacă toate condițiile sunt îndeplinite, se va adăuga noul medicament în baza de date.

Figura 4.3 Adăugarea unui medicament nou

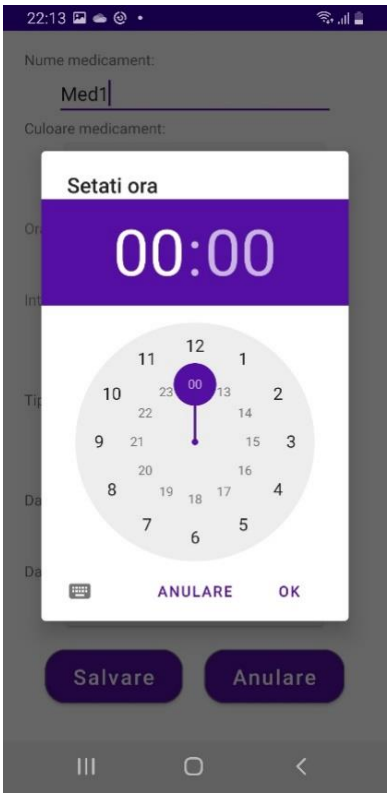


Figura 4.4 Selectarea orei

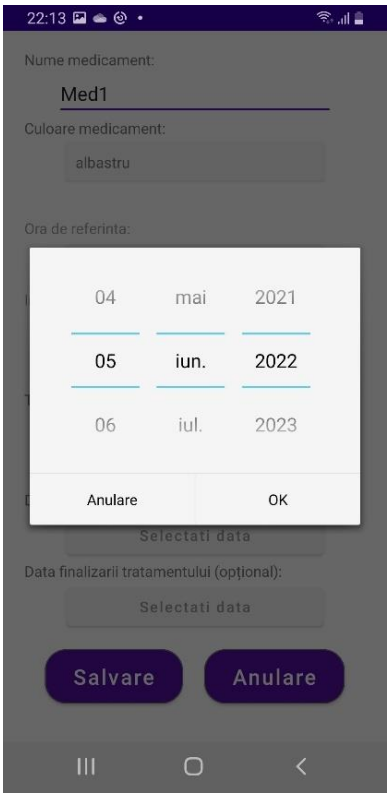


Figura 4.5 Selectarea datei

4.3.2. AFIȘAREA MEDICAMENTELOR

Pentru a afișa medicamentele înregistrate de utilizator am folosit un RecyclerView. Elementele din RecyclerView sunt aranjate de către clasa LayoutManager. Biblioteca RecyclerView include trei posibilități de aranjare. Pentru aplicație am ales LinearLayoutManager ce afișează elementele unul sub altul, pe o singură coloană.

Detaliile prezentate la afișarea listei de medicamente depind de cazul în care utilizatorul are și un dozator de medicamente sau nu. Dacă utilizatorul nu deține un dozator va fi afișat numele medicamentului, intervalul dintre două doze si ora de referință , după cum putem observa în figura 4.6. Dacă utilizatorul deține un dozator, vor apărea două câmpuri suplimentare, numărul de pastile rămase și compartimentul în care se află acestea, conform figurii 4.7.

Apoi am creat o clasă de tip Adapter și una de tip ViewHolder. Cele două clase conlucrează pentru a defini modul în care sunt afișate datele. Clasa ce extinde RecyclerView.ViewHolder este MyViewHolder. Clasa de tip Adapter creată pentru afișarea medicamentelor este MedicamenteAdapter. Aceasta suprascrie trei metode: onCreateViewHolder(), onBindViewHolder() și getItemCount().

Metoda onCreateViewHolder() este apelată de fiecare dată când este necesară crearea și inițializarea unui nou obiect de tipul MyViewHolder. Un astfel de obiect stabilește cum vor fi afișate datele unui medicament. Am ales să afișez datele prin intermediul layout-ului medicamente_afisare format dintr-un CardView ce conține o pictogramă (ImageView) și câteva câmpuri de tip TextView pentru a afișa diverse date despre medicament.

Metoda onBindViewHolder() face legătura dintre un obiect MyViewHolder și datele ce trebuiesc scrise. În cadrul acestei metode am extras din listă medicamentul curent, am scris datele în câmpurile de tip TextView, am setat culoarea pictogramei să corespundă cu cea a medicamentului și am realizat ștergerea medicamentului din baza de date prin apăsarea butonului X din colțul din dreapta sus.

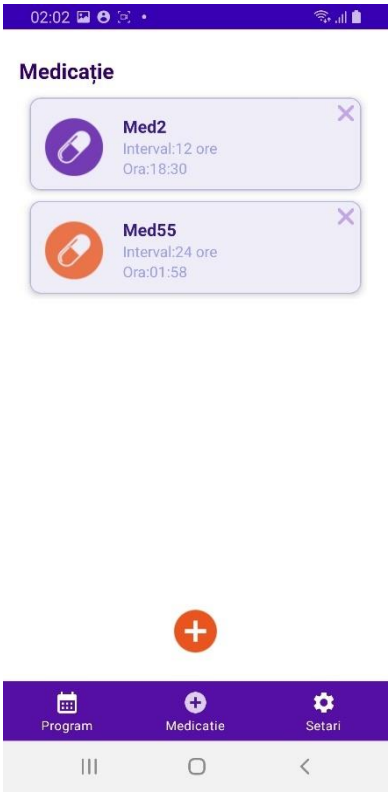


Figura 4.6 Afișarea medicamentelor pentru utilizatorii fără dozator de medicamente

Figura 4.7 Afișarea medicamentelor pentru utilizatorii cu dozator de medicamente: în dreptul fiecărui medicament este menționat compartimentul asociat și numărul de pastile disponbile ce sunt raportate de *dozatorul de medicamente*

4.3.3. EDITAREA MEDICAMENTELOR

Pentru editare utilizatorul trebuie să apese pe CardView-ul corespunzător medicamentului, după care va fi redirecționat către pagina de adăugare medicament, unde va fi însă completat formularul de adăugare.

În metoda onBindViewHolder() menționată anterior am realizat și redirecționarea utilizatorului către pagina de adăugare medicament, prin intermediul unui Intent. Pentru a ști că este vorba de editare, nu de adăugarea unui medicament nou, prin intermediul Intent-ului am transmis și numele medicamentului ce va fi editat.

În cadrul activității AdaugareMedicament, în metoda onCreate(), dacă prin intermediul Intent-ului s-a transmis și numele medicamentului, atunci se va apela metoda update(), prin care am extras din baza de date medicamentul cu numele respectiv și am completat câmpurile formularului cu datele preluate.

```
public void update(){

    String nume=getIntent().getStringExtra("nume");
    DatabaseReference database =
FirebaseDatabase.getInstance().getReference().child("medicament").child(uid);
    database.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
                Medicamente m= snapshot.child(nume).getValue(Medicamente.class);
                if(m!=null){
                medicament=m;
                numeET.setText(m.getNume());
                int spinnerPosition = adapter.getPosition(m.getCuloare());
                spinner.setSelection(spinnerPosition);
                spinnerPosition = adapter2.getPosition(m.getTip_alarma());
                spinner2.setSelection(spinnerPosition);
                spinnerPosition =
adapter3.getPosition(m.getIntervalStringUpd());
                spinner3.setSelection(spinnerPosition);
                if(m.getIntervalStringUpd().equals("la x ore"))
                    x.setText(m.getInterval());
                timeButton.setText(m.getOra_referinta());
                dateButton.setText(m.getData_referinta());
                if(m.getData_final()!=null&&!m.getData_final().equals(""))
                    dataf.setText(m.getData_final());
                if(!GestionareCutie.getCutie().equals("")){
                    compartimentET.setText(String.valueOf(m.getCompartiment()));
                }
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

Secvența de cod 4.7: Completarea automată a câmpurilor formularului de adăugare în cazul in care utilizatorul a ales să modifice datele unui medicament.

Utilizatorul va trebui doar să modifice câmpurile dorite, după care să apese pe butonul salvare. Dacă nu dorește să salveze modificările, poate să apese pe butonul anulare. În cazul în care utilizatorul face modificări, se vor dezactiva alarmele pentru ziua respectivă și se vor reseta conform cu noul program.

4.4. ISTORICUL MEDICAȚIEI

În pagina principală a utilizatorului (Figura 4.8) în partea superioară sunt afișate medicamentele ce trebuiesc luate în acea zi, iar în partea inferioară a paginii este afișat un calendar, pentru a putea vizualiza istoricul medicației.

Pentru afișarea calendarului am folosit un obiect de tip CalendarView. Utilizatorul poate să apese pe o anumită zi pentru vizualizarea istoricului din data selectată. Prin intermediul metodei onSelectedDayChange() am preluat data pe care am trimis-o mai departe prin intermediul unui Intent către activitatea AfisareIstoricZi, ce va afișa istoricul din ziua respectivă (Figura 4.9).

```
CalendarView cv = v.findViewById(R.id.calendar);
cv.setOnDateChangeListener(new CalendarView.OnDateChangeListener() {
    @Override
    public void onSelectedDayChange(@NonNull CalendarView calendarView, int i,
int i1, int i2) {
      String date=String.format(Locale.getDefault(),"%02d-%02d-%04d",i2,i1+1,i);
      Intent intent=new Intent(getContext(),AfisareIstoricZi.class);
      intent.putExtra("date",date);
      startActivity(intent);
    }
});
```

Secvența de cod 4.8: Selectarea unei date din calendar și redirecționarea către istoricul zilei respective.

În activitatea AfisareIstoricZi, în metoda onCreate(), am preluat data trimisă prin intermediul Intentului și am folosit-o pentru afișarea din baza de date a istoricului din ziua respectivă. Medicația a fost afișată cu ajutorul unui RecyclerView.

```
Intent intent=getIntent();
String data=intent.getStringExtra("date");
date.setText(data);

recyclerView = findViewById(R.id.programAzi);
String uid= FirebaseAuth.getInstance().getUid();
database =
FirebaseDatabase.getInstance().getReference().child("istoric").child(uid).child(
data);
recyclerView.setHasFixedSize(true);
recyclerView.setLayoutManager(new LinearLayoutManager(this));
list = new ArrayList<Istoric>();
medicamenteIstoricZiAdapter = new MedicamenteIstoricZiAdapter(this, list);
recyclerView.setAdapter(medicamenteIstoricZiAdapter);
database.addValueEventListener(new ValueEventListener() {
```

```
@Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        for (DataSnapshot dataSnapshot : snapshot.getChildren()){
            for (DataSnapshot dataSnapshot2 : dataSnapshot.getChildren()){
                Istoric m = dataSnapshot2.getValue(Istoric.class);
                list.add(m);}
        }
        medicamenteIstoricZiAdapter.notifyDataSetChanged();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError error) {

    }
});
```

Secvența de cod 4.9: Extragerea din baza de date a istoricului pentru ziua selectată și afișarea acestuia

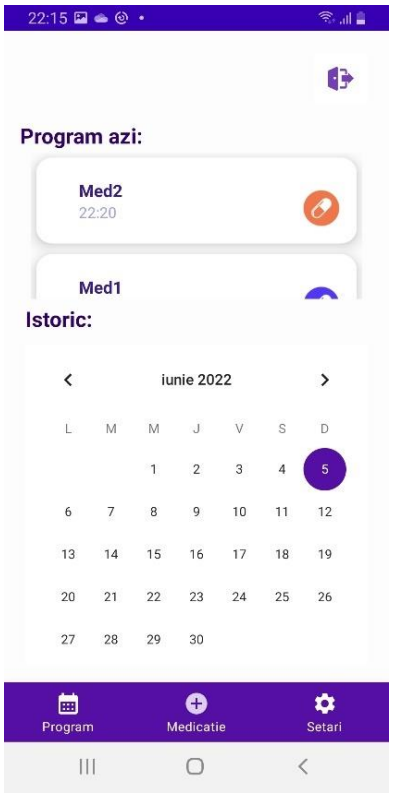


Figura 4.8 Pacgina principală a aplicației în care sunt afișate medicamentele ce trebuiesc luate în ziua respectivă și istoricul medicației

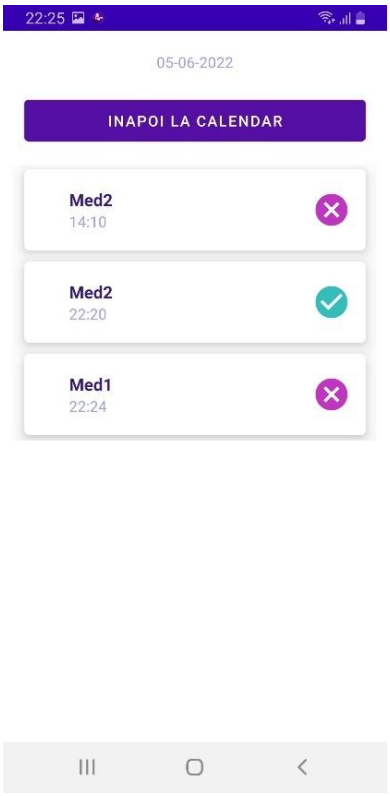


Figura 4.9 Afișarea istoricului pentru ziua selectată

4.5. ALARME

4.5.1. SETAREA ALARMELOR

Odată la 24 de ore sunt setate alarmele pentru ziua respectivă, extrăgând din baza de date medicamentele utilizatorului, și calculând pe baza acestora programul de medicație. Pentru fiecare doză ce trebuie luată este apelată funcția setareAlarma(). Pentru un medicament nou adăugat, dacă tratamentul începe în ziua respectivă, alarmele se vor seta imediat după adăugarea acestuia.

Pentru setarea alarmelor, inițial am adăugat în fișierul manifest permisiunile WAKE_LOCK, SET_ALARM și SCHEDULE_EXACT_ALARM.

Apoi am creat metoda setareAlarma() ce are ca și parametrii numele medicamentului (String), tipul alarmei (String), ora la care va porni (int) și minutul la care va porni(int). Clasa AlarmManager oferă posibilitatea de a executa acțiuni programate chiar și atunci când aplicația este închisă. Pentru setarea alarmei am folosit și un PendingIntent, cu ajutorul căruia la momentul pornirii alarmei se va lansa în execuție un Intent către AlarmeReceiver. Prin acest Intent am transmis către Receiver și numele medicamentului, tipul alarmei și codul pacientului.

```
AlarmManager alarmManager = (AlarmManager)
activity.getSystemService(Context.ALARM_SERVICE);
Intent intent = new Intent(context, AlarmeReceiver.class);
Bundle b = new Bundle();
b.putString("nume", nume);
b.putString("tip", tip);
b.putString("uid",uid);
b.putString("pastile",pastile+"");
intent.putExtras(b);
PendingIntent pendingIntent = PendingIntent.getBroadcast(context, ora+min,
intent, PendingIntent.FLAG_UPDATE_CURRENT);
alarmManager.setExact(AlarmManager.RTC_WAKEUP, cal.getTimeInMillis(),
pendingIntent);
```

Secvența de cod 4.10: Setarea unei alarme.

Am creat clasa AlarmeReceiver ce extinde clasa abstractă BroadcastReceiver. Această clasă suprascrie metoda onReceive(), ce va fi apelată la declanșarea alarmei. În această metodă verific tipul alarmei ce a fost setată.

Pentru ca utilizatorul să primească o notificare (Figura 4.10), se va crea un canal de comunicare și apoi notificarea propriu-zisă. Pentru crearea canalului am apelat metoda createNotificationChannel(). Clasa NotificationChannel este disponibilă începând cu Android 8.0 (API 26). De aceea am realizat inițial o verificare a versiunii. Apoi am creat un obiect de tip NotificationChannel ce are ca și parametrii un cod unic, un nume și importanța notificării. Am ales ca importanța notificării să fie ridicată (IMPORTANCE_HIGH). Odată ce canalul a fost trimis către NotificationManager, importanța notificării nu mai poate fi schimbată.

```
private void createNotificationChannel(){

    if (Build.VERSION.SDK_INT>=Build.VERSION_CODES.O){
        CharSequence name= "canal";
        String description= "canal pentru notificari medicamente";
        int importance= NotificationManager.IMPORTANCE_HIGH;
        NotificationChannel channel=new NotificationChannel("canal1", name,
importance);
        channel.setDescription(description);

        NotificationManager
notificationManager=c.getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Secvența de cod 4.11: Crearea unui canal de comunicare pentru afișarea notificării.

Inițializarea notificării a fost realizată cu ajutorul unui obiect NotificationCompat.Builder. Adăugarea unei pictograme, setată prin setSmallIcon() este obligatorie. Apoi am ales să setez un titlu, un text ce conține numele medicamentului ce trebuie luat, prioritatea notificării (pentru versiunile de Android până la 7.1), pagina către care utilizatorul este redirecționat atunci când apasă pe notificare și am adăugat un buton pentru confirmarea luării medicamentului. Pentru buton, am creat clasa NotificariReceiver ce extinde clasa BroadcastReceiver. În metoda onReceive(), ce va fi apelată la apăsarea butonului, am modificat în baza de date statusul medicamentului ca fiind luat, pentru data și ora la care trebuia luată pastila.

Dacă utilizatorul a ales să primească o alarmă, atunci va fi folosit un obiect de tip Ringtone. RingtoneManager oferă acces către diferite tipuri de sunete ce pot fi folosite, sunetul implicit pentru alarmă, pentru notificări sau pentru apel. Am ales tipul de suntet implicit pentru alarmă. Pentru acest tip de sunet am obținut identificatorul (Uri) pe care l-am trimis ca și parametru pentru a obține obiectul de tip Ringtone, ringtone. Pentru ca alarma să sune, pe acest obiect a fost apelată metoda play().

```
Uri alarmUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM);
ringtone = RingtoneManager.getRingtone(context, alarmUri);
ringtone.play();
```

Secvența de cod 4.12: Pornirea unui semnal sonor la declanșarea unei alarme.

Pe lângă semnalul sonor am creat și o fereastră ce afișează o imagine, un mesaj în care se specifică ce medicament trebuie luat și un buton prin care utilizatorul poate să oprească alarma din aplicație (Figura 4.11). Pentru aceasta am folosit un obiect AlertDialog.Builder. Aspectul ferestrei este dat de fișierul oprire_alarma.xml. Acesta a fost setat prin metoda setView() ce are ca și parametru un obiect de tip View. Pentru a afișa fereastra se apelează metoda show(), iar pentru a o închide dismiss().

```
AlertDialog.Builder dialogBuilder=new AlertDialog.Builder(getContext());
final View alarmView=getLayoutInflater().inflate(R.layout.oprire_alarma,null);
dialogBuilder.setView(alarmView);
```

```
dialog=dialogBuilder.create();
dialog.show();
```

Secvența de cod 4.13: Apariția unei ferestre pentru a opri alarma.

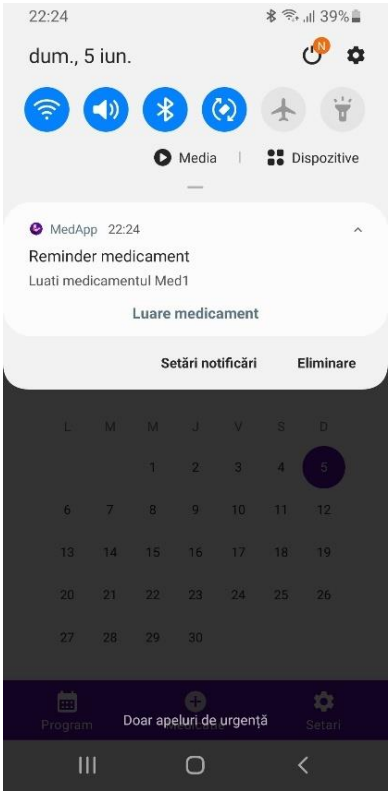


Figura 4.10 Primirea unei notificări

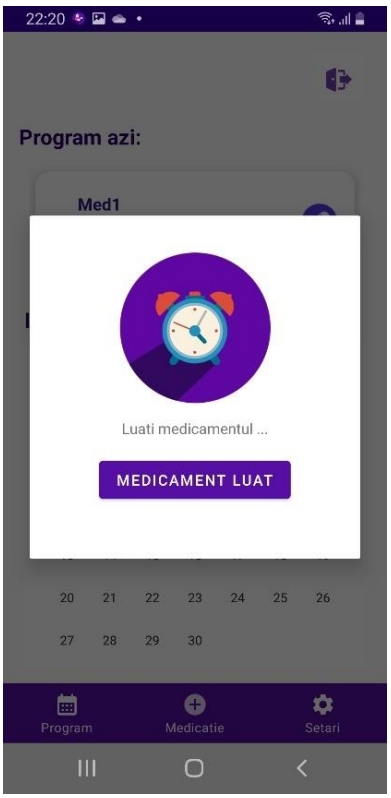


Figura 4.11 Afișarea ferestrei de oprire a alarmei

4.5.2. OPRIREA ALARMELOR

Alarmele vor fi oprite în cazul în care medicamentul a fost luat, sau la 5 minute după pornirea lor. De asemenea, în cazul în care utilizatorul modifică un medicament deja existent alarmele programate pentru ziua respectivă vor fi oprite și resetate.

Pentru a opri o alarmă am creat metoda oprireAlarmă() ce primește ca și parametrii ora (int) și minutul (int) la care alarma a fost setată. Am creat un obiect de tip PendingIntent ce are același cod cu cel folosit pentru setarea alarmei, pe care l-am dat ca și parametru metodei cancel() apelată pe un obiect de tipul AlarmManager(). De asemenea am oprit sunetul alarmei prin apelarea metodei stop() pe obiectul de tip Ringtone din clasa AlarmeReceiver.

```
public void oprireAlarma(int ora,int min) {
    if(activity!=null){
        AlarmManager alarmManager = (AlarmManager)
```

```
activity.getSystemService(Context.ALARM_SERVICE);
        Intent intent = new Intent(context, AlarmeReceiver.class);
        PendingIntent pendingIntent = PendingIntent.getBroadcast(context,
ora+min, intent, PendingIntent.FLAG_UPDATE_CURRENT);
        if (pendingIntent != null) {

            if(AlarmeReceiver.ringtone!=null)
            AlarmeReceiver.ringtone.stop();
            alarmManager.cancel(pendingIntent);

        }
    }else
        System.out.println("Null activity");
}
```

Secvența de cod 4.14: Funcția de oprire a alarmei.

4.6. TRIMITEREA SMS-URILOR

Aplicația trimite SMS-uri către o persoană de contact ce trebuie adăugată din fragmentul de setări (Figura 4.12), de la secțiunea „Adăugare persoană de contact” (Figura 4.13). Prin apăsarea butonului „Adăugare supraveghetor” va apărea un formular în care trebuie precizat numele și numărul de telefon către care vor fi trimise mesajele (Figura 4.14).

Pentru trimiterea mesajelor, în fișierul manifest trebuie adăugată permisiunea SEND_SMS. De asemenea,după ce aplicația a fost instalată, utilizatorul trebuie să își dea acordul pentru ca aplicația să poată transmite mesaje.

La 5 minute după pornirea alarmei, dacă medicamentul nu a fost luat, se verifică dacă utilizatorul are adăugați în baza de date supraveghetori. În cazul în care au fost găsiți supraveghetori, se va apela funcția trimiteSMS() ce are ca și parametrii numărul de telefon și mesajul ce va fi trimis.

```
Databasereference database =
firebasedatabase.getinstance().getreference().child("supraveghetori").child(uid)
;
database.addvalueeventlistener(new valueeventlistener() {
    @Override
    public void ondatachange(@nonnull datasnapshot snapshot) {
        for (datasnapshot datasnapshot : snapshot.getchildren()){
                Supraveghetori s = datasnapshot.getvalue(Supraveghetori.class);
                trimitesms(s.gettelefon(),mesaj);
        }
    }
    @Override
    public void oncancelled(@nonnull databaseerror error) {
    }
});
```

Secvența de cod 4.15: Căutarea supraveghetorilor în baza de date

În funcția trimiteSMS(String telefon, String mesaj), este creat un nou obiect de tip SmsManager, mySmsManager, pe care se va apela metoda sendTextMessage() ce are următorii parametrii: un String ce reprezintă numărul de telefon spre care se va trimite mesajul, un String ce reprezintă adresa service center-ului sau null pentru a folosi adresa implicită, un String ce reprezintă mesajul, un PendingIntent ce va fi lansat în execuție dacă mesajul a fost trimis cu succes sau null și un PendingIntent ce va fi lansat dacă mesajul a fost livrat.

```
public void trimiteSMS(String telefon, String mesaj){

    SmsManager mySmsManager=SmsManager.getDefault();
    mySmsManager.sendTextMessage(telefon,null, mesaj,null,null);

   }
```

Secvența de cod 4.16: Trimiterea unui mesaj către supraveghetor

Figura 4.12 Pagina de setări

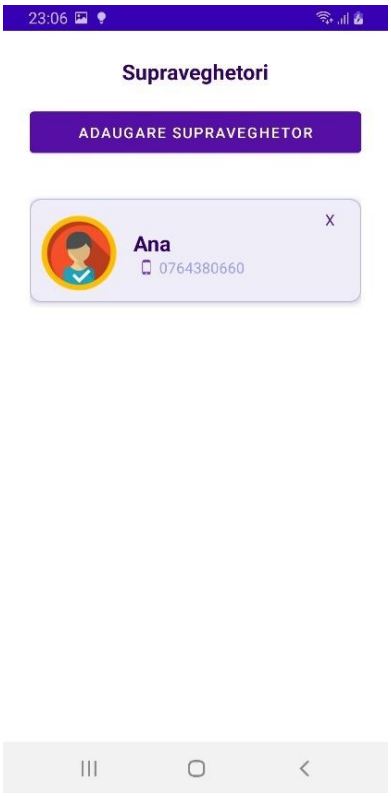


Figura 4.13 Afișarea supraveghetorilor

4.14 Formularul de adăugare a unui supraveghetor

4.7. INTERACȚIUNEA CU CEASUL

4.7.1. CONEXIUNEA BLUETOOTH

Pentru ca dispozitivele conectate la Bluetooth să transmită date unul către celălalt, trebuie realizat inițial un canal de comunicație între acestea. Un dispozitiv, în acest caz ceasul Lilygo T-Watch 2020 , va fi disponibil pentru realizarea unei cereri de conexiune. Prin intermediul aplicației mobile, telefonul va detecta dispozitivul disponibil și va transmite o cerere de conexiune. După asocierea celor două dispozitive, acestea pot să transmită și să recepționeze informații.

Folosirea Bluetooth-ului necesită declararea unor permisiuni în fișierul manifest. Permisiunea BLUETOOTH_SCAN este necesară în cazul în care aplicația dorește să găsească diverse dispozitive periferice cu care poate realiza o conexiune Bluetooth. Dacă aplicația comunică cu dispozitive Bluetooth ce au fost deja asociate, este necesară permisiunea BLUETOOTH_CONNECT. Pentru dispozitivele cu un API mai mic decât 31, în fișierul manifest se adaugă permisiunile BLUETOOTH și BLUETOOTH_ADMIN.

BluetoothAdapter este necesar pentru descoperirea altor dispozitive Bluetooth ce pot fi asociate, pentru obținerea unei liste cu dispozitive asociate, instanțierea unor obiecte de tipul BluetoothDevice și pentru a crea un obiect de tipul BluetoothServerSocket necesar în comunicarea între dispozitive. Prin apelul metodei getAdapter() pe un obiect de tipul BluetoothManager se va obține un obiect BluetoothAdapter. Dacă această funcție returnează null, atunci dispozitivul nu are opțiunea Bluetooth. Prin apelul funcției isEnabled() se verifică dacă Bluetooth-ul este activat. În cazul în care nu este activat, se va realiza o cerere în vederea obținerii permisiunii de utilizare.

```
bluetoothManager = getSystemService(BluetoothManager.class);
bluetoothAdapter = bluetoothManager.getAdapter();
if (bluetoothAdapter == null) {
    System.out.println("Dispozitivul nu are bluetooth");
} else {
    if (!bluetoothAdapter.isEnabled()) {
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.BLUETOOTH_CONNECT) != PackageManager.PERMISSION_GRANTED) {
            Log.d("Activare Bluetooth","deconectat");
            checkBtPermissions();
        } else {
            Log.d("Activare Bluetooth","conectat");
        }
    } else {
        Log.d("Activare Bluetooth","conectat");
    }
}
```

Secvența de cod 4.17: Verificarea dacă dispozitivul are activat Bluetooth-ul și cererea activării în cazul în care este oprit.

Pentru a găsi dispozitive disponibile se apelează metoda startDescovery(), care returnează o valoare de tip boolean ce indică faptul că a început procesul de căutare. Pentru a extrage informații legate de fiecare dispozitiv găsit, se va folosi un BroadcastReceiver. În momentul în care s-a gasit un dispozitiv nou, BroadcastReceiver-ul îl va adauga în listă. Lista de dispozitive va fi afișată pe ecran printr-un ListView (Figura 4.15).

```
private final BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            if(device!=null&&device.getName()!=null)
                if(!list.contains(device)){
                    list.add(device);
                    myBluetoothAdapter = new MyBluetoothAdapter(context,
R.layout.bluetooth_dev_adapter, list);
                    listView.setAdapter(myBluetoothAdapter);
                }
        }
    }
};
```

Secvența de cod 4.18: Afișarea listei cu dispozitive Bluetooth disponibile

Pentru a realiza o conexiune între două dispozitive, unul va avea rol de server și celălalt va avea rol de client. Serverul este cel care acceptă conexiuni, iar clientul este cel care transmite cererile. Aplicația mea are rolul de client, deoarece aceasta inițiază conexiunea cu ceasul.

Pentru a realiza o conexiune am creat clasa BluetoothConnectionService. Această clasă conține metoda startConnection() ce are ca și parametrii dispozitivul cu care se dorește realizarea conexiunii și un obiect de tipul UUID. Prin această metodă se va crea un nou fir de execuție de tipul ConnectThread, după care se va porni execuția acestuia. Între timp, va fi afișată o casetă ce va informa utilizatorul că se încearcă conectarea la dispozitivul selectat.

În cadrul firului de execuție creat, prin apelul metodei createRfcommSocketToServiceRecord(UUID), se va inițializa un obiect de tipul BluetoothSocket ce face posibilă conexiunea aplicației cu un BluetoothDevice. Prin apelul metodei connect(), se va încerca realizarea unei conexiuni. Dacă s-a realizat conexiunea (Figura 4.16), un nou fir de tipul ConnectedThread va fi creat și pus in execuție. Prin intermediul acestuia se pot transmite și citi date.

```
private class ConnectThread extends Thread {
    private BluetoothSocket mmSocket;
    public ConnectThread(BluetoothDevice device, UUID uuid) {
        mmDevice = device;
        deviceUUID = uuid;
```

```
 }
    public void run(){
        BluetoothSocket tmp = null;
        try {
            tmp = mmDevice.createRfcommSocketToServiceRecord(deviceUUID);
        } catch (IOException e) {
            Log.e(TAG, "ConnectThread: Could not create InsecureRfcommSocket " +
e.getMessage());
        }
        mmSocket = tmp;
        mBluetoothAdapter.cancelDiscovery();
        try {
            mmSocket.connect();
            connected(mmSocket,mmDevice);
        } catch (IOException e) {
            Log.d(TAG, "run: exception!!!"+String.valueOf(e));
        }
    }
    public void cancel() {
        try {
            mmSocket.close();
        } catch (IOException e) {
           Log.e(TAG, "cancel: close() of mmSocket in Connectthread failed. " +
e.getMessage());
        }
    }
}
```

Secvență de cod 4.19: Firul de execuție ce realizează conexiunea dintre două dispozitive Bluetooth

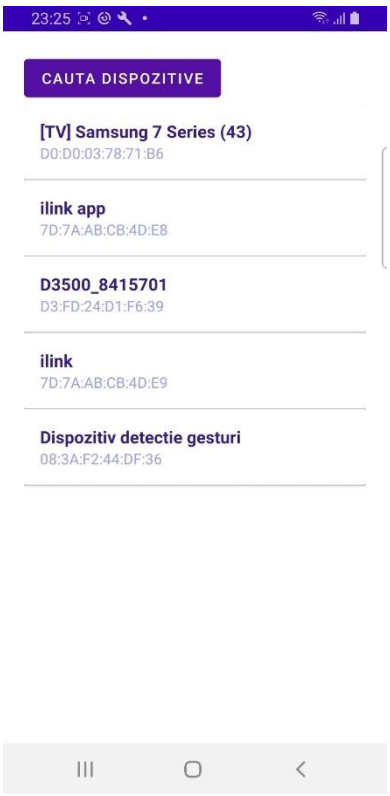


Figura 4.15 Căutarea dispozitivelor Bluetooth

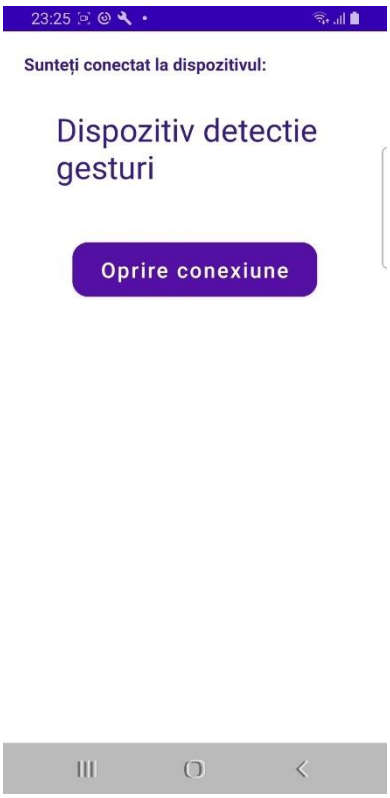


Figura 4.16 Afișarea dispozitivului Bluetooth conectat

4.7.2. LUAREA MEDICAMENTULUI ȘI OPRIREA ALARMEI

În momentul în care pornește o alarmă, va fi trimis către ceas un String cu numele medicamentului din alarmereceiver. Astfel, ceasul va ști că a pornit o alarmă și va putea să afișeze o notificare pe ecranul său. Pentru trimiterea datelor către ceas, acestea au fost inițial convertite în bytes.

```
String s=nume;
byte[] bytes = s.getBytes(Charset.defaultCharset());
BluetoothConnection.bluetoothConnectionService.write(bytes);
```

Secvență de cod 4.20: Trimiterea numelui medicamentului către ceas la declanșarea alarmei

După pornirea alarmei, în cazul în care utilizatorul deține doar un ceas, pentru următoarele 5 minute aplicația va aștepta un mesaj ce arată că utilizatorul a dus mâna la gură pentru a lua pastila.

În cazul în care Stringul „1” a fost primit de la ceas, se va apela funcția de oprire a alarmei, se va închide fereastra ce avertizează utilizatorul că medicamentul trebuie luat, se va șterge notificarea și se va scrie în baza de date că medicamentul a fost luat.

```
String s= BluetoothConnection.bluetoothConnectionService.getBrataraLuat();
System.out.println("citire bluetooth:"+s);
if(s!=null&&s.equals("1"))
{
    BluetoothConnection.bluetoothConnectionService.setBrataraLuat("0");

setariAlarme.oprireAlarma(AlarmeReceiver.getOraApel(),AlarmeReceiver.getMinApel(
));
    dialog.dismiss();
    Log.e("dialog","dismiss3");

setTrue(AlarmeReceiver.getOraApel(),AlarmeReceiver.getMinApel(),AlarmeReceiver.g
etNume());
    AlarmeReceiver.luat=true;
    AlarmeReceiver.notificationManager.cancel(100);
}
```

Secvență de cod 4.21: Oprirea alarmei și salvarea în baza de date a medicamentului ca fiind luat în cazul în care a fost primit un mesaj de la brățară.

Dacă aplicația nu primește mesajul, alarma va continua să sune timp de 5 minute, după care se va apela metoda oprireAlarma() și se va considera că medicamentul nu a fost luat. Se va anunța suparveghetorul, în cazul în care acesta este adăugat și în baza de date va fi înregistrat ca medicament neluat în istoric.

4.8. INTERACȚIUNEA CU DOZATORUL DE MEDICAMENTE

Pentru ca utilizatorul să realizeze legătura dintre dozatorul de medicamente și aplicație, acesta va trebui să introducă inițial un cod unic al dozatorului. După ce codul unic a fost adăugat într-un EditText, prin apăsarea butonului „Salvare”, se va adăuga în baza de date un nou subarbore JSON a cărui cheie va fi codul dozatorului, ce va avea următoarele câmpuri: ID-ul utilizatorului, numele următorului medicament, ora la care va suna următoarea alarmă și un câmp pentru verificarea dacă medicamentul a fost luat.

```
salvare.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String cod_cutie=t1.getText().toString();
        if(!cod_cutie.equals("")){
            Cutie c= new Cutie(uid,"","",false);
            db.child(cod_cutie).setValue(c);
            ProgramFragment.setariAlarme.trimiteMedUrmCutie(uid);
            Intent i=new Intent(GestionareCutie.this,GestionareCutie.class);
            startActivity(i);
            finish();
        }
    }
});
```

Secvența de cod 4.22: Crearea conexiunii dintre dozatorul de medicamente și aplicație

După ce sună alarma pentru a anunța utilizatorul că medicamentul trebuie luat, se va realiza o verificare dacă utilizatorul deține un dozator și un ceas, doar un dozator, doar un ceas, sau niciuna dintre ele. Dacă utilizatorul deține un dozator, când pastila va fi luată, se va modifica în baza de date câmpul corespunzător. Aplicația va verifica timp de 5 minute dacă a fost realizată această modificare prin metoda verificare_med_cutie(). Dacă dozatorul transmite că medicamentul a fost luat si utilizatorul nu deține și un ceas, se va opri alarma, și se va modifica în istoric statusul medicamentului ca fiind luat. De asemenea, se vor modifica numele medicamentului și ora la care sună următoarea alarmă în baza de date pentru a putea fi preluate de dozator.

```
if(verificare_med_cutie()){

setariAlarme.oprireAlarma(AlarmeReceiver.getOraApel(),AlarmeReceiver.getMinApel());

setTrue(AlarmeReceiver.getOraApel(),AlarmeReceiver.getMinApel(),AlarmeReceiver.getN
ume());
    AlarmeReceiver.luat=true;
    set_false();
    AlarmeReceiver.notificationManager.cancel(100);
    if(dialog!=null)
        dialog.dismiss();
    setariAlarme.trimiteMedUrmCutie(uid);
}
```

Secvența de cod 4.23: Oprirea alarmei în cazul în care dozatorul a transmis că medicamentul a fost luat

Dacă utilizatorul deține și un ceas, inițial se va verifica dacă este luat medicamentul din dozator, după care se va aștepta primirea mesajului de la ceas că mâna a fost dusă la gură. Dacă medicamentul a fost luat, alarma va fi oprită.

Pentru a trimite către dozator ora la care utilizatorul trebuie să ia pastila și numele acesteia am creat metoda trimiteMedUrmCutie() ce are ca si parametru codul unic al pacientului. Metoda extrage din baza de date medicația pentru ziua curentă, ordonată după oră. Se caută primul medicament al cărui oră este mai mare decât ora curentă și se vor modifica în baza de date câmpurile corespunzătoare.

```
public void trimiteMedUrmCutie(String uid){
    String date = new SimpleDateFormat("dd-MM-yyyy").format(new Date());
    Calendar cal = Calendar.getInstance();
    int o=cal.get(Calendar.HOUR_OF_DAY);
    int m= cal.get(Calendar.MINUTE);

    DatabaseReference dbRef=
FirebaseDatabase.getInstance().getReference().child("istoric").child(uid).child(
date);

    dbRef.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {

            int br=0;
            for(DataSnapshot snapshot3: snapshot.getChildren())
            for(DataSnapshot snapshot1:snapshot3.getChildren()){
                Istoric i=snapshot1.getValue(Istoric.class);
                if(i!=null&&br==0)
                if(i.getOraInt()*60+i.getMinInt()>o*60+m){
                    trimiteCutie(i.getNume(),i.getOraInt(),i.getMinInt());
                    Log.d("Trimitere",i.getNume()+" catre cutie, ora urm:
"+i.getOra());
                    br=1;
                }
            }
            if(br==0){
                trimiteCutie(" ",0,0);
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

Secvența de cod 4.24: Funcția de trimitere a medicamentului următor din ziua respectivă către dozatorul de medicamente

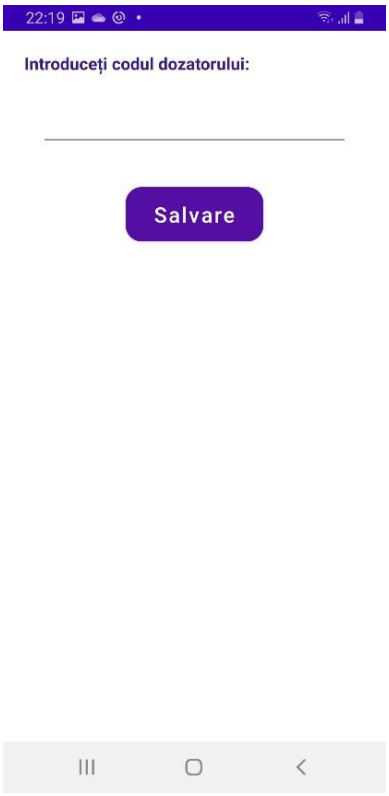


Figura 4.17 Pagina de adăugare a unui dozator

Figura 4.18 Trimiterea unui mesaj către supraveghetor pentru a-i anunța că au mai rămas 3 pastile din medicamentul Med2

În cazul în care utilizatorul deține un dozator de medicamente va fi contorizat și numărul de pastile din dozator. În cazul în care mai rămân cel mult trei pastile de un anumit fel, atunci se va transmite un mesaj în acest sens supraveghetorului. În figura 4.18 putem observa un mesaj transmis către un supraveghetor în momentul în care dozatorul mai avea 3 pastile.

Pentru a determina dacă numărul de pastile rămase este mai mic sau egal cu 3, la un minut după ce a sunat alarma, după ce medicamentul a fost eliberat de dozator, va fi aplelată metoda verificarePastile(). În cadrul acestei metode se va extrage din baza de date numărul de pastile, se va verifica dacă mai sunt cel mult 3 pastile rămase și, în cazul în care sunt se va trimite un mesaj către supraveghetor, menționându-se numărul de pastile rămase și numele medicamentului.

```
public void verificarePastile(){

    DatabaseReference database =
FirebaseDatabase.getInstance().getReference().child("medicament").child(uid).chi
ld(AlarmeReceiver.getNume()).child("nr_pastile");

    database.addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
```

```
            if(snapshot.getValue()!=null){

                int nrP= (int) snapshot.getValue();
                if(nrP<=3){
                    mesaj_supraveghetor(uid,"Au mai ramas "+nrP+" pastile din
medicamentul "+ AlarmeReceiver.getNume());
                }
            }
        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

Secvență de cod 4.25: Transmiterea unui mesaj către supraveghetor în momentul în care au mai rămas cel mult 3 doze dintr-un anumit medicament.

Dozatorul de medicamente oferă și posibilitatea realizării unei fotografii cu compartimentul în care a fost eliberată pastila după ce se presupune că aceasta a fost ridicată. Astfel se poate realiza o verificare suplimentară a luării pastilei. Aplicația va prelua această fotografie din baza de date și o va afișa în pagina de gestionare a dozatorului de medicamente, după cum poate fi observat in figura 4.19.

Fotografia este preluată de la dozatorul de medicamente din baza de date, din Firebase Storage.Inițial va fi creat un obiect de tipul StorageReference ce va conține calea către locul în care imaginea este stocată, după care se va încerca extragerea imaginii din baza de date. Deoarece acest proces poate să dureze, va fi afișat un ProgressDialog ce informează utilizatorul că se caută o imagine. Apoi se crează un obiect de tip File și se încearcă preluarea imaginii prin apelul metodei getFile() pe obiectul de tip StorageReference. În cazul în care imaginea a fost preluată cu succes, se va închide fereastra ce anunță utilizatorul că se caută o imagine și se va afișa imaginea găsită.

În cazul în care nu a fost găsită o imagine, se va închide fereastra ce anunță utilizatorul că se caută o imagine, pe ecran rămânând afișate doar codul dozatorului și butonul de oprire a conexiunii cu acesta, similar cu figura 4.20.

```
StorageReference storageReference =
FirebaseStorage.getInstance().getReference(uid+"/"+x);
try{
    pd=new ProgressDialog(GestionareCutie.this);
    pd.setMessage("Se cauta o imagine ...");
    pd.setCancelable(false);
    pd.show();

    File localFile=File.createTempFile("locFile",".jpg");
    Log.d("Gestionare cutie", "Creare imagine");

    storageReference.getFile(localFile).addOnSuccessListener(new
OnSuccessListener<FileDownloadTask.TaskSnapshot>() {
```

```
        @Override
        public void onSuccess(FileDownloadTask.TaskSnapshot taskSnapshot) {

            if(pd.isShowing()){
                pd.dismiss();
            }

            Log.d("Gestionare cutie", "Imagine gasita cu succes");
            Bitmap bitmap=BitmapFactory.decodeFile(localFile.getAbsolutePath());
            imageView.setVisibility(View.VISIBLE);
            imageView.setImageBitmap(bitmap);

        }

    }).addOnFailureListener(new OnFailureListener() {

        @Override
        public void onFailure(@NonNull Exception e) {
            if(pd.isShowing()){
                pd.dismiss();
            }
        }
    });
}catch (Exception e){
    Log.d("Gestionare cutie","eroare poza");
}
```

Secvența de cod 4.26: Extragerea unei imagini cu medicamentele rămase din baza de date și afișarea ei în pagina corespunzătoare gestionării medicamentelor din categoria setări.

Pentru întreruperea conexiunii cu dozatorul de medicamente, utilizatorul va apăsa pe butonul „Oprire conexiune”. Prin apăsarea acestui buton se vor șterge din baza de date toate câmpurile create prin realizarea conexiunii.

```
stergere.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

FirebaseDatabase.getInstance().getReference().child("users").child(uid).child("c
utie").removeValue();

FirebaseDatabase.getInstance().getReference().child(getCutie()).removeValue();
        setCutie("");
        Intent i=new Intent(GestionareCutie.this,GestionareCutie.class);
        startActivity(i);
        finish();
    }
});
```

Secvență de cod 4.26: Oprirea conexiunii cu dozatorul de medicamente în cazul în care a fost apăsat butonul corespunzător

Figura 4.19 Afișarea paginii de gestionare a dozatorului de medicamente în cazul în care este adăugată o poză ce arată dacă medicamentul a fost luat

Figura 4.20 Afișarea paginii de gestionare a dozatorului de medicamente în cazul în care în baza de date nu este adăugată poza ce arată dacă medicamentul a fost luat

5. TESTAREA

Testarea este realizată pentru a arăta faptul că un sistem îndeplinește cerințele menționate și pentru a descoperi posibilele erori ale acestuia, înainte de a fi pus pe piață [7].

Am ales să realizez testarea sistemului folosind Robo Test, un instrument de testare oferit de Firebase Test Lab prin intermediul căruia se navighează automat printre paginile aplicației pentru a observa posibilele erori din cadrul acesteia. Se va genera un set de fișiere cu operațiile ce au fost efectuate în cadrul testului și un video ce arată desfășurarea sa. Printre fișierele obținute se află o diagramă Robo Test, capturi de ecran, fișiere de tip *log* și detalii legate de performanță.

Pentru a realiza această testare este necesară generarea unui fișier .apk. Acest fișier poate să fie obținut în Android Studio, după care va fi transmis către Robo Test în Firebase. Utilizatorul poate să aleagă dispozitivul pe care se vor realiza testele, orientarea acestuia (verticală sau orizontală), durata testării și poate să predefinească anumite câmpuri pentru a vedea cum reacționează aplicația la setarea acestora.

Am ales să realizez un Robo Test pe dispozitivul Google Pixel 5e, API 30 cu orientarea verticală. Timpul testării am ales să fie de 5 minute. Pentru a mă asigura că se va realiza cel puțin o autentificare cu succes în vederea testării celorlalte funcționalități, am ales să predefinesc câmpurile corespunzătoare adresei de email și parolei, după cum se poate observa în figura 5.1.

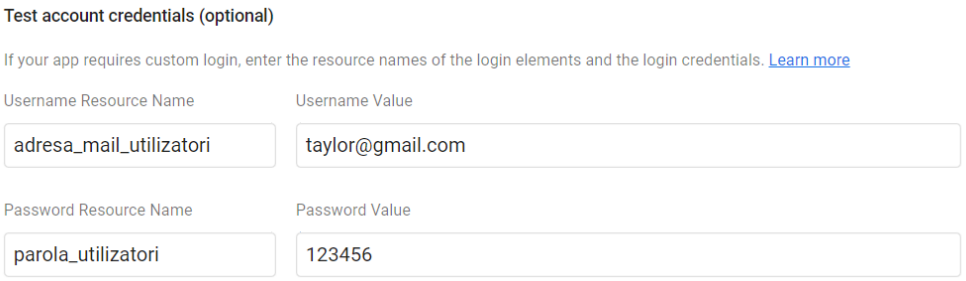


Figura 5.1 Adăugarea adresei de mail și a parolei pentru a asigura o autentificare validă în cadrul testului

Conform figurii 5.2 testul a fost finalizat cu succes și nu au fost găsite erori.

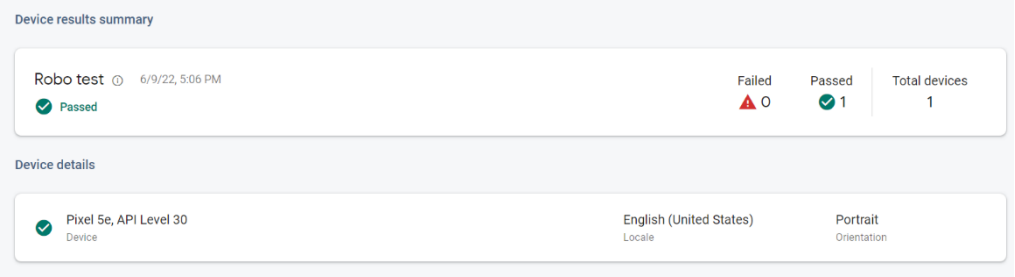


Figura 5.2 Rezultatul testării realizate cu Firebase Robo Test

În figura 5.3 sunt prezentate în detaliu acțiunile realizate în cadrul testului. Inițial, este deschisă pagina de autentificare. Este apăsat textul ce redirecționează utilizatorul către pagina de creare cont, după care se întoarce la pagina de autentificare. În pagina de autentificare vor fi completate datele predefinite, după care se va realiza conectarea. Sistemul este redirecționat către pagina principală a utilizatorului, unde se găsește calendarul și programul pentru ziua respectivă. Apoi, se comută la pagina de gestionare a medicației unde se încearcă ștergerea unui medicament. Apare o fereastră pentru confirmarea ștergerii sau anularea acesteia. Se va apăsa pe butonul de anulare, după care se apasă pe medicament și se comută către pagina de editare a medicamentului. Se apasă pe butonul de anulare, după care se va reapăsa pe medicament pentru editare. Se va modifica numele, după care se salvează. După se va încerca adăugarea unui medicament nou. Se va completa parțial formularul, după care se va încerca salvarea. Se va afișa un mesaj ce spune că toate câmpurile trebuiesc completate. Se va apăsa pe butonul de anulare. Se va comuta către pagina de setări. Din pagina de setări se va accesa pagina pentru conexiunea Bluetooth, după care se va întoarce la setări. Se va încerca adăugarea unui supraveghetor, după care se va adăuga un dozator de medicamente.

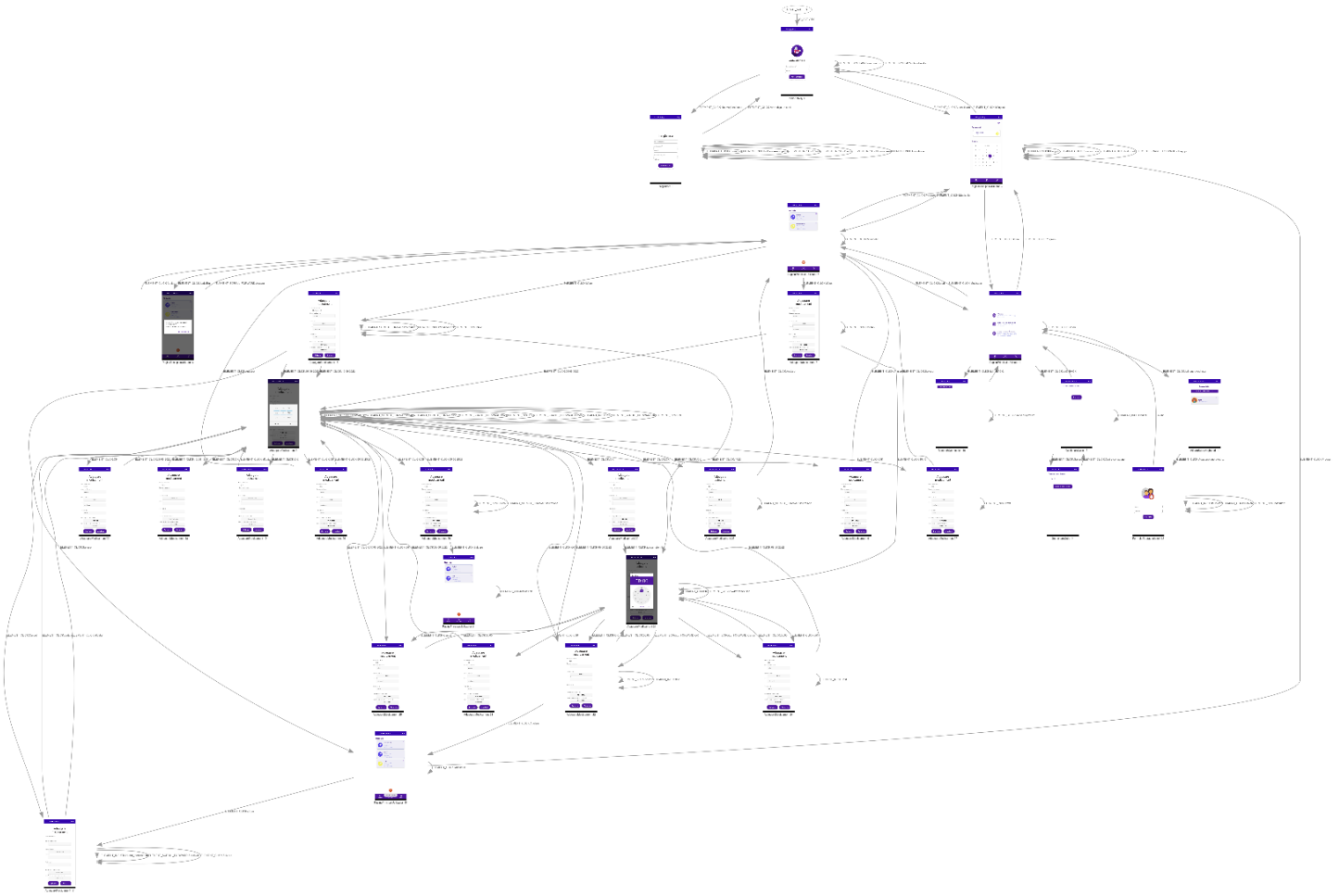
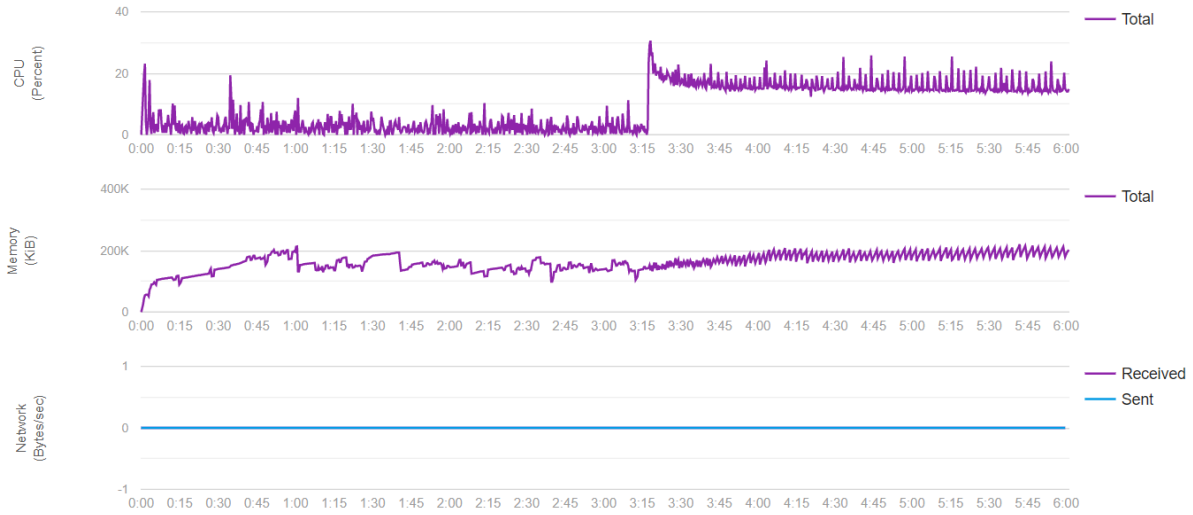


Figura 5.3 Diagrama Robo Test a testului realizat

Tot în cadrul acestui test sunt urmărite utilizarea procesorului, memoria RAM folosită și datele trimise și primite de la rețea . Aceste valori sunt afișate în graficul din figura 5.4.



Figură 5.4 Performanța aplicației

6. CONCLUZII

6.1. OBIECTIVE REALIZATE

Prin intermediul acestei lucrări am implementat o aplicație ce poate ajuta utilizatorii să își urmeze tratamentul medicamentos ambulatoriu conform indicațiilor medicului. Astfel, prin intermediul notificărilor și alarmelor aceștia își vor aminti să își ia medicamentele la timp, asigurând o mai bună eficiență a tratamentului și, respectiv o mai bună calitate a vieții.

În vederea realizării acestui proiect am parcurs mai multe etape.

Inițial, am analizat problema existentă și am venit cu o soluție, ce mai apoi a fost detaliată prin intermediul specificațiilor sistemului. În caplitolul de specificații am clasificat tipurile de cerințe în funcționale și non-funcționale. Tot în cadrul acestuia am analizat câteva aplicații concurente ce se află la momentul actual pe piață și au un număr mare de descărcări.

Pentru o mai bună înțelegere a modului în care va fi structurat sistemul, am proiectat arhitectura acestuia. Prin intermediul arhitecturii am organizat specificațiile și am stabilit cum vor fi stocate datele.

În cadrul implementării am creat o aplicație ce va rula pe sistemul de operare Android, folosind mediul de dezvoltare Android Studio. Pentru stocarea datelor am ales baza de date NoSQL Firebase.

În pagina principală a aplicației am afișat medicamentele ce vor fi luate în ziua respectivă, iar în partea inferioară se află un calendar prin intermediul căruia se poate vedea ce pastile au fost luate în fiecare zi. Apoi, am realizat o pagină de medicație prin care utilizatorul poate să își gestioneze tratamentul prin adăugarea, ștergerea și editarea medicamentelor. In cadrul paginii de setări se realizează conexiunea Bluetooth cu ceasul și conexiunea cu dozatorul. Tot în acestă pagină se pot adăuga și persoane de contact ce vor primii SMS-uri în cazul în care utilizatorul nu și-a luat medicamentele sau nu are suficiente pastile rămase.

Apoi, am ales să testez funcționalitatea aplicației folosind instrumentul de testare Robo Test oferit de Firebase. Prin intermediul acestuia s-a navigat automat printre paginile aplicației și s-a demonstrat faptul că sistemul îndeplinește cerințele menționate în subcapitolul de specificații ale sistemului.

6.2. DEZVOLTĂRI ULTERIOARE

In cadrul aplicației, la momentul de față există un singur tip de cont. Ca și dezvoltare ulterioară, se pot adăuga conturi și pentru supraveghetori. Astfel nu va mai fin necesară trimiterea SMS-urilor dintr-o aplicație exterioară, iar persoana aflată în grijă va fi mult mai ușor de monitorizat. Conexiunea dintre un utilizator și supraveghetorul său se poate realiza prin intermediul unui cod unic. Supraveghetorul va primi notificări prin intermediul aplicației în locul SMS-urilor. De asemenea va avea acces la istoricul

medicației și va putea gestiona medicamentele ce trebuiesc luate. Astfel, va putea edita informațiile medicamentelor existente și va putea adăuga medicamente noi. De asemene, va fi posibilă vizualizarea numărului de pastile rămase, în cazul în care utilizatorul are și un dozator.

O altă îmbunătățire ce poate să fie adusă aplicației este adăugarea unei secțiuni unde pot fi oferite informații despre programul de mese. Astfel, în funcție de orele la care utilizatorul ia masa să poată fi trimise înștiințări pentru luarea medicamentelor.

De asemenea, pe viitor ar putea să fie adăugate și chestionare cu posibilele reacții adverse ale utilizatorului la anumite medicamente luate. In dreptul fiecărui medicament poate să fie un buton de completare a acestor chestionare. Utilizatorul va putea bifa dintr-o listă posibilele reacții adverse.

BIBLIOGRAFIE

[1] Nancy Houston Miller, Martha Hill, Thomas Kottke, Ira S. Ockene. *The Multilevel Compliance Challenge: Recommendations for a Call to Action*. American Heart Association, 1997.
[2] Rajesh Balkrishnan, *The Importance of Medication Adherence in Improving Chronic-Disease Related Outcomes: What We Know and What We Need to Further Know*. Medical Care, Vol. 43, pp. 517-520, 2005.
[3] Adriana Melnic. *Aderența la tratament în bolile cronice netransmisibile*. https://www.viata-medicala.ro/opinii/aderenta-la-tratament-in-bolile-cronice-netransmisibile-13488, 2017. [Online; Accesat: 10.06.2022].
[4] https://www.mycirclecare.com/importance-taking-medications-time/
[5] *WHO reveals leading causes of death and disability worldwide: 2000-2019*, Site-ul oficial World Health Organization. https://www.who.int/news/item/09-12-2020-who-reveals-leading-causes-of-death-and-disability-worldwide-2000-2019, 2020 [Online; Accesat: 10.06.2022].
[6] Adam Feather,David Randall, Mona Waterhouse. *Medicina clinica*, editia a X-a, Editie aniversara, 2022.
[7] Ian Sommerville, *Software engineering*, 9th edition, Addison Wesley, 2010.
[8] Hofmeister C., Nord R., Soni, D., *Applied Software Architecture*, Addison Wesley, 2000.
[9] *Use-case diagrams*, Documentația IBM, https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case, 2022. [Online; Accesat: 20.05.2022].
[10] A. B. M. Moniruzzaman, Syed Akhter Hossain. *NoSQL Database: New Era of Databases for Big data Analytics - Classification, Characteristics and Comparison*. International Journal of Database Theory and Application,Vol. 6, No. 4, 2013.
[11] Katarina Grolinger , Wilson A Higashino, Abhinav Tiwari, Miriam AM Capretz. *Data management in cloud environments: NoSQL and NewSQL data stores.* Journal of Cloud Computing: Advances, Systems and Applications, 2013.
[12] C. Scott Brown. *What is Android? Here's everything you need to know*. Android authority, 2022. https://www.androidauthority.com/what-is-android-328076/ [Online; Accesat: 25.05.2022].
[13] Documentația Android, https://developer.android.com, 2022 [Online; Accesat: 25.05.2022].
[14] Documentatia Firebase, https://firebase.google.com/docs, 2022. [Online; Accesat: 25.05.2022].

DECLARAŢIE DE AUTENTICITATE A LUCRĂRII DE FINALIZARE A STUDIILOR *

Subsemnatul ŞTEABURDEA ŞTEFANIA,

legitimat cu CI seria TZ nr. 711483,

CNP 2990927385578

autorul lucrării APLICAȚIE MOBILĂ PENTRU MONITORIZAREA TRATAMENTULUI MEDICAMENTOS AMBULATORIU

elaborată în vederea susţinerii examenului de finalizare a studiilor de LICENȚĂ organizat de către Facultatea AUTOMATICĂ ȘI CALCULATOARE din cadrul Universităţii Politehnica Timişoara, sesiunea IUNIE a anului universitar 2021 - 2022, coordonator S.L. DR. ING. ALEXANDRU IOVANOVICI luând în considerare conţinutul art. 34 din *Regulamentul privind organizarea şi desfăşurarea examenelor de licență/diplomă şi disertație*, aprobat prin HS nr. 109/14.05.2020 și cunoscând faptul că în cazul constatării ulterioare a unor declaraţii false, voi suporta sancțiunea administrativă prevăzută de art. 146 din Legea nr. 1/2011 – legea educației naționale și anume anularea diplomei de studii, declar pe proprie răspundere, că:

- această lucrare este rezultatul propriei activități intelectuale,
- lucrarea nu conține texte, date sau elemente de grafică din alte lucrări sau din alte surse fără ca acestea să nu fie citate, inclusiv situația în care sursa o reprezintă o altă lucrare/alte lucrări ale subsemnatului.
- sursele bibliografice au fost folosite cu respectarea legislaţiei române şi a convenţiilor internaţionale privind drepturile de autor.
- această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență/diplomă/disertație.

Timişoara,

Data	Semnătura
18.06.2022	

* Declaraţia se completează *„de mână"* şi se inserează în lucrarea de finalizare a studiilor, la sfârşitul acesteia, ca parte integrantă.