Universitatea Politehnica București

Facultatea de Electronică, Telecomunicații și Tehnologia Informației

**INGINERIE SOFTWARE**

**DEZVOLTAREA APLICATIILOR PE PLATFORMA ANDROID**

**PROFESOR: STÂNCESCU ȘTEFAN**

**STUDENTI: Rădulescu Vicențiu-Mihail**

**Rădulescu Victor Cristian**

**443A**

**Cuprins:**

1.Introducere.

2. Practici de bună proiectare în android.

3. Exemplificare a unei aplicatii in Android utilizand o  baza de date pentru stocarea datelor primite de la un sistem embedded.

I. Sistemul format din Aplicatia Android, Server și Bază de date.

II. Sistemul este vazut ca aplicatie Android.

4. Concluzii

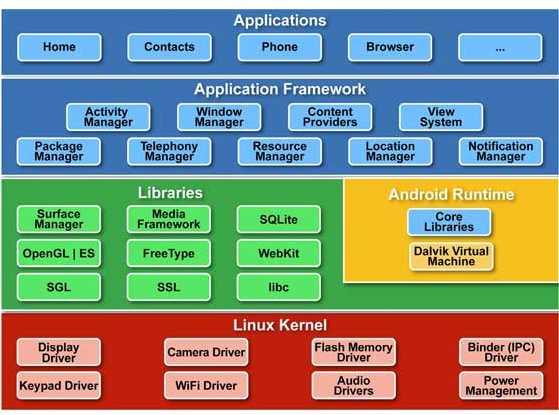
5. Bibliografie.

**Introducere**

**"Android** este o platformă software și un sistem de operare pentru dispozitive și telefoane mobile bazată pe nucleul Linux, fiind un model de arhitectură MVC.

Dezvoltarea aplicatiilor Android se face in limbajul Java, dezvoltatorul ajutandu-se de bibliotecile (API-uri) dezvoltate de Google. Se pot scrie si aplicatii in limbajul C, apoi fiind compilate in cod masina ARM si executate, dar Google nu are suport oficial pentru acest lucru.

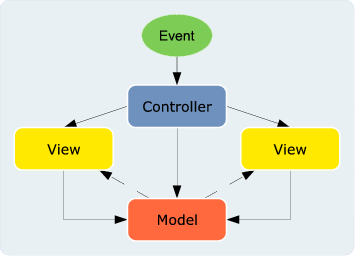
Sistemul de operare Android este o stivă de componente software care este împărțit în cinci secțiuni și patru straturi principalele cum se arată în diagrama de arhitectura de mai jos :



Schema Arhitecturi Android

**Ce este MVC?**

MVC, sau Model-View-Controller este un șablon arhitectural folosit in industria de software development(inclusiv web development). Aceasta modalitate de lucru reusește cu succes izolarea parții logice de interfața proiectului, rezultând în aplicații extrem de ușor de modificat. În organizarea MVC, modelul reprezintă informația (datele) de care are nevoie aplicația, viewerul corespunde cu elementele de interfața iar controller-ul reprezintă sistemul comunicativ și decizional ce procesează datele informaționale, făcând legătura între model și view.



Schema unei arhitecturi MVC

Practici de bună proiectare în android

Există unele practici,sfaturi pe care le poți urma în timp ce dezvolți o aplicație în Android .

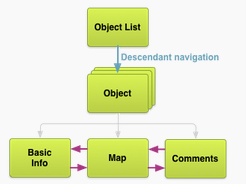
Cele mai bune proceduri,sfaturi pentru a avea o proiectare în android bună sunt legate de:interacția cu utlizatorul,perfomanța,designul,compatibilitatea, securitatea,testarea,distribuirea și monetizarea.

1.Pentru **interacție** și păstrarea utilizatorilor

Pentru a avea o aplicație ușor folosibilă trebuie să avem o interfață intuitivă,o navigație efectivă iar pentru a se păstra conexiunea între utilizator și producător sau pentru a avea mai mulți utlizatori trebuie ca aplicația ta să fie ușor căutată ( de exemplu prin google),utilizarea notificărilor,utilizarea unor conținuturi fără a mai trece prin laucherul aplicației.

Exemple de practici din această categorie:

a)Proiectarea unei navigați efective: cum să-ți planifici ierarhirea ecranelor și formele de navigare astfel încât utilizatori să poate utiliza intuitiv conținutul aplicației tale utilzând diverse tipare de navigare.



Exemplu de navigare:navigare laterală

b) Implemetarea unei navigări efective:cum să implementezi diverse tipare de navigare precum swipe views , un navigation drawer sau up navigation.

c)Notifcarea utlizatorului: afișarea unor mesaje înafara aplicației tale cu diverse scopuri.

d)Utilizarea funcționalități de căutare: de exemplu căutarea unui nume dintr-o bază de date.

e)Conținutul aplicației tale să fie căutat pe Google

2.Practici pentru interfața cu utilizatorul

Android oferă un framework flexibil pentru proiectarea UI care permite ca aplicația ta să fie suportată pentru diferite layouturi,pentru diferite dispozitive,pentru a crea widget-uri.

Exemple de practici :

a)Proiectarea pentru mai multe ecrane:aplicația trebuie să aibă un aspect plăcut pentru o gamă largă de dispozitive de la telefoane mici ,tablete la chiar televizoare.

b)Crearea unei vizualizări personalizate: crearea proprilor widget-uri UI care sunt interactive și plăcute.

c)Crearea unor Ui-uri compatibile cu vechile versiuni ale platformei android

d)Implementarea accesibilități: cum să-ți faci aplicația accesibilă utilizatorilor cu probleme de vedere sau alte probleme fizice.De exemplu utilizarea căutări vocale pentru nevăzători.

e)Administrarea UI sistemului: cum să arăți sau să ascunzi barea de stare și cea de navigare pe diferite versiuni de Android,În timp ce administrezi displayul altor componente ale ecranului.

3.Practici folosite pentru intrarea utilizatorului.

Se referă la modul prin care utilizatorul interacționează cu aplicația

a)Utilizarea gesturilor de touch (prin atingerea ecranului)

b)Manipularea intrări tastatură:fiecare text field este folosit pentru diferite joburi.De exemplu,câteva texfield-uri sunt pentru text și câteva pentru numere.Dacă este pentru numere este mai bine să afișezi tastatura când acel textfiled este focusat.Exemplu de sintaxă:

<EditText

android:id="@+id/phone"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:inputType="phone" />

Dacă acel field este o parolă.atunci ar trebui să arătăm un indiciu de parolă.astfel încât utilizatorul să poate memora ușor parola.Poate fi obținut astfel:

<EditText

android:id="@+id/password"

android:hint="@string/password\_hint"

android:inputType="textPassword" />

c)Să suporte controlere de gaming.

4. Practici pentru lucru în fundal

Folosite pentru înbunătățirea perfomanțelor aplicației tale și minimizarea consumului de energie a bateriei.

a)Rularea în fundalul unui serviciu:cum să înbunătățești perfomanțele și răspunsul trimițând conținut la un serviciu ce rulează în background(fundal).

b)Încărcarea datelor în fundal:cum să utilizezi CursorLoader pentru a interoga data fără să afecteze răspunsul UI.

c)Administrarea dispozitivului în starea de trezire :cum să folosești alarmele și lacătele pentru rularea taskurilor în starea de trezire(awake state)

5.Practici pentru perfomanță

O perfomanță mai bună în acest caz se referă la o aplicație ce rulează rapid,să răspundă bine la comenzi și să folosească cât mai puțină baterie posibilă.

Un exemplu de asemenea practică ar fi ca atunci când un dispozitiv se încarcă ,este recomandabil să updatezi setările aplicației tale asta însemnând maximizarea ratei de refresh când dispozitivul este conectat.El poate fi făcut astfel:

IntentFilter ifilter = new IntentFilter(Intent.ACTION\_BATTERY\_CHANGED);

Intent batteryStatus = context.registerReceiver(null, ifilter);

// Se încarcă / este încărcat? Full sau încă se încarcă.

int status = batteryStatus.getIntExtra(BatteryManager.EXTRA\_STATUS, -1);

// cum încarci? De la înărcător sau USB.

int chargePlug = batteryStatus.getIntExtra(BatteryManager.EXTRA\_PLUGGED, -1);

6.Practici pentru securitate

Este foarte important ca aplicația să fie secură și nu doar aplicația și ci datele aplicației precum cele de pe un server cu baze de date MySql.Securitatea poate fi crescută prin folosirea următorilor factori:

a)Folosirea stocare internă în loc de externă pentru stocarea fișierelor aplicației

b)Utilizarea de „content providers”ori de câte ori e posibilă.

c)Utilizarea SSL și HTTPS la conectarea cu web-ul.

d)Folosind permisi apropiate pentru accesarea diferitelor funcționalități ale dispozitivului.

# Exemplificare a unei aplicatii in Android utilizand o  baza de date pentru stocarea datelor primite de la un sistem embedded

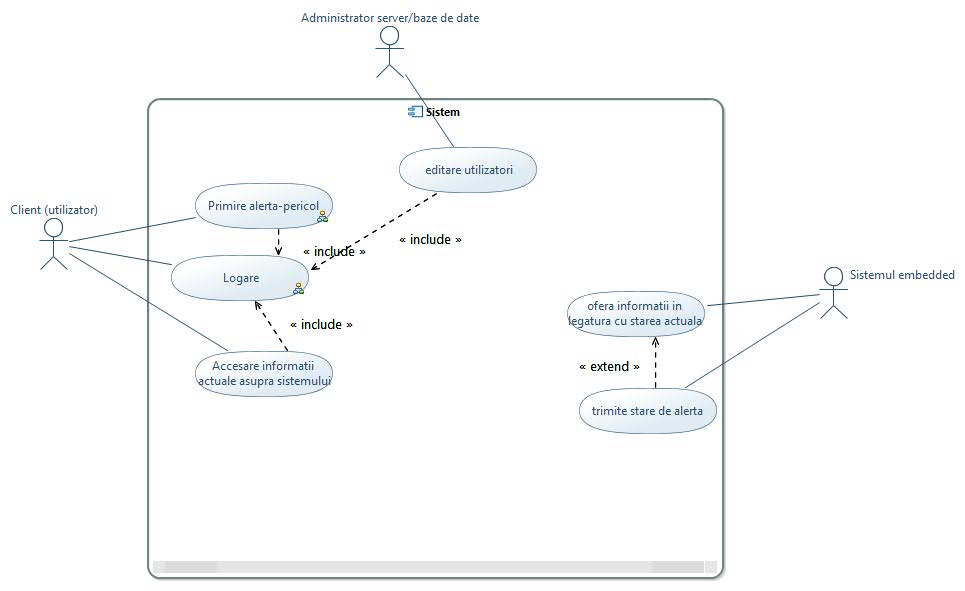
In acest paragraf se descrie cu ajutorul diagramelor UML o exemplificare de executie a unei aplicatii Android cu baze de date pe un server.

Se utilizeaza arhitectura MVC in realizarea sistemului. Controlerul gestioneaza formularele primite de la aplicatie si utilizeaza diferite clase pentru a realiza conexiunea cu baze de date si de gestiona datele. Aceste clase reprezinta modelul sistemului, ele realizand prelucrarile de date, stocarea si colectarea acestora in/din baza de date. Model de exemplu poate reprezenta clase java simple iar controlerul un servlet sau mai multe. Partea de View este data de aplicatia Android, acesta fiind modul interactiune a sistemului cu utilizatorul.

Aplicatia necesita un cont realizat pentru conectarea acesteia la baza de date, astfel putand primi notificari in legatura cu valorile monoxidului de carbon unde acesta este instalat. Utilizatorul poate afla informatii in legatura cu valorile senzorului pentru un interval definit in timp. Acesta primeste si notificari, mesaje urgente, daca se ating valorile anormale ale monoxidului de carbon.

Se ilustreaza urmatoarele descrieri:

1. Sistemul format din Aplicatia Android, Server si Baza de date.



I.1 Cazul de utilizare:**Logare**

**Rezumat:** Utilizatorul accesează sistemul (aplicația mobilă) și în activitatea de logare /pagina de logare îsi introduce informatiile de autentificare (parola si username-ul).

**Actori:** Client, Administrator

**Descrierea scenariului:**

**Preconditii:**

- persoana se afla in aplicatie

- serverul sa ruleze

**Scenariu Nominal:**

1. Sistemul solicita nume al utilizatorului si parola (view).

2. Persoana introduce numele de utilizator si parola in casutele box.

3. Persoana cere logare, se realizeaza o cerere (get/post).

4. Serverul primeste cererea (controler) si o analizeaza (prin model –beans).

5. Sistemul verifica numele de utilizator daca exista in baza de date (query).

6. Daca numele de utilizator este eronat va afisa un mesaj de eroare catre client.

7. Se verifica parola in baza de date asociate utilizatorului.

8. Daca parola nu corespunde cu cea aflata in baza de date va afisa un mesaj de eroare si cere reintroducerea parolei.

9. Daca numele si parola sunt valide se intra in activitatea ce urmareste monoxidul de carbon captat de sistemul embedded.

**Scenariu Alternativ:**

A1: Verificarea numelui de utlizator

Alegerea se face la punctul 4 din scenariul nominal. Daca numele de utilizator este eronat va afisa un mesaj de eroare si cere reintroducerea numelui de utilizator.

Cazul de utilizare se continua de la punctul 2 din scenariu nominal.

A2: Verificarea parolei

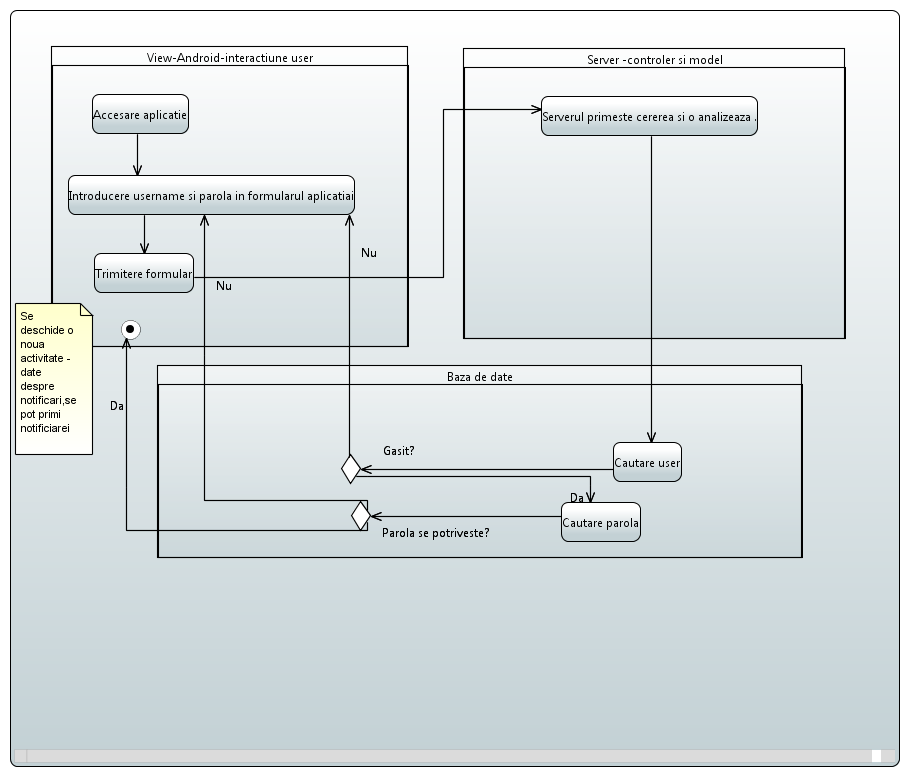
Alegerea se face la punctul 6 din scenariu nominal. Daca parola este eronata se va afisa un mesaj de eroare si se va cere reintroducerea parolei.

Cazul de utilizare se continua de la punctul 2 din scenariu nominal.

**Postconditii:**

**-** clientul sau administratorul poate vedea informatii in legatura cu datele de monoxid de carbon sau sa primeasca alerte, respectiv sa editeze utilizatorii.

Observatie: Similar se realizeaza toate conexiunle aplicatiei la baza de date, pentru a extrage informatii in legatura cu senzor.



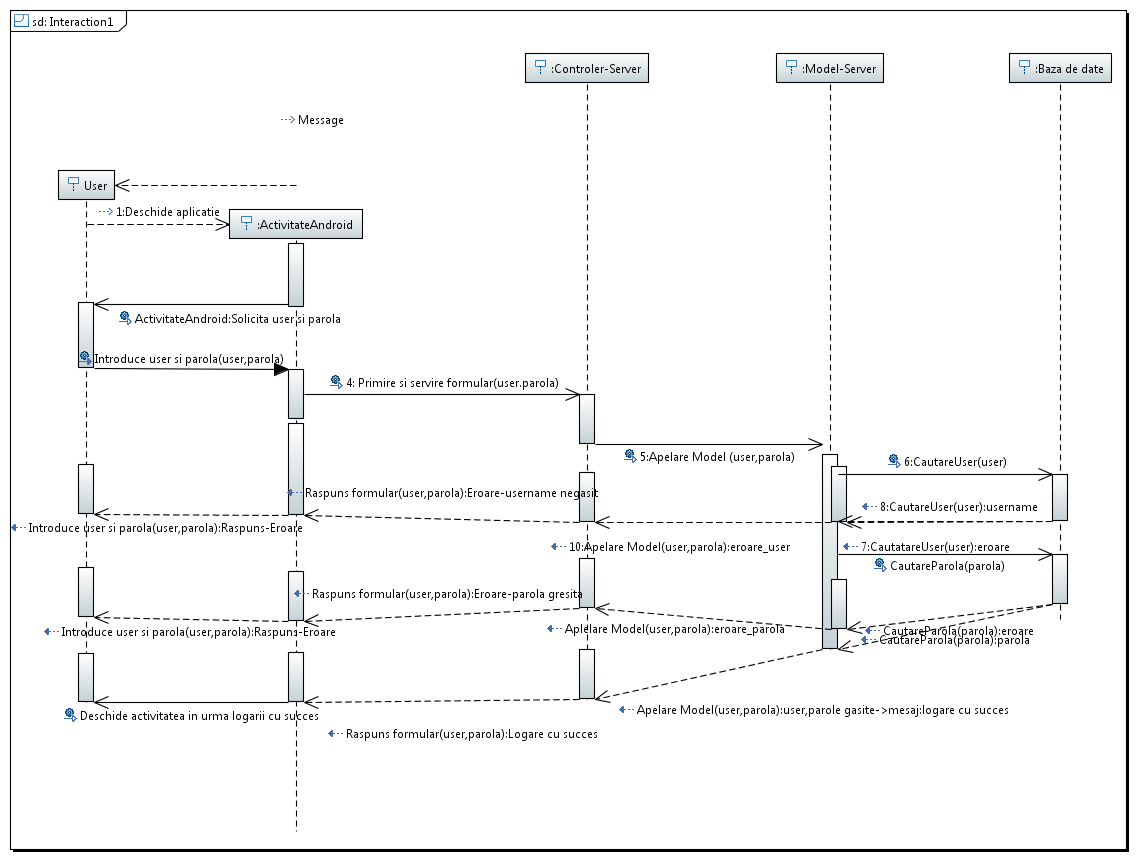
In urmatoarea diagrama se imparte sistemul in cele 3 componente ale sale:Model ,View si Controller.

Sintaxa folosita este astfel in utilizarea legaturilor:

Nume\_functie(parametrii):nume\_obiect\_returnat

Obs:-obiect nu se refera la conceptul de obiect

-nume\_obiect\_returnat se refera doar pentru valori returnate in urma utilizarii functiei



I.2 Trimitere stare de alerta

**Rezumat:** Utilizatorul odata logat folosind username-ul si parola primite, poate primi notificari de la server in legatura cu valori anormale ale monoxidului de carbon. In acest caz se descrie pe scurt cum acesta se realizeaza.

**Actori:** Client, Administrator, sistem embedded.

**Descrierea scenariului:**

**Preconditii:**

- persoana sa fie logata in sistem

- serverul sa ruleze

**Scenariu Nominal:**

1. Se masoara monoxidul de carbon.

2. Sistemul preia datele daca valoarea de la senzor depaseste un anumit prag.

3. Controllerul creeaza un formalar catre aplicatie (http post,get).

4. Se salveaza aceasta valoare in baza de date la referinta utilizatorului la care este inregistrat sistemul embedded.

5. Aplicatia preia informatia de la server.

6. Apoi aplicatia creeaza o noticare pentru utilizator.

**Scenariu Alternativ:**

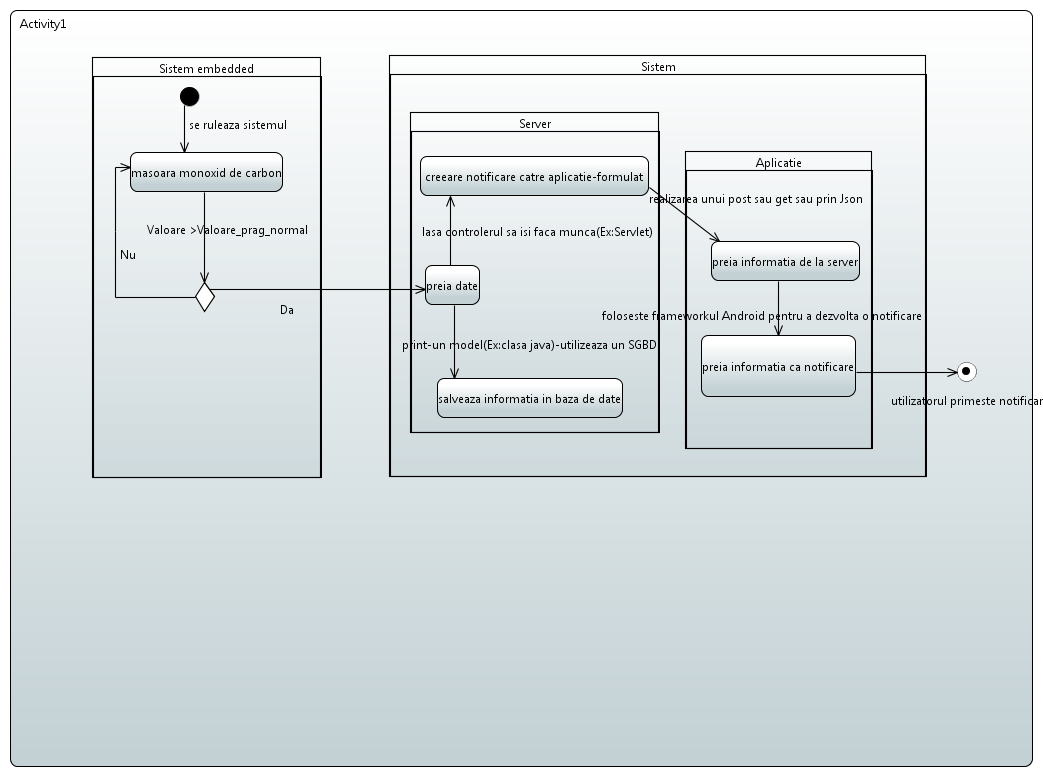
A1: Masurarea monoxidului de carbon.

Alegerea se face la punctul 2 din scenariul nominal. Daca valoarea este sub prag, se va relua masuratoarea.

Cazul de utilizare se continua de la punctul 2 din scenariu nominal.

**Postconditii:**

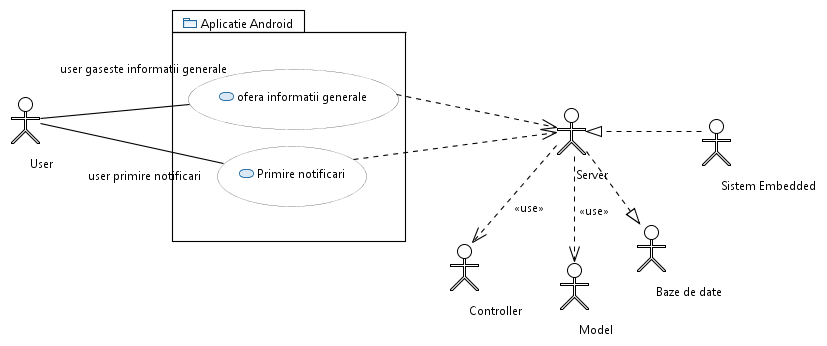
**-** clientul poate sa primeasca alerte.



1. Sistemul este vazut ca aplicatie Android.

Android poate fi văzut ca un sistem MVC de asemenea. Activitatea ce gestionează afisarea ferestrei către user, evenimente inițiate la utilizarea componentelor către user (listeners) si apelarea diferitelor clase. Deci Activitatea este un fel de controller( pseudo view-controller). View este dat de XML, dar poate fi realizat si in java (in diagrame este prevazut View doar prin XML). Modelul ,business logic, este realizat de clase simple java.

Mai jos avem diagrama use-case unde Android este vazut ca sistem.



II.1. Oferă informații generale

Actori: Client, Administrator

Descrierea scenariului: Se descrie în timp etapele sistemului în urmă acțiunii Clientului de a cere informații generale asupra monoxidului de carbon din urmă citirii senzorului.

Descrierea scenariului:

Precondiții:

- aplicația instalată pe telefonul mobil

- serverul să ruleze

Scenariu Nominal:

1. Utilizatorul intră în aplicație

2. Deschide activitatea Logare.

3. Activitatea Logare apelează metodele OnCreateView și seteazaResurseleOnView pentru a seta partea de vizualizare (realizată în XML), realizând corelația între obiectele grafice cu evenimentele asociate acestora .

4. Userul introduce nume user și parolă.

5. Modificând conținutul asupra obiectelor grafice (2 căsuțe de editare text, una pentru user și una pentru parolă), se aplică listenerii (cele care se ocupă cu evenimentele asupra componentelor grafice) corespunzând elementelor grafice utilizate, aplicând noile valori.

6. Apăsa buton logare duce la apelarea modelului ModelCautareBd , unde se vor luă valorile Strîng corespunzând căsuțelor user și parolă, și se vor trimite printr-un formular la sistem (server) . Acesta etapă este realizat de model.

7. Formularul este preluat de către server, unde se realizează căutarea în bază de date cu valorile primite și apoi este trimis un mesaj către aplicație.

8. Dacă parolă și/sau user au fost introduse corect o să se deschidă o activitate ce (printr-un intent) DateDeLaSenzor.

9. Se reia pasul 3 dar asociat activității DateDeLaSenzor.

10. Utilizatorul apăsa butonul de afișare.

11. Se aplică listenerul pentru butonul respectiv.

12 . Se apelează modelul ModelCautareBd folosind user-ul logat. Se realizează formularul asemănător că la punctul 6 dar folosind doar parametru user.

13.Asemănător cu 7 dar se caută datele asociate user-ului în bază de date, provenite de la senzorul aferent. Apoi se trimit datele cu informații cerute la aplicație.

14. Aplicația preia datele, activitatea apelează modelul pentru crearea unui tabel cu informațiile primite.

15. Apoi activitatea direcționează tabelul în View( partea de afișare ).

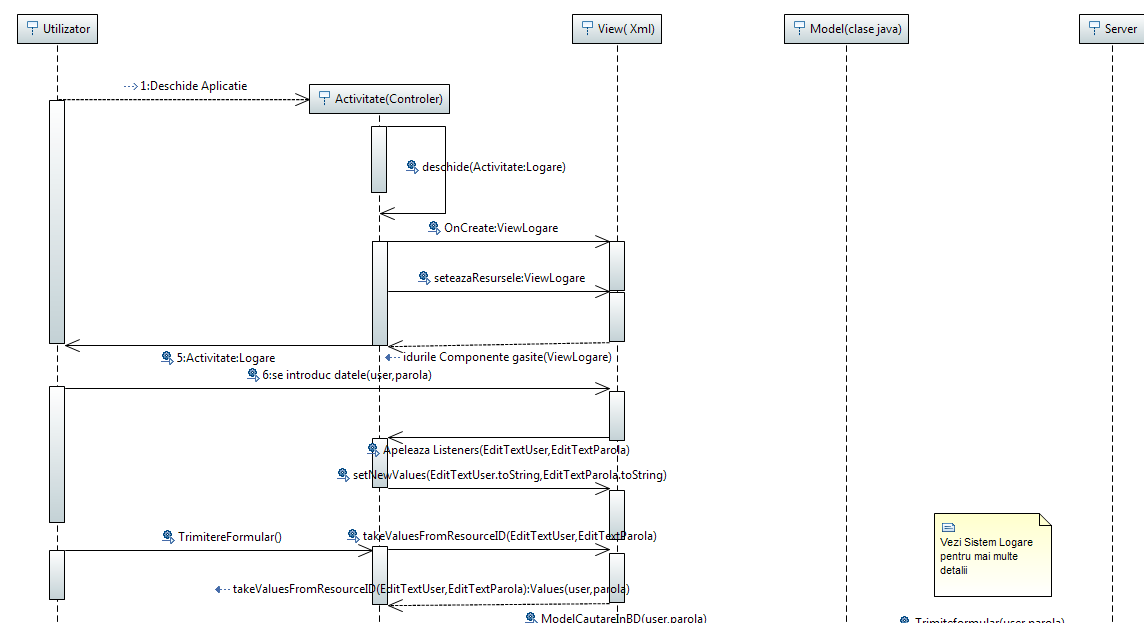
Postconditii:

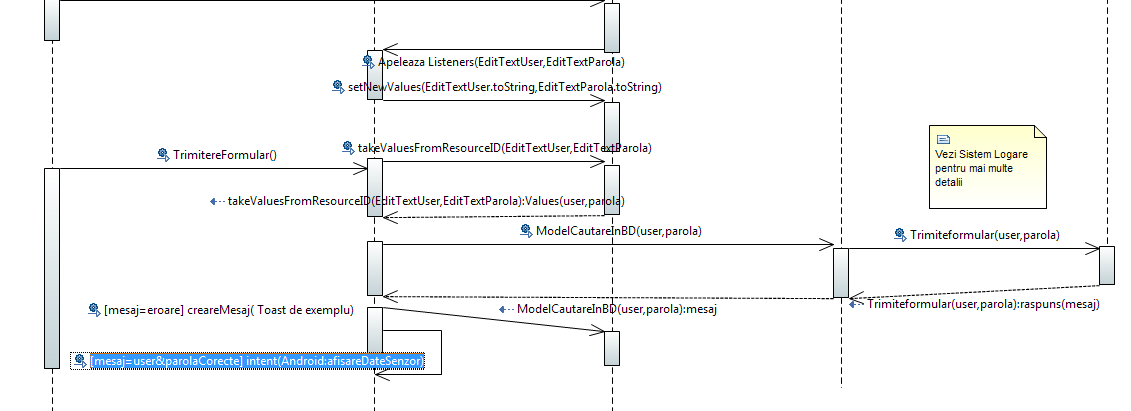
- clientul vede informațiile legate de monoxidul de carbon date de senzor.

Scenariu Alternativ:

A1: Verificarea numelui de utlizator și a parolei

Alegerea se face la punctul 8 din scenariul nominal. Dacă parolă și/sau ușer au fost greșite o să se trimită un mesaj, că user/parolă sunt greșite.





După ce este logat, se creează noua activitate: DateDeLaSenzor

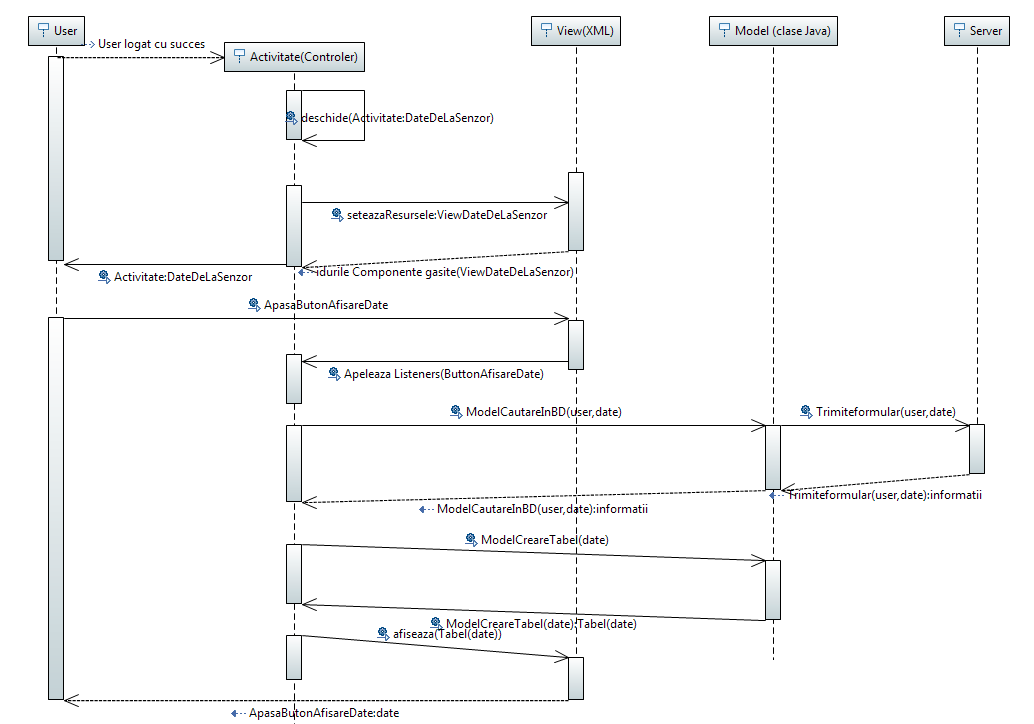
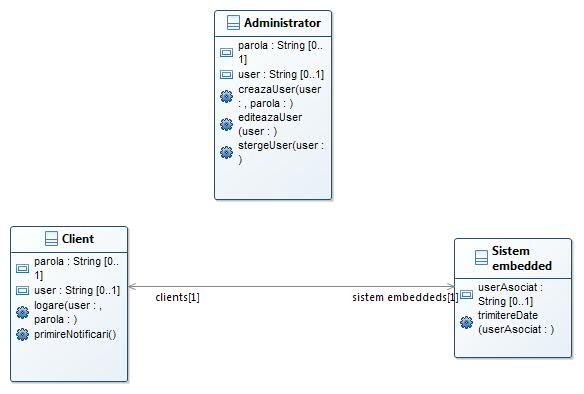


Diagrama statică : 

# Concluzii

În concluzie o aplicație în Android este gândită ca o arhitectură MVC indiferent de ustensilele pe care le utilizezi. Necesită o viziune bună pentru partea de design a aplicației pentru o usoară utilizare de către utilizator. Datorită memoriei mici a telefoanelelor folosim o bază de date externă pentru a stoca o cantitate mare de informații. Pentru transmisia informației între server si android se utilizeaza metode rapide și ușoare ( de exemplu JSON). Deci este necesar o atenție mare pentru proiectarea aplicației dar se pot realiza modificări pe parcus făra ca procesul de creeare a fi întârziat foarte multe sau resetat.

# Bibliografie

1. <https://developer.android.com/guide/index.html>
2. Computer - Addison-Wesley UML - UML Reference Manual