UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Tracker - Aplicație de înregistrare a activitaților în spațiul deschis

propusă de

Alexandru-Vasile Popuțoaia

Sesiunea: februarie, 2019

Coordonator științific

Conf. Dr. Anca Vitcu

UNIVERSITATEA ”ALEXANDRU-IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Tracker - Aplicație de înregistrare a activităților în spațiul deschis

Alexandru-Vasile Popuțoaia

Sesiunea: februarie, 2019

Coordonator științific

Conf. Dr. Anca Vitcu

Avizat,

Îndrumător lucrare de licență,

Conf. Dr. Anca Vitcu.

Data: ..... Semnătura: .....

Declarație privind originalitatea conținutului lucrării de licență

Subsemnatul **Popușoaia Alexandru-Vasile** domiciliat în **România, jud. Iași, mun. Pașcani, Strada Avram Iancu, nr. 50**, născut la data de **04 octombrie 1995**, identificat prin CNP **1951004225891**, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2018, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Tracker - Aplicație de înregistrare a activităților în spațiul deschis** elaborată sub îndrumarea domnului **Conf. Dr. Anca Vitcu**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: ..... Semnătura: .....

Declarație de consimțământ

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Tracker - Aplicație de înregistrare a activitaților în spațiul deschis**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test, etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Alexandru-Vasile Popuțoaia**

Data: ..... Semnătura: .....

Cuprins

Introducere

Locurile de muncă la birou devin din ce în ce mai populare de la an la an, deoarece nu implica muncă fizică și deobicei sunt mai bine platite decât celelalte joburi. Alegerea unui astfel de job sedentar poate avea efecte negative asupra corpului în timp dacă nu se efectuează activitați fizice cel puțin odată pe săptamană. Fiind constienți de efectul negativ pe care-l are sedentarismul asupra corpului, din ce in ce mai multa lume optează în a-și creea un jurnal cu activitățile pe care le efectuează, acest lucru ajutând la o mai bună organizare a activitaților fizice.

Lucrarea de față își propune creearea unei aplicații ce-i permite utilizatorului să-și creeze un jurnal cu activitățile pe care le efectuează în saptiul deschis, ce pot fi vizualizate atât pe mobil cat și într-o aplicație web. O activitate constă în înregistrarea coordonatelor GPS ale unei rute parcurse, ea putând fi vizualizată mai târziu.

Activitatea fizică necesită efort ce nu întotdeauna este plăcut ceea ce poate duce la demotivare. Stocarea activităților poate motiva utilizatorul în funcție de rezultatele pe care le obține, fie rele fie bune, făcând-l conștient de progresul pe care l-a obținut, acesta putând fi vizualizat și în viitor prin vizualizarea datelor din jurnal.

Aplicații populare ce sunt disponibile deja pe piață, ce realizeză maparea unui traseu, având atât aplicații disponibile pentru mobil cât și o platformă web, sunt Strava, Runnkepper sau Runtastic. Motivația din spatele alegeri dezvoltării unei aplicații asemanatoare, se află în placerea pe care o am pentru activitățile sportive cum ar fi alergatul, mersul cu bicicleta sau mersul la munte, dar și dorinței de a afla și de a experimenta ce pași implică dezvoltarea unei aplicații asemanătoare.

În consecință lucrarea de față își propune implementarea unei aplicații android și a unei platforme web, ce permit înregistrarea coordonatelor GPS ale unui traseu și vizualizarea acestora din aplicația android sau platforma web.

Contribuții

Ideea acestui proiect nu este una originală, ci mai mult o provocare în a vedea ce implică dezvoltarea unei aplicații. Chiar dacă idea nu este originală, pașii pe care i-am ales pentru a concepe și implemeta acest proiect reprezintă idei personale. Nu aș fi putut să afișez într-o hartă coordonatele GPS înregistrate fară ajutorul librăriei open-source Open Street Map, ce oferă o documentație bine structurată, același lucru îl pot spune și despre documentația API pentru android. Aplicația conține trei componente aplicația android, servărul Apache și interfața web.

Realizând acest proiect, am reușit să descopăr și să aprofundez o gramadă de cunoștințe noi atât în domeniul Android cât și cunoștințe ce țin de dezvoltarea unei aplicații web responsive.

Capitolul 1

Dezvoltarea și implementarea aplicațiilor

În timp ce dezvoltam aplicația android mi-am pus întrebarea dacă adăugarea unei platforme web este calea cea bună. La început nu am văzut necesar acest lucru, deoarece aplicația android îți oferă deja posibilitatea de vizualizare a tuturor înregistrarilor, dar pe măsura ce se adaugă înregistrări, un lucru simplu cum ar fi căutarea unei rute și vizualizarea ei pe un dispozitiv mobil ce are dimensiunile ecranului destul de mici, poate deveni puțin dificilă. Punând la dispoziție și o pagină web unde îți poți vizualiza activitatea realizată cu aplicația android, consider ca este un lucru destul de bun, deoarece facilitează interacțiunea untilizatorilor cu aplicația android, deasemenea pe baza datelor adunate se pot genera și afișa tot felul de statistici pe pagina web cum ar fi grafice, pie charturi si multe altele. Acest lucru se poate face deasemenea și din interiorul aplicației android, dar consider ca este mai dificil.

În consecință am ales ca pe langă dezvoltarea aplicației android să adaug și o platformă web ce permite utilizatorilor vizualizarea datelor de oriunde s-ar afla și atunci când nu au telefonul mobil în apropiere.

1.1 Aplicația Android

1.1.1 Sistemul de operare Android

Android este un sistem de operare pentru dispozitive și telefoane mobile dezvoltat de Google ce are la bază o versiune modificată a Linux Kernel și alte programe ce sunt open source. A fost proiectat în primul rând pentru dispozitive mobile cu ecran tactil, cum ar fi smartphone-urile și tabletele. De la lansarea din septembrie 2008 și pană acum sistemul de operare a suferit mai multe lansări de versiuni, cea curentă fiind versinea 9: Android "Pie". Sistemul de operare android vine cu un set de aplicații de bază, cum ar fi aplicația SMS, calendarul, browser-ul de navigare pe internet, contactele si multe altele. Utilizatorii pot alege sa înlocuiască aplicațiile de bază, mai puțin cele ce au legatură cu setarile de sistem, cu unele dezvoltate de catre o altă terță parte.

Întregul set de funcționalități al sistemului de operare Android a fost pus la dispoziție prin API-uri (Application Programing Interface) scrise in limbajul de programare Java, cunoscut și sub numele de "Java API Framework". Aceste API-uri formează blocurile de care este nevoie pentru a construi aplicații Android și de a simplifica reutilizarea componentelor și serviciilor modulare de bază ce includ următoarele:

- Sistemul "View" ce este folosit pentru a construi interfața grafică a unei aplicații, ce include liste, grile, cutii de text și butoane.
- "Managerul de Resurse" ce oferă acces la resurse statice, non-code cum ar fi șiruri de caractere, grafice și fișiere de layout ce sunt utilizate de cod.
- "Managerul de Notificari" ce permite tuturor aplicațiilor sa afișeze alerte personalizate în bara de stare.
- "Managerul de Activități" ce gestionează ciclul de viață al aplicațiilor și oferă o stivă obișnuită de navigare înapoi.
- "Furnizori de conținut" ce permite aplicațiilor să acceseze date din alte aplicații, cum ar fi aplicația de conttacte sau pentru a distribui proriile informații.

Dezvoltatorii au acces în totalitate la aceleași framework APIs pe care aplicațiile sistemului android le folosește. Folosind Linux Kernel permite Android-ului sa beneficieze de caractersiticile cheie de securitate, deasemenea permite producătorilor sa dezvolte drivere pentru un kernel bine cunoscut.

1.1.2 Ciclul de viață al unei activități

Instanțele activități unei aplicații tranziționează prin diferite stări pe parcursul ciclului de viața, acest lucru este exemplificat în Figura 1.1. Clasa "Activity" oferă o serie de metode (onCreate(), onStart() ...) ce permit activității să știe în ce stare se află, în execuție, întreruptă temporar, oprită sau inexistentă. Metodele oferite de clasa "Activity" pot fi suprascrise pentru a putea efectua diferite operații cu ajutorul lor, cum ar fi gestiunea memoriei sau verificarea producerii a diferitor evenimente, ce necesită ca oridinea în care se aplează aceste metode sa fie știută.

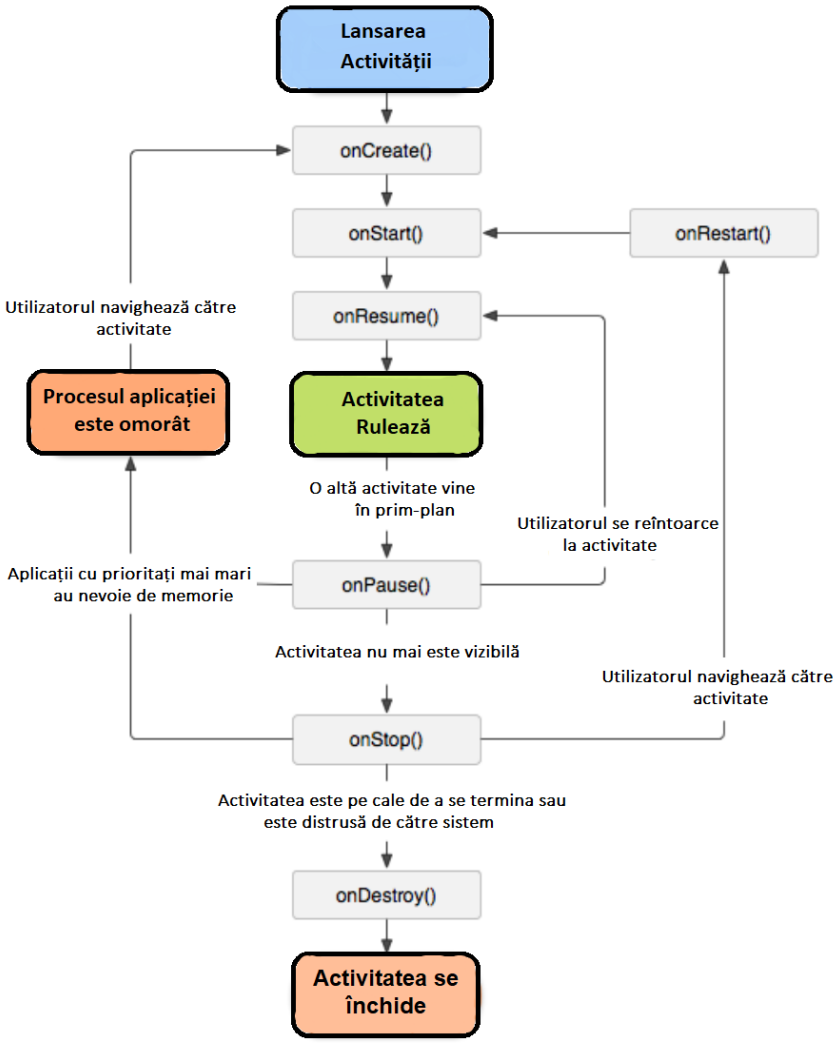

Figura 1.1: Ciclul de viață al unei activități

1.1.3 Android Studio

Android Studio este mediul oficial de dezvoltare (IDE - Integrated Development Enviroment) a aplicațiilor android, bazat pe IntelliJ IDEA. Pe langă editorul de cod, Android Studio oferă funcții care tind sa sporească productivitatea dezvoltării unei aplicaţtii, cum ar fi:

- Un sistem flexibil de construcție bazat pe Gradle
- Un emulator rapid și bogat în caracteristici
- Un mediu ce permite dezvoltarea aplicațiilor pentru toate dispozitivele Android
- "Instant Run" pentru a face update rapid aplicațiilor fară a mai fi necesar construirea unui nou APK (Application Package File)
- Și multe altele

1.1.4 Structura Proiectului

Fiecare proiect in Android Studio conține unul sau mai multe module ce conțin cod sursă și fișiere de resurse. Strunctura proiectului se poate schimba in funcție de nevoi. Atunci când un nou proiect este creat, Android Studio creează structura necesară pentru toate fișierele și le face vizibile in fereastra de proiect ce se află in partea stângă a proiectului. O parte din structura proiectului ce urmează a fi prezentat poate fi vizualizat în Figura 1.2.

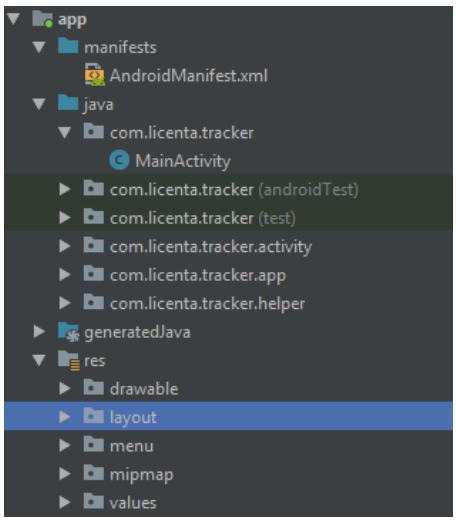

Figura 1.2: Structura Proiectului

1.1.5 Dezvoltarea aplicației android

Aplicația Tracker are la bază un API REST (Representional State Transfer) necesar pentru a comunica cu serverul, acesta fiind un stil arhitectural ce definește un set de constrângeri în ceea ce privește creearea unui serviciu web. Când vine vorba de arhitectura Client-server, clientul și serverul sunt complet separate unul de celălalt. Componenta de front-end este clientul, iar serviciul din spate este serverul. REST este cel mai frecvent utilizat stil pentru proiectarea API-urilor, mai ales in lumea mobilă.

Pentru a interacționa cu baza de date MySQL trebuie sa construim un REST API. Jobul unui REST API este de a lua o cerere de la client, de a interactiona cu baza de date și în final de a trimite un raspuns înapoi la client. Așadar vom crea un simplu PHP, MySQL API întai ce va putea efectua urmatoarele joburi:

- Acceptă cereri în metodele GET/POST
- Interacționează cu baza de date MySQL pentru a insera date și de a colecta date
- Și in final întoarce un raspuns în format JSON (JavaScript Object Notation)

În figura 1.3 de mai jos putem vizualiza structura API-ului folosit pentru aplicația Traker.

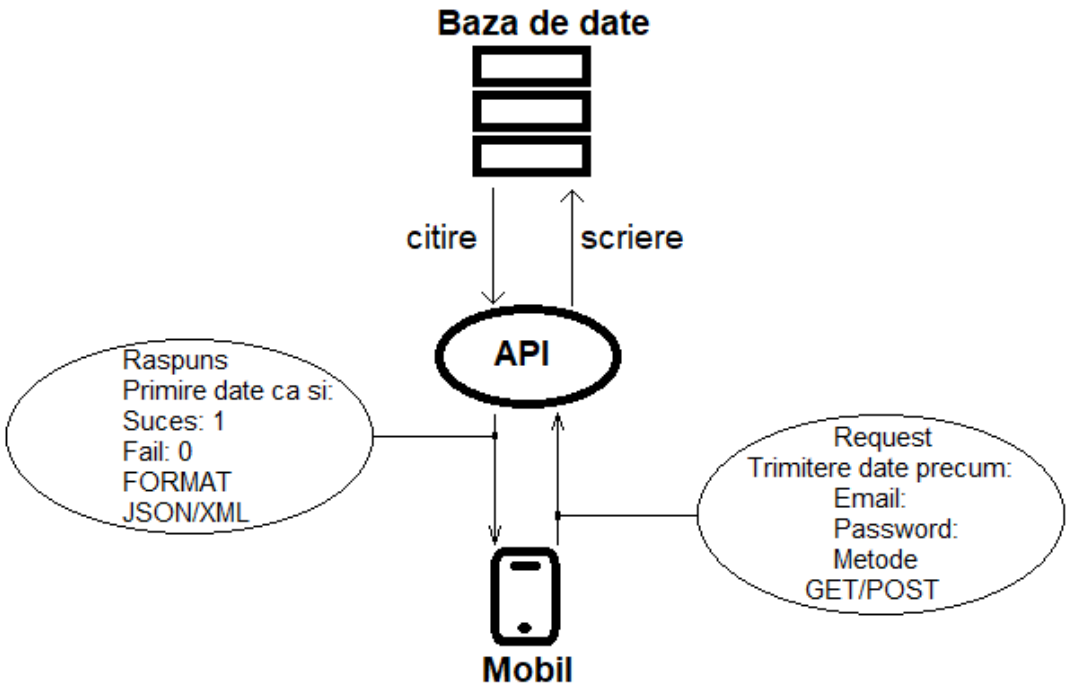

Figura 1.3: API Traker

Motivul din spatele alegeri folosirii MySQL ca bază de date este datorită faptului ca este potrivită pentru pagini web de dimensini mici sau medii. Baza de date o pot folosi gratis și este open-source cu o licență comercială disponibilă. MySQL are un avantaj major, pe langă faptul că este gratis, este deobicei disponibilă în pachetele de hosting și pot fi ușor de configurat într-un mediu Linux, Unix sau Windows.

Baza de date conține patru table necesare stocării datelor culese de către aplicatia android. Primul tabel este folosit pentru stocarea datelor pentru utilizatori și este definit în felul următor:

```
create table users (
    id int(11) PRIMARY KEY AUTO_INCREMENT,
    unique_id varchar(23) not null UNIQUE,
    name varchar(50) not null,
    email varchar(100) not null UNIQUE,
    encrypted_password varchar(80) not null,
    salt varchar(10) not null,
    created_at datetime,
    updated_at datetime null
);
```

Fiecare utilizator are un identificator unic, iar parola nu este stocată in plain text ci a fost criptată folosind un șir de caractere generat aleatoriu, fiind folosit ca data de intrare într-o funcție hash ”one-way”. O funcție ”one-way” este o funcție ușor de calculat la fiecare intrare, dar greu de inversat folosind o intrare aleatorie. Doua parole identice nu vor avea același rezultat hash, deoarece au primit ca input în funcția hash acel șir de caractere generat aleatoriu, numit ”salt” ce este stocat în baza de date deoarece este necesar verificări autentificării utilizatorilor în aplicație.

Pentru partea de back-end am decis să folosesc PHP (Hypertext Preprocessor). Acesta este un limbaj de scriptare comun cu utilizare open-source, ce este potrivit pentru dezvoltarea de pagini web, putând fi incorporat în HTML. Ceea ce distinge PHP de ceva asemănator cu JavaScriptul pe partea clientului este faptul că codul este executat pe server, generând HTML apoi fiind trimis înpoi clientului. Clientul primește rezultatele acestui script, dar nu va ști care a fost codul de bază. serverul se poate configura astfel încat toate fișierele HTML sa fie procesate cu PHP, atunci într-adevăr nu există nici o modalitate prin care utilizatorii să-și dea seama.

Ca și mediu de dezvoltare PHP local, am decis sa folosesc XAMPP, ce este o metodă ușoară pentru a instala o distribuție Apache ce conține o bază de date MariaDB, PHP și PEARL. XAMPP este un pachet open-source ce promovează web serverul Apache și are ca scop o instalare ușoară a distribuției pentru Windows/MAC de către dezvoltatori. Odată instalat, se poate deschide programul și testa prin deschiderea adresei "http://localhost/" într-un browser. Deasemenea se poate verifica si baza de date prin deschiderea adresei "http://localhost/phpmyadmin" acesta fiind un tool ce are ca scop administrarea acesteia.

Proiectul PHP se creează în locația fișierului "htdocs" unde XAMPP a fost instalat, în aceasta locație am adaugat un folder "Tracker" ce va fi rădacina proiectului PHP. Aici vom include toate scripturile necesare conectări la baza de date cât și administrarea operațiilor asupra acesteia, asfel vom creea niste clase ajutatoare și anume "Config.php" ce are ca rol definirea unor constante pentru configurarea bazei de date si clasa "DB_connect.php" ce creează o instanță ce se va ocupa de deschiderea și închiderea bazei de date. O altă clasă importantă în acest priect este "DB_Functions.php" ce conține funcțiile necesare pentru efectuarea operațiilor asupra bazei de date cum ar fi:

- Stocarea unui utilizator
- Ștergerea unui utilizator
- Preluarea unui utilizator
- Actualizarea unui utilizator
- Și multe alte funcții ce vor fi prezentate pe parcurs

Criptarea parolei utilizatorilor necesită două coloane în tabel, una pentru parola criptată și una pentru "salt" ce a fost folosită în metoda "base64encode" ce este necesară pentru a codifica datele cu un format MIME (Multipurpose Internet Mail Extensions) în base64 ce ocupă un spațiu cu 33% mai mare decât datele originale.

Un utilizator se poate înregistra cu nume, email, parola ca și parametri POST pentru stocarea acestuia în baza de date. POST și GET sunt cele mai comune metode folosite în HTTP (Hypertext Transfer Protocol) ce a fost proiectat pentru a permite comunicarea dintre clienți și servere.

Metoda GET este folosită pentru a solicita date dintr-o resursă specifică. Metoda POST este folosită pentru a trimite date către server pentru a crea sau actualiza o re-

sursă. Solicitările acestei metode nu sunt niciodată memorate în cache, nu rămân în istoricul browserului, nu pot fi marcate și nu au restricții privind lungimea datelor.

Aproape identic ca și metoda de înregistrare, email-ul si parola utilizatorului sunt trimise ca și parametri POST către server. După primirea acestor parametri se face o verificare în baza de date pentru a verifica dacă untilizatorul există. Daca utilizatorul a fost găsit, iar parola introdusă corespunde cu cea criptată atunci serverul întoarce un răspuns de succes în format JSON.

JSON (JavaScript Object Notation) reprezintă o sintaxă pentru stocarea și schimbul de date, se "auto-descrie" și este ușor de înțeles, fiind independent de limbaj. Deoarece acest format este numai text, acesta poate fi trimis cu ușurință către și de la un server și utilizat ca format de date de către orice limbaj de programare. În consecință raspunsul primit de la server, în format JSON este o foarte ușor de descifrat și parsat.

Un exemplu de răspuns în format JSON întors de către server în caz de autentificare cu succes, ce este mai apoi procesat de către aplicația android, are următoarea formă:

```
{
  "error": false,
  "uid": "5c55b727035903.71390557",
  "user": {
    "name": "test",
    "email": "test@test.com",
    "created_at": "2019-02-02 17:28:39",
    "updated_at": null
  }
}
```

Android permite aplicației sa se conecteze la internet sau la orice altă rețea locală. Pentru a-mi ușura lucrul cu operațiuni asura rețelei am ales sa folosesc librăria Volley oferită de catre Android.

Volley este o librărie ce face ca rețelistica pentru aplicațiile android să fie mai ușoara și cel mai important, mai rapidă. Această librărie a fost folosită inițial de echipa Play Store din aplicația Play Store și a fost lansată ca și librărie open-source în 2013. Volley poate face cam orice ce are legatură cu rețelele în Android, aceasta programează automat toate solicitările la rețea, cum ar fi preluarea răspunsurilor pentru o imagine

de pe web. Oferă cache transparent pe disc și memorie, oferă un puternic API pentru anularea unei singure cereri sau a unui bloc de cereri ce poate fi configurat, deasemenea oferă și instrumente pentru debug. Se integrază cu ușurință cu orice protocol și vine cu suport pentru șiruri de caractere, imagini și JSON. Oferă suport integrat pentru caracteristicile de care este nevoie și eliberează programatorul de a scrie o gramadă de cod ce au un inpact foarte mic asupra aplicației. Librăria Volley nu este potrivită pentru operațiuni mari de descărcare sau streaming, deoarece ține toate raspunsurile în memorie în timpul parsării. Deoarece aplicația Tracker nu va efectua operațiuni mari, face ca această librarie sa fie alegerea potrivită pentru lucrul cu rețeaua.

Cea mai ușoară metodă de a adăuga librăria la proiect este prin adăugarea unei dependențe în build.gradle-ul aplicației ce este prezentată mai jos.

```
dependencies {
    compile 'com.android.volley:volley:1.1.1'
}
```

Pentru a trimite o simplu request, trebuie creată o coadă ce va conține obiecte de tip request. Această coadă va gestiona firele de execuție lucrătoare ce rulează operațiile de rețea, anume citirea și scrierea în cache și parsarea răspunsurilor. Requesturile parsează raspunsurile, iar Volley are grijă să trimită răspunsul parsat înapoi către firul principal de execuție pentru livrare.

Volley oferă întotdeauna răspunsuri parsate pentru firul principal de execuție. Rularea pe firul principal de execuție este convenabila pentru popularea comenzilor UI (User Interface) cu datele recepționate, deoarece aceste comenzi se pot modifica direct din managerul de răspuns, fiind extrem de important semantica furnizată de bibliotecă, în special cea de anulare a requesturilor. Dupa ce requestul a fost creat, se adaugă la coadă folosind metoda ”add()”. Odată ce requestul a fost adăugat, acesta trece prin pipeline, este servit, apoi este analizat și livrat.

Când metoda ”add()” este apelată, Volley rulează un fir de execuție pentru procesarea cache-urilor și un grup de fire de execuție, pentru a fi expediate. Când un request este adăugat în coadă, el este preluat de firul de execuție al cache-ului și îi este asignat o prioritate: în cazul în care requestul poate fi servit din cache, atunci răspunsul va fi parsat în firul de execuție al cache-ului apoi va fi livrat către firul principal de execuție. Dacă requestul nu poate fi servit din memoria cache, atunci va fi plasat în coada de rețea. Primul fir de execuție disponibil al rețelei va lua un request din coadă, va efec-

tua o tranzacție HTTP și mai apoi va scrie răspunsul în cache, mai apoi va trimite răspunsul parsat înapoi către firul principal de execuție.

Mai jos voi descrie cum se poate trimite un request folosind metoda ”Volley.new RequestQueue”, ce stabilește o coadă și o pornește folosind valori prestabilite.

```
// Creeaza o referinta catre text-view-ul din layout
final TextView myText = (TextView) findViewById(R.id.myText);

// Instantiaza coada de requesturi
RequestQueue queue = Volley.newRequestQueue(this);
String url =".../Tracker/mobile/login.php";

// Cere un raspuns de la adresa introdusa
StringRequest stringRequest = new StringRequest(Request.Method.GET, url
    new Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
    //Afiseaza in TextView primele 500 de caractere ale raspunsului
        myText.setText("Raspunsul este: "+ response.substring(0,500));
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
    //Trateaza erorile
        myText.setText("Ceva nu a mers bine!");
    }
});
// Adauga requestul la coada.
queue.add(stringRequest);
```

Când un utilizator deschide aplicația Tracker pentru prima dată, el este redirecționat către pagina de autentificare, de unde poate accesa și pagina pentru înregistrare. În acel layout își poate crea un cont de utilizator, cu care se poate autentifica mai apoi. Layout-ul de înregistrare preia inputul de la utilizator cu ajutorul unor casete de text derivate din clasa publica "EditText" ce reprezintă un element de interfață cu utilizatorul pentru introducerea și modificarea textului. Caseta de text permite alegerea tipului de input cum ar fi aspectul textului de editare și configurează tipul de tastatură afișat. Să luăm spre exemplu caseta de text necesară pentru introducerea parolei. Pentru a nu afișa parola în caseta de text este selectat antributul "android:inputType = textPassword". Un atribut de tipul "textPassord" are ca rezultat mascarea textului introdus pentru confidențialitate. Când toate datele necesare au fost completate ṭn casetele de text, utilizatorul va apăsa butonul "REGISTER" ce declanșează un eveniment cu ajutorul metodei "setOnClickListener()" care la rândul ei apelează metoda "onClick()" pentru a salva datele introduse în baza de date, acest lucru este exemplificat și in codul de mai jos.

```
//Eveniment Inregistrare
btnRegister.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String name = inputFullName.getText().toString().trim();
        String email = inputEmail.getText().toString().trim();
        String password = inputPassword.getText().toString().trim();

        if(!name.isEmpty() && !email.isEmpty() && !password.isEmpty()){
            registerUser(name, email, password);
        }
        else{
            Toast.makeText(this, "Please enter credentials!",
            Toast.LENGTH_LONG).show();
        }
    }
});
```

Metoda "registerUser()" creează un POST request Vooley ce este trimis catre ser-

ver, apoi procesat de scriptul PHP ”register.php” ce verifică ca parametrii precum numele, emailul și parola au fost completate. Aceste date sunt procesate mai apoi pentru a verifica dacă untilizatorul nu există deja în baza de date. Daca totul este în regulă se crează o interogare asupra bazei de date ce inserează datele în tabel. Scriptul în final întoarce ca răspuns în format JSON ce este recepționat de requestul volley în metoda ”onResponse()”. Aici se poate parsa răspunsul în caz de succes, în caz de eșec metoda ”onErrorResponse()” va fi apelata, în caz ca respunsul nu este in format JSON sau în cazul în care clientul nu a putut comunica cu serverul, eroarea 404.

Pași asemanători se execută și pentru modulul de autentificare al untilizatorului. Odată ce utilizatorul sa autentificat o nouă sesiune este creată, ce ajută la stocarea datelor utilizatorului astfel încat următoarea dată când deschide aplicația să nu mai fie necesar să se autentifice. Acest lucru este realizat cu ajutorul clasei ”SharedPreferences” din Android ce permit aplicației de a salva și prelua date sub formă de cheie, valoare. Pentru a folosi ”SharedPreferences” trebuie apelată metoda ”getSharedPreferences()” ce returnează o instanță ce pointeză către fișierul ce conține valorile preferințelor. Primul parametru acceptat de metodă reprezintă cheia, iar al doilea reprezintă modul. Pe langă modul privat mai sunt și alte moduri disponibile, cum ar fi:

- MODE_APPEND acest mod va modifica preferințele deja existente
- MODE_PRIVATE setând acest mod, fișierul poate fi accesat doar atunci când este apelat de către aplicație
- MODE_WORLD_READABLE acest mod permite altor aplicații să citească preferințele
- MODE_WORLD_WRITEABLE acest mod permite altor aplicații sa modifice preferințele

Pentru a manipula datele din înăuntrul ”SharedPreferences” este nevoie de un editor. Metoda ”putString(”key”, ”value”);” este folosită pentru a adăuga preferințele, odată adaugate este necesar un ”commit()” ce asigură salvarea acestor preferințe. Odată salvate aceste preferințe, utilizatorul este redirecționat catre pagina principală a activității.

Layoutul activități principale ale aplicației oferă trei lucruri și anume: un textview ce afișează numarul total de activități înregistrate de către utilizator, un buton ce declanșează un eveniment ce pornește activitatea de mapare al traseului și un o listă ce conține jurnalul cu toate traseele înregistrate. Dacă un element din listă este selectat, acest eveniment va deschide o nouă activitate ce conține datele traseului selectat.

Un layout definește structura interfeței de utilizator ale unei activități în aplicația noastră. Toate elementele din layout sunt construite folosind o ierarhie a obiectelor ”View” și ”ViewGroup”. Un ”View” deobicei desenează ceva ce utilizatorul poate vedea și interacționa cu el, pe când un ”ViewGroup” este un container invizibil ce definește structura layoutului pentru un ”View” și alte obiecte de tip ”ViewGroup” așa cum este arătat in Figura 1.4 de mai jos.

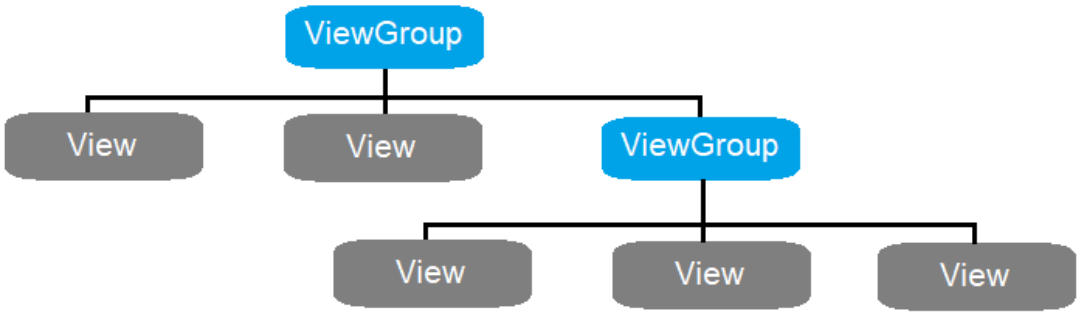

Figura 1.4: Layout View structure

Obiectele ”View” sunt deobicei numite widgets și pot fi una dintre multe subclase precum un Button sau TexView. Obiectele ”ViewGroup” sunt deobicei numite layouts și pot fi de mai multe tipuri ce oferă diferite structuri precum LinearLayout, RelativeLayout sau Constraintlayout.

În Android Studio un layout se poate defini in doua modalități:

- Declararea elementelor de interfață grafică în fișierul XML. Android oferă un vocabular destul de direct în ceea ce privește clasele și subclasele View, cum ar fi widgeturile și layouturile. Deasemenea se poate folosi si editorul pentru layout din Android Studio ce folosește o interfață drag-and-drop.

- Instanțierea elementelor la runtime. Aplicația poate crea obiecte de tip View și ViewGroup în mod programatic.

Declararea elementelor de interfață grafică în XML, ne permite să separăm prezentarea aplicației de codul ce controlează comportamentul ei. Deasemenea folosirea fișierelor XML face ușoară oferirea de fiferite layouturi pentru diferite mărimi de ecran și orientări.

Frameworkul Android oferă flexibilitate de a utiliza oricare dintre aceste două metode pentru a construi interfața grafică. Spre exemplu se poate declara un layout XML provizoriu ce poate fi apoi modificat la runtime.

La compilarea aplicației, layoutul XML este compilat într-o resursă View ce trebuie încarcată în codul aplicației în metoda Activity.onCreate(). Pentru a încarca layoutul se aplează metoda setContentView() ce primește ca referință resursa layoutului aflată în fișierul R. Un exemplu din aplicație ce face acest lucru este prezentat mai jos:

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
```

Lista din activitatea principală ce conține toate înregistrarile traseelor a fost realizată din trei widgeturi TextView ce conțin numele traseului, distanța totală parcursă și timpul. TextView-ul ce conține numele este aliniat si centrat la stânga, apoi celelalte doua widgeturi au fost aliniate la dreapta widgetului ce conține numele, fiind aliniate vertical, acest lucru se poate observa și in Figura 1.5.

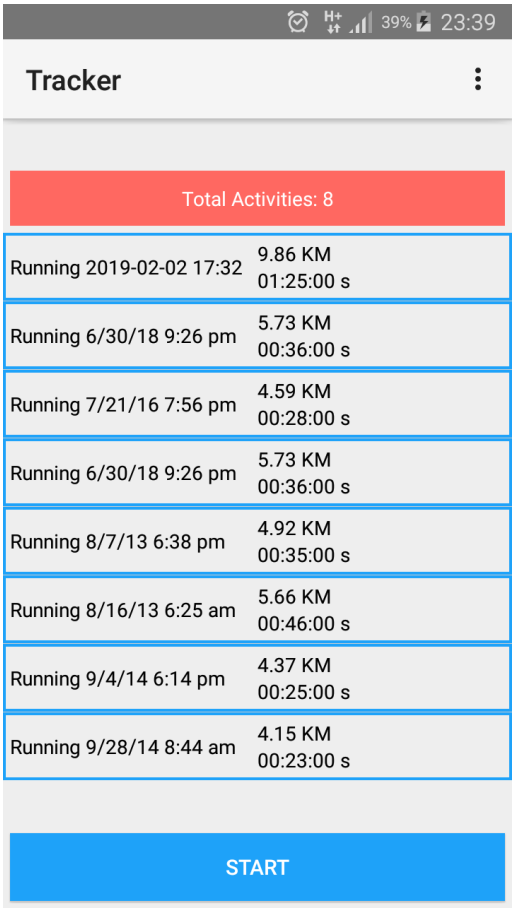

Figura 1.5: Layout Activitatea Principală

Pentru a afișa elementele din listă este necesar construirea unui obiect Adapter ce acționează ca o punte între un AdapterView și datele pentru acel view. Adapter-ul este responsabil pentru vizualizarea fiecărui element din setul de date, în cazul de față lista ce conține traseele. Construirea acestui Adapter necesită extinderea clasei "BaseAdapter" și suprascrierea metodelor getCount(), getItem(), getItemId() și cea mai importantă getView().

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    if(layoutInflater ==null){
        layoutInflater = (LayoutInflater) activity.getSystemService
        (Context.LAYOUT_INFLATER_SERVICE);
    }

    if(convertView == null){
        convertView = layoutInflater.inflate
        (R.layout.listview_row, null);
    }

    TextView txtActivityName= (TextView) convertView.findViewById
                                        (R.id.txtActivityName);
    TextView txtTotalDistance = (TextView) convertView.findViewById
                                        (R.id.txtTotalDistance);
    TextView txtTotalTime = (TextView) convertView.findViewById
                                        (R.id.txtTotalTime);
    RouteDetails r = activityDetailsList.get(position);

    txtActivityName.setText(r.getActivityName());
    txtTotalDistance.setText(r.getTotalDistance() + " KM");
    txtTotalTime.setText(r.getTotaltime() + " s");

    return convertView;
}
```

Metoda getView() este resposabilă pentru creearea widgeturilor, în cazul de față textview-urile din interiorul unui element din listview. Așadar în spate listview-ul aplează aceasta metodă pentru un element specific din listă. Metoda aceasta poate încarca un view dintr-o resursă sau îl poate crea programatic. Odată ce view-ul a fost încarcat poate fi umplut cu date. În cazul de față textview-urile sunt setate cu datele de la poziția "position" din obiectul RouteDetails.

Înregistrarea unui traseu se poate face prin apăsarea butonului START. Evenimentul declanșat de acest buton lansează un "Intent" catre clasa "OSM_MapActivity" ce lansează o nouă activitate și pauzează activitatea curentă, ceea ce duce la oprirea firului de execuție ce se ocupa de activitate.

Un intent reprezintă o descriere abstractă a unei operații ce urmează a fi executată. Poate fi folosit pentru a lansa o activitate, pentru a trimite informații către o componentă de tip BroadcastReceiver sau pentru a comunica cu un serviciu din background. Cea mai semnificativă utilizare este lansarea activităților, unde poate fi considerat drept adeziv între activități. Este o structură de date pasivă ce deține o descriere abstractă a unei acțiuni ce trebuie efectuată.

Activitatea nou lansată ce are rolul de a stoca datele despre traseul ce urmează a fi parcurs are un layout ce conține un mapview, mai multe textview-uri necesare pentru afișarea datelor ce se înregistrează live, cum ar fi coordonatele gps curente, viteza, altitudinea și distanța curentă parcursă. Pe langă acestea se mai pot adăuga și butoanele pentru captura unei poze și salvarea rutei. Layout ce poate fi vizualizat și in Figura 1.6.

MapView-ul ce este afișat este realizat cu ajutorul librariei osmdroid realizată de Open Street Map. Am decis să folosesc această librărie în locul Google Maps, deoarece este open-source. Osmdroid MapView este practic un înlocuitor pentru MapView-ul oferit de Google. Proiectul Open Street Map este un set de date gratuit ce permite programatorilor să-și îndeplinească nevoile fară a fi limitați fie de API pus la dispoziție de Google sau de termenii și condiițile acestora. OSM permite utilizatorilor să descarce mape pentru uzul offline, ceea ce este un mare avantaj peste Google, mapele fiind bazate pe internet. Pentru a implementa o simplă mapă este necesar adăugarea dependențelor pentru osmdroid-android, osmdroid-wms, osmdroid-mapsforge și osmdroid-geopackage.

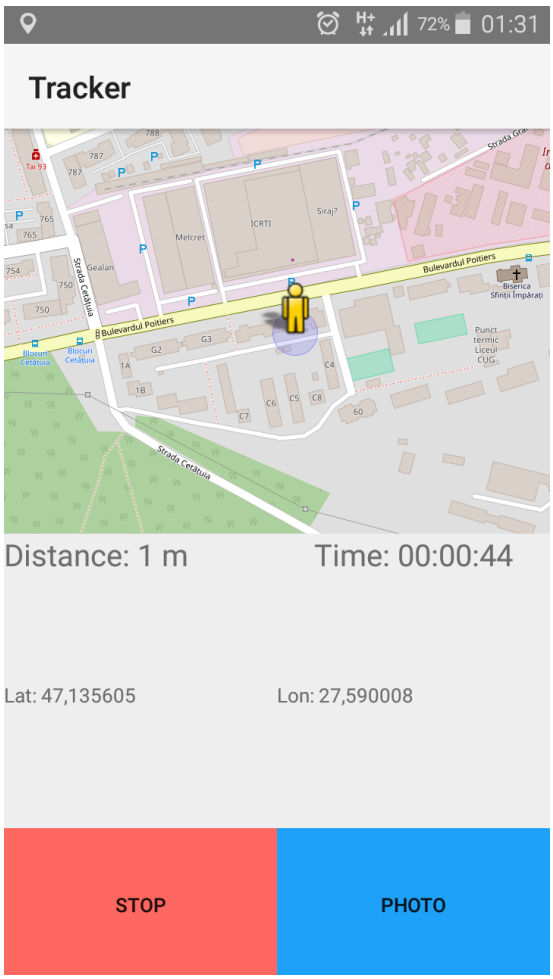

Figura 1.6: Layout Tracking

Pentru a face acestă activitate mai ușor de înțeles am împarțit codul în mai multe module independente. Cel mai important modul fiind clasa ”LocationServiceHandler” ce extinde clasa ”Service”. Un serviciu este o componentă de aplicație ce poate executa operații de lungă durată în background ce nu oferă o interfață catre utilizator. O altă componentă a aplicației este sa pornească un serviciu și să-l ruleze în background chiar daca utilizatorul comută de la o aplicație la alta. Un serviciu este legat de componenta unei aplicații prin metoda bindService(). Un asemenea serviciu oferă o interfață client-server ce permite componentelor să interacționeze cu serviciul, să trimită cereri, să primească rezultate și chiar să facă acest lucru în cadrul proceselor cu comunicare interprocesuală (IPC). Un serviciu legat rulează numai atâta timp cât o altă componentă a aplicației este legată de acesta. Componente multiple se pot lega simultan de serviciu, dar cand toate acestea se desprind, serviviul este distrus.

Serviciul implementat în aplicația Tracker se ocupă returnarea coordonatelor curente GPS și de contorul folosit pentru calcularea distanței totale. Odată ce serviciul a

fost pornit clasa "OsmMapTracking" își poate lua coordonatele gps și contorul ce sunt modificate la fiecare secundă. Toate aceste lucruri se realizează în metoda onStart().

```
@Override
protected void onStart() {
    super.onStart();
    if(locationManager == null){
        locationManager = (LocationManager)
        getSystemService(Context.LOCATION_SERVICE);
    }
    Intent intent = new Intent(this, LocationServiceHandler.class);
    bindService(intent,mServiceConnection, Context.BIND_AUTO_CREATE);
}
```

Odată obținute, datele se salvează într-un obiect JSON, până în momentul în care butonul STOP declansează evenimentul ce trimite obiectul ca șir de caractere către activitatea ce salvează acest JSON, printr-un intent. Opțiunea de a captura poze în timpul traseului a fost realizată cu ajutorul aplicației photo instalată deja în telefon. Partea mai complicată a fost de a găsi o modalitate de a stoca aceste poze. Atunci când se realizează o fotografie metoda OnActivityResult() este apelată ce întoarce poza capturată sub forma unui bitmap. Am decis să stochez imaginea sub forma unui șir de caractere codificat în baza 64. Pentru a realiza acest lucru a fost întâi necesară compresarea imaginii în format PNG ce nu-și pierde din calitate atunci când este compresat, apoi codificare în baza 64. Codificarea imaginii este stocată în obiectul JSON sub forma unui vector de imagini ce conține și coordonatele gps ale locației unde acele poze au fost făcute.

Declanșarea evenimentului de oprire a înregistrări traseului se face prin apăsarea butonului STOP. În acest eveniment se trimite un intent cu un obiect JSON ce conține toate datele, către activitatea ce are rolul de a stoca toate aceste date. Activitatea "SaveTrackActivity" are un layout asemanător cu cel precedent, cu unele mici modificări. Aici în widgetul mapview se poate vedea traseul parcurs marcat cu o linie, de la începutul traseului, legând oricare două coordonate geografice între ele, până la sfâșit.

Sub widgetul de mapview, nu mai avem textview-uri cum aveam în layout-ul precedent, în loc avem o listă ce conține toate datele finale ale traseului parcurs, cum ar fi, distanta și timpul total, numele traseului ce este format automat din numele zilei

și date-i curente și o medie pe kilometru a distanței parcurse. Sub acestă listă se află un gridview ce afișează cate trei imagini pe un rând. Imaginile sunt cele ce au fost capturate în timpul traseului. Spre sfârșitul layout-ului mai avem două butoane ce ne permit salvarea sau anularea salvării traseului. Acest layout poate fi observat și în Figura 1.7.

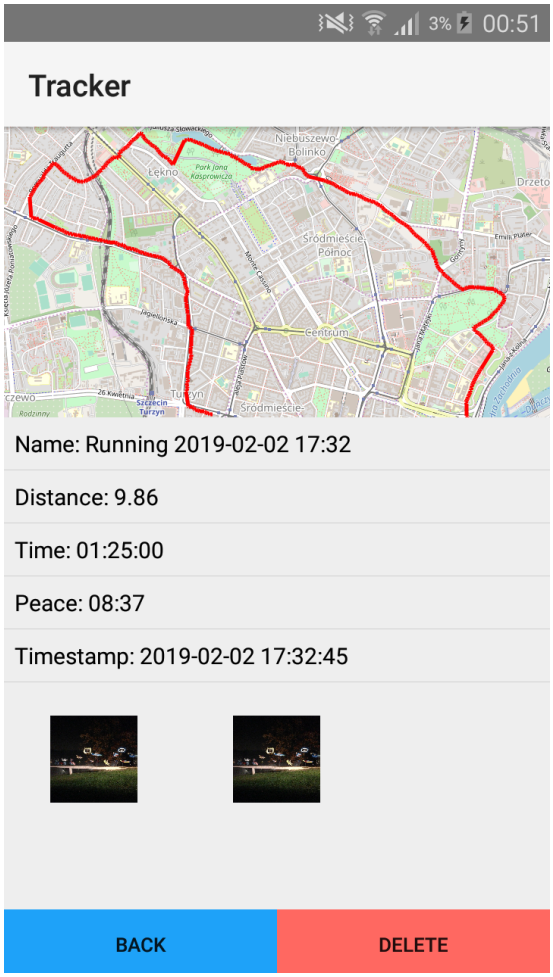

Figura 1.7: Salvarea Traseului

Salvarea traseului implică trimiterea unui request la server cu JSON-ul creat ce conține toate datele despre traseu, pentru a fi parsat și salvat în baza de date. Pentru salvarea datelor în baza de date, a fost necesar creearea mai multor tabele, si anume:

- Un tabel pentru salvarea detaliilor unui traseu, ce conține coloane precum numele traseului, timpul total, distanța totală
- Un traseu pentru salvarea tuturor coordonatelor GPS, fiecare traseu având un id unic generat ce ajută la diferențierea coordonatelor GPS.
- Un ultim tabel necesar stocării tuturor imaginilor capturate de-a lungul traseelor,

identic ca și pentru coordonate, imaginile dețin id-ul unic generat al traseului, ce ajută la identificarea pozelor pentru un traseu anume.

Aplicația Tracker oferă posibilitatea afișării fiecarui traseu în parte, acest lucru se poate face din activitatea principală selectând un element din listă ce ne afișează traseul dorit. Deoarece datele sunt salvate pe server în baza de date, vizualizarea traseelor se poate face și din aplicația web. Activitatea ce afișează un anumit traseu selectat permite utilizatorului să poată șterge toate datele acelui traseu.

Pentru a desena și afișa ruta unui traseu atunci când se deschide activitatea de detalii sau când se deschide activitatea ce salvează traseul, este necesară extragere coordonatelor GPS dintr-un obiect JSON. Aceste coordonate sunt introduse mai departe cu ajutorul API-ului OpenStreetMap într-un obiect din clasa "Polyline", ce acceptă in metoda setPoints() un vector de coordonate gps. Traseul este afișat in widget-ul mapview prin apelarea metodei add() ce primește ca argument obiectul polyline ca mai apoi sa deseneze pe mapview traseul.

Aplicația Android prezentată este un prototip ce poate suferi îmbunătățiri din puncte de vedere al design-ului sau al metodelor de dezvoltare alese. În general aplicația reprezintă un jurnal al utilizatorului ce-o folosește pentru a-și înregistra traseele de mers la munte, alergat sau mers cu bicicleta, ce pot fi analizate îndeaproape.

1.2 Aplicația Web

Procesul de dezvoltare al aplicației web urmărește aproape aceeași structură pe care o are aplicația Android și anume:

- Posibilitatea autentificării și înregistrării în aplicație
- Creearea unei pagini home ca va afișa toate traseele împreună cu niște statistici legate de numarul traseelor.
- Creearea unei pagini ce permite afișarea unui traseu selectat din lista ce este afișată în pagina de home.
- Oferirea posibilității de a încarca un fișier GPX
- Posibilitatea de a modifica setările utilizatorului
- Și în final o pagină de contact, unde utilizatorul poate contacta dezvoltatorii

Cu un server web precum Apache și o bază de date MySQL, avem tot ce ne trebuie pentru a construi un site web modern. Cheia ce îmbină toate aceste lucruri reprezintă metoda prin care serverul comunică cu baza de date, în alte cuvinte o avem nevoie de o metodă de a incorpora operațiile pe care le efectuăm asupra bazei de date în paginile web, iar limbajul care ne ajută să efecutuăm toate aceste sarcini este PHP.

PHP este un proiect open-source dezvoltat de către Apache Software Fundation și este cel mai populară componentă pentru serverele Apache cu o utilizare a serverelor HTTP Apache de aproximativ 53%. PHP este foarte potrivit pentru aplicațiile web ce dețin o baza de date, datorită instrumentelor sale de integrare pentru mediile web și a bazelor de date. Flexibilitatea incorporării scripturilor în paginile HTML permite integrarea ușoară a codului. Suportul de integrare a bazelor de date este, de asemenea, excelent, cu mai mult de 15 bibilioteci disponibile pentru a interacționa cu aproape toate serverele populare de baze de date.

Aplicația Tracker are o arhitectură simplă formată dintr-un client ce este reprezentat de browser, un server și o bază de date. Pentru ca aplicația web sa fie responsive am ales sa folosesc cea mai populară librarie, și anume Bootstrap versinea 4.0, deoarece oferă o documentație foarte bine structurată cu foarte multe exemple ceea ce ajută la o dezvoltare rapidă a aplicației web.

1.2.1 Autentificarea

Pentru a naviga în aplicația web, utilizatorul trebuie sa fie autentificat pentru a avea acces la resursele sale, dacă deține. Autentificarea unui utilizator se realizează print trimiterea unui request POST la server având ca parametri adresa de email și parola utilizatorului, ce cor fi verificate printr-o interogare a bazei de date. Dacă datele oferite de către utilizator sunt valide, o noua sesiune va fi pornită, iar utilizatorul va fi redirecționat catre pagina principală.

O sesiune PHP reprezintă o metodă de a stoca informații, în variabile, pentru a putea fi folosite în mai multe pagini. Spre deosebire de cookies, informațiile nu sunt stocate pe dispozitivul utilizatorului. Așadar variabilele pot deține informații despre sesiunea unui singur utilizator, ce sunt disponibile tuturor paginilor într-o singură aplicație.

Pentru a da start la o sesiune se folosește metoda session_start(). Pentru a seta variabile într-o sesiune se folosește variabila globală PHP, $_SESSION. În exemplul de mai jos putem observa cum am stocat în variabilele sesiunii, starea uilizatorului, logat sau nu, adresa de email, numele și indentificatorul unic al utilizatorului ce este prezent în tabelul din baza de date. Când utilizatorul deconectează această sesiune este reinițializată fară nici o variabilă, iar acesta este redirecționat catre pagina de autentificare.

```
session_start();
$_SESSION["loggedin"] = true;
$_SESSION["email"] = $_POST['email'];
$_SESSION["name"] = $user["name"];
$_SESSION["user_id"] = $user["unique_id"];
header("location: index.php");
//Distrugerea unei sesiuni
session_unset();
session_destroy();
```

Pe lângă faptul că sesiunea ne ajută să cunoaștem identitatea utilizatorului ce utilizează aplicația, dar ne mai ajută să stocăm datele acestuia pentru a nu fi necesară o nouă logare a utilizatorului atunci când revine la aplicație după ce browserul a fost închis. Așadar un utilizator pentru a putea accesa o pagină din aplicație trebuie sa fie mai întâi autentificat.

1.2.2 Structura aplicației web

Fiecare pagină web are nevoie de o bară de navigație ce-i permite utilizatorului sa acceseze resursele aplicației puse la dispoziție. De asemenea este nevoie și de un footer unde se pot afișa informațiile de contact sau detalii despre aplicație ce trebuie sa fie foarte ușor de găsit de catre un utilizator ce accesează aplicația. Bara de navigație este aliniată la dreapta și conține un element ce poate accesa pagina principală, un altul pentru pagina de contact, iar ultimul element este destinat pentru profilul utilizatorului, fiind un meniu de tip drop-down, ce conține elemente ce pot accesa pagina de editare a setărilor de utilizator, un alt element către pagina ce-i permite utilizatorului să importe un traseu în format GPX și un ultim element ce reprezintă un buton pentru deconectarea utilizatorului.

În următoarele subcapitole voi discuta despre conținutul paginilor și care a fost procesul de gândire atunci când au fost realizate.

Pagina Principală

Pagina principală are structura unui jurnal ce conține un tabel, unde fiecare rând din acesta deține o descriere a traseului ce a fost înregistrat. Conține date despre numele traseului, timpul și distanța totală, precum și un buton ce redirecționează utilizatorul către o pagină web ce afișează detaliile întregului traseu. În partea de sus a pagini sunt afișate două statistici:

- Prima reprezintă un calendar al lunei curente ce afisează un grafic pe două axe. Axa x reprezentând numarul de zile din luna curentă, iar axa y reprezentând numarul de activități ce s-au efectuat într-o zi.
- A doua statistică face o comparație între numărul total de trasee ce au fost întregistrate per total și numărul de trasee ce au fost înregistrate în luna curentă.

Pagina de contact

Pagina de contact ajută utilizatorii să comunice cu dezvoltatori aplicației. Pot transmite sugestii ce pot fi implementate sau chiar probleme ce le-au întampinat pe parcursul sesiunii. De asemenea pagina de contact mai poate reprezenta un mediu în care îți poți face publicitate paginilor de socializare, cum ar fi facebook, twitter sau instagram.

Pagina de Setări

Această pagină este localizată în meniul de tip drop-down al profilului și reprezintă o modalitate prin care utilizatorul își poate schimba datele. Această pagină conține mai multe casete de text, pentru nume, email, parolă, ce odată completate se poate trimite un request POST la server ce efectuează o interogare a bazei de date, unde se verifică daca datele introduse sunt valide, cum ar fi vechea parolă, în caz ca utilizatoul dorește să fie schimbată, iar dacă aceste date sunt valide se efectuează o actualizare a bazei de date cu datele introduse. Dacă această sarcină sa efectuat cu succes, utilizatorul este redirecționat către pagina de autentificare, deoarece datele sale au fost actualizate, iar cele existente în sesiune nu mai sunt valide.

Pagina de Import

Această pagină este destinată încarcarii de conținut în format GPX. Dacă sunt utilizatori, ce până în momentul de față au folosit alte aplicații de mapat trasee sau dețin dispozitive ce le permit maparea traseelor și exportarea lor în format GPX, această pagina le permite acestora să încarce conținut. Odată încarcat, traseul mapat încarcat poate fi vizualizat mai apoi în pagina principală. Am ales formatul GPX deoarece este un format standard ce poate fi folosit de aplicațtiile GPS. Conține date despre latitudine si longitudine, ce include puncte intermediare și trasee. Fișierele GPX sunt salvate în formatul XML ce permite datelor să fie ușor importate și citite de multiple programe și servicii web. Fișierele GPX stochează trei tipuri de date și anume:

- Waypoint - include coordonatele GPS ale unui punct, de asemenea poate include și alte informații descriptive cum ar fi viteza sau altitudinea curentă
- Route - o rută include o listă de puncte/waypoints ce conduc către o destinație
- Track - include o listă de puncte ce descriu o cale

Pagina unui traseu

Această pagină este printre una dintre cele mai complicate, dar și interesante, deoarece este una dinamică, iar conținutul pe care îl afișez este în funcție de traseul ce se dorește a fi vizualizat. În partea de sus a pagini este afișat un cantainăr ce afișează numele aplicației. Dedesubtul acestui container se află un altul ce conține mapa unde

este desenat întregul traseu, împreună cu un set de markere pentru începutul traseului, sfârșitul acestuia și câte un marker pentru fiecare poză ce a fost capturată pe durata traseului. Markerul pozei este poziționat exact la coordonatele GPS unde a fost capturată poza. Sub acest container, avem un rând de containere mai mici ce conțin detaliile traseului cum ar fi distanța și timpul total, media pe kilometru și altitudinea maximă care a fost atinsă pe parcursul traseului.

Tot în această pagină este afișată și o diagramă pentru vizualizarea altitudinei atinse dintre oricare două coordonate GPS. Pe axa x avem suma distanțelor dintre oricare două coordonate GPS, iar pe axa y avem altitudinea atinsă. În Figura 1.8 putem vizualiza un exemplu.

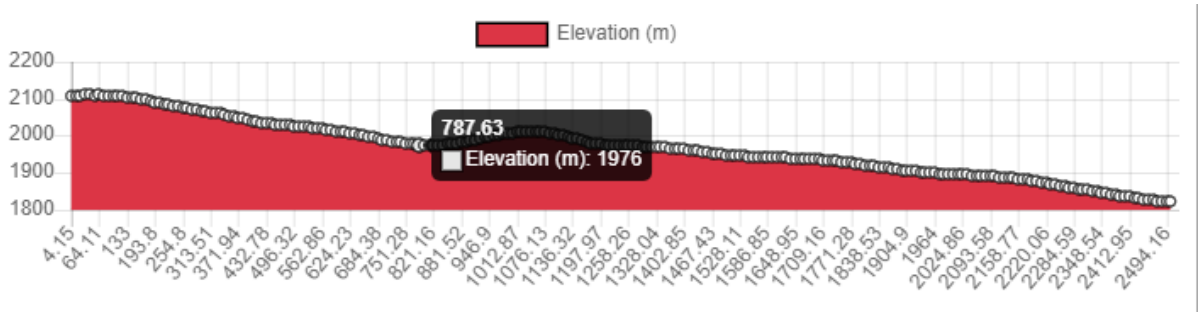

Figura 1.8: Diagrama altitudini

În josul paginii avem un ultim container ce afișează un carusel de imagini ce au fost capturate pe traseul respectiv, dacă nu a fost capturată nici o imagine, atunci nu este afișat nici un container. Pentru calcularea distanței dintre oricare două coordonate GPS am folosit formula lui Vincenty ce calculează distanța dintre două puncte pe suprafața unei sfere.

$$
\begin{aligned}
&\delta = endLong - startLong \\
&a = ((cos(endLong) * sin(\delta))^2 + \\
&+(cos(startLat) * sin(endLat) - sin(startLat) * cos(endLat) * cos(\delta))^2) \\
&b = sin(startLat) * sin(endLat) + cos(startLat) * cos(endLat) * cos(\delta) \\
&distance = atan2(\sqrt{a}, b) * 6378137
\end{aligned}
\tag{1.1}
$$

Capitolul 2

Instalarea locală a aplicațiilor

Deoarece nu am achiziționat un domeniu, am fost nevoit să-mi instalez local un server, pentru a putea testa și vizualiza aplicațiile. Pentru a putea instala și rula aplicația este necesară instalarea urmatoarelor unelte:

- Android Studio - necesar pentru a face import aplicației android
- XAMPP - cel mai popular mediu de dezvoltare pentru PHP, necesar deoarece conține o distribuție a serverului Apace, PHP, MySQL și Perl ce sunt nevoite pentru a putea rula aplicația web.
- NGROK - necesar pentru a face public serverul local ce este necesar pentru testarea aplicației android. Singurul dezavantaj îl reprezintă faptul că putem face public serverul local doar pentru o sesiune de 8 ore, mai apoi fiind necesară schimbarea adresei http cu una nouă ce este generată de catre aplicația ngrok după ce este restartat. Dacă se achiziționează o licență, acest lucru nu mai este necesar.

Importul aplicației android

Primul pas ce trebuie efectuat este descarcarea depozitului ce se află pe Github la adresa: https://github.com/a13x95/Tracker .

Pentru a importa un proiect în Andoid Studio trebuie respectați următorii pași:

- Pornește Android Studio prin a închide oricare dintre proiectele ce sunt deschise
- Din meniul Android Studio se face click pe File - New - Import Project

- Selectează folderul de proiect Eclipse ADT ce conține fișierul AndroidManifest.xml din arhiva depozitului apoi clik Ok
- Selectează folderul destinație, în cazul nostru fișierul ”Files” ce se află în arhiva depozitului ce tocmai a fost descărcat

Odată importat, proiectul ar trebui să aibă urmatoarea structură ce este prezentată in Figura 2.1.

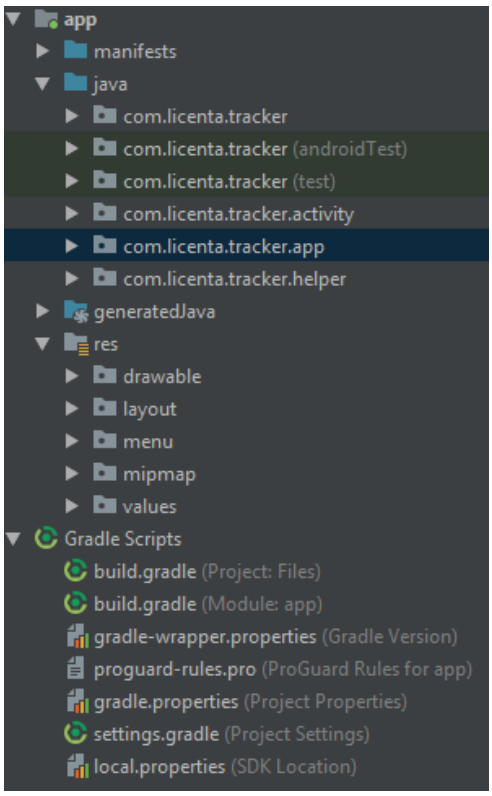

Figura 2.1: Structură proiect android

Proiectul aplicației web în XAMPP

Odată instalat XAMPP, trebuie sa mergem la locația în care a fost instalată unealta, iar în interiorul folderului ”htdocs” trebuie adăugat folderul Traker ce se află în arhiva descarcată. Următorul pas reprezintă configurarea bazei de date. Pentru a face acest lucru trebuie să lansăm aplicația XAMPP și dat start la modulul Apache și MySQL. Odată pornite putem accesa phpmyadmin de la adresa http: localhost/phpmyadmin/, aici vom selecta tabul ”SQL” ce ne permite să introducem scripturi SQL.

În fereastra aceasta vom introduce codul ce se află in scriptul descărcat din fișierul ”CreateDB_Tables.sql” ce va creea o bază de date cu numele Traker împreună cu tablele aferente, odată executat scriptul, baza de date ar trebui sa arate ca în Figura 2.2.

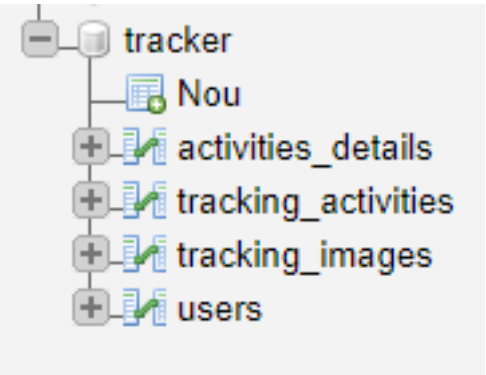

Figura 2.2: Structură baza de date

NGROK

Startul toolului ngrok se face prin rularea executabilului ”ngrok.exe”. Pentru a crea o nouă sesiune trebuie rulată în cmd comanda ”ngrok http 80”. Odată începută, sesiunea ne va oferi o adresă http ce va face public serverul nostru local. Adresa aceasta trebuie introdusa în classa AppConfig.java din proiectul android, deoarece este necesară, pentru ca la adresa respectivă se vor trimite toate requesturile aplicației ce pot fi vizualizate din fereastra cmd a toolului ngrok, exemplu ce poate fi vizualizat și in Figura 2.3.

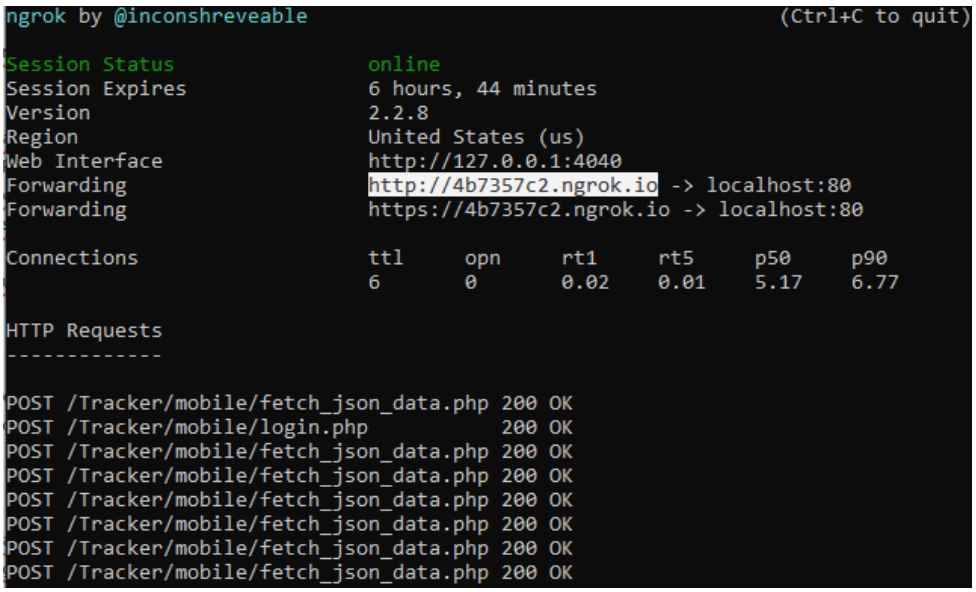

Figura 2.3: Server local făcut public

Concluzii și direcții de dezvoltare

De-a lungul anilor am început să dezvolt o pasiune din ce în ce mai mare pentru fitness, mersul cu bicicleta și ieșitul pe munte. De asemenea am folosit o gramadă de aplicații ce m-au ajutat la maparea traseelor pe care le-am parcurs. Am trecut de la aplicație la aplicație, în funcție de ce funcționalitați oferea aplicația respectivă sau oricât de rapidă putea fi din punctul de vedere al conectării la gps sau din punctul de vedere al preciziei de mapare al coordonatelor GPS.

Aplicația Tracker prezentată reprezintă un prototip al încercări mele de a creea o aplicație ce mă ajută la maparea traseelor. Folosind Android Studio,împreună cu cunoștintele legate de web design acumulate pe parcursul universității și libraria Open Street Map am reușit să implementez aplicația dorită.

Consider ca aplicația mai poate fi îmbunatățită din punctul de vedere al design-ului cât și din punctul de vedere al funcționalităților ce se pot adăuga, cum ar fi:

- Integrarea rețelelor de socializare, cum ar fi autentificarea în aplicație folosind contul de facebook, twiter, goole, precum și posibilitatea de a distribui traseele mapate pe aceste rețele de socializare, poate chiar și adaugarea de prieteni, aceștia putând vizualiza traseele înregistrate.

- Posibilitatea de a compara două trasee asemanatoare din punctul de vedere al coordonatelor GPS, adica au fost mapate pe aceași stradă, sau ruta parcursă este identică, pentru a verifica ce activitate a obținut rezultate mai bune, ce duce la generarea de noi statistici, precum cel mai bun traseu din punctul de vedere al distanței parcurse dintr-o anumită lună/an.

- Consider ca și baza de date poate suferi modificări, deoarece stocarea tuturor coordonatelor într-un singur tabel nu este o soluție eficientă, odată ce în tabel vor fi adăugate milioane de coordonate, interogarea bazei de date poate deveni înceată.

- Odată ce numărul utilizatorilor va crește iar servărul Apache va începe să raspundă greu la cereri, se poate creea o arhitectură bazată pe microservicii ce ar duce la creșterea stabilității sistemului.

Consider ca acest proiect m-a ajutat să mă dezvolt mai mult în ceea ce privește programarea pe platforma android cât și pe partea de tehnologii web. Aștept cu nerăbdare la dezvoltarea a noi aplicații ce mă vor provoca să descopăr și să învaț cât mai multe lucruri noi așa cum sa întâmplat de-a lungul dezvoltării acestui proiect.

Bibliografie

[1] Dragoș Gavriluț, Cursuri *Android Programing*, 2018

[2] Android Framework, *https://developer.android.com/*, 2019

[3] OpenStreetMap, *https://github.com/osmdroid/osmdroid/wiki*, 2018

[4] Android Volley Library, *https://developer.android.com/training/volley/*, 2018

[5] Android Hive, *https://www.androidhive.info/2014/01/how-to-create-rest-api-for-android-app-using-php-slim-and-mysql-day-12-2/*, 2018

[6] LearnOSM, *https://learnosm.org/en/beginner/*, 2018

[7] GET vs. POST, *https://www.diffen.com/difference/GET-vs-POST-HTTP-Requests*, 2018

[8] Overleaf, *https://www.overleaf.com/project*, 2018

[9] Leaflet API reference, *https://leafletjs.com/reference-1.4.0.html*, 2018

[10] Chart.js, *https://www.chartjs.org/*, 2019