UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

PROIECT DE DIPLOMĂ

Event Tracker
Aplicație mobilă pentru promovarea evenimentelor

Toma Bogdan-Nicolae

Coordonator științific:
Prof. dr. ing. Alexandru Predescu

BUCUREŞTI

2024

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT

DIPLOMA PROJECT

Event Tracker
Mobile application for promoting events

Toma Bogdan-Nicolae

Thesis advisor:
Prof. dr. ing. Alexandru Predescu

BUCHAREST

2024

CUPRINS

SINOPSIS

Într-o lume ce se află într-o continuă dezvoltare a tehnologiei și a inovării continue, fiecare dorește să găsească soluții avansate pentru a ne simplifica rutina zilnică. Astfel, Event Tracker își propune a simplifica nevoia oamenilor de recreație. Promovarea și organizarea evenimentelor necesită soluții eficiente pentru a răspunde așteptărilor oamenilor, ce doresc a găsi rapid evenimente si eficient evenimente din categoriile dorite, dar si organizatorilor, ce își doresc o platforma simpla, eficienta si portabila pentru a-si promova evenimentele. Proiectul propune o aplicație mobilă interactivă pentru vizualizarea, managementul evenimentelor, punând accentul pe feedback și transparență în promovare. Concluziile evidențiază relevanța și beneficiile acestei soluții pentru comunitatea de organizatori de evenimente, dar si pentru participanții acestora.

ABSTRACT

In a world that is in continuous technological development and innovation, everyone seeks advanced solutions to simplify our daily routines. Thus, Event Tracker aims to simplify people's need for recreation. Promoting and organizing events requires efficient solutions to meet the expectations of users who want to find events quickly and efficiently in their desired categories, as well as organizers who want a simple, efficient, and portable platform to promote their events. The project proposes an interactive mobile application for event visualization and management, emphasizing feedback and transparency in promotion. The conclusions highlight the relevance and benefits of this solution for the community of event organizers as well as for their participants.

MULȚUMIRI

Doresc să îi mulțumesc în mod deosebit domnului Asistent Inginer Mircea - Alexandru Predescu pentru îndrumarea și sprijinul acordat pe parcursul realizării acestui proiect de licență.

1 INTRODUCERE

În momentul de față, există pe piață câteva aplicații ce dispun de o gamă variată de funcționalități expuse într-un mod inovator și interactiv. Clienții pot alege diferite moduri de a-și selecta preferințele. Acestea pot include accesul la servicii predefinite, care oferă o ușoară rigiditate atunci când vine vorba de personalizarea produselor prezentate, sau utilizarea inteligenței artificiale, care poate captura și influența în mod sigur tot procesul de digitalizare.

Dezvoltarea unei astfel de aplicații trebuie să fie similară cu cerințele menționate mai sus, dar concepută pentru a se distinge prin caracteristici complexe care integrează mai multe tehnologii, oferind în cele din urmă un produs sustenabil care se adaptează nevoilor clienților.

Scopul aplicației Event Tracker este de a oferi o modalitate de comunicare și de expunere a organizatorilor de evenimente către clienții acestora, prin care se pot satisface și înțelege nevoile ambelor părți. Folosind tehnologii inovative și relativ noi, aplicația este dispusă la adaptarea continuă la noile funcționalități, dar totodată este și sustenabilă pe termen lung.

1.1 Context

Trăim într-o eră a tehnologiei, unde accesul la informații a devenit facil prin intermediul unui smartphone și al unei conexiuni la internet, utilizatorii tinzând sa prefere mai mult telefonul detrimentul laptop-ului, in ultimul timp, datorita portabilității acestuia. Antreprenorii recunosc importanța prezenței în mediul online, iar un studiu recent arată că o persoană petrece în medie 6 ore pe zi accesând internetul, astfel mediul online a devenit mediul din care oamenii își procură majoritatea informațiilor, aceștia dorind sa se digitalizeze indiferent de domeniu.

În această situație, aplicațiile mobile sunt indispensabile pentru a furniza utilizatorilor un acces rapid și simplu la informații și servicii. Există diverse categorii de aplicații pentru dispozitive mobile, inclusiv aplicații native, aplicații hibride și aplicații web mobile, fiecare având aspecte pozitive și negative distincte. Aplicațiile native oferă performanțe ridicate și o experiență optimizată pentru utilizator, însă necesită să fie dezvoltate separat pentru fiecare platformă. În schimb, aplicațiile hibride și web mobile sunt mai simple de creat și de menținut, însă pot avea performanțe mai slabe decât aplicațiile native.

Aplicația Event Tracker, propusă în acest proiect, este o aplicație mobilă dezvoltată în Kotlin și folosind un server de Java Spring Boot pentru funcționalități specifice si accesul la baza de date. Scopul său este de a simplifica promovarea și organizarea evenimentelor, oferind utilizatorilor o platformă interactivă unde pot vizualiza evenimente, lăsa recenzii și evaluări, contribuind astfel la informarea comunității. Organizatorii pot crea și gestiona evenimente, având acces la statistici detaliate pentru optimizarea acestora.

De asemenea, domeniul proiectului ar avea o mare anvergură atât pentru orașe mari precum București, unde se afla cel mai mare procent de evenimente din România, cât și pentru orașe

mai mici care doresc să își stabilească o bază solidă de clienți și să încurajeze participarea într-un mediu plăcut și relaxant.

1.2 Problema

Proiectul adresează problema accesului dificil și neorganizat la informații despre evenimentele dintr-un oraș mare, precum București. În prezent, informațiile despre evenimente sunt dispersate pe multiple platforme și nu oferă o experiență personalizată utilizatorilor. Exista o creștere exponențială a oamenilor ce doresc a participa tot mai des in multiple activități din diverse categorii, cum ar fi cele in aer liber, festivaluri, concerte sau opera.

Consider ca o astfel de aplicație, ce este ușor de folosit si aduna o gamă largă de evenimente din multe categorii, poate atrage utilizatori din diverse medii sociale sau grupuri de vârsta, consolidându-și astfel relevanța.

1.3 Obiective

Obiectivul principal al proiectului este de a dezvolta o aplicație mobilă user-friendly care să faciliteze accesul la informații despre evenimentele din București. Se urmărește crearea unei platforme intuitive ce oferă funcționalități precum:

- **Recenzii ale utilizatorilor:** Aplicația va permite utilizatorilor să lase recenzii detaliate despre recenziile la care au participat, dar si comentarii cu așteptări sau alte păreri pe care utilizatorul le are despre un eveniment ce va urma sa fie petrecut. Aceste recenzii pot sa includă evaluări bazate pe diverse criterii precum locația, organizarea sau calitatea evenimentului. De asemenea, utilizatorii vor putea sa citească și să se informeze, citind opiniile altor participanți despre evenimentul curent, sau viitoare evenimente.
- **Mapare a locațiilor:** Aaplicația va include o funcție de mapare integrată, care va afișa locațiile exacte ale evenimentelor pe o hartă interactivă. Utilizatorii vor putea vedea si accesa rapid si ușor unde se desfășoară evenimentele, împreuna cu un itinerariu generat pe baza locației curente.
- **Un sistem de feed:** Utilizatorul poate selecta categoria de evenimente dorită, împreuna cu filtre avansate de căutare după diverse criterii, cum ar fi data sau numele evenimentului, respectiv al organizatorului.
- **Suport pentru organizatori**: Organizatorii vor putea crea evenimente direct din interiorul aplicației, având la dispoziție o gama larga de detalii de adăugat, cum ar fi, descrieri, date de începere/finalizare, imagini, gamă variată de bilete, fie gratis, fie cu prețuri personalizate, la care organizatorul poate atașa un link către platforma de cumpărare efectiva a biletului.
- **Accesul ușor la evenimente** Utilizatorii vor dispune de un cod QR generat pentru fiecare eveniment, pe când organizatorii vor dispune de un scanner direct in aplicație pentru a valida aceste bilete.

Succesul proiectului ar putea conduce la creșterea participării publicului la evenimente culturale și sociale, stimulând astfel viața urbană și turismul în București.

1.4 Structura lucrării

Acest capitol conține o detaliere succintă a secțiunilor următoare, prezentată în 1-2 fraze, punând accentul pe elementele cele mai semnificative din fiecare secțiune.

1.4.2. Analiza și specificarea cerințelor

În acest capitol, se analizează rezultatul interogării potențialilor clienți, se expune motivația de la care a plecat proiectul. În final, se realizează diagrama de use-case ce definește clar funcționalitățile propuse.

1.4.3. Studiu de Piață/Abordări existente

În următorul capitol se realizează un studiu de piață, în care sunt analizate toate abordările existente deja. Acestea sunt de 3 feluri: platforme tip social-media (Facebook Events [6]), platforme centrate pe vânzarea de bilete (IaBilet [1]) și platforme tip blog pentru promovarea evenimentelor (ZileSiNopti [4]).

1.4.4. Soluția Propusă

În pasul urmator, se discuta despre tehnologiile propuse pentru implementarea soluției, si din ce cauza au fost alese. De asemenea, se prezintă relațiile dinte entitățile prezente în baza de date, împreuna cu un workflow general al aplicației.

1.4.5. Detalii de implementare

La acest pas, fiecare tehnologie menționata la capitolul anterior, este detaliată in amănunt, prezentându-se un workflow cât mai detaliat. În plus, sunt expuse și componente secundare, sau detalii de implementare specifice, ce nu au fost menționate in capitolul precedent.

1.4.6. Studiu de caz/Evaluarea rezultatelor

În acest capitol, se evaluează feedback-ul potențialilor clienți despre aplicație, punând accent pe tehnologiile cele mai apreciate. Totodată, se analizează parametrii în care rulează aplicația, cum ar fi memoria folosita și procentul de CPU folosit.

1.4.7. Concluzii

În acest ultim capitol, se compară obiectivele inițiale cu produsul final, menționându-se și dezvoltările ulterioare ce pot apărea pentru a îmbunătăți aplicația.

2 ANALIZA ȘI SPECIFICAREA CERINȚELOR

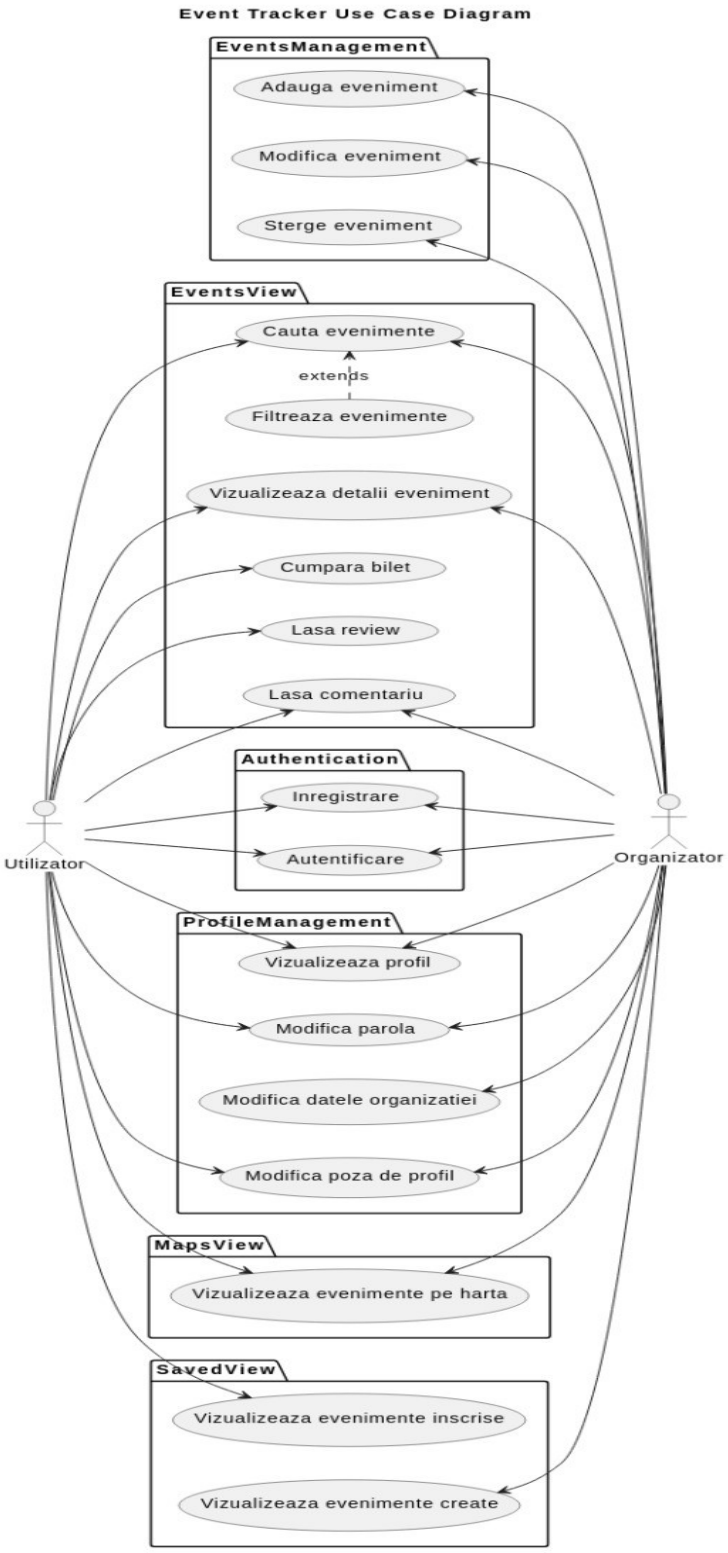
Acest capitol analizează cerințele produsului din prisma potențialilor clienți si a scenariilor de utilizare preconizate, fiind prezentata lista funcționalităților, dar totodată, introduce motivația realizării proiectului.

2.1 Rezultatul interogării clienților

Pentru a evalua cerințele si preferințele potențialilor utilizatori ai aplicației Event Tracker, am realizat un chestionar pe care l-am adresat acestora. În urma analizei răspunsurilor, am constatat:

- Procentul utilizatorilor de iOS si Android este unul asemănător (53.3% Android si 46.7% iOS)
- Prezenta la evenimente organizate devine din ce in ce mai sporita, astfel 35.6% dintre persoane au declarat ca participa săptămânal la un eveniment, pe când 33.3% au declarat ca participa lunar. Totodată, fiind si persoane mai active din punct de vedere social, ce au susținut ca se înscriu la diverse evenimente sau activități aproape zilnic (6.5%)
- Conform așteptărilor, categoriile de evenimente la care participă persoanele sunt diverse și relativ echilibrate. Evenimentele în aer liber ocupă primul loc, cu 67,4% dintre respondenți declarând că participă la acestea. La polul opus se află operele, cu doar 13% participare. Celelalte tipuri de evenimente, precum concertele, festivalurile și evenimentele sportive, au participări în jurul valorii de 50%, evidențiind un interes variat al utilizatorilor pentru diferite forme de divertisment și cultură.
- Majoritatea utilizatorilor (67,4%) consideră că găsirea rapidă a locațiilor evenimentelor este cea mai importantă caracteristică a unei funcții de mapare. Pentru 28,1%, direcțiile detaliate de deplasare sunt importante, în timp ce pentru 4,3%, vizualizarea mai multor evenimente pe o hartă este apreciată.
- Utilizarea unui sistem de feed inteligent pentru evenimente este considerată extrem de utilă de 67,4% și 28,1%, indicând că personalizarea și sugestiile pe baza preferințelor sunt esențiale pentru îmbunătățirea experienței utilizatorilor în aplicație.
- Majoritatea utilizatorilor (78,3%) au vârsta cuprinsă între 18 și 24 de ani, grupul de vârstă reprezentativ fiind de 25-34 de ani. Aplicația Event Tracker ar trebui să se concentreze pe nevoile și preferințele tinerilor adulți. Iar majoritatea acestora au declarat faptul ca își obțin informațiile despre evenimentele din București, in proporție de 80% din mediul online.

2.2 Evaluare rezultate si diagrama de Use Case



În urma interogării potențialilor clienți, am identificat o serie de funcționalități esențiale pentru aplicația Event Tracker. Aplicația va include două tipuri de utilizatori cu funcționalități specifice: organizatorul și utilizatorul obișnuit.

Organizatorul va putea crea, adăuga și modifica evenimente individuale folosind funcțiile avansate. În plus, organizatorii vor avea capacitatea de a urmări participarea clienților la eveniment.

Utilizatorul obișnuit, pe de alta parte, va avea capacitatea de a se înscrie la diferite evenimente, de a lăsa recenzii și de a evalua evenimentele la care a participat.

Diagrama de use-case anexată prezintă capacitățile și interacțiunile aplicației Event Tracker pentru fiecare tip de utilizator. Aceste funcționalități au fost dezvoltate pentru a oferi utilizatorilor noștri potențiali o experiență completă și satisfăcătoare.

2.3 Motivație

În zona urbană, mediul online ocupă un rol din ce în ce mai predominant în viețile noastre, inclusiv la locul de muncă. Această tendință de digitalizare intensificată a dus la o creștere a timpului petrecut în casă, la un stil de viață sedentar, și la o reducere semnificativă a interacțiunilor sociale fizice. În acest context, oamenii resimt o nevoie acută de a găsi noi modalități de recreere și socializare.

Astfel, a apărut motivația pentru dezvoltarea unei aplicații mobile ușor de utilizat, care să le ofere utilizatorilor multiple oportunități de a participa la diverse evenimente. Această aplicație își propune să încurajeze interacțiunile sociale și participarea activă la viața comunității, oferind un mijloc eficient de a descoperi și a se implica în evenimentele locale. Prin facilitarea accesului rapid la informații despre evenimente și prin integrarea unor funcționalități inovatoare, aplicația va răspunde nevoilor actuale de socializare și recreere ale utilizatorilor urbani.

3 STUDIU DE PIAȚĂ / ABORDĂRI EXISTENTE

În piața actuală există o varietate de platforme pentru evenimente în românia, aceste se află într-un format divers oferind utilizatorilor soluții variate. Platformele de evenimente se pot împărți pe mai multe categorii:

1) Platforme centrate pe vânzarea de bilete.

Acest tip de platforme au ca obiectiv personal distribuirea de bilete pentru diverse evenimente. Focusul principal fiind pe achizitia și validarea biletelor. În general pe astfel de platforme promovarea evenimentelor se face extern pe platforme de social media. Conform site-ului oficial IaBilet.ro detine urmatoarele cotari pe piata:

- Peste 3000 de clienti activi B2B
- Peste 3 milioane de bilete vandute anual
- Peste 3 milioane de vizualizari unice lunare ale platformei
- Peste 10 milioane de pagini accesate pe luna
- Peste 1 milion de abonati la newsletter
- Peste 600.000 de utilizatori ai aplicatiei online

A. IaBilet.ro

IaBilet.ro este probabil cea mai cunoscuta platforma de evenimente la ora actuala. Aceasta si-a castigat pozitia ca lider de piata in domeniu fiind platforma principala prin care organizatorii de evenimente vand bilete pentru diversele spectacole organizate de acestia.

Ca si tip-uri de utilizatori, platforma se focuseaza pe urmatoarele 2 categorii:

1) Organizatori de evenimente.

Pentru aceasta prima categorie sunt adresate cele mai multe functionalitati, crearea de anunt, adaguarea de bilete, editarea anunturilor si incadradea acestora in categorii pentru o identificare mai usoare. IaBilet.ro ofera acestora posibilitatea de a pune spre vanzare biletele, dar si procesarea platilor. Pentru crearea acestui tip de utlizator este necesara contactarea personalului IabBilet.ro prin mail [1] .

Pe langa posibilitatea de a crea evenimente, platforma ofera urmatoarele tip-uri de servicii catre utilizatorii sai [2]:

- Customizarea biletelor
- Aplicatie de scanare a biletelor
- Posibilitate de a gestiona locatile detinute
- Statistici in timp real despre vanzarea bileteloar atat prin site cat si prin aplicatia mobile
- Statistici in timp real despre biletele scanate in timpul unui eveniment
- Posibilitatea de a seta pachete de discount pentru achizitionarea biletelor

2) Consumatori de evenimente (utilizator standard)

Acest tip de utilizatori reprezinta cea mai mare proportia din baza de utilizatori a-i platformei. Pentru utilizatori normali platforma ofera urmatoarele functionalitati:

- Posibilitatea de a achizitiona bilete online
- Posibilitatea de a vedea biletele achizitionate
- Posibilitatea de a introduce coduri de voucher si a le aplica pe mai multe evenimente
- O pagina principala pentru cautarea evenimentelor dupa diverse categorii precum (tip, locatie).

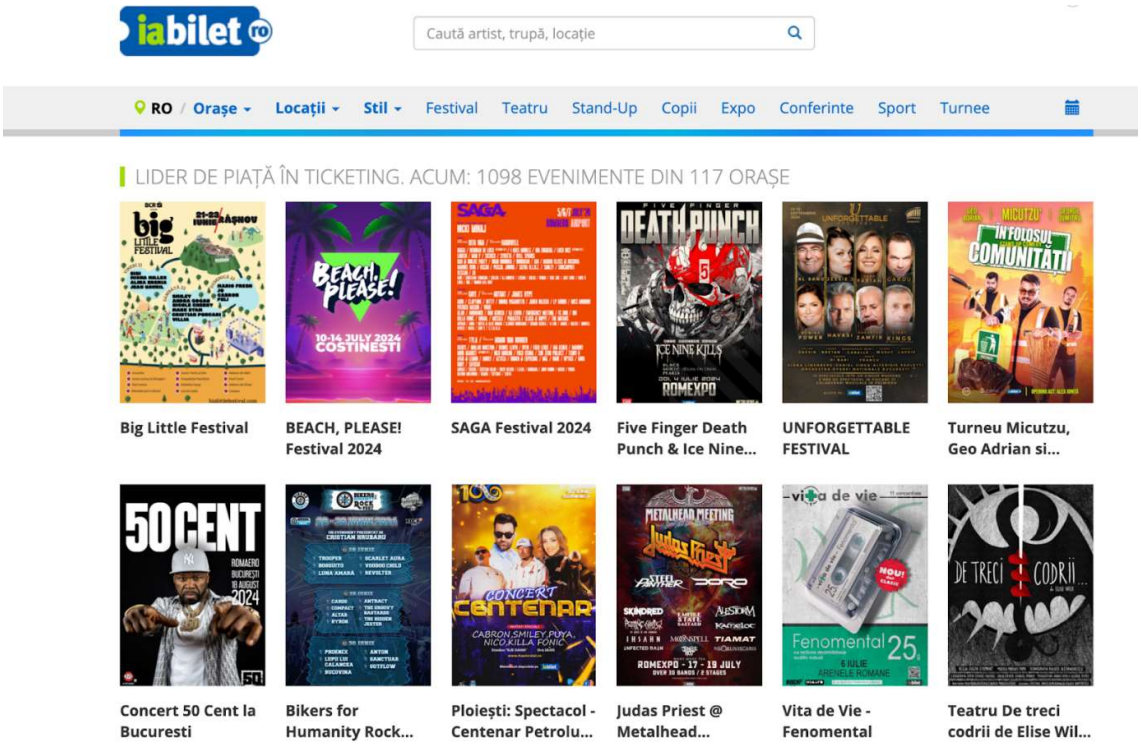


Fig 1. Pagina principala destinata utilizatorilor clasici [1]

B. LiveTickets.ro

Această platformă este a doua cea mai cunoscută platforma pentru comercializare de bilete. Ca și funcționalitate este asemănătoare cu platforma anterior menționată însă vine cu cateva avantaje pentru ambele tipuri de utilizatori.

1) Organizator de evenimente: pentru acest tip de utilizatori platforma oferă următoarele funcționalități: [3]
   - Posibilitatea de a gestiona evenimentele din platformă
   - Aplicație de scanare a biletelor
   - Dashboard pentru vizualizarea statisticilor în timp real

Avantaj: Înregistrarea unui cont de utilizator este simplă și gratuită. Spre deosebire de IaBilet.ro, unde este necesară discuția cu un reprezentant al companiei.

2) Consumator de evenimente:
    - Lista de evenimente
    - Posibilitatea de a achiziționa bilete
    - Posibilitatea de a urmări organizatori de evenimente pentru a fi notificat de viitoare evenimente
    - Platforma vine cu un aspect mai modern și mai ușor de utilizat.

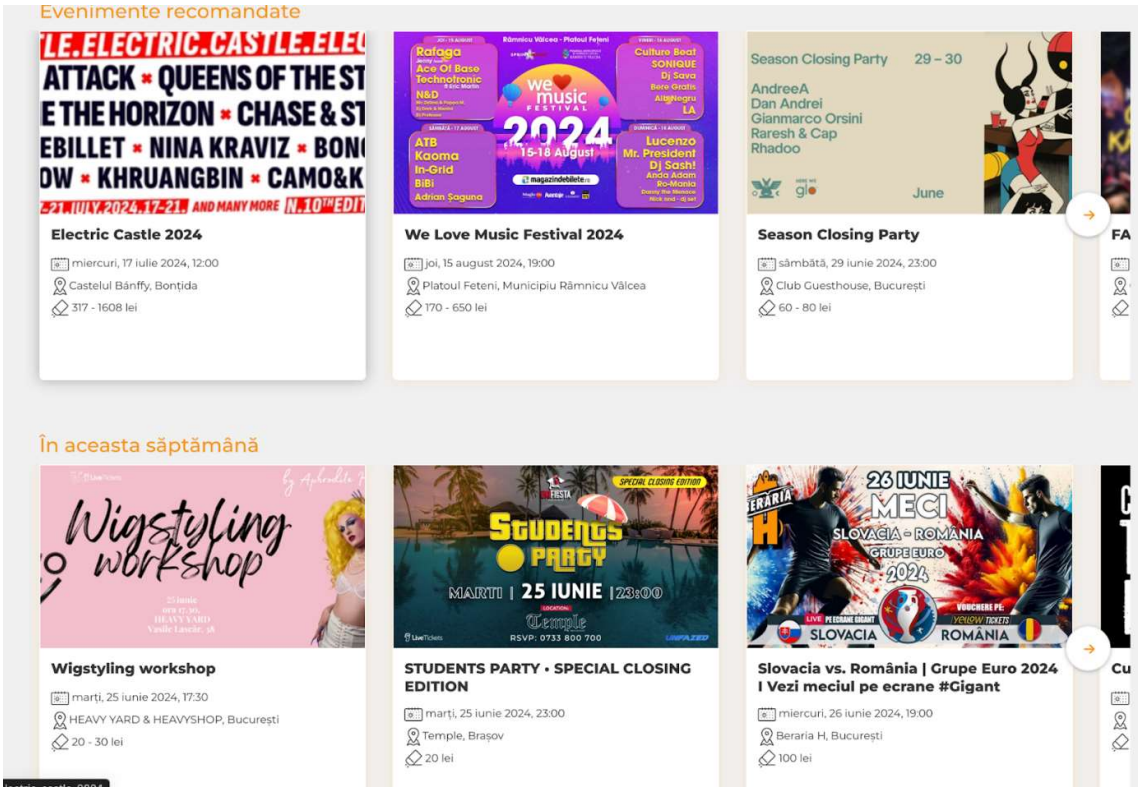


Fig2. Pagina principala de evenimente [3]

2) Platforme tip blog pentru promovarea evenimentelor

Acest tip de platforme se focuseaza pe promovarea evenimentelor prin intermediul postarilor de articole despre acele evenimente. De multe ori focusul principal al platformei nu este de a oferi un mod eficient de a cauta evenimente, platformele de acest tip se focuseaza mai mult pe a oferi o informatie calitativa despre evenimentele de interes in detrimentul unei informatii cantitative.

A. Zilesinopti.ro

Aceasta platforma este una de tip blog, nișată pe zona de evenimente. Aici pot fi găsite informații și noutăți despre evenimentele de interes pentru vizitator.

Tipuri de utilizatori:

1) Vizitator:
   - Poate vizualiza diversele postări despre evenimente.
   - Poate căuta noutăți după evenimente dintr-o anumită locație, perioadă sau tip
   - Poate lăsa comentarii la postările despre evenimente.
2) Editor:
   - Poate face anunțuri în platformă

Spre deosebire de platformele de vânzare de bilete, acestă platformă se axează mai mult pe promovarea evenimentelor decât pe gestiunea lor.

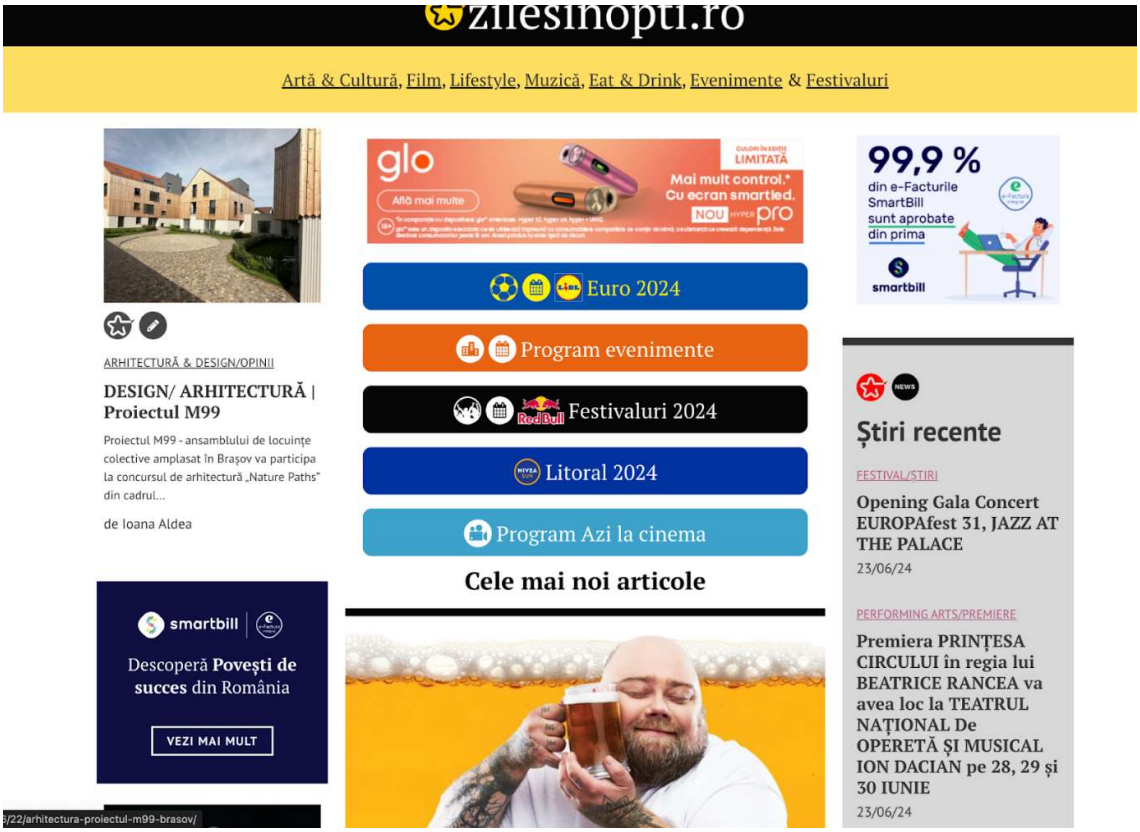


Fig3. Pagina principala [4]

B. Timeoutbucuresti.ro

Similar cu prima platformă, timeoutbucuresti.ro este un blog despre evenimente. Spre deosebire de prima platformă menționată, așa cum reiese și din nume, timeoutbucuresti.ro se axează doar pe evenimentele din capitală. Această platformă oferă un avantaj utilizatorilor care sunt interesați de evenimente stric din București.

Tipuri de utilizatori:

1) Vizitator
2) Editor

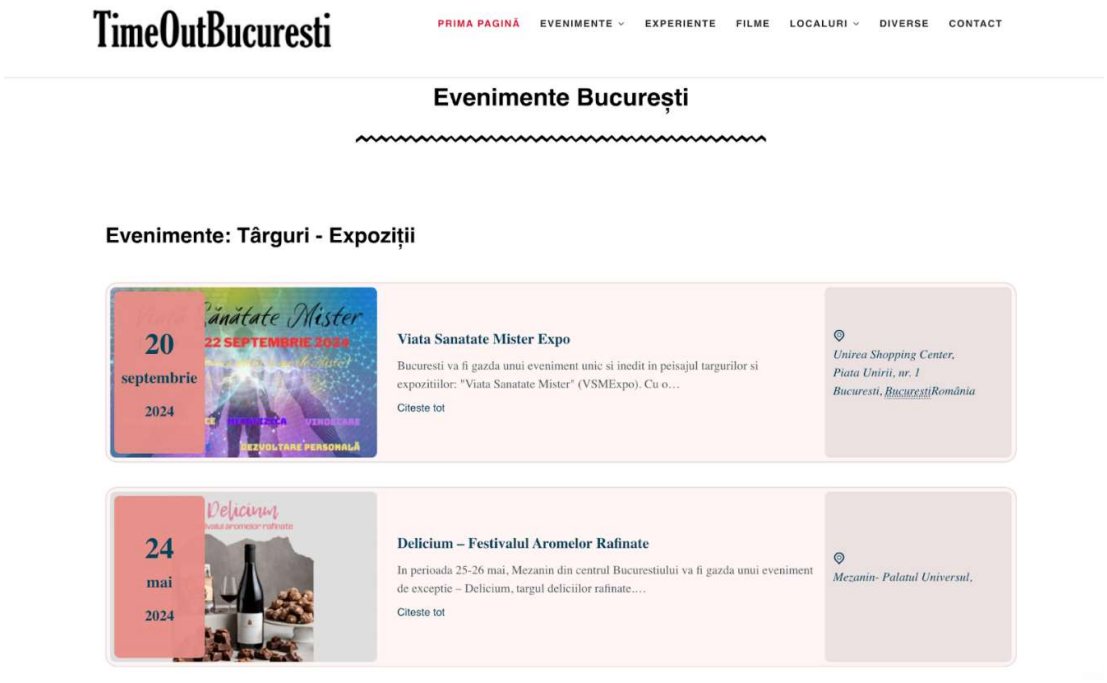


Fig 4. Pagina principală [4]

3) Platforme tip social-media

Aici intervin marile platforme de social media. În contextul promovării evenimentelor, este esențial să considerăm și aspectul social al acestora: câte persoane vor participa? ce tip de public va fi prezent la eveniment? ce cunoscuți sunt interesați de următorul eveniment?

A. Facebook

Facebook este cea mai mare platformă de socializare din prezent. Este o platformă care se bucură de un număr de 2.9 miliarde de utilizatori activi lunar. Facebook s-a schimbat de mult de la o simplă platformă de socializare la o platformă cu funcționalități multiple precum organizare de evenimente și magazin online.

În continuare lucrarea va detalia funcționalitatea de organizare de evenimente, aceasta fiind cea de interes pentru aplicația propusă.

Funcționalități oferite:

- Posibilitatea de a crea evenimente
- Posibiliitatea de a promova evenimentele într-un mod optim prin Facebook Ads
- Posibilitatea de a monitoriza engagement-ul utilizatorilor cu evenimentele prin facebook analytics.
- Posibilitatea de a căuta evenimente după locație, interese, sau prieteni comuni care participă

Impactul pe care îl aduce componenta socială în zona organizării de evenimente este unul semnificativ, organizatorii de evenimente putând astfel să ajungă la o audiență mai mare și să atragă participanți mai ușor. Ce nu oferă facebook events este posibilitatea de a gestiona biletele sau a pune spre vânzare bilete pentru participarea unor evenimente.

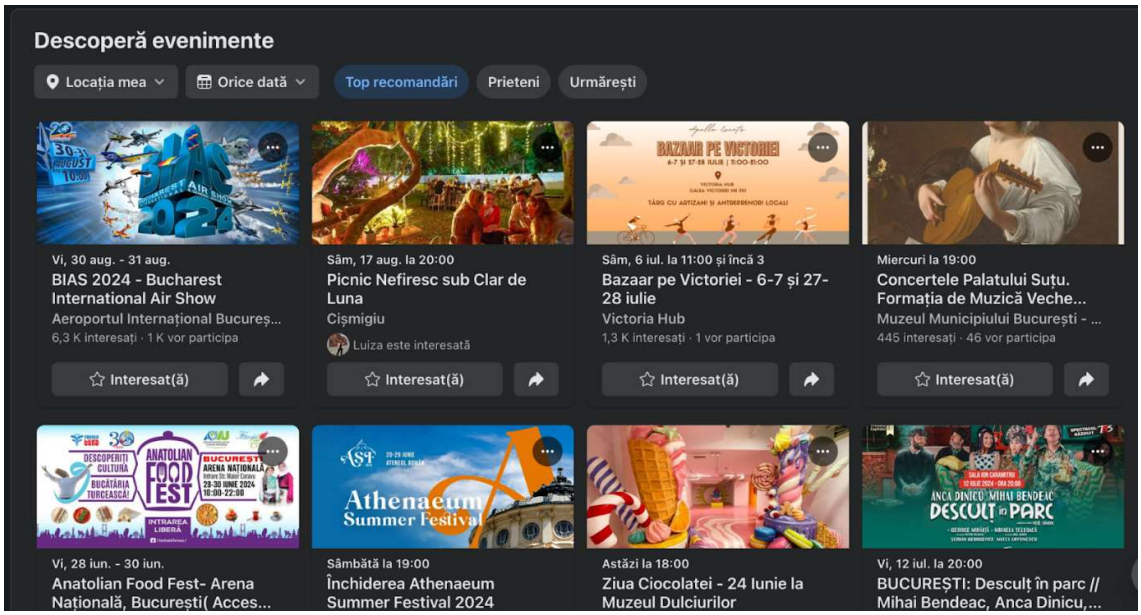


Fig 5. Pagina de evenimente [6].

Concluzie:

În prezent există o gamă variată de platforme de evenimente. Acestea se pot împărți pe 3 mari categori menționate anterior (vânzare de bilete, blog-uri, social media). Fiecare componentă are un impact pozitiv pentru buna organizare a evenimentelor. O aplicație care poate aduce impact în acest domeniu trebuie să poată să combine aceste 3 componente într-un mod optim și la îndemâna utilizatorului.

4 SOLUȚIA PROPUSĂ

4.1 Cerințe Sistem

Cerințele de sistem pentru a putea rula aplicația sunt:

Pentru server:

- Sistem de operare: Windows, Linux sau Mac
- Versiune Java: mai mare decât 17
- Spring boot: 3.3.1

Baza de date:

- PostgreSQL 12 sau versiuni ulterioare

Pentru aplicatia mobila:

- Sistem de operare telefon: Android 6.0 sau versiuni ulterioare
- IDE: Android Studio 4.1 sau versiuni ulterioare
- SDK: Android SDK 26 sau mai nou

4.2 Descrierea arhitecturii

Arhitectura aplicației Event Tracker este structurata pe in doua componente principale: Serverul de backend, ce comunica cu baza de date, si aplicația mobila ce comunica cu serverul. Această structură a fost aleasă pentru a asigura scalabilitatea proiectului și pentru a permite extinderea viitoare a aplicației și pe platformele web.

4.2.1 Limbajul Java

Java este un limbaj de programare „high-level”, bazat pe clase și orientat pe obiecte. Arhitectura sa este proiectată pentru a minimiza dependențele între diferitele implementări, ceea ce îi conferă o portabilitate ridicată. Ca limbaj de programare general, Java permite dezvoltatorilor să scrie cod o singură dată și să-l execute pe orice platformă fără a fi necesară recompilarea atunci când codul este mutat pe o altă arhitectură. Această caracteristică, cunoscută sub numele de "scrie o dată, rulează oriunde" (WORA), este unul dintre motivele pentru care Java este atât de popular și versatil în dezvoltarea de aplicații. [12]

Avantajele acestui limbaj de programare:

1) Java este un limbaj de programare orientat pe obiect, ceea ce îi conferă o arhitectură modulară, flexibilă și extensibilă. Această abordare permite utilizarea claselor și a moștenirii pentru a structura codul eficient, facilitând astfel dezvoltarea de aplicații complexe și ușor de întreținut. Datorită orientării sale pe obiecte, Java promovează reutilizarea codului și îmbunătățirea clarității și organizării acestuia.

2) Datorită JVM-ului (Java Virtual Machine), Java este un limbaj cross-platform care poate fi rulat pe orice arhitectură. Codul scris în Java este compilat în bytecode, care poate fi executat pe orice platformă care suportă JVM.

3) Java are o sintaxă foarte simplă și familiară pentru cei care utilizează C sau C++, reducând riscul de erori complexe, cum ar fi problemele legate de manipularea pointerilor sau moștenirea multiplă.

4) Java oferă câteva funcționalități built-in esențiale, cum ar fi verificarea bytecode-ului, încărcarea securizată a claselor și un manager de securitate care restricționează rularea aplicației dacă sunt detectate erori ce pot apărea în timpul execuției.

5) De asemenea, Java beneficiază de un management robust al memoriei, un sistem eficient de gestionare a excepțiilor și mecanisme de detecție a erorilor la compilare, asigurând astfel o mai mare stabilitate si fiabilitate a aplicațiilor dezvoltate.

6) Java ofera suport pentru multithread la nivel de limbaj, care permite programatorilor să ruleze numeroase task-uri simultan.

Mediul Java si componentele lui:

Mediul Java este alcătuit din numeroase componente care funcționează simultan pentru a permite rularea aplicațiilor Java.

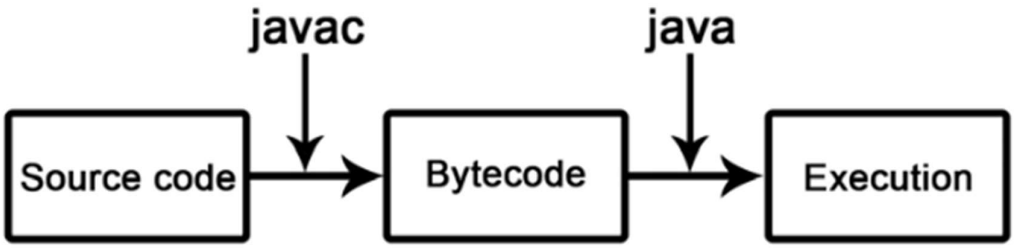


Fig 1. Reprezentarea ecosistemului Java [13]

JDK (Java Development Kit) este un mediu de dezvoltare software folosit pentru aplicații Java. Include JRE (Java Runtime Environment) și un set complet de instrumente necesare dezvoltarii: un interpretor sau încărcător (Java), un compilator (javac), un arhivator (jar), un generator de documentație (javadoc), și alte unelte esențiale.

Java Runtime Environment (JRE) furnizează librării, JVM (Java Virtual Machine) și alte componente necesare pentru a rula aplicațiile scrise în Java. JRE-ul nu include însă unelte de dezvoltare cum ar fi compilatoare sau instrumente de depanare.

Java Virtual Machine (JVM) este o mașină abstractă de calcul care permite altor mașini să execute programe Java. În momentul în care scrii un program în Java, codul sursă este compilat în bytecode, un format pe care JVM-ul îl poate interpreta și executa.

Java este limbajul, fiind un limbaj „high-level”, conține multe funcționalități esențiale, ce ar fi mult mai greu de implementat in alte limbaje. Câteva dintre acestea ar fi:

- Gestionarea erorilor: Java oferă un mecanism foarte puternic de gestionare a erorilor de runtime prin excepții. Acest lucru permite un flux normal al aplicației chiar dacă întâmpină o eroare.
- Framework-ul "Collection": Java Collections Framework (JCF) oferă un set de interfețe și clase care manipulează un set de obiecte. Include clase precum: ArrayList, HashSet, HashMap.

Aceste concepte avansate sunt esențiale în dezvoltarea aplicațiilor Java moderne și contribuie semnificativ la eficiența și scalabilitatea acestora.

Ecosistemul Java:

1) Mediul de dezvoltare: Printre cele mai renumite medii de dezvoltare pentru Java se numără: IntelliJ IDEA, NetBeans, care oferă robustețe pentru procesul de codare, debugging și testarea aplicațiilor în Java.

2) Tool-urile pentru Build: Tool-uri precum Maven sau Gradle sunt folosite automat de către procesul de build, ajutând în managementul dependințelor.

3) Framework-uri și librării: Java oferă un bogat sistem de framework-uri și librării menite să simplifice procesul de dezvoltare software, precum Spring, Hibernate și Apache Struts.

Aceste resurse sunt fundamentale în dezvoltarea eficientă și scalabilă a aplicațiilor Java, contribuind la creșterea productivității și la gestionarea corectă a dependințelor și a procesului de build.

Concluzii Java:

Java oferă o combinație de întrebuințări atât ca mediu de dezvoltare, cât și ca mediu cross-compile pentru rularea aplicațiilor pe diferite mașini cu arhitecturi variate. Java rămâne un limbaj de programare versatil și în zilele noastre.

4.2.2 Spring FrameWork

Spring este un framework de tip open-source pentru construirea aplicațiilor Java. A fost creat inițial de către Rod Johnson în jurul anului 2003 și este acum întreținut de către SpringSource, o divizie a companiei Pivotal Software. Este unul dintre cele mai populare și influente framework-uri pentru dezvoltarea aplicațiilor Java.

Scopul si filosofia framework-ului Spring este de a simplifica dezvoltarea de aplicații enterprise in Java prin abordarea unor provocări comune precum: gestionarea memoriei,

accesul la baza de date, securitatea aplicației și altele. Cheile principale ale acestui framework include [14]:

1**. Inversion of Control (IoC):** Spring împinge în față acest proces prin container-ul Spring, care administrează și controlează ciclul de viață al obiectelor în timpul rulării aplicației.

2. **Aspect-Oriented Programming (AOP):** Spring suportă AOP, oferind separarea îngrijorărilor de tip cross-cutting (precum logging, securitate, tranzacționare) de către business logic.

3. **Simplificarea dezvoltării**: Spring reduce complexitatea prin configurarea și abstractizarea detaliilor tehnice, permițând dezvoltatorilor să se concentreze doar pe implementarea facilităților business.

Câteva dintre modulele esențiale ale framework-ului Spring, utile pentru dezvoltarea aplicației sunt [9]:

1. **Spring JDBC**: Simplifică dezvoltarea aplicațiilor bazate pe JDBC prin eliminarea codului repetitiv și eficientizarea resurselor.

2. **Spring MVC:** Spring MVC este un framework web MVC (Model-View-Controller) care facilitează dezvoltarea aplicațiilor scalabile și robuste.

3. **Spring Security:** Securitatea în Spring oferă suport pentru autentificare, autorizare și alte aspecte de securitate în aplicațiile Java.

4. **Spring Boot:** Simplifică dezvoltarea de aplicații prin furnizarea unui setup automat pentru configurare și gestionarea dependențelor.

Avantajele utilizarii framework-ului Spring:

1. **Modularitate și Extensibilitate**: Spring Framework este modular și oferă integrarea cu alte framework-uri și librării.

2. **Testabilitate**: Componentele Spring sunt ușor de testat, permițând dezvoltatorilor să implementeze eficient teste unitare și de integrare.

3. **Suport activ și o comunitate largă**: Spring are o comunitate foarte mare și puternică, oferind suport constant și resurse abundente.

Concluzii

Spring Framework este un puternic și versatil instrument pentru construirea aplicațiilor enterprise în Java. Cu module și funcționalități multiple, Spring oferă scalabilitate, securitate și mentenanța aplicațiilor.

4.2.3 Introducere in Kotlin pentru dezvoltarea aplicației Android

Fig. 2 Kotlin MultiPlatform [17]

Kotlin este un limbaj de programare static, dezvoltat de către JetBrains, același dezvoltator care a creat și IntelliJ IDEA. Arhitectura lui a fost făcută pentru a funcționa perfect cu Java, dar și pentru a oferi funcționalități de limbaj modern pe care Java nu le încorporează. Prima versiune stabilă de Kotlin a fost în 2016 și a fost adoptată rapid de dezvoltatori, ceea ce a dus la o versiune securizată a limbajului

În 2017, Google a anunțat suport oficial pentru Kotlin pe Android, făcându-l primul limbaj de programare pentru dezvoltarea aplicațiilor care funcționează alături de Java și C++. Acest lucru a crescut popularitatea Kotlin, mulți dezvoltatori începând să-l folosească în proiectele lor Android. Sintaxa Kotlin este foarte concisă și expresivă, ceea ce adesea duce la mai puține linii de cod decât în alte limbaje. În plus, este un limbaj "null safety", care ajută la eliminarea erorilor de tip NullPointerException în aplicațiile Java.

Pentru dezvoltarea unui proiect Android folosind Kotlin, trebuie mai întâi configurat un mediu de dezvoltare. Cel mai utilizat IDE pentru dezvoltarea aplicațiilor Android este Android Studio, care oferă suport integrat și pentru Kotlin.

Principalul motiv pentru alegerea limbajului Kotlin pentru dezvoltarea aplicației mobile este compatibilitatea pe care o are atât cu Android, cat si cu iOS. Prin intermediul a Kotlin Multiplatform [15] codul poate fi împărtășit de către ambele sisteme de operare, astfel aplicația ar putea fi extinsa pentru a fi cross-platform, în urma unei dezvoltări ulterioare.

4.2.4 Crearea interfețelor pentru utilizatori folosind Jetpack Compose

Jetpack Compose este un toolkit modern pentru Android creat pentru a dezvolta interfețe utilizator native. Simplifică și accelerează procesul de dezvoltare a UI-ului prin scrierea de cod mai puțin, cu unelte foarte puternice și integrarea unui API Kotlin foarte intuitiv. Compose este declarativ, însemnând că descrii UI-ul într-un mod clar și consistent, starea UI-ului la actualizandu-se la fiecare modificare.

Jetpack oferă posibilitatea de a crea componente Composable. Aceste funcționalități îți permit să definești UI-ul într-un mod programabil, acționând automat la schimbările de stare, facilitând procesul de construcție dinamică și crearea de UI-uri interactive.

De asemenea, Kotlin Compose extinde funcționalitatea de cross-platform a limbajului Kotlin si pe partea de UI, nu doar pe partea de business logic.

Activity in contextul Kotlin Compose:

Spre deosebire de modelul traditional Android, unde fiecare Activity sau Fragment gestionează un singur ecran sau o parte a interfeței, Kotlin Compose permite gestionarea întregii interfețe într-un singur CompenentActivity, ce extinde Activity, si furnizeaza suport pentru utilizarea componentelor arhitecturale Jetpack, cum ar fi ViewModel, LiveData si Compose.

State Management în Kotlin Compose:

State Management-ul Compose este crucial în crearea UI-urilor interactive. Compose folosește metode declarative, însemnând ca UI-urile descrise se bazează pe stări. Când o stare se schimba, UI-ul este automat updatat și aceste stări sunt modificate. Un exemplu simplu de modificare a UI-ului, in functie de stare, este:

```
@Composable
fun UserInput() {
    val text = remember { mutableStateOf("") }

    TextField(
        value = text.value,
        onValueChange = { newText -> text.value = newText },
        label = { Text("Enter something") }
    )
}
```

Fig.1 State Management in compose [16]

În rezumat, Jetpack Compose revoluționează metoda de creare a UI-urilor în Android oferind o abordare bazată pe stări, simplificând UI-ul și interacțiunile, reducând codul redundant. Cu Compose, poți construi într-o manieră responsivă, dinamică și, printr-o viziune mai bună, mai plăcută și mai eficientă.

4.3 Introducere in PostgreSQL

PostgreSQL, denumită și în mod popular ca PostGres, este o bază de date powerful, open-source, relațională cu o reputație puternică pentru fiabilitate, robustețe și performanță. Inițial, a fost lansată în 1989 și de atunci a cunoscut o creștere continuă a comunității. PostgreSQL este de tip ACID, ceea ce înseamnă că oferă fiabilitate în tranzacții și suport

pentru numeroase funcționalități precum interogări complexe, chei străine, triggere, view-uri actualizabile și proceduri de stocare.

Unul dintre cele mai puternice caracteristici este extensibilitatea, care oferă utilizatorilor posibilitatea să își definească propriile tipuri de date, tipuri de index, limbaje funcționale și multe altele. Această flexibilitate face ca Postgres să fie perfect pentru o gamă largă de aplicații, de la aplicații web simple la aplicații complexe.

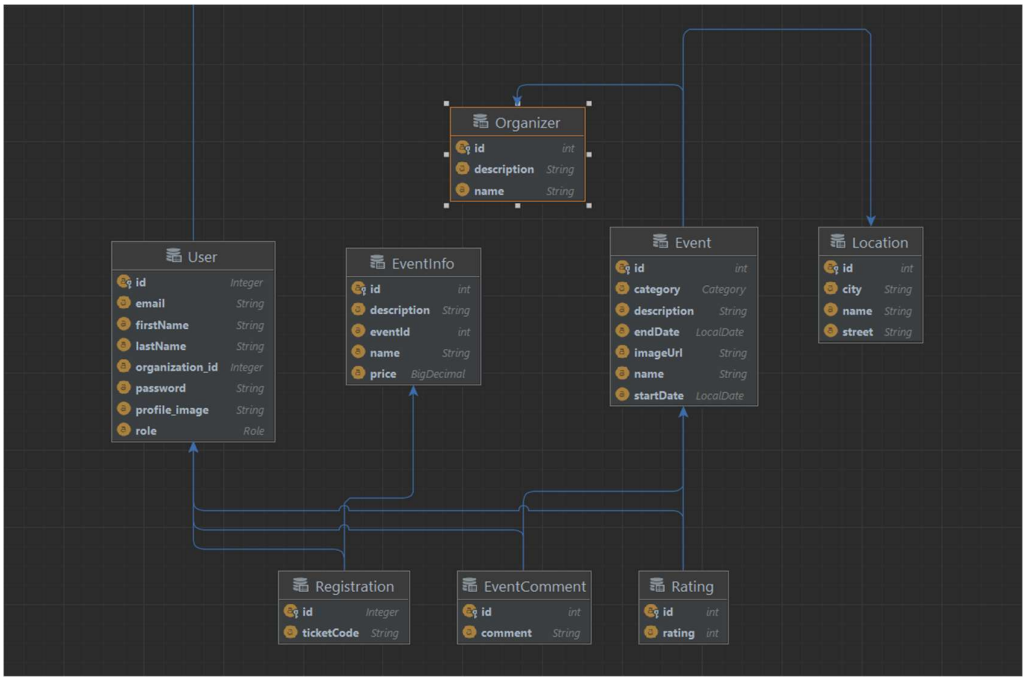
Avantajele PostgreSQL:

1) Robust, fiabil și stabil: PostgreSQL este cunoscut pentru robustețea și fiabilitatea sa. Acesta dispune de Multi-Version Concurrency Control (MVCC), care gestionează tranzacțiile simultane cu conflicte, asigurând integritatea și îmbunătățind concurența. Baza de date este extrem de stabilă, oferind caracteristici cum ar fi "recuperarea în timp", tabele și replicile sincronizate.
2) Sursă deschisă și comunitate: PostgreSQL este gratuit și dezvoltat continuu de către dezvoltatori, oferind rezolvarea continuă a erorilor, documentație și suport. Nu implică costuri de licențiere, reducând în mod semnificativ costurile de aplicare.
3) Tipuri avansate de date și indexare: PostgreSQL suportă o varietate de tipuri de date, inclusiv JSON, XML și hstore. De asemenea, oferă metode avansate de indexare, cum ar fi B-tree, hash, GiST, SP-GiST, GIN și BRIN, optimizând accesul la date pentru diferite tipuri de date și metode de indeksare.
4) Extensibilitate și personalizare: PostgreSQL este extrem de extensibil, permițând utilizatorilor să adauge noi tipuri de date, funcții, operatori și metode de indexare. Printre extensiile populare se numără PostGIS și pgAdmin, un instrument grafic puternic pentru gestionarea bazelor de date.
5) Standarde puternice și consecvente: PostgreSQL respectă cele mai noi standarde SQL, asigurând compatibilitatea cu aplicațiile SQL și facilitând migrarea către alte baze de date SQL. Acesta include ANSI-SQL și SQL: 2008, permițând dezvoltatorilor să scrie SQL într-un mod portabil, care funcționează pe baze de date diferite.

Concluzionând, am ales PostgreSQL, fiind un tool versatil și foarte puternic pentru managementul bazelor de date, care oferă numeroase avantaje precum robustețe, performanță, extensibilitate și standarde puternice. Prin natura sa open-source și cu o comunitate foarte activă, PostgreSQL beneficiază de îmbunătățiri constante. Aceste caracteristici fac ca PostgreSQL să fie o excelentă alegere pentru o multitudine de aplicații.

4.3.2 Integrare PostgreSQL

Diagrama bazei de date, pentru aplicația Event Tracker, este următoarea:



Soluția propusa include crearea a 8 tabele de entități fiecare cu funcții si relații diferite:

1) Tabelul Event conține detalii despre un eveniment, cum ar fi numele, categoria din care face parte, data de începere, cat si de încheiere, descriere, dar si o relație de ManyToOne către tabelul de Organizatori (ce conține numele si descrierea unui grup de organizatori) si spre cel de locații (ce conține numele si adresa locației). Se folosește aceasta relații între tabele deoarece pot exista mai multe evenimente care sa aibă același organizator, sau aceeași locație.
2) Tabelul de User conține detalii despre un utilizator, cum ar fi numele, adresa de email, poza de profil din aplicație, dar și rolul acestuia (organizator sau user). Tabelul are relatii de OneToMany cu tabelul Registration (înregistrarea unui utilizator la un eveniment), tabelul EventComment si tabelul Rating.
3) Tabelul EventInfo reprezintă un bilet al unui eveniment, acesta avand o relatie de OneToMany cu Tabelul de Registrations
4) Tabelele EventCommnet si Rating au o relație de ManyToOne atât cu tabelul de evenimente, cat si cu cel de utilizatori, acestea având rolul de a stoca recenziile utilizatorilor despre eveniment.

4.4 Workflow general

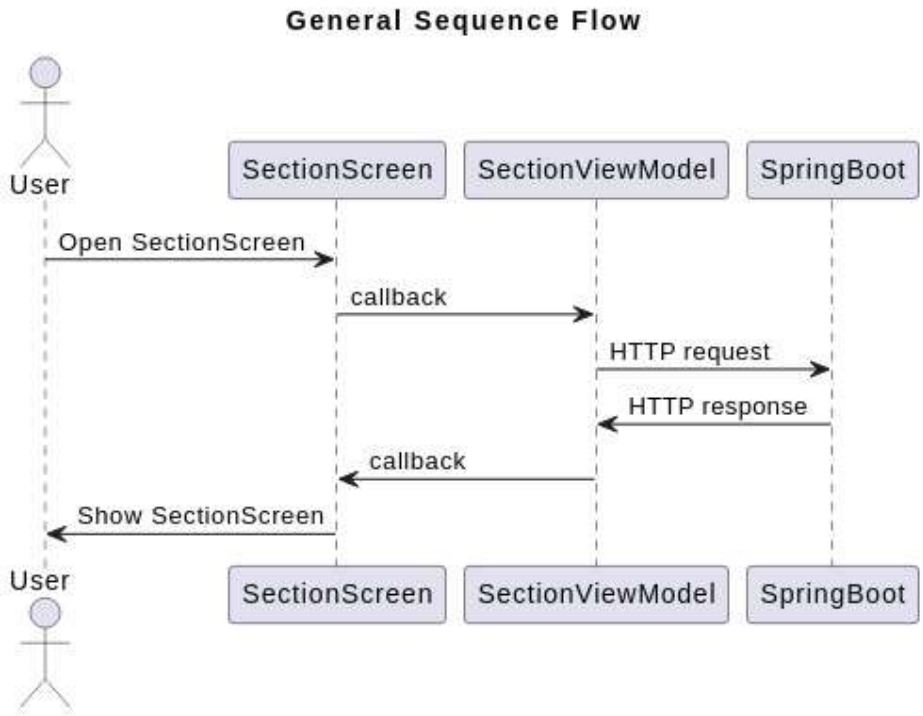
Sumarizând ideile expuse, proiectul este structurat pe trei straturi:

- Screens: ce reprezintă interfața utilizatorului (UI) și sunt dezvoltate utilizând Kotlin Compose Fiecare ecran este proiectat pentru a oferi utilizatorilor o experiență interactivă și intuitivă. Aceste ecrane includ diverse componente vizuale și

elemente de navigare care permit utilizatorilor să interacționeze cu aplicația într-un mod eficient și plăcut.

- ViewModels: ce se ocupa de partea de business. Ele sunt responsabile pentru procesarea cererilor venite de la interfața utilizatorului, actualizarea stării UI-ului și interacționarea cu datele de la server. Scrise în Kotlin, ViewModels separă clar logica de business de logica de prezentare și gestionează stările și actualizarea interfeței utilizatorului în funcție de schimbările din date.
- Server: gestioneaza partea de backend a aplicației, inclusiv interacțiunea cu baza de date. Prin intermediul API-urilor, serverul furnizează datele necesare către ViewModels, care le procesează și le afișează pe ecranele aplicației.

Pentru a oferi o imagine mai clară a funcționării aplicației, prezentăm diagrama de mai jos, care ilustrează workflow-ul general în cadrul aplicației:



5 DETALII DE IMPLEMENTARE

Acest capitol va conține elemente specifice ale rezolvării problemei și va fi detaliat modul in care au fost utilizate tehnologiile. De asemenea, se va parcurge un workflow tehnic mai detaliat ce va conține implementarea amănunțita a proiectului.

5.1 Implementare Server

Pentru a pune bazele proiectului, am utilizat Spring Initializr, care permite generarea rapidă și automată a scheletului serverului cu dependințele esențiale. [7]

1) Tabelul din baza de date

Pentru crearea si sincronizarea tabelelor din baza de date utilizam următorul format:

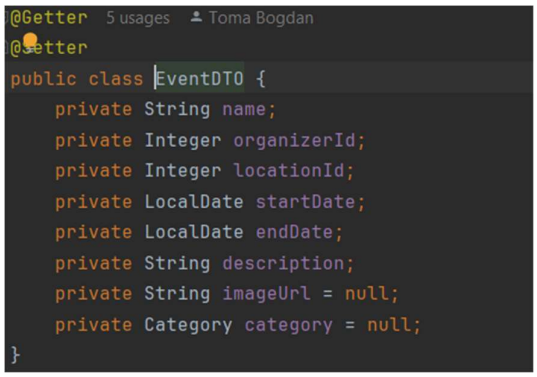
```
@Setter 9 usages  Toma Bogdan *
@Getter
@Entity
@Table(name = "locations")
public class Location {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(nullable = false)
    private String name;

    @Column
    private String street;

    @Column
    private String city;

}
```

Adnotația Table si Entity provin din biblioteca Spring Data JPA [8] si se ocupa de maparea clasei către un tabel din baza de date. În cazul in care, tabelul nu se află deja in baza de date, acesta va fi creat automat. Pentru a conecta doua tabele intre ele, în funcție de caz, folosim adnotațiile ManyToOne, OneToMany sau OneToOne

De asemenea, folosim adnotațiile Getter si Setter din biblioteca Lombook [9] pentru a genera automat Getters și Setters pentru toate câmpurile prezente într-o clasă, reducând numărul de linii din cod, acesta devenind mult mai lizibil.

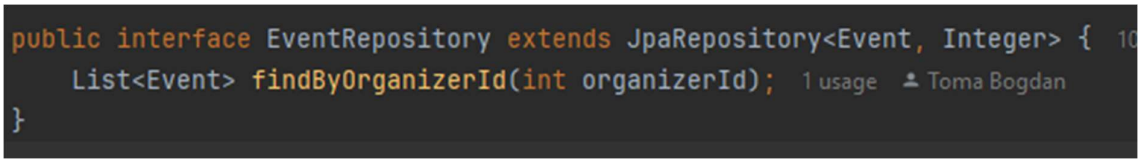
2) Data Transfer Object

Un Data Transfer Object (DTO) este un obiect simplu, fără logică de comportament , utilizat pentru a transporta date între cele doua componente, serverul Spring și aplicația Kotlin.



DTO-urile ajuta la încapsularea datelor, facilitând transferul de date între client și server fără a expune structura internă, sau câmpuri sensibile cum ar fi parolele utilizatorilor. În plus, acestea oferă o bună separare business logic-ului, îmbunătățind modularitatea codului.

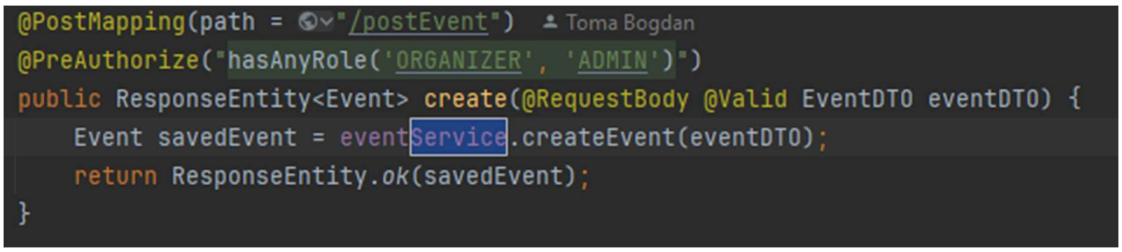
3) Repository



Repository-ul este componenta ce oferă o abstractizare a operațiilor CRUD (create, read, update, delete), fără a trebui scris cod SQL implicit pentru interogarea cu baza de date. Cu ajutorul interfeței JpaRepository, se moștenesc metode precum save, findAll, si alte operații generice, dar se pot defini si metode personalizate, după cum se poate observa in imagine, pentru a căuta toate evenimentele ce au un anumit organizator.

4) Clasele Controller

Aceste componente, adnotate cu RestController, generează endpoint-uri pentru fiecare metoda creata, si gestionează cererile HTTP corespunzător. Sunt extrași parametrii si body-ul din cererea HTTP, apelează serviciile necesare pentru a returna răspunsurile corespunzătoare. Avantajul folosiri unui controller este faptul ca oferă o comunicare ușoara cu interfața utilizatorului si accesul la date, oferind si o modularitate ridicată codului.



De asemenea, folosim adnotarea PreAuthorize, pentru restricționarea accesului pentru diverși utilizatori la anumite resurse. Aceasta restricționare este făcută după rolul utilizatorului, Organizator sau User, deoarece un utilizator obișnuit, nu ar trebui sa aibă

acces la crearea unui eveniment, sau organizatorul sa aibă acces la înregistrarea unui eveniment.

5) Clasele Service

Componentele Service reprezintă clase ce se ocupă cu procesarea necesităților unei cereri HTTP parțial prelucrate de Controller., comunicând direct cu Repository-urile pentru a extrage datele necesare pentru prelucrarea cererii HTTP. Acestea centralizează procesarea cererii propriu-zise, oferind o separare si o modularitate a codului

6) JWT (Json Web Token)

Un Json Web Token este o modalitate sigura de transfera date criptate între doua entități separate. Acest tip de token este folosit pentru înregistrarea si autentificarea unui utilizator in aplicatie.

Atunci, când un utilizator dorește sa se înregistreze, un token JWT este generat in funcție de detaliile acestuia, de un algoritm de hashing si de o cheie secreta, specifica pentru aceasta aplicație. Un astfel de token este valabil 24 de ore din motive de siguranță, si este folosit ca si header de autorizare în majoritatea cererilor HTTP către serverul de Spring, pentru ca aplicația sa fie securizată.

De asemenea, token-ul păstrează detalii despre rolul unui utilizator, restricționând accesul acestuia la anumite funcționalități pe baza rolului deținut.

Pentru a filtra cererile HTTP care conțin un token JWT valid, utilizăm o clasă ce extinde OncePerRequestFilter. Aceasta permite procesarea și validarea cererilor după criteriile stabilite.

```
@Component  Toma Bogdan
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter {
    private final JwtService jwtService;
    private final UserDetailsService userDetailsService;

    /*Validate and parse the jwt token*/
    @Override  Toma Bogdan
    protected void doFilterInternal(
            @NonNull HttpServletRequest request,
            @NonNull HttpServletResponse response,
            @NonNull FilterChain filterChain
    ) throws ServletException, IOException {
        final String authorizationHeader = request.getHeader( s: "Authorization");
        final String jwt;
        final String userEmail;

        if (authorizationHeader == null || !authorizationHeader.startsWith("Bearer ")) {
            filterChain.doFilter(request, response);
            return;
        }
```

7) Spring Security

Pentru ca aplicația sa fie mai securizata folosim următorul fișier de configurare:

```java
@Configuration  Toma Bogdan
@RequiredArgsConstructor
public class ApplicationConfig {
    private final UserRepository userRepository;
    @Bean  Toma Bogdan
    public UserDetailsService userDetailsService() {...}
    @Bean  Toma Bogdan
    public AuthenticationProvider authenticationProvider() {...}

    @Bean  Toma Bogdan
    public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {...}

    @Bean  Toma Bogdan
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
}
```

UserDetailsService este folosit pentru a accesa date specifice utilizatorului. AuthenticationProvider integrează mecanismul de autorizare și securizare din Spring pentru a autentifica un utilizator, asigurând validitatea credențialelor acestuia.

Pe de altă parte, PasswordEncoder utilizează algoritmul BCrypt pentru criptarea parolelor utilizatorilor în baza de date, sporind securitatea credențialelor stocate prin asigurarea că parolele sunt stocate în mod securizat înainte de a fi folosite.

De asemenea, pentru a aplica toate funcționalitățile de mai sus folosim o clasa de configurare cu adnotările de EnableWebSecurity și EnableMethodSecurity din Spring Security [11]. Putem observa mai jos cum sunt integrate caracteristicile menționate:

```java
@Bean  Toma Bogdan
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
            .csrf() CsrfConfigurer<HttpSecurity>
            .disable() HttpSecurity
            .authorizeHttpRequests() AuthorizationManagerRequestMat...
            .requestMatchers("/auth/**", "/api/images/**") AuthorizedUrl
            .permitAll() AuthorizationManagerRequestMat...
            .anyRequest() AuthorizedUrl
            .authenticated() AuthorizationManagerRequestMat...
            .and() HttpSecurity
            .sessionManagement() SessionManagementConfigurer<HttpSecurity>
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and() HttpSecurity
            .authenticationProvider(autheticationProvider)
            .addFilterBefore(jwtAuthFilter, UsernamePasswordAuthenticationFilter.class);

    return http.build();
}
```

5.2 Implementarea aplicației Kotlin

Similar cu discuția din capitolul privind soluția propusă, aplicația mobila este alcătuita din doua mari componente: Screens(UI) și ViewModels. În plus, există și alte componente secundare, cum ar fi cele responsabile pentru navigarea în aplicație, stocarea datelor (DataStore), și alte biblioteci utilizate. Aceste aspecte vor fi detaliate în acest capitol.

5.2.1 Navigarea

Pentru navigarea între ecranele aplicatiei folosim NavController [18]. Controller-ul functioneaza ca o stiva de destinatii. Fiecare ecran are o rută asociata, iar acesta este accesat, controller-ul salveaza aceasta destinatie in stiva sa.

Deoarece ne-am dorit implementarea unui Bottom Bar al aplicației, pentru o interfața ușor de folosit pentru utilizatori, ecranele sunt grupate in funcție de graph-uri,:

- Principalul graph il reprezinta cel din Bottom Bar, ce cuprinde ecranele de Home, Map, Saved si Profile
- Graph-ul de autentificare/înregistrare
- Graph-ul de adaugare/vizualizare eveniment

5.2.2 Preferences DataStore

Preferences DataStore este o modalitate din kotlin, de a stoca date primitive sau format JSON după o anumita cheie. DataStore utilizează corutine și Flow pentru a stoca datele asincron, consistent și tranzacțional.

Event Tracker utilizează DataStore pentru a îmbunătăți eficiența și performanța aplicației în două moduri esențiale.

În primul rând, DataStore este folosit pentru a stoca tokenul JWT (JSON Web Token) după autentificare, facilitând astfel cererile ulterioare către serverul backend prin includerea automată a tokenului în header-ul cererilor HTTP.

În al doilea rând, DataStore stochează informațiile utilizatorului, cum ar fi înregistrările la evenimente și datele contului. Aceasta reduce numărul de apeluri către serverul backend, minimizând overhead-ul și îmbunătățind performanța aplicației. Prin stocarea locala a datelor esențiale, aplicația poate accesa rapid informațiile utilizatorului fără a depinde de o conexiune constantă la internet, asigurând astfel o experiență fluidă și eficientă.

De asemenea, la deconectarea utilizatorului sau la expirarea token-ului JWT, datele stocate sunt golite.

5.2.3 ViewModels

ViewModel-urile, cum este menționat in capitolul 4, separă logica business de logica de prezentare. Acestea comunică asincron cu DataStore prin intermediul unui StateFlow, care funcționează ca un canal între componente.

StateFlow este proiectat pentru a deține si emite stări imutabile, notificând toate colectoarele despre noile valori atunci când starea se schimbă.

Atunci când o valoare trebuie schimbată în DataStore, viewModelul validează si actualizează starea, care va fi apoi transmisa către toate colectoarele active.

```
private val userToken = repository.getToken
    .distinctUntilChanged()
    .stateIn(viewModelScope, SharingStarted.WhileSubscribed( stopTimeoutMillis: 5000), initialValue: "")
var events by mutableStateOf<List<Event>?>( value: null)
init {  Toma Bogdan
    viewModelScope.launch {
        userToken.collect{
            if (!it.isNullOrEmpty()) {
                val response = eventsRepository.getAllEvents(userToken.value!!)
                events = response
            }
        }
    }
}
```

Comunicarea client-server:

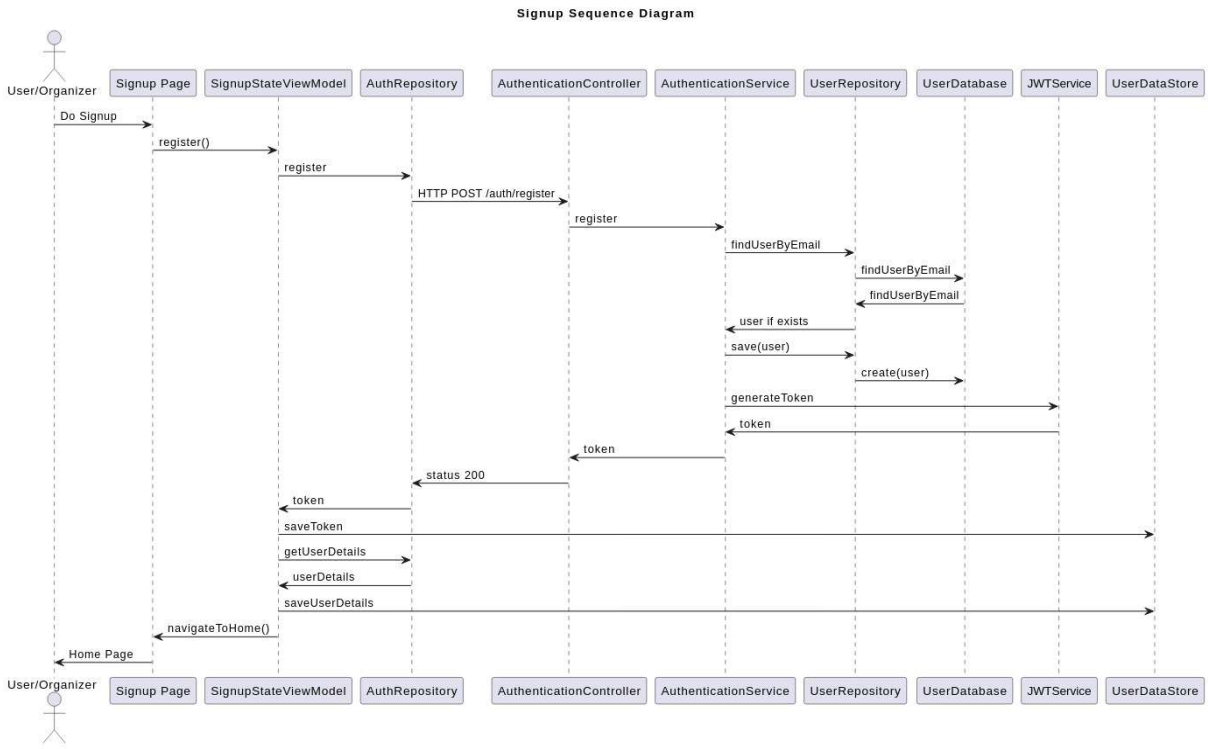
Pentru comunicarea dintre client si server, se folosește framework-ul Ktor, renumit pentru capacitatea sa de a gestiona conexiunile într-un mod asincron si eficient. De asemenea, acesta este integrat cu biblioteca KotlinxSerializer, pentru serializarea/deserializarea JSON-ului, facilitând procesul de transfer între client și server.

Sintaxa ktor este una intuitiva si ușor de urmărit:

```
suspend fun login(authRequest: AuthRequest): AuthResponse {  Toma Bogdan
    val response = client.post( urlString: "${AppConfig.SERVER_URL}/auth/authenticate") {
        contentType(ContentType.Application.Json)
        setBody(authRequest)
    }
    return response.body()
}
```

5.2.4 Pagina de Login/Register

Paginile de login/Register sunt formate din căsuțe de text pentru completarea câmpurilor necesare. În secțiunea de register, utilizatorul are un checkbox, pentru crearea unui cont de organizator. Complexitatea acestor componente vine însă la autentificarea utilizatorului. Putem observa cel mai bine acest flow in diagrama de mai jos:



5.2.5 Paginile de Home și Saved Events

Aceste două pagini au la bază o modalitate de implementare similară, însă diferențele constau în funcționalitățile specifice pe care le oferă. În pagina de home, utilizatorii pot filtra evenimentele în funcție de dată, categorie sau nume. În schimb, în pagina de evenimente salvate (saved events), utilizatorii au posibilitatea de a vizualiza codurile QR pentru fiecare eveniment, iar organizatorii pot edita evenimentele proprii și scana codurile QR.

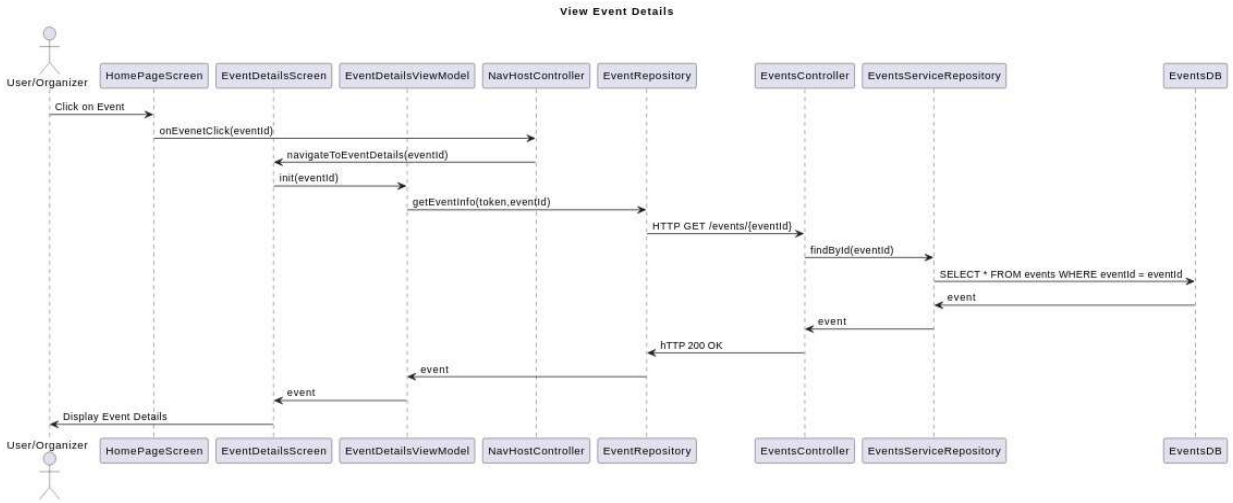
Pentru a simula un feed de evenimente, similar cu cel al unei rețele de socializare, fiecare eveniment este încadrat într-un card ce conține detaliile esențiale ale acestuia, iar pentru a simula scroll-ul infinit, folosim LazyColumn.

LazyColumn încarcă si afișează doar elementele vizibile pe ecran, astfel aplicația funcționează optim si pentru liste foarte mari de evenimente. În plus, pentru îmbunătățirea performantei, imaginile evenimentelor sunt încărcate asincron

```
} else {
    LazyColumn(
        modifier = Modifier
            .fillMaxSize()
            .padding(horizontal = 16.dp),
        contentPadding = PaddingValues(vertical = 16.dp),
        verticalArrangement = Arrangement.spacedBy(16.dp)
    ) {
        items(filteredEvents) { event ->
            EventCard(event = event, onEventClick = onEventClick)
        }
    }
}
```

5.2.6 Pagina de vizualizare eveniment

Pentru a putea vizualiza toate detaliile si recenziile unui eveniment, putem naviga către pagina corespunzătoare prin apăsarea pe cardul respectiv, din pagina de Home sau de Save. Acesta este procesul de inițializare al paginii:



5.2.7 Adăugarea imaginilor

O altă funcționalitate importantă a aplicației este posibilitatea de a încărca imagini atât pentru profilul utilizatorului, cât și pentru evenimente. Imaginile sunt salvate într-un director local, folosind un nume de fișier generat unic pentru a evita coliziunile și problemele de suprascriere. În baza de date, se stochează link-ul către imaginea locală, facilitând accesul rapid și direct la imaginea stocată pe server.

Aplicația va solicita utilizatorului permisiunea de acces la spațiul de stocare al telefonului. După selectarea unei imagini, aplicația va reține această imagine sub forma unui URI (Uniform Resource Identifier). Ulterior, imaginea va fi convertită într-un format binar pentru a putea fi transmisă printr-o cerere HTTP către serverul Spring

```
@PostMapping("/upload")  Toma Bogdan *
public ResponseEntity<String> uploadImage(@RequestParam("file") MultipartFile file) {
    if (file.isEmpty()) {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("File is empty");
    }
    try {
        String filename = UUID.randomUUID().toString() + "_" + file.getOriginalFilename();
        Path path = Paths.get( first: uploadDir + File.separator + filename);
        Files.write(path, file.getBytes());

        String imageUrl = serverUrl + "/api/images/" + filename;
        return ResponseEntity.ok(imageUrl);
    } catch (IOException e) {
        return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Failed to upload image");
    }
}
```

5.3 Codurile QR

Event Tracker nu dispune de un sistem de plata incorporat direct in aplicație încă, astfel scanarea codurilor QR se face doar pentru a oferi detalii si statistici în timp real către organizator.

Atunci când un utilizator se înregistrează la un eveniment, se generează un cod de tichet unic ce este păstrat in baza de date. Funcția care genereaza codurile QR primește codul unic al utilizatorului care va fi codificat, și returnează un obiect Bitmap reprezentând codul QR. Biblioteca ZXing [19] este utilizata pentru a codifica textul într-un BitMatrix, folosind sugestii pentru nivelul de corecție a erorilor. Bitmap-ul rezultat este creat și fiecare pixel este setat în funcție de valorile din BitMatrix, asigurând astfel generarea corectă și eficientă a codurilor QR.

```
fun generateQrCode(text: String, width: Int = 200, height: Int = 200): Bitmap? {  Toma Bogdan
    return try {
        val hints = hashMapOf<EncodeHintType, Any>()
        hints[EncodeHintType.ERROR_CORRECTION] = ErrorCorrectionLevel.H
        val bitMatrix: BitMatrix = MultiFormatWriter().encode(text, BarcodeFormat.QR_CODE, width, height, hints)
        val bitmap = Bitmap.createBitmap(width, height, Bitmap.Config.RGB_565)
        for (x in 0 ≤ until < width) {
            for (y in 0 ≤ until < height) {
                bitmap.setPixel(x, y, if (bitMatrix[x, y]) android.graphics.Color.BLACK else android.graphics.Color.WHITE)
            }
        }
        bitmap
    } catch (e: Exception) {
        e.printStackTrace()
        null
    }
}
```

Pentru a scana codul QR, ca si organizator, se pornește un nou Activity ce îi solicită utilizatorului accesul la camera telefonului, care citește codul unic criptat in codul QR daca acesta este valid, sau nu a fost deja utilizat.

5.4 ArcGIS API

Consider că, un avantaj semnificativ este funcționalitatea care permite utilizatorului să vizualizeze evenimentele și locația lor actuală pe hartă. Distanța până la un eveniment este un factor important pentru multe persoane atunci când decid să participe sau nu la un eveniment. Această funcționalitate îmbunătățește experiența utilizatorului prin furnizarea unei vizualizări geografice ușor de înțeles.

Astfel, am ales să folosesc API-ul arcGIS, creat de Esri (Environmental Systems Research Institute). ArcGIS este o platforma ce permite crearea de harți customizabile atât într-o pagină web, cat si in dezvoltarea unei aplicații, prin crearea unui cont in arcgis developers [20]. API-ul ArcGIS este compatibil cu o gamă largă de limbaje de programare, inclusiv Kotlin, oferind frecvent suport și dezvoltări noi.

Translatarea adreselor:

Pentru a adăuga indicatoare pe harta pentru fiecare eveniment, avem nevoie sa translatăm adresele de forma „Bucuresti, Bd. Unirii Nr. 2” , in latitudine si longitudine. Obținem acest lucru prin utilizarea unui LocatorTask, ce funcționează ca un client ce trimite cereri HTTP către serverul ArcGIS de geo codare, pentru a întoarce coordonatele geografice.

```
val geocodeResults = locatorTask.geocode( searchText: event.location.city + " " + event.location.street).getOrNull()
val firstResult = geocodeResults?.get(0)
```

Crearea harții:

```
MapView(
    modifier = Modifier
        .fillMaxSize()
        .padding(innerPadding),
    arcGISMap = map,
    graphicsOverlays = listOf(graphicsOverlay),
    mapViewProxy = viewModel.mapViewProxy,
    onSingleTapConfirmed = {
        coroutineScope.launch {
            viewModel.identify(it)
        }
    },
    onPan = { viewModel.dismissBottomSheet() },
locationDisplay = locationDisplay
)
```

Inițializarea hărții este realizată prin intermediul funcției createMap(), care returnează un obiect ArcGISMap cu un stil de bază topografic și un punct de vedere inițial centrat pe coordonatele Bucureștiului. Harta este configurata cu un Overlay ce conține cate un punct cu un simbol specific, pentru fiecare locație prezenta in evenimente (nu exista duplicate).

ViewModel-ul harții gestionează, de asemenea, starea aplicației, cum ar fi afișarea unui BottomSheet pentru detalii suplimentare despre evenimente, atunci când utilizatorul apasă pe un simbol al unei locații.

6 STUDIU DE CAZ / EVALUAREA REZULTATELOR

Aplicația Event Tracker se încadrează în categoria de aplicație pentru gestiunea vânzării biletelor.

Ca și funcționalități de bază aplicația are implementate următoarele lucruri:

1) Căutarea Evenimentelor

Aplicațita afișează utilizatorilor evenimentele active. Utilizatorii pot de asemenea să aleagă mai multe filtre după care să caute evenimente precum locație, tip. Aceste funcționalități sunt prezenta și în alte platforme existente în piață și reprezintă un standard pentru acest tip de platforme.

2) Achiziționarea biletelor

Aplicația oferă evenimente cu intrare gratuită sau cu bilet contra cost. În această lucrare a fost implementat procesul de obținere a unui bilet pentru evenimentele de interes dar și vizualizarea lor în cont. Ce nu este implementat este procesatorul de plăți. Ca soluție pentru o implementare viitoare, Stripe [21] este o soluție viabilă pentru a implementa un procesator de plăți, fiind cel mai utilizat procesator dar și pentru suportul pe care îl oferă în etapa de dezvoltare.

3) Review și Comentarii

Aplicația permite utilizatorilor să lase review-uri și comentarii la evenimentele pe care le selectează. Acestea sunt afișate pentru vizitatori, oferindu-le un feedback despre evenimentul la care doresc să participe. Atât utilizatorii cât și organizatorii de evenimente pot lăsa comentarii, aceasta secțiune fiind implementată cu scopul de a facilita comunicarea între cele 2 tipuri de utilizatori a-i aplicației.

4) Gestiunea de evenimente și locații.

Aceasta este o funcționalitate de bază pentru acest tip de platformă. Aplicația prezentată oferă posibilitatea organizatorilor de evenimente să adauge în platformă evenimente dar și să aloce locații. Dacă locația nu a mai fost folosita pana acum de vreun organizator de evenimente, acesta are posibilitatea de a adăuga o locație noua.

5) Căutare bazată pe hartă

Aceasta este o funcționalitate nouă pe care aplicația din lucrare propusă dorește să o ofere. În prezent, marile platforme de gestiune a achizițiilor pentru evenimente nu oferă posibilitatea de a căuta evenimente pe bază de hartă. Acest tip de funcționalitate însă este des întâlnit în aplicații destinate anunțurilor imobiliare precum Imobiliare.ro (referință aici). În acest domeniu a adus un plus valoare. Pe baza feedback-ului oferit de la utilizatori, se poate observa că această funcționalitate a fost percepută bine.

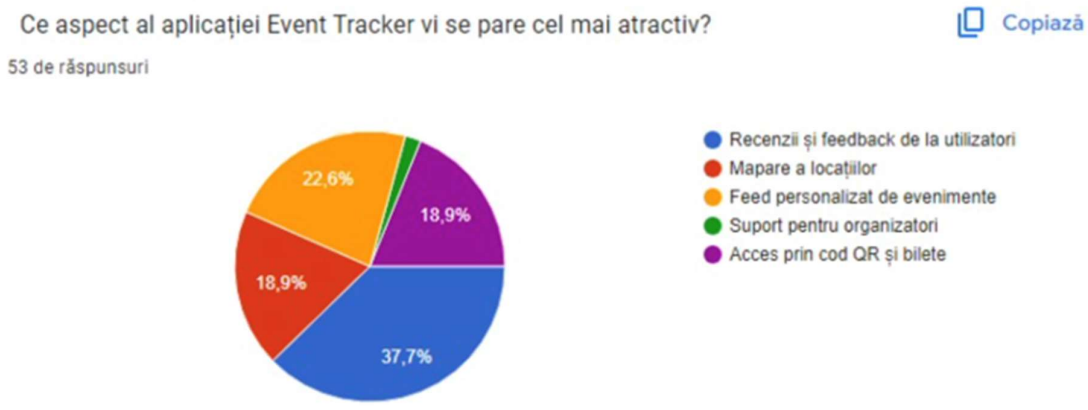


Fig 1. Funcționalități apreciate de utilizatori

6.1 Feedback-ul utilizatorilor

Pentru a evalua mai bine aplicația propusă, au fost colectate date de la mai mulți utilizatori prin intermediul unui chestionar de pe platforma Google Forms. În baza feedback-ului următoarele rezultate au fost obținute.

Pondere utiliizatori:

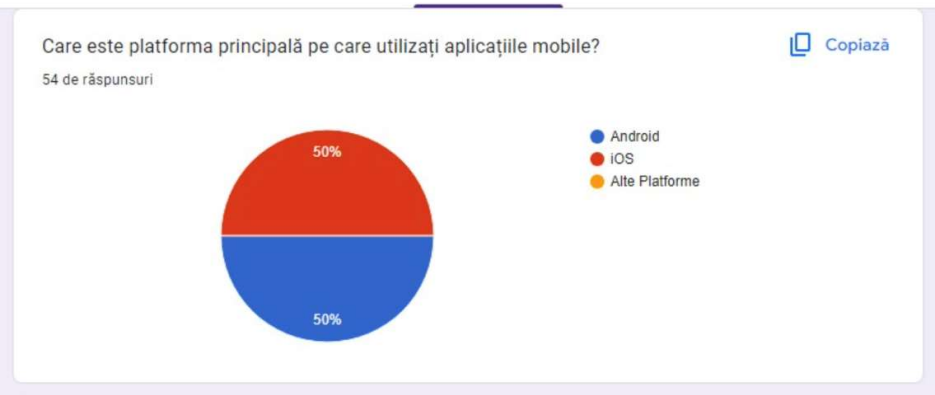


Fig 2. Pondere utilizatori

Acest chestionar a fost oferit unui număr de 54 de persoane. Dintre acestea, jumătate sunt utilizatori de Android, platforma pentru care este dezvoltată aplicați

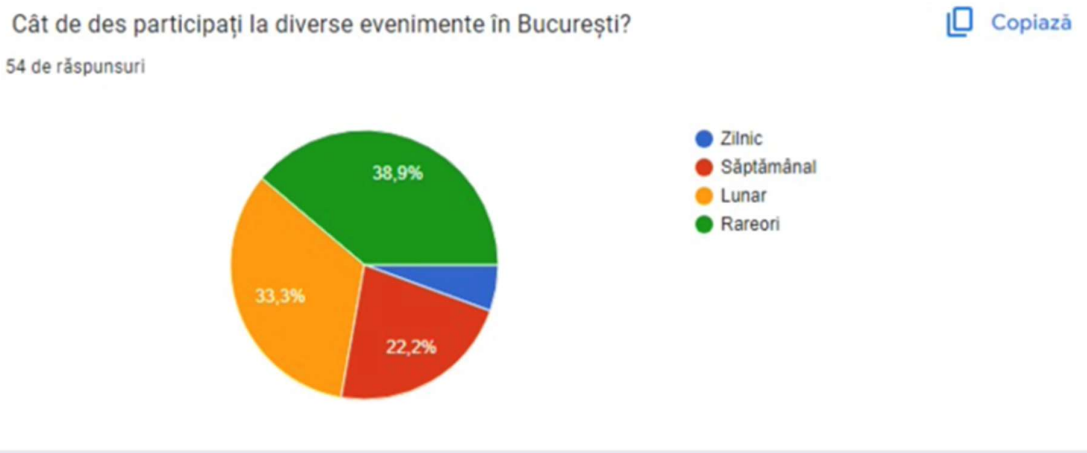


Fig.3 Interes pentru evenimente

Analizând gradul de participare a-l utilizatorilor la evenimente putem deduce că prioritar, sunt funcționalități care să faciliteze căutarea evenimentelor, în detrimentul unor funcționalități care să notifice utilizatorul când apar anumite evenimente. Acest feedback a fost important pentru a stabili funcționalitățile aplicației, fiind implementat un sistem de filtrare dar fiind omis unul de notificări.

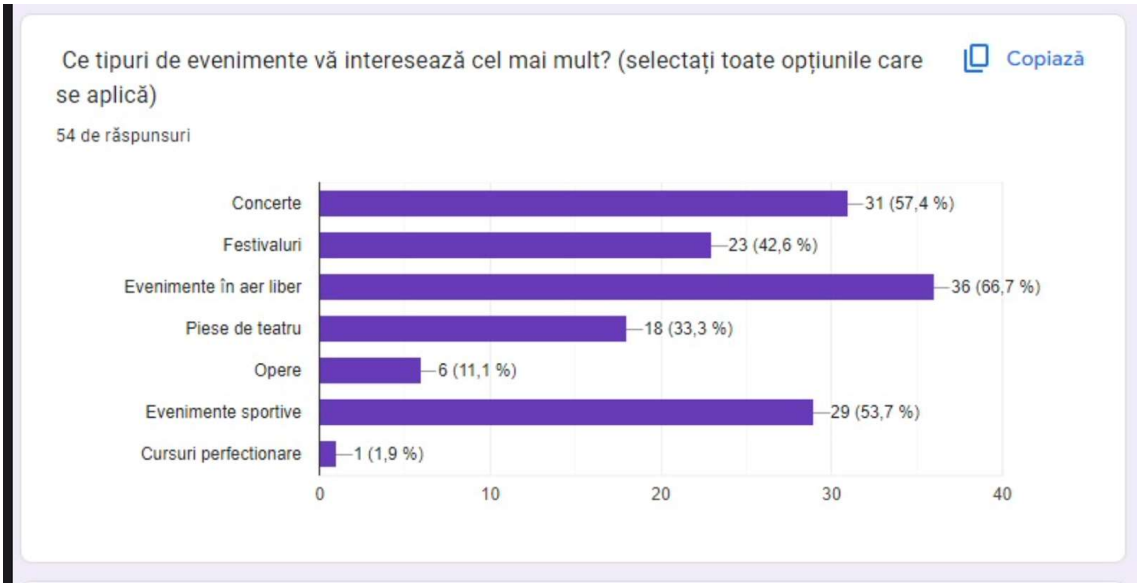


Fig 4. Tipuri de evenimente relevante

Fiecare nișă de eveniment are particularitățile ei. După tipul de eveniment de interes am decis ce să fie inclus în aplicație pentru căutarea eficientă a evenimentelor. Majoritatea evenimentelor de interes sunt evenimente fizice, motiv pentru care a fost implementată și o funcționalitate de găsire a evenimentelor pe o hartă.

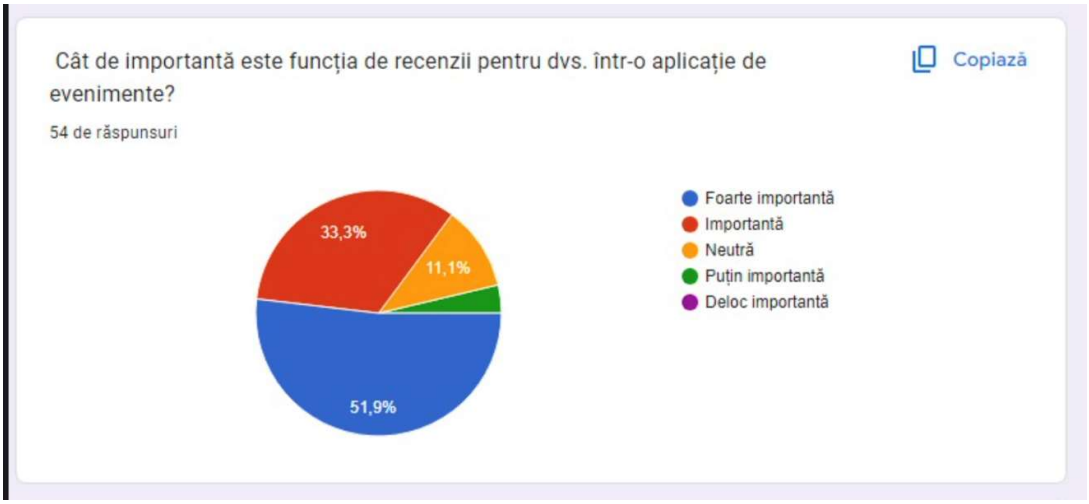


Fig 5. Importanță recenzii

În urma studiului de caz recenziile au fost alese ca cea mai importantă funcționalitate pe o astfel de platformă. În urma acestui rezultat, 2 funcționalități au fost adăugate pentru a facilita interesul utilizatorilor, rating și comentarii. Aceste 2 funcționalități vin

complementare, pentru orice evaluare este nevoie și de feedback, pe baza acestor informații utilizatorii pot face alegerile potrivite.

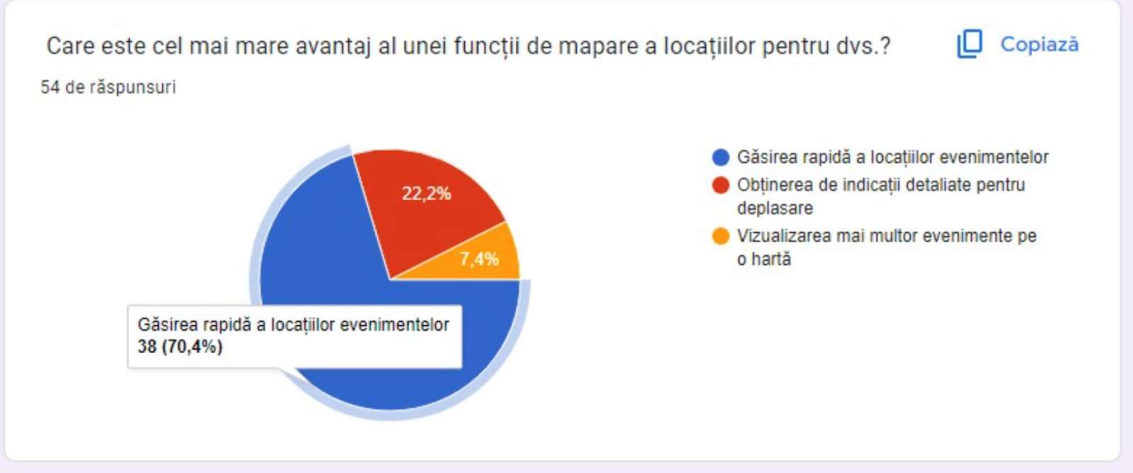


Fig 6. Avantaj funcționalitate mapare

În urma feedback-ului primit de utilizatori, prin introducerea unui funcționalități noi față de ce este în piață, utilizatorii pot găsi evenimentele mai rapid.

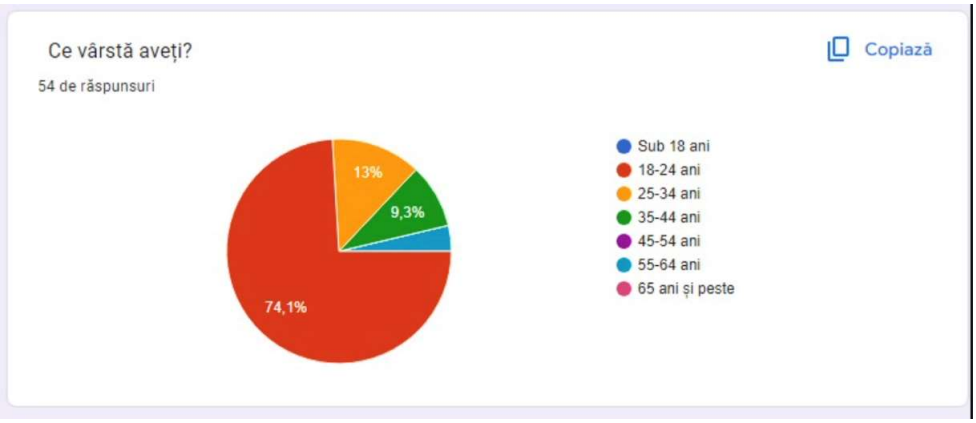


Fig 8. Statistici demografice

Așa cum e de așteptat, majoritatea consumatorilor de evenimente o reprezintă tineretul, în special studenți. Pentru a dezvolta o platformă care să aibă impact în piață au fost luate în considerare nevoile acestor categorii de vârstă.

6.2 Evaluarea performanței aplicației

Pentru a evalua performanța aplicației Event Tracker, am utilizat Android Profiler în Android Studio. Din capturile de ecran furnizate, observăm următoarele aspecte:

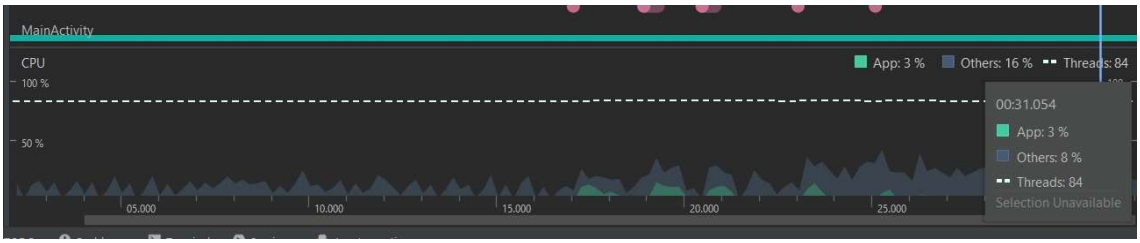


Fig 9. Utilizare CPU

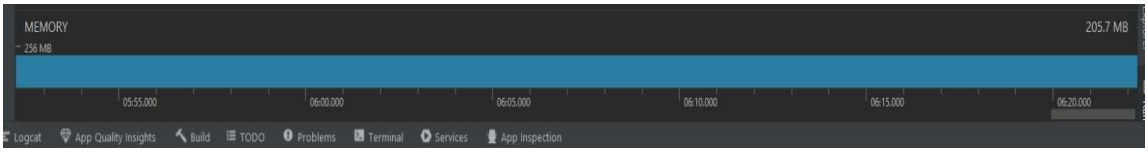


Fig 10. Utilizare Memorie

1) **Utilizarea CPU**: Aplicația utilizează între 0-5% din CPU, ceea ce indică o gestionare eficientă a resurselor și o performanță stabilă în timpul execuției activităților principale.
2) **Memoria**: Aplicația consumă aproximativ 205.7 MB de memorie, ceea ce este rezonabil pentru o aplicație cu funcționalități precum gestionarea evenimentelor și afișarea hărților.
3) **Thread-uri**: Aplicația utilizează 84 de thread-uri, care includ operațiuni de fundal și procese de rețea, asigurând astfel o experiență fluidă pentru utilizatori.

Măsurătorile au fost luate pe un telefon Samsung SM-G975F și indică faptul că aplicația funcționează eficient, cu un consum minim de resurse.

7 CONCLUZII

Proiectul a avut ca și obiectiv principal dezvoltarea unei aplicații pentru vizualizarea si promovarea evenimentelor. Consider că, aplicația și-a atins obiectivele in mare măsura, aceasta oferind o platformă intuitiva si eficientă atât pentru utilizatori, cat si pentru organizatori.

Aplicația apropie cat mai mult nevoile utilizatorilor, care pot lăsa recenzii si comentarii, de așteptările organizatorilor, care pot vizualiza statistici in timp real, si comunica cu utilizatorii, prin intermediul comentariilor. Acest proiect constituie un punct de plecare pentru o soluție viabilă, cu potențial de a avea un impact semnificativ asupra interacțiunilor sociale în mediul urban. Este un exemplu de cum tehnologia poate facilita conexiunile și îmbunătăți experiențele utilizatorilor și organizatorilor de evenimente.

7.1 Dezvoltări ulterioare

Aplicația are toate funcționalitățile de bază, dar există multiple oportunități pentru îmbunătățire și extindere pentru a deveni într-adevăr revoluționară, cum ar fi:

- Cross-platform: extinderea aplicației pe alte platforme, cum ar fi iOS, ceea ce ar fi posibil fara mult efort datorita Kotlin Multiplatform, sau web, prin implementarea unui frontend in React sau alte tehnologii.
- Posibilitatea de a cumpăra bilete la evenimente direct din aplicație, folosind un API pentru plăți precum Stripe.
- Adăugarea de noi funcționalități, cum ar fi, notificările, personalizarea mult mai avansată a feed-ului de utilizator, integrarea cu platforme de socializare și posibilitatea de a distribui evenimentele.

8 BIBLIOGRAFIE

[1] "Bilete La Concerte, Spectacole, Teatru, Sport Și Standup." IaBilet.ro, contact@iabilet.ro. Accessed 23 June 2024.

[2] "Servicii." Www.iabilet.ro, www.iabilet.ro/servicii.

[3] "LiveTickets - Vinde Bilete Simplu Si Usor." Www.livetickets.ro, www.livetickets.ro/vinde-bilete.

[4] "Experiențe Pop Culture • Zile Și Nopți." Zilesinopti.ro, 16 Dec. 2021, zilesinopti.ro.

[5] Evenimente Bucuresti - Timp Liber de Calitate - TimeOutBucuresti.ro. timeoutbucuresti.ro/. Accessed 24 June 2024.

[6] "Events near Me | Facebook." Facebook.com, 2019, www.facebook.com/events.

[7] Spring.io, 2021, start.spring.io.

[8] ("Spring Data JPA") https://spring.io/projects/spring-data-jpa

[9] baeldung. "Introduction to Project Lombok | Baeldung." *Www.baeldung.com*, 12 June 2016, www.baeldung.com/intro-to-project-lombok.

[10] "Spring Projects." Spring.io, 2017, spring.io/projects/spring-framework.

[11] "Spring Projects." *Spring.io*, 2019, spring.io/projects/spring-security.

[12] Oracle. "Java Documentation." Oracle Help Center, docs.oracle.com/en/java.

[13] “Java Fundamentals.” Subscription.packtpub.com, subscription.packtpub.com/book/programming/9781789801736/1/ch01lvl1sec03/the-java-ecosystem.

[14] “Spring Framework Documentation :: Spring Framework.” Docs.spring.io, docs.spring.io/spring-framework/reference/.

[15] “Kotlin Multiplatform | Kotlin.” Kotlin Help, kotlinlang.org/docs/multiplatform.html#kotlin-multiplatform-use-cases.

[16] Joseph James (JJ. “Mastering Jetpack Compose State Management: A Deep Dive into Modern UI Data Flow.” Medium, Medium, 13 Aug. 2023, iamjosephmj.medium.com/mastering-jetpack-compose-state-management-a-deep-dive-into-modern-ui-data-flow-8392e298e56. Accessed 24 June 2024.

[17] “First Steps into Kotlin Multiplatform.” Karumi Blog, 10 June 2019, blog.karumi.com/first-steps-into-kotlin-multiplatform/. Accessed 25 June 2024.

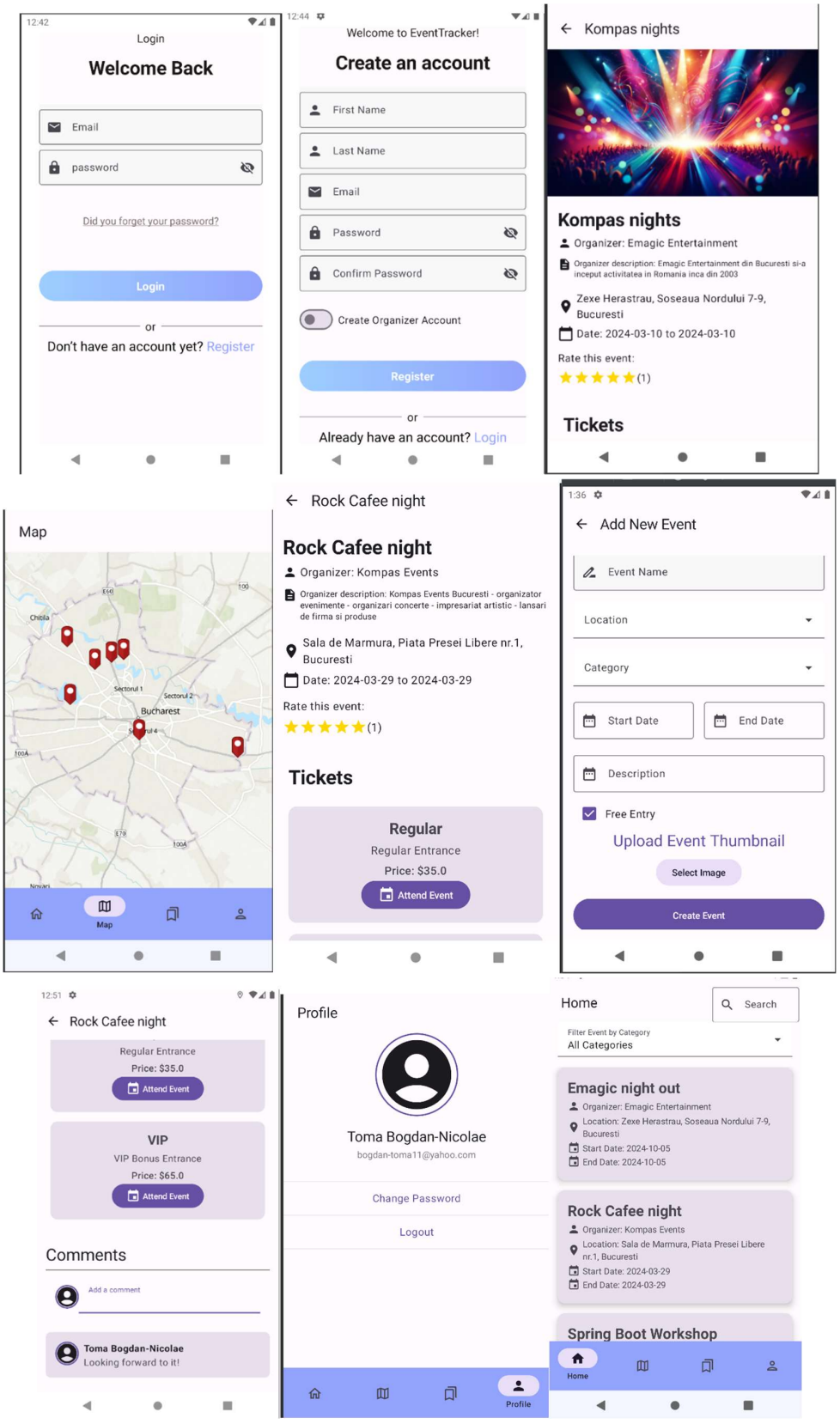
[18] “Navigation with Compose | Jetpack Compose.” Android Developers, developer.android.com/develop/ui/compose/navigation. Accessed 25 June 2024.

[19] “Journeyapps/Zxing-Android-Embedded.” GitHub, 26 Mar. 2021, github.com/journeyapps/zxing-android-embedded.

[20] “ArcGIS for Developers.” ArcGIS for Developers, 2020, developers.arcgis.com.

[21] “Developer Tools.” Stripe.com, 2024, docs.stripe.com/development. Accessed 25 June 2024.

9 ANEXE


Login
Welcome Back
Email
password
Did you forget your password?
Login
or
Don't have an account yet? Register
Welcome to EventTracker!
Create an account
First Name
Last Name
Email
Password
Confirm Password
Create Organizer Account
Register
or
Already have an account? Login
Kompas nights
Kompas nights
Organizer: Emagic Entertainment
Organizer description: Emagic Entertainment din Bucuresti si-a inceput activitatea in Romania inca din 2003
Zexe Herastrau, Soseaua Nordului 7-9, Bucuresti
Date: 2024-03-10 to 2024-03-10
Rate this event:
(1)
Tickets
Map
Bucharest
Map
Rock Cafee night
Rock Cafee night
Organizer: Kompas Events
Organizer description: Kompas Events Bucuresti - organizator evenimente - organizari concerte - impresariat artistic - lansari de firma si produse
Sala de Marmura, Piata Presei Libere nr.1, Bucuresti
Date: 2024-03-29 to 2024-03-29
Rate this event:
(1)
Tickets
Regular
Regular Entrance
Price: $35.0
Attend Event
Add New Event
Event Name
Location
Category
Start Date
End Date
Description
Free Entry
Upload Event Thumbnail
Select Image
Create Event
Rock Cafee night
Regular Entrance
Price: $35.0
Attend Event
VIP
VIP Bonus Entrance
Price: $65.0
Attend Event
Comments
Add a comment
Toma Bogdan-Nicolae
Looking forward to it!
Profile
Toma Bogdan-Nicolae
bogdan-toma11@yahoo.com
Change Password
Logout
Profile
Home
Search
Filter Event by Category
All Categories
Emagic night out
Organizer: Emagic Entertainment
Location: Zexe Herastrau, Soseaua Nordului 7-9, Bucuresti
Start Date: 2024-10-05
End Date: 2024-10-05
Rock Cafee night
Organizer: Kompas Events
Location: Sala de Marmura, Piata Presei Libere nr.1, Bucuresti
Start Date: 2024-03-29
End Date: 2024-03-29
Spring Boot Workshop
Home