PEDOMETER APP

Echipă :

- Cătălin Butacu (Team Leader)
- Denisa Olaru
- Mihai Alexandru
- Cosmin Chiruța

Platforme și softwares utilizate :

- Android Studio (IDE)
- Git&Github (Versionare)
- Sourcetree (Interfață versionare)
- Jira Software (Organizare task-uri)
- Discord & Messenger :^)

Rolurile avute :

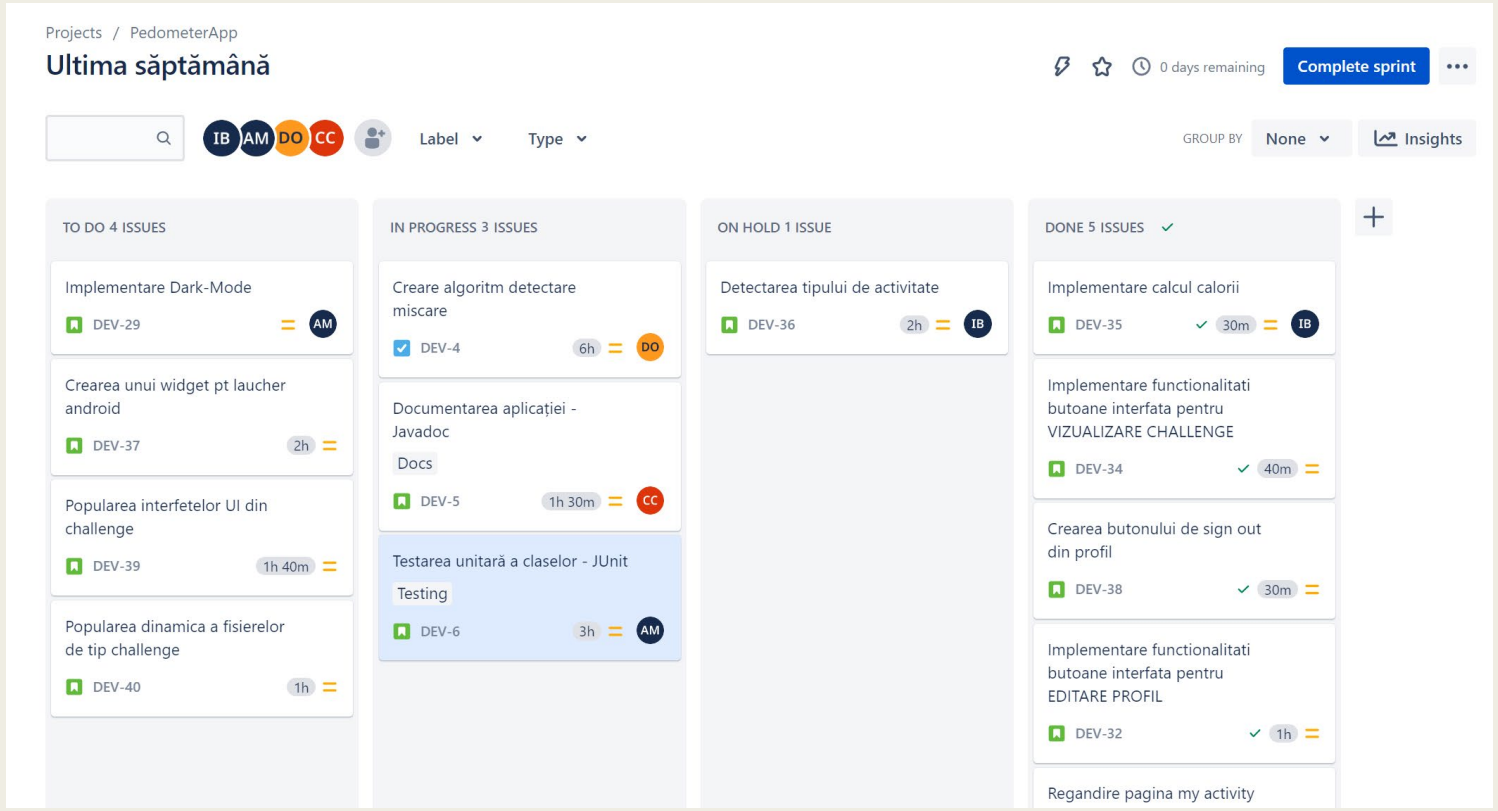
- Cătălin – Team Leader
- Denisa – Designer & Frond-end Developer
- Mihai – Testing & Development
- Cosmin – Research & Development

Scopurile aplicației :

- Realizarea unui GUI atractiv pentru android
- Dezvoltarea unui algoritm de măsurare a pașilor
- Calcularea a diverse statistici caracteristice
- Conectarea între users prin competiții de tip Challenge

Asignarea taskurilor in echipa

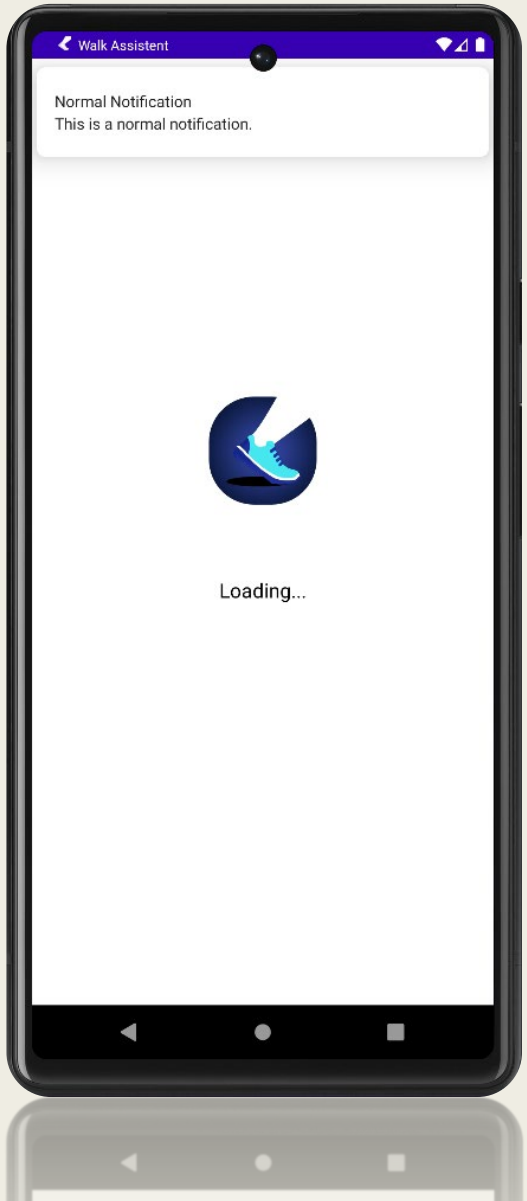
- In procesul de dezvoltare, am folosit Jira ca un mod de asignare a taskurilor si estimari, precum si o verificare periodica a progresului.
- Taskurile au fost estimate dupa cantitate necesare de timp care ar fi necesara rezolvarii taskului respectiv.
- Asignarea se facea dupa preferinta membrilor echipei, fiecare isi alegea ce taskuri tot si se pastra un update constant a ceea ce s-a lucrat.
- Monitorizarea se realiza prin urmarirea procesului taskului respectiv. Daca cineva se bloca la un task muta taskul pe On hold si echipa era informata

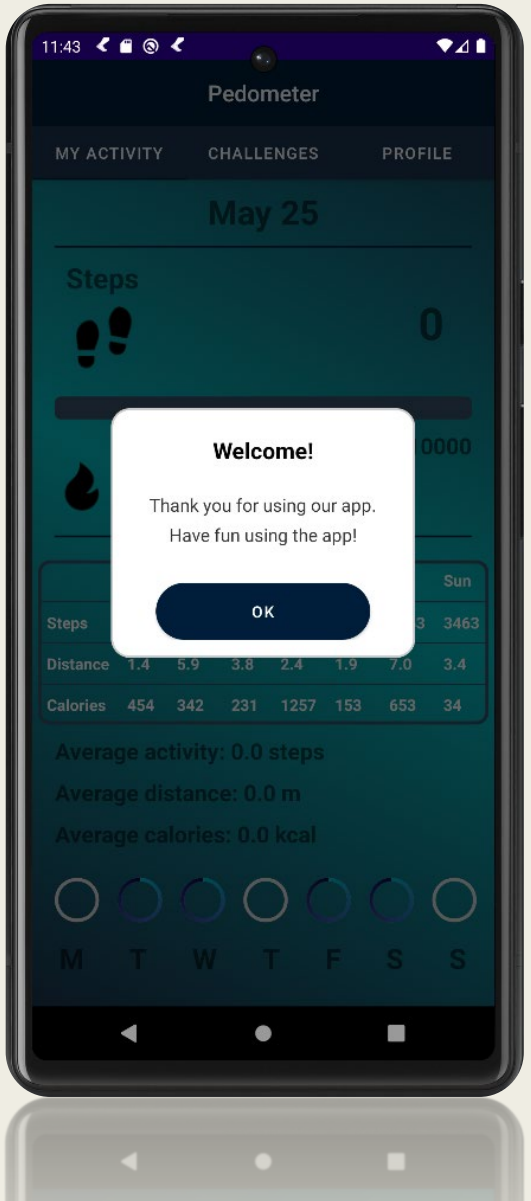


Demo cu noi:

App Demo :

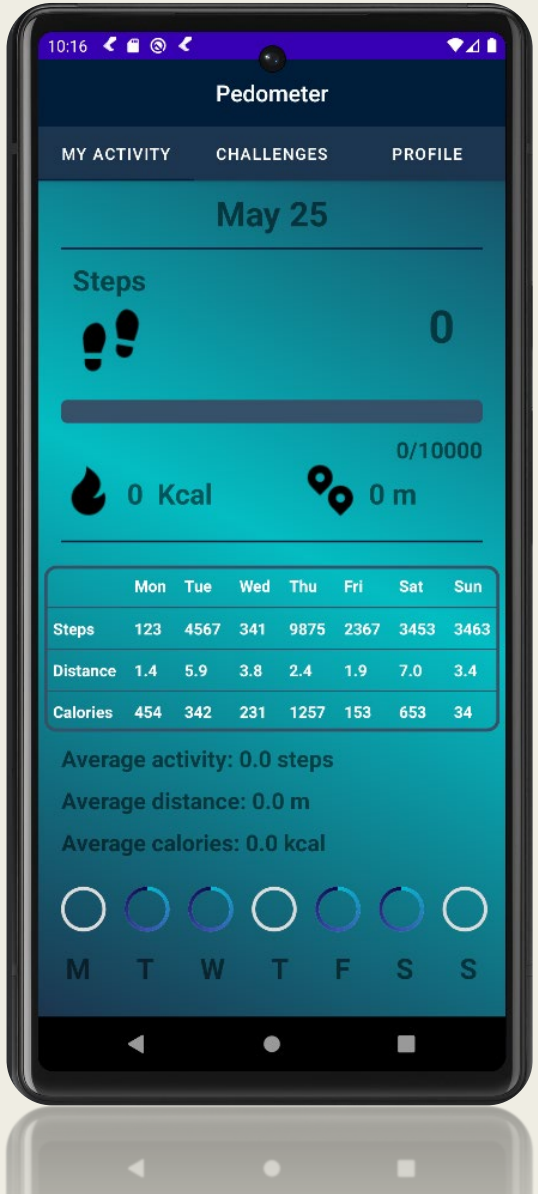
- In ceea ce priveste aplicatia noastra de tip pedometru, aceasta foloseste senzorii unui dispozitiv Android pentru a prelua date privind pozitia, acceleratia, atractia gravitationala pentru a detecta miscarile facute de utilizator si a prelucra acestea pentru a oferi informatii precum numarul de pasi, distanta parcursa si calorii arse.
- Aplicatiei are implementate un system de notificari prin care sa tina utilizatorii informati despre activitatea acestora.





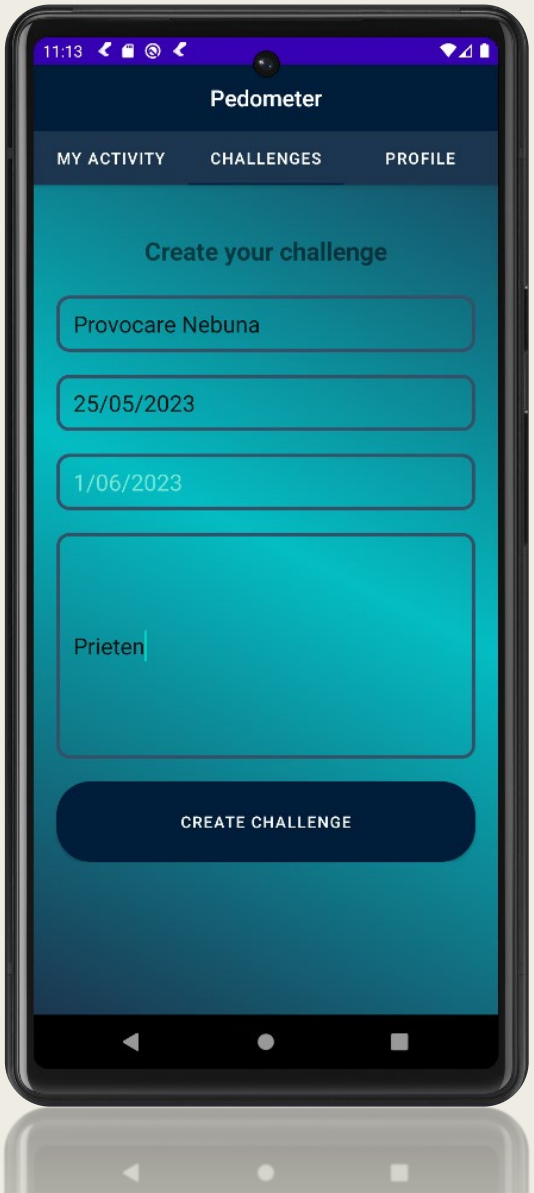
App Demo :

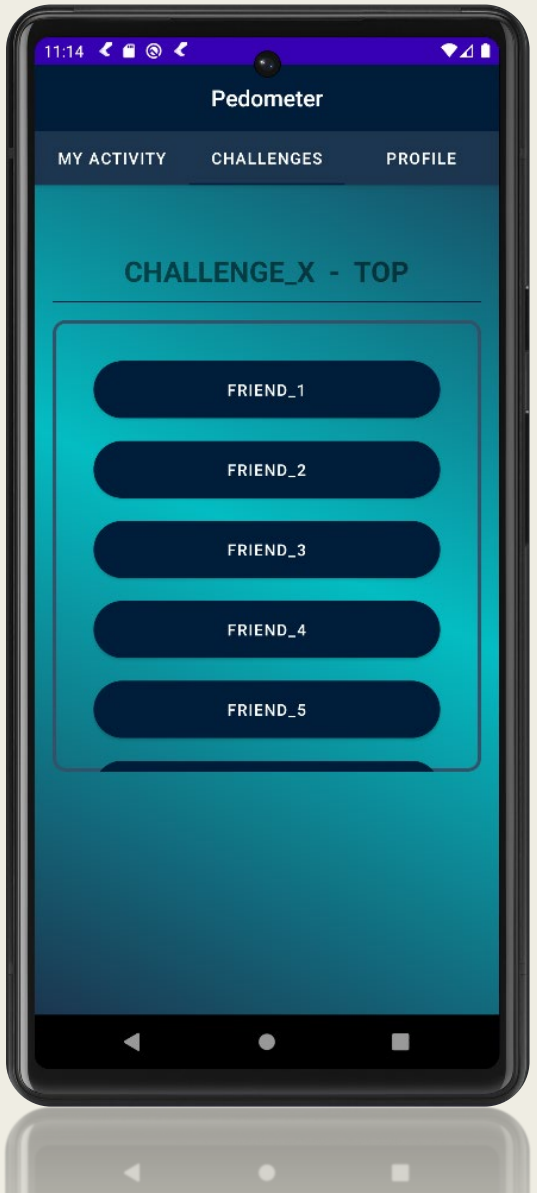
- Aplicatia beneficiaza de 3 zone principale Activity, Challenges si Profile, fiecare avand multe fragmente definite.
- Pagina de Activity contine in general informatii despre activitatea utilizatorului, precum numarul de pasi, caloriile arse si distanta parcursa de utilizator in ziua respectiva, pentru zona de sus a ferestrei, iar in partea de jos fiind istoricul activitatilor din ultima saptamana stocata local pe dispozitiv.
- Pagina de Challenge contine provocarile vizibile tuturor utilizatorilor, prin care acestia se pot intrece intre ei si a-si urmari reciproc activitatea prin intermediul acesteia



App Demo :

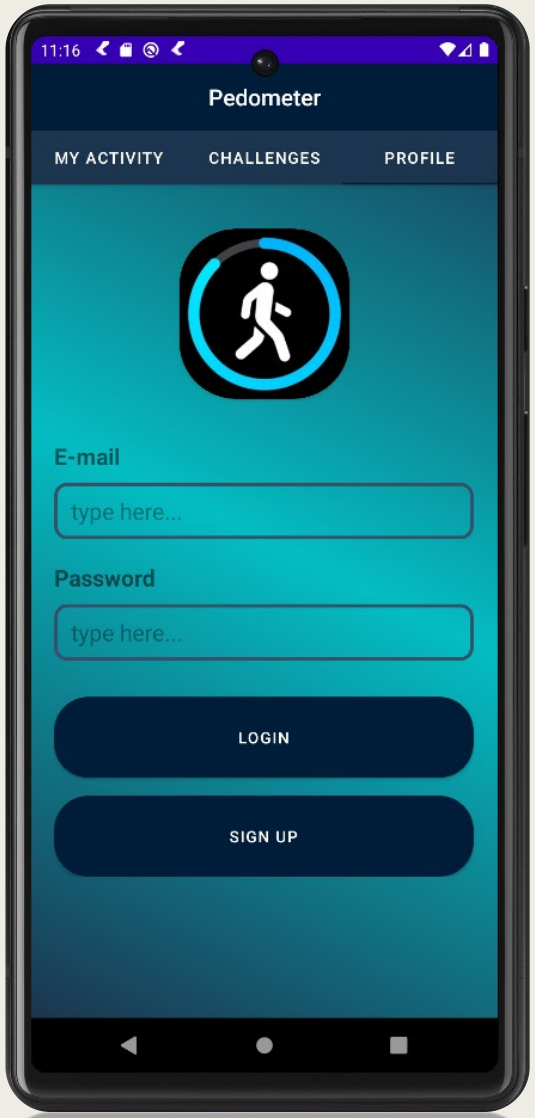
- Prin intermediul paginii de Challenges orice utilizator poate crea challenge-uri pentru prietenii sai, avand posibilitratea de ai invita si pe acestia.
- Lista de prieteni este actualizata la fiecare 30 de secunde de la baza de date. Monitorizarea activitatii acestora fiind deci aproape instantanee

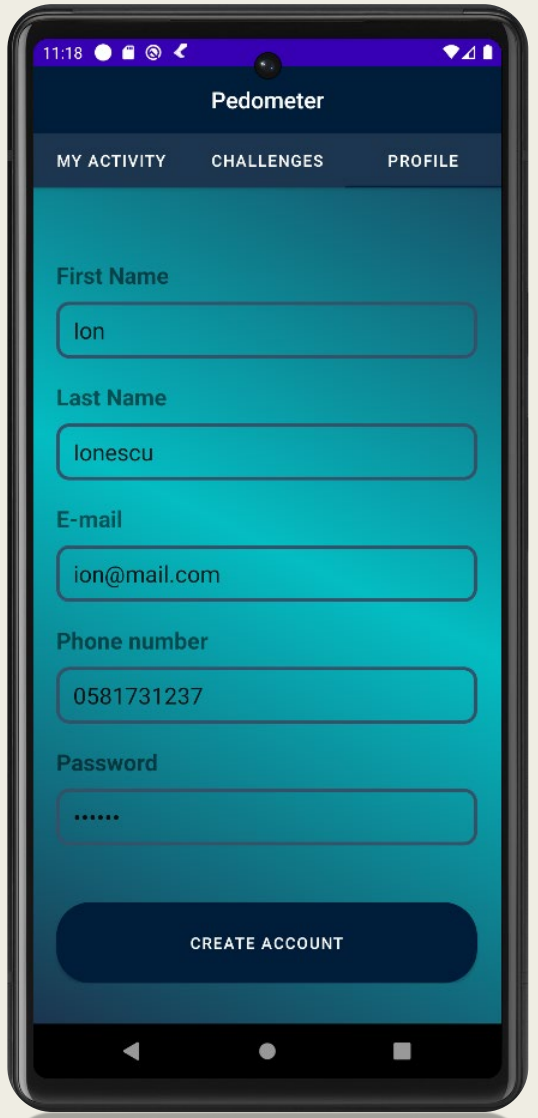




App Demo :

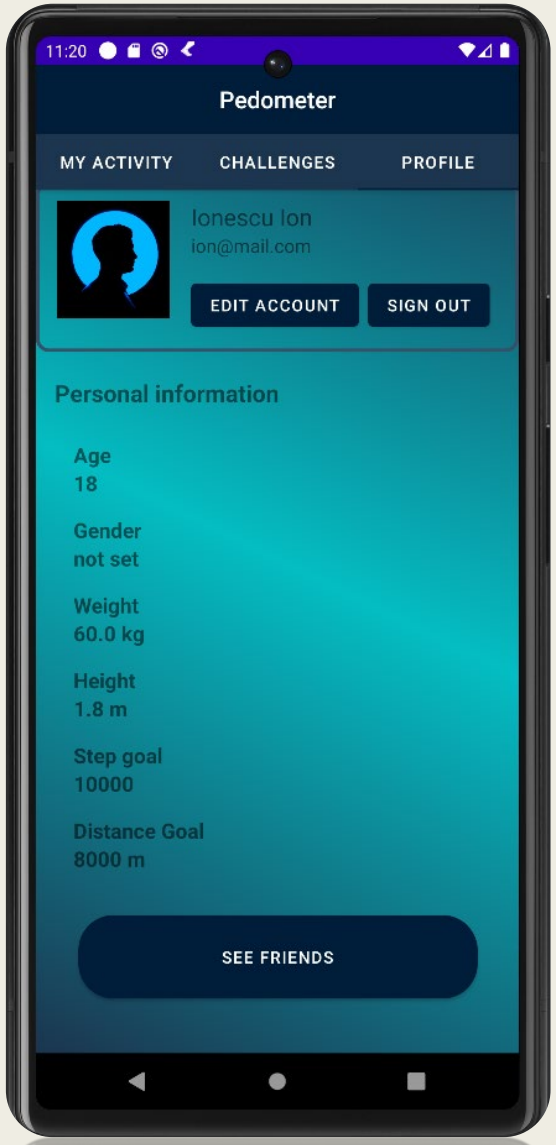
- Pentru a putea participa la astfel de activitate partajata si cu altii, e necesara crearea unui cont, acesta se poate face prin pagina de Profile, daca utilizatorul este intrat pentru prima data in aplicatie sau se poate loga la un cont deja existent.

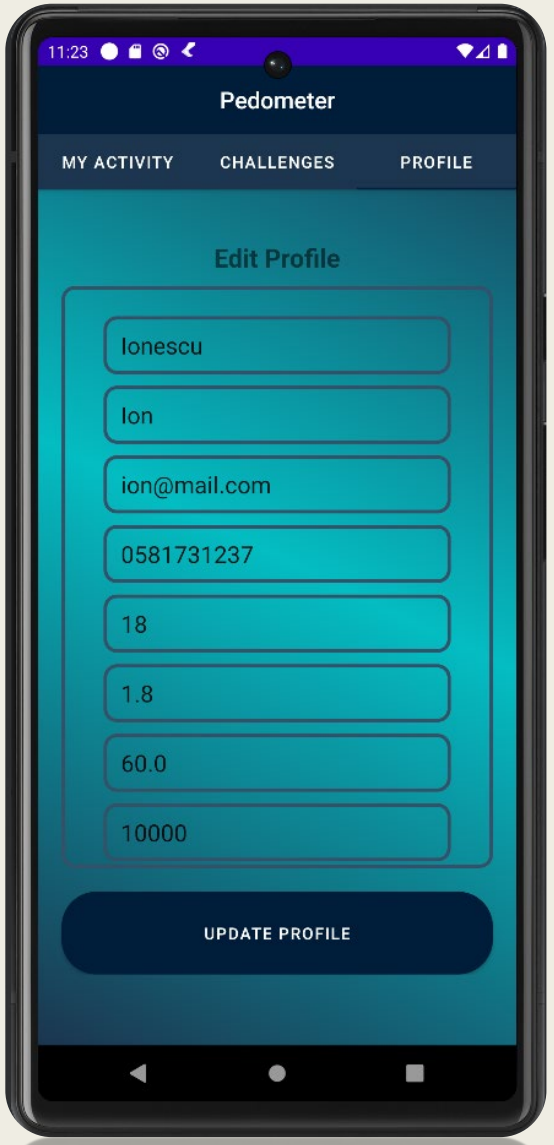




App Demo :

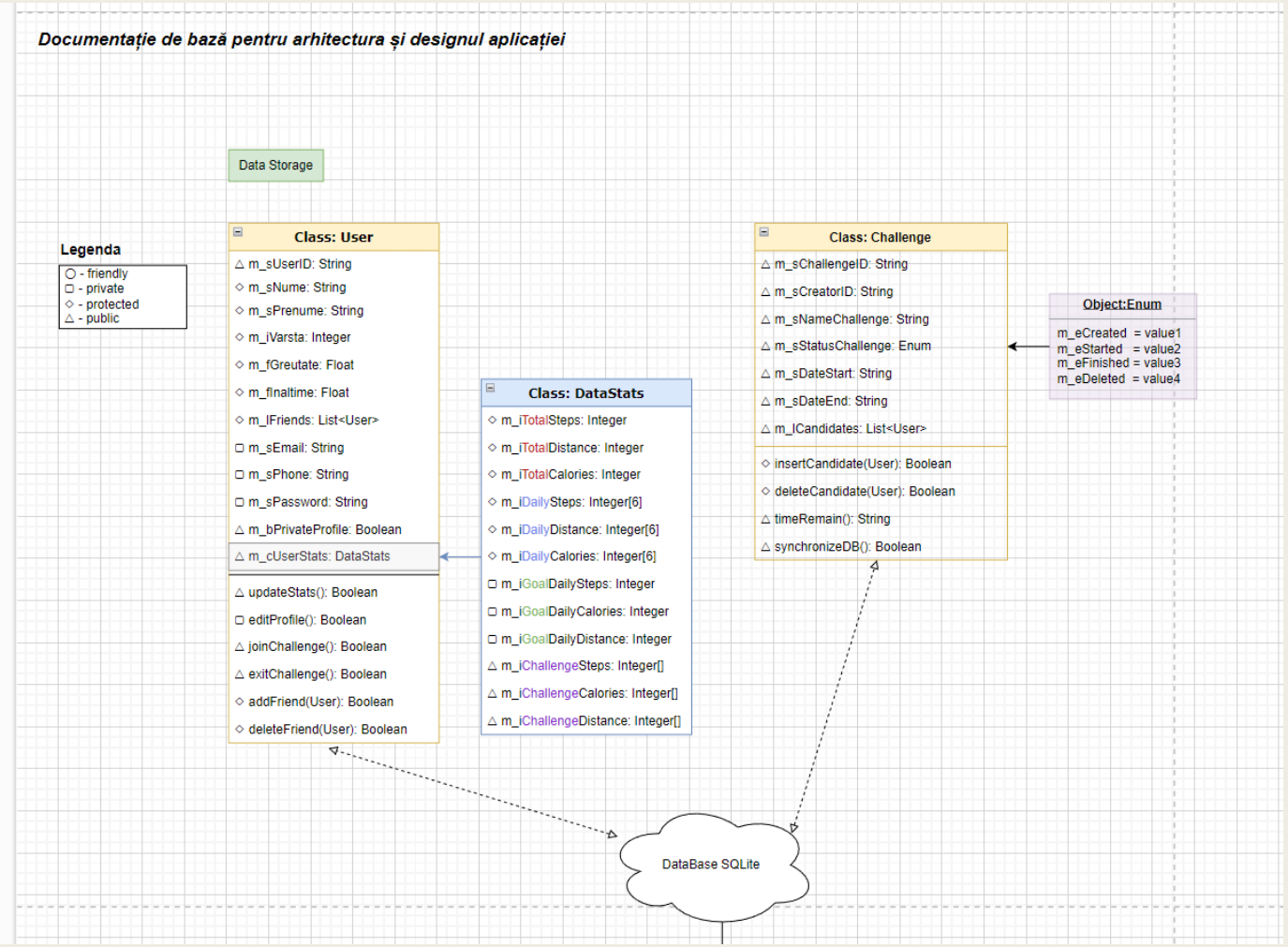
- Profilul odata creat el poate fi reactualizat de cate ori este nevoie si se va sincroniza automat si cu baza de date locala.





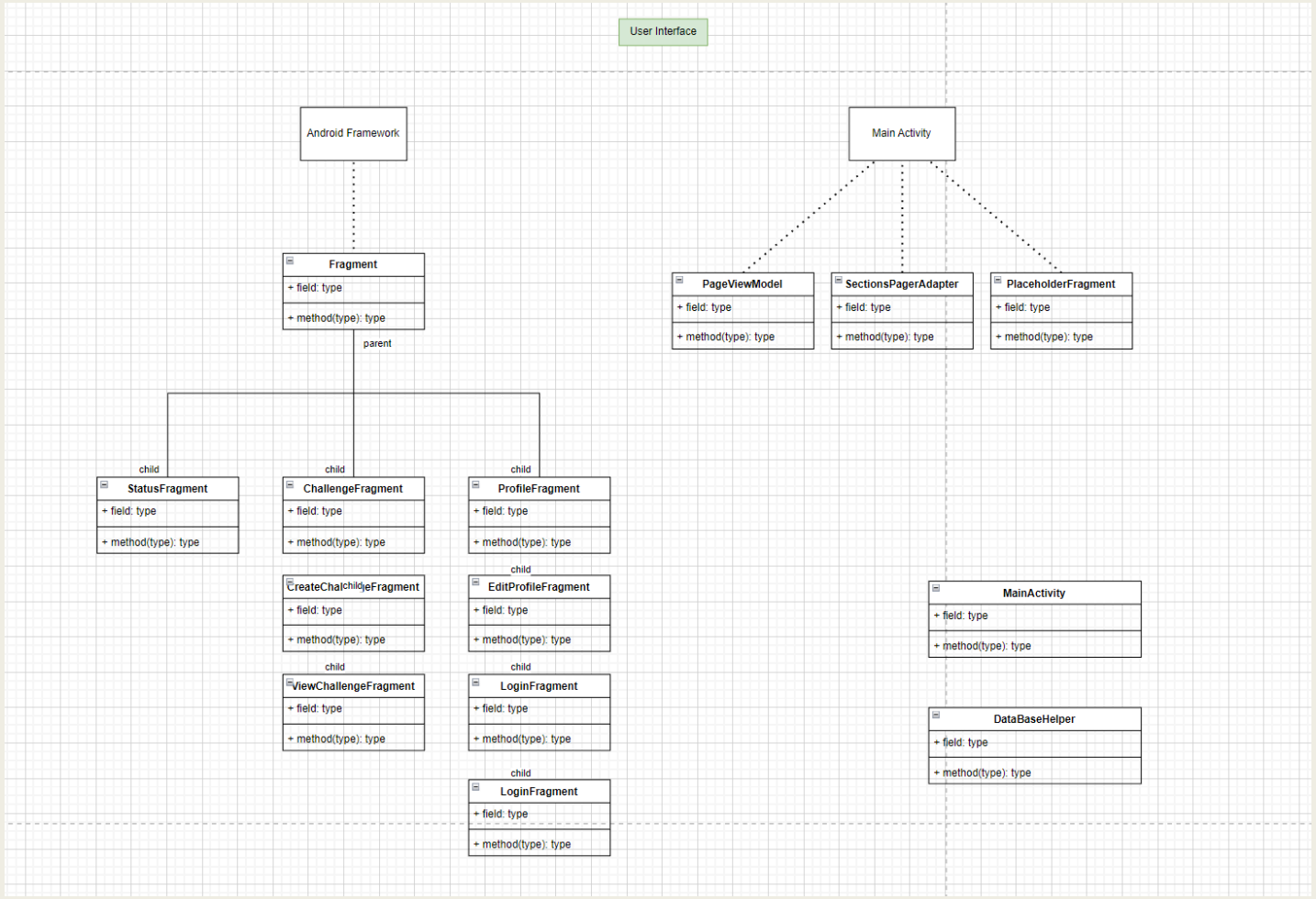
Structura și principalele module:

Modelul de structura locala de memorare a datelor:



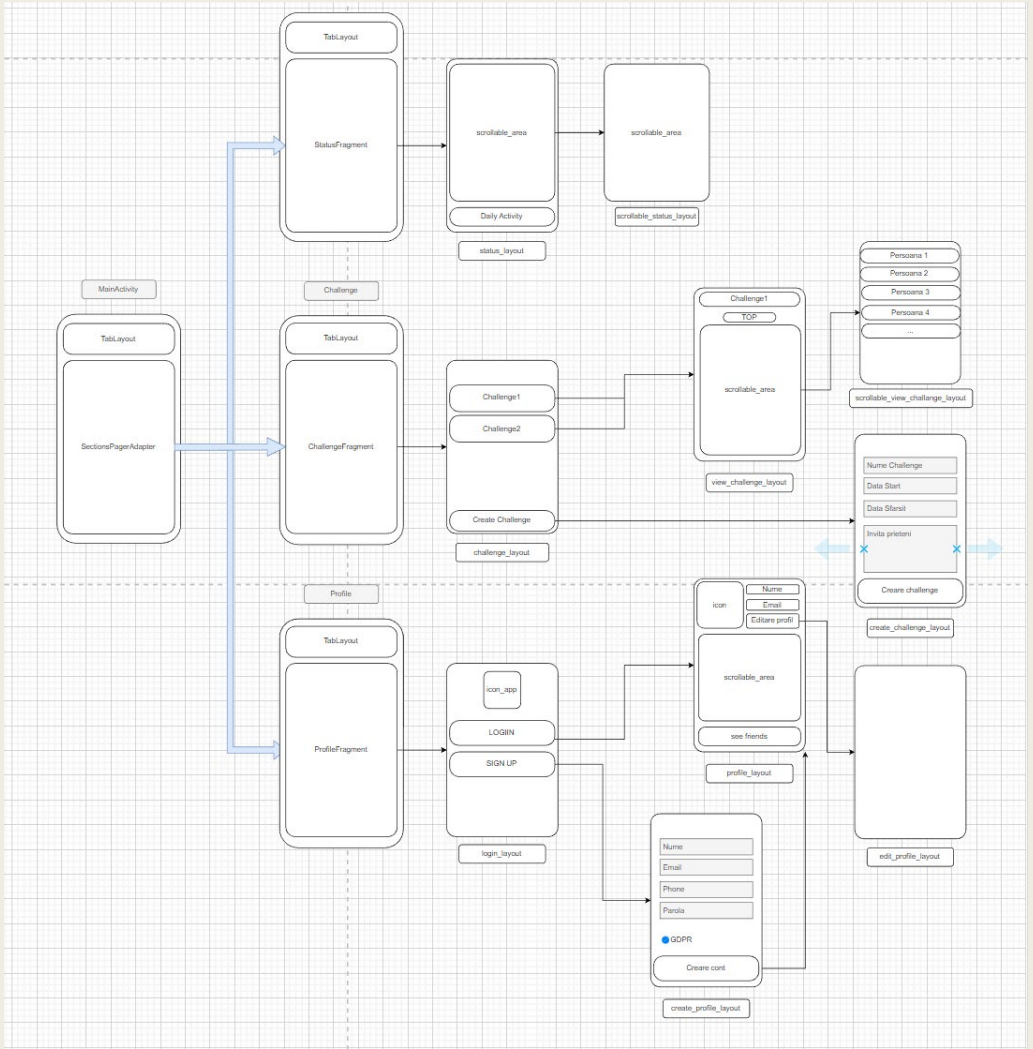
Structura și principalele module:

Modelul de structura general al interfetei aplicatiei:



Structura și principalele module:

Layouturi constructie aplicatie



Structura și principalele module:

Structura baze de date a aplicatiei

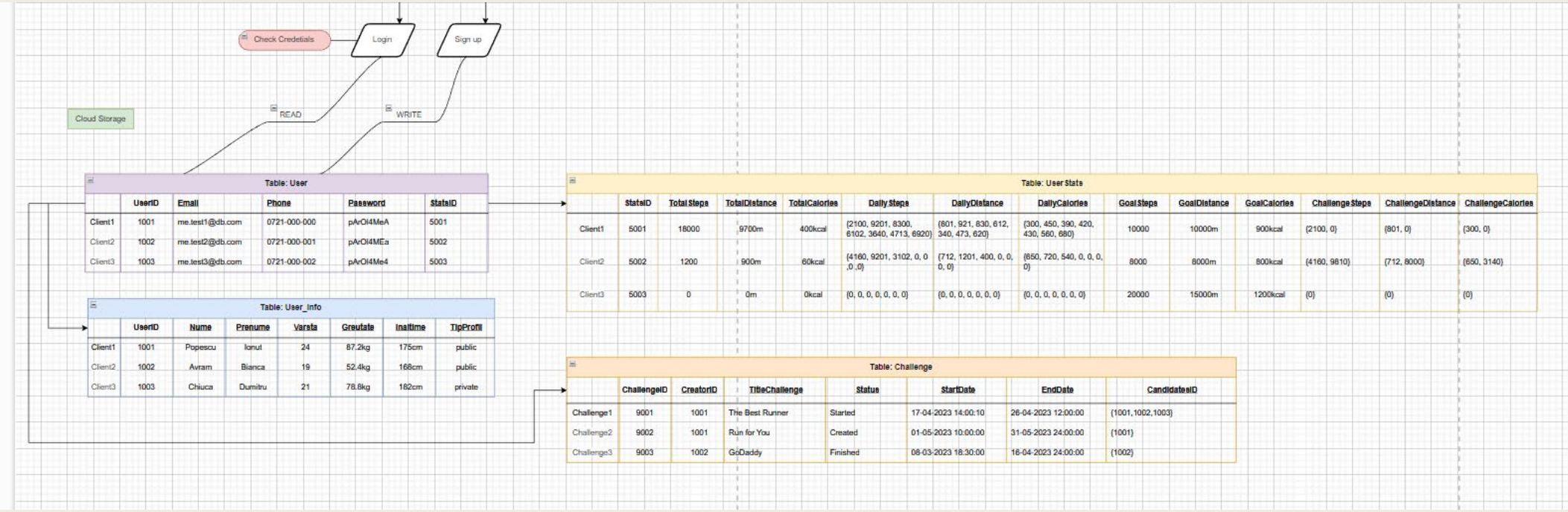


Table: User

	UserID	Email	Phone	Password	StatsID
Client1	1001	me.test1@db.com	0721-000-000	pArOl4MeA	5001
Client2	1002	me.test2@db.com	0721-000-001	pArOl4MEa	5002
Client3	1003	me.test3@db.com	0721-000-002	pArOl4Me4	5003

Table: User_Info

	UserID	Nume	Prenume	Varsta	Greutate	Inaltime	TipProfil
Client1	1001	Popescu	Ionut	24	87.2kg	175cm	public
Client2	1002	Avram	Bianca	19	52.4kg	168cm	public
Client3	1003	Chiuca	Dumitru	21	78.8kg	182cm	private

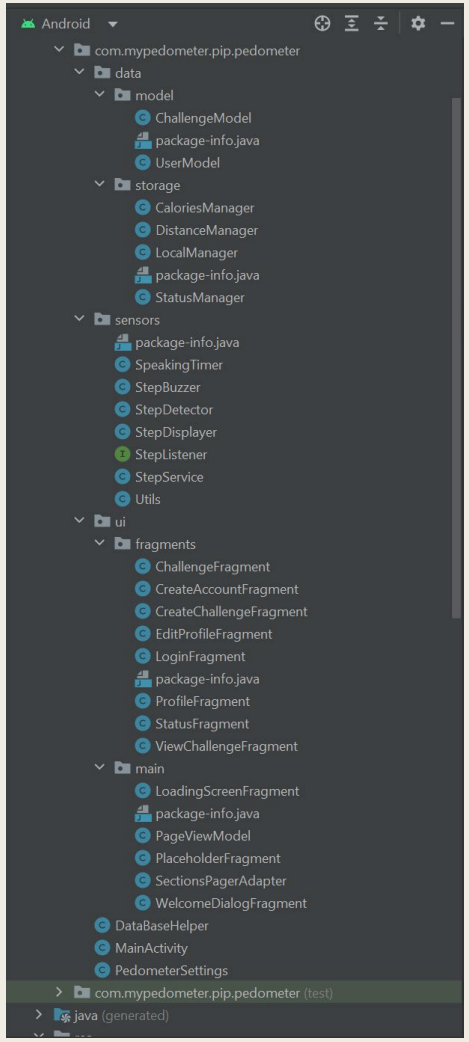
Table: User Stats

	StatsID	Total Steps	TotalDistance	TotalCalories	Daily Steps	DailyDistance	DailyCalories	Goal Steps	GoalDistance	GoalCalories	Challenge Steps	ChallengeDistance	ChallengeCalories
Client1	5001	18000	9700m	400kcal	{2100, 9201, 8300, 6102, 3640, 4713, 6920}	{801, 921, 830, 612, 340, 473, 620}	{300, 450, 390, 420, 430, 560, 680}	10000	10000m	900kcal	{2100, 0}	{801, 0}	{300, 0}
Client2	5002	1200	900m	60kcal	{4160, 9201, 3102, 0, 0 ,0 ,0}	{712, 1201, 400, 0, 0, 0, 0}	{650, 720, 540, 0, 0, 0, 0}	8000	8000m	800kcal	{4160, 9810}	{712, 8000}	{650, 3140}
Client3	5003	0	0m	0kcal	{0, 0, 0, 0, 0, 0, 0}	{0, 0, 0, 0, 0, 0, 0}	{0, 0, 0, 0, 0, 0, 0}	20000	15000m	1200kcal	{0}	{0}	{0}

Table: Challenge

	ChallengeID	CreatorID	TitleChallenge	Status	StartDate	EndDate	CandidatesID
Challenge1	9001	1001	The Best Runner	Started	17-04-2023 14:00:10	26-04-2023 12:00:00	{1001,1002,1003}
Challenge2	9002	1001	Run for You	Created	01-05-2023 10:00:00	31-05-2023 24:00:00	{1001}
Challenge3	9003	1002	GoDaddy	Finished	08-03-2023 18:30:00	16-04-2023 24:00:00	{1002}

Structura și principalele module:



<u>Structura generala a claselor folosite:</u>

data.model :

reprezinta un model a cum trebuie sa fie salvate informatiile in aplicatie, o structura clara care sa poate fi sincronizata cu baza de date

data.storage:

locatia in care se stocheaza toate informatiile locale ale aplicatiei, principalul fiind LocalManager, clasa de tip Singleton, care se creaza o data la instalarea aplicatiei, si mai apoi se actualizeaza regulat cu DB

sensors:

aici se face prelucrarea informatiile brute venite de la sensor, se capteaza informatiile prin niste reguli de validare, iar daca miscarea se considera a fi de reliable, atunci se modifica in LocalManager informatia

ui.fragments:

reprezinta totalitatea fragmentelor necesare in aplicatie, aici se face prelucrearea informatiilor de pe local si se afiseaza live in interfata, informatia fiind persistenta si actualizandu-se regulat

ui.main

reprezinta mecanismul aplicatiei prin care comutam taburile si pastram informatiile prin LiveData

DataBaseHelper – faciliteaza comunicarea cu baza de date
MainActivity – clasa/activitatea de start a aplicatiei

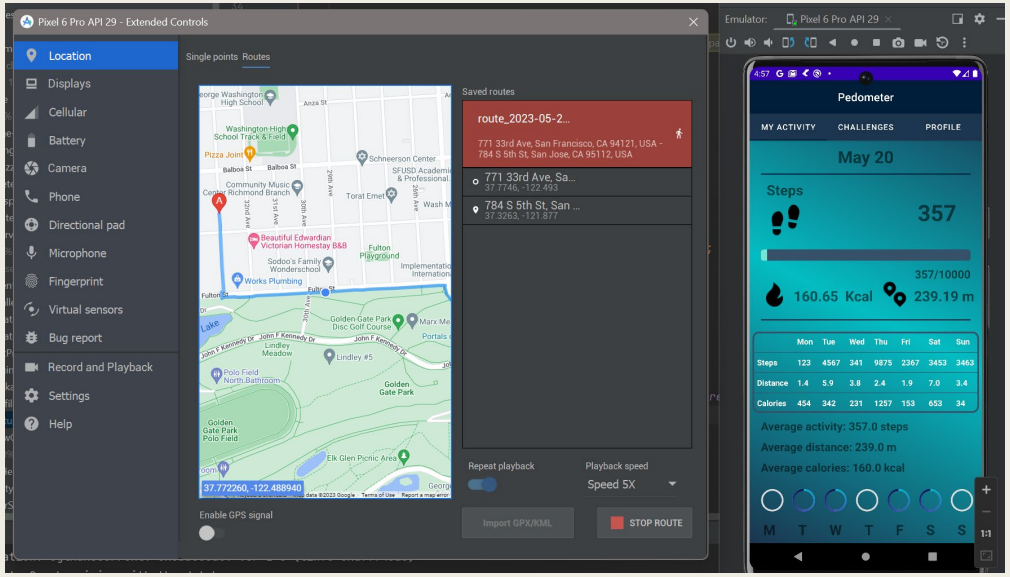
Testare: JUnit & Emulator environment

Element	Class, %	Method, %	Line, %
com	5% (16/276)	20% (328/1588)	12% (800/6444)
mypedometer	5% (16/276)	20% (328/1588)	12% (800/6444)
pip	5% (16/276)	20% (328/1588)	12% (800/6444)
pedometer	5% (16/276)	20% (328/1588)	12% (800/6444)
data	36% (16/44)	49% (328/668)	35% (800/2256)
model	100% (16/16)	70% (328/468)	53% (800/1504)
ChallengeModel	100% (2/2)	17% (4/23)	16% (17/101)
UserModel	100% (2/2)	82% (78/94)	66% (183/275)
storage	0% (0/28)	0% (0/200)	0% (0/752)
sensors	0% (0/32)	0% (0/220)	0% (0/636)
ui	0% (0/164)	0% (0/428)	0% (0/2000)
BuildConfig	0% (0/1)	0% (0/1)	0% (0/1)
DataBaseHelper	0% (0/3)	0% (0/25)	0% (0/211)
MainActivity	0% (0/4)	0% (0/18)	0% (0/127)
PedometerSettings	0% (0/1)	0% (0/24)	0% (0/49)

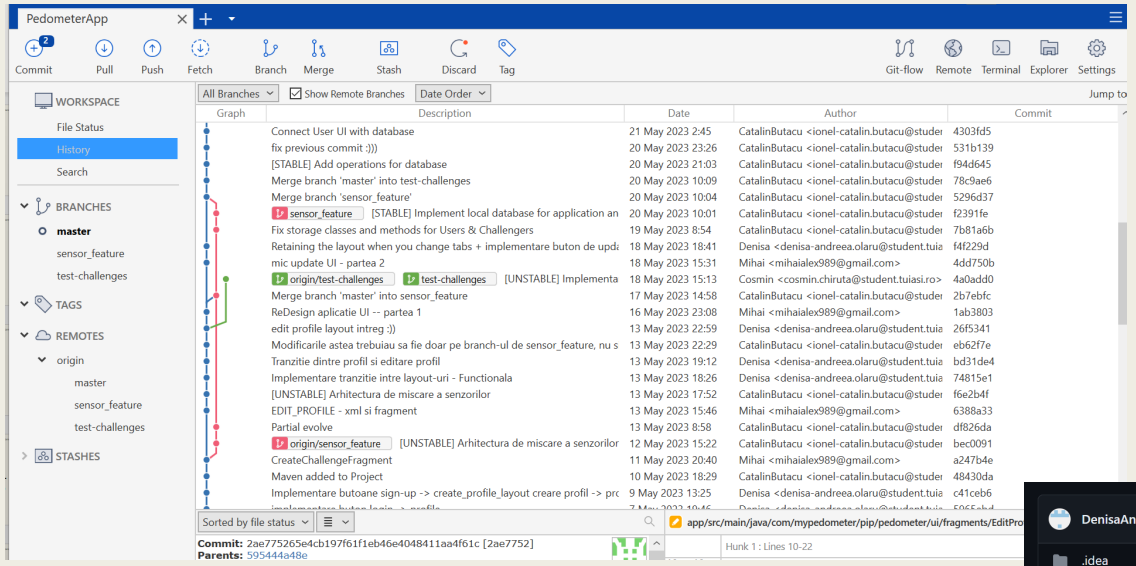
Aditional am folosit emulatorul din Android Studio pentru a putea testa functionalitatile, dar se putea face live debugging si direct pe dispozitivul personal.

Structura generala a claselor folosite:

Testarea in limita timpului s-a focusat pe captarea si retinerea informatiilor privind utilizatorii, ne-am axat pe zona din structrura Model a aplicatiei, pentru a avea confirmearea ca structura locala si baza de date corespunde cu baza de date aleasa si vor aparea surprinze datorita acesteia, un MVP altfel spus.

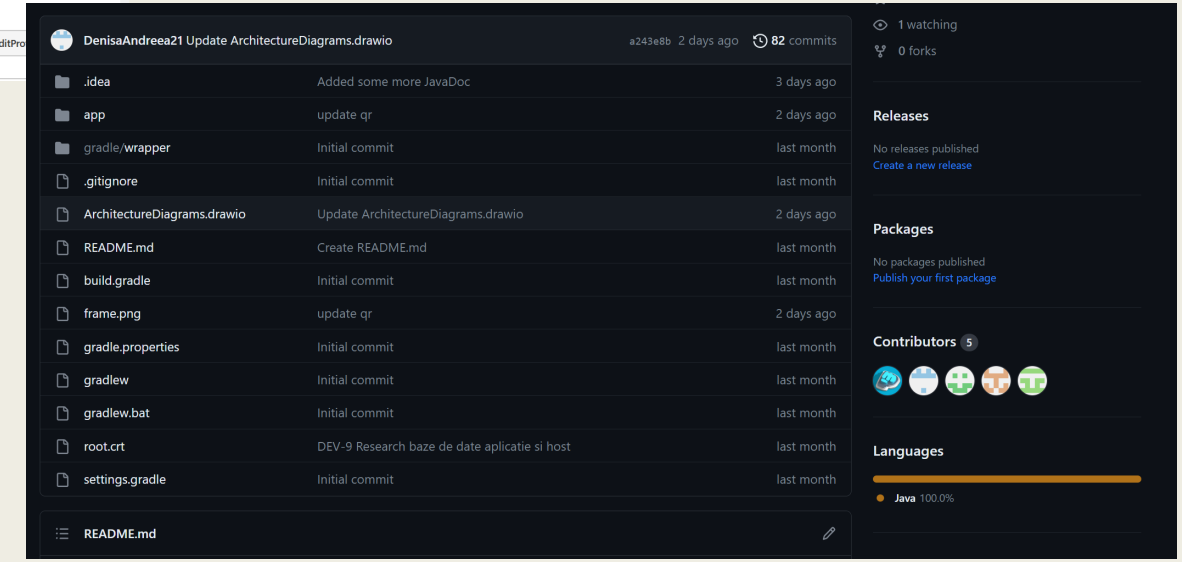


Utilizare Git/Github



Noi am folosit Sourcetree ca interfata pentru prelucrarea cu Git-ul, astfel gestionarea codului sursa a fost foarte usor de realizat. In total am avut 82 de commit-uri si 7 saptamani de dezvoltare a aplicatie ☺

Actualizarea codului s-a incercat a se mentine cat mai constanta pentru a fi probleme foarte mari privind integrarea, astfel chiar si taskurile partial facute erau aduse pe branch pentru a se putea incepe si alte taskuri.



Viitoare implementări/îmbunătățiri:

- Crearea unu serviciu de sincronizare a bazei de date locale cu un Cloud
- Algoritm mai complex de detectare a pașilor
- Algoritmi mai corecți de calcul al caloriilor si distanțelor

Întrebări?

Mulțumim pentru atenție!

Surse bibliografice & documentatii:

https://developer.android.com/guide/components/fundamentals
https://developer.android.com/topic/architecture/intro
https://developer.android.com/jetpack/compose/tutorial
https://developer.android.com/develop/ui/views/layout/declaring-layout
https://developer.android.com/develop/ui/views/notifications
https://www.mdpi.com/1424-8220/18/1/297
https://vladmihalcea.com/yugabytedb-connection-pooling/
https://www.youtube.com/watch?v=312RhjfetP8
https://www.youtube.com/watch?v=PiExmkR3aps
https://www.youtube.com/watch?v=pkT7DU1Yo9Q&list=PLc2rvfiptPSQrErIlwHz7DZrNfhFdDXa6