Arhitectura Aplicatiei

-Lost Pets-

**1.Detalii aplicatie-Flows**

Aplicatia pune la dispozitia clientilor un serviciu de gasire a animalutului pierdut.

Utilizatorii pot adauga anunturi cu animale pierdute/gasite, isi pot crea un cont pentru a primi mesaje, notificari si de asemenea, au acces la diferite servicii cum ar fi alerte, locatia unde a fost ultima data vazut un animal, precum si descarcarea de statistici privind numarul de cazuri de animale disparute.

Pentru a putea adauga un anunt, pentru a putea primi notificari si a fi la curent cu toate noutatile, utilizatorul va trebui sa-si creeze un cont in aplicatie.

La momentul inregistrarii, aplicatia va interoga baza de date pentru a vedea daca nu mai exista un utilizator cu acelasi email.Utilizatorul va completa un formular de inregistrare, acesta va trebui sa furnizeze informatii precum numele si prenumele, o parola (care va fi hash-uita) si toate aceste informatii vor fi salvate in database.

Fiecare user va fi notificat pe site atunci cand un animal este pierdut in zona sa.

De asemenea, un user va avea acces zilnic la articole de specialitate .

**2.Database**

Am folosit o baza de date de tipul MySQL care are 3 tabele.

Tabela ***users*** in care stocam urmatoarele informatii despre user:

* + Un id generat automat(primary key)
  + Numele introdus la inregistrare
  + Prenumele introdus la inregistrare
  + Parola introdusa la inregistrare (sha256)
  + Adresa introdusa la inregistrare
  + Latitudinea detectata cu OpenCageData Geocode
  + Longitudinea detectata cu OpenCageData Geocode

Latitudinea si longitudinea sunt obtinute in functie de adresa introdusa.

Tabela ***pets*** in care stocam urmatoarele informatii despre anunturi:

* + Un id generat automat(primary key)
  + Id-ul user-ului care a postat anuntul
  + Numele animalutului introdus
  + Adresa unde acesta a fost pierdut/gasit
  + Longitudinea detectata cu OpenCageData Geocode
  + Latitudinea detectata cu OpenCageData Geocode
  + Descrierea introdusa
  + Rasa introdusa
  + O descriere a zgardei (poate fi NULL)
  + Recompensa (poate fi NULL – recomandat pentru cazurile de LOST)
  + Un timestamp al postarii (pentru afisare in ordine cronologica)
  + Un status ( poate fi 1 sau 2 , FOUND sau LOST)
  + Un display-url catre imagina uploadata in anunt

Tabela ***persistence*** care este folosita pentru persistenta user-ului pe site si in care stocam urmatoarele informatii:

* + Un id generat automat(primary key)
  + Id-ul user-ului logat
  + Un timetamp al logarii
  + IP-ul user-ului ($\_SERVER[‘REMOTE\_ADDR’])

**3.Autentificare**

Dupa inregistrare, user ul se poate loga in cont folosind email-ul ales si parola, si are acces la formularele de gasit/pierdut animal. Parola va fi hash-uita si server-ul va face un query in baza de date cu email-ul introdus. Daca mail-ul este gasit si hash-urile fac match, atunci user-ul va fi logat cu success.

In cazul logarii cu succes, server-ul va stoca in tabela persistence data un timestamp al logarii, ID-ul user-ului si IP-ul.

**4.Persistenta utilizatorului logat**

Persistenta se va realiza cu ajutorul tabelei persistence care tine evidenta utilizatorilor logati recent.

Cand user-ul intra pe site, site-ul apeleaza endpoint-ul check\_persistence care verifica in baza de date IP-ul user-ului.

Daca acesta apare in tabela si s-a logat recent (in ultima ora), acesta va fi logat automat in ultimul cont logat. Daca acesta apare in tabela dar nu s-a logat recent, va fi sters din tabela iar endpoint-ul va returna false, fortand astfel user-ul sa se logheze iar.

User-ul este sters automat daca da logout.

**5.API-uri externe**

Am folosit api-ul de la OpenStreetMap pentru a gasi anunturile din zona unui user dar si pentru mapa care apare pe anunt. Precizare: OpenStreetMap ofera 2 API-uri, unul live si unul pentru unit testing si development, noi l-am folosit pe cel pentru unit testing si development.

Am folosit api-ul de la OpenCageData pentru a obtine o adresa folosind latitudinea si longitudinea si invers.

Am folosit api-ul de la ImgBB pentru stocarea imaginilor uploadate la anunt, in database stocam doar un link de pe imgbb care duce la imagine.

**6.API REST**

Am facut un API REST pentru partea de backend, disponibil la path-ul /api/ la care facem apeluri ajax.Toate endpoint-urile primesc/returneaza body-uri de tip JSON. Toate endpoint-urile fac verificari de integritate al body-ului si + alte verificari specifice fiecarui endpoint.

**API/register**

**METHOD: POST**

Api/register se ocupa cu inregistrarea a unui nou user in baza de date. Acesta verifica daca datele introduse sunt valide (password = confirm\_password, nu lipseste nici un camp din body, email-ul nu a mai fost folosit), in caz ca exista erori de orice natura, endpoint-ul intoarce “error” la status si un mesaj de eroare impreuna cu un cod HTTP corespunzator

**API/login**

**METHOD: POST**

Api/login se ocupa cu verificarea credentialelor de login si intoarce ID-ul user-ului respectiv pentru a putea fi folosit pe partea de front. Acest endpoint salveaza si ID-ul respectiv in database pentru persistenta.

**API/persistence**

**METHOD: GET**

Acest endpoint verifica daca IP-ul user-ului exista in baza de date si daca s-a logat recent, returneaza “not logged” sau ID-ul contului din database corespunzator.

**API/delete\_ann**

**METHOD: POST**

Acest endpoint primeste un id\_pet si sterge anuntul corespunzator din baza de date

**API/download\_statistics**

**METHOD: POST**

Acest endpoint se ocupa cu crearea fisierelor corespunzatoare de download al statisticilor. Primeste toate statisticile si un file\_type care poate fi “csv”,”html” sau “pdf” si genereaza fisierul de download corespunzator.

**API/get\_statistics**

**METHOD: GET**

Acest endpoint calculeaza statistici precum numarul total de animale pierdute/gasite, zona cea mai vulnerabila,etc cu ajutorul informatiilor din baza de date si le returneaza.

**API/get\_all\_ann**

**METHOD: POST**

Acest endpoint returneaza toate anunturile in functie de un anumit status (LOST/FOUND), primeste 0,1 sau 2 si returneaza anunturile corespunzatoare din baza de date

**API/get\_ann**

**METHOD: POST**

Acest endpoint primeste id-ul unui user si ia coordonatele sale din database dupa care apeleaza api-ul de la OpenStreetMap care returneaza latitudinea minima/maxima si longitudinea minima/maxima pe care un anunt trebuie sa le aiba pentru a se afla in vecinatatea user-ului respectiv.

Returneaza toate anunturile din vecinatatea acelui user.

**API/get\_area\_ann**

**METHOD:POST**

Acest endpoint primeste latitudinea si longitudinea unei zone, un status (LOST/FOUND) si o rasa. Calculeaza latitudinea minima, latitudinea maxima, longitudinea minima si longitudinea maxima (folosind OpenStreetMap), dupa care ia toate anunturile cu statusul respectiv si cu rasa respectiva si le returneaza.

**API/get\_bbox**

**METHOD: POST**

Acest endpoint este folosit la afisarea hartii pe pagina. Harta are nevoie de 4 coordonate (bbox) pentru a afisa zona corecta, le obtinem folosind api-ul OpenStreetMap

**API/get\_user\_ann**

**METHOD: POST**

Acest endpoint ia toate anunturile puse de un anumit user. Este folosit pentru sectiunea “MY POSTED ENTRIES”.

**API/get\_user\_info**

**METHOD: POST**

Acest endpoint returneaza datele unui user, cum ar fi e-mailul, numele, prenumele, etc. Primeste doar id-ul din baza de date. Este folosit la personalizarea paginilor.

**API/logout**

**METHOD: POST**

Acest endpoint este apelat la logout. Primeste id-ul user-ului care face logout si il sterge din tabela de persistence pentru a nu fi logat automat inapoi.

**API/post\_announcement**

**METHOD: POST**

Acest endpoint primeste toate datele din formularul de pe site si le salveaza in baza de date impreuna cu ID-ul user-ului care a completat formularul.

Este diferit fata de celelalte endpoint-uri deoarece primeste id-ul user-ului prin headers nu prin body. ID-ul user-ului este pus in headers, intr-un camp cu numele “Bearer”.

**API/update\_location**

**METHOD: POST**

Acest endpoint primeste id-ul unui anunt, noile coordonate si noua adresa reprezentata de acele coordonate si face update in baza de date.