Car Planner

Document de Proiectare Arhitecturala

Studenti:

Stefan Stefan, MTI

Rizescu Eusebiu, MTI

1. Introducere
2. Prezentarea generala a arhitecturii sistemului
3. Decompozitia in subsisteme si responsabilitatile fiecarui subsistem
4. Distributia subsistemelor pe platforme hardware/software
5. Managementul datelor persistente
6. Controlul accesului utilizatorilor la sistem
7. Fluxul global al controlului

1. Introducere

- **Scopul sistemului**

CarPlanner are ca scop sa ajute proprietarii si utilizatorii de vehicule sa gestioneze cu usurinta autovehiculele. Fiecare autovehicul are nevoie de schimburi periodice variabile (durata timp, durata kilometri), dar totodata are nevoie si de documente valabile pentru a putea circula in legalitate (asigurare, inspectie tehnica periodica, taxa de drum). Utilizatorii pot adauga/edita/sterge autovehicule, pot adauga/edita/sterge documente, scadente, revizii. Toate aceste documente si revizii sunt de trei categorii: cu data de expirare, cu numar maxim de kilometri sau ambele. Cu cateva zile inainte de atingerea expirarii, utilizatorul este anuntat prin email cu privire la documentul / revizia care se apropie de expirare.

- **Obiective de proiectare**
    1. Mentenabilitatea aplicatiei. Extensibilitate. Adaptabilitate. Portabilitate
    2. Folosirea traficului criptat (HTTPS)
    3. Parolele tinute criptate in baza de date
    4. Site-ul sa arate bine atat pe desktop cat si pe telefoane/tablete
    5. Timpul de raspuns al aplicatiei sa fie decent (sub 1 secunda per pagina)
    6. Cache peste baza de date (memcached spre exemplu)
    7. Verificarea contului nou prin trimitere link unic pe mail
    8. Proceduri bine puse la punct pentru backup-ul bazei de date

2. Prezentarea generala a arhitecturii sistemului

Solutia propusa consta intr-o aplicatie web, bazata pe microservicii. Microserviciile, de asemenea cunoscute si ca arhitectura de microservicii, este un stil arhitectural care structureaza o aplicatie ca o colectie de servicii care:

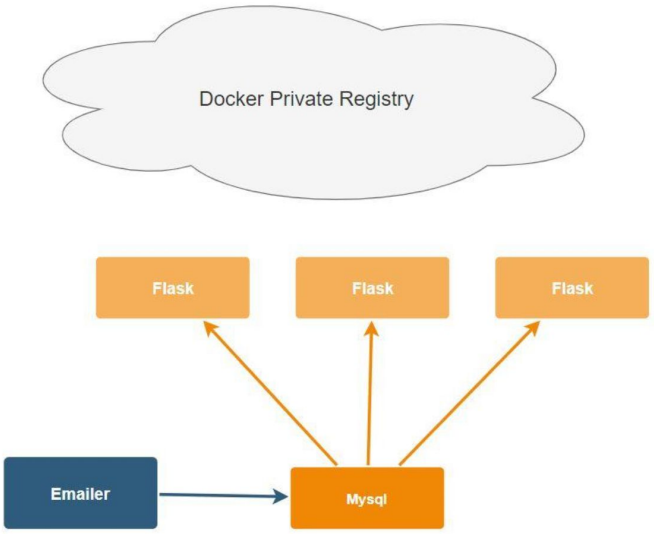
- se pot intretine si testa usor
- sunt slab cuplate
- se pot implementa independent

Aplicatia prezentata va rula in prima faza pe un singur host. In cazul in care aplicatia va fi intens folosita, datorita faptului ca este implementata folsind docker, este usor sa se poata scala pentru a rula pe mai multe servere.

Bineinteles, in functie de utilizare a aplicatiei si de costuri, putem migra spre servicii dedicate de la Amazon spre exemplu, cum ar fi baza de date MySQL (sau Aurora) administrata complet de AWS (upgrade-uri, redundanta), cat si a aplicatiei web in sine, cu ajutorul ECS (Amazon Elastic Container Service)

3. Decompozitia in subsisteme si responsabilitatile fiecarui subsistem

Structura aplicatiei este urmatoarea:



Componenta **Flask** din diagrama, adica serviciul WEB este implementat cu ajutorul framework-ului de Python numit Flask.

Componenta **Emailer** functioneaza ca un daemon. In fiecare zi scaneaza baza de date si trimite mail utilizatorului cu scadentele care sunt aproape de a expira. De asemenea, aceasta componenta trimite in fiecare saptamana o lista cu urmatoarele 10 scadente.

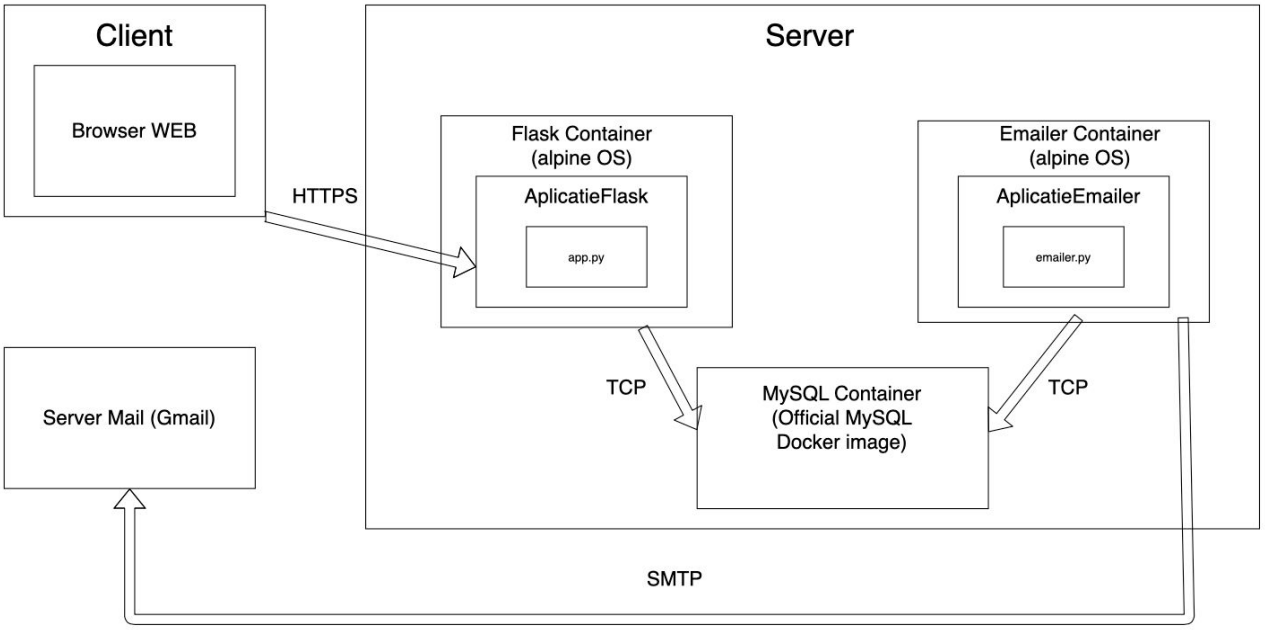
Componenta **MySQL** este bineinteles, baza de date, de care avem nevoie pentru a stoca persistent datele utilizatorilor si ale autovehiculelor lor.

Componenta **Docker Registry** este o componenta optionala aplicatiei, in care se stocheaza imaginile de Docker pentru a fi complet independenti de alte repository-uri si pentru a avea istoricul cu fiecare versiune. Astfel, putem aduce live pentru teste orice versiune a aplicatiei.

Daca dorim sa trecem cu totul la arhitectura serverless, acest lucru se poate face foarte usor (bineinteles, totul depinde de cost):

- Flask - cu ajutorul Fargate peste ECS
- Emailer - Lambda functions
- MySQL - oferita ca serviciu de AWS
- Docker Registry - AWS ECR (Elastic Container Registry)

4. Distributia subsistemelor pe platforme hardware/software



Dupa cum se poate observa si in diagrama de distributie prezentata mai sus, sistemul nostru se bazeaza pe arhitectura client-server, dar contine si o componenta care se ocupa cu persistenta datelor. Totodata, componenta emailer se conecteaza la serverul de mail al Google, pentru a putea trimite notificari utilizatorilor de pe o adresa configurata in aplicatie. In viitor, aceasta metoda ar trebui elimitana, si ar trebui folosit un server de mail propriu de pe care sa se trimita emailurile.

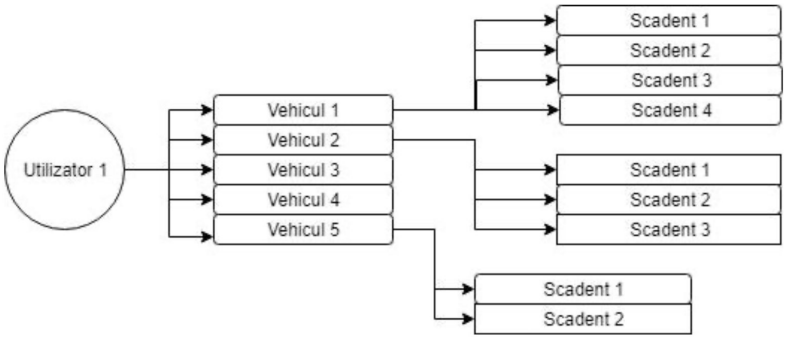
Clientul se va conecta in aplicatie cu ajutorul unui browser WEB de pe orice device doreste (telefon, tableta, desktop). Comunicarea intre client si server se va face prin protocolul HTTPS, comunicarea intre Flask si Emailer catre Baza de Date se va face prin TCP, iar comunicarea dintre Emailer si Google Mail se va face prin SMTP

5. Managementul datelor persistente

Principalele entitati pe care aplicatia este construita sunt:

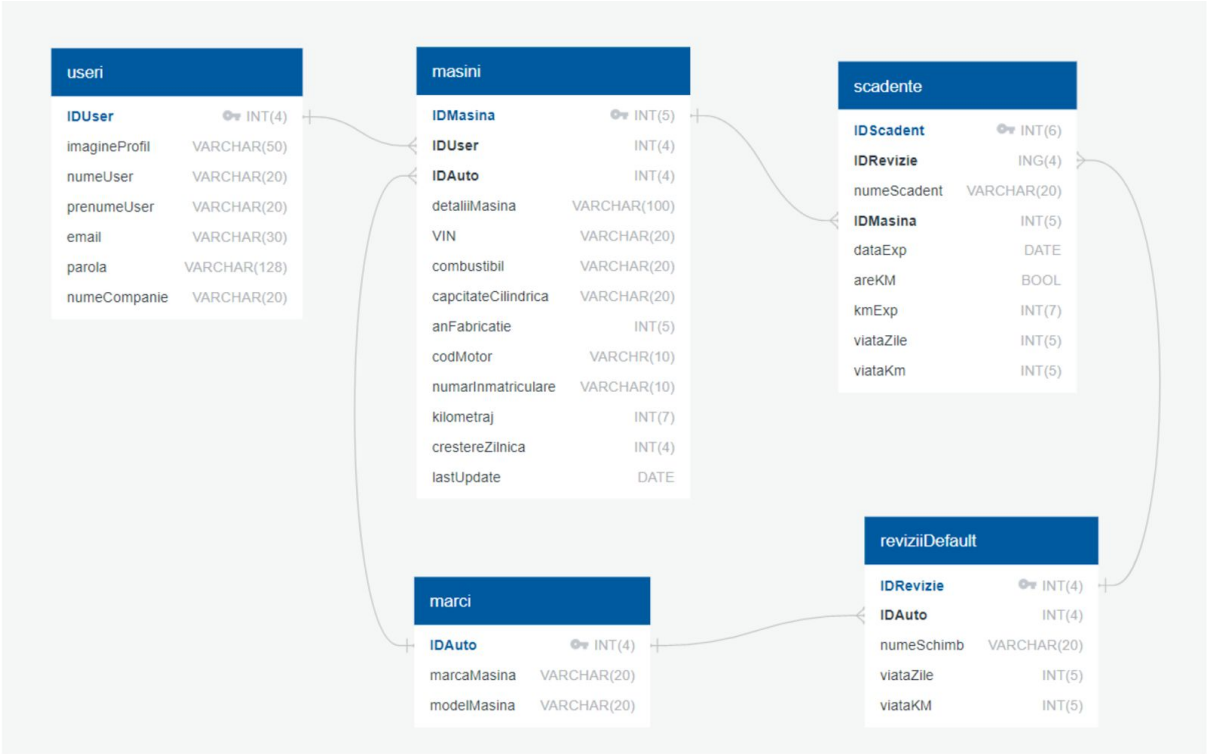
- Utilizator
- Vehicul
- Scadent

Pe scurt relatiile dintre aceste entitati sunt urmatoarele:



Pentru a stoca persistent cele trei entitati (user, vehicul, scadent) avem nevoie de o baza de date, accesibila atat din componenta Flask (pentru operatiile CRUD), cat si din componenta Emailer (pentru raportare scadente).

Structura bazei de date este urmatoarea:



Detalii despre tabelele prezente in baza de date si legaturile dintre acestea:

1) Marci – tabela este populata cu cele mai comune marci si modele de masini populare in Europa. Cheia primara este IDAuto

2) ReviziiDefault – tabela care contine, pentru fiecare model de masina, o serie de revizii default pe care utilizatorul le poate asocia cu o masina proaspat adaugata. Cheia primara este IDRevizie. Are si o cheie straina IDAuto catre tabela Marci

3) Useri – tabela care contine toti userii inregistrati in aplicatie. Aici sunt stocate numele, prenumele, email-ul (care este unic), parola (stocata in hash) si numele copaniei (daca este cazul). Cheie primara este IDUser

4) Masini – tabela care contine toate masinile adaugate de utilizatori. Aici sunt stocate pe langa detaliile masinii, ultimele actualizari ale kilometrajului pentru fiecare masina. Cheie primara este IDUser, o cheie straina este IDUser catre tabela Useri si alta cheie straina IDAuto catre tabela Marci

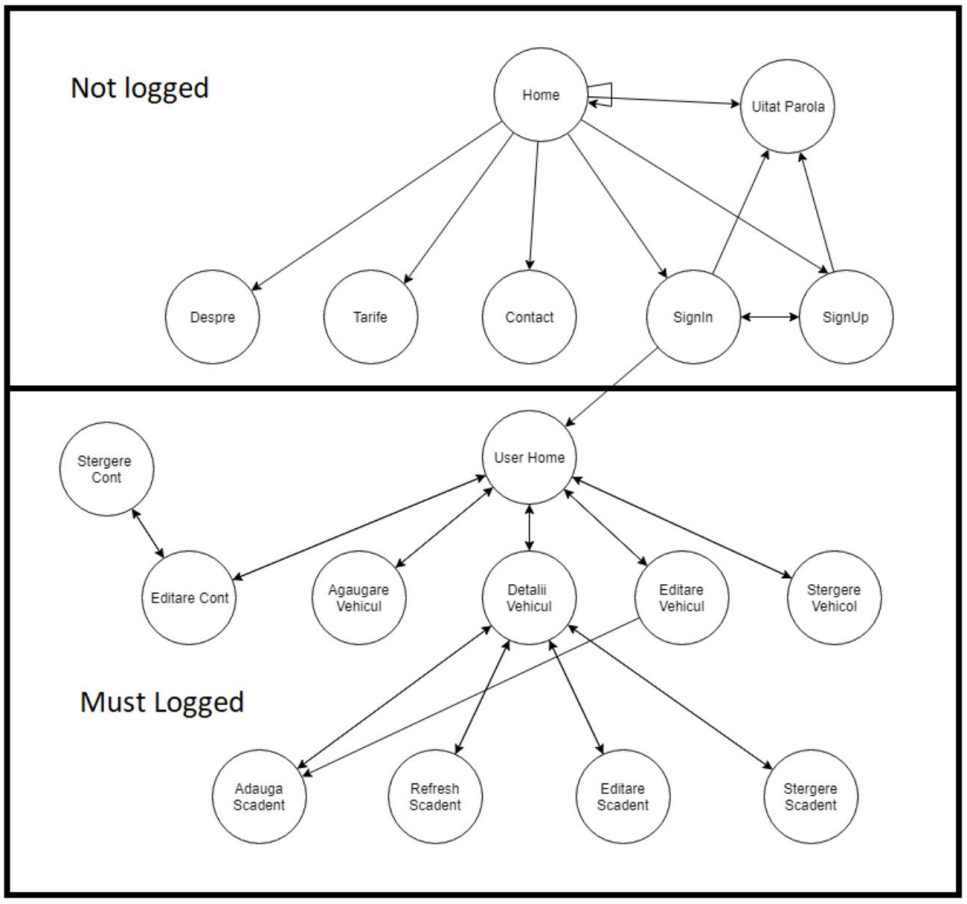
5) Scadente – tabela care contine toate scadentele ale tuturor masinilor ale tuturor utilizatorilor. Cheia primara este IDScadent, o cheie straina este IDMasina catre tabela Masini si alta cheie straina IDRevizie catre tabela ReviziiDefault

6. Controlul accesului utilizatorilor la sistem

Tip Utilizator	Normal	Admin	Support
Creare cont nou	X	X	X
Autentificare	X	X	X
Editare/Stergere cont propriu	X	X	X
Editare/Stergere orice cont		X	
Adaugare/Editare/Stergere autovehicul	X	X	
Adaugare/Editare/Stergere document	X	X	
Vizualizare autovehicul/document cont propriu	X	X	X
Vizualizare autovehicul/document orice cont		X	X
Vizualizare toate conturile din baza de date		X	X

7. Fluxul global al controlului

Harta aplicatiei web se poate vedea in figura de mai jos. In partea de sus sunt paginile care pot fi vizualizare fara autentificare, iar in partea de jos sunt paginile care pot fi vazute doar dupa autentificare.



Fiecare utilizator, pentru a putea folosi aplicatia trebuie sa isi faca un cont personal cu care va trebui sa se autentifice ulterior pentru a putea adauga, edita sau sterge vehicule si scadente. Totodata, un utilizator normal nu poate accesa nici o pagina a altui utilizator. Daca totusi un utilizator normal va incerca sa acceseze o pagina a altui utilizator (pagina care necesita autentificae bineinteles), va fi redirectat catre pagina custom “403”.

Atunci cand un utilizator doreste sa isi creeze un cont, contul nu va fi implicit creat, ci va fi intr-o stare de asteptare, pana cand utilizatorul va accesa linkul primit in mailul de verificare. Astfel, o persoana de rea-credinta nu isi poate face un cont in aplicatie

cu o adresa de mail care nu ii apartine. Acelasi rationament este valabil si cu functionalitatea “Am uitat parola”. Pentru a se resta parola, utilizatorul va trebui sa acceseze un link primit pe email la adresa specificata in formular.

Parolele nu sunt stocate in clar in baza de date. In baza de date se tine un hash al parolei, hash calculat cu ajutorul functiei “generate_password_hash” din pachetul “werkzeug.security”, pachet Python cunoscut pentru nivelul de securitate pe care il ofera.

Datorita faptului ca se foloseste wrapper-ul SQLAlchemy, si nu direct comenzi de inserare / updatare in baza de date, posibilitatea de SQL injection este eliminata deoarece acest pachet Python are inclusa sanitizarea fiecarei operatii cu baza de date in vederea SQL injection.