**Universitatea „Alexandru Ioan Cuza” Iași**

`

**Facultatea de Informatică**

****

LUCRARE DE LICENȚĂ

**SmartCams**

propusă de

**Mărtinaș Alexandru**

**Sesiunea:** Iulie, 2018

Coordonator științific

**Conferențiar, Dr. Sabin Corneliu Buraga**

**Universitatea „Alexandru Ioan Cuza” Iași**

**Facultatea de Informatică**

**SmartCams**

Managementul dispozitivelor interconectate prin intermediul camerelor de supraveghere

**Mărtinaș Alexandru**

**Sesiunea:** Iulie, 2018

Coordonator științific

**Conferențiar, Dr. Sabin Corneliu Buraga**

Avizat,

Îndrumător Lucrare de Licenţă

Titlul, Numele şi prenumele \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAŢIE privind originalitatea conţinutului lucrării de licenţă/diplomă/disertaţie/absolvire**

Subsemntatul(a) ………………………………………………………………………………………

domiciliul în …………………………………………………………………………………………………..

născut(ă) la data de ………………..…., identificat prin CNP ………….……………..………………..., absolvent(a) al(a) Universităţii „Alexandru Ioan Cuza” din Iaşi, Facultatea de ………………………. specializarea …………………………………………………………, promoţia ………………………….,

declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată sub îndrumarea dl. / d-na \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, pe care urmează să o susţină în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă/diplomă/disertaţie/absolvire să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licenţă cu titlul „*SmartCams*”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iaşi să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iaşi, *29.06.2018*

Mărtinaș Alexandru

(semnătura în original)

**Rezumat**

*„Internet of Things”*  este un concept foarte popular in ultima vreme. Pe larg, IoT, cum este prescurtat, reprezinta o noua viziune in care internetul se extinde in lumea reală incluzând obiectele de zi cu zi care pot fi conectate la Internet. Ce este special însă e faptul ca permite conectarea și schimbul de date dintre dispozitive, care, cu ajutorul unor sisteme software, pot fi folosite in orice domeniu pentru a controla sau automatiza diferite sarcini sau acțiuni.

Prin intermediul aplicației *„SmartCams”* incercăm sa facilităm interacțiunea dintre utilizatori si dispozitivele inteligente si sa aducem un plus de siguranța mediului în care trăiește sau își desfășoară activitatea. Aceast lucru se realizează prin transformarea sistemul de supraveghere video al utilizatorilor într-un sistem inteligent care monitorizează orice eveniment vizibil și, prin intermediul unui set de reguli definite de utilizatori, reacționeaza la diferite evenimente modificând starea dispozitivelor conectate și trimițând diferite notificări utilizatorilor cu privire la acestea sau la posibilele amenințari la siguranta sa sau a mediului supravegheat (e.g incendiu, hoti, etc).

Aplicația a fost dezvoltată cu ajutorul *framework*-ului Django, un *framework* bazat pe limbajul de programarePython pentru dezvoltarea de aplicații web complexe intr-un mod rapid și eficient. Notificarea utilizatorilor se realizează prin două metode. Prima este serviciul de *Push Notification* suportat de majoritatea *browser*-elor. A doua optiune este furnizata de AWS SNS și constă în trimiterea de SMS-uri utilizatorului sau apropiaților acestuia. Pentru stocarea de date am ales să folosim o baza de date relaționala(e.g MySQL) mai puțin pentru imaginile furnizate de camerele de supraveghere, care sunt stocate în *Cloud* de serviciu de stocare AWS S3. Mai trebuie menționată procesarea acestora, care este realizata cu ajutorul serviciului AWS Rekognition.

Următoare capitole vor prezenta arhitectura si tehnologiile folosite in dezvoltarea aplicației. Se va detalia soluția oferita si avantajele acesteia. De asemenea, se va prezenta modul de utilizare al aplicației impreună cu exemplificarea unor posibile scenarii majore. Prezenta lucrare se incheie cu un capitol rezervat concluziilor si a unor posibile imbunătățiri.

Cuprins

[1.Introducere 8](#_Toc517825233)

[1.1 Obiective 8](#_Toc517825234)

[1.2 Formularea problemei 8](#_Toc517825235)

[1.3 Soluții existente 8](#_Toc517825236)

[1.4 Prezentarea soluției 9](#_Toc517825237)

[2. Analiza și proiectarea aplicației 10](#_Toc517825238)

[2.1 Analiza problemei 10](#_Toc517825239)

[2.1.1 Problema existentă 10](#_Toc517825240)

[2.1.2 Identificarea soluției 10](#_Toc517825241)

[2.2 Proiectarea aplicației 10](#_Toc517825242)

[2.2.1 Cerințele aplicației 10](#_Toc517825243)

[2.2.2 Arhitectura generala 12](#_Toc517825244)

[2.2.3 Cameras 13](#_Toc517825245)

[2.2.4 People 15](#_Toc517825246)

[2.2.5 Things 16](#_Toc517825247)

[2.2.6 Rules 17](#_Toc517825248)

[2.2.7 Notifications 18](#_Toc517825249)

[2.2.8 Stocarea persistentă a datelor 19](#_Toc517825250)

[3 Tehnologii folosite 21](#_Toc517825251)

[3.1 Django 21](#_Toc517825252)

[3.2 MySQL 22](#_Toc517825253)

[3.3 Amazon Web Sevices 22](#_Toc517825254)

[3.3.1 Simple Storage Service 22](#_Toc517825255)

[3.3.2 Amazon Rekognition 23](#_Toc517825256)

[3.3.3 Simple Notification Service 23](#_Toc517825257)

[3.4 Push Notification 24](#_Toc517825258)

[4.Implementare 26](#_Toc517825259)

[4.1 Resurse 26](#_Toc517825260)

[4.2 Procesarea imaginilor 29](#_Toc517825261)

[5. Manual de utilizare 32](#_Toc517825262)

[6. Scenarii de utilizare 39](#_Toc517825263)

[7. Concluzii 42](#_Toc517825264)

[8. Bibliografie 44](#_Toc517825265)

# 1.Introducere

## Obiective

Prin intermediul acestei lucrari ne propunem să identificam problemele comunicarii intre utilizator si dispozitivele inteligente și cum ne putem folosi de un sitem de supraveghere video normal pentru a facilita aceasta comunicare. De asemenea, vom explora cum putem transforma un asemenea sistem într-un sistem inteligent care să recunoască diferite evenimente posibil periculoase și ce acțiuni poate lua acesta pentru a le combate.

## Formularea problemei

Conceptul de *„Internet of Things”* este din ce in ce mai intâlnit in jurul nostru. In fiecare zi apar noi dispozitive inteligente care să ne usureze viata și care sa automatizeze munca fizică. Am dori ca aceste dispozitive sa se adapteze automat în funcție de nevoile sau preferințele noastre. Cu alte cuvinte, am dori ca aceste dispozitive să poată recunoaște prezența utilizatorilor în mod individual și să își modifice comportamentul in funcție de preferințele acestora fără a necesita intervenția lor fizică.

## Soluții existente

O soluție de bază ar fi folosirea unor senzori de miscare care sa modifice starea dispozitivelor inteligente deținute de utilizator atunci când detectează miscare. Dar asta nu rezolva nici pe departe problema noastră. Un senzor de mișcare nu poate recunoaste identitatea persoanei prezente pentru a reda preferințele acesteia. Se poate defini doar un comportament general indiferent de cine generează activarea senzorilor. De asemenea, am avea nevie de mai multe tipuri de senzori pentru a recunoaște și alte tipuri de evenimente pe langa cel de mișcare(e.g fum).

O altă soluție ar fi utilizarea unui sistem de supraveghere video inteligent care să recunoască identitatea utilizatorului și să comunice dispozitivelor preferințele acestuia. Dar această solutie are o problemă majoră. Camerele video inteligente costă foarte mult având un preț chiar și de 10 ori mai mare față de o cameră obișnuită ceea ce face ca această soluție să fie foarte costisitoare[3]. Cum factorul finaciar este unul crucial pentru majoritatea utilizatorilor, aceasta soluție poate fi folosită de foarte puțini dintre ei.

În concluzie, avem nevoie de o soluție cât mai accesibilă care să poată fi adoptată de cât mai mulți posesori de dispozitive inteligente și care să rezolve cât mai bine problema identificată.

## Prezentarea soluției

*„SmartCams”* se foloseste de soluția sistemulul de supraveghere video inteligent eliminând problema financiară a acestuia. Acest lucru se realizează prin transformarea un sistem de supraveghere normal intr-un sistem inteligent cu ajutorul unei aplicatii web. Prin procesarea imaginilor furnizate de camerele de supraveghere, aplicatia poate recunoaste identitatea utilizatorului și poate reda astfel preferintele acestuia tuturor dispozitivelor conectate. De asemenea, aplicatia poate recunoaste diferite evenimente posibil periculoase care ar putea pune in pericol siguranta utilizatorilor sau a mediului supravegheat (e.g incendiu, hoți) notificându-i urgent pe utilizatori sau cunoscuții acestora.

Aplicația este structurată pe mai multe module. Primul modul face managementul camerelor de supraveghere din sistem. Aici utilizatorul poate înregistra noi camere si poate vedea o transmisiune în timp real a acestora. Urmatorul modul este destinat dispozitivelor inteligente unde sunt înregistrate și unde se poate vedea starea fiecăruia. Urmeaza apoi un modul destinat rețelei de cunoscuți ai utilizatorului. Aici sunt inregistrate toate persoanele pe care dorim ca sistemul să le recunoască. Legatura dintre aceste 3 module este realizata de modulul de reguli. Prin intermediul acestui modul, utilizatorul definește comportamentul dispozitivelor in funcție de persoana sau anumite elemente/evenimente recunososcute de camerele video. De fiecare dată cand una din reguli este recunoscută de sistem, utilizatorul este notificat de catre modulul de notificari prin *push notifications* sau SMS, in funcție de alegerea acestuia.

# 2. Analiza și proiectarea aplicației

## 2.1 Analiza problemei

### 2.1.1 Problema existentă

După cum am mai spus, pe piața dispozitivelor inteligente nu există incă un sistem care să recunoască utilizatorii in mod individual și să interacționeze cu dispozitivele proprii pentru a crea un mediu personalizat in funcție de preferințele acestora. Soluțiile existente în momentul de față sunt incomplete și foarte costisitoare.

### 2.1.2 Identificarea soluției

Soluția identificată de noi incearcă să realizeze această conexiune dintre utilizator și dispozitive transformând un sistem de supraveghere normal intr-un sistem de supraveghere inteligent prin intermediul unei aplicații web. Pe lângă aceasta conexiune, aplicația duce sistemul de supraveghere video la un alt nivel. Cu ajutorul acesteia se vor putea recunoaste automat diferite evenimente care ar putea pune in pericol siguranta utilizatorilor sau a mediului supravegheat (e.g incendiu, hoți)

## 2.2 Proiectarea aplicației

### 2.2.1 Cerințele aplicației

Pentru a înțelege mai bine cum funționează aplicația, este important să prezentăm cerințele pe care aceasta trebuie să le indeplinească.

* Cerințe funcționale:
  + Inregistrarea sistemului video si a dispozitivelor inteligente
  + Vizualizare in timp real a dispozitivelor inregistrate impreună cu starea lor individuală
  + Posibilitatea utilizării platformei pe o scară cât mai largă(e.g Smart House, Smart Office, Smart School)
  + Notificarea utilizatorilor în momentul în care apar modificari în sistem (modificare stare dispozitiv)
  + Notificarea utilizatorilor în cazul în care diferite evenimente definite de acesta sunt înregistrate (Exemple: strain în perimetrul locuinței, incendiu)
  + Automatizarea acțiunilor în funcție de utilizatorul recunoscut
* Cerințe privind interactivitatea:
  + Vizualizarea tuturor notificărilor generate de sistem în interfața web
  + Interacțiune cu dispozitivele
  + Vizualizare in timp real a inregistrarilor video din sistem
  + Utilizatorul va primi notificari în interfața web cât si prin SMS atunci când apar modificari în sistem
* Cerințe de securitate:
  + Confidențialitatea datelor utilizatorilor
  + Rezistență față de atacurile frecvente asupra aplicațiilor web
  + Garantarea faptului că fiecare utilizator are acces doar la dispozitivele sale

### 2.2.2 Arhitectura generala

Principalul obiectiv al acestei aplicații este acela de a automatiza interacțiunea dintre utilizatori și dispozitive prin recunoașterea lor automată ceea ce face posibilă redarea preferințelor într-un mod individual. Acest lucru este posibil prin intermediul unui sistem de supraveghere video care să recunoască în timp real prezența unei presoane și să transmită dispozitivelor conectate starea preferată de utilizator. Utilizatorul trebuie să își inregistreze în aplicație camerele video și dispozitivele cu care doreste să interacționeze precum și toate persoanele care pot interacționa cu ele. Odată realizate aceste aspecte, acestia își vor putea crea diferite scenarii de utilizare prin definirea de reguli. Mai jos puteți vedea o arhitectură generală.

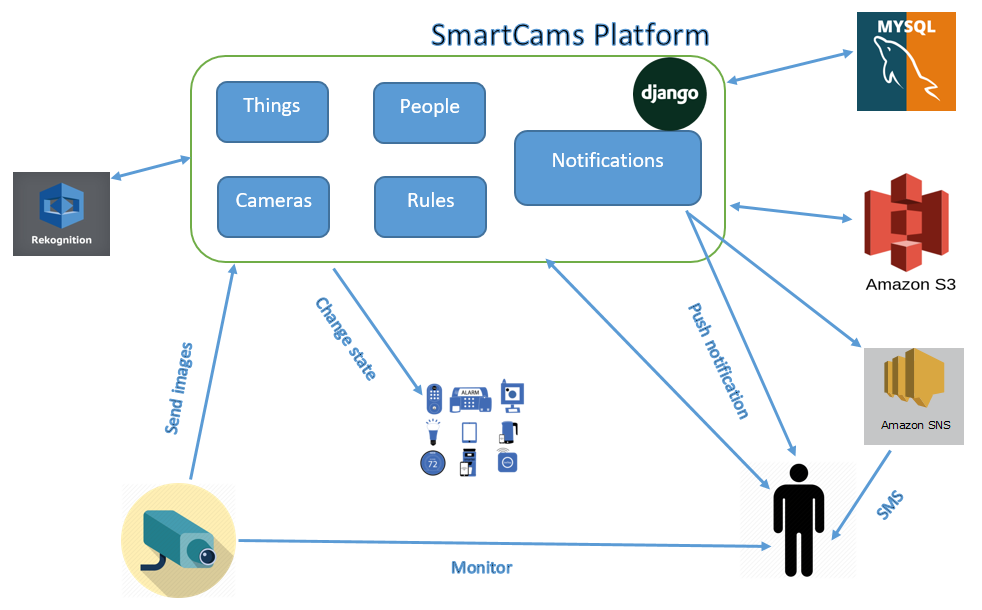


Figura 2. Arhitectura generală

### 2.2.3 Cameras

Acest modul joacă un rol cheie în cadrul aplicației concentrându-se pe recunoasterea utilizatorilor sau a diferitelor evenimente. Utilizatorul își înregistrează în aplicație camerele de supraveghere disponibile. Odată realizat acest lucru, acesta are posibilitatea de a vizualiza in timp real inregistrările acestora. Deoarece nu am avut acces la un sistem de camere de supraveghere, am utilizat camera web a dispozitivului folosit pentru accesarea aplicatiei (laptop). De asemenea, orice camera web auxiliară conectată la acesta este recunoscută de aplicație in mod automat pentru ca autilizatorul să o poată înregistra și folosi în contextul aplicației. Mai jos se poate vedea o diagrama de secventă ce redă modul de inregistrare în aplicație a unei camere noi.

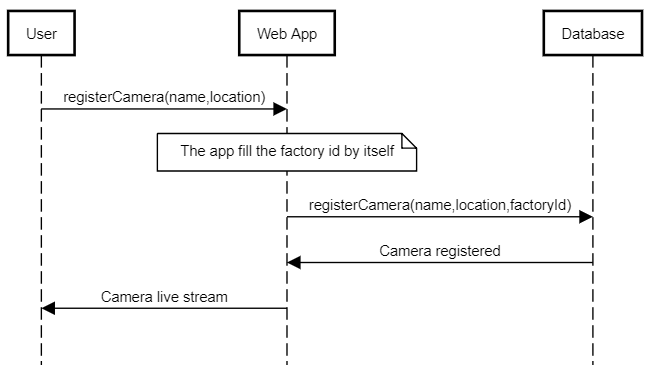


Figura 3. Adăugarea unei camere video în sistem

Pentru ca aplicația să recunoască prezența unei persoane și să o identifice, aceasta procesează imagini furnizate de camerele de supraveghere. Acest lucru se întâmplă doar în cazul în care se detectează miscare. Pentru a simula un senzor de miscare, pentru fiecare cameră din sistem se calculează în mod periodic, diferența dintre ultima imagine capturată și imaginea actuala. Această diferentă este compusă din suma tuturor diferențelor ditre pixelii aflați pe aceeasi poziție în imagine. În cazul în care diferența este mai mare de 10%, putem considera că a aparut o schimbare în sistem și trimitem aplicației imaginea spre a fi procesată. Marja de eroare de 10% a fost luată in calcul pentru cazurile in care apar mici modificari în zona monitorizată fără o importanță majoră (e.g modificarea luminozității din cameră în funcție de momentul zilei).

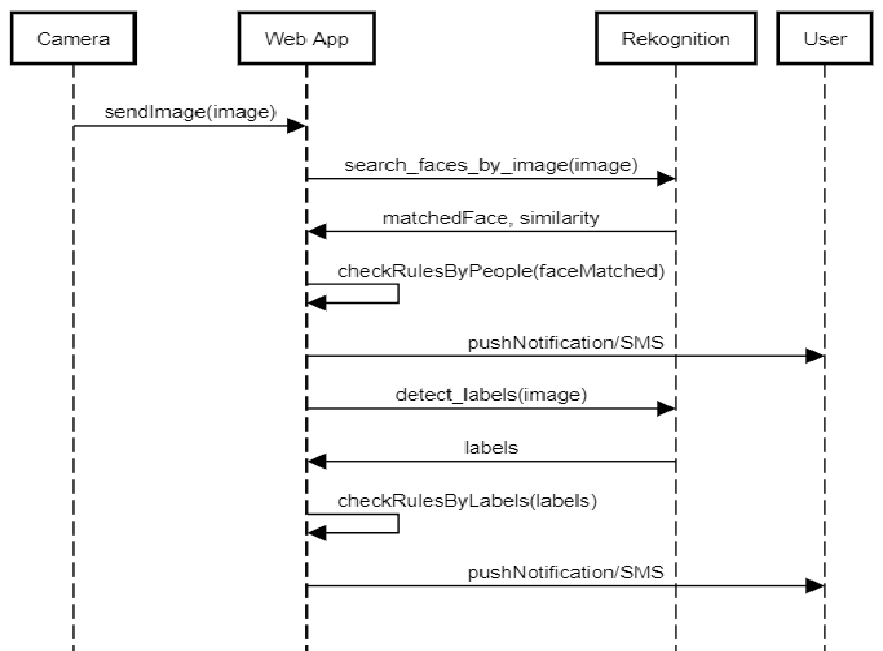
 Pentru recunoasterea utilizatorului, aplicația folosește serviciul AWS, Amazon Rekognition, care preia imaginea și compară fețele identificate cu cele stocate în colecția de cunoscuți a utilizatorului. Pentru a avea o acuratețe cât mai mare, o regulă este activată doar daca serviciul identifică o similaritate de minim 90%. Pe lângă recunoasterea faciala, aplicația foloseste funcția de recunoastere a etichetelor pentru a identifica diferite scenarii sau activități definite de utilizator. Diagrama de secvență următoare redă pașii prin care este recunoscută o regulă plecând de la o imagine a unei camere de supraveghere.

Figura 3. Adăugarea unei camere video în sistem

### 2.2.4 People

Acest modul se ocupă de înregistrarea persoanelor cunoscute utilizatorului. Acestea pot fi membri ai familiei, prieteni sau cunoscuți care pot interacționa cu dispozitvele înregistrate în sistem sau utilizatorul doreste sa fie recunoscute de camerele de supraveghere cu scopul de a recunoaste diferite evenimente. Pentru ca acest lucru să fie posibil, fiecare profil al unei persoane trebuie sa fie insoțit de o poza a acestuia. Acestea sunt stocate folosind serviciul AWS S3 urmând ca după aceea să fie inregistrate într-o colecție privată a utilizatorului folosită de AWS Rekognition pentru recunoasterea facială de fiecare dată când se procesează o imagine furnizată de camere. Procesul de adăugare a unui cunoscut este redat mai jos printr-o diagrama de secvență.

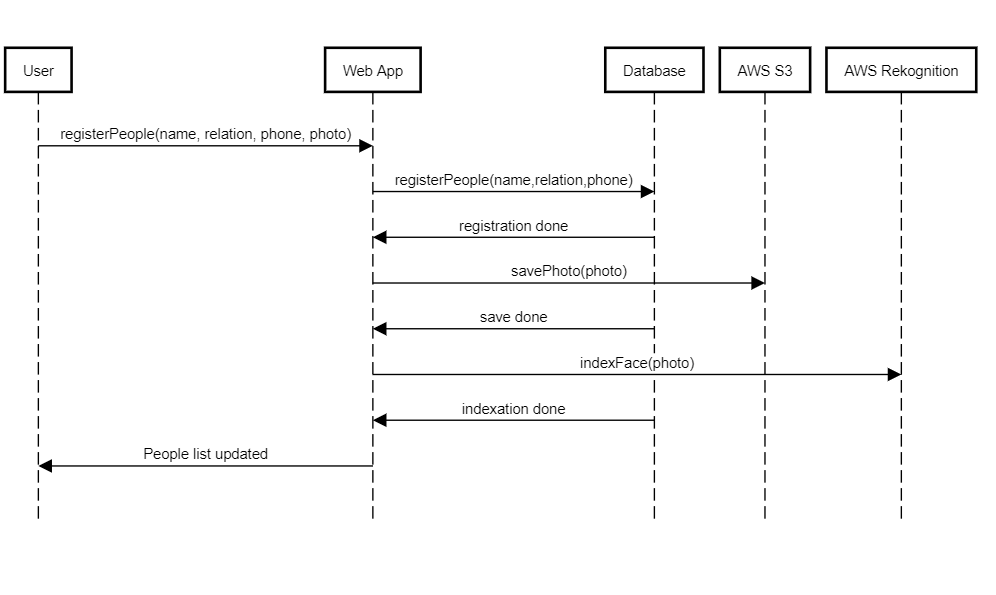


Figura 4. Adăugarea unui cunoscut în sistem

### 2.2.5 Things

Prin intermediul acestui modul, utilizatorul își poate înregistra în sistem toate dispozitivele inteligente cu care dorește să interacționeze. Pentru că nu am avut acces la dispozitive reale și la o infrastructură necesară controlului acestora de la distanță, am simulat existența lor doar cu baza de date. Pentru a simplifica lucrurile, am ales ca toate dispozitivele să fie reprezentate de 2 stări posibile, *închis*  respectiv *deschis.* Motivul pentru care am ales aceste 2 stări este caracterul lor general care le permite să se găsească la cât mai multe dispozitive (e.g usă, geam, aer condiționat, energie electrică, gaz, televizor etc)*.* Astfel putem simula o gamă largă de dispozitive și putem reda cât mai multe scenarii de utilizare a aplicației. Faptul că dispozitivele sunt simulate nu afectează în mod special aplicația deoarece principiul de funcționare și interacțiunea dintre ele este ceea ce vrem să scoatem în evidență.

Pentru înregistrarea unui nou dispozitiv, utilizatorul trebuie să introducă doar numele și locația acestora, informații ce au doar scop informativ pentru acesta. Odată realizat acest lucru, noul dispozitiv apare în lista dispozitivelor înregistrate, de unde utilizatorul îi poate modifica starea în orice moment. Am redat mai jos procesul de adăugare a unui dispozitiv folosind o diagramă de secvență.

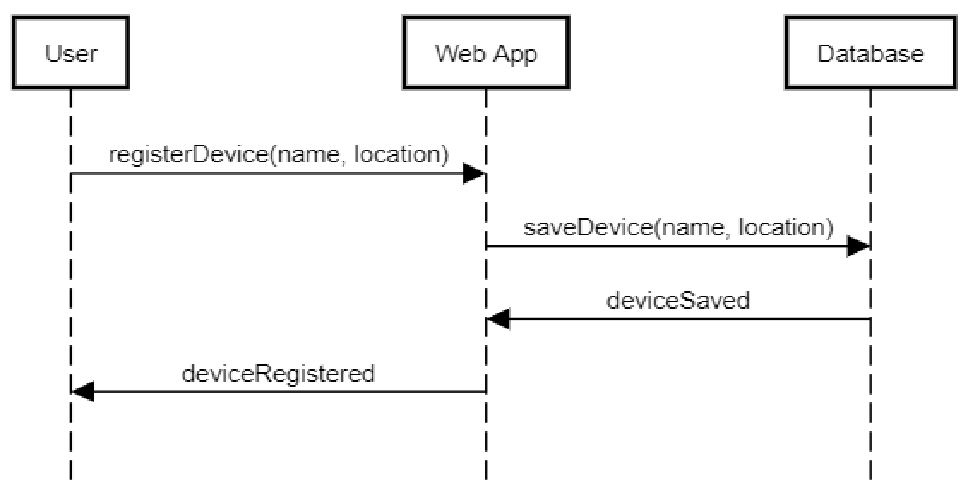


Figura 5. Adăugarea unui dispozitiv în sistem

### 2.2.6 Rules

Modulul de reguli este responsabil de automatizarea interacțiunii dintre utilizator și rețeaua de dispozitive. Este modulul care conectează celelalte 3 module, *Cameras*, *People* și *Things.* Principiul de baza este următorul: când o anumită cameră recunoaște o anumită persoană, sistemul va modifica starea unui dispozitiv. Pentru ca regulile să acopere cât mai multe scenarii posibile și pentru ca utilizatorul să poată defini evenimente cât mai specifice, se pot seta parametri auxiliari precum anumite zile in care să fie validă regula, un interval orar, diferite obiecte/actiuni din jur care să fie recunoscute de camerele de supraveghere precum și o listă de persoane care să fie notificate prin SMS la activarea regulii. Următoarea diagramă de secventă redă procesul de adăugare a unei noi reguli.

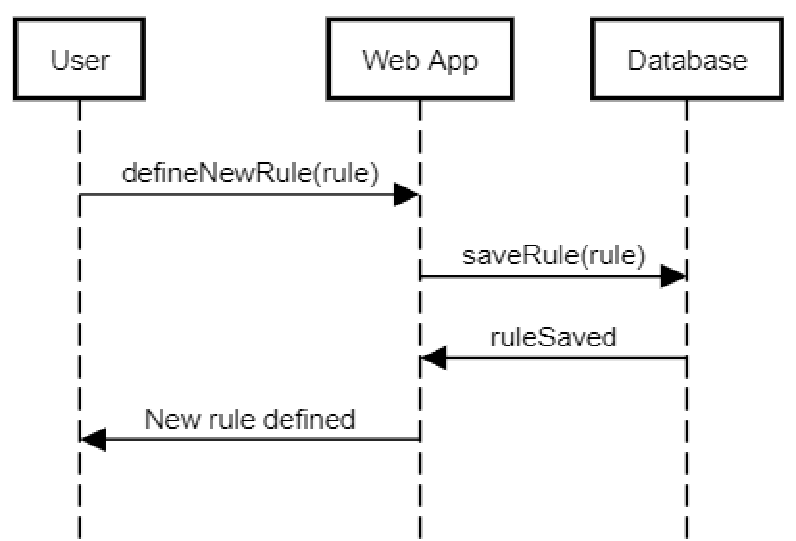


Figura 6. Adăugarea unei reguli în sistem

### 2.2.7 Notifications

Modulul de notificări este cel care îl ține la curent pe utilizator cu privire la toate modificările din sistem. În cadrul sistemului am optat pentru 2 modalități de notificare, *push notification* și SMS. Am ales *push notification* deoarece este o modalitate simplă și foarte practică, implementată la nivelul browser-ului. Avantajul acesteia este că aplicația poate trimite notificări chiar daca utilizatorulnu este active pe aplicație, notificarea fiind trimisă la nivelul browser-ului. Pentru cazurile mai extreme, când utilizatorul consider că un anumit eveniment este mai important, acesta poate opta pentru notificarea prin SMS atât a lui cât și a unor persoane cunoscute. Acest lucru se realizează prin intermediul serviciului AWS, Amazon SNS.

O notificare este generată atunci când utilizatorul modifică starea unui dispozitiv din aplicație sau atunci când o regulă este activată. Am descries, prin intermediul diagramei de secvențe următoare, primul scenario.

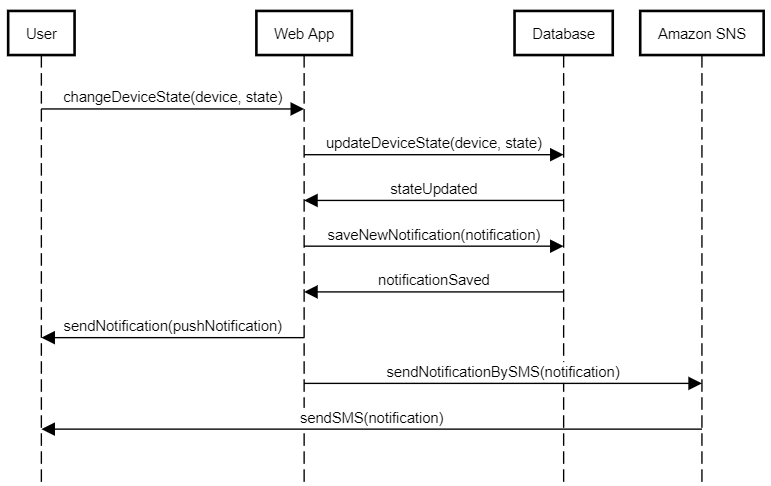


Figura 6. Adăugarea unei reguli în sistem

### 2.2.8 Stocarea persistentă a datelor

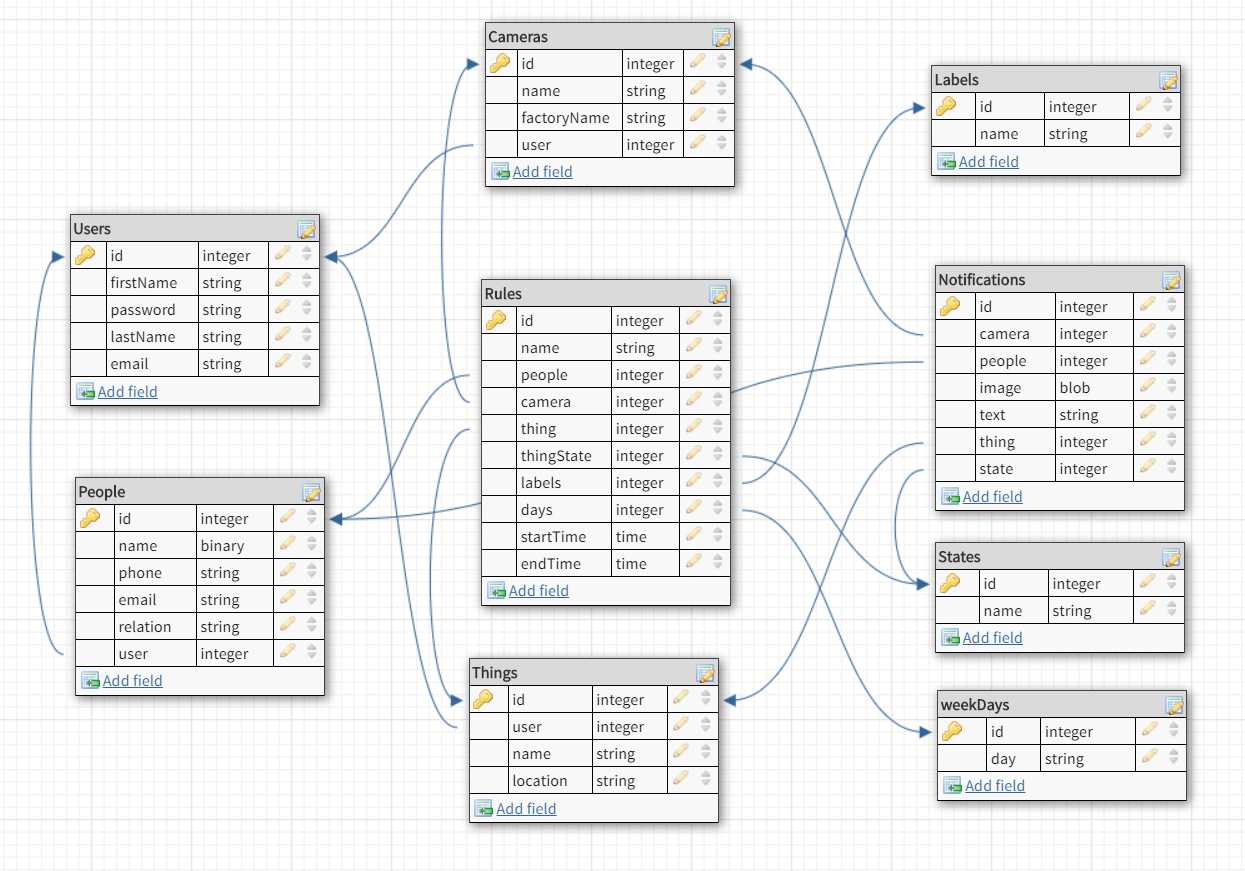
 Pentru stocarea datelor am ales să folosim un sistem de management al datelor relaționale: MySQL. Acest lucru este justificat prin simplul fapt că datele sunt structurate. Alegerea bazei de date MySQL nu este la întâmplare. MySQL este usor de integrat cu moajoritatee *framework-*urilor. Pe lânga asta, MySQL dispune de preformanțe foarte bune. Este recunoscută pentru nivelul ridicat de securitate pe care îl oferă împreună cu gradul de scalabilitate greu de egalat.[15] Comunicarea cu baza de date este realizată prin intermediul ORM-ului (*object-relational mapping)* oferit de *framework-*ul Django, care facilitează interacțiunea și rezolvă unele probleme de securitate (*sql injection)*.Mai jos se poate observa schema bazei de date:

Figura 7. Schema bazei de date

După cum se poate observa din figura de mai sus, baza de date este compusă din 9 tabele:

* Tabela ***Users***: conține informațiile legate de utilizatori, numele, prenumele, parola și adresa de email.
* Tabela ***People***: conține informații legate de persoanele cunoscute de utilizator, care pot fi recunoscute de camerele de supraveghere – numele, numarul de telefon, relatia cu utilizatorul si adresa de email.
* Tabela ***Cameras***: pastrează detaliile camerelor de supraveghere înregistrate precum modelul acesteia și numele sugestiv pentru utilizator.
* Tabela ***Things***: pastrează detaliile legate de dispozitivele inteligente înregistrate în platforma, în special numele și locația acestora.
* Tabela ***States***: conține toate stările posibile pe care le poate avea un dispozitiv.
* Tabela ***Labels***: memorează toate etichetele recunoscute de camerele video, stocând numele acestora.
* Tabela ***Rules:*** reține toate regulile de automatizare definite de utilizator. Fiecare înregistrare conține referintă la persoana care va fi recunoscută, camera video care va realiza recunoasterea faciala, obiectul care își va modifica starea etc.
* Tabela ***Notifications***: conține toate notificările generate de sistem în urma recunoasterii unei reguli. Înregistările conțin referințe la obiectul care și-a modificat starea precum și noua stare, persoana care a generat modificarea și imaginea înregistrată.
* Tabela ***WeekDays***: conține zilele săptămânii, folosite pentru crearea de reguli valide doar în anumite zile.

Pentru stocarea imaginilor, am ales un serviciu de stocare în *cloud,* Amazon S3, având în vedere că este integrat cu serviciul de procesare al imaginilor pe care îl folosim, Amazon Rekognition, dar si pentru și securitatea, durabilitatea si scalabilitatea oferită.

# 3 Tehnologii folosite

## 3.1 Django

Django este un *framework* web de nivel înalt care utilizează Python ca si limbaj de programare. Este un *framework* gratuit, disponibil in regim open-source, destinat dezvoltării de aplicații web rapid și eficient. Rezolvă foarte multe probleme comune legate de securitate ceea ce îi permite programatorului să se axeze pe scrierea aplicației fără prea multe complicații. Are la baza modelul arhitectural MVC (*model-view-controller*) venind totuși o implementare proprie. Deoarece partea de *controller* este realizată de *framework* iar majoritatea logicii se întâmplă in modele, *template*-uri și *view*-uri, Django este adesea considerat ca fiind un *framework* bazat pe MTV (*model-template-view).*[4]

Am ales folosirea *framework-*ului Django atât pe partea de server cat si pe partea de client din mai multe motive:

* **Python** - unul dintre cele mai ușor de învățat limbaje de programare, clar, usor de citit și rapid de scris
* **Securitate** - face managementul problemelor de securitate comune existente (e.g *SQL injection*, *cross-site request forgery, cross-site scripting*)
* **Scalabilitate** **și** **stabilitate** - se comportă foarte bine în cazul unui număr mare de utilizatori, fiind regăsit în aplicații mari de pe piață (e.g Instagram, BitBucket, Pinterest)[5]
* **Dezvoltare rapidă** – conține o multitudine de pachete ce servesc componete comune oricarei aplicații web reducând efortul dezvoltatorului: managementul sesiunilor, autentificarea utilizatorilor, interfață administrator etc.[6]

## 3.2 MySQL

MySQL este un sistem de gestiune al bazelor de date relaționalale *open-source*, ușor de integrat cu majoritatea *framework-*urilor. Am ales folosirea unei baze de date relaționale deoarece datele utilizatorilor sunt structurate iar pentru dispozitivele inteligente am folosit o reprezentare a datelor comună.

## 3.3 Amazon Web Sevices

*Amazon Web Services,* prescurtat AWS, este o platforma de servicii *Cloud* dezvoltată de compania Amazon. Aceasta furnizează o multitudine de servicii destinate dezvoltarii de aplicații web printre care putere computațională, stocare de date, management, analiza etc.

Principalele caracteristici care ne-au motivat să alegem acest furnizor sunt flexibilitatea, securitatea ridicată și costul redus. Un alt mare avantaj demn de menționat este modalitatea de plata *pay-as-you-go*. Utilizatorii platformei AWS plătesc doar pentru serviciile pe care le folosesc și doar pentru timpul în care le folosesc fără a implica anumite contracte pe termen lung. Este foarte asemanator cu modalitatea de plată a utilităților, platești doar pentru cât consumi[8]

### 3.3.1 Simple Storage Service

*Simple Storage Service*, sau S3 cum mai este prescurtat, este un serviciu de stocare a datelor sub orice format, simplu și sigur. Garantează o durabilitatea a informațiilor de până la 99.999999999% prin distribuirea lor în minim 3 zone geografice diferite . Pe partea de securitate, oferă 3 forme de criptatre a datelor iar accesul la date se realizează prin definirea unor politici de acces asupra *bucket-*ului. Asta garantează un control clar al utilizatorilor care pot accesa resursele.[9] Toate acestea dar și costul mic de utilizare și faptul că este integrat cu Amazon Rekognition, serviciul folosit pentru procesarea de imagini, ne-au determinat să alegem acest serviciu pentru stocarea fotografiilor utilizatorilor și a celor furnizate de camerele de supraveghere.

### 3.3.2 Amazon Rekognition

Amazon Rekognition este un serviciu AWS destinat procesării de imagini și conținut video, simplu și ușor de folosit. Tot ceea ce trebuie sa faci este să îi oferi o imagine sau un video iar el va detecta obiecte, text, activități sau oameni în funcție de necesarul tău ca utilizator.[10]

Ceea ce ne-a motivat să îl folosim însă este funcția de recunoaștere faciala. Amazon Rekognition poate recunoaste, analiza și compara fețe cu o acuratețe foarte mare atât din poze cât și din video-uri. Pe lângă recunoasterea faciala, Amazon Rekognition pune la dispoziție o funcție ce returnează un set de etichete identificate într-o imagine. Etichetele pot fi reprezentate de obiecte, acțiuni, concepte sau activități. Este integrat cu serviciul de stocare S3 ceea ce îl face și mai usor de folosit. Serviciul este antrenat în mod constant pe noi date pentru ași îmbunătăți acuratețea și a oferi rezultate cât mai exacte.[10] Conform studiilor de pe piața, acest serviciu ofera cele mai bune rezultate în materie de analiză faciala la un preț foarte bun relativ la ceilați competitori.[11] Noi îl vom folosi pentru a procesa imaginile furnizate de camerele de supraveghere în scopul recunoasterii persoanelor care intră în contact cu dispozitivele inteligente pentru a le reda prin intermediul acestora preferințele lor. De asemenea, vom analiza imaginile pentru a recunoaste diferite scene importante pentru utilizator sau evenimente care pot pune în pericol siguranța lui sau a mediului supravegheat (e.g hoți).

### 3.3.3 Simple Notification Service

Simple Notification Service (SNS) este un serviciu flexibil destinat trimterii/primirii de notificari de tipul *push notifications* sau SMS. Este o soluție foarte bună pentru aplicații decuplate sub forma de microservicii, sisteme distribuite sau aplicații *serverless*. Pune la dispoziție o multitudine de subscripții permițându-ți să trimiți mesaje unor *endoint-*uri HTTP sau altor servicii AWS precum Amazon EC2, Amazon S3 sau Amazon CloudWatch. Este o soluție simpla și eficientă din punct de vedere al costului pentru a trimite notificari dispozitivelor de tipul iOS, Android, Fire OS sau Windows.[12]

Noi am ales acest serviciu pentru trimiterea de SMS-uri utilizatorilor în special pentru cazurile în care sistemul identifică o situație periculoasă în zona monitorizată. Astfel, utilizatorul este notificat chiar daca nu foloseste aplicația sau nu este conectat la internet. Prin această metodă, sistemul îi poate notifica și pe apropiații acestuia ceea ce mareste șansa ca mesajul să fie citit de cineva în caz că utilizatorul primar nu este disponibil. În cazul unor situații periculoase, acest lucru este vital.

## 3.4 Push Notification

*Push notification* este un mesaj care apare pe dispozitivele mobile sau browsere în urma trimiterii lor de către o aplicație. Acestea din urmă le pot trimite oricând, fără a fi necesar ca utilizatorii să folosească aplicatiile în momentul respectiv. Pentru ca utilizatorii să poată primi acest tip de notificare, ei trebuie să accepte ca browser-ul să primească *push notifications*  de la aplicație. Odată ce permisiunile au fost oferite, de fiecare dată când aplicația trimite o notificare, ea va fi interceptată de browser.[13] Dacă browserul nu este deschis, notificarea va fi stocată și afișată la următoarea pornire.

Am decis să folosim acest tip de notificări deoarece utilizatorul poate primi mesaje chiar și atunci când nu interacționează cu aplicația la momentul trimiterii acesteia. Astfel, utilizatorul este ținut la curent cu toate evenimentele din sistem în timp real.

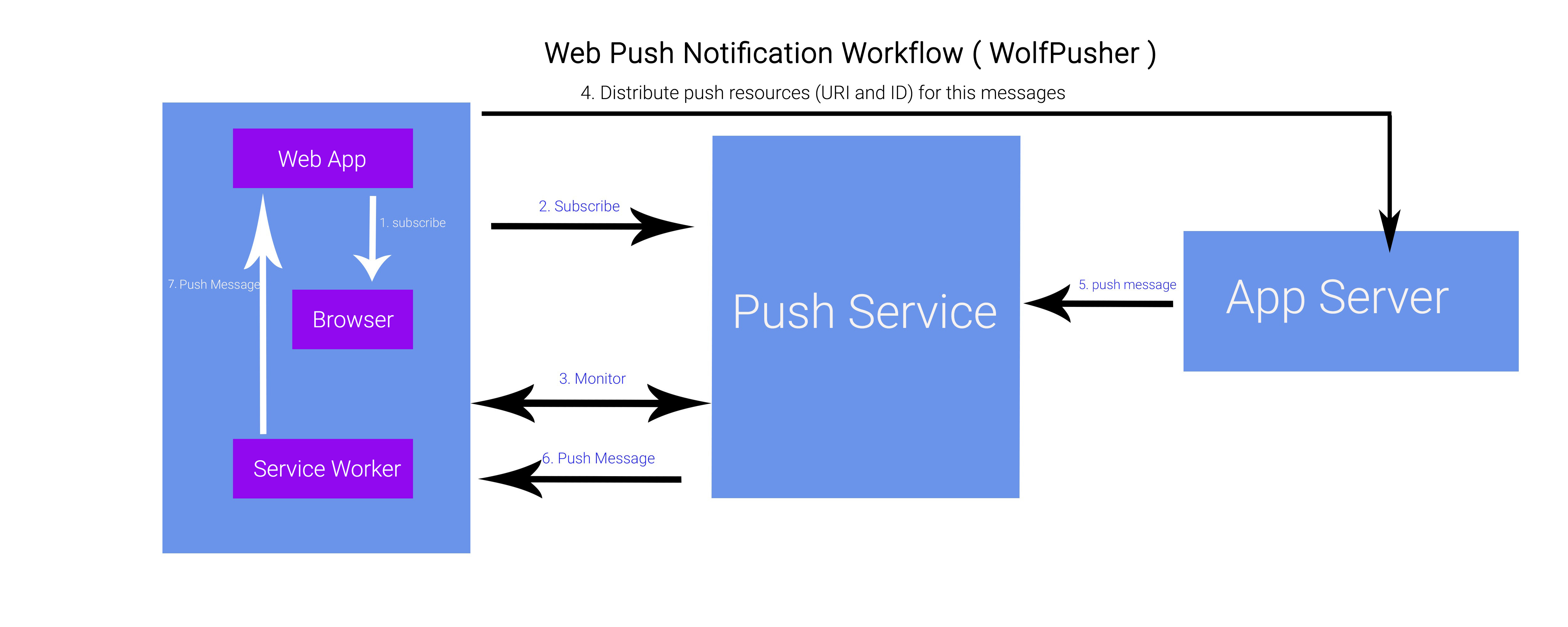


Figure 1 Modul de notificare a utilizatorilor folosind push notifications

https://i.stack.imgur.com/DzEDv.jpg

# 4.Implementare

Aplicația a fost dezvlotată folosind framework-ul Django atât pe partea de server cât si pe partea de client. Foloseste ca limbaj de programare Python ceea ce face ca partea de delopement să fie foarte rapidă și usor de înțeles. Pe partea de server s-a ales o arhitectură de tip REST ce permite o comunicare *stateless* cu clientul. Aceasta se realizează prin cereri HTTP prin metodele expuse de acest protocol (GET, POST, PUT, DELETE).

## 4.1 Resurse

Aplicația modelează 5 resurse principale și anume *Things, Cameras, People, Rules* și *Notifications,* reprezentate de rutele următoare:

* /things
* /cameras
* /people
* /rules
* /notifcations

Toate acțiunile asupra acestor rute necesită ca utilizatorul să fie autentificat, în caz contrar acesta fiind trimis automat pe pagina de autentificare/înregistrare. În continuare vom descrie actiunile posibile și rezultatul primit pentru fiecare resursă în parte.

Prin intermediul resursei *Things* se face managenentul dispozitivelor din cadrul aplicatiei. Metodele premise pentru această resursă sunt:

GET */things/index*

* Returnează o pagină ce conține formularul de înregistrare a unui dispozitiv nou precum și toate dispozitivele înregistrate de utilizator până în momentul respectiv.

POST /things/index

* Metodă folosită pentru înregistrarea unui nou dispozitiv. Aceasta trebuie să conțină în corpul cererii numele și locația dispozitivului.

POST */things/changeState*

* Metodă folosită pentru modificarea stării unui dispozitiv direct din aplicație. Corpul cererii trebuie să conțina ID-ul dispozitivulul și noua sa stare.

DELETE */things/delete*

* Prin intermediul acestei metode se sterge un dispozitiv înregistrat transmițând ca parametru ID-ul acestuia. Dacă dispozitivul există în baza de date, metoda va returna codul de stare 200 sau 404 (Not Found) în caz contrar.

Pentru managementul camerelor video din sistem, folosim resursa *Cameras* cu următoarele metode permise:

GET */cameras/index*

* Rezultatul acestei acțiuni este reprezentat de o pagină HTML care conține formularul pentru înregistrarea unei camere video precum și lista tuturor camerelor înregistrate.

POST */cameras/index*

* Metodă folosită pentru înregistrarea unei noi camere în sistem. Această metodă necesită ca parametri numele și locația camerei. În cazul în care acesti parametri nu sunt trimiși, se va returna codul de stare 400 (Bad request).

POST */cameras/index/process-picture*

* Acceastă metodă este folosită de fiecare data când o camera înregistrează miscare în zona monitorizată. În acest caz, se trimite ID-ul si o captură a camerei în cauză, returnându-se codul de stare 200.

DELETE */cameras/delete*

* Prin intermediul acestei metode se sterge o cameră video din colecție transmițând ca parametru ID-ul acesteia. Dacă există o camera cu ID-ul furnizat în baza de date, metoda va returna codul de stare 200, altfel codul returnat va fi sau 404 (Not Found).

Urmează metodele pentru resursa *People,* prin intermediul căreia se face managementul persoanelor cunoscute de utilizator ce pot interacționa cu dispozitivele inteligente înregistrate de acesta:

GET */people/index*

* Această metodă returnează pagina prin intermediul căreia se pot înregistra persone noi și se pot vizualiza persoanele înregistrate deja

POST */people/index*

* Această metodă este folosită pentru a înregistra o persoană nouă în sistem. Pentru ca acestă metodă să se realizeze cu succes (cod stare 200), este neesar ca în corpul cererii să fie trimise ca parametric numele persoanei, adresa de email, relatia cu utilizatorul, numarul de telefon și o poză.

DELETE */people/delete*

* Metodă ce sterge un utilizator înregistrat prin furnizarea ID-ului acesteia ca parametru în cadrul cererii.

Prin intermediul resursei *Rules* se face managementul regulilor de automatizare a interațiunii dintre utilizator și dispozitivele inteligente înregistrate în sistem. Actiunile premise asupra acestei resurse sunt:

GET */rules/index*

* Această metodă întoarce ca răspuns pagina prin intermediul căreia se pot crea noi reguli de recunoastere a persoanelor/evenimentelor și vizualizarea celor create deja.

POST */rules/index*

* Această metodă este folosită pentru a înregistra o regulă nouă în sistem. Pentru ca acestă metodă să se realizeze cu succes (cod stare 200) este necesară trimiterea următorilor parametrii: numele regulii, dispozitivul ce își va schimba starea, noua stare a acestuia, camera responsabilă de recunoasterea evenimentului și o persoană care să fie identificată. Optional se poate trimite un interval orar la care să fie valabilă regula, anumite zile din săptămînă sau o listă cu personane care să fie notificate în caz de activare a regulii.

DELETE */rules/delete*

* Metodă ce sterge o regulă creată de utilizator prin furnizarea ID-ului acesteia ca parametru în cadrul cererii.

## 4.2 Procesarea imaginilor

Pentru a verifica prezența unei persoane în zona de acțiune a unui dispozitiv, sistemul preia și procesează imagini de la camerele de supraveghere. În cazul în care este identificată o persoană cunoscută, sistemul parcurge toate regulile definite în care aceasta apare, încercând să valideze toate constrângerile.

Pentru recunoasterea utilizatorului, aplicația folosește serviciul AWS, Amazon Rekognition, care preia imaginea și compară fețele identificate cu cele stocate în colecția de cunoscuți a utilizatorului. Tinând cont că pentru fiecare imagine procesată Amazon Rekognițion percepe o taxă, am implementat o funcție care să identifice atunci când este miscare în zonele supravegheate de camerele video. Astfel, reducem considerabil numărul de imagini procesate și în consecință, costurile aplicației.

Senzorul de miscare implementat funcționează astfel: pentru fiecare cameră din sistem se calculează în mod periodic, diferența dintre ultima imagine capturată și imaginea actuala. Această diferentă este compusă din suma tuturor diferențelor ditre pixelii aflați pe aceeasi poziție în imagine. Codul arată astfel:

**function** *getImageDiff*(imageData, cameraId, width, height) {  
 **if**(!***previousImage***[cameraId]){  
 ***previousImage***[cameraId] = imageData;  
 **return** 0  
 }  
  
 **let** diff = 0;  
  
 **for** (**let** i = 0; i < imageData.**length** / 4; i++) {  
 diff += ***Math***.abs(imageData[4 \* i] - ***previousImage***[cameraId][4 \* i]) / 255;  
 diff += ***Math***.abs(imageData[4 \* i + 1] - ***previousImage***[cameraId][4 \* i + 1]) / 255;  
 diff += ***Math***.abs(imageData[4 \* i + 2] - ***previousImage***[cameraId][4 \* i + 2]) / 255;  
 }  
 ***previousImage***[cameraId] = imageData;  
  
 **return** 100 \* diff / (width \* height \* 3)  
}

În cazul în care diferența este mai mare de 10%, putem considera că a aparut o schimbare în zona monitorizată și trimitem aplicației imaginea spre a fi procesată. Odată ajunsă imaginea pe partea de server, aceasta este salvată provizoriu într-un *bucket* privat folosind serviciul Amazon S3. Pentru procesarea imaginii, folosim 2 funcții din cadrul serviciului Recognition, una pentru recunoasterea faciala și una pentru identificarea de etichete. Funcția de recunoastere faciala preia imaginea și încearcă să gasească fețe similare cu cele existente în colecția de fețe cunoscute a utilizatorulului. Aceasta este creată la început și de fiecare dată când utilizatorul înregistrează o persoana cunoscută în sistem, fața acesteia este adaugată în colecție. Adăugarea unei persoane noi în colecție se face astfel:

response = client.index\_faces(  
 CollectionId=User.objects.get(username=request.user).profile.facesCollection,  
 DetectionAttributes=[],  
 ExternalImageId=str(people.id),  
 Image={  
 **'S3Object'**: {  
 **'Bucket'**: settings.AWS\_STORAGE\_BUCKET\_NAME,  
 **'Name'**: settings.AWS\_PRIVATE\_MEDIA\_LOCATION + **'/'** + people.photo.name,  
 }  
 }  
)

Funcția de recunoastere facial are nevoie de 4 parametri: colecția în care se realizează cautarea, similaritatea minima dintre fețe, imaginea și numarul maxim de fete returnate. Pentru o acuratețe cât mai mare, am setat similaritatea minima egala cu 90%. Cum serviciul S3 este deja integrat cu Amazon Rekognition, pentru imagine putem furniza obiectul creat la salvarea provizorie. Apelul funcției de recunoastere facial arata astfel:

facesRequest = client.search\_faces\_by\_image(  
 CollectionId=User.objects.get(username=request.user).profile.facesCollection,  
 FaceMatchThreshold=90,  
 Image={  
 **'S3Object'**: {  
 **'Bucket'**: settings.AWS\_STORAGE\_BUCKET\_NAME,  
 **'Name'**: **'hello.png'**,  
 }  
 },  
 MaxFaces=5,  
)

În cazul în care se găseste similaritate cu o persoană din colecție, se vor parcurge toate regulile în care aceasta apare încercându-se validarea și celorlalte constrângeri.

Pasul următor este identificarea etichetelor din imagine. Această operație necesită doar imaginea și se relizează folosid următoarea funcție:

labelsRequest = client.detect\_labels(  
 Image={  
 **'S3Object'**: {  
 **'Bucket'**: settings.AWS\_STORAGE\_BUCKET\_NAME,  
 **'Name'**: **'hello.png'**,  
 }  
 },  
 MaxLabels = 10,  
 MinConfidence = 75  
)

Fiecare regulă care conține o etichetă identificată este verificată și validată împotriva celorlalte constrângeri.

În cazul în care nicio regulă nu este activată, sistemul nu suferă nicio modificare.

# 5. Manual de utilizare

Odata intrat pe pagina aplicației, utilizatorul este întâmpinat de formularul de înregistrare. În cazul în care acesta este deja înregistrat, se poate autentifica folosindu-se de formularul din colțul dreapta sus.

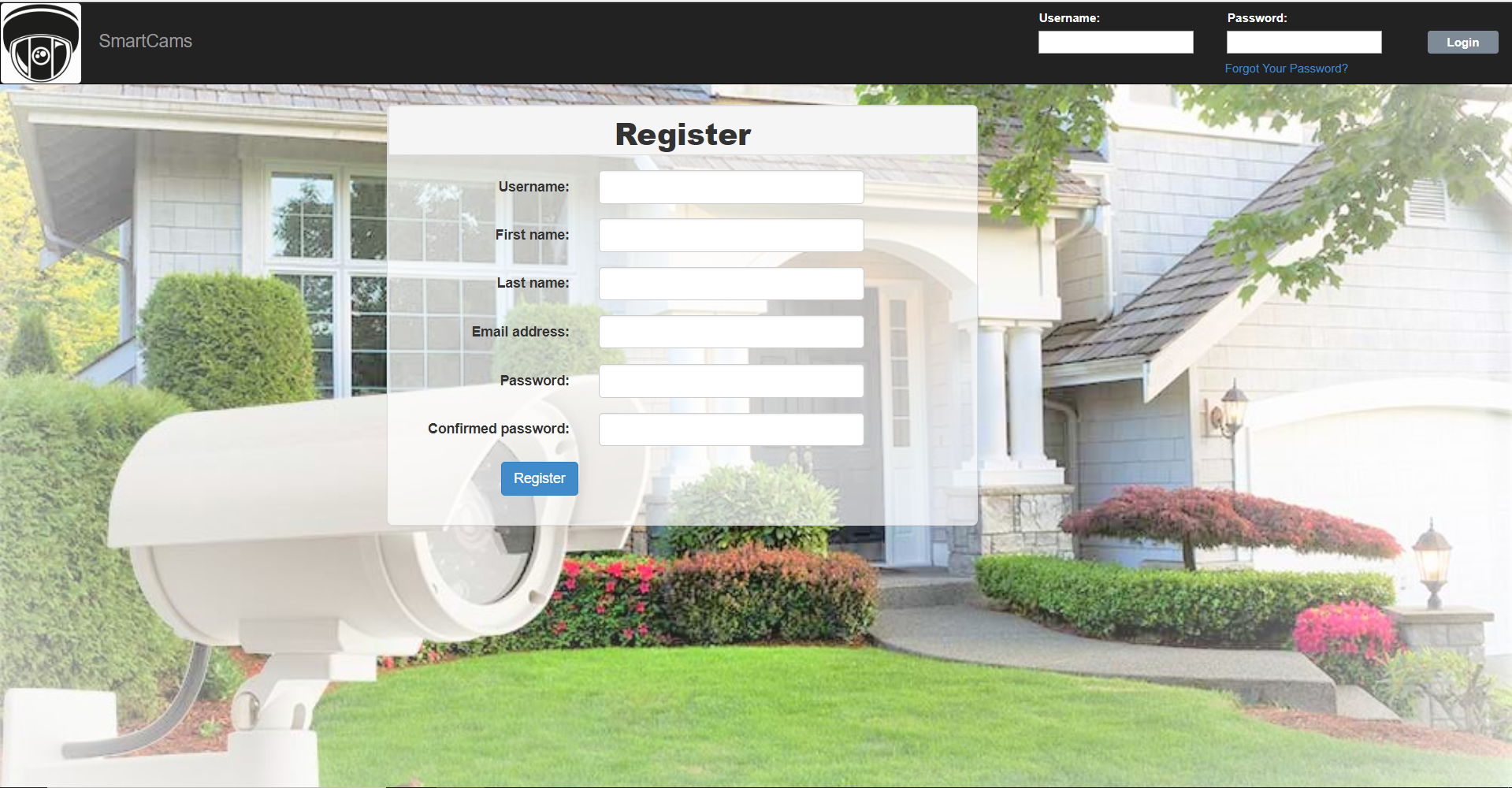


Figura 8. Pagina de start

Imediat dupa ce acesta s-a autentificat, este redirecționat pe pagina principală a aplicației unde poate vizualiza în timp real înregistrările camerelor de supraveghere din sistem sau poate adăuga alte camere noi. De aici utilizatorul poate accesa celelalte pagini ale aplicației folosindu-se de meniul de navigare.

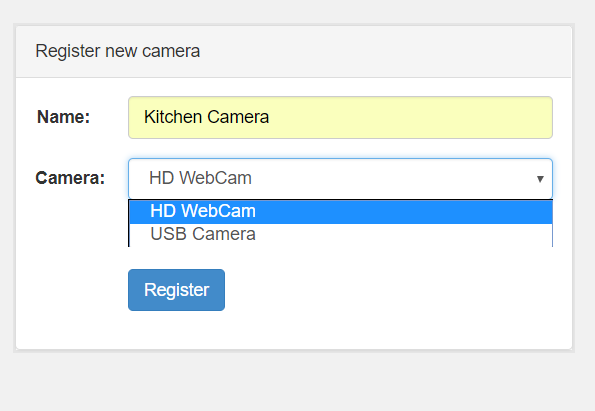


Figura 9. Adaugare camera noua in sistem

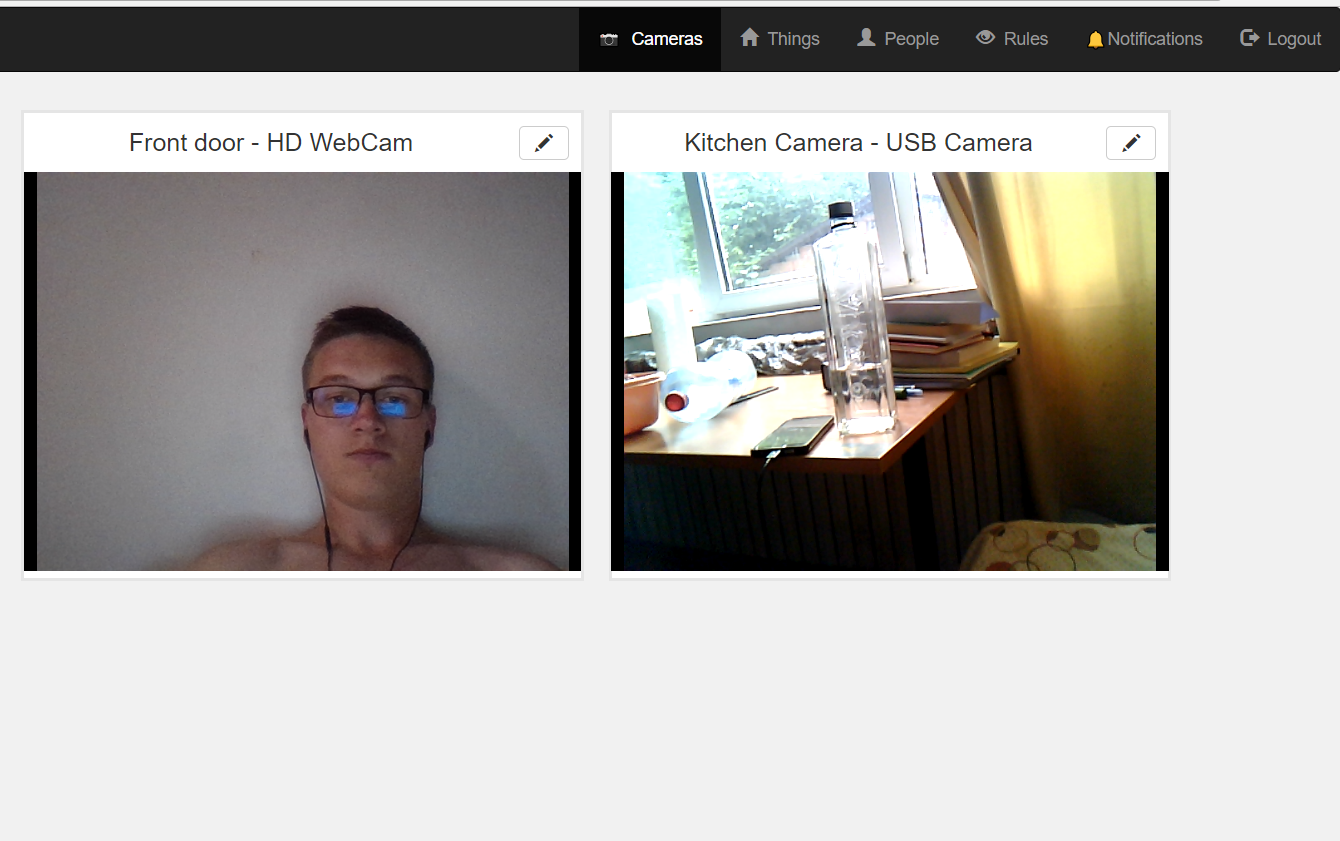


Figura 10. Live stream from cameras

În cadrul paginii *Things* utilizatorul are posibilitatea de a-și înregistra dispozitivele inteligente deținute și de a le vizualiza sau interacționa cu cele deja existante în aplicație. Pentru înregistrarea unui nou dispozitiv, utilizatorul trebuie să introducă numele și locația acestuia urmând ca starea lui să fie atribuită de aplicație după înregistrare.



Figura 11. Înregistrare dispozitiv

După înregistrare, starea unui dispozitiv poate fi modificată de utilizator prin intermediul interfeței sau direct de aplicație ca urmare a recunoasterii unei reguli. Interfața va fi actualizată de fiecare dată când un dispozitiv își schimbă starea urmând ca sistemul să îl notifice pe utilizator cu privire la schimbarea înregistrată.

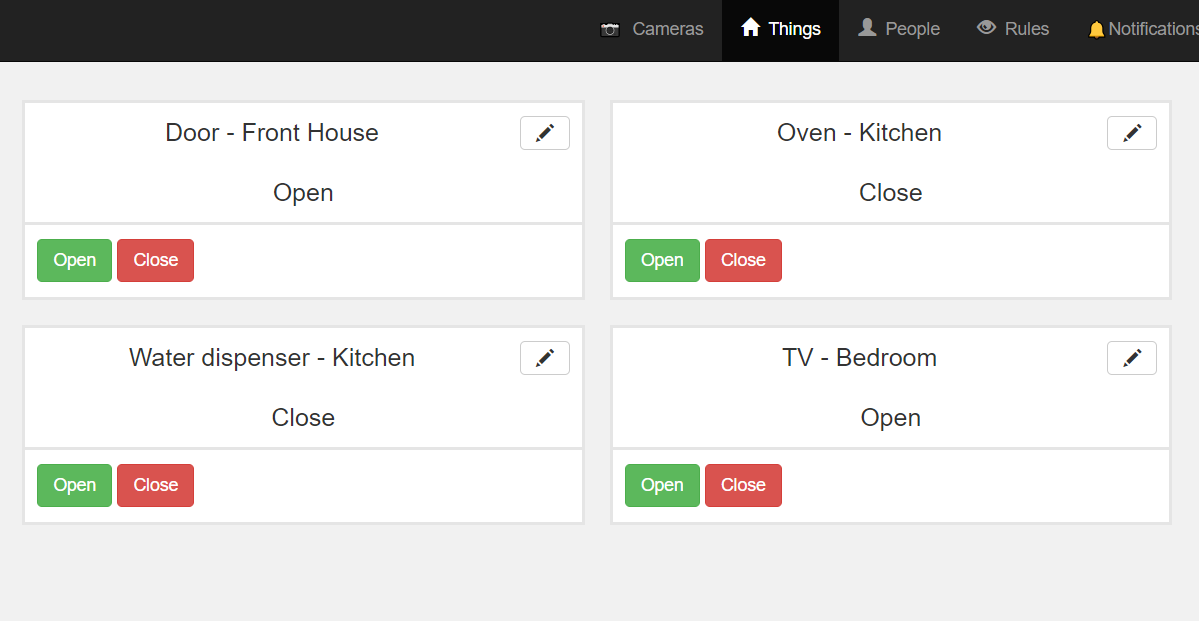


Figura 12. Dispozotive înregistrate

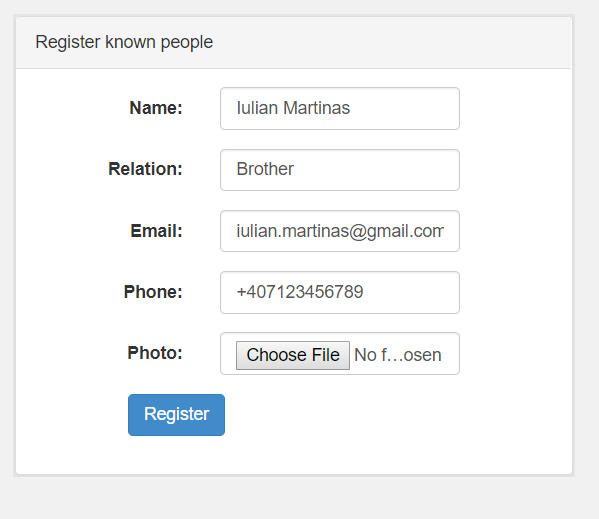
 Având în vedere că pot fi mai multe persoane care să interacționeze cu dispozitivele înregistrate, există posibilitatea înregistrării acestora prin intermediul paginii *People*.

Figura 13. Înregistrare persoană cunoscută

Utilizatorul va trebui să completeze un formular cu informațiile acestora precum nume, email, relatia sa cu acestea, numărul de telefon (pentru a putea fi notificate prin SMS în caz de nevoie) și nu în ultimul rând, o poză de profil, care va fi folosită de aplicație pentru a le recunoaste prezența. După înregistrare, acestea vor apărea în lista de persoane cunoscute, de unde utilizatorul le va putea modifica profilul.

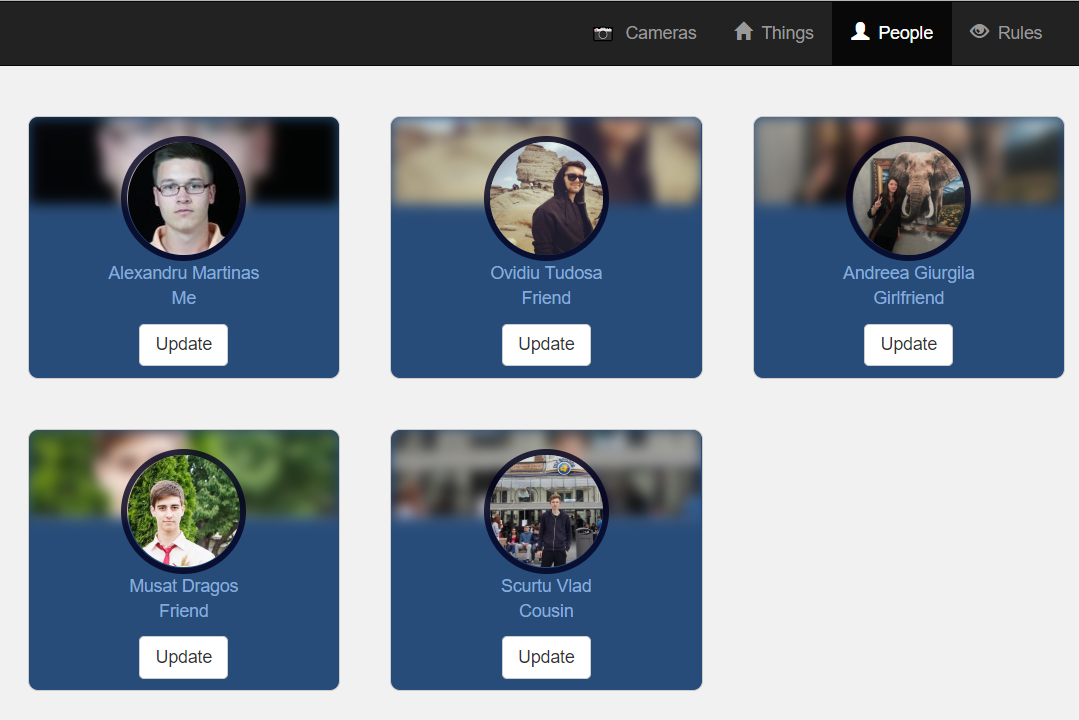


Figura 13. Lista de persoane cunoscute

Odată inregistrate camerele de supraveghere, dispozitivele și persoanele cunoscute, utilizatorul își poate defini diferite reguli prin intermediul paginii *Rules.* Regulile au ca scop principal automatizarea interacțiunii dintre utilizator și dispozitive prin recunoasterea automată a acestuia de catre camerele de supraveghere și trimiterea de semnale dispozitivelor astfel încât acestea să redea preferințele acestuia. Adăugarea unei noi reguli se face completând formularul din partea stângă a paginii *Rules*.

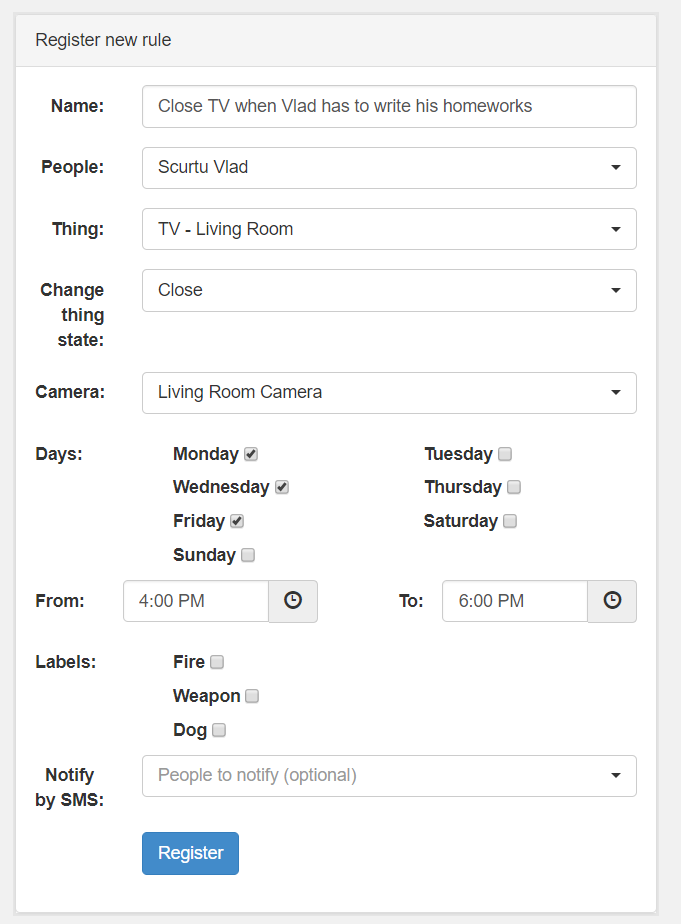


Figura 13. Formular adăugare regulă nouă

Toate regulile create pot fi vizualizate în lista de reguli. Fiecare regulă creată este verificată de fiecare dată când este identificată o față, în urma procesarii unei imagini de la camerele de supraveghere.

Pagina *Notifications* contine toate notificările generate de sistem. De fiecare dată când o regulă este activată, este generată o notificare sub formă de *push notification* sau SMS, în funcție de opțiunile utilizatorului. Acestea sunt stocate în baza de date și afisate sub formă de tabel ordonate descrescător după data generării.

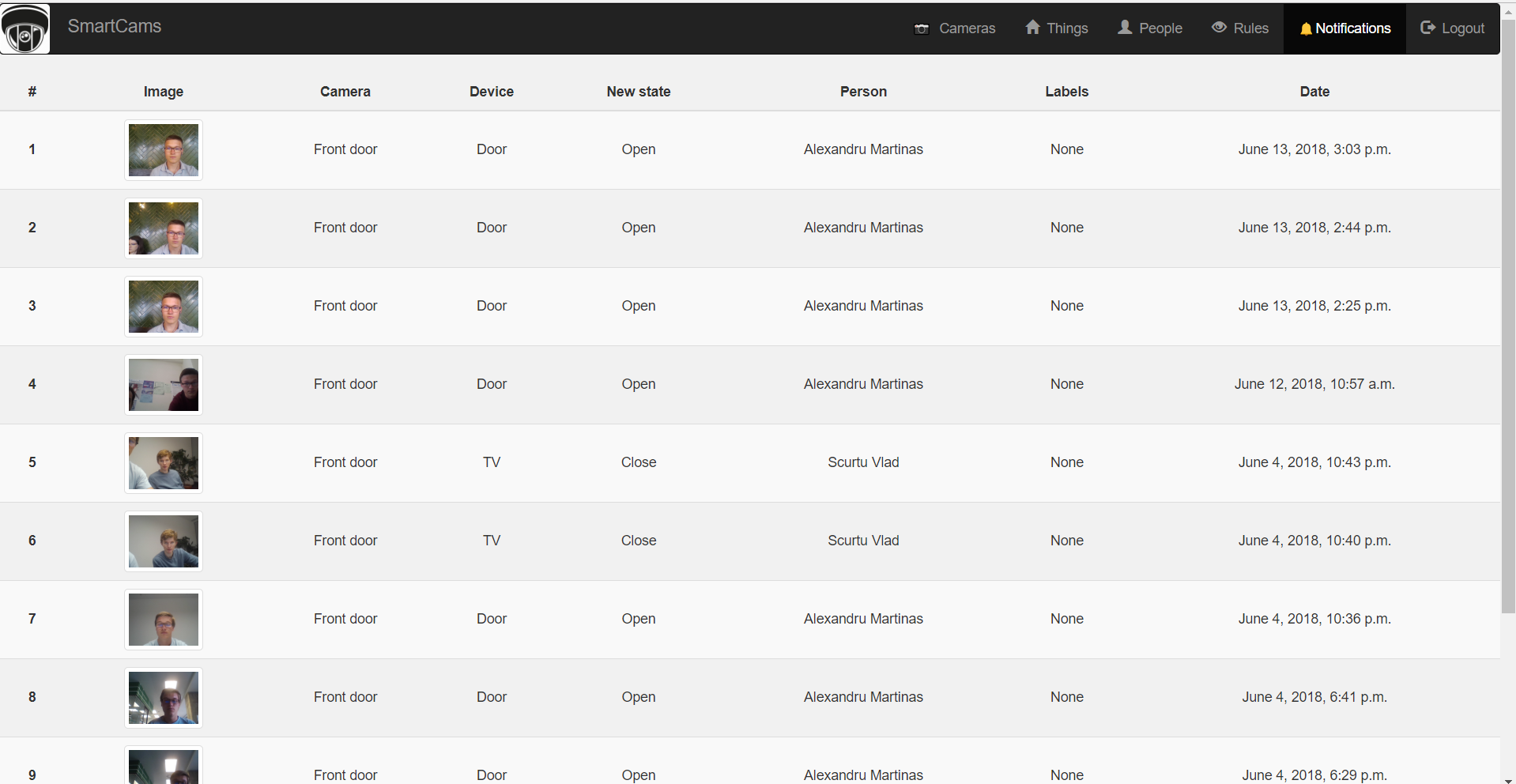
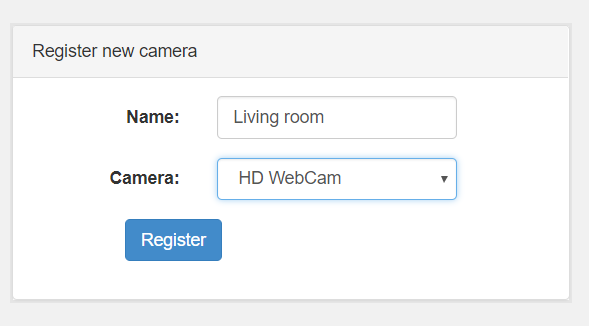
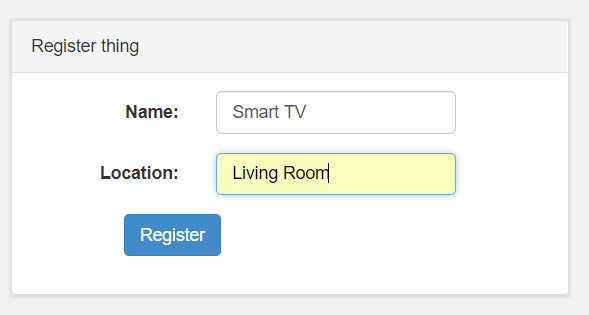


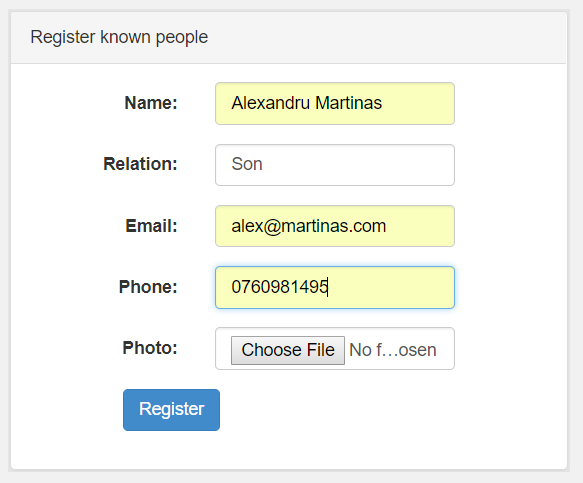
Figura 14. Notificări

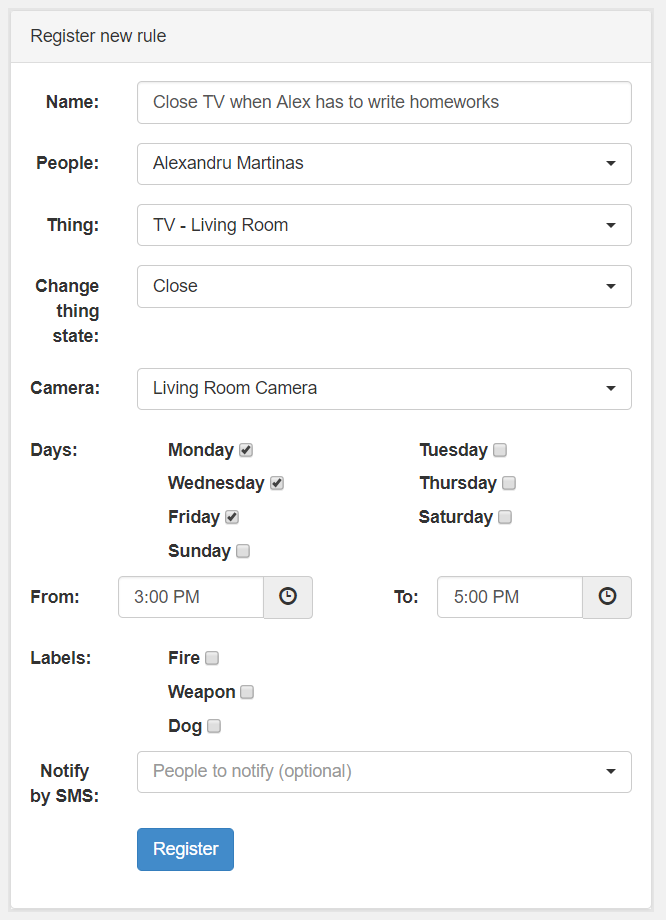
# 6. Scenarii de utilizare

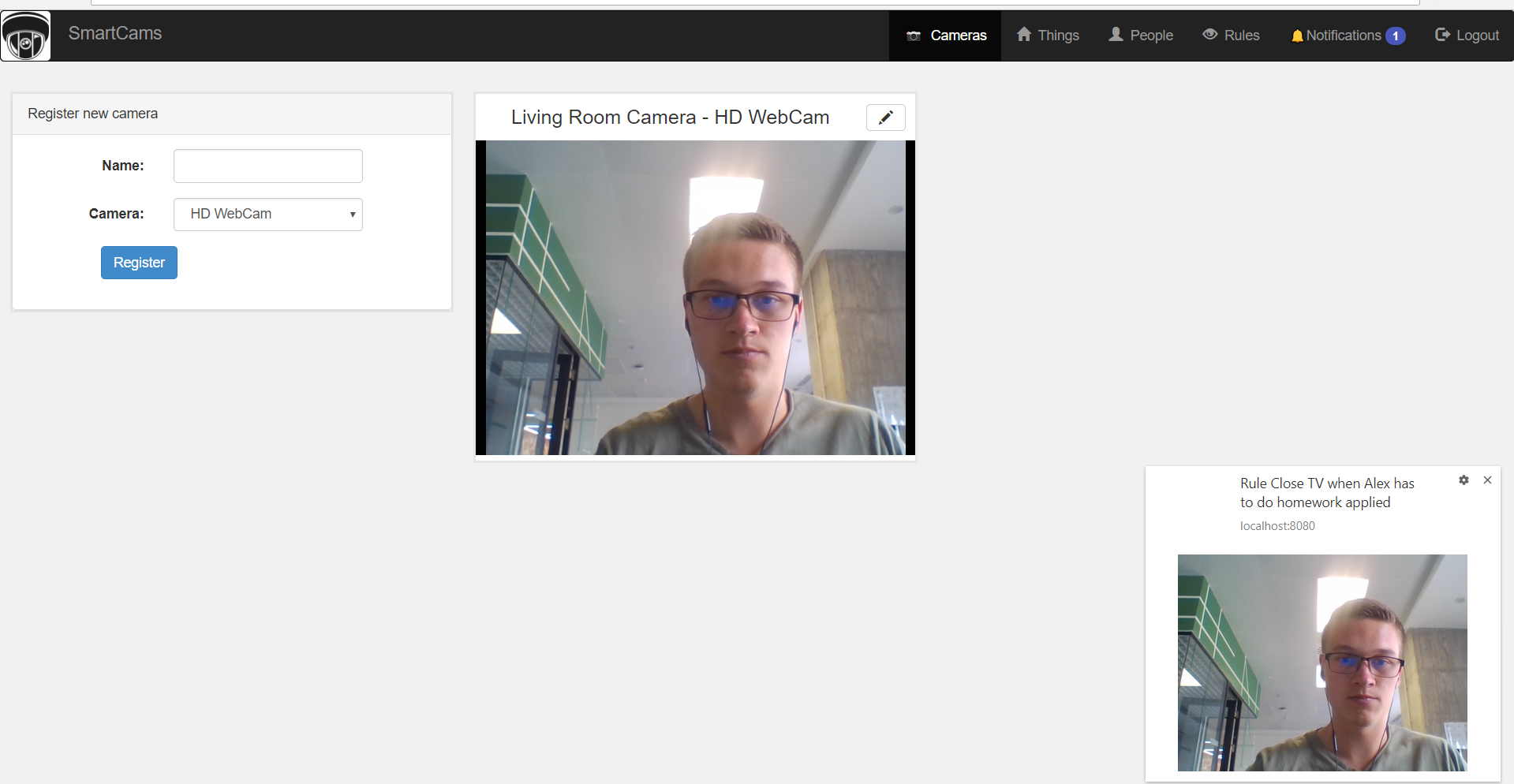
Vom prezenta în continuare un posibil scenariu de utilizare al acestei aplicații.

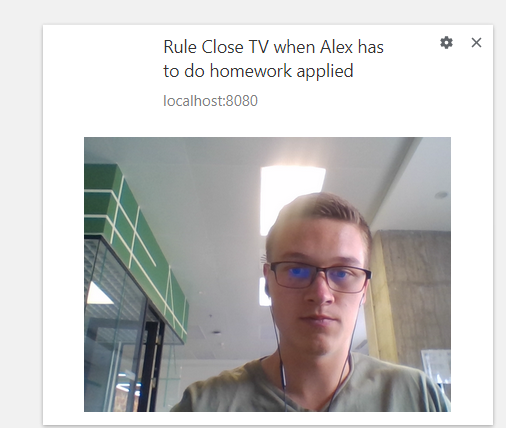
 Un proprietar de apartament a achiziționat o multime de dispozitive inteligente printre care și un televizor inteligent. El doreste să automatizeze controlul acestuia în funcție de membrul familiei. De exemplu, el vrea ca în timpul săptămânii, în intervalul orar 15-17 când fiul său trebuie să își facă temele, acesta să fie închis. Pentru ca acest lucru, proprietarul trebuie să înregistreze în sistem o cameră video care să poată monitoriza încăperea, televizorul ca și dispozitiv controlabil și pe fiul său ca persoană cunoscută.





Odată realizate acestea, proprietarul poate defini o regulă în care să redea acest comportament

Începând din momentul în care regula a fost înregistrată, de fiecare dată când camera înregistrată detectează miscare, va verifica daca se verifică toate condițiile din regulă. În caz de succes, proprietarul va fi notificat prin intermediul sistemulul de *Push Notification*.



Pe lângă asta, proprietarul își poate înregistra și alte dispozitive, poate vizualiza starea lor și defini alte reguli, în funcție de necesitățile lui, toate acestea prin intermediul unei singure aplicații.

# 7. Concluzii

„*SmartCams”* a fost concepută pentru a automatiza interacțiunea dintre utilizatori si dispozitivele inteligente. Dorind ca această interacțiune să fie personalizată, în funcție de utilizatorul retelei de dispozitive, am conceput o soluție care să realizeze acest lucru, prin identificarea unică a utilizatorilor. Pentru a realiza acest lucru, aplicația se foloseste de sistemul de supraveghere video a utilizatorului prin intermediul căruia poate recunoaste unic persoane si evenimente și adapta starea dispozitivelor conform situației.

Soluția finală este oferită utilizatorilor sub forma unei platforme web. Acest lucru face posibilă interacțiunea cu utilizatorul mult mai usoară, fără a depinde în mod special de locația acestuia. Pe lângă aceasta, sistemul de notificări pus la dispoziție oferă înformații cu privire la modificările din sistem, în timp real, chiar daca utilizatorul nu utilizează platforma la momentul respectiv. Acest lucru este posibil celor 2 modalități de notificare folosite, *Push Notification* și SMS.

Pentru automatizarea interacțiunii dintre utilizator și dispozitive, utilizatorul poate defini un set de reguli. Acesta poate alege ce actiuni se vor întreprinde la recunoasterea unei persoane sau a unui eveniment. De asemnea, poate seta un interval orar sau anumite zile în care regula să fie verificată. În cazul în care utilizatorul consideră că evenimentul recunoscut este de o importanță majoră, acesta poate seta o listă de persoane care să fie notificate prin SMS, sporind astfel șansa ca evenimentul să fie luat la cunostință de cineva.

Ca și direcție de dezvoltare pe viitor, am putea include niste algoritmi de învățare automată care să identifice diferite acțiuni frecvente ale utilizatorului și să îi propuna acestuia automatizarea actiunii prin crearea unei noi reguli. De asemenea, am putea include un factor suplimentar de autentificare care să sporească securitatea în cazul unor reguli vulnerabile. De exemplu, în cazul în care utilizatorul doreste deschiderea usii de la intrarea în locuință la recunoasterea feței sale, am putea valida suplimentar prezența acestuia prin conectarea lui la reteaua *Wi-fi* a locuinței. Un alt lucru pe care ar trebui să-l luăm în calcul pe viitor este posibilitatea adăugarii de etichete personalizate. Momentan, etichetele recunoscute de sistem sunt limitate (e.g knife, animal). Pentru a putea permite crearea de reguli cât mai complexe și personalizate nevoilor utilizatorilor, am putea permite acestuia să-și înregistreze propriile etichete care să fie recunoscute de camerele de supraveghere.

Pe lângă asta, ar mai fi cresterea tipurilor de dispozitive care pot fi înregistrate în cadrul platformei. Momentan acestea sunt limitate de starea *open-close* aleasă pentru cazul general. Pentru a putea include mai multe dispozitive ar trebui să putem recunoaste mai multe stări. Pentru ca acest lucru să fie posibil, aplicația a fost structurată pe module pentru ca modificările să fie cât mai puține și cât mai usor de realizat.

# 8. Bibliografie

[1] Friedemann Mattern and Christian Floerkemeier, ,,*From the Internet of Computers to the Internet of things*”, Distributed Systems Group, Institute for Pervasive Computing, ETH Zurich, 2010, ISBN 978-3-642-17226-7

[2] O. Vermesan, P. Friess, P. Guillemin, H. Sundmaeker, et al., „*Internet of Things Strategic Research Agenda*”, Chapter 2 in Internet of Things – Global Technological and Societal Trends, River Publisher, 2011, ISBN 978-87-92329-67-7

[3] video cameras: https://www.spy-shop.ro/

[4] \* \* \* The Django book, <https://djangobook.com/>

[5] \* \* \*Top 10 websites built with Django, <https://www.linkedin.com/pulse/top-10-sites-built-django-framework-vladimir-bogdanov/>

[6] \* \* \* why-django, <https://djangobook.com/tutorials/why-django/>

[7] \* \* \* about-aws, <https://aws.amazon.com/about-aws/>

[8] \* \* \* AWS pricing, <https://aws.amazon.com/pricing/>

[9] \* \* \* AWS S3, <https://aws.amazon.com/s3/>

[10] \* \* \* Amazon Rekognition, https://aws.amazon.com/rekognition/

[11] \* \* \* We compared the 3 best image analysis API’s — here’s what we learned

<https://engineering.musefind.com/we-compared-the-3-best-image-analysis-apis-here-s-what-we-learned-2d54cff5ae62>

[12] \* \* \*Amazon SNS, https://aws.amazon.com/sns/

[13] \* \* \*, *Push Notifications Explained:* https://www.urbanairship.com/push-notifications-explained

[14] \* \* \*Sequence diagrams , <https://sequencediagram.org/>

[15] \* \* \* MySQL 8.0 Reference Manual

, <https://dev.mysql.com/doc/refman/8.0/en/>

[16] \* \* \* MySQL Connector/Python Developer Guide, <https://dev.mysql.com/doc/connector-python/en/>

[17] Sabin Buraga , „*Cursurile de Tehnologii Web și Dezvoltarea aplicaţiilor Web*

*la nivel de client*”, 2015-2016

[18] Ovidiu Vermesan, Peter Friess, ,,*Internet of Things Applications: From Research and Innovation to Market Deployment*”, River Publishers, 2014

[19] Miao Wu et. al., *„Research on the architecture of Internet of things”,* in the proceedings of 3rd International Conference on Advanced Computer Theory and Engineering, 20-22 August, 2012,Beijing, China,

[20] H. Tshofenig, J. Arkko, D. Thaler, D. McPherson*, „Arhitectural Considerations in Smart Object Networking”*(RFC 7452), Martie 2015

[21] Paul Barry, „Head First Python”, O’Reilly Media, Second Edition, Noiembrie 2016,

ISBN 978-1-491-91953-8

[22]Jenifer Tidwell, „Designing Interfaces”, O’Reilly Media, Noiembrie 2005,

ISBN 987-0-596-00803-1

[23] Stan Z. Li, Anil K. Jain, „Handbook of Face Recognition”, Springer, Second Edition, 2005,

ISBN 978-0-85729-931-4