

UNIVERSITATEA POLITEHNICA BUCUREȘTI  
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
DEPARTAMENTUL CALCULATOARE



## PROIECT DE DIPLOMĂ

Magic Mirror

Aplicație pentru oglindă inteligentă cu recunoaștere facială

Silviu Darius Marin

**Coordonator științific:**

Conf. dr. ing. Cornel Popescu

**BUCUREȘTI**

2018

## CUPRINS

CUPRINS .....	2
Sinopsis .....	5
1   Introducere .....	6
1.1   Context .....	6
1.2   Problema .....	6
1.3   Obiective .....	6
1.4   Soluția propusă .....	7
1.5   Rezultatele dorite .....	8
1.6   Structura lucrării .....	8
2   Analiza cerințelor .....	9
2.1   Cerințe generale de funcționare .....	9
2.2   Cerințe pentru funcția de configurare a oglinzii .....	9
3   Studiu de piață / Metode existente .....	10
3.1   Soluții existente pe piață .....	10
3.1.1   Preliminarii .....	10
3.1.2   Allview SMART MIRROR .....	10
3.1.3   MagicMirror <sup>2</sup> .....	11
3.1.4   HomeMirror .....	12
3.1.5   Smart Mirror – Evan Cohen .....	13
3.1.6   PANL .....	14
3.2   Concluzii studiu de piață .....	15
3.3   Tehnologii folosite în realizarea proiectului .....	16
3.3.1   Tehnologii Hardware .....	16
3.3.1.1   Raspberry Pi 3 Model B .....	17

3.3.1.2	Camera PI NoIR V2.....	17
3.3.1.3	Modul Senzor de Temperatură și Umiditate DHT22.....	18
3.3.1.4	REFLECTIV MIR - Folie colorată tip oglindă - one-way.....	18
3.3.1.5	Radiatoare pentru Raspberry Pi 3 și ventilator.....	19
3.3.1.6	Bandă LED RGB.....	19
3.3.1.7	Monitor.....	20
3.3.2	Software.....	20
3.3.2.1	Raspbian.....	20
3.3.2.2	Python.....	21
3.3.2.3	Biblioteca Python GPIO.....	22
3.3.2.4	Biblioteca Adafruit Python DHT Sensor.....	22
3.3.2.5	Biblioteca OpenCV.....	23
3.3.2.6	Tkinter Framework.....	23
3.3.2.7	Flask Framework.....	23
3.3.2.8	Peewee.....	24
3.3.2.9	Baza de date.....	25
3.3.2.10	HTML / CSS / Bootstrap.....	25
4	Soluția propusă.....	26
4.1	Arhitectura generală.....	27
4.2	Arhitectura modului central (Raspberry Pi).....	28
4.2.1	Aplicația centrală.....	29
4.2.2	Serverul Web.....	29
4.2.3	Baza de date SQLite.....	30

5	Detalii de implementare .....	32
5.1	Schema electrică a Magic Mirror și funcționarea acesteia.....	32
5.2	Implementarea modulului central.....	33
5.2.1	Implementarea Aplicației centrale.....	33
5.2.1.1	Modulul de recunoaștere facială.....	33
5.2.1.2	Modulul de Vreme.....	35
5.2.1.3	Modulul de Ceas.....	35
5.2.1.4	Modulul de Temperatura.....	36
5.2.1.5	Modulul de Mail.....	36
5.2.1.6	Modulul de Trafic.....	37
5.2.2	Implementarea Serverului web.....	38
5.2.3	Implementarea Bazei de date.....	38
6	Evaluare .....	40
6.1	Corectitudinea implementării.....	40
6.1.1	Corectitudinea recunoașterii utilizatorului.....	40
6.1.2	Corectitudinea afișării informațiilor.....	40
6.1.3	Corectitudinea măsurării temperaturii și umidității.....	41
6.2	Respectarea cerințelor.....	41
7	Concluzii.....	42
7.1	Concluzii generale.....	42
7.2	Dezvoltări ulterioare.....	42
8	REFERINȚE.....	43
9	Bibliografie.....	45
10	Anexe .....	46

## SINOPSIS

Magic Mirror a fost dezvoltat în contextul curent, în care multe din dispozitivele pe care le folosim în ziua de azi tind să devină inteligente (smartTV, smartPhone, smartWatch), astfel conceptul de *Internet of Things* luând amploare. Nu există bariere când vine vorba de „smart” – orice obiect poate deveni „smart”, iar oglinzile au un mare potențial pe piața obiectelor inteligente.

Motivat de utilitatea unei oglinzi deștepte, acest proiect își propune dezvoltarea lui Magic Mirror la costuri reduse, pentru a putea fi utilizat la o scară cât mai mare. Această lucrare folosește ca la bază o oglindă one-way view, un Raspberry Pi 3 Model B și sistemul de operare Raspbian, care vor contribui la realizarea unei oglinzi ce prezintă câteva funcții simple, însă esențiale. Abordarea mea propune, în plus față de modele existente, integrarea recunoașterii faciale, ceea ce va permite customizarea informațiilor afișate în funcție de utilizator, totul la un preț cât mai redus.

## ABSTRACT

Magic Mirror was designed within its present context, when many of the devices we use nowadays are becoming smart (smartTV, smartPhone, smartWatch), thus the concept of the *Internet of Things* becoming more prominent. There are no limits when it comes to smart – any object can become smart, and mirrors have great potential on the smart technology market.

Motivated by the usefulness of a smart mirror, this work aims at developing Magic Mirror at reduced expenses, so as to be used on a much larger scale. This paper therefore bases itself on a one-way view mirror, a Raspberry Pi 3 Model B and the Raspbian operating system, which will contribute towards the creation of a mirror that showcases simple but essential features. My approach proposes, in addition to existing models, the integration of facial recognition, which will allow for the customisation of the displayed information depending on the person using the mirror, all these at a reduced cost.

# 1 INTRODUCERE

## 1.1 Context

Cu toții știm ce este o oglindă. Cu toții avem în casă o oglindă. Cu toții folosim oglinda pentru a vedea reflexia cuiva sau a ceva. Zi de zi observăm o prezență cât mai proeminentă și o evoluție rapidă a tehnologiei. Internetul ne-a transformat viețile și modul în care ne desfășurăm activitățile, aducând informațiile necesare nouă la doar un click distanță. Dar ce se întâmplă atunci când combinăm oglinda cu tehnologia de astăzi? Această idee a început ușor, în ultimul an, să prindă avânt.

## 1.2 Problema

În prezent, lipsa timpului este una din cele mai predominante probleme din viața oamenilor, aceștia fiind ocupați cu treburile zilnice, cu joburile, cu familia etc. Astfel, oamenii își doresc să piardă cât mai puțin timp cu activitățile zilnice de rutină. În acest sens, vin în ajutorul lor dispozitivele smart (smartPhone, smartWatch, etc.), care aduc mai aproape oamenii informațiile dorite, dar și o interacțiune plăcută între obiect și utilizator. Pe lângă acestea, dispozitivele ce intră în componența unei case inteligente (smart home), cum ar fi termostate, becuri, încuietori, mașini de spălat, frigidere inteligente etc., au început să intre ușor în viața oamenilor.

Un dispozitiv pe care oamenii îl folosesc zi de zi este oglinda. Aceasta ar putea foarte ușor să devină „smart” și să simplifice activitățile utilizatorului prin punerea la dispoziție a informației cât mai rapid. Marea problemă a acestor dispozitive este prețul ridicat cerut de marile companii producătoare, motiv pentru care oglinda smart încă nu a căpătat printre utilizatorii de tehnologie o popularitate pe măsura potențialului ei de a îmbunătăți calitatea vieții moderne.

## 1.3 Obiective

O oglindă inteligentă ajută utilizatorul cu diverse informații, pentru ca acesta să nu mai fie nevoit să piardă timpul pentru a le căuta în altă parte. Orice persoană vrea să afle înainte să iasă din casă cum este și cum va fi vremea în acea zi. Pentru aceasta, automat va verifica pe telefon sau va urmări la TV prognoza meteo. O oglindă inteligentă te scutește de acest „efort” și îți afișează prognoza meteo.

Tot dimineața, când te pregătești de o noua zi, îți dorești să știi cum este traficul, dacă sunt blocaje, accidente sau alte evenimente în drumul tău spre serviciu. Pentru a afla toate acestea, ești nevoit să pornești GPS-ul pe telefon și să verifici. O oglindă inteligentă îți poate arăta în fiecare dimineața care este cel mai scurt/rapid drum spre locul în care trebuie să ajungi, dacă ești în întârziere și poate face o estimare a timpului de deplasare.

O modalitate de comunicare gratuită și rapidă în ziua de astăzi este email-ul (poșta electronică). Oglinda îți poate arăta ce mail-uri ai primit, pentru a le putea citi în timp ce faci alte activități de rutină.

Oglinda poate, cu ajutorul unei camere video și a recunoașterii faciale, să deducă cine este în fața ei în fiecare moment. Cu ajutorul acestei informații ea poate să afișeze informațiile personalizat, fiecare utilizator beneficiind de informațiile necesare în funcție de configurațiile stabilite anterior pentru contul și utilizatorul respectiv.

Obiectivul principal este acela de a integra toate aceste servicii, astfel încât oglinda să ofere funcții foarte utile, însă de bază, iar costurile producerii unui astfel de obiect să fie reduse pe cât posibil. În perspectivă, îmi propun ca oglinda să contribuie, alături de alte obiecte deștepțe, la dezvoltarea unui stil de viață *smart*, accesibil cât mai multor persoane.

## 1.4 Soluția propusă

Soluția propusă de mine este construirea unei aplicații hardware-software care să rezolve cele două obiective enumerate mai sus: simplificarea vieții utilizatorului și reducerea prețului unei oglinzi smart, iar această soluție este una destul de simplă.

Ea propune și o abordare nouă, aceea a integrării recunoașterii faciale, ceea ce va duce la personalizarea serviciilor oferite de oglinda inteligentă.

Pentru aceasta, am proiectat și construit o oglindă inteligentă care să ajute utilizatorul afișând informațiile dorite de acesta. Oglinda se bazează pe un dispozitiv *embedded*, mai mulți senzori, un ecran și o oglindă de tip *one-way* care permite trecerea razelor pe o parte iar pe partea cealaltă se comportă întocmai ca o oglindă.

## 1.5 Rezultatele dorite

Se dorește obținerea unui produs funcțional, format din partea hardware și aplicația software, împreună formând o oglindă inteligentă, Magic Mirror, care să ușureze viața utilizatorului prin aducerea mai aproape a informațiilor dorite de acesta.

## 1.6 Structura lucrării

Lucrarea conține 9 capitole cu următoarea structură:

- Capitolul 1, Introducerea, prezintă subiectul lucrării, împreună cu problema și soluția propusă. Capitolul 2, Analiza cerințelor, ilustrează dorința potențialilor utilizatori care s-au prezentat atrași de ideea proiectului;
- Capitolul 3, Studiu de piață, oferă o viziune de ansamblu a alternativelor existente în acest moment pe piață, împreună cu punctele tari și slabe ale acestora, precum și o comparație între acestea și soluția propusă de mine. Tot aici se regăsesc și informații despre tehnologiile folosite în soluția propusă;
- Capitolul 4, Soluția propusă, expune arhitectura soluției identificate și tehnologiile folosite pentru implementarea acesteia;
- Capitolul 5 prezintă detaliile de implementare și modul de realizare al soluției;
- Capitolul 6 este o evaluare a rezultatelor proiectului, în care se analizează dacă obiectivele și cerințele au fost îndeplinite;
- Capitolul 7 conține rezumatul lucrării și concluziile trase;
- Capitolul 8 prezintă bibliografia cu toate materialele studiate pentru acest proiect;
- Capitolul 9 este capitolul cu anexe, unde se găsesc alte informații referitoare la proiect.



## 2 ANALIZA CERINȚELOR

Pentru a stabili dorințele potențialilor utilizatori/clienti, am purtat, de-a lungul ultimelor luni, numeroase discuții cu multe cunoștințe pasionate de tehnologie pe care eu le-aș vedea fiind utilizatori de oglinzi inteligente. În urma acestor discuții, am reușit să înțeleg ce nevoi și dorințe au aceștia și ce și-ar dori să aibă și să le ofere o oglindă inteligentă, precum și prețul pe care ar fi dispuși să îl investească.

### 2.1 Cerințe generale de funcționare

- *Plug & Play* – toată lumea și-a dorit ca oglinda să fie foarte simplu de utilizat (o bagi în priză și aceasta funcționează);
- Posibilitatea de a-și crea propriile module – o mare parte dintre ei și-au exprimat dorința ca la nevoie să poată să-și creeze propriile module pentru oglindă;
- Posibilitatea utilizării aceleiași oglinzi de către mai multe persoane – toată lumea și-a dorit ca oglinda să poată fi utilizată de mai multe persoane, nefiind nevoie de configurări speciale de fiecare dată;
- Prețul – în legătură cu prețul, părerile au fost împărțite, fiecare fiind dispus să investească o anumită sumă pentru achiziționarea unei oglinzi inteligente sau preferând să investească mai puțin financiar, însă mai mult timp în scopul confecționării unui astfel de obiect; toți au fost însă de acord că pentru ca oglinda să nu își piardă utilitatea și să poată fi parte din investiția într-un ansamblu de obiecte inteligente, aceasta nu ar trebui să depășească 1 000 RON;
- Depanarea – mulți dintre cei cu care am vorbit și-au exprimat dorința ca oglinda să poată fi reparată/updatată fără a fi nevoie să o dea jos din perete;
- Consum redus de energie.

### 2.2 Cerințe pentru funcția de configurare a oglinzii

- Simplitate – utilizatorii își doresc ca modul de configurare al oglinzii să fie intuitiv și să nu necesite citirea unui manual de folosință;
- Interfață plăcută și minimalistă;
- Să poată fi accesată de pe orice dispozitiv – răspunsul primit la întrebarea „De pe ce dispozitiv electronic v-ar plăcea să va puteți face configurarea oglinzii?” a fost unul foarte variat, fiecare utilizator având propria părere (telefon, tableta, laptop).

### 3 STUDIU DE PIAȚĂ

Marile companii au reușit să scoată pe piață câteva modele, marele lor dezavantaj fiind prețul inaccesibil. Ideea fiind la început dar și în plin avânt, prețurile variază foarte mult. Prețul de producție al unei oglinzi fiind relativ mic, diferența dintre prețul la raft și prețul de producție este profitul companiilor. Acestea tind să facă cât mai mult profit, deoarece sunt destul de puține modele stabile pe piață. O oglindă inteligentă se poate face și acasă cu destul de puțini bani și câteva cunoștințe de programare.

#### 3.1 Soluții existente pe piață

Multe din modelele existente de oglinzi inteligente au fost create acasă, de către diferite persoane pasionate de tehnologie, motiv pentru care evaluarea acestora s-a efectuat pe baza informațiilor și a recenziilor disponibile online.

##### 3.1.1 Preliminarii

Ca notă generală, o oglindă inteligentă (precum toate modelele evaluate mai jos) are la bază o oglindă care reflectă razele pe o parte, iar pe partea cealaltă permite trecerea lor (one-way view). În spatele acesteia, pe partea care permite trecerea razelor, se montează un ecran pe care se afișează informații, imagini etc. În momentul în care ecranul este stins, oglinda se comportă ca o oglindă obișnuită. Dacă ecranul este pornit și sunt afișate informații pe el, oglinda proiectează pe ea informațiile și utilizatorul le va vedea.

Ecranul se va conecta la un calculator/SBC (Single-Board Computer) pe care va rula aplicația oglinzii. În cele de mai jos se va vedea cum aplicația diferă în funcție de firma care produce oglinda sau de programatorul care a creat-o (dacă se construiește acasă).

##### 3.1.2 Allview SMART MIRROR

Cei de la Allview comercializează pe site-ul oficial [1] un model de oglindă inteligentă. Prețul este unul destul de mare (3.999 lei).

Oglinda vine în două mărimi diferite, 80/60 și 50/80 cm. Are Bluetooth și WiFi încorporate și un senzor pentru detectarea mișcării.



Figura 1. Allview SMART MIRROR<sup>1</sup>

Software-ul oglinzii este structurat sub forma a 3 secțiuni, Secțiunea *Health*, unde se poate urmări un istoric al greutății, Secțiunea *Entertainment*, unde se găsesc aplicații și se poate naviga pe internet și Secțiunea *Smart Home*, de unde se pot accesa ceilalți senzori din casă conectați la oglindă.

### 3.1.3 MagicMirror<sup>2</sup>

Este o oglindă creată de Michael Teeuw într-o ramă de lemn, un televizor cu ecran plat, un Raspberry Pi 2 și cu un software care rulează în browser. Conceptul modelul a apărut prima dată în 2014 pe un blog scris de Teeuw și a atras foarte rapid atenția, ajungând la 2.5 milioane de vizualizări. Codul sursă și un ghid de construire a hardware-ului au fost și ele postate pe blog.

Software-ul a fost dezvoltat și este în continuă dezvoltare de o întreagă comunitate și în 2016 a fost rescris folosind Electron. Oglinda se numește acum MagicMirror<sup>2</sup>, suportă widget-uri numite module și oricine își poate crea propriul modul. Instalarea de bază vine cu câteva module elementare, calendar, vreme și știri și se pot instala separat alte module.



Figura 2. MagicMirror<sup>2</sup> și Michael Teeuw<sup>2</sup>

MagicMirror<sup>2</sup> este o oglindă interesantă, cu un aspect plăcut și o interfață cu utilizatorul simplă. În spatele ei se află o întreagă comunitate care lucrează la noi module și la perfecționarea celor existente.

#### 3.1.4 HomeMirror

HomeMirror este o oglindă creată de Hannah Mittens Morrison, care a făcut cunoștință cu publicul pentru prima oară printr-o postare pe Reddit în 2015. Această oglindă are la bază o tabletă Nexus 7 ce rulează Android și care e poziționată în spatele oglinzii. Astfel nu mai este nevoie de un calculator și de un ecran separat.

Software-ul oglinzii este scris în Java și rulează pe Android sub forma unei aplicații. Ea arată informații utile cum ar fi vremea, ceasul, data și evenimente. HomeMirror intenționează să fie folosită ca un panou de informații, nepermițând interacțiunea cu ea. Software-ul nu suportă adăugarea de module proprii, dar codul este open-source, putând fi modificat la nevoie.

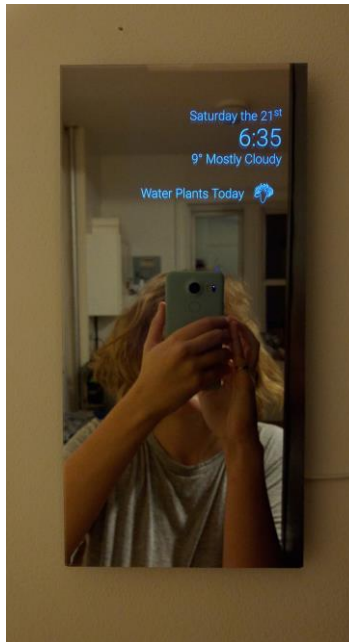


Figura 3. HomeMirror și Hannah Mittens Morrison<sup>3</sup>

HomeMirror este un dispozitiv care arată frumos și este mai ușor de construit decât altele, deoarece necesită doar 2 componente, o tabletă și oglindă.

### 3.1.5 Smart Mirror – Evan Cohen

Oglinda lui Evan Cohen este inspirată din ambele proiecte MagicMirror<sup>2</sup> și HomeMirror. Acest proiect se axează pe partea de software, implementând o gamă largă de comenzi vocale pentru a putea interacționa cu oglinda. Comenzile vocale suportă interacțiunea cu vremea, ceasul, data, afișarea hărților, afișarea pozelor.



Figura 4. Evan Cohen și oglinda lui inteligentă<sup>4</sup>

Pentru a crea hardware-ul pentru acest proiect este nevoie de o oglindă de tip one-way, un ecran și un Raspberry Pi și un microfon USB pentru a putea folosi comenzile vocale.

Software-ul este open-source, este creat în Electron și poate fi găsit pe GitHub-ul lui Evan. Chiar dacă este mult mai avansat decât software-ul celorlalte proiecte, el a fost gândit ca o aplicație de sine stătătoare care poate face mai multe lucruri, astfel fiind destul de dificil să îți adaugi propriile module la această aplicație.

### 3.1.6 PANL

PANL (<http://getpanl.com/>) este una din primele oglinzi inteligente cu ecran tactil. Ea a fost creată de Ryan Nelwan și a fost dezvăluită în aprilie 2016 în California când și-a înființat un start-up. Ryan postase un videoclip cu oglinda pe Reddit, clipul devenind viral în foarte scurt timp și fiind citat de majoritatea blogurilor despre tehnologie..

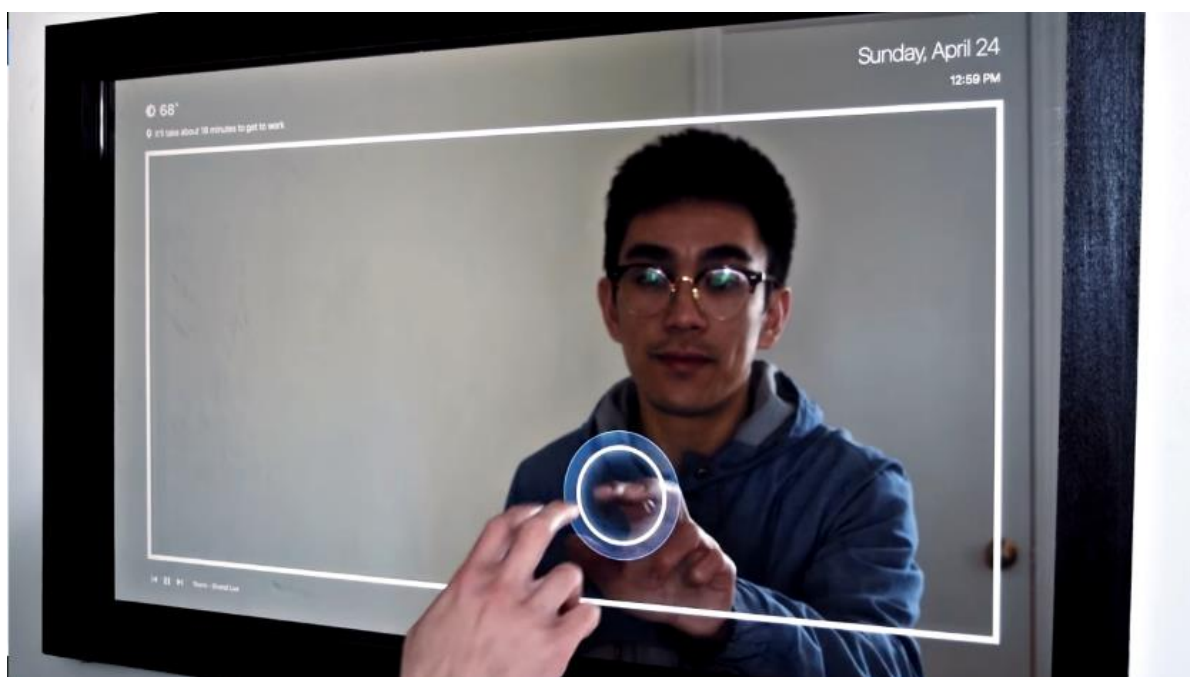


Figura 5. Ryan Nelwan prezentând PANL<sup>5</sup>

Singurele informații pe care le știm despre acest proiect sunt ce putem observa în videoclipul pe care Ryan l-a postat pe YouTube. El nu a spus nimic despre partea hardware sau software a proiectului, dar putem vedea în video că oglinda are suport multi-touch și poate arăta informații utile precum vremea, ora, data. Pe lângă acestea oglinda are suport pentru muzică și pentru clipuri de pe YouTube și are aplicații precum Uber, Messages, Photos, DropBox, Reddit. Oglinda vine și cu o tastatură virtuală care apare pe ecran când este nevoie.

### 3.2 Concluzii studiu de piața

În urma studiului făcut asupra modelelor existente pe piață la momentul actual, se poate observa că modelul comercializat de cei de la Allview este unul cu finisaje deosebite și care vine cu un set de utilitare gândite pentru nevoile oamenilor. Cei de la Allview încearcă să integreze oglinda lor în conceptul de „Internet of Things” (IoT) prin conectarea oglinzii cu alte dispozitive smart, oglinda devenind un Hub din care se pot controla celelalte dispozitive. Marele său dezavantaj este prețul, acesta fiind unul foarte ridicat și nu multă lume este pregătită să investească 4000 de lei. Oamenii ar fi dispuși să investească în dispozitive smart, dar la un preț accesibil.

Trecând peste gama de produse comercializate de companii, regăsim câteva modele foarte interesante proiectate și lansate ca aplicații open-source.

În opinia mea, ideea lui Michael Teeuw este una genială, el creând o bază a oglinzii iar apoi fiecare om putând să și-o personalizeze cu propriile module. Ideea a fost dezvoltată fiind creată o comunitate care susține proiectul cu noi module și cu suport pentru cele deja existente, astfel oferind oricui își dorește un Smart Mirror posibilitatea procurării software-ului pe gratis. Singurul dezavantaj pe care l-am văzut în acest proiect este faptul că este scris în Electron, un Framework de Java Script, iar crearea unui modul de recunoaștere facială stabil ar fi fost complicat.

Ideea lui Ryan Nelwan de a crea o oglindă cu ecran tactil este una de efect, utilizatorii fiind interesați la prima vedere de acest proiect. La o cercetare mai amănunțită în ce ar putea consta proiectul am descoperit că „folia touch” care este aplicată pe oglindă se găsește de cumpărat foarte greu pe internet și are un preț foarte mare. Prețul variază în funcție de dimensiunile foliei, pentru o oglindă de dimensiunile celei a lui Ryan prețul foliei ajungând undeva la 500\$.

	Allview SMART MIRROR	MagicMirror <sup>2</sup>	Evan Cohen S.M.	Magic Mirror
Design oglinda	Standard	Dinamic	Dinamic	Dinamic
Recunoaștere facială	X	-	-	X
Comenzi vocale	-	-	X	-
Ecran tactil	X	-	-	-
Limba software	Android	Electron	Electron	Python
Preț	4000 RON	~750 RON	~750 RON	~750 RON

Tabel 1. Comparație între Magic Mirror și alte soluții existente pe piață

### 3.3 Tehnologii folosite în realizarea proiectului

#### 3.3.1 Tehnologii Hardware

Termenul „sistem înglobat” sau „sistem încorporat” provine de la expresia în engleză, embedded system. Sistemul încorporat este o combinație de hardware (un mic calculator bazat pe un microprocesor sau microcontroler) și software specializat (și înglobat), proiectat să îndeplinească o anumită funcție, sau câteva sarcini, de obicei în timp real (aproape instantaneu). Informație din [19].

Sistemele înglobate sunt foarte folosite în ziua de astăzi deoarece au un cost redus, dimensiunea este redusă, lucrează în timp real, consumă puțină energie electrică..



### 3.3.1.1 Raspberry Pi 3 Model B

Partea cea mai importantă a proiectului este microcontrolerul. Am ales un Raspberry Pi 3 Model B.



Figura 6. Raspberry Pi 3 Model B<sup>6</sup>

Ca specificații: SoC – Broadcom BCM2837, CPU – 4 x ARM Cortex-A53 de 1.2 GHz, 1GB de RAM la o frecvență de 900MHz, 40 de pini GPIO, 4 x USB 2.0, port HDMI, slot de Micro SD Card, 10/100 Ethernet, Bluetooth, Wireless, port CSI pentru conectarea camerei Raspberry Pi, port DSI pentru conectarea unui ecran touchscreen. Acesta se alimentează la 2.5A cu o tensiune de ieșire de 5V.

### 3.3.1.2 Camera PI NoIR V2

Pentru partea de recunoaștere facială am ales să folosesc camera PI NOIR CAMERA V2 de la Raspberry. Ea este proiectată să se conecteze foarte ușor la Raspberry Pi și să se utilizeze la fel de ușor cu ajutorul bibliotecii Picamera.



Figura 7 și 8. Camera PI NoIR V2 și conectarea sa la Raspberry Pi<sup>7</sup>

Ca specificații, camera are un senzor Sony IMX219 de 8 mega pixeli. Oferă aceleași funcționalități ca o camera obișnuită, singura diferență fiind aceea că nu are un senzor infraroșu (NoIR = No Infrared). Acest lucru înseamnă că pozele făcute pe timp de zi vor arăta puțin ciudat dar se pot face poze pe timp de noapte cu o lumină infraroșie. Rezoluția video este 1080p 30fps, 720p 60fps, 640x480p 60/90fps.

### 3.3.1.3 Modul Senzor de Temperatură și Umiditate DHT22

Pentru monitorizarea temperaturii ambientale și a umidității interioare am ales să folosesc un senzor DHT22. Este foarte simplu de conectat la Raspberry Pi și se utilizează ușor cu ajutorul bibliotecii de la Adafruit.

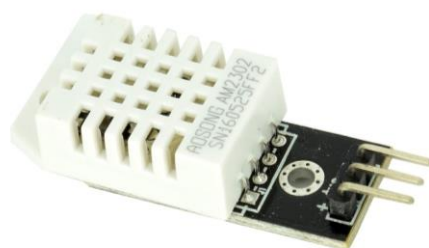


Figura 9. Modul senzor de temperatură și umiditate DHT22<sup>8</sup>

Senzorul DHT22, un senzor de umiditate și temperatură, are avantajele unei dimensiuni reduse și a unui preț accesibil. Prin intermediul unui senzor capacitiv de umiditate și a unui termistor, acesta măsoară aerul din jur. Mai mult, senzorul oferă semnal digital pe pinul de date (așadar nu este nevoie de pin analogic).

Senzorul se alimentează cu o tensiune între 3.3 și 6 volți. Poate opera între -40 și 80 de grade Celsius. Timpul de citire a informațiilor este de 2 secunde și acuratețea senzorului este de  $\pm 0.5$  °C.

### 3.3.1.4 REFLECTIV MIR - Folie colorată tip oglindă - one way

Pentru oglinda propriu zisă am ales o variantă ieftină și practică, folie adezivă care se poate aplica pe o sticlă/geam sau pe o bucată de plexiglas. Folia MIR arată ca o oglindă pe o parte și păstrează transparența sticlei pe cealaltă parte.



Figura 10. Modele de folie Reflectiv MIR<sup>9</sup>

### 3.3.1.5 Radiatoare pentru Raspberry Pi 3 și ventilator

Pentru partea de răcire a Raspberry Pi-ului am ales să folosesc radiatoare din aluminiu pentru a evita supraîncălzirea și un ventilator pentru a disipa căldura produsă de Raspberry Pi.



Figura 11. Radiatoare<sup>10</sup>



Figura 12. Ventilator<sup>11</sup>

### 3.3.1.6 Bandă LED RGB

Pentru un efect vizual plăcut al oglinzii am ales o bandă LED RGB pe care am montat-o în spatele ramei și am conectat-o la Raspberry Pi. Aceasta își schimbă culoarea în funcție de utilizatorul curent. Fiecare utilizator își poate seta o culoare preferată.



Figura 13. Banda LED RGB

### 3.3.1.7 Monitor

Pentru partea de ecran am ales un monitor cu HDMI pentru a facilita conectarea ușoară la Raspberry Pi. Pentru a câștiga spațiu, i-am scos suportul și carcasa, păstrând doar ecranul propriu-zis, pe care l-am fixat în rama oglinzii.

### 3.3.2 Software

#### 3.3.2.1 Raspbian



Figura 14. Logo Raspberry Pi și Raspbian<sup>12</sup>

Ca sistem de operare am ales să folosesc Raspbian. Acesta este un sistem de operare gratuit care are la bază Debian și este conceput special și optimizat pentru Raspberry Pi. Este sistemul de operare oficial oferit de Raspberry Pi Foundation pentru familia sa de microcontrolere.

Se poate instala foarte ușor cu ajutorul utilitarului NOOBS (New Out Of the Box Software), ceea ce oferă un mare avantaj pentru începători, deoarece șansele să strici ceva în sistemul de operare sunt foarte mari, acesta necesitând reinstalarea.

Mai mult decât atât, Raspbian vine cu mai mult de 35000 de pachete și multe soft-uri instalate deja (Python, Scratch, Sonic Pi, Java, Mathematica) și oferă o interfață simplă pentru activarea comunicării cu porturile de Cameră, SPI, I2C, Serial, 1-Wire, SSH, precum și pentru WiFi și Bluetooth, de câteva din ele eu având nevoie.

Sistemul de operare este în continuă dezvoltare, comunitatea lucrând mereu la update-uri noi și la optimizarea sa. Făcând parte din familia UNIX este foarte ușor de lucrat cu el pentru cineva care a avut contact cu Linux sau cu orice distribuție de UNIX.

### 3.3.2.2 Python



Figura 15. Logo Python<sup>13</sup>

Python este un limbaj de programare interpretat pentru programarea în orice domeniu. Fiind un limbaj interpretat este mult mai rapid deoarece nu necesită compilarea în cod mașină. Este un limbaj foarte popular în rândul programatorilor astfel găsim foarte mult suport pe internet. Pe lângă acest suport oferit de alți programatori pe diverse forumuri, Python beneficiază și de o gamă foarte mare și foarte variată de biblioteci facilitând programarea în orice domeniu. Python suportă mai multe paradigme de programare printre care Programarea Orientată pe Obiecte, Programarea Imperativă, Programarea Funcțională și Programarea Procedurală.

Alternativele existente și motivele pentru care nu au fost alese:

- Java – este un limbaj compilat, necesită instalarea mașinii virtuale Java (JVM) și este cunoscut ca un mare consumator de resurse.
- C/C++-este și el un limbaj compilat, consumă puține resurse dar ar fi fost greu de creat tot proiectul în C/C++ proiectul necompilând dacă se strecoară erori
- Java Script – este un limbaj interpretat la fel ca Python dar în implementarea proiectului tot ar fi fost nevoie de Python pentru controlul pinilor de pe Raspberry Pi

### 3.3.2.3 Bibliotecă Python GPIO

Această bibliotecă este una dintre cele mai importante biblioteci Python pentru Raspberry Pi. Cu ajutorul ei se pot controla foarte ușor pinii de pe placă și implicit și senzorii conectați.





















Raspberry Pi 3 GPIO Header				
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Figura 16. Pini Raspberry Pi 3 Model B<sup>14</sup>

### 3.3.2.4 Bibliotecă Adafruit Python DHT Sensor

Este o bibliotecă scrisă în Python care facilitează utilizatorului citirea ușoară a datelor de la senzorul de temperatură și umiditate DHT22. Funcționează cu toate versiunile de Python și pe toate platformele care pot rula Python.

### 3.3.2.5 Biblioteca OpenCV



Figura 16. Logo OpenCV<sup>15</sup>

Biblioteca OpenCV (Open Source Computer Vision) este o bibliotecă open-source scrisă în C/C++ care conține peste 300 de funcții pentru diverse domenii. Biblioteca are interfațare pentru C++, Python și Java și poate fi rulată pe Windows, Linux, Mac OS, iOS și Android.

Eu am ales această bibliotecă pentru a crea un modul de recunoaștere facială pe Raspberry Pi astfel făcându-se autentificarea utilizatorului, oglinda oferind informații personalizate fiecărui utilizator.

### 3.3.2.6 Tkinter Framework

Python pune utilizatorilor la dispoziție numeroase metode prin care aceștia pot dezvolta interfețe grafice (GUIs). Tkinter este una dintre acestea, el fiind o interfață la „Tk GUI toolkit” oferit automat când se instalează Python. Tkinter a devenit o bibliotecă Python standard pentru crearea de interfețe grafice deoarece este foarte ușor de lucrat cu el.

Cu ajutorul lui am implementat interfața grafică a aplicației care rulează pe ecran în spatele oglinzii. Aici sunt integrate modulele și afișate informațiile oferite de acestea.

### 3.3.2.7 Flask Framework



Figura 18. Logo Flask<sup>16</sup>

Flask este un micro Framework web scris în Python cu ajutorul căruia se pot construi cu ușurință aplicații web. El a fost conceput pentru a putea începe construcția aplicațiilor rapid și ușor și pentru a putea scala până la aplicații complexe. A devenit rapid unul dintre cele mai populare Framework-uri de Python pentru aplicații web deoarece se instalează foarte ușor, se folosește la fel de ușor deoarece nu vine cu pachete de biblioteci lăsând la latitudinea utilizatorului ce biblioteci decide să mai folosească.

Cu ajutorul Framework-ului Flask am creat un server web prin care ajung la baza de date din aplicația de frontend și astfel pot ajusta setările oglinzii.

Alternativele existente și motivele pentru care nu au fost alese:

- Django Framework – este un framework cu mult mai complex și mai vechi pe piață (2003) decât Flask (2010), scris tot în Python care are ca scop crearea de website-uri complexe. Vine la pachet cu multe biblioteci și un panou administrativ cu ajutorul căruia se pot efectua operații pe baza de date. Nu am ales Django din simplul motiv că aveam nevoie de o aplicație web foarte simplă care nu necesita toate bibliotecile cu care vine Django și voiam să am posibilitatea de a îmi alege ce bază de date vreau să folosesc și ce biblioteci pentru abstractizarea lor vreau.

### 3.3.2.8 Peewee



Figura 19. Logo Peewee<sup>17</sup>

Ca ORM (Object Relational Mapping) am folosit Peewee. Este mic, foarte ușor de folosit și simplu. Acceptă multe tipuri de baze de date și se integrează rapid și ușor cu Flask și SQLite pe care am ales să le folosesc.



### 3.3.2.9 Baza de date



Figura 20. Logo SQLite<sup>18</sup>

O oglindă (Magic Mirror) nu va avea mulți utilizatori, fiind folosită acasă, de o familie, astfel nu are nevoie de o bază de date mare. Am ales să folosesc pentru baza de date SQLite deoarece aceasta încapsulează un motor de baze de date SQL și nu necesită instalare și configurare. Baza de date se stochează într-un singur fișier astfel fiind foarte ușor de mutat, șters sau modificat.

Alternativele existente și motivele pentru care nu au fost alese:

- Microsoft SQL Server – necesita instalarea utilitatelor de la Microsoft și a serverului SQL și configurarea acestora. Ar fi ocupat foarte multă memorie pe Raspberry Pi care are memoria limitată.
- MongoDB – este o bază de date nerelațională și exista posibilitatea ca legarea datelor să fie îngreunată

### 3.3.2.10 HTML / CSS / Bootstrap

HTML (Hypertext Markup Language) este un limbaj folosit în domeniul web pentru a crea pagini web. CSS (Cascading Style Sheets) este un standard pentru a face design-ul unei pagini web. Bootstrap este un Framework open-source care combină atât HTML cât și CSS pentru a ușura crearea paginilor web și a oferi creatorului o unealtă de design.

## 4 SOLUȚIA PROPUȘĂ

Soluția propusă de mine este construirea unei aplicații, aceasta fiind formată din 3 părți: partea software, partea hardware și partea fizică (oglinză și ramă) minimizând astfel costul de producție. Pentru partea de software totul a fost gândit modular astfel adăugarea sau eliminarea unor module să poată fi făcută ușor.

Proiectul a început prin achiziționarea unui sistem embedded și a unor senzori. În același timp am început și documentarea referitoare la alte modele de oglinzi inteligente deja create, ce soluție de oglinză este recomandată, ce alte componente ar mai fi necesare și prețurile fiecăror componente.

Pentru partea de oglinză am ales să folosesc folie adezivă pe care o lipesc pe o foaie de sticlă. Astfel am libertatea de alege orice dimensiune a oglinzii la un preț mic. Aceasta este înrămată într-o ramă din lemn.

Pentru partea de ecran am atașat în interiorul ramei un monitor cu HDMI. Acestuia i-au fost înlăturate rama și suportul pentru a ocupa cât mai puțin spațiu.

Raspberry Pi-ul este pus și el tot în interiorul ramei pentru a se putea face cât mai ușor conectarea la ecran.

În interiorul ramei se găsește o sursă de curent (un triplu) pentru alimentarea ecranului și a sistem Embedded. Toate conexiunile sunt făcute în interiorul ramei, din ramă ieșind doar un cablu care va fi băgat în priză.

Pentru partea de soft se folosește sistemul de operare al Raspberry Pi-ului (Raspbian (Linux)) pe care este scrisă aplicația în Python. Pentru partea de recunoaștere facială se folosește biblioteca open-source, OpenCV. Pentru celelalte module se folosesc API-uri de la Google sau alți parteneri.

## 4.1 Arhitectura generală

Proiectul este compus din:

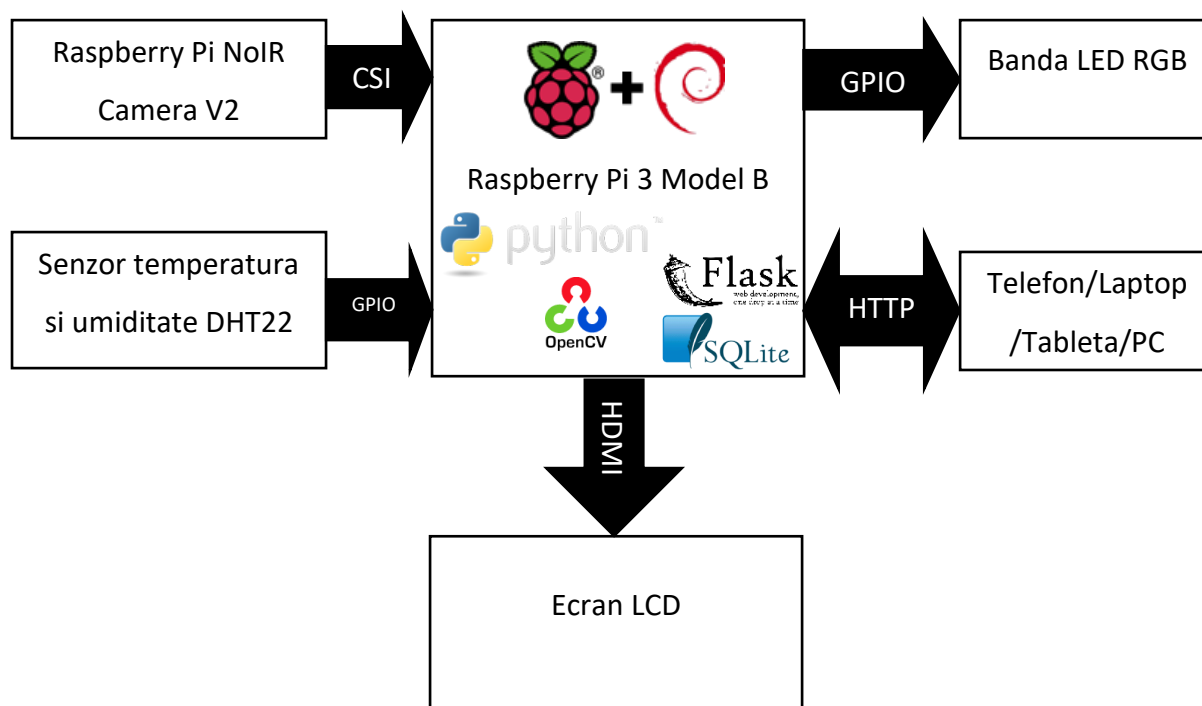


Diagrama 1. Modulele proiectului și modul de comunicare

Raspberry Pi este modulul central care asigură comunicarea între restul componentelor, face procesarea informațiilor și trimite către ecran informațiile necesare utilizatorului.

Camera Raspberry Pi captează imagini pe care le trimite spre Raspberry Pi pentru a fi procesate și pentru a detecta dacă există fețe (facial detection) și pentru a le recunoaște (facial recognition) în cazul în care există.

Senzorul DHT22 trimite către Raspberry Pi informații despre temperatură și umiditate. Astfel putem spune că el are 2 roluri. Raspberry-ul poate afișa utilizatorului datele despre temperatură și umiditatea din încăpere și Raspberry-ul știe când oglinda nu mai poate funcționa și trebuie oprită (umiditatea este excesivă sau temperatura este prea mare).

Banda LED este doar pentru design, utilizatorul putând să-și seteze o culoare favorită iar la recunoașterea sa oglinda să-și schimbe culoarea luminilor.

## 4.2 Arhitectura modului central (Raspberry Pi)

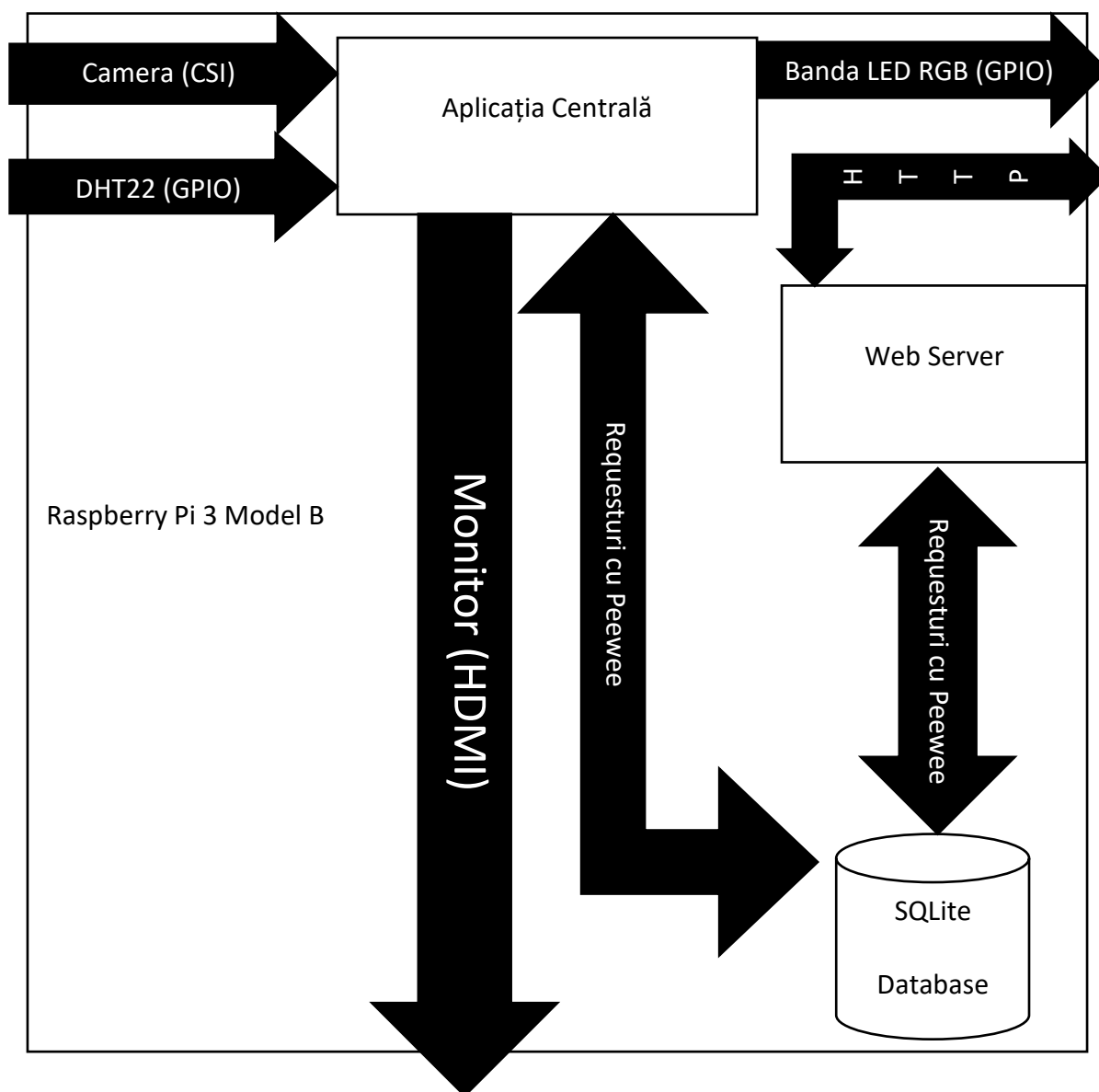


Diagrama 2. Arhitectura modului central

#### 4.2.1 Aplicația centrală

Aplicația centrală este o aplicație scrisă în Python care înglobează în ea modulele de Recunoaștere facială, Vreme, Ceas, Mail și Trafic. Ea comunică cu baza de date pentru a avea acces la informațiile despre utilizatori și despre preferințele acestora.

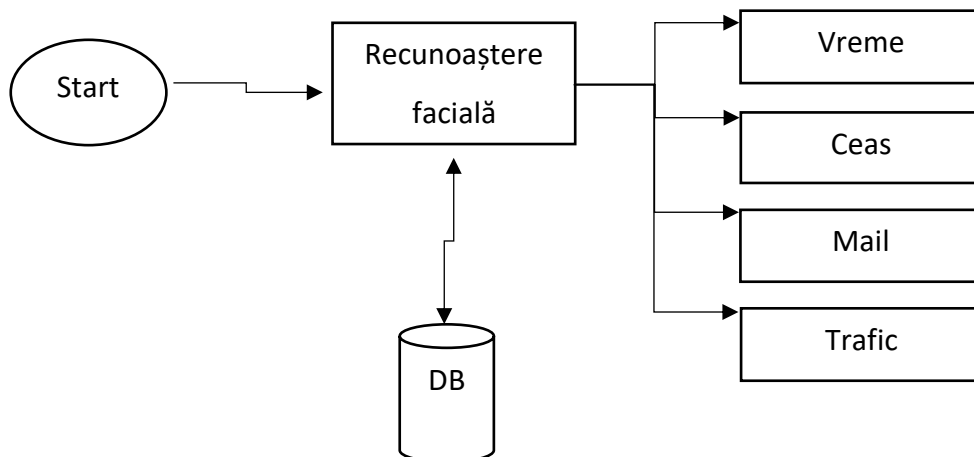


Diagrama 3. Arhitectura Aplicației centrale

#### 4.2.2 Serverul Web

Prin intermediul serverului web utilizatorul își poate crea un cont și configura oglinda în funcție de preferințe. Serverul web comunică cu baza de date unde setările sunt salvate. Serverul web are o interfață formată din 4 pagini web pentru interacțiunea ușoară a utilizatorului cu baza de date.

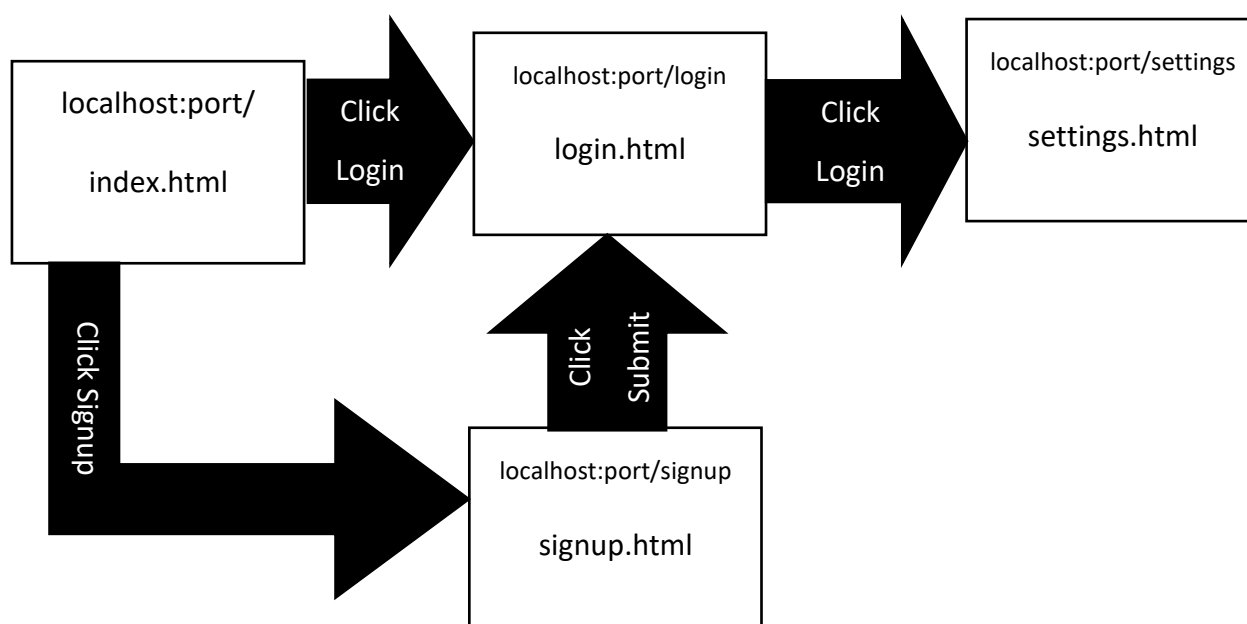
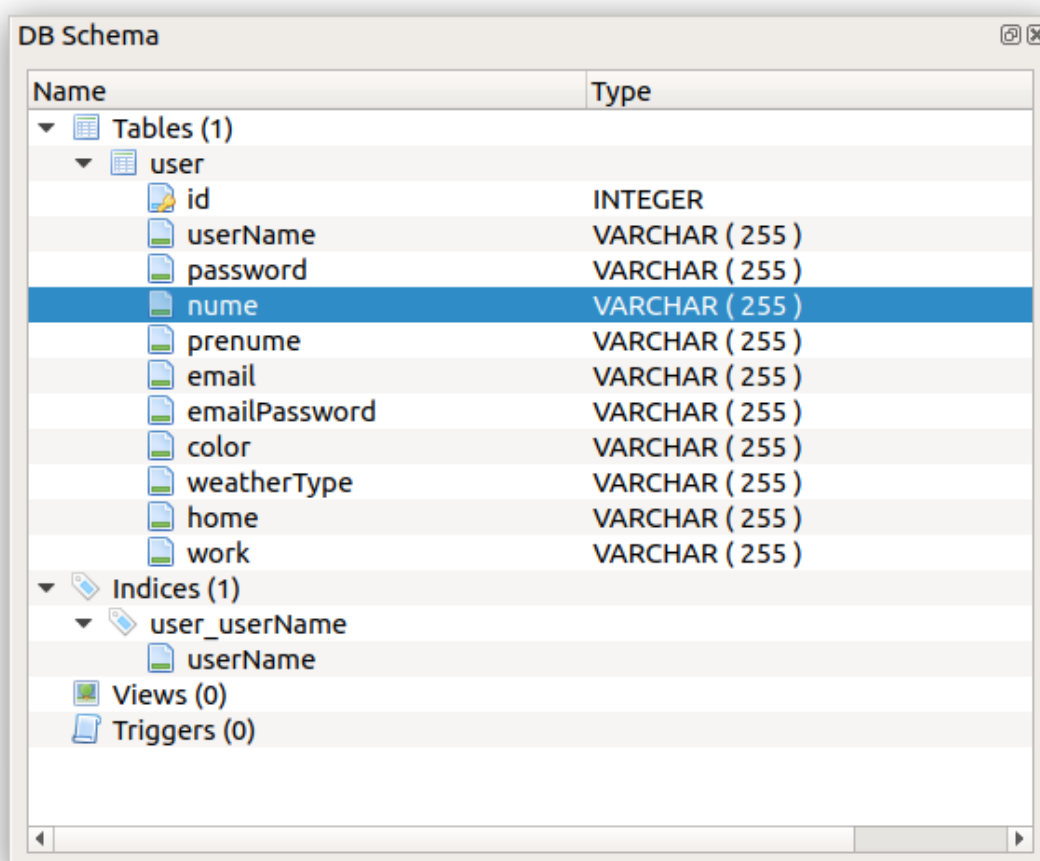


Diagrama 4. Arhitectura paginilor serverului web

Se observă în diagrama de mai sus că în momentul când se intră pe server pagina default este pagina index.html. De aici avem 2 obținui disponibile pentru a ajunge în pagina de configurare (settings.html) și anume ne logăm sau ne creăm un cont nou. Dacă nu avem cont și dorim să creăm unul nou, la final suntem direcționați automat către pagina de login. Pagina de configurare este protejată și nu poate fi accesată fără a fi logați.

#### 4.2.3 Baza de date SQLite

Baza de date este locul unde sunt stocate informațiile despre utilizatori și preferințele acestora pentru configurarea oglinzii. Fiecare utilizator are propriul său profil cu preferințele sale. Baza de date conține tabela User care are următoarele câmpuri:



The screenshot shows a 'DB Schema' window with a tree view on the left and a table of columns on the right. The tree view shows 'Tables (1)' expanded to 'user'. The table of columns lists the following fields and their types:

Name	Type
id	INTEGER
userName	VARCHAR ( 255 )
password	VARCHAR ( 255 )
nume	VARCHAR ( 255 )
prenume	VARCHAR ( 255 )
email	VARCHAR ( 255 )
emailPassword	VARCHAR ( 255 )
color	VARCHAR ( 255 )
weatherType	VARCHAR ( 255 )
home	VARCHAR ( 255 )
work	VARCHAR ( 255 )

Below the table, the 'Indices (1)' section shows an index named 'user\_userName' on the 'userName' column. The 'Views (0)' and 'Triggers (0)' sections are empty.

Figura 21. Schema bazei de date/ Tabela user

- Id – Coloană standard într-o baza de date, de aceasta se face legarea între Id-ul returnat de programul de recunoaștere facială și numele utilizatorului

- userName – Un nume pe care fiecare utilizator și-l alege și cu acesta se va loga în aplicația de configurare
- password – Parolă aleasă pentru a se putea loga în aplicația de configurare
- nume – Numele utilizatorului necesar pentru ca oglinda să afișeze mesaje personalizate
- prenume-Prenumele utilizatorului necesar pentru ca oglinda să afișeze mesaje personalizate
- email – Email-ul utilizatorului necesar modulului de Email pentru a afișa Email-uri
- emailPassword – Parola contului de email
- color – Culoare preferată de utilizator pentru banda LED de pe marginile oglinzii
- weatherType – C sau F necesare modulului de Vreme pentru afișarea temperaturii în Celsius sau Fahrenheit
- home – Punctul de start necesar modului de Trafic pentru a calcula distanța și timpul până la punctul final
- work-Punctul final necesar modului de Trafic pentru a calcula distanța și timpul de la punctul de start

## 5.1 Schema electrică a Magic Mirror și funcționarea acesteia

## 5.1 Schema electrică a Magic Mirror și funcționarea acesteia



32



## 5.2 Implementarea modului central

Modulul central este format din 3 mari părți: Aplicația centrală, Serverul web, Baza de date

### 5.2.1 Implementarea Aplicației centrale

Aplicația centrală este scrisă în Python și cu ajutorul Framework-ului Tkinter este creată o interfață grafică pentru fiecare dintre cele 5 module de Vreme, Ceas, Temperatură, Mail și Trafic. Tot aici se face și recunoașterea facială a utilizatorului.

#### 5.2.1.1 Modulul de recunoaștere facială

Recunoașterea facială se face cu ajutorul bibliotecii OpenCV. Pentru ca un utilizator să fie recunoscut acesta trebuie să antreneze Raspberry Pi-ul cu fața sa. Acest lucru se face în 2 pași. Prima dată se creează un set de date de antrenament ce constau într-un număr de 30 de poze cu fața utilizatorului. După aceea se creează un model de antrenament pentru Raspberry Pi și se salvează local.

```
# Creeaza Dataset cu fata utilizatorului
def create_Dataset(self, id):
    # Numarul de poze facute
    count = 0

    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port = True):
        print("Am un frame...")
        image = frame.array

        # Converteste frame-ul in gri
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

        # Detecteaza fete in frame
        faces = detector.detectMultiScale(gray, 1.3, 5)

        for (x,y,w,h) in faces:
            print ("Am gasit o fata...")
            count+=1

            # Creeaza un patrat in jurul fetei
            cv2.rectangle(image, (x,y), (x+w, y+h), (255,0,0), 2)

            # Salveaza poza la noile dimensiuni
            cv2.imwrite("dataset/User." + str(face_id) + "." + str(count) + ".jpg", gray[y:y+h, x:x+w])
            cv2.imshow("Frame", image)

        key = cv2.waitKey(1) & 0xFF

        rawCapture.truncate(0)

    if key == ord("q"):
        break
    elif count > 30:
        break
```

Figura 22. Funcția de creare a setului de date pentru antrenament

```

# Creaza Histograma
def create_Histogram(self):
    print ("Starting RaspberryPi training...")
    # Ia toate fetele si toate id-urile din setul de date
    faces,ids = getImagesAndLabels('dataset')

    # Antreneaza modelul cu fetele si id-urile
    recognizer.train(faces, np.array(ids))

    # Salveaza modelul in trainer.yml
    recognizer.save('trainer/trainer.yml')

    print ("Training is done")

```

Figura 23 . Functia de antrenare

Modelul fiind salvat putem spune că Raspberry Pi-ul este antrenat, acesta având informațiile necesare pentru a recunoaște fața utilizatorului (fișierul trainer.yml). În continuare el scanează și caută să vadă dacă găsește fețe iar în momentul în care a găsit o față verifică dacă poate recunoaște fața. În cazul pozitiv se returnează user-ul iar în cazul negativ continuă căutarea.

```

# Recunoastere User
def recognize(self):
    print ("Inerc sa recunos subiectul...")
    # Create Local Binary Patterns Histograms for face recognition
    recognizer = cv2.face.LBPHFaceRecognizer_create()

    # Incarca fisierul de antrenament
    recognizer.read('trainer/trainer.yml')

    # Incarca modelele pentru fata
    cascadePath = "haarcascade_frontalface_default.xml"

    # Creeaza clasificator din modele pentru fata
    faceCascade = cv2.CascadeClassifier(cascadePath);

    font = cv2.FONT_HERSHEY_SIMPLEX

    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port = True):
        print("Am un frame...")

        im = frame.array

        # Converteste frame-ul in gri
        gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)

        # Detecteaza fete in frame
        faces = faceCascade.detectMultiScale(gray, 1.2,5)

        # Daca detecteaza fete
        for(x,y,w,h) in faces:
            print("Am gasit o fata...")

            # Creeaza un patrat in jurul fetei
            cv2.rectangle(im, (x-20,y-20), (x+w+20,y+h+20), (0,255,0), 4)

            # Recunoaste fata
            Id, confidence = recognizer.predict(gray[y:y+h,x:x+w])

            if confidence < 80:
                # Cauta userul in baza de date
                user = GetUser(Id)

                # Userul este...
                userIs = user.name
            else:
                userIs = "Unknown"

```

```

        # Pune numele userului pe patratul creat
        cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)
        cv2.putText(im, str(userIs), (x,y-40), font, 2, (255,255,255), 3)

    rawCapture.truncate(0)

    # Afiseaza frame
    cv2.imshow('im', im)

# Opreste camera
cam.release()

# Inchide toate ferestrele
cv2.destroyAllWindows()

```

Figura 24. Funcția de recunoaștere a utilizatorului

### 5.2.1.2 Modulul de Vreme

Modulul de vreme folosește API-ul de la <https://darksky.net/>. Aici mi-am creat un cont pentru a obține o cheie secretă și a putea trimite cereri API-ului.

```

        # Create request 1
        weather_req_url = "https://api.darksky.net/forecast/%s/%s,%s?lang=%s&units=%s" %
(weather_api_token, lat, lon, weather_lang, weather_unit)
    else:
        location2 = ""
        # Create request 2
        weather_req_url = "https://api.darksky.net/forecast/%s/%s,%s?lang=%s&units=%s" %
(weather_api_token, latitude, longitude, weather_lang, weather_unit)

    r = requests.get(weather_req_url)
    weather_obj = json.loads(r.text)

```

Figura 25. Creare model de request și apelarea API-ului

### 5.2.1.3 Modulul de Ceas

Modulul de ceas este unul foarte simplu, el doar citind ceasul sistemului de operare și afișând în interfața grafică informațiile.

#### 5.2.1.4 Modulul de Temperatura

Modulul de temperatură face citirea de la senzorul DHT22 și returnează umiditatea și temperatura către aplicația centrală. Temperatura este returnată în grade Celsius sau Fahrenheit în funcție de preferințele utilizatorului.

```
import Adafruit_DHT

# Modelul de senzori DHT
MODEL = 22
# Pin-ul de Date de pe Raspberry Pi la care este conectat senzorul
GPIO_PIN = 2

class DHT22:
    # Citire senzor
    def readData(self, scale):
        Umiditate, Temperatura = Adafruit_DHT.read_retry(MODEL, GPIO_PIN)

        if scale == 'F':
            Temperatura = Temperatura * 9/5.0 + 32

        return Umiditate, Temperatura
```

Figura 26. Funcția pentru citirea datelor de la senzorul DHT22

#### 5.2.1.5 Modulul de Mail

Modulul de Mail accesează contul de mail al utilizatorului și verifică dacă acesta are mailuri noi. Dacă da atunci le afișează, în caz contrar anunță că nu există mailuri noi.

```
import time
import imaplib
import email

ORG_EMAIL = "@gmail.com"
FROM_EMAIL = "smtp.test.sm" + ORG_EMAIL
FROM_PWD = ""
SMTP_SERVER = "imap.gmail.com"
SMTP_PORT = 993

class MailModule:
    def read_email_from_gmail(self):
        try:
            mail = imaplib.IMAP4_SSL(SMTP_SERVER)

            # Login
            mail.login(FROM_EMAIL, FROM_PWD)

            # Selecteaza folder
            mail.select('inbox')

            # Cauta in folderul INBOX mailuri noi
            result, data = mail.search(None, 'UNSEEN') #ALL / UNSEEN
            mail_ids = data[0]

            if len(data) == 1:
                print "No new Mails"
            else:
                id_list = mail_ids.split()
                first_email_id = int(id_list[0])
                latest_email_id = int(id_list[-1])
```

Figura 27. Citirea informațiilor despre mailuri

### 5.2.1.6 Modulul de Trafic

Modulul de Trafic folosește API-ul de la Google pentru hărți. Se creează un request cu valorile de pe câmpurile Home și Work din baza de date salvate de fiecare user. Google răspunde cu un Json cu toate informațiile necesare.

```
"routes" : [
  {
    "bounds" : {
      "northeast" : {
        "lat" : 44.497902,
        "lng" : 26.1312899
      },
      "southwest" : {
        "lat" : 44.451442,
        "lng" : 26.0626127
      }
    },
    "copyrights" : "Map data ©2018 Google",
    "legs" : [
      {
        "distance" : {
          "text" : "9.1 km",
          "value" : 9140
        },
        "duration" : {
          "text" : "19 mins",
          "value" : 1166
        },
        "end_address" : "Pipera, Voluntari, Romania",
        "end_location" : {
          "lat" : 44.4978247,
          "lng" : 26.1248752
        },
        "start_address" : "Calea Griviței, București, Romania",
        "start_location" : {
          "lat" : 44.451442,
          "lng" : 26.0677281
        },
        "steps" : [
```

Figura 28. Json returnat de API-ul de la Google cu informațiile cerute

La final se parsează Json-ul și se salvează informațiile necesare. După aceasta se afișează informațiile pe oglinda.

### 5.2.2 Implementarea Serverului web

Pentru a se putea crea și configura un profil de către utilizator am creat un server web folosind microframework-ul de Python, Flask. Am început prin a crea rutele aplicației cu ajutorul directivei `@app.route()` care se apelează înainte de definirea acțiunii. Am creat toate acțiunile necesare serverului și le-am asociat cu câte o rută (de exemplu: `/login` cu funcția `login()`) și am creat template-urile asociate fiecăreia. Se poate observa că în directivea `@app.route()` se pot specifica și tipurile de cereri permise (GET, POST, PUT, DELETE).

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'GET':
        return render_template('login.html')

    elif request.method == 'POST':
        # Ia informatiile din formular
        username = request.form['username']
        password_candidate = request.form['password']

        # Ia user-ul din baza de date in functie de userName-ul introdus
        result = mma.GetUser(username)

        # Daca userName-ul exista in baza de date
        if result.userName:
            # Verifica daca parolele coincid
            if result.password == password_candidate:
                session['logged_in'] = True
                session['username'] = username

                return redirect(url_for('settings'))
            # Daca parolele nu coincid
            else:
                return render_template('login.html')

        # Daca userName-ul nu exista in baza de date
        else:
            return render_template('login.html')
```

Figura 29. Funcția de login

### 5.2.3 Implementarea Bazei de date

Pentru ca serverul web să poată comunica cu baza de date am implementat o clasă pe care cu ajutorul ORM-ului Peewee am mapat-o la tabela din baza de date.

```

class User(BaseModel):
    userName = peewee.CharField(unique=True)
    password = peewee.CharField()

    nume = peewee.CharField()
    prenume = peewee.CharField()

    # Informatii necesare modulului de Email
    email = peewee.CharField()
    emailPassword = peewee.CharField()

    # Informatie necesara modulului de Lumina
    color = peewee.CharField()

    # Informatie necesara modulului de Vreme
    weatherType = peewee.CharField()

    # Informatii necesare modulului
    home = peewee.CharField()
    work = peewee.CharField()

```

Figura 30. Tabela user sub formă de obiect Python

În clasa BaseModel se configurează legătură către baza de date. Clasa User extinde această clasă și folosind o clasă specifică Peewee (CharField) se face maparea către câmpul cu același nume din tabela User. Peewee are câte o clasă specifică (FloatField, TimeField, DateField) pentru fiecare tip SQL.

Peewee ne ajută să facem interogări ale bazei de date fără a folosi cod SQL. Astfel în loc să scriem un select clasic de SQL putem scrie User.get(). La fel de simplu este și ștergerea totul reducându-se la delete\_instance().

```

def GetUser(self, userName):
    dbUser = User.get(User.userName == userName)
    return dbUser

```

Figura 31. Aducerea unui user din baza de date în funcție de userName

```

def DeleteUser(self, userName):
    dbUser = User.get(User.userName == userName)
    dbUser.delete_instance()

```

Figura 32. Ștergerea unui user din baza de date în funcție de userName

## 6 EVALUARE

### 6.1 Corectitudinea implementării

#### 6.1.1 Corectitudinea recunoașterii utilizatorului

Modulul de recunoaștere facială funcționează cu o precizie destul de bună. Acesta a fost testat la final fiind antrenat cu 2 utilizatori și o poză de pe internet cu încă un utilizator. Recunoașterea făcându-se în timp real s-au observat erori pe anumite frame-uri, astfel existând riscul ca eroarea să apară pe primul frame și oglinda să creadă că alt utilizator o folosește. Pentru combaterea acestei posibilități s-a introdus o nouă funcție și s-a regândit sistemul astfel încât înainte de a returna user-ul găsit oglinda scanează de 20 de ori și face o medie a rezultatelor astfel chiar dacă avem eroare pe primele frame-uri oglinda va returna un rezultat corect.

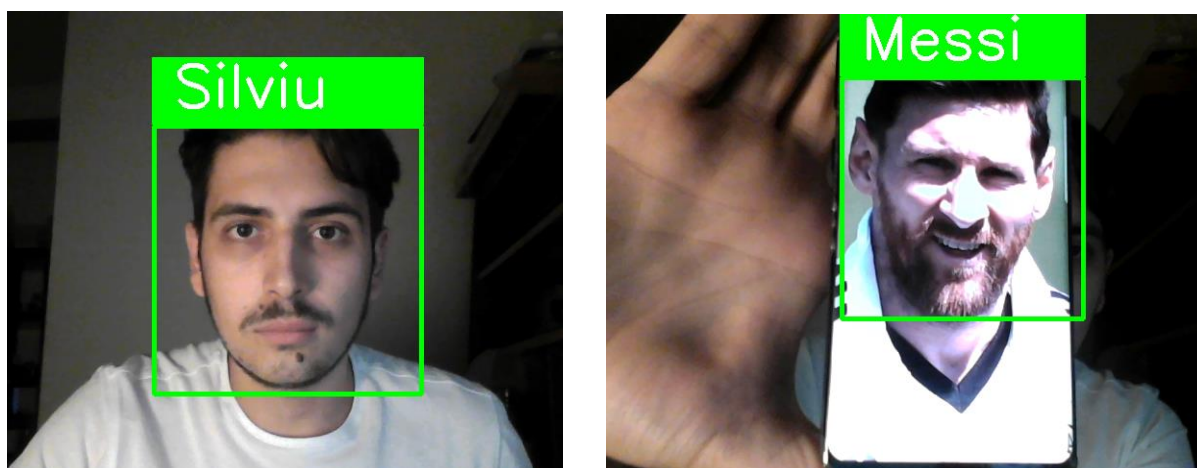


Figura 32,33. Recunoaștere utilizatori

#### 6.1.2 Corectitudinea afișării informațiilor

După ce utilizatorul a fost recunoscut pe ecran au apărut și informațiile dorite de acesta în formatele specificate.





Figura 34. Screen shot cu modulul de vreme

### 6.1.3 Corectitudinea măsurării temperaturii și umidității

În final a fost verificat senzorul DHT22 pentru a verifica corectitudinea informațiilor oferite de acesta. După cum am mai spus senzorul are 2 roluri, importanța testării fiind pentru cel de al doilea rol al său și anume acela că sistemul are o protecție când temperatura este prea mare sau umiditatea prea crescută, acesta se închide pentru a preveni eventualele pagube.

A fost supus la mai multe măsurători în paralel cu un termometru clasic. Temperatura afișată este aproximativ aceeași fiind o marjă foarte mică de eroare. Umiditatea nu am putut să o compar deoarece nu am avut un higrometru disponibil dar am făcut teste pentru a observa dacă măsurătorile își schimbă valoarea teste fiind mai mult decât concludente.

## 6.2 Respectarea cerințelor

La finalul implementării, proiectul a fost prezentat potențialilor utilizatori în vederea obținerii unui feed-back. Feed-back-ul a fost unul pozitiv, proiectul satisfăcând în principiu toate cerințele utilizatorilor.

Totul funcționează conform cerințelor, oglinda pornește în momentul în care este băgată în priză, recunoașterea facială detectează prezența utilizatorului și recunoaște profilul său oferind informații personalizate astfel oglinda putând fi folosită de mai mulți utilizatori iar aplicația de configurare este foarte simplă și ușor de folosit, prețul final fiind de doar 750 RON mult sub prețul oglinzilor inteligente din comerț.

## 7 CONCLUZII

### 7.1 Concluzii generale

În cele de mai sus, am descris un concept nou de oglindă inteligentă, care, în plus față de modelele anterioare, integrează funcția de recunoaștere facială. În același timp, am reușit să propun un model care să ofere opțiuni de bază, astfel încât costurile realizării unei astfel de oglinzi să fie minime, cu intenția promovării ideii unei vieți *smart* la scară cât mai largă. Ulterior, se vor putea adăuga servicii și funcții adiționale la un cost suplimentar, elemente ce vor permite o interacțiune sporită obiect-utilizator (cum ar fi rularea unui canal de muzică/ Youtube sau a unei pagini web, precum și a altor aplicații), însă aceste funcții adiționale nu constituie obiectul lucrării de față.

### 7.2 Dezvoltări ulterioare

Proiectul a ieșit foarte bine și îmi doresc dezvoltarea lui în continuare deoarece îl folosesc în viața de zi cu zi. Următoarele dezvoltări la care voi lucra vor fi:

- Implementarea unui modul de interacțiune prin gesturi cu oglinda
- Adăugarea unui modul de muzică
- Posibilitatea conectării telefonului la ea

După ce aceste dezvoltări vor fi terminate și funcționale plănuiesc să ofer oglinzii o altfel de funcționalitate și anume să o transform într-un hub pentru Smart Home, aceasta fiind centrul de comandă a altor dispozitive smart din casa.

## 8 REFERINȚE

- [1] „Allview Mirror”, allviewsmarthome.ro, 2018. [Online] Disponibil: <http://www.allviewsmarthome.ro/images/mirror/mirror.jpg> [Accesat 02.06.2018]
- [2] „Magic Mirror<sup>2</sup>”, michaelteeuw.nl, 2018. [Online] Disponibil: <http://michaelteeuw.nl/post/84026273526/and-there-it-is-the-end-result-of-the-magic> [Accesat 02.06.2018]
- [3] „HomeMirror”, Hannah Mitt, 2018. [Online] Disponibil: <https://github.com/HannahMitt/HomeMirror> [Accesat 02.06.2018]
- [4] „Smart Mirror”, Evan Cohen, 2018. [Online] Disponibil: <https://smart-mirror.io/> [Accesat 02.06.2018]
- [5] „PANL”, Ryan Nelwan, 2018. [Online] Disponibil: <https://getpanl.com/> [Accesat 02.06.2018]
- [6] „Raspberry Pi 3 Model B”, raspberrypi.org, 2018. [Online] Disponibil: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/> [Accesat 02.06.2018]
- [7] „PI NOIR CAMERA V2” raspberrypi.org, 2018. [Online] Disponibil: <https://www.raspberrypi.org/products/pi-noir-camera-v2/> [Accesat 02.06.2018]
- [8] „DHT22”, optimusdigital.ro, 2018. [Online] Disponibil: [https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/1449-modul-senzor-de-temperatura-i-umiditate-dht22.html?search\\_query=dht+22&results=14](https://www.optimusdigital.ro/ro/senzori-senzori-de-temperatura/1449-modul-senzor-de-temperatura-i-umiditate-dht22.html?search_query=dht+22&results=14) [Accesat 02.06.2018]
- [9] „Reflectiv MIR”, trimax.ro, 2018. [Online] Disponibil: [http://www.trimax.ro/779-thickbox\\_default/reflectiv-mir-folie-colorata-tip-oglanda-one-way.jpg](http://www.trimax.ro/779-thickbox_default/reflectiv-mir-folie-colorata-tip-oglanda-one-way.jpg) [Accesat 02.06.2018]
- [10] „Radiatoare”, optimusdigital , 2018. [Online] Disponibil: [https://static.optimusdigital.ro/5886-thickbox\\_default/set-3-radiatoare-pentru-raspberry-pi-3.jpg](https://static.optimusdigital.ro/5886-thickbox_default/set-3-radiatoare-pentru-raspberry-pi-3.jpg) [Accesat 02.06.2018]
- [11] „Ventilator”, optimusdigital , 2018. [Online] Disponibil: [https://static.optimusdigital.ro/10407-thickbox\\_default/ventilator-pentru-raspberry-pi-3.jpg](https://static.optimusdigital.ro/10407-thickbox_default/ventilator-pentru-raspberry-pi-3.jpg) [Accesat 02.06.2018]

- [12] „Raspbian logo”, raspbian.org , 2018. [Online] Disponibil: [https://www.raspbian.org/static/common/raspbian\\_logo.png](https://www.raspbian.org/static/common/raspbian_logo.png) [Accesat 02.06.2018]
- [13] „Python logo”, python.org , 2018. [Online] Disponibil: <https://www.python.org/static/img/python-logo.png> [Accesat 02.06.2018]
- [14] „Raspberry Pi 3 Model B GPIO 40 Pin Block Pinout” , element14.com, 2018. [Online] Disponibil: <https://www.element14.com/community/docs/DOC-73950/l/raspberry-pi-3-model-b-gpio-40-pin-block-pinout> [Accesat 12 06 2018]
- [15] „OpenCV logo”, opencv.org , 2018. [Online] Disponibil: <https://opencv.org/assets/theme/logo.png> [Accesat 02.06.2018]
- [16] „Flask logo”, flask.pocoo.org , 2018. [Online] Disponibil: <http://flask.pocoo.org/static/logo/flask.png> [Accesat 02.06.2018]
- [17] „Peewee logo”, docs.peewee-orm.com , 2018. [Online] Disponibil: [http://docs.peewee-orm.com/en/latest/\\_images/peewee3-logo.png](http://docs.peewee-orm.com/en/latest/_images/peewee3-logo.png) [Accesat 02.06.2018]
- [18] „SQLite logo”, sqlite.org , 2018. [Online] Disponibil: [https://www.sqlite.org/images/sqlite370\\_banner.gif](https://www.sqlite.org/images/sqlite370_banner.gif) [Accesat 02.06.2018]
- [19] „Sistem înglobat”, wiki[edia.org, 2018. [Online] Disponibil: [https://ro.wikipedia.org/wiki/Sistem\\_%C3%AEnglobat](https://ro.wikipedia.org/wiki/Sistem_%C3%AEnglobat) [Accesat 19.05.2018]

## 9 BIBLIOGRAFIE

- [1] „Our Documentation”, python.org, 2018. [Online] Disponibil: <https://www.python.org/doc/> [Accesat 12 04 2018]
- [2] „User’s Guide”, pocoo.org, 2018. [Online] Disponibil: <http://flask.pocoo.org/docs/1.0/> [Accesat 12 04 2018]
- [3] „DHT22 Datasheet”, sparkfun.com, 2018. [Online] Disponibil: <https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf> [Accesat 13 03 2018]
- [4] „PeeWee Documentation”, docs.peewee-orm.com, 2018. [Online] Disponibil: <http://docs.peewee-orm.com/en/latest/> [Accesat 13 03 2018]
- [5] „Bootstrap Documentation”, getbootstrap.com, 2018. [Online] Disponibil: <https://getbootstrap.com/> [Accesat 18.02.2018]
- [6] „OpenCV Documentation”, docs.opencv.org, 2018. [Online] Disponibil: [https://docs.opencv.org/3.4.1/d0/de3/tutorial\\_py\\_intro.html](https://docs.opencv.org/3.4.1/d0/de3/tutorial_py_intro.html) [Accesat 19.05.2018]

## 10 ANEXE

Studiul de piață a fost realizat în urma discuțiilor cu un număr de 27 de persoane. În cazul participanților mai puțin informați, am purtat mai întâi conversații prin care am explicat conceptele de baza ale unei oglinzi inteligente. Am parcurs împreună cu fiecare din participanți următorul set de întrebări:

1. Ai fi interesat să achiziționezi sau să folosești un *smart mirror*?
2. Cum ți-ai dori să se pornească oglinda? (Ți-ai dori să funcționeze după modelul *Plug & Play* sau crezi că altă metodă de funcționare ar fi mai utilă/ eficientă?)
3. Ai dori să poți fi singurul utilizator al oglinzii sau ești de acord că e util ca mai multe persoane să se poată conecta la un profil individual de pe același dispozitiv?
4. În caz că dorești ca mai multe persoane să aibă acces la un profil de pe același dispozitiv, ar trebui ca la fiecare conectare să fie nevoie de o reconfigurare?
5. Ce preț ai fi dispus să investești într-o oglindă smart?
  - a. până în 1 000 RON;
  - b. între 1000 și 2 500 RON;
  - c. între 2 500 și 4 000 RON;
  - d. peste 4 000 RON.
6. Cum dorești să se facă depanarea și updatarea oglinzii? (spre exemplu: updatare automată, depanare în reprezentanțe/ la centre speciale etc.)
7. Ai fi dispus să investești timp pentru a citi manualul de folosire sau preferi un mod de configurare intuitiv, eventual cu suport alcătuit din mici mesaje și instrucțiuni care apar pe ecran?
8. Ce așteptări ai de la interfața unui astfel de produs?
9. De pe ce dispozitiv electronic ți-ar plăcea să îți poți face configurarea oglinzii?
10. Vrei să adaugi ceva? Mai sunt alte aspecte care crezi că ar fi important de luat în considerare?