

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

**FACULTATEA DE INFORMATICA**



LUCRARE DE LICENTA

**Home Smartify - soluție pentru controlul  
dispozitivelor și securitatea casei**

propusă de

**Liviu-Andrei Petrache**

Sesiunea: iunie, 2023

Coordonator științific

**Lect. dr. Vidrașcu Cristian**

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

**FACULTATEA DE INFORMATICA**

**Home Smartify - soluție pentru  
controlul dispozitivelor și securitatea  
casei**

**Liviu-Andrei Petrache**

**Sesiunea:** iunie, 2023

Coordonator științific

**Lect. dr. Vidrașcu Cristian**

Avizat,

Îndrumător lucrare de licență,

Lect. dr. Vidrașcu Cristian.

Data: .....

Semnătura: .....

## **Declarație privind originalitatea conținutului lucrării de licență**

Subsemnatul **Petrache Liviu-Andrei** domiciliat în România, jud. Iași, mun. Iași, strada Stejar, numărul 41, bloc O10, scara A, apartament 3, parter, născut la data de 04 ianuarie 2001, identificat prin CNP 5010104226711, absolvent al Facultății de informatică, **Facultatea de informatică** specializarea **informatică**, promoția 2023, declar pe propria răspundere cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art. 143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul **Home Smartify - soluție pentru controlul dispozitivelor și securitatea casei** elaborată sub îndrumarea domnului **Lect. dr. Vidrașcu Cristian**, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului ei într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data: .....

Semnătura: .....

## **Declarație de consumămant**

Prin prezenta declar că sunt de acord ca lucrarea de licență cu titlul **Home Smar-tify - soluție pentru controlul dispozitivelor și securitatea casei**, codul sursă al pro-gramelor și celealte conținuturi (grafice, multimedia, date de test, etc.) care însășesc această lucrare să fie utilizate în cadrul Facultății de informatică.

De asemenea, sunt de acord ca Facultatea de informatică de la Universitatea "Alexandru-Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent **Liviu-Andrei Petrache**

Data: .....

Semnătura: .....

# Cuprins

<b>Motivătie</b>	<b>2</b>
<b>Introducere</b>	<b>4</b>
<b>1 Aplicații asemănătoare</b>	<b>5</b>
1.1 Home Assistant . . . . .	5
1.2 Apple Home . . . . .	6
1.3 IFTTT . . . . .	7
<b>2 Tehnologii utilizate</b>	<b>9</b>
2.1 Flutter . . . . .	9
2.2 Flask . . . . .	10
2.3 Raspberry Pi Zero W . . . . .	10
2.4 Microcontrolere folosite . . . . .	11
2.4.1 Arduino Nano . . . . .	12
2.4.2 ESP8266 . . . . .	12
2.5 Module . . . . .	13
2.6 Motivatia tehnologiilor folosite . . . . .	14
<b>3 Implementarea soluției</b>	<b>15</b>
3.1 Pagina principală . . . . .	16
3.1.1 Cărțile senzorilor . . . . .	16
3.1.2 Locația senzorului . . . . .	17
3.1.3 Modul de securitate . . . . .	18
3.1.4 Butonul de refresh . . . . .	19
3.2 Pagina senzorului static . . . . .	20
3.3 Pagina senzorului dinamic . . . . .	21
3.4 Pagina de informații . . . . .	22

3.5	Pagina de genereare de rapoarte . . . . .	24
3.6	Modul debug . . . . .	26
3.7	Frontend . . . . .	27
3.8	REST API . . . . .	27
3.8.1	DNS dinamic . . . . .	28
3.9	Backend . . . . .	28
3.9.1	Stocarea datelor . . . . .	30
3.10	Senzori statici . . . . .	31
3.11	Senzori dinamici . . . . .	32
3.12	Versionarea . . . . .	33
<b>4</b>	<b>Scenarii de utilizare</b>	<b>34</b>
4.1	Colectare de date și logging . . . . .	34
4.2	Comutarea luminilor din holul intrării . . . . .	34
4.3	Detectarea problemelor în comunicare . . . . .	35
4.4	Generarea raportului . . . . .	35
<b>Concluzii</b>		<b>36</b>
<b>Bibliografie</b>		<b>38</b>

# Motivație

Lucrarea de licență intitulată "**Home Smartify - soluție pentru controlul dispozitivelor și securitatea casei**" prezintă cum prin integrarea dispozitivelor "**Internet of things**"<sup>1</sup>, se poate ajunge la un nivel ridicat de control asupra locuinței personale: monitorizarea mișcărilor dintr-o casă, conectarea/deconectarea unui consumator de la alimentare, toate fiind ușor de executat cu ajutorul aplicației mobile.

Home Smartify a fost adusă la viață datorită dorinței personale de a-mi monitoriza și controla apartamentul într-un mod ușor, intuitiv și privat. Jucându-mă cu plăcile **Raspberry Pi**<sup>2</sup> împreună cu diferiți senzori, am reușit să creez un sistem ce securizează ușa camerei mele cu ajutorul unui cod numeric, roboți ce urmează un traseu predefinit și prize inteligente. Problema pe care am întâmpinat-o a fost că nu exista o entitate ce să comande fiecare modul în parte, eu fiind responsabil de a trimite informațiile necesare către modulul respectiv. Observând utilitatea acestei funcționalități și faptul că mă distrez în timp ce programez și conectez componentele între ele prin fire, am decis să realizez un sistem ce poate "*vorbi*" cu aceste module, putând fi comandate din aplicația de telefon.

Înțial, ideea a fost concepută doar cu latura de monitorizare, folosind un singur tip de senzori. Oferit pe post de feedback de către domnul profesor coordonator Cristian Vidrașcu, am realizat că implementarea unei modalități de control a acestor dispozitive adaugă o nouă arie de funcționalitate și creativitate în gestionarea sarcinilor din casă.

---

<sup>1</sup><https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT>.

<sup>2</sup><https://www.raspberrypi.org/>.

# Contribuții

Soluțiile de a face o casă "smart" cresc zilnic în număr, astfel am optat pentru a crea una originală, de la framework-uri, până la modulele hardware pe care le-am folosit.

Aplicația de mobil "Home Smartify" a luat naștere ca răspuns al întrebării "*Ce pot face pentru a-mi automatiza casa?*". Având la îndemâna diferite componente și senzori, am decis să creez o aplicație de telefon care să controleze aceste dispozitive de la distanță.

De adaugat continuarea

Aș dori să mulțumesc domnului profesor coordonator Cristian Vidrașcu pentru libertatea alegerii temei licenței și profesionalismul de care a dat dovadă.

# Introducere

Lucrarea **"Home Smartify - soluție pentru controlul dispozitivelor și securitatea casei"** aduce în prim plan o abordare inovatoare pentru a facilita și îmbunătăți modul de viață al utilizatorului în cadrul casei sale. În era tehnologiei inteligente, unde conectivitatea și controlul la distanță devin tot mai importante, această soluție oferă consumatorului posibilitatea de a avea controlul complet asupra dispozitivelor și senzorilor din propria casă, indiferent de locația în care se află. Securitatea datelor utilizatorului este cea mai mare prioritate în dezvoltarea acestei aplicații, fiind stocate în totalitate în mediul local. O abordare inovatoare și ușor de utilizat pentru tehnologia Smart Home este propusă în această lucrare, care contribuie semnificativ la îmbunătățirea atât a confortului, cât și a securității locuinței.

Structura proiectului este afișată în următorul tabel ce încearcă să ajute la înțelegerea în detaliu a aplicației prin descrierea sumară a fiecărui capitol.

- Capitolul I. Prezentarea unor aplicații similare utilizate în domeniul tehnologiei Smart Home, cum ar fi Home Assistant, Apple Home și IFTTT.
- Capitolul II. Descrierea tehnologiilor utilizate în implementarea soluției, cum ar fi Flutter SDK, Flask, Raspberry Pi Zero W și microcontrolerele Arduino Nano și ESP8266.
- Capitolul III. Prezentarea implementării soluției Home Smartify ce conține aplicația de mobil cu paginile principale, modul debug, precum și elemente legate de Application Programming Interface și echipamentul senzorilor.
- Capitolul IV. Descrierea scenariilor de utilizare ale aplicației, cum ar fi colectarea de date, comutarea luminilor din holul intrării, generarea raportului și detectarea problemelor în comunicare.
- Concluzii. Imbuimatiri pe viitor

# Capitolul 1

## Aplicații asemănătoare

Acest capitol se focusează asupra prezentării unor soluții asemănătoare ca idee sau ca și tehnologie cu proiectul **Home Smartify**.

În prezent, există numeroase astfel de platforme, fiind alese cele mai preferate în rândul utilizatorilor care doresc să își adapteze casa după preferințele personale.

### 1.1 Home Assistant

Home Assistant (Figura 1.1) este o platformă open-source<sup>1</sup> pentru controlul și automatizarea locuințelor smart. Oferind un mediu centralizat pentru gestionarea dispozitivelor și serviciilor, ea suportă peste 1000 de servicii și device-uri diferite pe care le identifică prin scanarea inițială a rețelei de internet la care sunt conectate.

Orice utilizator are asignat o *amprentă digitală*<sup>2</sup> care este folosită spre a îl identifica în mediul online. Confidentialitatea constituie unul din punctele forte ale aplicației Home Assistant, astfel că păstrează toate informațiile în domeniul local și evită pe cât posibil comunicarea cu alte servicii din cloud.

Este recomandat ca acesta să fie instalat pe o platformă Raspberry Pi prin folosirea sistemului de operare pus la dispoziție pe website sau poate fi executat într-un *container Docker*<sup>3</sup>, metodă similară cu cea implementată în această licență.

---

<sup>1</sup><https://opensource.com/resources/what-open-source>.

<sup>2</sup><https://www.kaspersky.com/resource-center/definitions/what-is-a-digital-footprint>.

<sup>3</sup><https://www.docker.com/>.



Figura 1.1: Interfața meniului principal, temperatură, grafice, consum energie.

<https://rbx.dk/w/home-assistant/hvad-er-home-assistant.html>

## 1.2 Apple Home

Apple este una dintre primele companii care au încercat să integreze controlul casei smart într-o aplicație mobile, lansând HomeKit (Figura 1.2) integrat în iOS 8 care a apărut în septembrie 2014.

Cu suport de peste 100 brand-uri de produse, printre care termostate, prize inteligente, lumini și jaluzele, utilizatorii de Mac, Iphone, Ipad își pot automatiza sarcinile, numite *Scenes* (care pot fi separate pe zone), prin apăsarea unui buton din interfața ușor de înțeles sau pot opta pentru integrarea cu asistentul vocal Siri.

Cristopher Null<sup>4</sup> de la TechHive afirmă că procesul de instalare este impresionant, dar folosirea acestui program zilnic poate fi *mediocru*, fiind clasificată ultima când vine vorba de control din cauza pierderii semnalului și esuarea executării task-urilor. Speră ca aplicația să primească update-uri și noi funcționalități, interacțiunea sa cu Apple Home lăsând mult de dorit. Se poate observa numărul de 10 ori mai mic de dispozitive suportate față de Home Assistant, fiind o barieră destul de observabilă atunci când se pune problema extinderii către alte funcționalități.

<sup>4</sup><https://www.techhive.com/article/579157/essential-homekit-guide.html>.

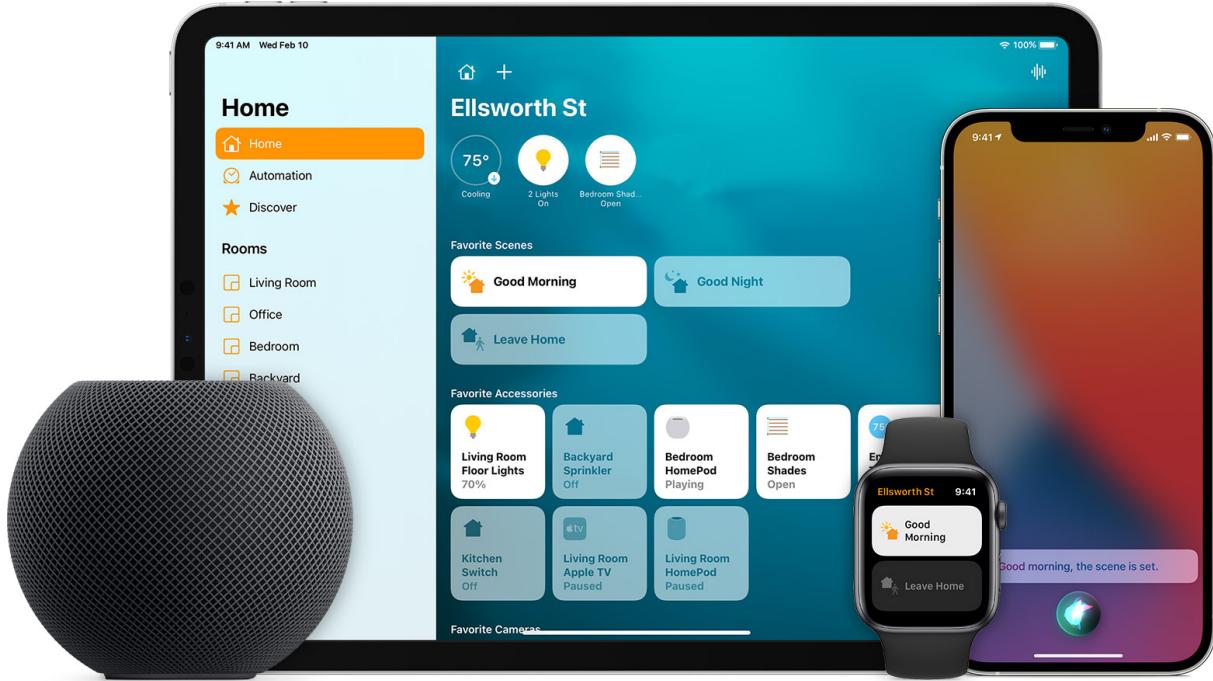


Figura 1.2: Meniul principal împreună cu dispozitivele ce sunt în control.

<https://support.apple.com/en-us/HT208280>

### 1.3 IFTTT

Fie că dorești să fii notificat pe telefon cu o zi înaintea unui eveniment sau că vrei ca asistentul Google să îți schimbe intensitatea luminilor din garaj, acestea sunt exemplele de bază pe care platforma *IF this then that* (prescurtat IFTTT) (Figura 1.3) le poate realiza.

Acesta este un serviciu ce îți oferă posibilitatea de a conecta aplicații, servicii și dispozitive fără a fi preocupat de programarea lor. Website-ul IFTT și aplicația de telefon te ajută să construiești comenzi (pe care IFTTT obișnuia să le numească *rețete*, acum sunt *applets*) folosind card-uri colorate puternic pentru identificare.

Odată intrat pe pagina de start, se pot vedea applet-urile care sunt conectate cu ajutorul informațiilor de la capătul cardului. Pentru crearea unui astfel de item, trebuie specificat serviciul ce declanșează automatizarea respectivă. Odată ce se întâmplă acel trigger (*if this*), se vor executa acțiunile definite de către utilizator (*then that*). Ex: dacă ai adăugat o melodie nouă într-un playlist pe Spotify, adaug-o și pe Apple Music.

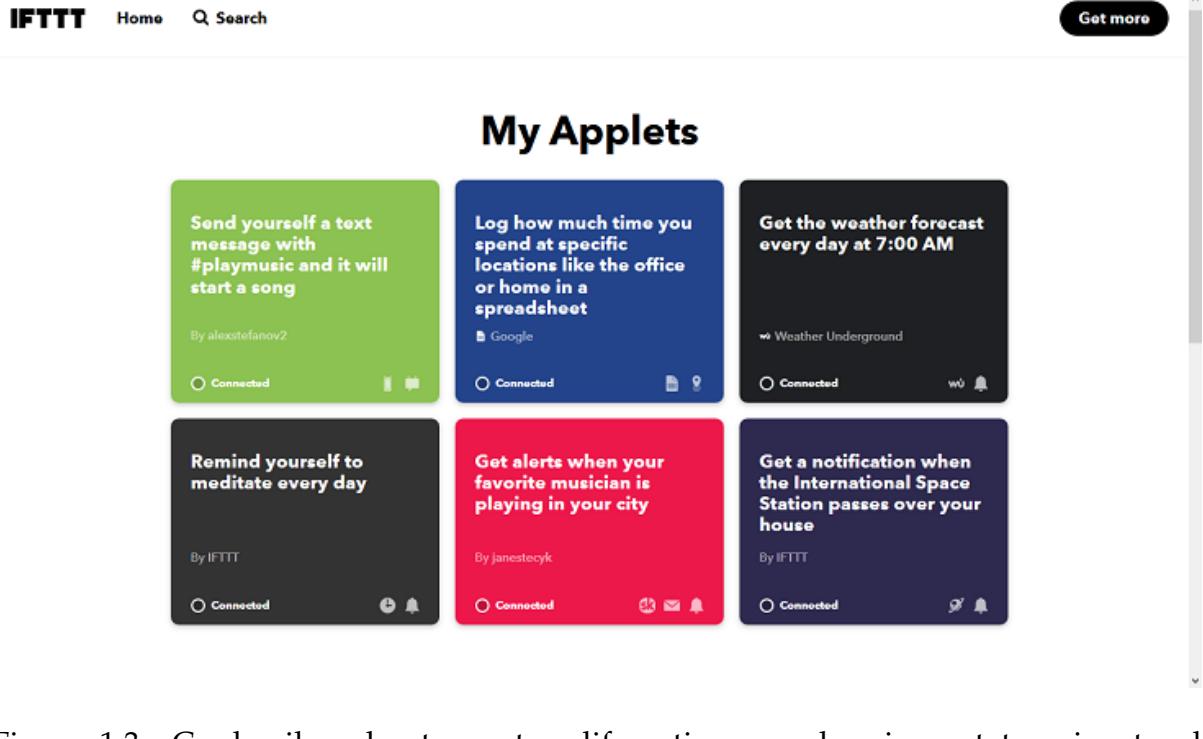


Figura 1.3: Cardurile colorate pentru diferențiere, cu descriere, status și autorul informației.

<https://www.pcmag.com/reviews/ifttt>

Prin intermediul ideilor prezentate precedent, putem observa că soluțiile pentru o casă smart sunt din ce în ce mai populare. Este foarte probabil ca un utilizator să dețină dispozitive IoT pe care le poate inter-conecta cu scopul de a obține o utilizare eficientizată a spațiului de lucru.

Soluțiile pentru o casă smart oferă oportunități diverse de a îmbunătăți controlul și automatizarea locuinței. Fiecare platformă are propriile caracteristici și avantaje, iar utilizatorii pot alege în funcție de nevoile și preferințele lor specifice.

# Capitolul 2

## Tehnologii utilizate

### 2.1 Flutter

**Flutter** este o platformă pentru dezvoltare de aplicații publicată inițial în 2017. Cu ajutorul acestui **SDK** (*Software Development Kit*), programatorii pot construi aplicații web, desktop și cross-platform pentru Android și IOS. Se folosește împreună cu limbajul de programare Dart.

Pentru a dobândi o eficiență asemănătoare celei native, este folosită compilarea **AOT** (*ahead-of-time*) pe toate platformele în afară de Web, unde codul este convertit în JavaScript.

Fundația acestui SDK este scrisă în **C++**, oferă randare low-level folosind biblioteca *Skia* de la Google sau un strat de abstractizare grafic *Impeller*. Componenta de bază în Flutter este *widget-ul* care poate conține și alte widget-uri. Aceasta descrie logica, interacțiunea și design-ul unei componente UI. Flutter conține două pachete de elemente UI: Material Design (Google) și Cupertino (Apple).

```
1 return Scaffold(  
2   body: Column(  
3     children: [  
4       Text('Acest text va fi pozitionat'),  
5       Text('Deasupra acestui text'),  
6     ],  
7   ),  
8 );
```

Listing 2.1: Widget Scaffold

## 2.2 Flask

Flask este un micro-framework web ușor de utilizat și flexibil, scris în Python. A fost creat cu scopul de a fi simplu și minimalist, oferind totodată funcționalitățile esențiale necesare pentru dezvoltarea rapidă a aplicațiilor. Acesta permite definirea de rute pentru a măpa cererile HTTP/HTTPS primite către funcții. Prin intermediul adnotărilor, se pot atribui funcțiilor URL-uri și metode HTTP/S specifice, permitând astfel aplicației să proceseze cererea în funcție de parametrii primiți.

Datorită abordării sale minimalistă și comunității active, Flask a devenit unul dintre cele mai populare micro-framework-uri pentru dezvoltarea aplicațiilor web în Python.

Figura 2.2 reprezintă un script ce expune o rută care returnează un paragraf HTML cu textul *Hello, World!*.

```
1 from flask import Flask
2 app = Flask(__name__)
3 @app.route("/")
4 def hello_world():
5     return "<p>Hello, World!</p>"
```

Listing 2.2: Exemplu minimal de aplicație Flask

## 2.3 Raspberry Pi Zero W

Raspberry Pi este o familie de calculatoare, de mărimea unui card de credit sau mai mici, care au revoluționat industria cu prețul accesibil de doar \$25 al primei plăci. În ultimii 5 ani, acestea au dobândit o atenție foarte mare în rândul comunității *Do it yourself (DIY)* datorită spectrului larg de utilizări în proiecte digitale și IoT.

Comunitatea Raspberry a crescut mult și consistent. În momentul de față, în urma căutării "Raspberry pi projects", ne sunt returnate 47.2 milioane de pagini care conțin tutoriale despre imprimante 3D, sisteme de irigare, de luat vederi, ad-blocker pentru rețea și multe altele.

În contextul versiunii Zero W (Figura 2.1), avem disponibilă o placă de mărimea unui pachet de gume: 6.5 x 3 x 0.5 cm pe care poate fi instalat un sistem de operare bazat pe GNU/Linux numit Raspbian OS. Are un procesor single-core de 1GHz, 512 MB LPDDR2 RAM, ieșire mini HDMI, două porturi micro USB, Bluetooth 4.0 și 2.4GHz

802.11n Wi-Fi.

Aspectul special al acestor computere constă în cei 40 pini **GPIO** (*General purpose input-output*). Aceștia sunt folosiți pentru conectarea de senzori, fiind ușor programabili cu ajutorul limbajului Python, sau pentru audiente tinere, Scratch.

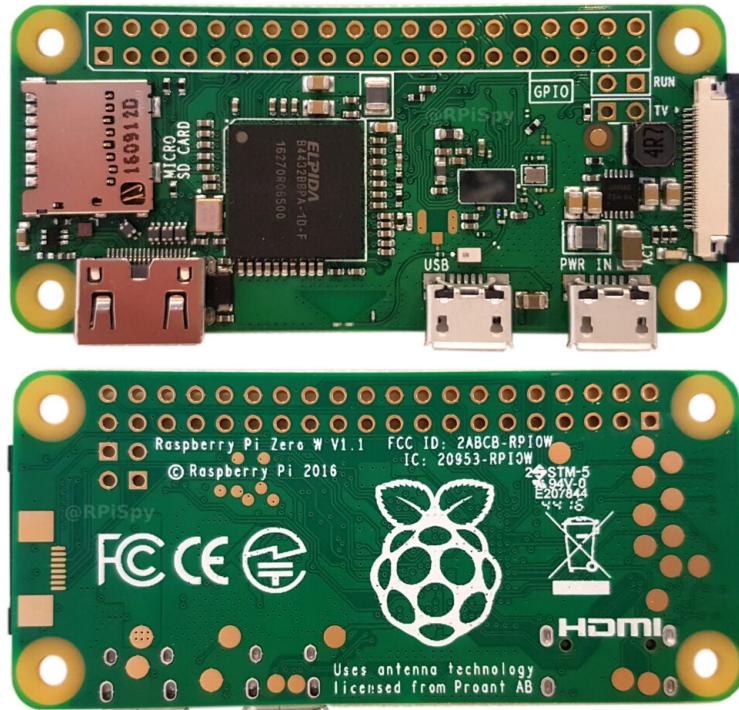


Figura 2.1: Față și spatele plăcii Raspberry Pi Zero W

<https://www.raspberrypi-spy.co.uk/>

## 2.4 Microcontrollore folosite

Un microcontroller (abreviat *MCU*) este un circuit integrat care combină funcționalitatea microporcesorului (de obicei aflat sub un radiator de aluminiu pentru disiparea căldurii) cu componente periferice: module *Input/output*, memorie și interfețe de comunicare cu alte module. Dețin mai mulți pini de tip analog, digital și **PWM** (*Pulse Width Modulation*) pentru conectarea cu alte module

### 2.4.1 Arduino Nano

Arduino Nano (Figura 2.2) este fratele mai mic al placii Uno cu care împărtășește majoritatea funcționalității: microcontroller ATmega328, 32KB memorie flash, viteza de 16MHz și 22 pini I/O. Singurele diferențe sunt mărimea redusă și portul de USB mini. Este perfect pentru persoanele care doresc să învețe tainele electronicii și a programării, fiind potrivit pentru proiecte ce au constrângeri legate de spațiul disponibil.

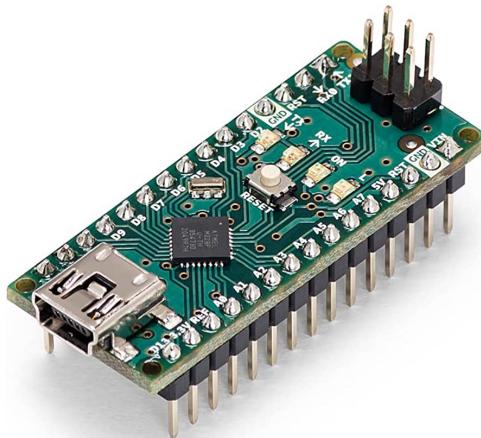


Figura 2.2: Arduino Nano

<https://store.arduino.cc/collections/boards/products/arduino-nano>

### 2.4.2 ESP8266

Modulul ESP8266 (Figura 2.3) este un **SOC** (*System on a chip*) produs de firma Espressif Systems ce folosește stack-ul TCP/IP cu ajutorul căruia se poate conecta la internet.

Acesta a fost popularizat în anul 2014 de către alt SOC asemănător **ESP-01** al firmei **Ai-Thinker**. Chip-urile se pot conecta la rețele Wifi și să realizeze conexiuni prin utilizarea comenziilor de tipul *Hayes, AT commands*.

La început, nu existau documentații în limba engleză. Dat fiind faptul că SoC-ul este ieftin și detine un număr mic de componente externe, acesta poate fi produs în volume mari la preț mic, ceea ce a atras mulți hackeri să-l exploreze și să traducă documentația.

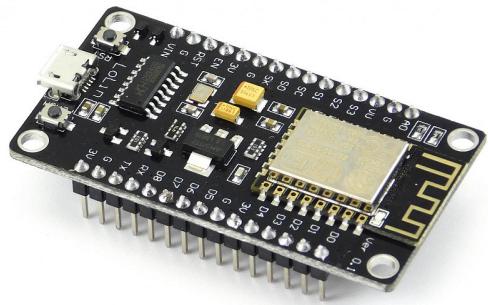


Figura 2.3: ESP8266 pe o placă *Breakout*

[https:](https://)

//www.teachmemicro.com/esp8266-spiffs-web-server-nodemcu/

## 2.5 Module

Modulul de comutare este echipat cu 2 relee (Figura 2.4a) controlate individual. Acesta poate fi utilizat împreună cu orice placă de dezvoltare care dispune de 2 pini digitali și un pin VCC de 5 V. Produsul este util în multe proiecte de electronică în care trebuie controlate diferite dispozitive care sunt alimentate cu o tensiune maximă de 250 V AC sau 30 V DC.

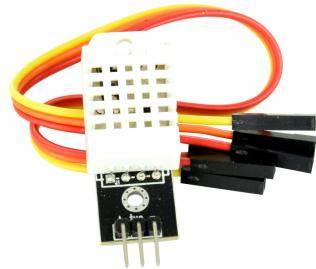
Senzorul DHT22 (Figura 2.4b) este un senzor de umiditate și temperatură mic, ieftin și la îndemână. Folosește un senzor capacitive de umiditate și un termistor pentru a măsura aerul din jur. Acesta oferă semnal digital pe pinul de date. Este ușor de folosit, dar este nevoie de atenție pentru citirea datelor la momentul potrivit.

Senzorul infraroșu HC-SR501 (Figura 2.4c) este folosit pentru a detecta prezența oamenilor. Este des utilizat în aprinderea sau stingerea automată a luminii într-o încăpere atunci când o persoană ajunge sau părăsește o incintă. Suportă o tensiune de alimentare mai ridicată (5 – 20 V ) și are un consum mai mare. Are o rază de sensibilitate de 110° și detectează obiecte până la o distanță de 7m.

Placa HC-SR04 (Figura 2.4d) deține un emițător și un receptor ultrasonic ce ne pot oferi distanța până la cel mai apropiat obstacol într-un mod ușor de programat. Este compatibilă cu o gamă variată de plăci de dezvoltare, având unele avantaje față de senzorii de distanță de tip analog: are nevoie doar de pini I/O digitali, are toleranță mai mare la zgomotul din circuit și precizia de până la 4.5 metri, toate la un consum continuu de maxim 2mA.



(a) Placă cu două relee pentru comutarea tensiunii prizei



(b) Senzor de temperatură și umiditate



(c) Senzor de detectare a mișcării



(d) Senzor de măsurare a distanței

Figura 2.4: Senzorii folosite în acest proiect

## 2.6 Motivatia tehnologiilor folosite

Domeniul Smart Home este foarte vast și oferă o gamă largă de oportunități creative. Tehnologia a evoluat rapid, permitând controlul și automatizarea diverselor aspecte ale locuinței, de la iluminat și securitate la gestionarea energiei și confortul locatarilor.

Modulele și plăcile utilizate se pliază pe arhitectura aplicatiei implementate, dar cum tehnologia avanseaza pe zi ce trece, va exista posibilitatea ca in viitor sa apară device-uri care pot creste performanta. Prin urmare, în domeniul caselor inteligente, nu există limite, iar utilizatorii sunt încurajați să-și folosească imaginația și să exploreze noi soluții ce se pliază vieții personale.

# Capitolul 3

## Implementarea soluției

Prin parcurgerea acestui capitol, cititorul va avea o imagine de ansamblu asupra modului în care senzorii sunt integrați în aplicație, cum este realizată comunicarea cu serverul și care sunt aspectele legate de programarea și funcționalitatea aplicației mobile. Aceste informații sunt esențiale pentru a înțelege arhitectura (Figura 3.1) și funcționalitatea aplicației în cadrul sistemului smart home propus.

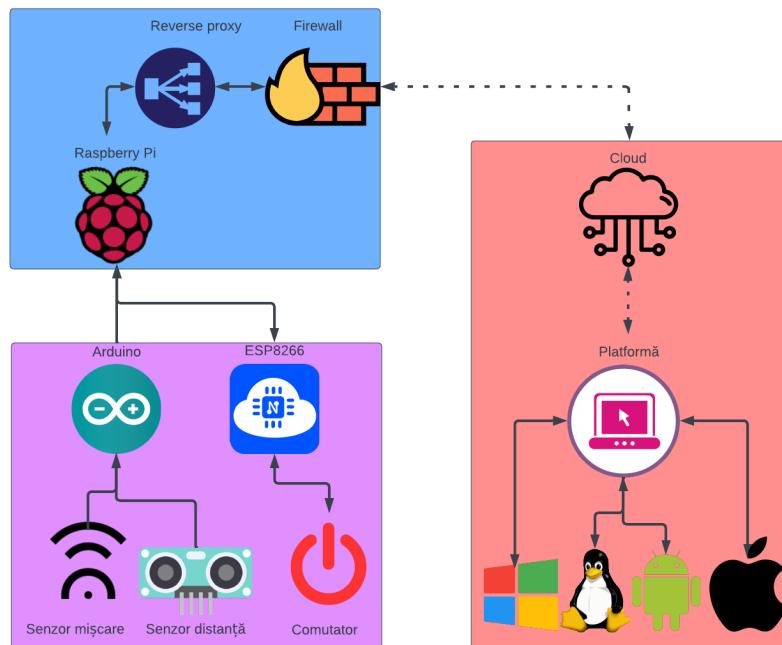


Figura 3.1: Legătura dintre componentele soluției

In imaginea prezentata anterior putem observa un exemplu al arhitecturii aplicatiei si cum sunt legate componentele intre ele. Aceasta aplicatie poate fi utilizata pe platformele: Windows/Mac/Linux/iOS/Android. Ele sunt conectate la senzorii casei prin intermediul serverului care ruleaza pe un Raspberry Pi cu ajutorul API-urilor.

### 3.1 Pagina principală

Primul punct de contact dintre un utilizator și acest proiect este aplicația de telefon (Figura 3.2). Odată lansată, va prezenta un splash screen, urmat de pagina principală. Aici se regăsesc toți senzorii conectați la sistem, împreună cu id-ul, valoarea cu timpul la care au fost înregistrate și locația din casă a senzorului.

Aceștia sunt sortați după parametrul pe care îl colectează, fie detectarea mișcării, calcularea distanței, a temperaturii sau a umidității, lucrul care se observă din titlurile situate deasupra fiecărui cartonaș. Aceste device-uri se numesc *senzori statici*, care sunt folosiți doar pentru colectare de informații. Mai există alt tip de senzor numit *dinamic* a cărui caracteristici seamănă cu cele statici, singura diferență fiind posibilitatea de comunicare bidirectională. În cazul acestui proiect, singurul senzor de acest tip va închide și deschide o lampă.

În partea de sus se află două butoane, unul de activare/ dezactivare a *modului de securitate* alături de cel de refresh care preia cele mai recente valori ale senzorilor, fiecare având o secțiune dedicată funcționalității și a modului de utilizare.

Pe post de footer sunt alte două butoane care servesc pe post de notificări atunci când valoarea unui senzor scade sub o anumită limită, respectiv cel de informații personale ale creatorului licenței.

#### 3.1.1 Cărțile senzorilor

Cărțile reprezintă principala sursă de informații pe care un utilizator o regăsește instant când accesează aplicația. Ele oferă date importante despre senzorul respectiv într-o formă ușor de înțeles și de citit rapid.

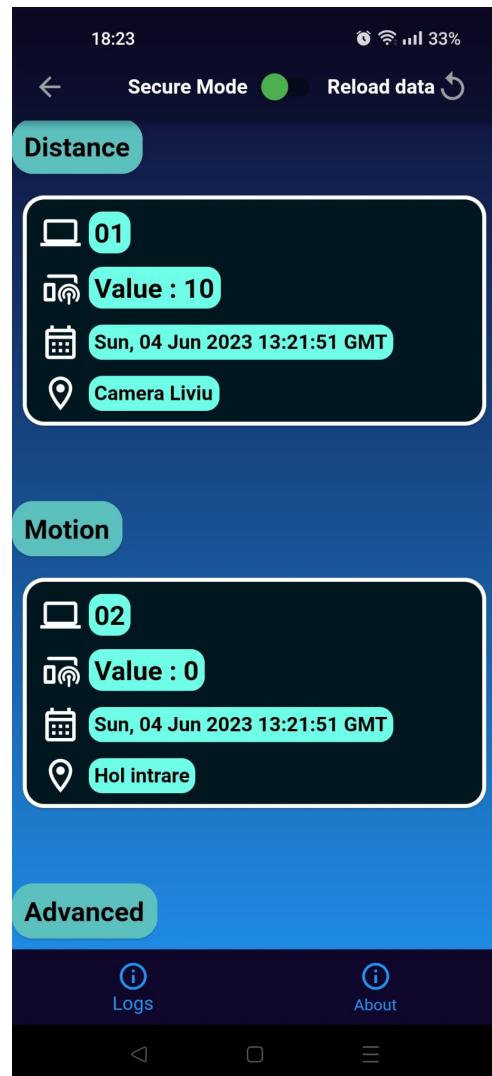


Figura 3.2: Pagina principală

În prima parte putem observa numele de identificare a senzorului. Acesta îi este asignat manual atunci când placa Arduino este programată și trebuie neapărat să fie unic în toată rețeaua.

În cazul în care există multiple dispozitive cu același ID, stația de bază va crede că a primit două pachete de la același senzor, când defapt au fost trimise de către senzori care, foarte probabil, sunt programati să colecteze date diferite și astfel vor apărea discrepanțe vizibile între valorile trimise.

După, putem observa valoarea primită de către aplicație la data și ora respectivă. Odată ce este reîmprospătată pagina, dacă device-ul este conectat la rețea, telefonul va primi cea mai recentă măsurătoare din baza de date care primește informații noi odată la două secunde.

Și nu în ultimul rând, apare și locul fizic exact al senzorului, el fiind setat de către utilizator în aplicație. Inițial, acest câmp nu este vizibil deoarece senzorii nu au atașat un loc stabil, atașarea și schimbarea lui fiind făcute de către utilizator.

Când vine vorba despre device-urile dinamice, au o interfață asemănătoare, singura diferență fiind câmpul de adresă care deține IP-ul local al senzorului.

Atunci când acest widget este apăsat, utilizatorul va intra pe pagina senzorului respectiv, detaliile căreia vor fi prezentate în secțiunea 3.2.

### 3.1.2 Locația senzorului

Cu un suport larg de diferite gesturi cu ajutorul clasei GestureDetector din Flutter, avem acces la o suiată de detectări a mișcărilor degetelor care pot executa anumite funcții. În cazul aplicației, odată ce se ține apăsat pe un cartonaș, va apărea un dialog în care putem introduce locația fizică a senzorului.

Aceasta este salvată în memoria locală a telefonului cu ajutorul bibliotecii de gestionare a bazelor de date NoSQL (non-relationale). Am ales biblioteca res-



Figura 3.3: Valorile unui modul

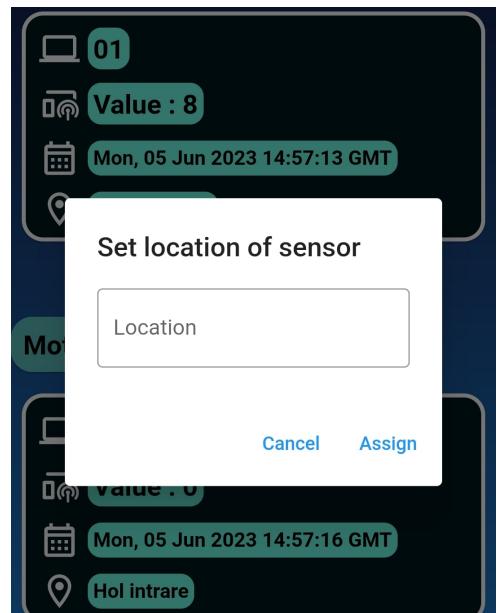


Figura 3.4: Pop-up pentru locație

pectivă datorită performanței ridicate, simplitatea oferită și compatibilitatea cross-platform care este critică deoarece aplicația aceasta poate fi compilită pentru majoritatea device-urilor populare.

### 3.1.3 Modul de securitate

În ultimii ani, numărul de furturi din locuințe se află pe un trend descrescător, conform informațiilor oferite de către Inspectoratul General al Poliției Române (IGPR). Așadar, în anul 2019 au fost înregistrate 22341 de furturi, iar în 2020 infracțiunile au ajuns la 18060, cu 19% mai puține față de anul precedent. Chiar și în aceste circumstanțe, acest număr se datorează mentalității defensive în continuă dezvoltare și a sistemelor de alarmă care devin tot mai sofisticate. Am decis că introducerea unei astfel de abilități va contribui pozitiv la securizarea locuinței în care sistemul este instalat.

Modul de securitate se încadrează la categoria de bariere *psihice* deoarece *"descurajează infractorii sau detectează în fază incipientă o tentativă de pătrundere prin efracție (sisteme de securitate perimetrală, antiefracție, control al accesului sau sisteme de supraveghere și înregistrare video)"*<sup>1</sup>

Este recomandat ca, atunci când utilizatorul pleacă din incinta în care este instalată soluția smart home, să activeze modul de securitate. Fiecare senzor îi poate fi atribuit o limită astfel încât orice valoare preluată care scade sub acest prag (numit și *Threshold*), va apărea sub forma unei notificări pe pagina de *Logs*.

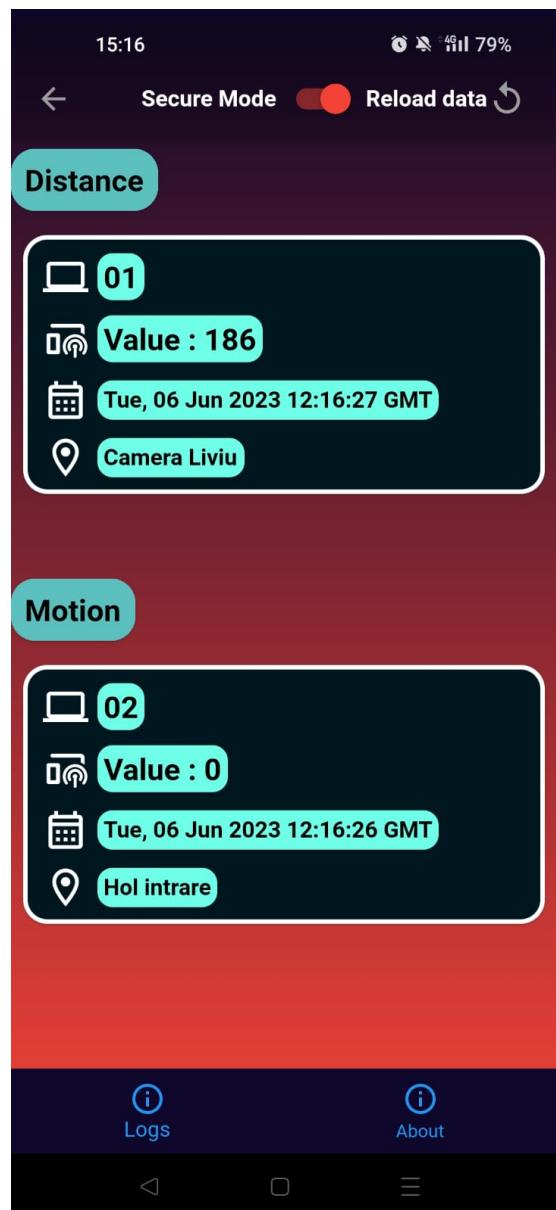


Figura 3.5: Interfață secure mode

<sup>1</sup><https://www.politiaromana.ro/ro/prevenire/furturi-din-lochuinte>.

După ce este activat, utilizatorului îi este făcut prezent acest fapt prin schimbarea culorii de background. Valoarea acestui buton este salvată în baza de date Hive, împreună cu data și ora la care a fost schimbată. Aceasta din urmă va fi folosită la filtrarea alarmelor false.

Senzorii dinamici pot fi configurați astfel încât să se folosească de această valoare și din acest motiv, la fiecare schimbare a acestui buton, valoarea va fi postată pe ruta <https://andr3w.ddns.net/secure>, urmând să fie disponibilă tuturor device-urilor pentru a o putea citi.

### 3.1.4 Butonul de refresh

Funcționalitatea sa este intuitivă, întrucât apăsarea lui rezultă în re-împrospătarea valorilor din aplicație cu cele mai recente citiri ale tuturor senzorilor sistemului. Deoarece tot backend-ul este hostat pe un procesor cu un singur nucleu, am decis să implementez o metodă de *caching* a datelor pentru sesiunea curentă. În acest mod salvăm serverul de la multe request-uri și economisim datele mobile.

Una din limitările sistemului este că informațiile afișate nu sunt actualizate constant. Acest lucru duce la discrepanțe între valorile pe care utilizatorul le vede și cele pe care senzorii le trimit. Vom vorbi despre aceasta în capitolul dedicat viziunii pe viitor.

## 3.2 Pagina senzorului static

Pe această rută se află informații și setări care contribuie la sensul de Smart home: componente ce lucrează împreună pentru a aduce utilizatorului un aport esențial de securitate.

AppBar-ul este populat cu un buton de setare a unei limite, valoare despre care am vorbit în secțiunea 3.1.3. Odată ce *threshold* este setat (Figura 3.6), va fi salvată în baza de date locală, astfel ea persistând chiar și când restartăm aplicația. Ca să fie vizibilă utilizatorului, pe graficul de mai jos se va trasa o linie roșie (care nu există dacă nu avem o limită). Ca utilizare, limita va fi folosită la notificările de pe pagina de *Logs* pentru a determina dacă o înregistrare trebuie arătată utilizatorului în caz că valoarea ei scade sub limita asignată.

În prim plan se află graficul care afișează evoluția ultimelor N înregistrări ale senzorului respectiv în funcție de data la care au fost colectate individual. Fiindcă sunt foarte multe puncte afișate, nu am putut include vizual și valoările lor, dar, utilizatorul o poate vedea, împreună cu data aferentă dacă apasă pe oricare punct (Figura 3.7a). Pentru construcția sa, am decis să folosesc librăria **fl\_chart** datorită popularității și a documentației excelente. Este un grafic fluid deoarece odată ce se modifică valorile afișate, motorul **fl\_chart** creează o animație de tranziție de la informațiile vechi la cele noi.

Dedesubtul figurii apare un slider care setează numărul de puncte extrase de la server. A existat o problemă aici care îmi făcea un API call la fiecare schimbare de valoare, adică dacă de la 39 de puncte doresc să îmi afișez 200, va face  $200 - 39 = 161$  request-uri către backend. Remediul a fost ca reîmprosătarea să fie făcută în funcția **onChangeEnd** care va trimite cererea către server atunci când s-a terminat de ales cantitatea de informații, evitând astfel un *Denial of Service* pentru server.

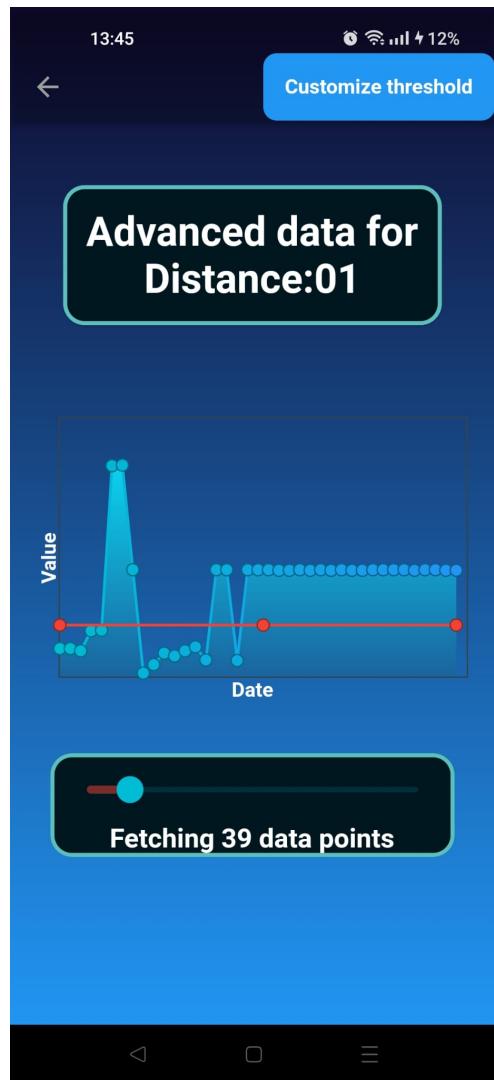
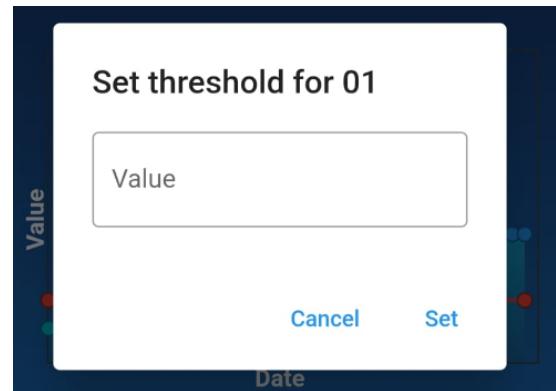


Figura 3.6: Date avansate despre senzorul 01



(a) Grafic



(b) Dialog limită

Figura 3.7

### 3.3 Pagina senzorului dinamic

Modulele dinamice sunt destul de diferite față de cele statice, întrucât se conectează direct la rețeaua locală de Wi-Fi (fiindu-le asignat o adresă IP) și pot primi sau trimite comenzi la server sau la alte module dinamice. M-am decis să adaug acest tip de senzori datoreită recomandării îndrumătorului licenței, domnul profesor Vidrașcu, o adiție ce se pliază perfect pe subiectul abordat în această teză: o soluție smart home.

După cum se poate observa, avem afișate datele despre senzor într-o manieră similară cu cea de la cei statici, diferență fiind IP-ul. Senzorii dinamici pot fi controlați de către utilizator și, în acest caz, este prin intermediul butoanelor ce apeleză *endpoint-urile* hostate pe ESP8266. Un endpoint reprezintă o locație specifică către care se pot face cereri HTTP (GET, POST, PUT, DELETE) pentru a accesa și manipula resursele unui serviciu. Serverul de pe ESP8266 își publică endpoint-urile la stația de bază care arată astfel: <http://192.168.0.101/turnOffLamp>.

Fiecare rută pe care o apelăm va instrui senzorul să execute diferite funcționalități care pot fi deduse din numele afișat pe buton.

**IntruderLampAlert** desemnează cum reacționează

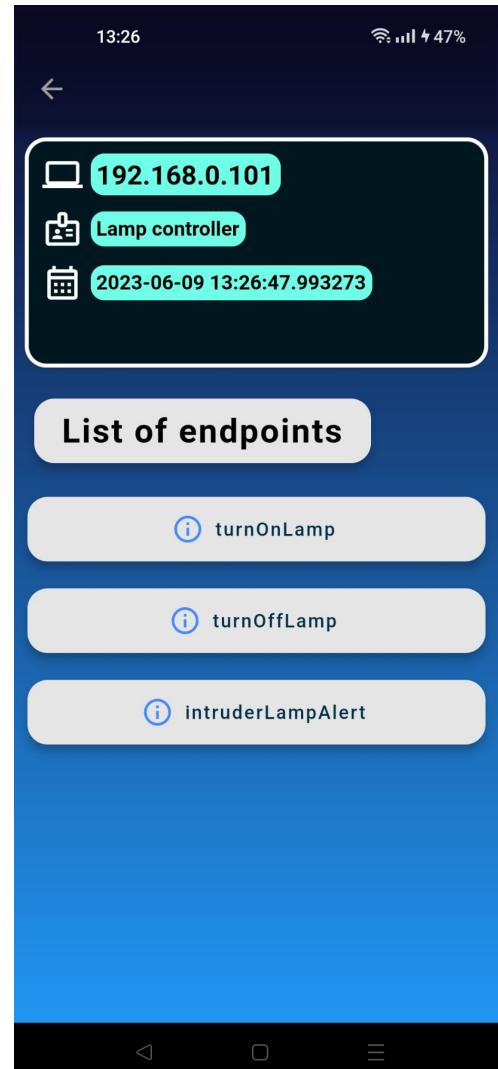


Figura 3.8: Informații și rutete senzorului dinamic

acest modul atunci când modul de securitate este activat și senzorul de mișcare emite un "1" logic.

### 3.4 Pagina de informații

Fiecare anomalie din sistem (orice senzor ce înregistrează o valoare care a scăzut sub limita impusă) va crea o notificare pe care utilizatorul o poate verifica în secțiunea de **Logs**. Aceasta poate fi accesată din bara de footer al meniului principal.

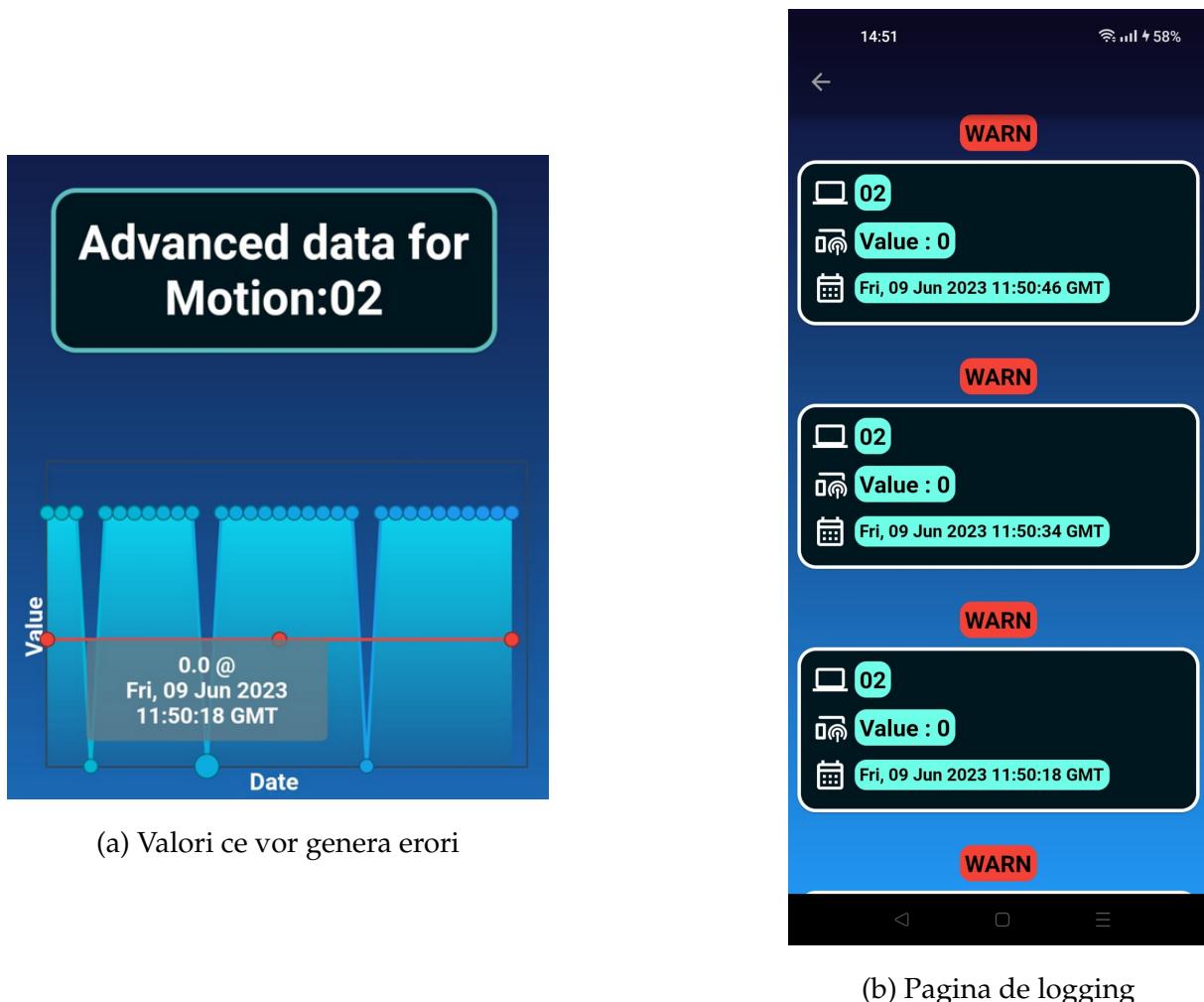


Figura 3.9

Initial, aplicația nu generează notificări dacă un senzor a înregistrat o abatere de la normal atunci când **Secure Mode** nu este activat deoarece se presupune că utilizatorul nu a plecat încă de acasă. Activarea modului de securitate va cauza afișarea anomaliei pe ecran.

Din figura 3.9a se pot extrage anumite informații astfel: există trei puncte care se află sub bariera setată cu ajutorul butonului **Set threshold**, ceea ce înseamnă că

utilizatorul și-a părăsit apartamentul, dar încă există activitate.

Aplicația va prelua ultimele înregistrări a fiecărui senzor pe care le compară cu limita (stocată în baza de date locală) și afișează cartonașele corespunzător. Se poate observa în figura 3.9b. că ultima alertă este cea din figura anterioară.

### 3.5 Pagina de genereare de rapoarte

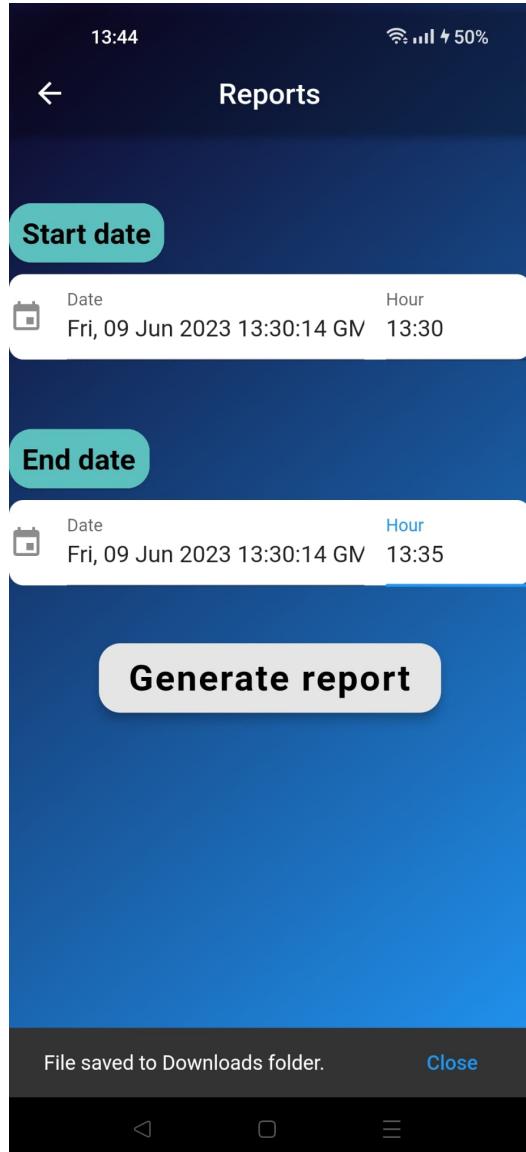
În cazul în care utilizatorul este o victimă a unui jaf sau unei intruziuni în casă, este important să acționeze în mod corespunzător. Primul pas este să își asigure siguranța personală și să sună imediat la serviciile de urgență sau la poliție, încercând să nu intre în contact cu posibili infractori sau să manipuleze evidențele rămase la fața locului, pentru a nu distrugă potențiale probe. După anunțarea autorităților, trebuie făcută o inventariere a obiectelor pierdute și să fie extrase cât mai multe informații din ziua respectivă legate de incident.

Aplicația din această soluție ajută utilizatorul în cazul unui astfel de eveniment prin colectarea tuturor datelor senzorilor din casă, expunându-le în formă de PDF gata să fie oferite autorităților. Trebuie menționat faptul că senzorii au fost plasati specific în holul de la intrarea în apartamentul meu, poziționați astfel încât orice persoană care intră va modifica starea modulelor.

Interfața (Figura 3.10a) este alcătuită din cele două câmpuri ce permit introducerea timpului de început și cel de sfârșit. Acestea vor fi folosite de către server pentru a filtra toate înregistrările senzorilor, luându-le doar pe cele ce se încadrează în intervalul specificat. Cele două valori trebuie neapărat setate, însă dacă nu sunt ambele. Odată apăsat, va trimite un POST request la stația de bază care îi va răspunde cu datele aferente. Utilizatorul este notificat de terminarea procesului printr-un pop-up în josul ecranului ce oferă și folderul în care a fost salvat.

Cu ajutorul librăriei **pdf** din cadrul SDK-ului Flutter, am reușit generarea documentului (Figura 3.10b). Pe post de header, am specificat intervalul din care au fost extrase informațiile senzorilor și data la care a fost creat acest document. Footer-ul conține numele aplicației și deținătorul drepturilor licenței. În centrul atenției este un tabel ce conține, în ordine, numărul curent al înregistrării, adresa, data, ID-ul, tipul și valoarea senzorului. Aceasta este extins pe multiple foi, cifra din partea de jos al ecranului desemnând numărul curent al paginii. Pentru raportul acesta avem un număr de 300 de înregistrări preluate în decursul a 5 minute. Stația de bază este responsabilă să recepționeze și să citească pachetele trimise de toți senzorii, lucru care devine destul de greu atunci când fiecare device transmite în aceeași secundă după cum se observă în PDF. Datorită librăriei **RF24Network** despre care vom vorbi în secțiunea senzorilor statici, toate pachetele vor fi decodeate cu succes.

Datele acestui document sunt valoroase și pot ajuta foarte mult investigatorii, crescând astfel șansele de a prinde intrușii. Asigurând proprietarii ce folosesc acest sistem cu un raport plin de informații, soluția de smart home propusă joacă un rol important în creșterea securității utilizatorului.



(a) Interfața paginii

The screenshot shows the generated PDF report. At the top, it has a header with icons for back, forward, search, and share. Below this is a table titled 'Data recorded between Fri, 09 Jun 2023 10:30:00 GMT and Fri, 09 Jun 2023 10:35:00 GMT. Report has been generated on: Fri, 09 Jun 2023'. The table has columns for Count, Address, Date, ID, Type, and Value. The data consists of 168 rows of motion and distance events. At the bottom of the table, it says 'Licensed to: Fil\_Report generated by: Smart Home App 2023'. Below the table is a toolbar with icons for Comment, Highlight, Draw, Text, Fill & Sign, and More tools. At the very bottom, there are navigation icons.

Count	Address	Date	ID	Type	Value
141	02	Fri, 09 Jun 2023 10:32:39 GMT	63188	Motion	0
142	01	Fri, 09 Jun 2023 10:32:39 GMT	63187	Distance	8
143	02	Fri, 09 Jun 2023 10:32:37 GMT	63186	Motion	0
144	01	Fri, 09 Jun 2023 10:32:37 GMT	63185	Distance	8
145	02	Fri, 09 Jun 2023 10:32:35 GMT	63184	Motion	0
146	01	Fri, 09 Jun 2023 10:32:35 GMT	63183	Distance	8
147	02	Fri, 09 Jun 2023 10:32:33 GMT	63182	Motion	0
148	01	Fri, 09 Jun 2023 10:32:33 GMT	63181	Distance	8
149	02	Fri, 09 Jun 2023 10:32:31 GMT	63180	Motion	0
150	01	Fri, 09 Jun 2023 10:32:31 GMT	63179	Distance	8
151	02	Fri, 09 Jun 2023 10:32:29 GMT	63178	Motion	0
152	01	Fri, 09 Jun 2023 10:32:29 GMT	63177	Distance	8
153	02	Fri, 09 Jun 2023 10:32:27 GMT	63176	Motion	0
154	01	Fri, 09 Jun 2023 10:32:27 GMT	63175	Distance	8
155	02	Fri, 09 Jun 2023 10:32:25 GMT	63174	Motion	0
156	01	Fri, 09 Jun 2023 10:32:25 GMT	63173	Distance	8
157	02	Fri, 09 Jun 2023 10:32:23 GMT	63172	Motion	0
158	01	Fri, 09 Jun 2023 10:32:23 GMT	63171	Distance	8
159	02	Fri, 09 Jun 2023 10:32:21 GMT	63170	Motion	0
160	01	Fri, 09 Jun 2023 10:32:21 GMT	63169	Distance	8
161	02	Fri, 09 Jun 2023 10:32:19 GMT	63168	Motion	0
162	01	Fri, 09 Jun 2023 10:32:19 GMT	63167	Distance	8
163	02	Fri, 09 Jun 2023 10:32:17 GMT	63166	Motion	0
164	01	Fri, 09 Jun 2023 10:32:17 GMT	63165	Distance	8
165	02	Fri, 09 Jun 2023 10:32:15 GMT	63164	Motion	0
166	01	Fri, 09 Jun 2023 10:32:15 GMT	63163	Distance	8
167	02	Fri, 09 Jun 2023 10:32:13 GMT	63162	Motion	0
168	01	Fri, 09 Jun 2023 10:32:13 GMT	63161	Distance	8

(b) PDF-ul generat

Figura 3.10

## 3.6 Modul debug

**Network\_rx.py** reprezintă modul de debug, fiind o unealtă care ajută utilizatorul la momentul instalării senzorilor sau atunci când unii nu mai funcționează corespunzător.

În timpul execuției, acesta afișează în consolă (Figura 3.11) toate pachetele pe care le primește în timp real. Acest lucru este util deoarece permite utilizatorului să confirme că senzorii funcționează corect și trimit datele corespunzătoare. Problemele de comunicare sunt depistate din timp, de obicei acestea se rezolvă cu o restartare a modulelor.

Fiecare linie deține datele unei înregistrări: **payload len** reprezintă lungimea în octeți a pachetului, **sensor type** e tipul senzorului (distanță, temperatură, mișcare), **value** conține valoarea citită de device, iar **header** are date precum id-ul pachetului, emițătorul și receptorul, care în cazul acesta este stația de bază la adresa **00**.

```
Tilix: andrew@raspberrypi: ~/Licenta/flask/debug_scripts
payload len: 20, sensor type: Motion,6, value: 1, header: id 121 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 140 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 122 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 141 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 123 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 142 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 124 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 143 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 125 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 144 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 126 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 145 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 127 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 146 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 128 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 147 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 129 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 148 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 130 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 149 from 01 to 00 type 0
payload len: 20, sensor type: Motion,6, value: 1, header: id 131 from 02 to 00 type 0
payload len: 20, sensor type: Distance,8, value: 8, header: id 150 from 01 to 00 type 0
```

Figura 3.11: Detaliile senzorilor conectați la stația de bază

Este important ca user-ul să aibă instalate toate librăriile de care acest script depinde, lucru despre care vom intra în detaliu la secțiunea backendului. Fiind un script scris în python aflat în folderul **debug\_scripts**, este executat prin introducerea comenzi **python3 Licenta/flask/debug\_scripts/network\_rx.py**. Fiindcă rulează la infinit, se poate ieși din el apăsând **Ctrl + C** sau urmând instrucțiunile de aici.<sup>2</sup>

<sup>2</sup><https://github.com/hakluke/how-to-exit-vim#the-hardware-way>

## 3.7 Frontend

Înainte de a intra în detaliu la fiecare element, este important de a înțelege că Flutter detine o gamă largă de elemente și componente care formează bazele unei interfețe grafice, noțiunile pe care le-am folosit în acest proiect fiind de dificultate medie.

Printre aceste elemente, regăsim:

- Scaffold: este unul din componentele fundamentale din acest *SDK*, oferindu-ne o suprafață pe care o putem popula cu **AppBar** sau **BottomNavigationBar**;
- Column și Row: sunt widget-uri pentru îmbunătățirea aspectului. Elementele ce vor fi poziționate în ele vor fi aranjate vertical, respectiv orizontal;
- FloatingActionButton: reprezintă un buton care, odată apăsat, va notifica sistemul să execute funcția sa asociată;
- AppBar: este elementul vizual care acționează pe post de Header al aplicației, se îmbină de obicei cu Scaffold;
- FutureBuilder: această componentă va construi în mod concurrent elemente de interfață care sunt extrase de la API-uri.

## 3.8 REST API

Un **API** (**Application programming interface**) este un mecanism care permite unui serviciu să acceseze resursa unui alt serviciu. REST API se bazează pe principii precum utilizarea verbelor HTTP (GET, PUT, POST, DELETE) pentru transmiterea și manipularea datelor într-un format standardizat (XML, JSON), împreună cu ajutorul **URI** (uniform resource identifier) pentru specificarea resursei asupra cărei operațiile vor fi executate. Am creat astfel o arhitectură REST pentru a servi diferite informații către aplicația de mobil, afișându-le într-o manieră ușor de înțeles.<sup>3</sup>

Serverul expune rute de accesare a informațiilor din baza de date pentru ambele tipuri de senzori: cei statici se află la adresa <https://andr3w.ddns.net/API>, iar cei dinamici la <https://andr3w.ddns.net/advanced/API>

---

<sup>3</sup><https://www.ibm.com/topics/rest-apis>.

### 3.8.1 DNS dinamic

Provider-ul de internet asignează dinamic adresele IP utilizatorilor conectați, motiv pentru care după o perioadă de curent, de obicei se resetează această adresă. Pentru a rezolva această problemă, am utilizat serviciul gratuit de **Dynamic DNS**<sup>4</sup> al companiei **No-IP**<sup>5</sup>. Setările au fost realizate direct pe router-ul personal, un **TP-Link WR740N**. Odată configurație, rețeaua locală poate fi accesată de la adresa <https://andr3w.ddns.net>.

## 3.9 Backend

Backend-ul reprezintă partea aplicației responsabilă de logică și funcționalitate. În contextul acestei licențe, am decis să folosesc limbajul **Python** versiunea 3.10 cu microframework-ul **Flask** către care sunt trimise pachetele de la **gunicorn** și **nginx**.



Figura 3.12: Placa Raspberry ce găzduiește serverul

Acesta rulează pe stația de bază (Figura 3.12) pentru a satisface fiecare cerere a utilizatorului, fără a fi nevoie să își deschidă calculatorul personal pentru a putea interacționa cu dispozitivele sale.

Pentru o productivitate crescută, am ales să instalez sistemul de operare *Raspberry Pi OS Lite* deoarece este rapid, lucru datorat lipsei unei interfețe grafice și a

<sup>4</sup><https://www.noip.com/support/knowledgebase/geek-terms>

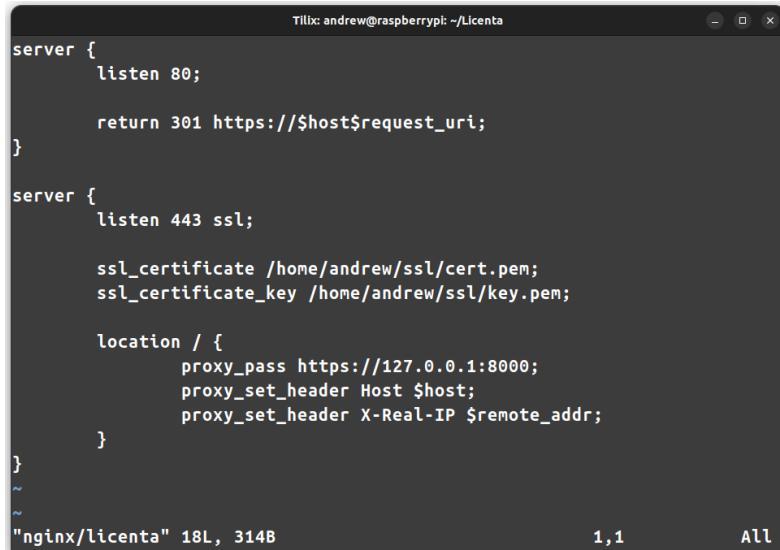
<sup>5</sup><https://www.noip.com/>

programelor pre-instalate. Instalarea a fost foarte ușoară, urmând instrucțiunile din documentația lor<sup>6</sup>.

Comunicarea cu placa a fost făcută doar prin **SSH** care a fost deschis din setările router-ului. Fiindcă această conexiune va fi publică, am instalat programul **ufw** (*uncomplicated firewall*) ce mi-a ușurat securizarea host-ului, fiind o unealtă ce înlocuiește setările complicate ale **iptables**<sup>7</sup> cu comenzi simple: *sudo ufw allow 22*. Astfel am limitat numărul de conexiuni la portul 22 și am permis accesul la portul 443, acestea fiind singurele căi de acces.

Nginx este un **reverse proxy**<sup>8</sup> care se află între client și backend. Acesta primește cererile din exterior pe care le redirecționează serverului de gunicorn după setările din figura 3.13.

Criptarea traficului e realizată direct de către nginx, specificând locația certificatului și a cheii private cu ajutorul comenziilor *ssl\_certificate* și *ssl\_certificate\_key*. Am decis să îmi semnez propriul **certificat SSL** (Secure sockets layer) cu utilitarul **openssl**.



```
Tilix: andrew@raspberrypi: ~/Licenta
server {
    listen 80;

    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;

    ssl_certificate /home/andrew/ssl/cert.pem;
    ssl_certificate_key /home/andrew/ssl/key.pem;

    location / {
        proxy_pass https://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
~
~

"nginx/licenta" 18L, 314B
```

Figura 3.13: Configurare Nginx

Prima componentă numită *server* interceptează fiecare cerere HTTP pe care o redirecționează la ruta de HTTPS asignată portului 443, returnând astfel codul 301 ce notifică utilizatorul că a fost redirectat de la ruta **http://** la cea de **https://**.

A doua componentă asaltă la portul 443 (HTTPS) și, datorită blocului *location /*, orice cerere va fi redirecționată către serverul Flask, fiind configurat a primi request-

<sup>6</sup><https://www.raspberrypi.com/documentation/>

<sup>7</sup><https://wiki.archlinux.org/title/iptables>.

<sup>8</sup><https://www.nginx.com/resources/glossary/reverse-proxy-server/>.

uri la portul 8000. Trebuie menționat faptul că ambele aplicații rulează pe același host, ceea ce face posibilă trimiterea cu succes către adresa de *loopback*<sup>9</sup>. Această utilizare este ideală deoarece ne aflăm la o scară de nivel de proiect personal. Pentru uz în producție se recomandă folosirea unui proxy server<sup>10</sup> pe un computer separat.

### 3.9.1 Stocarea datelor

**SQLAlchemy** este o colecție de unelte ce ajută programatorul să acceseze și manipuleze baze de date SQL în limbajul Python. Se pot scrie query-uri în formă de *string-uri* (raw sql) sau înlănțuind obiecte din python într-o manieră **ORM** (Object relational mapping)<sup>11</sup>. Motivele pentru care am folosit acest toolkit sunt simplicitatea creării interogărilor, rapiditatea codării și multitudinea tuturialelor valabile.

	id	address	type	value	date
	Filter	Filter	Filter	Filter	Filter
99450	99450	02	Motion	1	2023-06-11 12:59:24
99451	99451	01	Distance	187	2023-06-11 12:59:25
99452	99452	02	Motion	1	2023-06-11 12:59:26
99453	99453	01	Distance	186	2023-06-11 12:59:27
99454	99454	02	Motion	1	2023-06-11 12:59:28

Figura 3.14: Informațiile oferite de DB Browser for SQLite

Există două tabele în `/flask/application/models.py`: **Sensor** și **AdvancedSensor**, ambele stocând date (figura 3.14) precum id-ul, numele, valoarea și data la care a fost adăugată înregistrarea senzorului în baza de date.

<sup>9</sup><https://study-ccna.com/loopback-interface-loopback-address/>

<sup>10</sup><https://browserjet.com/blog/advantages-and-disadvantages-of-a-proxy-server>

<sup>11</sup><https://docs.sqlalchemy.org/en/20/orm/queryguide/index.html>.

Senzorii emit date odată la două secunde care trebuie stocate. Pentru acest lucru, am creat un fir de execuție nou care inserează fiecare citire pe care serverul o primește folosindu-se de codul din figura de mai jos.

```

1 EXPECTED_SIZE = struct.calcsize("<16sL")
2 def update_database(app, db):
3     try:
4         while True:
5             network.update()
6             while network.available():
7                 header, payload = network.read()
8                 sensor_type, value = struct.unpack("<16sL", payload[:EXPECTED_SIZE
9                     ])
10                sensor_type = sensor_type.decode('utf-8')
11                sensor_type_clean = ''.join(letter for letter in sensor_type if
12                    letter.isalnum())
13                address = header.to_string().split(' ')[3]
14
15                with app.app_context():
16                    sensor_db_entry = Sensor(address=address, type=sensor_type_clean,
17                        value=value)
18                    db.session.add(sensor_db_entry)
19                    db.session.commit()
20
21    except KeyboardInterrupt:
22        print("powering down radio and exiting.")
23        radio.power = False

```

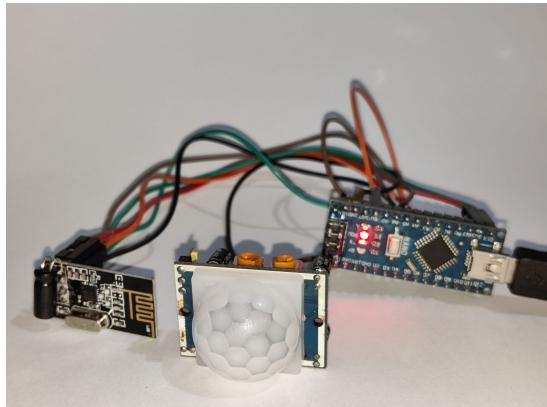
În primă fază este calculată lungimea pachetelor transmise de senzori, aceștia conținând un cuvânt de 16 caractere și un număr întreg. Cât timp serverul este deschis, se vor citi aceste pachete, vor fi sanitizate și de-serializeate, operații ce vor rezulta în obținerea valorilor trimise de la senzor. La final va fi creat un obiect de tipul *Sensor* care va fi introdus în baza de date locală.

### 3.10 Senzori statici

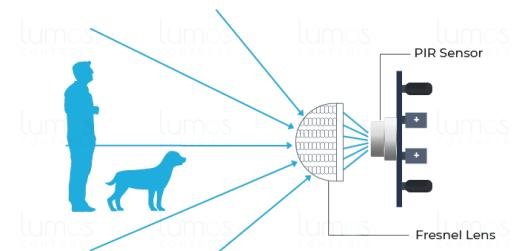
Acest tip de device-uri stă la baza metodei de colectare a informațiilor despre casa respectivă și cea de generare a unui raport cu toate valorile din baza de date în intervalul cerut.

Figura 3.15a ne prezintă cum a fost creat un senzor static: un Arduino Nano care

este cablat la senzorul de detectare a mișcării și modulul de transmitere a datelor.



(a) Arduino Nano cablat la senzorul de mișcare și modulul transmisie



(b) Componența piroelectrică

Figura 3.15: Arduino și demonstrație

Senzorul are 3 pini care trebuie conectați la placa de dezvoltare: împământarea, 5V și ieșirea. Ultimul pin emite un 0 logic dacă nu a detectat niciun obiect și 1 logic dacă a detectat o persoană sau un animal. Acest lucru este posibil datorită faptului că este emisă căldură în formă de radiații infra-roșii care alimentează componenta *piroelectrică* (Figura 3.15b) amplasată dedesubtul unei lentile Fresnel ce ajută focusarea acestor raze pe lentila senzorului. Modulul are încorporat două potențiometre: unul pentru reglarea razei de acțiune (până în 5 metri), iar celălalt pentru reglarea timpului de schimbare a valorii detectate (între 0.3 - 300 secunde).

Odată alimentat la 5V, microcontroller-ul începe colectarea de date la un interval setat în faza configurării de către utilizator. În cazul meu, acesta este de două secunde. După ce acest interval a trecut, va fi creat un pachet ce conține ID-ul și tipul modulului, valoarea și data la care a fost făcută măsurătoarea.

### 3.11 Senzori dinamici

Senzorii dinamici dețin o complexitate mai ridicată față de tipul anterior prezentat datorită faptului că pot trimite și primi date simultan de la server.

Imaginea 3.16 demonstrează o utilitate simplă a modulului albastru care comută orice consumator ce poate fi alimentat de la priză. Fiecare componentă albastră acționează asemănător unui întrerupător care este controlat apăsând unul din butoanele prezente

în interfața aplicației de telefon (Figura 3.8). Ruta apelată este predefinită de către utilizator în faza de configurare a senzorului.

Pentru ca persoana ce deține soluția Smart Home să aibă cele mai recente funcționalități, senzorii își vor trimite toate informațiile serverului, acesta ulterior modificându-le în baza de date.

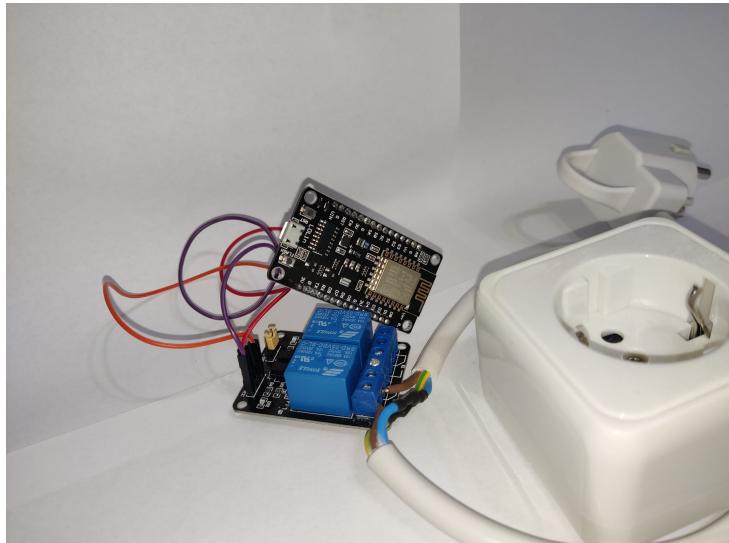


Figura 3.16: ESP8266 cu modul releu alimentat la 220V

## 3.12 Versionarea

Câteva din caracteristicile licenței au solicitat folosirea metodologiei de *branching*, schimbările codului fiind aplicate pe o copie a celui original. Am ales astfel să folosesc utilitarul **git**<sup>12</sup>, proiectul fiind postat pe **Github**<sup>13</sup> deoarece sunt familiar cu ambele tehnologii. Autentificarea bazată pe chei a fost folosită pentru a evita introducerea parolei contului la fiecare *push* sau **pull**.

---

<sup>12</sup><https://git-scm.com/>

<sup>13</sup><https://github.com/.>

# **Capitolul 4**

## **Scenarii de utilizare**

În acest capitol vor fi prezentate 4 scenarii în care un utilizator ar putea folosi aplicația, oferind un ghid clar despre modul utilizării fiecărui procedeu.

### **4.1 Colectare de date și logging**

Funcționalitatea de bază a acestei soluții este de a citi valorile curente ale fiecărui senzor pentru a fi stocate în baza locală de date.

Utilizatorul dorește să verifice statutul fiecărui device din locuința sa. Acesta deschide aplicația de telefon, trece de splash-screen și îi sunt afișate toate cartonașele senzorilor conectați la stația de bază. La prima vedere, se poate observa doar ultima înregistrare, dar dacă user-ul dorește să vadă graficul ultimelor citiri, trebuie să apese pe cartonașul specific curiozității sale și să selecteze mărimea eșantionului dorit, astfel fiindu-i prezentate datele necesare.

### **4.2 Comutarea luminilor din holul intrării**

Înainte de a ajunge acasă, utilizatorul intră în aplicație și dezactivează modul de securitate. După ce acesta intră pe hol, luminile din încăpere se vor aprinde automat datorită senzorilor care au detectat mișcarea, rămânând aprinse pentru toată perioada cât utilizatorul nu a părăsit încăperea.

În cazul în care utilizatorul a configurat și alte comutatoare (lumini, jaluzele smart, termostat), acțiunile dorite pot fi accesate din pagina dedicată senzorului respectiv și apăsând pe butonul ce reprezintă cerințele persoanei.

## 4.3 Detectarea problemelor în comunicare

Fiind un proiect de tipul *Do it yourself*, utilizatorul trebuie să își configureze fiecare senzor, de la programare până la alimentarea constantă. În timpul acestui proces, user-ul observă că unul din senzori nu apare pe interfața mobilă, aşa că:

- se loghează prin *ssh* la stația de bază pentru a executa scriptul de debug și
- deschide *Serial Monitor* din *Arduino IDE*.

Acesta citește erorile care sunt afișate la linia de comandă și remediază problema.

## 4.4 Generarea raportului

Utilizatorul este victimă unei intruziuni în casă din cauza faptului că nu a verificat aplicația de telefon pentru eventualele anomalii semnalate de senzori în modul de securitate.

Persoana în cauză intră în aplicație, la secțiunea **Report** și introduce intervalul aproximativ al timpului intruziunii, urmând ca pe telefonul său să fie descărcat un fișier ce conține toate datele senzorilor: valoare și timpul cîrșitii, fișier ce ulterior va fi oferit investigatorilor și poliției.

# Încheiere

## Direcții pe viitor

Consider că proiectul poate fi îmbunătățit în mai multe arii. O primă remediere ar fi **înlocuirea plăcii Raspberry Pi Zero W cu placă Raspberry Pi 4** deoarece procesorul cu un singur nucleu al primei plăci nu poate face față unui val de cereri API. Nucleele adiționale vor crește viteza de procesare a datelor care sunt citite de la senzori și vor scădea timpul de așteptare între cererile făcute de către telefonul consumatorului.

**Stabilitatea conexiunii** dintre senzorii statici și stația de bază poate fi întreruptă temporar din cauza altor dispozitive care transmit pachete de date în același moment, remediu fiind un mecanism de sincronizare între toți senzorii la momentul comunicării. În alte cuvinte, varianta actuală de comunicare poate urma un proces în care fiecare senzor trimite pe rând mesajele fără a întrerupe sau a fi întrerupt de interferențele altor trimiteri.

Un alt punct ce ar putea fi corectat ar fi **afișarea celor mai recente informații** din baza de date fără ca utilizatorul să fie nevoie să apese pe butonul de reîmprospătare. Acest lucru este posibil ca la fiecare citire recent introdusă în baza de date, serverul să notifice fiecare telefon de acest lucru și să trimită informațiile actualizate, metodă ce este restricționată momentan de puterea de calcul disponibilă.

Nu în ultimul rând, **stația de bază reprezintă un cost adițional** atunci când o persoană dorește să își instaleze sistemul smart. Pentru remedierea acestui lucru, hostarea în cloud ar fi cea mai bună soluție pentru a servi mai mulți utilizatori, scăpându-i astfel și de prețul relativ mare al acestor plăci.

## Concluziile lucrării

Lucrarea "Home Smartify - soluție pentru controlul dispozitivelor și securitatea casei" propune o abordare inovatoare pentru a simplifica modul de viață al utilizatorului, oferindu-i controlul complet asupra dispozitivelor din propria casă.

Prin intermediul aplicației mobile integrate, utilizatorul poate gestiona senzorii și dispozitivele de la distanță, indiferent de locația sa. Această soluție permite minimizarea legăturii fizice și facilitează adaptarea la un stil de viață modern și dinamic. În plus, abordarea orientată spre securitate asigură confidențialitatea și protecția datelor utilizatorului, oferind o experiență Smart Home sigură. Lucrarea reprezintă o contribuție valoroasă în domeniul tehnologiei Smart Home, oferind o soluție inovatoare și accesibilă pentru îmbunătățirea confortului și securității în mediul locuinței.

# Bibliografie

- Author1, *Book1*, 2018
- Ghid Licență
- Creating the ESP8266 server
- Raspberry Pi Foundation
- IFTTT
- Raspberry Pi Zero W
- Raspberry Pi Zero W (1)
- Raspberry Pi Zero W (2)
- Arduino Nano
- PIR Sensor
- Senzor de temperatură și umiditate
- Funcționalitate senzor PIR
- Funcționalitate senzor PIR (2)
- Prevenire furturi
- Setări firewall