

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

SECURITATE PE MAI MULTE NIVELE ÎN CLĂDIRI

PROIECT DE DIPLOMĂ

Autor: **Viviana POPA**

Conducător științific: **SL.dr.ing. Dan RADU**

2021



Vizat,

DECAN

Prof.dr.ing. Liviu MICLEA

DIRECTOR DEPARTAMENT AUTOMATICĂ

Prof.dr.ing. Honoriu VĂLEAN

Autor: **Viviana POPA**

Securitate pe mai multe nivele în clădiri

1. **Enunțul temei:** *Tema aleasă se referă la un sistem de monitorizare a securității într-o clădire. Acest sistem cuprinde atât partea software, cât și partea hardware. Partea hardware se referă la: senzori de monitorizare video (cctv) și sistemul de deblocare/blocare uși securizate prin diferite metode (keyboard password, rfid tag, fingerprint). Partea software cuprinde aplicația in sine (aplicație web client-server care se ocupă cu managementul sistemelor de securitate instalate în clădire și modele de recunoaștere a diferitelor scenarii de amenințare utilizând camerele video, în timp real).*
2. **Conținutul proiectului:** *Acest proiect cuprinde: Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Introducere, Studiu bibliografic, Analiză și fundamentare teoretică, Proiectarea și implementarea sistemului, Testare și rezultate, Concluzii și Bibliografie.*
3. **Locul documentării:** *Online*
4. **Consultanți:** *SL.dr.ing. Dan RADU*
5. **Data emiterii temei:** *01.10.2020*
6. **Data predării:** *08.07.2021*

Semnătura autorului

Semnătura conducătorului științific

MINISTERUL EDUCAȚIEI



UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**UNIVERSITATEA TEHNICĂ**

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

**Declarație pe proprie răspundere privind
autenticitatea proiectului de diplomă**

Subsemnatul(a) **Viviana POPA**,
legitimat(ă) cu CI seria XB nr. 531138, CNP 2981007125822,
autorul lucrării: Securitate pe mai multe nivele în clădiri

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la
Facultatea de Automatică și Calculatoare, specializarea **Automatică și Informatică
Aplicată (la Satu Mare)**, din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Iulie
2021 a anului universitar 2020-2021, declar pe proprie răspundere, că această lucrare
este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza
informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au
fost folosite cu respectarea legislației române și a convențiilor internaționale privind
drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte
comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile
administrative, respectiv, *anularea examenului de licență*.

Data

08.07.2021

Prenume NUME

Viviana POPA

(semnătura)



SINTEZA

proiectului de diplomă cu titlul:

Securitate pe mai multe nivele în clădiri

Autor: **Viviana POPA**

Conducător științific: **SL.dr.ing. Dan RADU**

1. Cerințele temei: Realizarea unei soluții de securitate pentru clădiri adaptabilă pentru diverse utilizări: case, clădiri de birouri, bănci, hoteluri, magazine, spitale, etc.
2. Soluții alese: Pachet ce conține atât parte software cât și hardware. Partea hardware este pentru proof of concept și cuprinde câte unul din fiecare circuit de senzori (sistem de blocare ușă prin keypad, sistem de blocare ușă prin rfid, sistem de blocare ușă prin fingerprint) restul senzorilor fiind simulați. Partea software reprezintă o aplicație web client-server în care se poate face monitorizarea automată a fiecărei încăperi din clădire (monitorizează senzorii de deblocare uși și camerele de supraveghere inteligente ce recunosc diferite scenarii prestabilite).
3. Rezultate obținute: S-a implementat aplicația atât software cât și hardware și s-a reușit monitorizarea automată a tuturor senzorilor. Utilizatorul poate vedea statusul senzorilor de la fiecare etaj și rezultatele modelelor de recunoaștere video.
4. Testări și verificări: S-au testat modelele de recunoaștere video și fiecare funcționalitate a aplicației în parte. Erorile găsite au fost remediate și după s-a testat din nou.
5. Contribuții personale: Chiar dacă s-au folosit tehnologii deja existente în unele părți (de exemplu tensorflow pentru modelele de recunoaștere video) ideea în sine și toate funcționalitățile aduse într-o singură soluție reprezintă contribuția personală încercând să se aducă un produs inovativ într-o ramură nu foarte dezvoltată a sistemelor de securitate în momentul de față.




UNIVERSITATEA TEHNICĂ

DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

6. Surse de documentare: online

Semnătura autorului  _____

Semnătura conducătorului științific  _____

Cuprins

1	INTRODUCERE.....	3
1.1	CONTEXT GENERAL	3
1.2	OBIECTIVE.....	4
1.3	SPECIFICAȚII	5
2	STUDIU BIBLIOGRAFIC.....	7
2.1	EVOLUTIA SISTEMELOR DE MANAGEMENT AL CLADIRILOR.....	7
2.2	POTENTIALE PUNCTE DE ACCES INTR-UN SISTEM BAS.....	9
2.2.1	<i>Camerele de supraveghere video.....</i>	<i>9</i>
2.2.2	<i>Sisteme HVAC.....</i>	<i>10</i>
2.2.3	<i>Control acces.....</i>	<i>10</i>
2.2.4	<i>Vulnerabilități ce pot fi descoperite într-un atac.....</i>	<i>12</i>
2.2.5	<i>Cum se pot reduce riscurile din rețelele de tip BAS.....</i>	<i>14</i>
3	ANALIZĂ SI FUNDAMENTARE TEORETICA.....	15
3.1	CONCEPTE GENERALE	15
3.2	TEHNOLOGII UTILIZATE	16
3.2.1	<i>MERN Stack</i>	<i>16</i>
3.2.1.1	<i>Ce este MERN Stack</i>	<i>16</i>
3.2.1.2	<i>Cum functioneaza MERN Stack</i>	<i>16</i>
3.2.1.3	<i>React.js Front End</i>	<i>17</i>
3.2.1.4	<i>Express.js si Node.js Server Tier</i>	<i>17</i>
3.2.1.5	<i>MongoDB Database Tier</i>	<i>17</i>
3.2.1.6	<i>SQL vs NoSQL</i>	<i>18</i>
3.2.1.7	<i>MongoDB Atlas</i>	<i>19</i>
3.2.1.8	<i>De ce s-a ales MERN Stack</i>	<i>19</i>
3.2.2	<i>Arduino</i>	<i>19</i>
3.2.2.1	<i>Ce este un Arduino.....</i>	<i>19</i>
3.2.2.2	<i>Arduino UNO</i>	<i>22</i>
3.2.2.3	<i>Arduino Mega.....</i>	<i>23</i>
3.2.2.4	<i>Arduino IDE</i>	<i>24</i>
3.2.2.5	<i>Alte componente folosite pentru circuitele Arduino realizate in acest proiect</i>	<i>24</i>
3.2.3	<i>Proteus.....</i>	<i>27</i>
3.2.4	<i>Tensorflow</i>	<i>28</i>
3.2.4.1	<i>Object Detection (coco-ssd)</i>	<i>28</i>
3.2.4.2	<i>MediaPipe Handpose</i>	<i>29</i>
3.2.4.3	<i>Tensorflow.js</i>	<i>29</i>
3.2.4.4	<i>Fingerpose.....</i>	<i>30</i>
3.2.5	<i>React Icons.....</i>	<i>30</i>
3.2.6	<i>React Webcam.....</i>	<i>30</i>
3.2.7	<i>Axios.....</i>	<i>31</i>
4	PROIECTAREA SI IMPLEMENTAREA SISTEMULUI	32
4.1	ARHITECTURA APLICAȚIEI.....	32
4.2	CAZURI DE UTILIZARE	33
4.3	IMPLEMENTARE.....	35
4.3.1	<i>Aplicația de MERN Stack.....</i>	<i>35</i>
4.3.1.1	<i>Configurarea MongoDB Atlas.....</i>	<i>36</i>
4.3.1.2	<i>Configurarea Aplicației Web</i>	<i>37</i>

4.3.1.3	Crearea Backend-ului	38
4.3.1.4	Crearea Frontend-ului	40
4.3.1.5	Implementare funcționalități pe partea de Hardware	41
4.3.1.6	Implementare funcționalități pe partea de Frontend	43
4.3.1.7	Implementare funcționalități pe partea de Backend	44
4.3.1.8	Integrare Fullstack.....	45
5	TESTARE SI REZULTATE.....	46
5.1	METODE DE TESTARE.....	46
5.1.1	Testarea manuală a senzorilor	46
5.1.2	Testarea manuală a emulatorului	46
5.1.3	Testarea request-urilor pe partea de backend	46
5.1.4	Testarea request-urilor pe partea de frontend	46
5.1.5	Testarea interfeței grafice – cât de responsive e	47
5.1.6	Testarea modelelor de recunoaștere video.....	47
5.1.7	Testarea conexiunii fullstack (FE->DB, DB->FE)	47
5.2	CAZURI DE TESTARE.....	47
6	CONCLUZII.....	49
6.1	REZUMAT CONTRIBUȚII	49
6.2	REALIZAREA OBIECTIVELOR	49
6.3	DIRECȚII DE DEZVOLTARE.....	49
7	BIBLIOGRAFIE.....	52

1 Introducere

1.1 Context general

Sistemele de management al clădirilor (BMS: Building Management Systems), cunoscute și ca sistemele clădirilor automatizate, reprezintă controlul unei construcții (poate fi atât din interior cât și pentru exterior) folosind o rețea de microprocesoare inteligente instalate pentru a controla și monitoriza sisteme tehnice și serviciile clădirii.

Exemple de sisteme controlate: sistemul de securitate, încălzire, incendiu, ventilație, iluminare, electricitate; în această lucrare s-a implementat sistemul de securitate, urmând ca alte sisteme să fie introduse dezvoltării ulterioare. Aceste sisteme sunt esențiale pentru managementul operațiilor industriale. BMS și sistemele centralizate de management ale clădirilor cresc echipamentul electric și mecanic al unei clădiri și managementul infrastructurilor industriale.

Surse externe precum ASIS International, dețin studii a căror concluzie este că industria sistemelor BMS continuă să crească (15%-34% creștere pe an), estimându-se ca în anul 2022 să acopere o piață de 104 miliarde de dolari [1]. Numărul aplicațiilor de sisteme bms implementate este într-o continuă creștere. Primele sisteme bms acopereau doar controlul răcirii și căldurii clădirii, pentru a reduce din costuri. Astăzi este folosită o mare varietate de gadgeturi inteligente pentru a crește productivitatea angajaților, securiza afacerile și reduce costurile operaționale. Sistemele de management al clădirilor sunt folosite și pentru a monitoriza și securiza data centers, aeroporturi, spitale și hoteluri.

Odată cu dezvoltarea, dispozitivelor inteligente interconectate crește, obținând astfel un număr ridicat de entry points prin care clădirile pot fi compromise. Așadar, împreună cu multitudinea de beneficii operaționale ale tehnologiei clădirilor inteligente, toate aceste dispozitive și interconectivitatea lor aduc noi riscuri de securitate cibernetică.

Având sute de mii dispozitive între o singură clădire, suprafețele de atac potențiale de la dispozitivele nesecurizate sunt de norme, iar datele ce pot ajunge în mâna atacatorilor pot fi critice. Spre exemplu incidentul cibernetic produs asupra companiei Target: atacatorii au obținut acces la sistemul de vânzări din care au extras date ce conțin detaliile cardurilor de credit și debit asociate cu peste 110 milioane de conturi. Hackerii nu au atacat direct sistemul de vânzări (POS: point-of-sale), ci au început prin a fura credențiale folosite de furnizorii de încălzire, aer condiționat și ventilație, în momentul în care aceștia erau conectați la aplicațiile web targetate. În urma acestui atac, atacatorii au reușit să obțină accesul la directorul activ al companiei Target, ca mai pe urmă, să ajungă la sistemul POS de unde au colectat numerele cardurilor de credit și alte date personale [2].

Alte exemple în care sisteme BMS critice au fost atacate: ransomware attacks ce pot lua controlul sistemelor critice, cum a fost în cazul unui hotel din Austria, unde clienții au fost închiși în camerele lor de hotel [3]. Sau atacuri de tip denial-of-service, în care sistemele inteligente ale clădirilor sunt suprasolicitate pentru a întrerupe sisteme critice

ca și încălzirea în timpul iernii spre exemplu [4]. De asemenea, majoritatea companiilor dețin dispozitive inteligente. acestea pot fi sparte de către atacatori și se poate obține accesul la întregul sistem IT central, obținând mai departe toate datele companiei [5].

Sistemele BMS fac parte din clasa tehnologiilor operaționale (OT: Operational Technology), și diferă de sistemele standard IT primare prin gama largă de dispozitive, funcționalități și protocoale ale rețelelor. În aceste sisteme sunt incluse tehnologii embedded care rulează funcții fizice, generează date și comunică folosind protocoale OT specifice precum BACnet și LonWork. În urma protocoalelor specifice din BMS folosite și a multitudinii de dispozitive ce sunt implicate în sistem, clădirile inteligente aduc provocări mari ale securității fiind nevoie de soluții de securitate proiectate individual pentru fiecare provocare și clădire în parte.

Rețelele de tip OT și BMS au ca diferența principală motivul ocupării spațiului din incintă. În timp ce în fabrici (care sunt rețele de tip OT specifice) spațiul este bine delimitat pentru materiale și alte utilaje, în clădirile de tip BMS spațiul fizic nu este bine delimitat. Clădirile inteligente sunt caracterizate de un număr larg de vizitatori în incinta lor. Spre exemplu într-o clădire de birouri, un apartament complex, hotel sau chiar spital, sute sau chiar mii de vizitatori pot să intre zilnic. Definiția permisiunilor de acces este în continuă schimbare astfel crește și posibilitatea apariției unor noi breșe de securitate în sistemul prezent. Astfel identificarea anomaliilor în comportamentul acestor sisteme devine tot mai greu de identificat datorită aparițiilor continue de noi breșe și multiplicarea acestora. În general sistemele OT sunt distribuite pe toată suprafața clădirii cu multe puncte de acces în rețea.

O altă parte importantă cu care pot fi comparate sistemele BMS este privindu-le în oglindă cu sistemele de control industrial (ICS: Industrial Control Systems), crezând că securitatea ICS e controlată la fel ca și securitatea sistemelor inteligente. Problema este chiar din contra, sistemele BMS fiind cu mult mai distribuite în conexiuni decât sistemele ICS. Mai mult, sistemele inteligente conțin și dispozitive IoT (Internet of Things), ce nu sunt întâlnite așa des în sistemele ICS, și din păcate aceste dispozitive reprezintă adesea entry points pentru atacuri cibernetice, de cele mai multe ori de mare amploare.

1.2 Obiective

În timp ce trendul este apariția tot mai multor clădiri inteligente, complexitatea acestor clădiri crește la rândul ei. Tot mai multe aspecte ale managementului unei clădiri sunt digitalizate și manageriate de un punct central. Beneficiile unei clădiri mai inteligente decât înainte includ: reducerea costurilor operaționale, optimizarea consumtei de energie, și îmbunătățirea siguranței și securității personalului. Pentru a obține aceste beneficii, sistemele de management al clădirilor au nevoie de o conectivitate crescută: între fiecare BMS, între clădiri separate sau chiar între sisteme întregi de orașe inteligente prin internet.

Toate aceste progresii în conectivitatea clădirilor aduc, din păcate, și riscuri de securitate. Dispozitivele ce controlează senzorii și sistemele inteligente sunt acum mult mai predispuse la atacuri malițioase. Din cauză că aceste sisteme sunt critice deoarece

multe sunt în legătură cu date personale sau clasificate, atrag tot mai multe atacuri intenționate asupra targetului. Atacurile distribuite sau erori în tandem, urmate de alte evenimente de urgență, pot crește și mai mult potențialul la pagube semnificative cum ar fi pierderea datelor sau blocarea sistemului. De exemplu, erori simultane în sistemele de incendiu, controlul al ușilor, HVAC și a altor sisteme, în timpul unui incendiu, pot aduce rezultate dezastruoase.

Pentru a întâlnii provocările de securitate în rețelele BMS, este esențială deținerea unor sisteme de securitate care descoperă fiecare dispozitiv conectat în rețea, găsește orice dispozitiv necunoscut, și detectează activități care pot pune în pericol stabilitatea operațională ale acestor dispozitive critice. Prin monitorizarea traficului și comportamentului rețelelor OT și prin analiza protocoalelor folosite în BMS, precum LonWorks și BACnet, soluția ideală identifică rapid și monitorizează fiecare dispozitiv conectat la orice rețea și oferă 100% monitorizare asupra întregului trafic realizat de către toate dispozitivele.

Echipele de securitate ce se ocupă cu rețelele BMS trebuie să integreze soluții demonstrate pe fiecare subsistem, cum sunt spre exemplu punctele de acces control ale lifturilor la fiecare etaj al clădirii. Odată ce sunt conectați la rețeaua BMS, platforma unei soluții ideale de securitate trebuie să ofere vizibilitatea și capacitatea necesară de monitorizare pentru a obține control peste mediile lor complexe și de dimensiuni mari.

Principalele obiective acoperite în această lucrare sunt:

- Analiza aplicabilității și necesității acestei implementări de sistem.
- Studiarea facilităților oferite de aplicațiile de securitate pe mai multe nivele ale clădirilor, precum și cercetarea câtorva sisteme existente și a funcționalităților acestora.
- Conceperea unei arhitecturi fundamentale, aplicabilă unui sistem de securitate al unei clădiri, cu module participante la proces.
- Proiectarea și implementarea unei aplicații care permite monitorizarea și controlul automat al sistemelor de securitate din clădiri.
- Testarea aplicației și evaluarea rezultatelor obținute. Studiarea eventualelor posibilități de îmbunătățire a soluției software și hardware alese.

1.3 Specificații

Beneficiile ideale ale unei soluții de securitate în rețele BMS scalabile includ:

- Un sistem de monitorizare non-intrusiv/agentless cu influență zero asupra performanței dispozitivelor inteligente aflate în rețea.
- Abilitatea de a descoperi și inventaria toate dispozitivele aflate în clădire sau campus.
- O bază dinamică și adaptivă, care învață comportamentele normative și detectează automat orice anomalie sau activitate malițioasă în rețea.
- Un pachet întreg și larg de suport al inspecției protocoalelor folosite în BMS, ca și LonWorks sau BACnet.

- Trebuie să fie ușor de utilizat și operat atât pentru utilizatorii OT cât și utilizatorii IT.
- Independența soluției de securitate în conformitate cu arhitectura BMS ce e în continuă dezvoltare, abilitatea de a se adapta imediat la configurări noi și dispozitive trebuie să fie automatizată.
- Trebuie să securizeze sisteme critice, ca și HVAC (Heating, Ventilation, Air Conditioning), lifturi, camere de supraveghere și puncte de control acces.
- În cele din urmă, trebuie să integreze securitatea BMS OT, cu raport de 1 la 1, în controalele de securitate existente.

2 Studiu bibliografic

Sistemele de management al clădirilor sunt cel mai des introduse în proiecte de mare amploare ce conțin sisteme mecanice, electrice și HVAC extinse. Sistemele legate de un sistem inteligent de management al clădirii reprezintă de obicei, 40% din consumul de energie al unei clădiri. Dacă este inclusă iluminarea, acest număr se apropie de 70%. Acestea reprezintă o componentă esențială pentru gestionarea consumului de energie. Se spune că sistemele BMS cu configurări necorespunzătoare utilizează până la 20% din capacitatea totală energetică a clădirilor sau aproximativ 8% din consumul total de energie (rezultate calculate în statele Unite ale Americii, potrivit [6] și [7]).

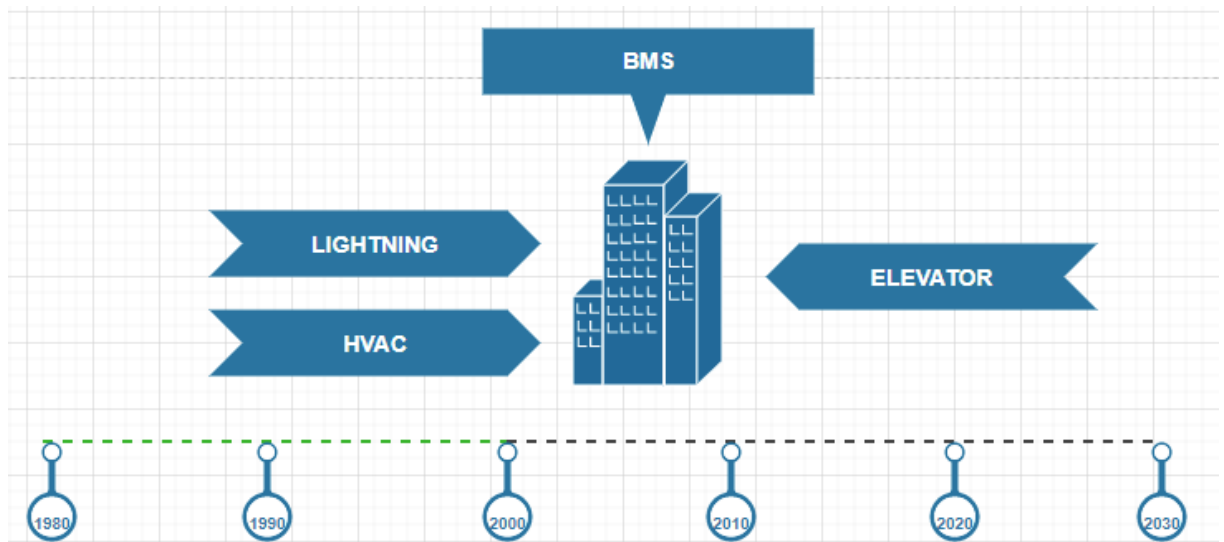
În afară de controlul mediului intern al clădirii, sistemele BMS au și posibilitatea de a controla accesul în clădire cu ajutorul sistemelor de turnichet și a ușilor de acces, sisteme care permit controlarea persoanelor care intră sau ies din clădire. De asemenea, un BMS poate controla și alte sisteme de securitate a clădirii, precum ar fi sistemele de televiziune cu circuit închis (CCTV) și detectoarele de mișcare. Sistemele de alarmă împotriva incendiilor și elevatoarele pot fi legate de un sistem BMS pentru o monitorizare mai amplă a tuturor sistemelor de protecție a clădirii. Acest aspect este esențial pentru siguranța persoanelor din clădire deoarece, în cazul în care se detectează un incendiu, doar panoul de alarmă de incendiu, din cadrul sistemului BMS, are capacitatea de a închide clapetele sistemului de ventilație pentru a opri răspândirea fumului, a închide manipulatorii de aer, a porni ventilatoarele de evacuare a fumului și a trimite toate elevatoarele la parter și a le bloca pentru a preveni utilizarea acestora.

Sistemele de gestionare a clădirilor au inclus și mecanisme de răspuns în caz de dezastre pentru a oferi clădirilor un avantaj în fața cutremurelor. În ultimul timp, întreprinderile și oficialii guvernelor au depus eforturi masive cu scopul de a găsi soluții similare de protecție și pentru zonele inundabile și zonele de coastă expuse riscului de creștere a nivelului mării. Un astfel de exemplu de BMS este SAFE Building System dezvoltat de către Arx Pax Labs Inc, sistem proiectat pentru a asigura plutirea clădirilor și drumurilor în câțiva metri de apă. Acest mediu plutitor cu auto-reglare se bazează pe tehnologii existente, folosite pentru a menține podurile și piste de beton deasupra apei, ca și în cazul podului Governor Albert D. Rosellini de pe ruta 520 din Washington și a Mega-Float din Japonia [8].

În data de 11 noiembrie 2019 a fost publicat un document de cercetare în domeniul securității de 132 de pagini intitulat "Eu Dețin Clădirea Dumneavoastră (Sisteme de management)" de Gjoko Krstic și Spike Mellema, document în care au abordat peste 100 de vulnerabilități care afectează diferite sisteme BMS și soluții de control al accesului ale mai multor companii [9].

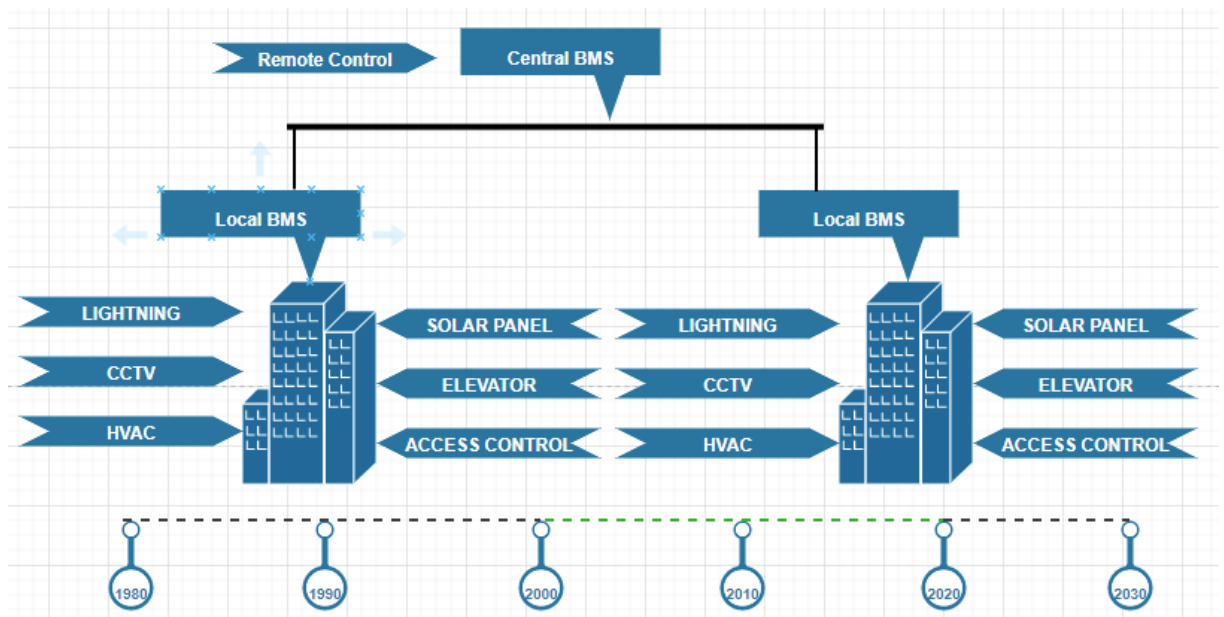
2.1 Evoluția sistemelor de management al clădirilor

Clădirile au oferit servicii foarte de bază, constând doar într-un sistem central de gestionare a clădirilor (BMS) și unul sau două subsisteme, cum ar fi HVAC, lifturi sau sisteme de iluminat, acestea fiind izolate între ele și în afara rețelilor.



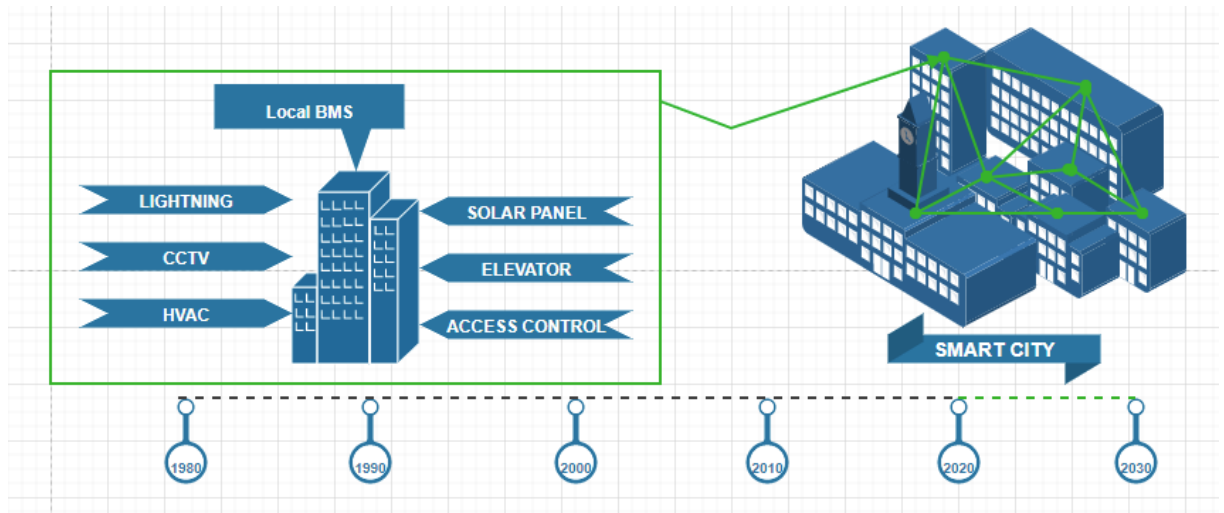
Figură 2.1 BMS de ieri

Clădirile de astăzi sunt clădiri inteligente, cu un BMS central care se poate integra cu BMS-uri locale a mai multor clădiri din rețeaua sa. Fiecare BMS local se conectează la diferite subsisteme, inclusiv HVAC, supraveghere, controlul accesului în clădire și sisteme energetice.



Figură 2.2 BMS de azi

Orașele inteligente reprezintă următorul pas, inevitabil, în această evoluție a BAS. „clădirile inteligente” BMS vor fi în curând în măsură să se integreze cu orice alte BMS locale de construcție, precum și infrastructura industrială din jurul lor.



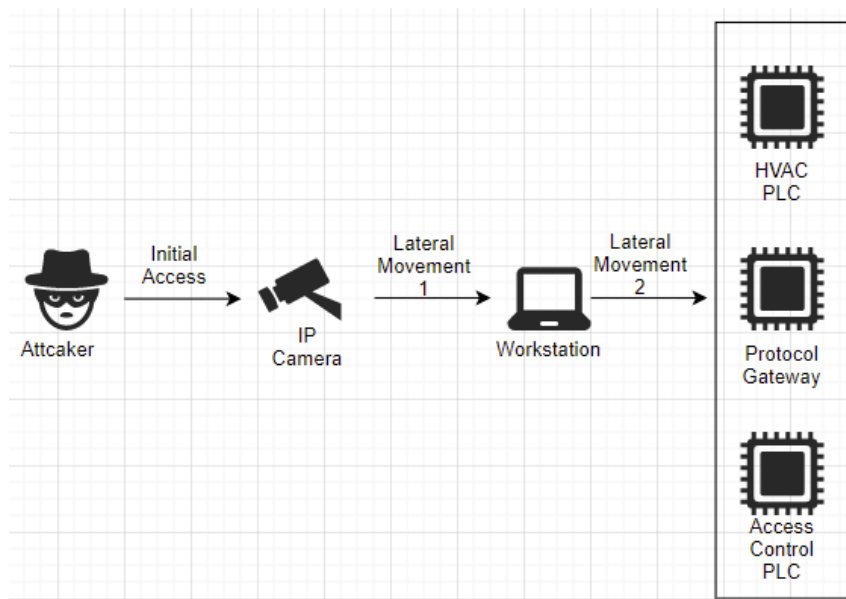
Figură 2.3 BMS de maine

Conform [10], în 2026 vor fi peste 56 milioane de noi dispozitive BAS. De asemenea, numărul vulnerabilităților în BAS a crescut cu peste 500% în ultimii 3 ani, conform [11]. 39.3% din dispozitivele BAS la care are access publicul, sunt vulnerabile (din modulele folosite în research). Aceste dispozitive sunt în mare parte PLC-uri HVAC, PLC-uri pentru control acces și Protocol Gateways.

2.2 Potentiale puncte de acces într-un sistem BAS

2.2.1 Camerele de supraveghere video

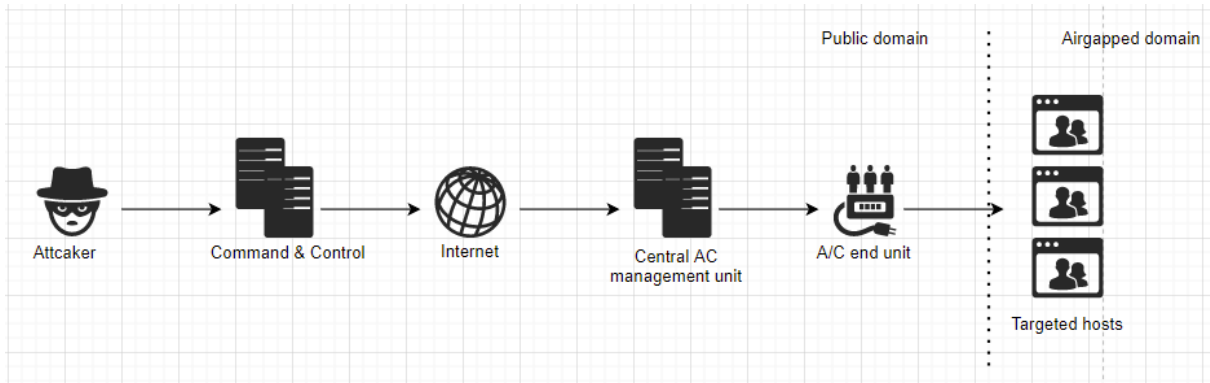
Conform [12] dintr-un total de 11269 IP-Cameras accesibile publicului, 10312 dintre acestea au fost vulnerabile la exploits (91.5%). Atacatorii pot să folosească acest canal de comunicare pentru a vizualiza virtual clădirea și extrage informații senzitive din aceasta (ei pot obține control asupra altor subsisteme la nivelul automatizării, sau obține control la nivel de management pentru a orchestra atacuri mai mari, coordonate).



Figură 2.4 Atac asupra camerelor de supraveghere

2.2.2 Sisteme HVAC

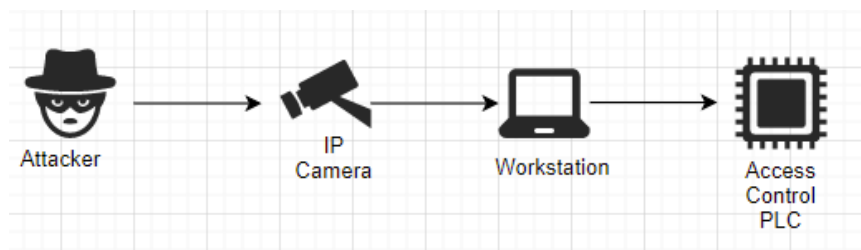
Atacatorii pot să folosească sistemele de tip HVAC pentru a crea turbulențe în sistemul de ventilație sau cel termic. Aceste atacuri diferă de la creșterea temperaturii într-un data center pentru a cauza o perturbare a zilei de muncă, până la obținerea controlului asupra rețelei network pentru a orchestra atacuri coordonate mai mari și distrugerea aparaturii fizice.



Figură 2.5 Atac asupra sistemelor HVAC

2.2.3 Control acces

Atacatorii pot folosi id-uri de acces compromise, cititoare de carduri, controlere și baze de date care stochează credențialele angajaților/ clienților pentru a controla ușile și obține acces la încăperi restricționate, bloca persoanele în incinta clădirii pentru a cere răscumpărare, sau pentru a obține acces la rețeaua de management pentru a orchestra atacuri coordonate mai mari.



Figură 2.6 Atac asupra punctelor de control access

Metodologia unui atac cibernetic:

Recunoaștere:

- Adunarea informațiilor despre target.
- Căutarea rețelelor network și tehnologiilor folosite pentru a găsi posibile vulnerabilități.
- Găsirea unor puncte de acces în sistem.

Cercetare:

- Adunarea exploit-urilor existente.
- Găsirea 0-days dacă există.

Pregătire:

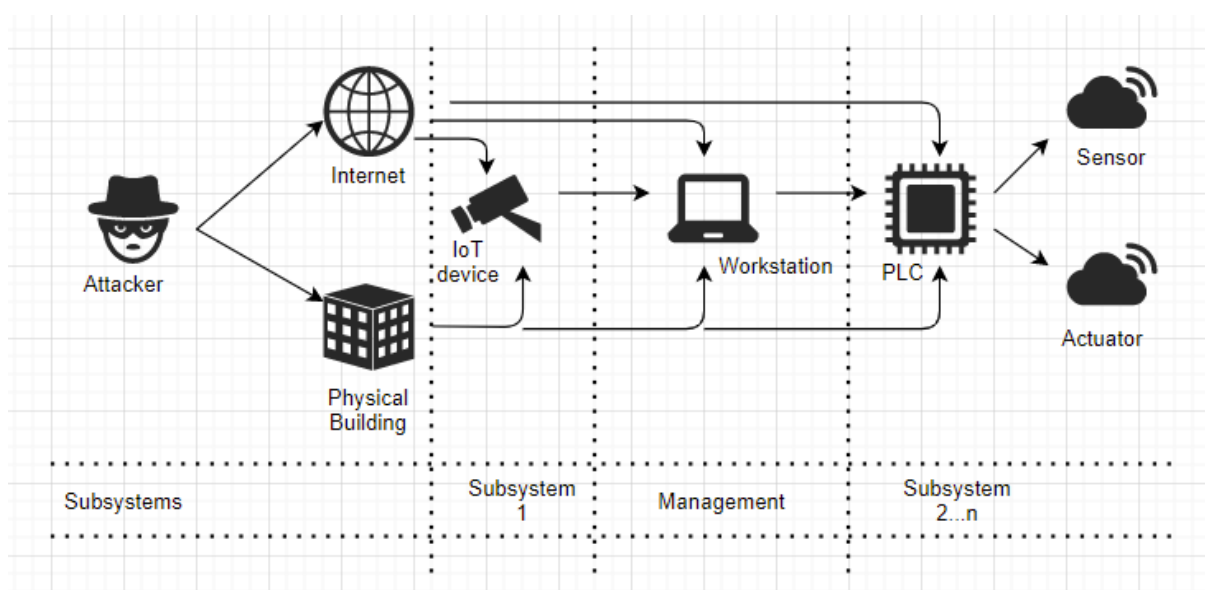
- Planificarea unui atac.
- Dezvoltarea atacului (implementarea în sine, de exemplu dacă sunt fișiere de cod ce vor fi rulate).

Compromitere:

- Compromiterea punctelor de acces.
- Mișcare laterală.
- Executare de cod.

Persistența:

- Persistă și după repornirea sistemelor.
- Își curăță trace-urile.



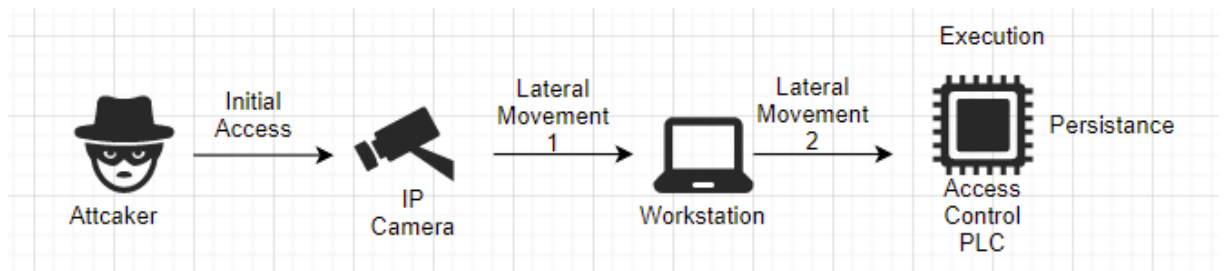
Figură 2.7 Posibile cai de atac

1. PLC-uri accesibile public: Folosind această cale, malware-ul poate intra direct de pe Internet și exploata programabil controlerele logice (PLC-uri) care controlează senzorii și actuatorii la nivelul câmpului, astfel încât nu este nevoie de nicio comandă laterală sau deplasarea la alte dispozitive.

2. Stații de lucru accesibile public: Folosind această cale, malware-ul poate introduce o stație de lucru de pe Internet la nivelul de gestionare și deplasare laterală către PLC-uri.

3. Dispozitive IoT accesibile publicului: Folosind această cale, malware-ul poate introduce un dispozitiv IoT, cum ar fi o cameră IP sau un router WiFi, de pe Internet și de a folosi acel punct de intrare pentru a obține acces la rețeaua internă, de obicei, trecerea la nivelul de management mai întâi și apoi către alte subsisteme.

4. Rețea prin aer: Folosind această cale, atacatorul trebuie să aibă acces fizic la rețeaua de construcție (care ar putea fi realizat prin sistemul HVAC) reușind prin aceasta deplasare laterală pentru a ajunge la PLC-uri.



Figură 2.8 Plan de atac

Pasul 1 – accesul inițial

Camera IP poate fi exploatată utilizând o combinație de CVE-2018-10660 [13], CVE-2018-10661 [14] and CVE-2018-10662 [15]. Vulnerabilitățile și exploatarea sunt pe baza activității, credențialelor default sau utilizând exploit-uri cunoscute, găsite în biblioteci precum Peles [16] și Metasploit [17].

Pasul 2 - mișcare laterală 1

Odată intrat în device, malware-ul curăță urmele sale lăsate în loguri prin editarea fișierelor log, sau trace (ex.: /var/volatile/log/{auth,info}.log), apelează netstat și caută stația de lucru conectată la dispozitivul de pe care e conectat (utilizată pentru comunicațiile video înregistrate în rețea) și se mută de la cameră la stația de lucru prin exploatarea serverului MS-SQL dacă acesta este configurat greșit (din nou, credențiale default care e cea mai întâlnită problemă în securitatea sistemelor IoT sau alte exploit-uri cunoscute deja, la care nu s-a făcut patch).

Pasul 3 – mișcare laterală 2

În timp ce rulează pe stația de lucru, malware-ul caută punctul de control acces la toate PLC-urile de lucru și citește fișierele lor de configurații pentru a găsi dispozitivele conectate și a infecta cât mai multe dintre acestea, până când ajunge pe dispozitivul de unde poate obține controlul asupra sistemului dorit.

Pasul 4 – executare

După ce a ajuns pe dispozitivul țintă, primul obiectiv este de a perturba comportamentul normal al PLC-ului, adăugarea unui utilizator nou și a unui ecuson nou în baza de date, dând accesul la o altă persoană neautorizată.

Pasul 5 – persistență

După ce sarcina finală a fost executată, malware-ul poate să persiste pe dispozitiv chiar și după reporniri.

2.2.4 Vulnerabilități ce pot fi descoperite într-un atac

#	Product	Vulnerability Type	Notes
---	---------	--------------------	-------

1	Protocol Gateway	XSS	0-day patched by the vendor and CVE assigned
2	Protocol Gateway	Path traversal	0-day patched by the vendor and CVE assigned
3	Protocol Gateway	Arbitrary file deletion	0-day patched by the vendor and CVE assigned
4	HVAC PLC	XSS	0-day patched by the vendor and CVE assigned
5	HVAC PLC	Authentication bypass	0-day patched by the vendor and CVE assigned
6	AccessControl PLC	XSS	0-day patched by the vendor and CVE assigned
7	AccessControl PLC	Hardcoded secret	Not 0-day; the vulnerability was known and patched by the vendor, but never disclosed.
8	AccessControl PLC	Buffer overflow	Not 0-day; the vulnerability was known and patched by the vendor, but never disclosed.

Tabel 2.1 Vulnerabilități descoperite

Vulnerabilitățile XSS permit unui atacator să injecteze scripturi rău intenționate în interfețe web de încredere care rulează pe dispozitivele vulnerabile, care poate fi executat de către browser-ul unui utilizator nebănuitor pentru a accesa cookie-uri, sesiuni de jetoane, sau alte informații sensibile, precum și efectuarea unor acțiuni rău intenționate în numele utilizatorului.

Vulnerabilitățile de traversare a căii și de ștergere a fișierelor permit unui atacator să manipuleze referințele căii și accesarea sau ștergerea fișierelor și directoarelor (inclusiv fișierele critice de sistem) care sunt stocate în afara folderului root aplicației web ce rulează pe dispozitiv.

Vulnerabilitatea de bypass de autentificare permite unui atacator să fure informațiile de acreditare ale utilizatorilor aplicației, inclusiv text simplu parole, prin manipularea identificatorului de sesiune trimis într-o solicitare.

Cele mai grave vulnerabilități sunt problemele #7 și #8, care permit unui atacator de la distanță executarea unui cod arbitrar pe dispozitivul țintă și obținerea controlului complet asupra acestuia.

- secret hardcodat: Cadrul Java utilizat pentru Control acces PLC și pe sistemul software. În acest exemplu se pot găsi configurații de control aflate în fișiere de tipul daemon.properties și configurații de aplicații în fișiere numite config.bog, care este un xml comprimat. Aceste fișiere conțin

nume de utilizator și parole, printre alte informații. Parolele sunt hashed sau criptate în funcție de versiunea cadrului.

- Buffer overflow: Există un daemon binar care rulează pe Control Access PLC care expune mai multe puncte finale HTTP ce face posibilă utilizatorilor de la distanță, accesarea pentru a gestiona dispozitivul.

2.2.5 Cum se pot reduce riscurile din rețelele de tip BAS

Aceste riscuri pot fi reduse prin implementarea unor soluții de securitate care să ofere:

Vizibilitate completă asupra dispozitivelor:

- Se stabilește automat inventarul dispozitivelor active prin amprentarea dispozitivului.
- Evaluează automat vulnerabilități comune & expuneri (CVE) pentru dispozitive.

Detecrie a amenințărilor în timp real:

- Monitorizează continuu rețea pentru modificări în comportament.
- Verifică automat comportamentul dispozitivului și evaluează dacă apar indicatori de amenințare, sau găsește activități de nerespectare a protocoalelor standard pre-definite.
- Alerte în timp real cu vizualizări interactive de amenințări și riscuri.

Securitate IT-OT convergentă:

- Monitorizează atât rețelele IT, cât și OT dintr-un singur ecran.
- Oferă automatizare extinsă, capacități cross-funcționale.

Sistemele de automatizare a clădirilor (BAS) pot fi la fel de critice ca și sistemele de control industrial (ICS) din punct de vedere al siguranței și securității, primesc totuși mult mai puțin atenția comunității de securitate.

Îmbunătățirea programelor de securitate cibernetică BAS prin vizibilitatea dispozitivelor și monitorizarea rețelei poate oferi organizațiilor o înțelegere aprofundată a mediului și conexiunile lor, facilitând proiectarea eficientă a arhitecturi de securitate, identificarea vectorilor de atac și localizarea punctelor oarbe.

3 Analiză si fundamentare teoretica

3.1 Concepte generale

Într-adevăr se găsesc multe soluții de securitate pe piață, însă majoritatea nu sunt așa cum s-ar aștepta oricine. Totul merge spre inovare, apar aplicații ce automatizează task-uri de pretutindeni (de exemplu, aplicații pentru a rezerva slot-uri în camera de jocuri din marile corporații, care de cele mai multe ori nu sunt secure, angajații crezând că doar ei au acces la aplicație, însă acestea pot reprezenta un entry point în rețeaua network a companiei). Apar inovații peste tot, însă majoritatea companiilor de securitate rămân la vechiul concept că instalează doar senzorii, și dacă se strică ceva, trimit o echipă să vadă și să repare. Însă securitatea nu constă doar în asta. Cine monitorizează traficul pe internet? Dar senzorii? Pot fi achiziționate servicii externe specializate pe fiecare dintre aceste nevoi (de exemplu un departament IT care se ocupă de trafic, o companie de security care se mai plimbă prin încăperi și verifică să nu fie ceva suspect (dispozitive noi conectate, persoane neautorizate, etc)). Foarte puține companii au trecut la nivelul următor și încearcă să țină pasul cu nevoile companiilor.

În cadrul acestei lucrări, s-a efectuat un studiu de piață în care s-au analizat serviciile de securitate prezente și ce oferă acestea. S-a constatat că sunt foarte puține cele care oferă servicii cu adevărat inteligente (asta pe plan global, nu doar în România, deci situația nu e chiar așa de preferabilă în acest domeniu). Mai jos sunt prezentate câteva din puținele soluții găsite care totuși oferă un pachet mai complet de servicii de securitate (de exemplu prin recunoașterea unor scenarii din înregistrările video în timp real, sau monitorizarea senzorilor în clădiri).

Xeoma este o platformă de supraveghere video inteligentă ce conține o gamă largă de diferite module smart. Recunoaștere facială, numărător de persoane, recunoaștere vârstă, recunoaștere și clasificare obiecte, detecție lipsă obiect, recunoaștere plăcuțe de înmatriculare și recunoașterea emoțiilor sunt câteva din modulele smart pe care Xeoma le oferă clienților. Xeoma poate fi folosită ca și aplicație Web, aplicație mobil sau aplicație Desktop.

Avigilon este o aplicație Desktop, cu posibilitate de integrare în aplicațiile Web, ce permite supraveghere video inteligentă, gestionare senzori și sisteme securizate de acces. Printre opțiunile de supraveghere inteligentă se regăsesc recunoașterea facială, recunoaștere plăcuță de înmatriculare, detectare de activitate supicioasă și detectare de mișcare neobișnuită.

SmartPSS, disponibilă pe Windows și Mac, este o aplicație de securitate pentru companii, principalele obiective fiind supravegherea video și managementul alarmelor. Există și opțiunea de a afișa camerele și senzorii pe o hartă electronică, încărcată în aplicație de către client, cu scopul vizualizării a tuturor sistemelor.

Innovative Security Manager (ISM) este o platformă de gestionare a dispozitivelor de siguranță și de organizare a resurselor din cadrul unei companii. Supraveghere video și planuri interactive a etajelor, locație GPS în timp real a vehiculelor, management al

alarmelor și detectoarelor de pericole precum și gestionarea personalului și a resurselor sunt principalele puncte cheie, ale platformei, cu care ajută la păstrarea unui mediu sigur în cadrul unei organizații. Platforma se poate integra în cadrul unui ERP proprietar dar se poate folosi și ca o aplicație Web. Security Operation Center din cadrul Aeroportului din Copenhaga folosește platforma ISM pentru a asigura securitatea aeroportului, iar Politia Națională din Danemarca folosește aceasta platformă pentru gestionarea alarmelor, autovehiculelor și a personalului.

Aceste servicii de securitate oferă soluții inteligente la problemele actuale în clădirile de birouri, bănci, spații comerciale, case, spitale, hoteluri, și multe altele. Din moment ce sunt atât de puține soluții ce conțin pachete ce se aproprie puțin de un pachet complet, în această lucrare s-a încercat aducerea unei soluții inteligente inovative care să ajute la dezvoltarea acestei ramuri ale industriei, într-un mod eficient și smart.

3.2 Tehnologii utilizate

3.2.1 MERN Stack

3.2.1.1 Ce este MERN Stack

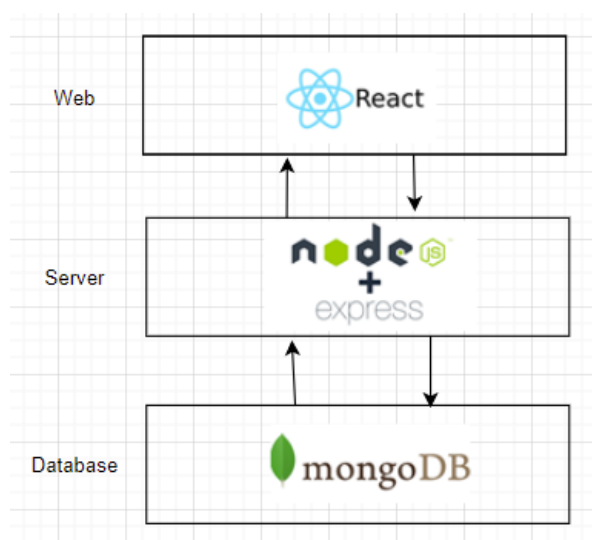
MERN reprezintă MongoDB, Express, React, Node, după cele patru tehnologii cheie care alcătuiesc stiva.

- MongoDB - baza de date de documente
- Express (.js) - Cadrul web Node.js
- React (.js) - un cadru JavaScript pe partea clientului
- Node (.js) - serverul web principal JavaScript

MERN este una dintre câteva variante ale stivei MEAN (MongoDB Express Angular Node), unde cadrul tradițional Angular.js folosit pentru partea de frontend este înlocuit cu React.js. Alte variante includ MEVN (MongoDB, Express, Vue, Node), sau chiar și orice alt cadru JavaScript de frontend poate funcționa. Express și Node alcătuiesc nivelul mediu (aplicația în sine). Express.js este un cadru web din partea serverului, iar Node.js este cea mai populară și puternică platformă de server JavaScript. Indiferent de varianta aleasă, ME (RVA) N este abordarea ideală pentru a lucra cu JavaScript și JSON, pentru a construi aplicații integrale ce conțin frontend-ul, backend-ul și baza de date (full-stack).

3.2.1.2 Cum funcționează MERN Stack

Arhitectura MERN permite construirea cu ușurință a unei arhitecturi pe 3 niveluri (frontend, backend, bază de date) în întregime folosind JavaScript și JSON .



Figură 3.1 Arhitectura MERN

3.2.1.3 React.js Front End

Nivelul superior al stivei MERN este React.js, cadrul JavaScript declarativ pentru crearea de aplicații dinamice din partea clientului în HTML. React permite construirea unei interfețe complexe prin componente simple, conectarea la datele de pe serverul backend și redarea ca HTML. Punctul forte al React-ului este manipularea interfețelor, bazate pe date, cu cod minim și cu un efort minim, având tot ce este de așteptat de la un cadru web modern: un suport excelent pentru formulare, gestionare a erorilor, evenimente, liste și multe altele.

3.2.1.4 Express.js si Node.js Server Tier

Următorul nivel, mergând în jos, este cadrul Express.js din partea serverului, care rulează în interiorul unui server Node.js. Express.js se prezintă ca un „cadru web rapid, neopinionat, minimalist pentru Node.js” și într-adevăr exact asta este. Express.js are modele puternice pentru rutarea adreselor URL (potrivirea unei adrese URL primite cu o funcție de server) și gestionarea cererilor și răspunsurilor HTTP.

Prin efectuarea de solicitări HTTP XML (XHRs) sau GET-uri sau POST-uri din partea frontală React.js, se poate face conectarea la funcțiile Express.js care alimentează aplicația. La rândul lor, aceste funcții folosesc driverele Node.js ale MongoDB, fie prin callback-uri, fie prin utilizarea promisiunilor, pentru a accesa și actualiza datele din baza de date MongoDB.

3.2.1.5 MongoDB Database Tier

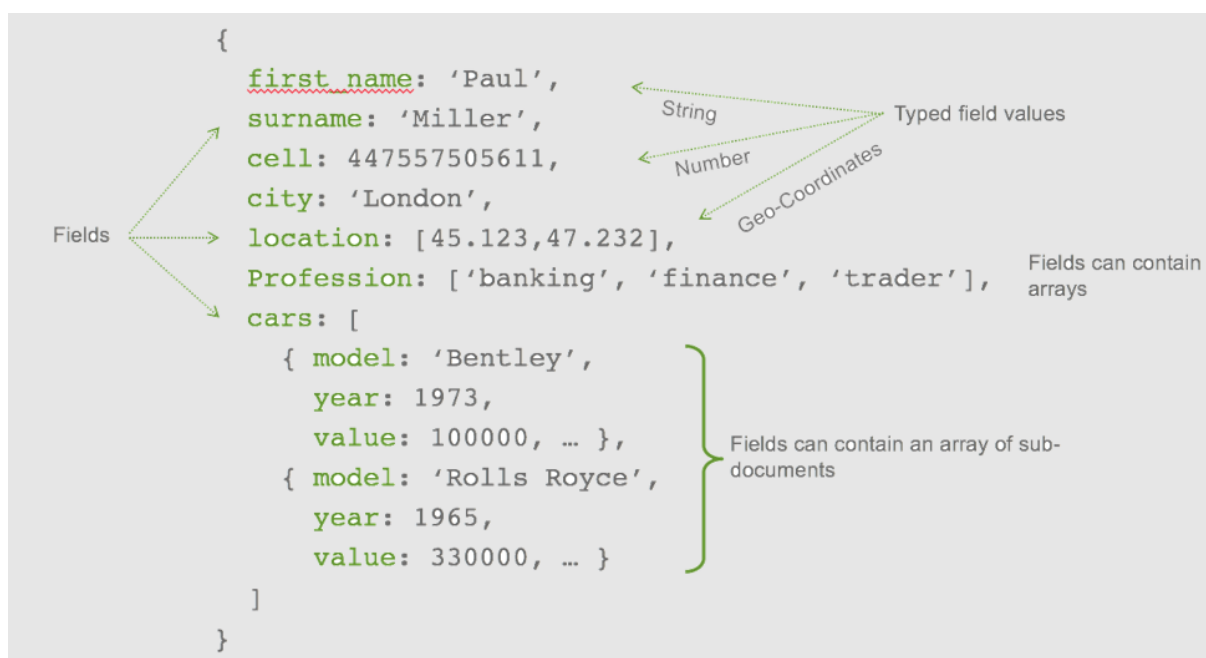
Dacă aplicația stochează date (profiluri de utilizator, conținut, comentarii, încărcări, evenimente etc.), atunci se dorește o bază de date la fel de ușor de lucrat ca și în React, Express și Node. Aici intervine MongoDB: documentele JSON create în partea frontală React.js pot fi trimise serverului Express.js, unde pot fi procesate și (presupunând că sunt valabile) stocate direct în MongoDB pentru regăsirea ulterioară. Dacă se construiește în cloud, cel mai probabil va fi utilizat Atlas.

3.2.1.6 SQL vs NoSQL

În lumea tabelară sau relațională, apar noțiuni precum baze de date, tabele, rânduri etc.. MongoDB are concepte similare, dar folosește termeni diferiți. În loc de tabel, avem colecții. În loc de rânduri, avem documente. Putem face operațiuni JOIN cu operatorul de căutare \$. Și în loc de Foreign Keys folosim referințe. MongoDB este foarte potrivit pentru manipularea datelor cu o mare varietate de relații.

Tabular (Relational)	MongoDB
Database	Database
Table	Collection
Row	Document
Index	Index
Join	\$lookup
Foreign Key	Reference

Tabel 3.1 SQL vs NoSQL



Figură 3.2 Document MongoDB [18]

MongoDB stochează date pe disc în format BSON sau Binary JSON. Aceasta oferă o mare varietate de suport pentru tipurile de date, cu mult peste cele acceptate de JSON, cum ar fi Decimal128, ISODate și multe altele. Modelul de document permite, de asemenea, nesting-ul documentelor în interiorul celui alt. Aceste subdocumente sunt unul dintre lucrurile inovatoare despre modelul de document. Permite să se aplice conceptul de „Datele ce sunt accesate împreună, sunt stocate împreună” în aplicație.

De asemenea, există capacitatea de a stoca informații în arrays, care este o altă caracteristică puternică a modelului de document. Aceste documente sunt obiecte structurate JSON, acesta fiind modul în care majoritatea dezvoltatorilor moderni gândesc

structurile. O persoană este un obiect care are diverse atribute, cum ar fi un titlu, o adresă etc. Acest lucru permite practicilor moderne de dezvoltare să utilizeze modelul de document într-un mod foarte intuitiv, fără a fi nevoie să se desprindă datele pentru a le pune în tabele și a normaliza lucrurile.

3.2.1.7 MongoDB Atlas

Baza de date MongoDB poate fi găzduită local, dar și în cloud, a doua opțiune fiind mai ușoară, folosind MongoDB Atlas. Se creează un cont pe site-ul MongoDB Atlas. După autentificare se creează un nou proiect și apoi un cluster. În Cluster putem accesa bazele de date, verifica diferite Metrics, seta IP Whitelists și multe alte opțiuni.

3.2.1.8 De ce s-a ales MERN Stack

Să începem cu MongoDB, baza de date a documentelor de la rădăcina stivei MERN. MongoDB a fost proiectat pentru a stoca datele JSON în mod nativ (folosește tehnic o versiune binară a JSON numită BSON), iar totul, de la interfața liniei sale de comandă până la limbajul său de interogare (MQL, sau MongoDB Query Language) este construit pe JSON și JavaScript. MongoDB funcționează extrem de bine cu Node.js și face stocarea, manipularea și reprezentarea datelor JSON la fiecare nivel al aplicației incredibil de ușor. Pentru aplicațiile native cloud, MongoDB Atlas oferă un cluster MongoDB cu scalare automată pe furnizorul de cloud la alegere, la fel de ușor în câteva clicuri pe buton.

Express.js (care rulează pe Node.js) și React.js fac ca aplicația JavaScript / JSON MERN să fie completă. Express.js este un cadru de aplicații din partea serverului care înfășoară cererile și răspunsurile HTTP și face ușoară maparea adreselor URL către funcțiile din partea serverului. React.js este un cadru JavaScript frontend pentru construirea de interfețe de utilizator interactive în HTML și comunicarea cu un server de la distanță.

Combinția înseamnă că datele JSON curg în mod natural din față în spate, ceea ce face ca acesta să se stabilizeze rapid și să fie destul de simplu de depanat. În plus, trebuie să se cunoască doar un limbaj de programare și structura documentului JSON, pentru a înțelege întregul sistem. MERN este teancul ales pentru dezvoltatorii web de astăzi care doresc să se deplaseze rapid, în special pentru cei cu experiență React.js.

3.2.2 Arduino

3.2.2.1 Ce este un Arduino

Placa Arduino a fost proiectată în Ivrea Interaction Design Institute, destinată studenților fără o experiență în electronică și concepte de programare. Aceste plăci Arduino au început să se modifice pentru a se adapta la noile cerințe și provocări, separându-și prezentul de plăci simple pe 8 biți de produsele pentru aplicații IoT (Internet of Things), imprimare 3D, aplicații pentru industria textilă și aplicații încorporate. Toate plăcile sunt complet open-source, permițând utilizatorilor să le construiască separat și să le adapteze în cele din urmă la nevoile lor exacte. De-a lungul anilor, plăcile Arduino au fost folosite pentru a construi mii de proiecte, de la obiecte zilnice la instrumente științifice compuse. O comunitate internațională de designeri, artiști, studenți, programatori, pasionați și experți s-au reunit în jurul acestei etape open source, donațiile

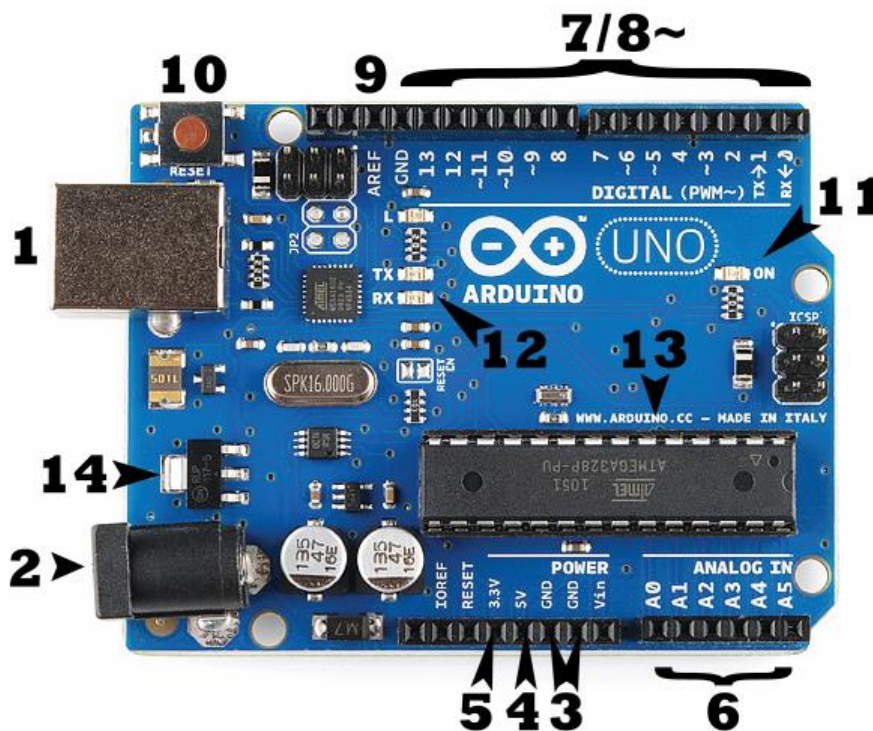
lor s-au adăugat la o cantitate incredibilă de cunoștințe disponibile care pot fi de mare ajutor pentru începători și specialiști. Arduino poate interacționa cu butoane, LED-uri, motoare, boxe, unități GPS, camere de luat vederi, internet și chiar telefonul inteligent sau televizorul. Această flexibilitate combinată cu faptul că software-ul Arduino este gratuit, plăcile hardware sunt destul de ieftine, și atât software-ul, cât și hardware-ul sunt ușor de învățat, a dus la o comunitate mare de utilizatori care au contribuit cu coduri și au lansat instrucțiuni pentru o mare varietate de proiecte bazate pe Arduino.

Pentru orice, de la roboți, pătură de încălzire a mâinii, până la mașini oneste de vânzare a averii și chiar o manevră de aruncare a zarurilor Dungeons and Dragons, Arduino poate fi folosit ca și creierul din spatele aproape oricărui proiect electronic.

Placa Arduino nu este un Microcontroller, ci este o platformă electronică open source. Placa Arduino este un PCB care are Microcontrolere, LED-uri și multe alte conexiuni. În general, este utilizat pentru a efectua operațiuni de intrare și ieșire, cum ar fi controlul unui motor, citirea de la senzor și în calcule mici. Există diferite plăci Arduino, printre care:

- Arduino UNO (R3)
- LilyPad Arduino
- Red Board
- Arduino Mega (R3)
- Arduino Leonardo

Majoritatea Arduino-urilor au aceste componente în comun:



Figură 3.3 Componente placa Arduino

Power (USB / Barrel Jack):

Fiecare placă Arduino are nevoie de o modalitate de a fi conectată la o sursă de alimentare. Arduino UNO poate fi alimentat cu un cablu USB provenit de la computer sau de la o sursă de alimentare din perete, care este încheiată cu un barrel jack. În imaginea de mai sus este etichetată conexiunea **USB (1)**, iar **mufa barrel-ului este (2)**. Conexiunea USB este, de asemenea, modul se încarcă codul pe placa Arduino. Tensiunea recomandată pentru majoritatea modelelor Arduino este cuprinsă între 6 și 12 volți.

Pini (5V, 3.3V, GND, Analog, Digital, PWM, AREF):

Pinii de pe Arduino sunt locurile în care sunt conectate firele pentru a construi un circuit. De obicei, au „headers” din plastic negru care permit conectarea doar un fir în placă. Arduino are mai multe tipuri diferite de pini, fiecare fiind etichetat pe placă și utilizat pentru diferite funcții:

- **GND (3):** Scurt pentru „Ground”. Pe Arduino există mai mulți pini GND, oricare dintre acestea putând fi folosit pentru a pune împământarea circuitului
- **5V (4) și 3.3V (5):** Pinul de 5V furnizează 5 volți de putere, iar pinul de 3,3 V furnizează 3,3 volți de putere. Majoritatea componentelor simple utilizate cu Arduino rulează 5 sau 3,3 volți.
- **Analog (6):** Zona de pini sub eticheta „Analog In” (A0 până la A5 pe UNO) sunt pini analogici. Acești pini pot citi semnalul de la un senzor analogic (precum un senzor de temperatură) și îl pot converti într-o valoare digitală ce poate fi citită de utilizatori.
- **Digital (7):** Peste pinii analogici sunt pinii digitali (0 până la 13 pe UNO). Acești pini pot fi folosiți atât pentru intrare digitală (cum ar fi să spunei dacă este apăsat un buton), cât și pentru ieșire digitală (cum ar fi alimentarea unui LED).
- **PWM (8):** Caracterizat prin prezența tildei (~) lângă unii dintre pinii digitali (3, 5, 6, 9, 10 și 11 pe UNO). Acești pini acționează ca pini digitali normali, dar pot fi folosiți și pentru Pulse-Width Modulation (PWM). Acești pini sunt capabili să simuleze ieșirea analogică (cum ar fi decolorarea unui LED în interior și în exterior).
- **AREF (9):** Reprezintă o referință analogică. De cele mai multe se poate lăsa acest pin singur. Este uneori utilizat pentru a seta o tensiune de referință externă (între 0 și 5 volți) ca limită superioară pentru pinii de intrare analogici.

Reset Button:

Arduino are un **buton de resetare (10)**. Apăsând-ul va conecta temporar pinul de resetare la împământare și va reporni orice cod încărcat pe Arduino. Acest lucru poate fi foarte util dacă codul nu se repetă, dar se dorește testarea de mai multe ori. Restartarea pe Arduino nu rezolvă de obicei probleme.

Power LED Indicator:

Chiar sub și în dreapta cuvântului „UNO” de pe placa de circuit, există un LED minuscul lângă cuvântul „ON” (11). Acest LED ar trebui să se aprindă ori de câte ori e conectat Arduino la o sursă de alimentare. Dacă această lumină nu se aprinde, există șanse mari să nu fie ceva în regulă, poate cablul de alimentare e stricat, led-ul, sau chiar placa.

TX RX LEDs:

TX este scurt pentru transmisie (transmit), RX este scurt pentru primire (receive). Aceste marcaje apar destul de mult în electronică pentru a indica pinii responsabili de comunicarea în serie. În cazul de față, există două locuri pe Arduino UNO unde apar TX și RX - o dată cu pinii digitali 0 și 1, și a doua oară lângă LED-urile indicatoare **TX și RX (12)**. Aceste LED-uri oferă câteva indicații vizuale ori de câte ori Arduino-ul primește sau transmite date (cum ar fi atunci când este încărcat un nou program pe placă).

Main IC:

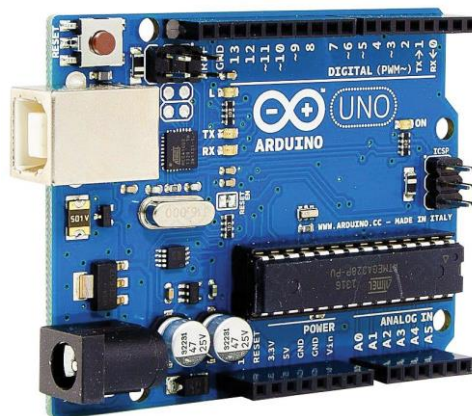
Este un IC sau un **circuit integrat (13)** ce reprezintă creierul Arduino. IC-ul principal de pe Arduino este ușor diferit față de alte tipuri de plăci, dar face parte de obicei de la linia ATmega a IC-urilor de la compania ATMEL. Acest lucru poate fi important, deoarece poate fi necesară cunoașterea tipului de IC (împreună cu tipul de placă) înainte de a încărca un nou program din software-ul Arduino. Aceste informații pot fi găsite de obicei în scris în partea de sus a IC-ului.

Voltage Regulator:

Regulatorul de tensiune (14) nu este de fapt ceva cu care se poate interacționa pe Arduino. Regulatorul de tensiune face exact ceea ce spune - controlează cantitatea de tensiune care este lăsată în placa Arduino. Acesta va opri o tensiune suplimentară care ar putea dăuna circuitului, având desigur limitele sale.

3.2.2.2 Arduino UNO

Arduino UNO R3 este un noua placă și, în comparație cu plăcile anterioare Arduino, are câteva caracteristici suplimentare. Arduino UNO folosește Atmega16U2 în loc de 8U2 și permite o rată de transfer mai rapidă și mai multă memorie. Nu este nevoie de dispozitive suplimentare pentru Linux și Mac și posibilitatea de a afișa UNO ca tastatură, mouse, joystick etc.



Figură 3.4 Arduino UNO [19]

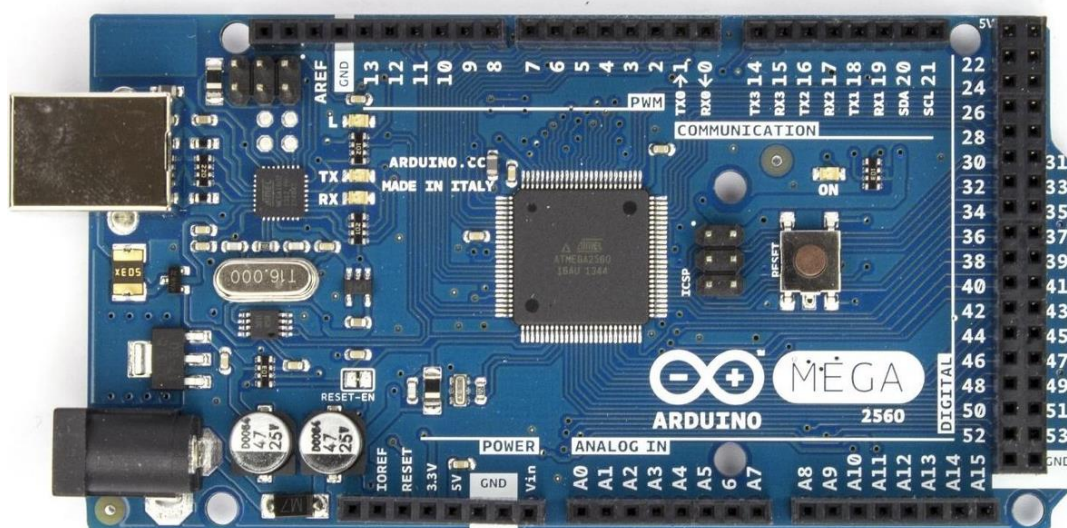
Arduino R3 adaugă pini SDA & SCL care se află lângă AREF și, în plus, există doi pini care sunt așezați lângă pinul RESET. Primul pin este IOREF; acesta va permite scuturilor să se adapteze tensiunii de pe placă. Celălalt pin nu este conectat și este rezervat pentru utilizări ulterioare. Funcționarea Arduino R3 se face prin toate scuturile existente și va adapta noi scuturi care folosesc acești pini suplimentari.

Caracteristici:

- Microcontroller ATmega328
- Tensiune de operare: 5V
- Tensiune de alimentare recomandată: 7-12V
- Limită de tensiune: 6-20V
- Pini intrare/ieșire digitali: 14 (dintre care 6 pot oferi ieșire PWM)
- Pini analogici de intrare: 6
- Memorie Flash 32 KB
- SRAM 2 KB
- EEPROM 1 KB
- Frecvență de lucru: 16 MHz

3.2.2.3 Arduino Mega

Arduino Mega este un tip de Microcontroller ce se bazează pe ATmega2560. Este format din 54 de pini de intrare / ieșire digitali. Din totalul de pini sunt folosiți 14 pini pentru ieșirea PWM, 16 pini pentru intrările analogice, 4 pini sunt folosiți pentru portul serial hardware al UART. Există pini precum oscilatorul de cristal de 16 MHz, Conexiune USB, RESET pin, Antet ICSP, și o mufă de putere.



Figură 3.5 Arduino Mega [20]

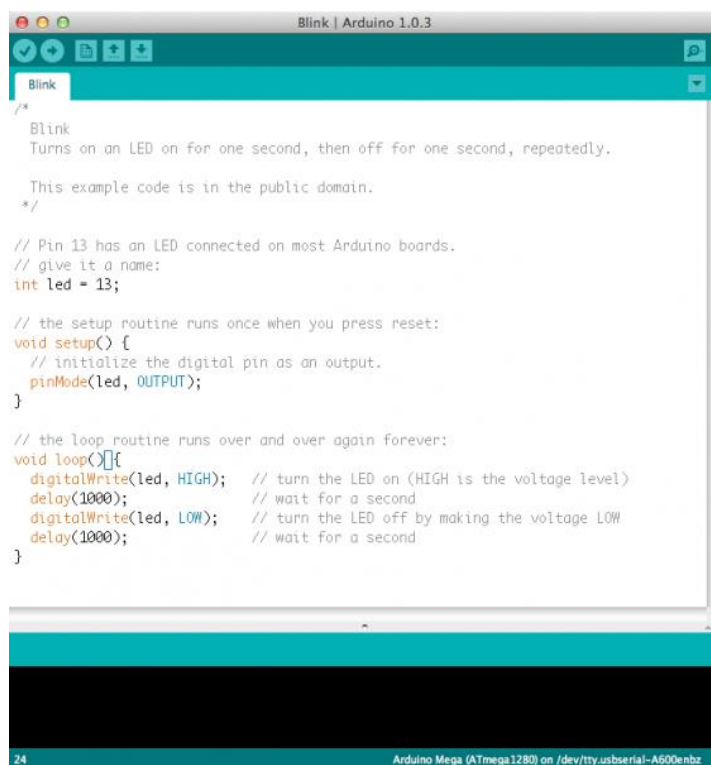
Acest Arduino Mega are, de asemenea, pini SDA și SCL, care se află lângă AREF. Există doi pini noi în apropierea pinului RESET, care sunt IOREF, ce permit scuturilor să se adapteze tensiunii furnizate de placă. Celălalt este neconectat și este rezervat în utilizări ulterioare.

Caracteristici:

- Microcontroller ATmega2560
- Tensiune de alimentare: 5V
- Tensiune de intrare recomandată: 7-12V
- Pini Digital Input / Output: 54 (din care 14 oferă ieșire PWM)
- Pini de intrare analogici: 16
- Curent DC pe Input / Output: 40 mA
- Curent DC pentru Pinul 3.3V: 50 mA
- Flash Memory: 256 KB
- Frecventa: 16MHz

3.2.2.4 Arduino IDE

Software-ul Arduino open-source (IDE) facilitează scrierea codului și încărcarea acestuia pe placă. Acest software poate fi utilizat cu orice placă Arduino.



Figură 3.6 Arduino IDE

3.2.2.5 Alte componente folosite pentru circuitele Arduino realizate in acest proiect

Fingerprint Sensor:

- Supply voltage: 3.6 - 6.0VDC
- Operating current: 120mA max
- Peak current: 150mA max
- Fingerprint imaging time: <1.0 seconds
- Window area: 14mm x 18mm
- Signature file: 256 bytes
- Template file: 512 bytes
- Storage capacity: 162 templates
- Safety ratings (1-5 low to high safety)
- False Acceptance Rate: <0.001% (Security level 3)
- False Reject Rate: <1.0% (Security level 3)
- Interface: TTL Serial
- Baud rate: 9600, 19200, 28800, 38400, 57600 (default is 57600)
- Working temperature rating: -20C to +50C
- Working humidity: 40%-85% RH
- Full Dimensions: 56 x 20 x 21.5mm
- Exposed Dimensions (when placed in box): 21mm x 21mm x 21mm triangular
- Weight: 20 grams



Figură 3.7 Fingerprint sensor

RFID Sensor:

Caracteristici tehnice:

- Curent: 13-26mA / DC 3.3V
- Curent inactiv: 10-13mA / DC 3.3V
- Consum in stand-by: <80uA
- Viteza maximă: <30mA
- Frecvență de operare: 13.56MHz
- Tipuri de carduri acceptate: S50, 570, UltraLight, Pro, Desfire
- Temperatura mediului de operare: -20-80 grade Celsius
- Mediu Temperatura de depozitare: -40-85 grade Celsius
- Umiditate relativă: 5% -95%
- Rata de transfer a datelor: Max. 10Mbit / s
- Caracteristici fizice ale produsului: Dimensiuni: 40mm x 60mm

Conținut pachet:

- 1 x Modul RFID RC522
- 1 x Key Tag 13.56 Mhz
- 1 x Card RFID 13.56Mhz



Figură 3.8 RFID Sensor

Keypad:

Caracteristici tehnice:

- Timp de revenire 1 (ms)
- Timp de funcționare: 100 de milioane (ori)
- Temperatura de operare: -40°C ~ +80°C
- Buclă de șoc electric: <5ms
- Puterea dielectrică: 250VRms (50 ~ 60Hz 1min)
- Dimensiune: 70mm x 77mm x 1mm
- Greutate: 7.5 g



Figură 3.9 Keypad

Motor Servo MG90S 180G:

Caracteristici tehnice:

- Greutate: 13.4g
- Dimensiune: 22.5 x 12 x 35.5 mm
- Viteza de operare: 0.1 s/60 grade (4.8V) și 0.08s/60 grade (6V)
- Alimentare: 4.8V - 6V
- Lățimea benzii: 5 μs
- Temperatură de funcționare: 0°C - 55°C

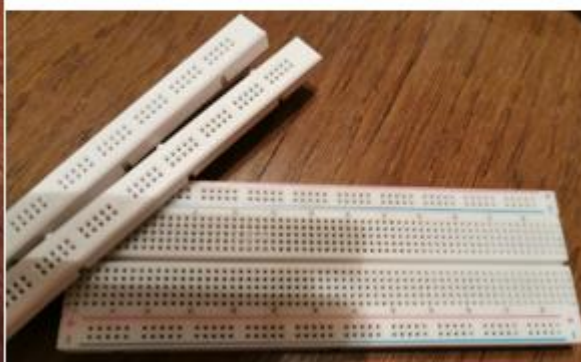


Figură 3.10 Motor Servo MG90S 180G

Fire:



Breadboard-uri:



Figură 3.11 Fire și Breadboard-uri

Breadboard-ul este folosit în general pentru realizarea rapidă a montajelor fără a fi nevoie de lipirea firelor, pentru testarea proiectelor. Piese se pot conecta prin fire de tip tată-tată, mamă-mamă sau direct în găurile din placa breadboard. Fiecare pin se poate conecta prin cele 4 găuri așezate perpendicular pe circuit. Două magistrale amplasate în lateral, se folosesc în mod normal pentru alimentare, placa având în total, 4 alimentări independente.

Rezistente:

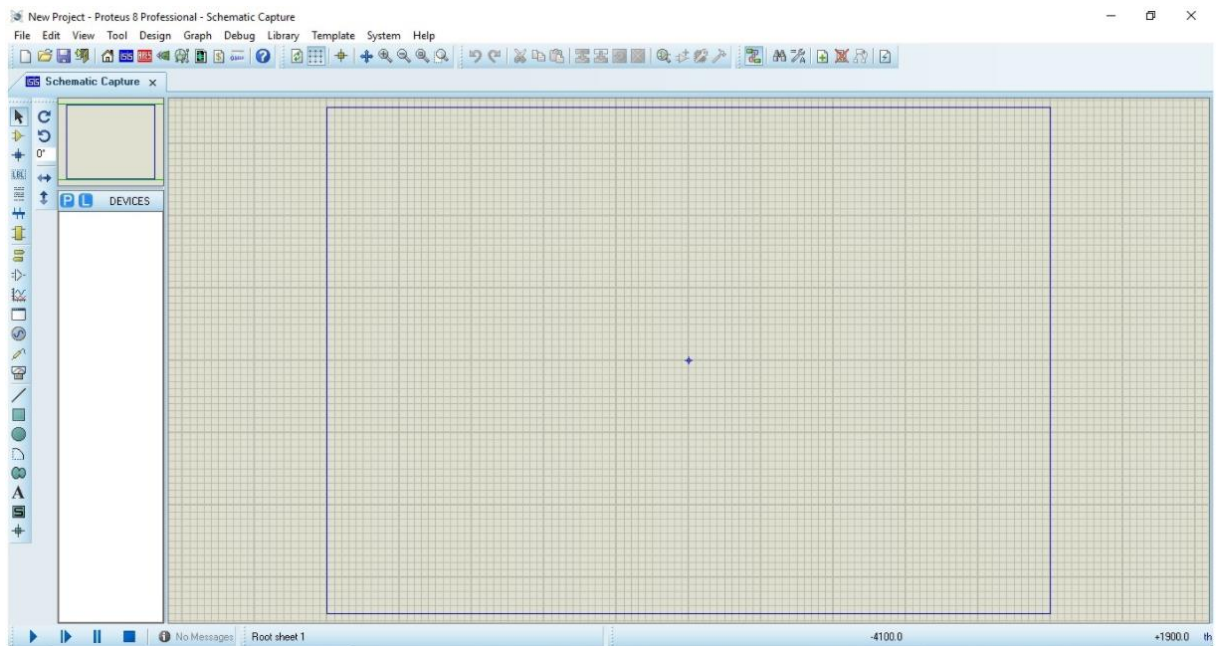


Figură 3.12 Led-uri și Rezistente

3.2.3 Proteus

În 1988, prima versiune a softului Proteus, cunoscută sub numele de PCB-B, a fost creată de John Jameson, care a fost președintele companiei. Proteus este utilizat pentru simularea, proiectarea și desenarea circuitelor electronice. A fost inventat de Electronic Labcenter. Folosind proteus se pot realiza modele de circuite bidimensionale. Cu ajutorul acestui software de inginerie, se poate construi și simula diferite circuite electrice și electronice pe computerele sau laptopurile personale.

Există numeroase avantaje pentru a simula circuitele pe proteus înainte de a le face practic. Proiectarea circuitelor pe proteus durează mai puțin timp decât construcția practică a circuitului. Posibilitatea de eroare este mai mică în simularea software, cum ar fi conexiunea liberă, care necesită mult timp pentru a afla problemele de conexiune într-un circuit practic. Nu există posibilitatea zero de a arde și deteriora orice componentă electronică din proteus. Instrumentele electronice care sunt foarte scumpe pot intra cu ușurință în proteus, cum ar fi un osciloscop. Folosind proteus puteți găsi diferite părinți de circuite precum curentul, o valoare a tensiunii oricărei componente și rezistență în orice moment, care este foarte dificil într-un circuit practic.



Figură 3.13 Proteus IDE

Există 2 părți principale ale proteus-ului, mai întâi, este utilizat pentru proiectarea și desenarea diferitelor circuite, iar a doua este pentru proiectarea machetei PCB. Prima parte este ISIS care a proiectat și simulat circuitele. A doua parte, ARES, este utilizată pentru proiectarea unei plăci imprimate de circuit. De asemenea, oferă caracteristici legate de vizualizarea tridimensională a designului în PCB.

Editarea ferestrei

Aceasta este o porțiune de desen a proteus-ului în care se simulează circuitele și proiectele de inginerie.

Prezentare generală

În fereastra de ansamblu, se vede vizualizarea completă a designului complet.

Selector de obiecte

Această secțiune are 2 butoane P și E. P este utilizată pentru a selecta diferite componente și este afișată în această casetă. Butonul E este pentru a edita.

Opțiunea zoom

Utilizând aceasta opțiune se poate mări și micșora cu ușurință aspectul ecranului și ajutând la observarea completă a sistemului.

Opțiunea tool

Folosind această opțiune, se pot selecta diferite dispozitive precum voltmetru, ammetru, osciloscop etc.

Butoane de rulare

În partea stânga jos sunt 4 butoane: rulați, opriți, întrerupeți și opriți. Aceste butoane sunt ca controlul de remorcare, pornirea și oprirea circuitului.

3.2.4 Tensorflow

3.2.4.1 Object Detection (coco-ssd)

Coco-ssd este un model de detectare a obiectelor care își propune să localizeze și să identifice mai multe obiecte într-o singură imagine. Acest model detectează obiecte definite în setul de date COCO, care este un set de date de detectare, segmentare și legendare a obiectelor pe scară largă. Modelul este capabil să detecteze 80 de clase de obiecte. (SSD înseamnă detecție cu mai multe fotografii cu o singură fotografie). Poate lua intrare ca orice elemente de imagine bazate pe browser (, <video>, <canvas> elemente, de exemplu) și returnează o serie de căsuțe de delimitare cu nume de clasă și nivel de încredere.

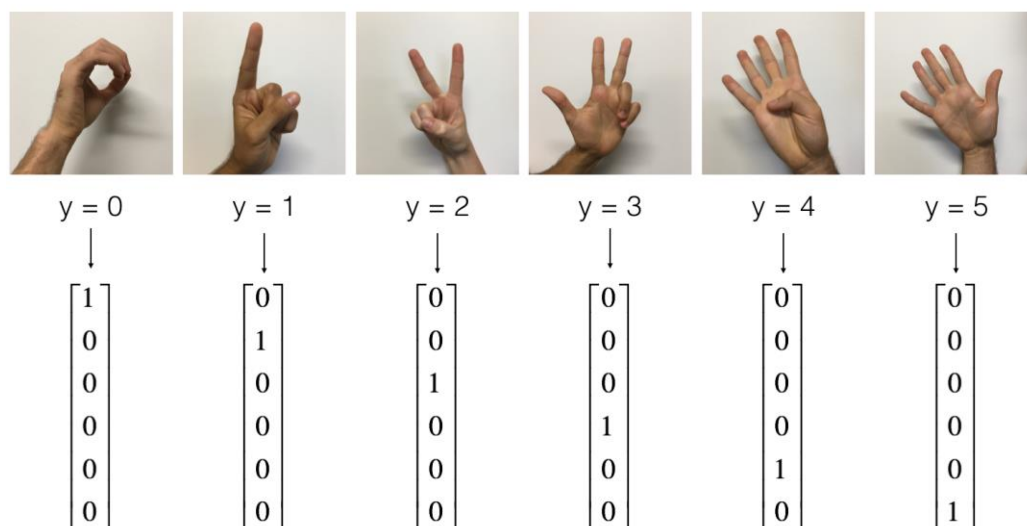
Există două moduri principale de a obține acest model în proiecte de tip JavaScript: prin etichete script sau prin instalarea acestuia de la NPM și folosind un instrument de compilare precum Parcel, WebPack sau Rollup.



Figură 3.14 Object Detection [21]

3.2.4.2 MediaPipe Handpose

MediaPipe Handpose este o conductă ușoară ML formată din două modele: un detector de palme și un model de urmărire a degetelor scheletului manual. Prezice 21 de puncte cheie 3D pe mână detectată. Având în vedere o intrare, modelul prezice dacă conține o mână. Dacă da, modelul returnează coordonatele pentru caseta de delimitare din jurul mâinii, precum și 21 de puncte cheie în interiorul mâinii, care conturează locația fiecărei articulații a degetelor și a palmei. MediaPipe Handpose are aproximativ 12 MB de weights și este potrivit pentru inferența în timp real pe o varietate de dispozitive (40 FPS pe un MacBook Pro 2018, 35 FPS pe un iPhone11, 6 FPS pe un Pixel3).



Figură 3.15 MediaPipe Handpose [22]

3.2.4.3 Tensorflow.js

Conține logica și scripturile care combină mai multe pachete.

API-uri:

- TensorFlow.js Core, API flexibil la nivel scăzut pentru rețele neuronale și calcul numeric.
- TensorFlow.js Layers, API la nivel înalt care implementează funcționalități similare cu Keras.
- TensorFlow.js Data, API simplu pentru încărcarea și pregătirea datelor analog cu tf.data.
- TensorFlow.js Converter, instrumente pentru importul unui TensorFlow SavedModel în TensorFlow.js
- TensorFlow.js Vis, vizualizare în browser pentru modelele TensorFlow.js
- TensorFlow.js AutoML, Set de API-uri pentru încărcarea și rularea modelelor produse de AutoML Edge.

Backends / Platforme:

- Backend CPU TensorFlow.js, backend pur JS pentru Node.js și browser.
- TensorFlow.js WebGL Backend, back-back WebGL pentru browser.
- TensorFlow.js WASM Backend, backend WebAssembly pentru browser.
- TensorFlow.js WebGPU, backGPU WebGPU pentru browser.

- TensorFlow.js Node, platforma Node.js prin adaptorul TensorFlow C ++.
- TensorFlow.js React Native, React Native platform prin adaptor expo-gl.

3.2.4.4 *Fingerpose*

Clasificatorul de poze cu degetul pentru reperele de mână detectate de modelul de mână al lui TensorFlow.js. Poate detecta gesturi ale mâinii precum „Victorie” , „Thumbs Up” în interiorul unei imagini sursă a camerei web. Se pot defini gesturi suplimentare de mână folosind descrieri de gesturi.

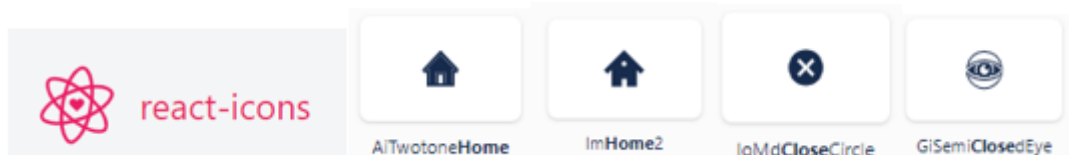
Cum funcționează? Detectarea gesturilor funcționează în trei etape:

- Detectează reperele mâinii din interiorul imaginii video
- Estimează direcția și ondularea fiecărui deget individual
- Compară rezultatul cu un set de descrieri de gesturi

Pasul (1) este realizat de „handpose” Tensorflow, Pasul (2) și (3) sunt gestionate de această bibliotecă.

3.2.5 **React Icons**

Include pictograme populare pentru a fi integrate în proiecte React cu ușurință cu pictograme de reacție, care utilizează importurile ES6 ce permite să includerea doar pictogramelor pe care le utilizează proiectul. Câteva dintre pictogramele folosite în proiect au fost integrate cu rezultatele algoritmilor din Tensorflow (de exemplu pentru a avertiza că ceva e în neregulă pe o cameră video selectată, algoritmul generând o pictogramă de eroare sau atenție), restul fiind utilizate în alte parti ale aplicației.



Figură 3.16 React Icons

3.2.6 **React Webcam**

Componenta folosită pentru utilizarea camerei web, în interfață client. Ca și metode, se poate folosi: getScreenshot – care returnează un string encodat în base64 reprezentând imaginea curentă de pe webcam. Este posibilă capturarea videoclipurilor cu <Webcam /> folosind API-ul de înregistrare MediaStream. Aceasta a fost folosită pentru a oferi o sursă video pe care algoritmii din Tensorflow să o poată analiza și oferi rezultate ale analizei video.

Conține următoarele proprietăți:

prop	type	default	notes
audio	boolean	true	Enable/disable audio

audioConstraints	object		MediaStreamConstraint(s) for the audio
forceScreenshotSourceSize	boolean	false	Uses size of underlying source video stream (and thus ignores other size related props)
imageSmoothing	boolean	true	Pixel smoothing of the screenshot taken
mirrored	boolean	false	Show camera preview and get the screenshot mirrored
minScreenshotHeight	number		Min height of screenshot
minScreenshotWeight	number		Min weight of screenshot
onUserMedia	function	noop	Callback for when component receives media stream
onUserMediaError	function	noop	Callback for when component can't receive a media stream with MediaStreamError param
screenshotFormat	string	Image /webp	Format of screenshot
screenshotQuality	number	0.92	Quality of screenshot (0 to 1)
videoConstraints	object		MediaStreamConstraint(s) for the video

Tabel 3.2 Proprietăți React Webcam

3.2.7 Axios

Axios este o bibliotecă HTTP bazată pe promises cu care se pot trimite request-uri între browser și node.js. Axios oferă o bibliotecă simplă de utilizat într-un pachet mic, cu o interfață foarte extensibilă. Această bibliotecă a fost folosită în aplicația client (React) pentru a comunica cu baza de date (MongoDB), adică atunci când a fost nevoie de date din baza de date, s-a trimis un request GET către node.js, care a luat datele din MongoDB, și le-a dat înapoi la client pentru a le afișa. De asemenea, tot cu axios s-au efectuat și operații de editare, utilizând POST requests, adăugare, tot cu POST requests dar link diferit, și ștergere, utilizând DELETE.

4 Proiectarea si implementarea sistemului

4.1 Arhitectura aplicației

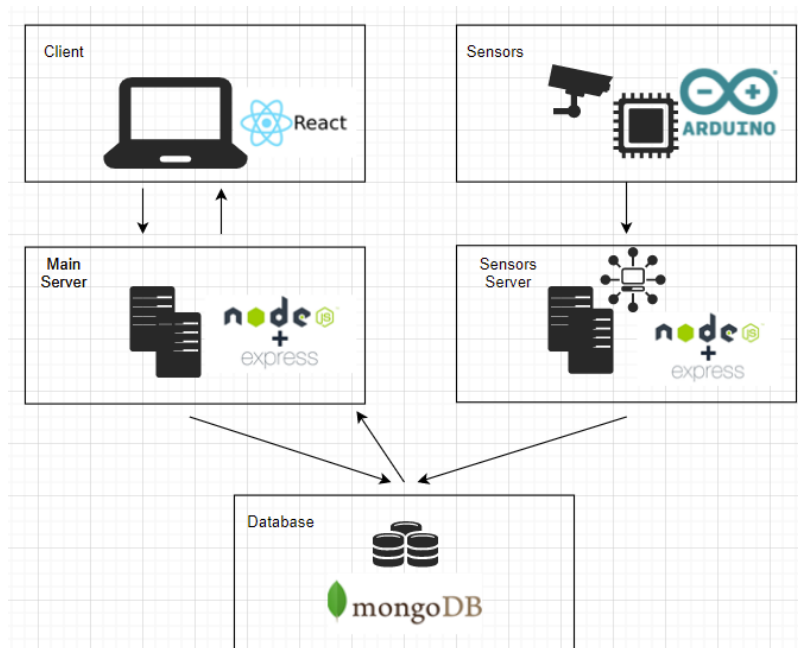
Aceasta lucrare permite utilizatorilor să monitorizeze întregul sistem de securitate într-o clădire. De asemenea unele funcții sunt controlate automat (de exemplu sistemul de recunoaștere video, sau sistemul centralizat de monitorizare al senzorilor de pe fiecare nivel al clădirii). Sistemul de securitate propus constă în următoarele componente principale:

- Modulul ce se ocupă cu înregistrarea activității senzorilor de securitate din clădire
- Modulul de recunoaștere video
- Modulul de monitorizare status senzori

Soluția software propusă reprezintă o aplicație web client-server care permite utilizatorilor să monitorizeze senzorii din clădire și să verifice rezultatele modelelor automate de recunoaștere video în timp real. Interfețele utilizator ce satisfac aceste cerințe sunt:

- Monitorizarea senzorilor de pe fiecare etaj al clădirii, utilizând planuri arhitecturale peste care s-au suprapus pictogramele senzorilor, senzorii fiind elemente interactive care își schimbă statusul automat în funcție de ce se întâmplă cu aceștia.
- Monitorizarea materialului produs de fiecare cameră video din clădire, alegerea algoritmului de detecție în timp real.

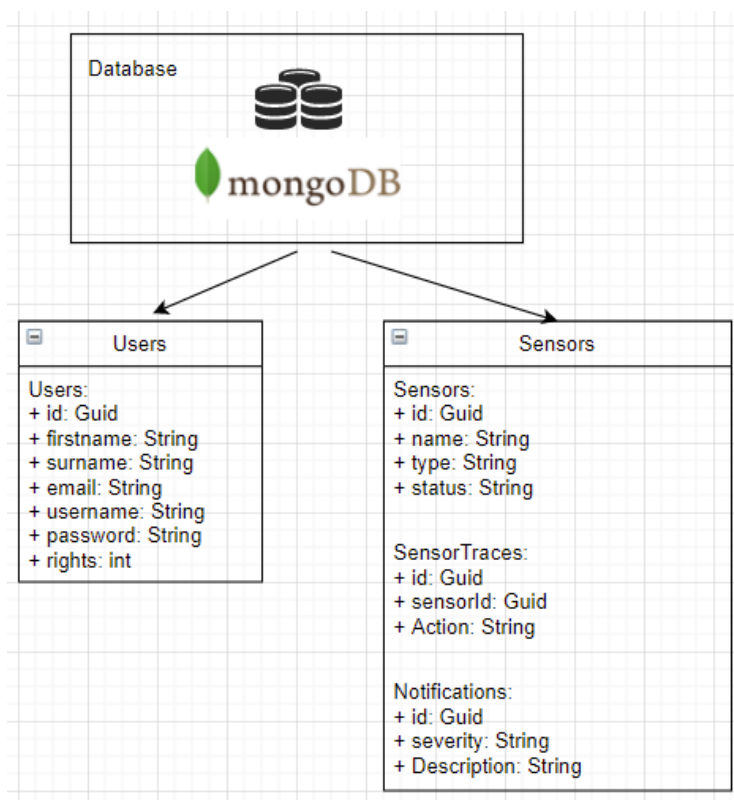
În figura de mai jos este afișată schema ce demonstrează legăturile de comunicație dintre modulele prezentate anterior:



Figură 4.1 Arhitectura aplicației

Acțiunile înregistrate de senzori sunt citite de Sensors Server, care le înregistrează mai departe în Baza de Date. Aplicația web constă în Client și Main Server, care comunică între ele, stocând, editând și citind date din Baza de Date.

Figura de mai jos arată schema bazei de date. S-a folosit MongoDB, utilizând două baze de date: Users și Sensors. Users conține tabelul Users, iar Sensors conține 3 tabele: Sensors, SensorTraces și Notifications. Sensors conține lista tuturor senzorilor înregistrați în clădire (id, name, type, status). SensorTraces înregistrează toate acțiunile realizate pe senzori (id, sensorId, action). De exemplu când cineva încearcă să se logheze cu un tag, iar acesta este invalid, sau când se loghează cu un tag, iar acesta este valid. Aceste date sunt esențiale în forensics dacă se ajunge până acolo, pentru că se poate analiza cine a intrat sau sub ce identitate s-a camuflat atacatorul. Tabelul de Notifications înregistrează warning-urile sau erorile ce pot fi înregistrate de senzori (id, severity, description). Acestea sunt afișate în interfață client la notifications icon.



Figură 4.2 Structura baze de date

4.2 Cazuri de utilizare

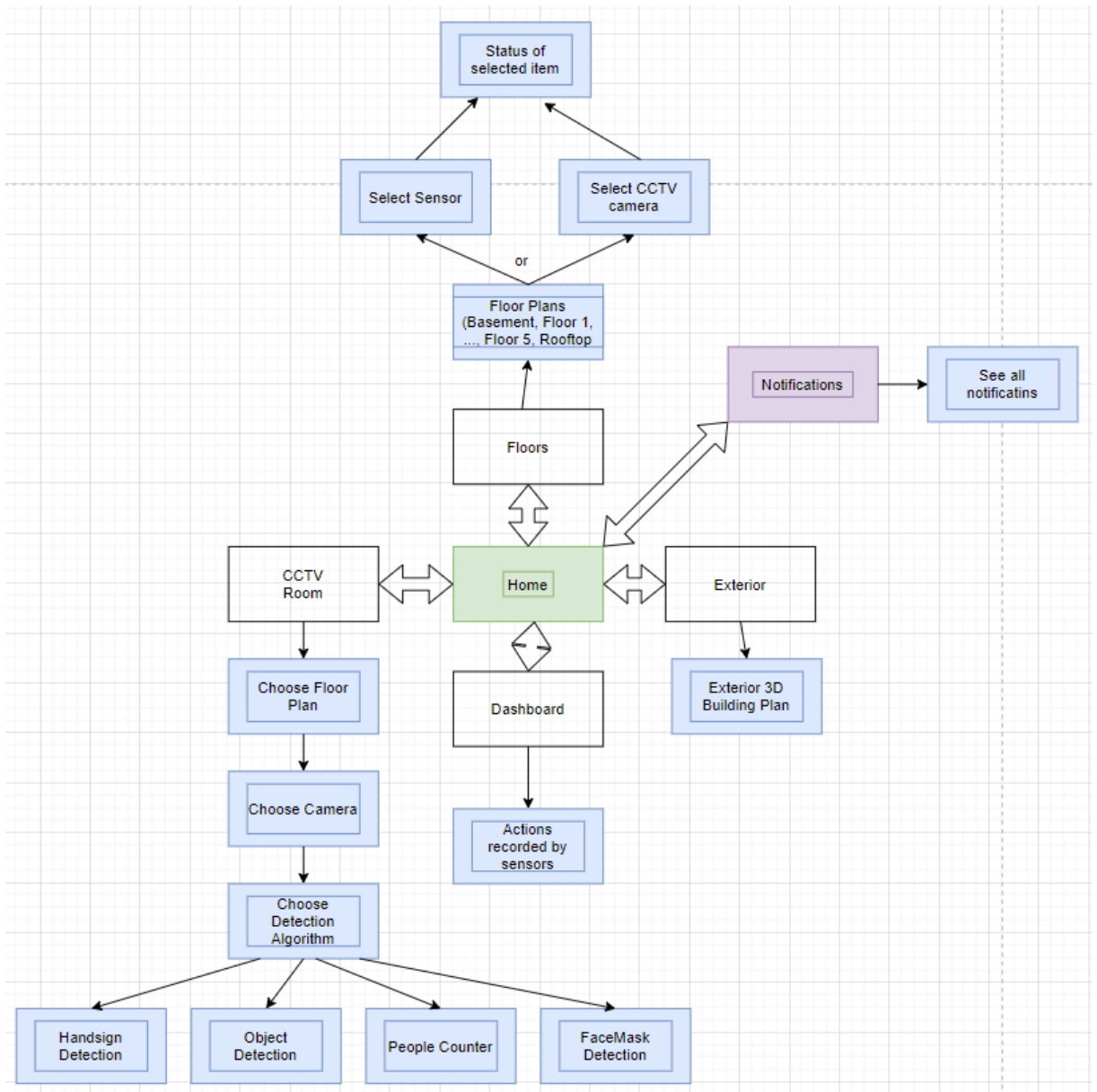
Userul se loghează în aplicație și ajunge la ecranul de Home (care automat merge pe Dashboard. Pe Dashboard se pot vizualiza metrice legate de stocarea datelor. Utilizand sidebar-ul din stânga, acesta poate naviga între Dashboard, Exterior, Basement, Floors, Roof și CCTV Room. Pagina de Exterior conține planul 3D al clădirii. Acesta se poate monitoriza în cazul în care apar probleme în clădire, pentru a găsi soluții cât mai eficiente de evacuare sau evita potențiale atacuri din surse externe.

Basement, Floors și Rooftop sunt pagini ce conțin planurile arhitecturale ale fiecărui etaj, împreună cu senzorii amplasați în clădire. Senzorii sunt elemente interactive ce își schimbă culoarea și statusul automat în funcție de ce se întâmplă cu aceștia. De exemplu, în această implementare, senzorii sunt categorizați în două mari clase: camere video de supraveghere și sisteme de blocare/deblocare uși. Fiecare clasă are o pictogramă diferită. Mergând cu mouse-ul pe iconiță, apare un tooltip ce arată numele sensorului. Dând click pe iconiță, apare un widget în dreapta ecranului ce oferă detalii complete legate de sensorul ales (nume, tip, status). În funcție de statusul sensorului, culoarea pictogramei se poate schimba automat (roșu -> problemă cu sensorul, verde -> sensorul funcționează corect, gri -> sensorul este inactiv din diverse motive, cum ar fi control de rutină, instalarea unui patch de update, scoaterea din funcțiune).

CCTV Room este pagina pe care utilizatorul poate intra pentru a monitoriza ce se vede pe camerele video de supraveghere din interiorul clădirii. În mijlocul ecranului este sursa video, iar în dreapta se poate alege etajul iar apoi camera de pe care se dorește să se facă monitorizarea, aflată pe etajul ales. De asemenea, au fost introduse diverse modele de recunoaștere automată, în timp real, a diverselor situații ce se pot întâmpla în clădire. Astfel, din panoul din dreapta se poate alege și algoritmul pe care ar vrea să se facă monitorizarea (lista curentă de modele este: hand sign recognition, object recognition, people counter, face mask recognition).

Modelul de HandSign Recognition poate fi ales, spre exemplu, în situația în care programul este instalat pentru spații comerciale, cum sunt magazinele. Adeseori se întâmplă furturi în care vânzătorul e amenințat cu arma sau cu orice altceva, acesta fiind nevoit să îi ofere bani sau alte bunuri din magazin atacatorului. În cazul în care vânzătorul nu are acces la pedala sau butonul de alarmă furt (de obicei în magazine sunt câte o pedală sau un buton pe care vânzătorul poate apăsa în caz de amenințare, acesta trimițând un semnal la dispecerat și o echipă de securitate este trimisă la fața locului, după ce contract are magazinul cu firma de securitate), camera de supraveghere de la casă poate recunoaște atacul și simula același tip de avertizare, ca acționarea unei pedale sau buton de alarmă. Folosind acest model, în timp ce vânzătorul pregătește bunurile (bani sau alte obiecte din magazin), poate ține mâna într-un mod prestabilit (ales de dinainte și cunoscut doar de vânzător și echipa care a instalat programul) pentru a declanșa alarma, fără ca atacatorul să știe că omenii legii sunt pe drum (atacatorul s-ar aștepta să fie apăsător un buton). Acest model a fost format din 2 sub-modele: finger position și hand recognition. Prima dată recunoaște mâna, apoi poziția fiecărui deget.

Modelul Object Recognition, cum îi spune și numele, este folosit pentru a recunoaște obiectele de pe înregistrările video, în timp real. Modelul este folosit în această aplicație pentru mai multe acțiuni. În primul rând, a fost modificat pentru a recunoaște persoanele. După aceea, programul numără câte persoane sunt în încăpere. Această funcționalitate e folosită pentru a limita numărul de persoane dintr-o încăpere. În al doilea rând, dacă sistemul detectează obiecte amenințătoare (de exemplu cuțit sau pistol), se trimite automat o alarmă sistemului. Aceste sisteme pot fi alese din fereastra din dreapta de pe pagina CCTV Room.



Figură 4.3 Diagrama secventiala

4.3 Implementare

4.3.1 Aplicația de MERN Stack

Mern Stack conține următoarele tehnologii:

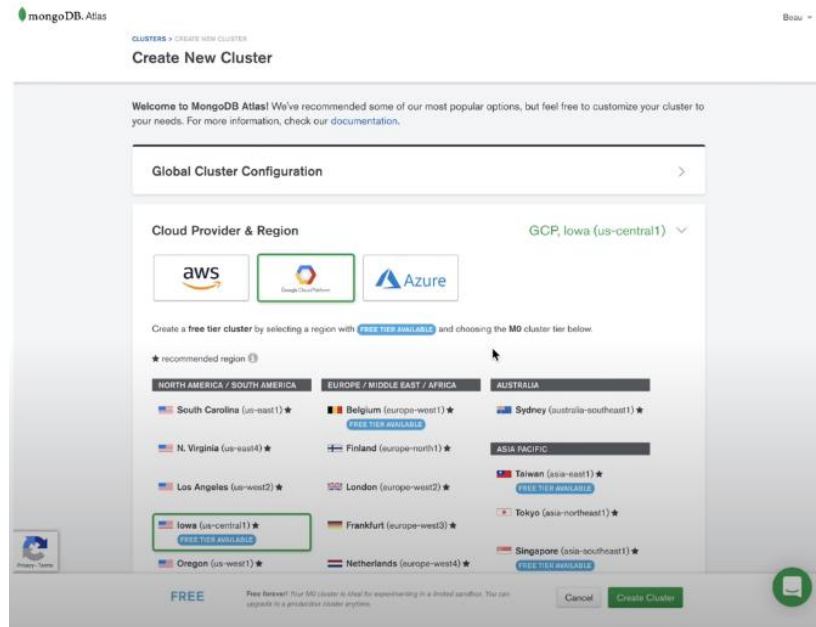
- MongoDB: o bază de date open source bazată pe documente
- Express: un cadru de aplicații web pentru Node.js
- React: o bibliotecă de front-end javascript pentru construirea interfeței cu utilizatorul
- Node.js: mediu de rulare javascript care execută cod javascript în afara unui browser (cum ar fi un server)

- Mongoose: soluție simplă, bazată pe scheme, pentru modelarea datelor aplicației

4.3.1.1 Configurarea MongoDB Atlas

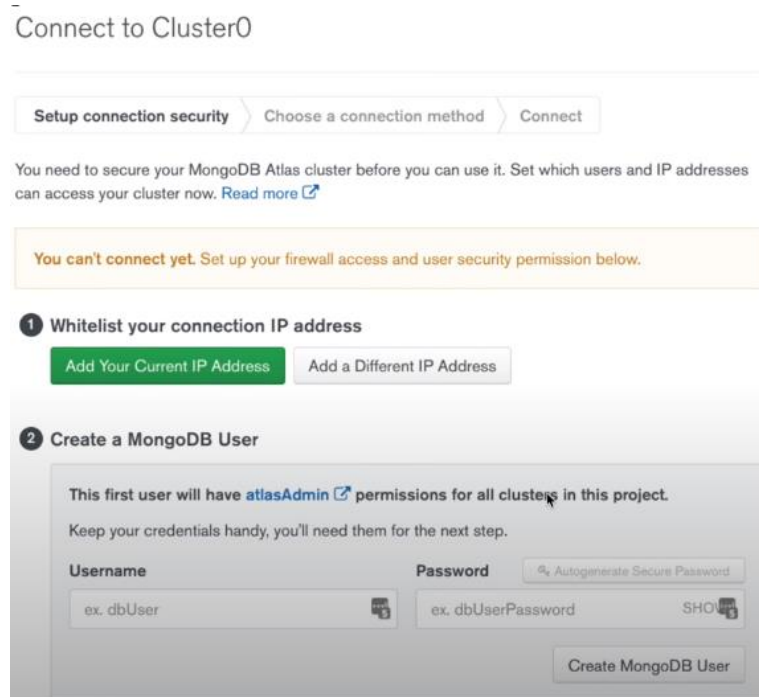
Se creează un cont (dacă nu este unul existent deja) pe mongodb.com/cloud/atlas

Se creează un Proiect nou și apoi se construiește un Cluster:



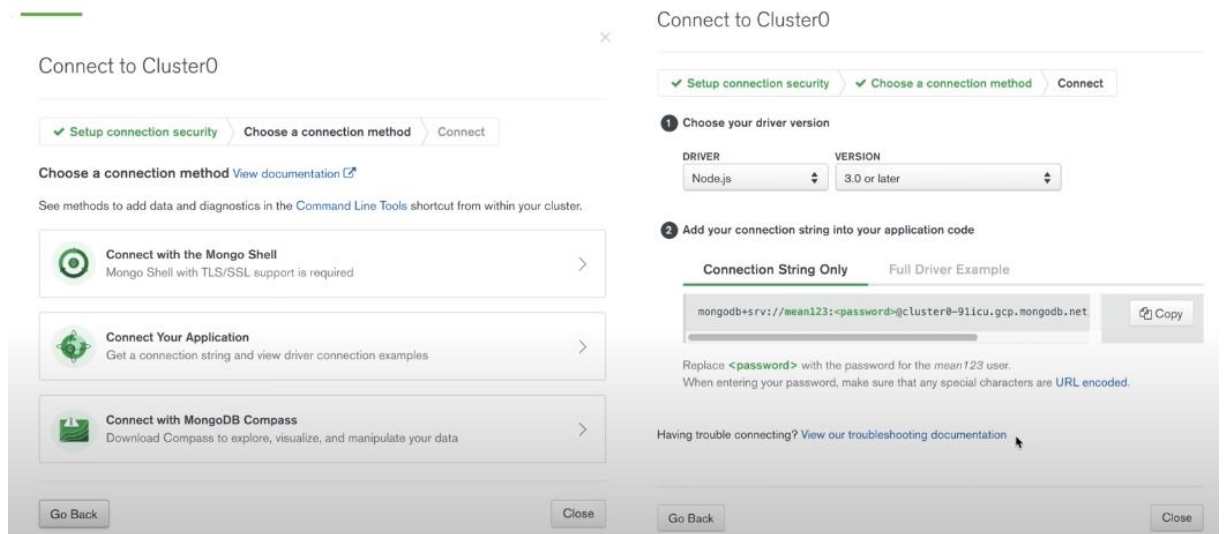
Figură 4.4 MongoDB – Crearea cluster-ului

Apoi se fac câteva setări de securitate pentru cluster-ul creat. Se setează IP-urile care au acces să facă modificări la acest Cluster. E foarte important ca accesul la cloud să fie cât mai limitat pentru a evita eventuale breșe de securitate.



Figură 4.5 MongoDB – Setarea conexiunii de securitate

După acești pași, configurarea MongoDB e finalizată și se poate lua connection string-ul ce va fi folosit în aplicație pentru a comunica cu baza de date.



Figură 4.6 MongoDB – Obținerea string-ului de conectare

Aplicația va conține 2 baze de date: Users si Sensors.

Baza de date de Users va avea un tabel numit User. Fiecare User va avea: id (foreign key), firstname, surname, email, username, password și rights. De asemenea se va stoca data creării contului și data ultimei acțiuni de update, din motive de securitate (de exemplu pentru analiza de forensic în cazul în care e sesizată o activitate suspectă a unui user).

Baza de date de Sensors va avea un tabel numit Sensor ce conține: name, type, status, și de asemenea data adăugării senzorului și ulimului update legat de informațiile acestuia. De asemenea, va mai exista un tabel numit SensorTrace, ce va conține activitățile înregistrare de către senzori. Acest tabel va conține: id, sensorId, action.

4.3.1.2 Configurarea Aplicației Web

S-a folosit Visual Studio Code ca și IDE. Prima dată s-a verificat dacă e deja node instalat (dacă nu, trebuie instalat):

```
PS C:\Users\vpopa\Desktop\BachelorsDegree> node -v
v12.22.1
```

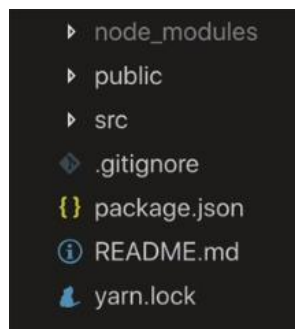
Figură 4.7 Verificarea versiunii de node.js

Apoi s-au creat două directoare: backend și frontend. Directorul de backend va conține tot codul sursă ce ține de partea de server (asta include conexiunea la baza de date MongoDB, modelele tabelor bazei de date, routes pentru fiecare pagină sau funcționalitate ce va avea request-urile de GET, POST, UPDATE, DELETE, și altă logică ce ține de partea de server), de asemenea se va face un fișier care va comunica cu senzorii și baza de date pentru activitățile înregistrate de senzori). Directorul de frontend va conține toată logica de care e nevoie pentru a afișa utilizatorului datele de care acesta are nevoie, precum codul html și css responsive pentru diferite tipuri de ecran, elementele din pagină, planurile arhitecturale ale clădirii, și așa mai departe).

Pentru a creea aplicația de frontend, s-a rulat următoarea comandă: `npx create-react-app frontend`. De unde a rezultat punctul de start al proiectului pe partea frontend, cum se vede și în imaginea de mai jos. Fișierul `package.json` conține comenzile ce pot fi executate asupra fișierului, punctul de start al aplicației, versiunile bibliotecilor utilizate, etc. Fișierul de `.gitignore` este folosit pentru optimizarea transferului de fișiere între proiectul local și cel din repository (în acest fișier sunt declarate fișierele / folderele ce nu vor fi încărcate pe git – de obicei sunt fișiere generate de IDE sau de alte procese, ce nu conțin codul sursă al proiectului, și care pot fi instalate printr-o comandă, sau se auto-generează la build-ul proiectului când este salvat local pe un alt calculator).

Directorul `src` conține codul sursă al proiectului. Aici vor fi adăugate fișierele ce țin de schema bazei de date, fișierele ce țin de rutele care duc la fiecare funcționalitate a aplicației, fișierul `server`, și tot ce ține de partea de stilizare a frontend-ului, cum sunt fișiere de tip CSS. De asemenea vor fi prezente și componentele; fiecare funcționalitate din aplicație cum ar fi butoane, field-uri, senzori sau camere video, vor fi folosite ca și componente ce pot fi reutilizate de-a lungul aplicației.

Fișierul `node_modules` este unul dintre fișierele ce nu vor fi încărcate pe Git. Acesta se poate genera la o instalare nouă, rulând comanda `npm install`. Comanda va lua toate bibliotecile din `package.json` și le va instala automat. Astfel, directorul `node_modules` va fi creat prin rularea acestei comenzi la fiecare nouă instalare pe un dispozitiv nou.



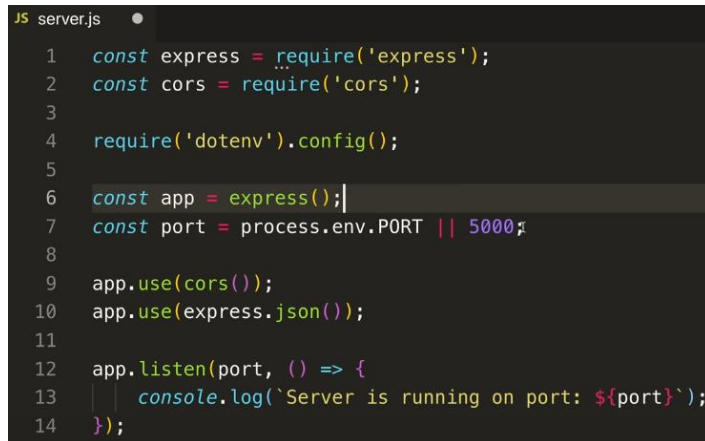
Figură 4.8 Pachetul default de frontend

4.3.1.3 Crearea Backend-ului

Pentru a creea aplicația de backend, s-a rulat următoarea comandă: `npm init` pentru a inițializa un fișier `package.json`. Acest fișier conține toate versiunile dependențelor aplicației, printre multe altele (adică dacă aplicația folosește anumite biblioteci, acestea vor fi trecute în acest fișier împreună cu versiunile lor). În continuare, s-a rulat comanda `npm install express cors mongoose dotenv nodemon` pentru a adăuga în fișierul `package.json` câteva cadre cu care se va face dezvoltarea pe partea de backend. Express este un cadru pentru Node.js. Cors (Cross Origin Resource Sharing) permite request-urilor de tip Ajax să acceseze resurse din surse remote; cu acest pachet se poate accesa ceva din afara serverului, utilizând serverul aplicației. Pachetul `mongoose` este folosit pentru a avea acces la pachetul MongoDB (conține metode de conectare la baza de date, accesare a elementelor de tabel, etc.). `Dotenv` este un pachet folosit în proiectele în care e nevoie de un fișier `.env` care deține string-uri de configurare. Pachetul `nodemon` face posibilă dezvoltarea aplicației și verificarea schimbărilor în timp real. Adică dacă se face o schimbare într-un fișier, la salvarea acestuia schimbările vor fi vizibile automat din

aplicație (nu e nevoie de reload la pagină sau repornită aplicația pentru a se pune schimbările).

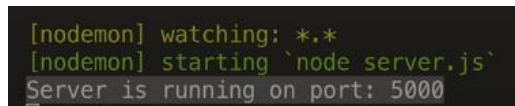
Următorul pas este crearea server-ului. În directorul backend se va crea un fișier numit server.js:



```
1  const express = require('express');
2  const cors = require('cors');
3
4  require('dotenv').config();
5
6  const app = express();
7  const port = process.env.PORT || 5000;
8
9  app.use(cors());
10 app.use(express.json());
11
12 app.listen(port, () => {
13   console.log(`Server is running on port: ${port}`);
14 });
```

Figură 4.9 Fisierul server.js initial

Primele doua linii salvează pachetele de care e nevoie în server ca și constante. Linia patru setează existența unui fișier de tip .env ca și configuration file. Următoarele două linii crează server-ul express și portul la care poate acesta să fie accesat. Se poate observa la linia șapte că în cazul în care nu este specificat un port în fișierul .env, acesta va fi cel default de 5000. Linia 9 specifică faptul că se va folosi cors middleware în aplicația express, în timp ce linia 10 specifică utilizarea formatului json în server, pentru a putea lucra cu obiecte json. Apoi serverul este pornit (acesta începe să asculte pe un anumit port) pe linia 12, afișând și un mesaj în loguri cum că serverul a pornit cu succes. Pentru a verifica, se poate rula *nodemon server*:



```
[nodemon] watching: *.*
[nodemon] starting `node server.js`
Server is running on port: 5000
```

Figură 4.10 Verificare server- ruleaza cu succes

Conexiunea la baza de date a fost făcută în felul următor: s-a importat mongoose ca și o constantă exportată din pachetul mongoose, și înainte de a porni serverul (`app.listen()`), s-a creat conexiunea: *uri* reprezintă connection string-ul ce a fost obținut la început din MongoDB (când s-a creat cluster-ul, ultimul pas). Acest uri a fost salvat în fișierul ce păstrează datele de configurare (.env) sub numele de *ATLAS_URI*.

Apoi s-a făcut conexiunea cu aceste date de conectare, și s-a logat faptul că database connection a fost inițializată cu succes. Se mai pot observa 2 field-uri în linia 14 (`useNewUrlParser` și `useCreateIndex`); acestea au fost setate că și `true` pentru a evita warning-uri primite în consola datorită pachetului express care a trecut la metode noi, acestea ce vin default în pachet fiind deprecated. Deci au fost marcate că și `true` pentru a folosi noile metode și evita warning-urile.

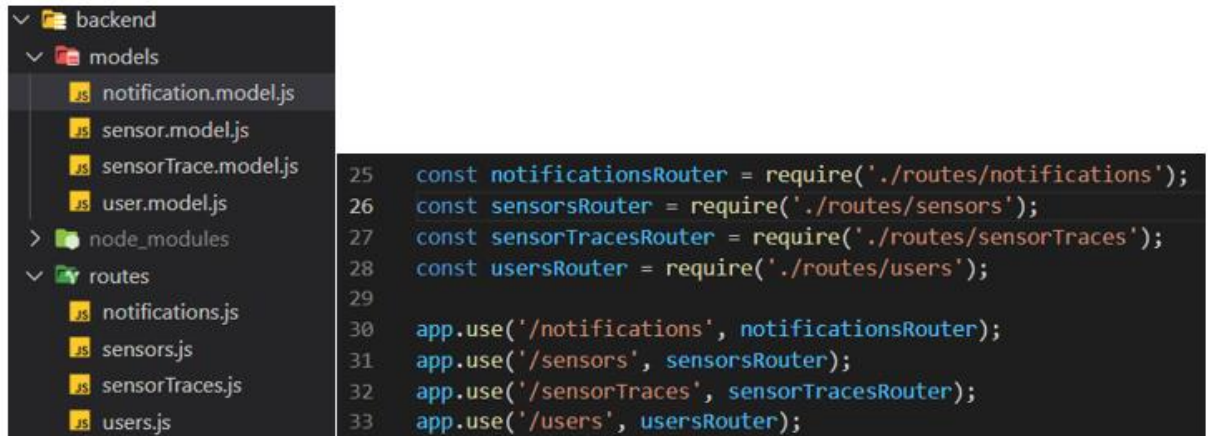
```

13 const uri = process.env.ATLAS_URI;
14 mongoose.connect(uri, { useNewUrlParser: true, useCreateIndex: true }
15 );
16 const connection = mongoose.connection;
17 connection.once('open', () => {
18   console.log("MongoDB database connection established successfully");
19 })

```

Figură 4.11 Conectarea la baza de date

Apoi se vor defini schemele bazei de date: astfel, s-au creat 4 fișiere: notification.model.js, sensor.model.js, sensorTrace.model.js și user.model.js. Primele trei tabele vor salva datele în baza de date: Sensors, ultimul tabel va salva datele în baza de date: Users. S-au creat schemele pentru fiecare tabel și s-a exportat fiecare tabel pentru a putea fi accesat din codul sursă al serverului (din routes). Apoi s-a creat directorul routes, în care se vor face acțiunile request-urilor. Astfel, în interiorul acestui director s-au creat 4 fișiere: notifications.js, sensors.js, sensorTraces.js și users.js. Notifications.js are două rute: GET all notifications și GET specific notification. Sensor.js are 5 rute: Get all sensors, Get specific sensor, ADD specific sensor, UPDATE specific sensor și DELETE specific sensor. Fișierul sensorTrace.js are 3 rute: GET all traces, GET specific trace și ADD specific trace. Fișierul users.js are 5 rute: GET all users, GET specific user, ADD specific user, UPDATE specific user și DELETE specific user. După ce am terminat de creat request-urile, se adaugă middleware-ul acestor rute în fișierul server.js Conform imaginii de mai jos, când se accesează localhost:5000/notifications, se va accesa ruta notifications.js, deci se vor putea face request-uri de către frontend pentru tot ce ține de notifications.



Figură 4.12 Routing

4.3.1.4 Crearea Frontend-ului

În pașii de configurare a proiectului, descriși anterior, s-a creat și template-ul proiectului de frontend (utilizând comanda npm create-react-app frontend). Astfel au rezultat următoarele directoare: node_modules (la fel ca cel descris anterior), public (aici sunt fișierele index.html și logo-uri), și directorul src (acesta conține tot codul sursă). De asemenea se găsesc și aici un fișier package.json ce deține toate bibliotecile împreună cu versiunile lor, ce sunt folosite în acest proiect de frontend. În continuare se va discuta de conținutul src (fișierele App.js, App.css, index.js, index.css).

Fișierul `index.js` randează componenta `App.js` și o trimite către `index.html` din directorul public. Pentru a rula clientul de frontend, se va rula din terminal (aflat la locația directorului frontend) comanda: `npm start`. Aceasta v-a rula aplicația și se va deschide o fereastră nouă de browser afișând:



Figură 4.13 Aplicație frontend React default, pornită cu succes

În continuare, s-au instalat următoarele pachete: `@tensorflow-models/coco-ssd`, `@tensorflow-models/handpose`, `@tensorflow/tfjs`, `fingerpose`, `jquery`, `react-dom`, `react-icons`, `react-router-dom` și `react-webcam`.

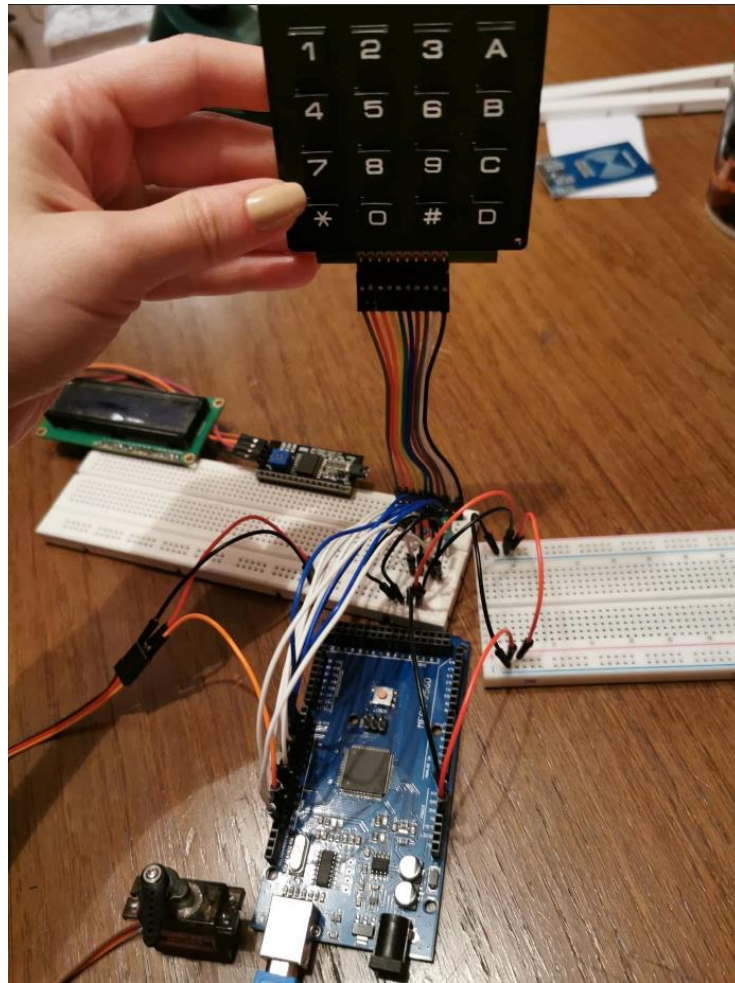
Object Detection Model (sau `coco-ssd`) este un pachet din biblioteca `tensorflow` care va fi folosit pentru detecția obiectelor din încăperi, recunoscute prin camerele video de supraveghere. E nevoie de această detecție pentru a alerta echipa de securitate și sistemul în momentul în care sunt detectate obiecte amenințătoare, precum cuțite sau pistoale. De asemenea, acest model mai poate fi folosit pentru detecția persoanelor și limitarea numărului acestora într-o încăpere.

MediaPipe Handpose Model (`handpose`) este un alt pachet din biblioteca `tensorflow`, ce detectează mâna persoanei. Cu ajutorul acestei biblioteci se va obține detecția semnelor. De exemplu, în cazul unui jaf în magazin sau bancă (în care angajatul nu are acces sau e amenințat să nu apese pedala de alarmă-jaf), angajatul poate să deschidă casa de marcat sau seiful, făcând o poziție a mâinii specială, cunoscută doar de angajat și program. Astfel, se poate declanșa alarma și trimite notificare la dispecer, ce va anunța echipajul de securitate cu care are compania contract. Pentru a realiza acest lucru este nevoie și de pachetul `fingerpose`, cu care se poate defini poziția și orientarea fiecărui deget al mâinii, după ce mâna a fost recunoscută.

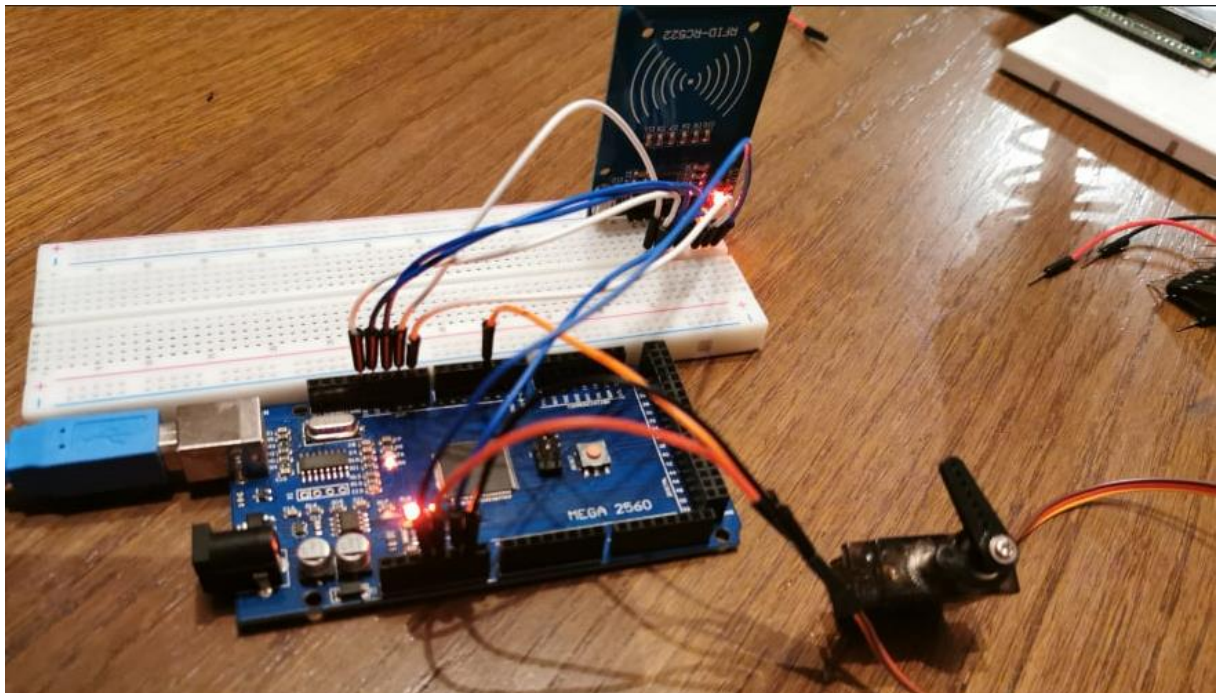
React Webcam va fi folosit pentru a demonstra funcționalitatea modelelor de recunoaștere video. React Router Dom va fi folosit pentru rutarea request-urilor către path-urile corespunzătoare. React Icons va fi folosit la stilizarea aplicației folosind pictograme moderne.

4.3.1.5 Implementare funcționalități pe partea de Hardware

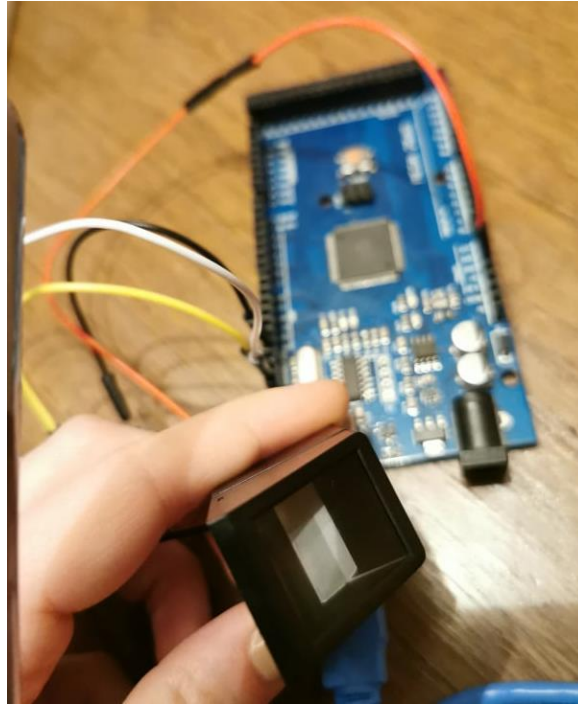
S-au implementat circuitele electrice pentru fiecare tip de senzor de deblocare ușă, folosind Arduino, precum se vede în următoarele imagini:



Figură 4.14 Circuit Door-lock: Keypad



Figură 4.15 Circuit Door-lock: RFID

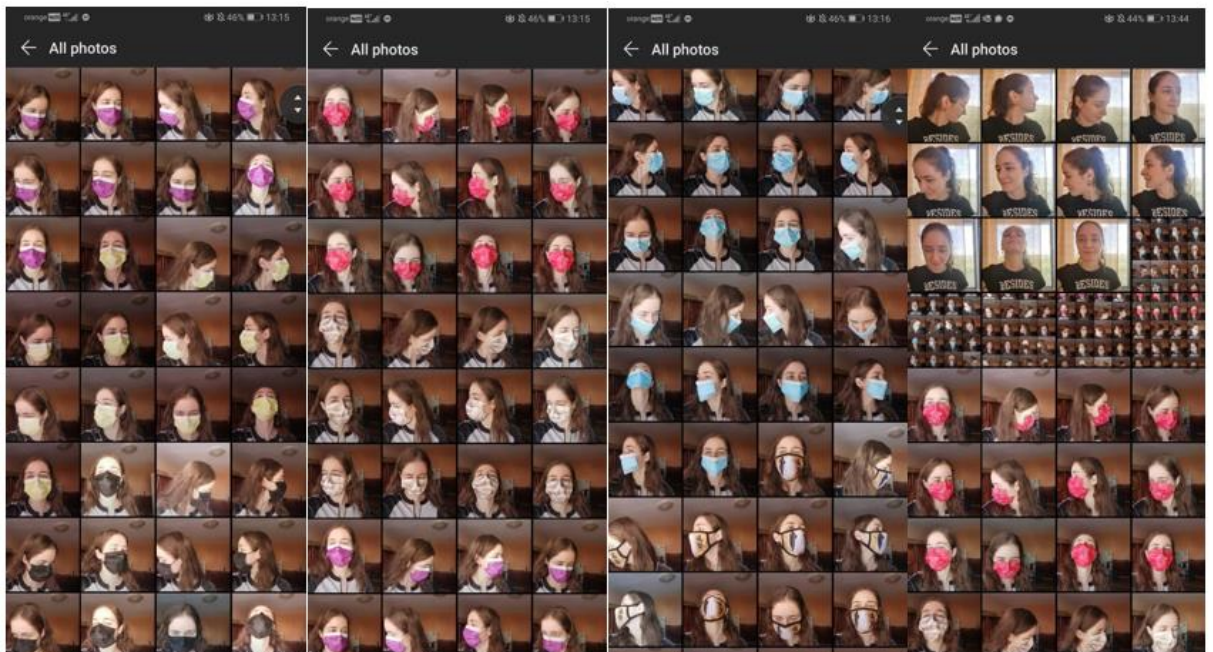


Figură 4.16 Circuit Door-lock: Fingerprint

Apoi, s-a creat codul arduino pentru fiecare și s-a verificat funcționalitatea în sine. Următorul pas în implementare a fost crearea simulatoarelor pentru restul senzorilor. S-a folosit Proteus pentru a simula restul senzorilor din clădire, și s-au multiplicat mai multe circuite de door-lock keypad. S-a încărcat codul și s-a testat funcționalitatea.

4.3.1.6 Implementare funcționalități pe partea de Frontend

S-au creat componentele pentru fiecare funcționalitate din pagini, apoi s-au adunat date de detecție pentru modelele care nu au avut destule date de antrenare, precum FaceMaskDetection, după cum se vede și în imaginea de mai jos:



Figură 4.17 Date de antrenare si testare pentru FaceMask detection

După ce s-au testat algoritmurile de detecție, s-au introdus în aplicație ca și componente în pagina de CCTV Room. Fiecare algoritm poate fi folosit pentru orice cameră de supraveghere de pe orice etaj, prin selectarea opțiunii dorite dintr-un widget aliniat în dreapta ecranului.

S-a implementat styling-ul pentru fiecare pagină în parte. Așadar, pe prima pagină pe care se deschide aplicația (Home), va fi ca default active pagina de Dashboard. Aceasta continue toate activitățile înregistrate de senzori (de exemplu când s-a încercat de mai multe ori să deblocheze cineva o anumită ușă cu parolă, sau cu un card expirat -> lucru ce poate ridica o alarmă asupra unei posibile intrări frauduloase în incinta clădirii). Toate datele sunt afișate sub formă de tabel. Următoarea pagină este Exterior. Aceasta va continue planul 3D al clădirii, pentru care s-a folosit biblioteca @react-three/fiber pentru a face posibil controlul fiecărui mesh al modelului .glTF din aplicația react. Acest model se rotește constant și se poate roti manual în orice direcție, de asemenea sunt posibile și acțiuni de zoom in și zoom out.

Paginile Basement, Floor 1, Floor 2, Floor 3, Floor 4 și Floor 5 conțin un plan al fiecărui etaj, pe care se află senzorii din clădire. Planurile, camerele de supraveghere și restul senzorilor (senzorii de deblocare uși) sunt reprezentați prin figuri .svg. Astfel, senzorii și camerele sunt elemente interactive care își schimbă culoarea în funcție de statusul luat din baza de date (funcționează bine -> verde, are o problemă -> roșu, inactive -> gri). De asemenea, prin trecerea cu mouse-ul deasupra unui element (sensor sau cameră), se afișează un tooltip lângă acesta cu numele item-ului. Prin acțiunea de click, se deschide un widget în partea din dreapta a ecranului, în care apar toate detaliile sensorului / camerei selectate.

Pagina de CCTV Room, conține înregistrarea web în timp real, centrată în mijlocul paginii, cu tipul modelului de recunoaștere și etajul de la care face parte camera selectată, de o parte și de alta a ecranului. Sub cadrul web, se poate selecta camera de pe care se dorește să se monitorizeze înregistrarea video. În partea dreaptă a ecranului se găsesc setările cadrului video. Widget-ul de setări este format din 2 secțiuni: secțiunea de algoritmică (de unde se alege algoritmul de detecție) și secțiunea de planuri arhitecturale (de unde se alege etajul de la care se dorește monitorizarea video).

De asemenea, s-a mai implementat funcționalitatea de notifications, în care se poate naviga de pe orice pagină pe See All Notifications page, printr-un dropdown din navbar. Aceasta pagină prezintă toate notificările înregistrate în aplicație (în momentul de față de senzori când își schimbă statusul), fiind afișate într-un tabel.

4.3.1.7 Implementare funcționalități pe partea de Backend

Pe lângă request-urile create în punctele anterioare, s-a mai implementat un server nou, mai mic, pentru a construi funcționalitatea de comunicare cu senzorii. Acesta este dedicat conexiunii între senzori, baza de date sensorTraces, Sensors, Notifications, și aplicația de frontend. Senzorii înregistrează acțiunile, acestea sunt luate de server-ul mic dedicat senzorilor, adăugate în baza de date. Din baza de date, datele sunt luate de server-ul central, și sunt afișate în front-end. Serverul central se ocupă în mare de comunicarea între toate modulele aplicației. Serverul mic, dedicat senzorilor, de ocupă doar de datele preluate de senzori și conexiunea acestora cu restul aplicației.

4.3.1.8 Integrare Fullstack

Request-urile dintre backend și baza de date au fost implementate în punctele anterioare, așadar, în această secțiune s-a implementat conexiunea între frontend și restul aplicației (conexiunea fullstack: frontend -> backend -> baza de date -> backend -> frontend). Pentru request-urile trimise de frontend, ce folosesc rutele din backend s-a folosit biblioteca axios, astfel:

```
axios.post('http://localhost:5000/sensors/add', sensor)
  .then(res => console.log(res.data))
  .catch(err => console.log(err));
```

Figură 4.18 Post Request: Sensors

```
axios.post('http://localhost:5000/sensors/update/' + this.props.match.params.id, sensor)
  .then(res => console.log(res.data))
  .catch(err => console.log(err));
```

Figură 4.19 Update Request: Sensors

```
deleteSensor(id) {
  axios.delete('http://localhost:5000/sensors/' + id)
    .then(res => console.log(res.data));
  this.setState({
    sensors: this.state.sensors.filter(el => el._id !== id)
  })
}
```

Figură 4.20 Delete Request: Sensors

```
axios.get('http://localhost:5000/sensors/' + this.props.match.params.id)
  .then(response => {
    this.setState({
      name: response.data.name,
      type: response.data.type,
      status: response.data.status
    })
  })
  .catch(function (error) {
    console.log(error);
  })
```

Figură 4.21 Get Request: Sensors - pentru updatare automată(fără refresh)

5 Testare si rezultate

5.1 Metode de testare

5.1.1 Testarea manuală a senzorilor

Se iau senzorii și modulele de care țin și se testează dacă funcționează corespunzător, se testează led-urile, firele dacă transmit curentul așa cum e de așteptat. Se încarcă un program default pe placă pentru a verifica dacă merge (de exemplu programul blink). După ce există confirmarea că toate componentele din circuit merg, se crează programul ce urmează să fie încărcat pe placă. Acesta se compilează și se verifică dacă nu sunt erori de sintaxă. După aceea se încarcă programul pe placă, circuitul fiind deja asamblat, și se testează circuitul. Dacă merge totul corespunzător, testarea manuală se oprește aici. Dacă nu, se ia pe rând fiecare linie de cod și se face debugging pentru a găsi unde e problema. De asemenea se verifică din nou circuitul fizic, pentru a elimina posibilitatea ca un element să nu fie conectat corespunzător.

5.1.2 Testarea manuală a emulatorului

Se instalează programul de emulare și se rulează un exemplu din tutorial pentru a verifica dacă toate setările sunt făcute. Dacă nu merge, se caută pe forumuri fix-uri sau se instalează o altă versiune de emulator, funcțională. După ce există validarea că emulatorul funcționează conform așteptărilor, se construiește proiectul, se încarcă codul pe placă și se rulează aplicația simulată. Se verifică din nou toate entry-points, valorile trebuie să fie conform schemei tehnice.

Dacă simularea se comportă conform așteptărilor, testarea manuală a soluției simulate se oprește aici. Dacă apar probleme, se face debugging la cod. În același timp, poate nu sunt instalate anumite pachete sau biblioteci în emulator, acest aspect este un alt lucru de care trebuie să fie ținut cont (un exemplu similar de scenariu a fost găsit în timpul implementării acestui proiect).

5.1.3 Testarea request-urilor pe partea de backend

Pentru testarea părții de backend, pe lângă testarea conexiunii cu baza de date (s-au folosit loggings în consolă: când este pornit serverul apare un mesaj de succes sau eroare în urma conexiunii cu baza de date), s-au efectuat teste cu toate request-urile definite în server. Postman este o aplicație folosită în acest scop. Toate request-urile (GET, POST, DELETE) pentru toate rutele folosite în aplicație au fost testate cu Postman, pentru a verifica dacă datele sunt adăugate/ pot fi citite/ sunt șterse / sunt updatate corespunzător în baza de date.

5.1.4 Testarea request-urilor pe partea de frontend

În această secțiune, la adăugarea fiecărui element, s-a verificat cum se afișează în pagină elementul implementat, cum apare în comparație cu alte elemente în pagină, dacă este o componentă ce acceptă valori, cum arată dacă nu sunt valori de acceptat (de exemplu când se afișează o listă de notificări, dacă nu e nici o notificare ar trebui ca frontend-ul să specifice că: Momentan nu sunt notificări de afișat), sau dacă sunt mai

multe valori decât se așteaptă (în acest caz, ar trebui să apară un tabel pagination în care să afișeze doar câteva elemente, iar apoi să treacă următoarele într-o pagina nouă, dând acces la utilizator la o navigare ușoară între pagini).

5.1.5 Testarea interfeței grafice – cât de responsive e

Din cauză că există dispozitive de diferite dimensiuni și această soluție vine cu o aplicație web care poate fi accesată de pe orice dispozitiv (PC, laptop, tabletă, smartphone), trebuie testată dacă e responsive. Adică oricum s-ar modifica ecranul aplicației, elementele din pagină trebuie să fie încă vizibile, să nu se suprapună, să își micșoreze / crească dimensiunea dacă e nevoie, sau să se adapteze (de exemplu la unele aplicații, versiunea de desktop are un meniu întreg integrat în navigation bar, în timp ce pentru versiunea de mobile, în nav-bar apare doar o pictogramă cu care se poate deschide/închide un dropdown care afișează elementele din meniu). Developer Tools din Chrome are opțiunea de a alege până și modelul telefonului pentru a verifica diverse dimensiuni ale ecranului telefoanelor mobile, de asemenea se poate da și resize la fereastră.

5.1.6 Testarea modelelor de recunoaștere video

Fiecare model de recunoaștere utilizat de camerele de supraveghere video a fost testat în situații reale pentru a verifica validitatea acestuia, și demonstra că funcționează. Pentru modelul de recunoaștere a obiectelor, s-a verificat că recunoaște persoane, cuțite, și multe alte obiecte din casă. De asemenea s-a verificat că poate recunoaște mai multe obiecte în același timp fără a avea probleme și într-un timp de reacție destul de rapid. Pentru modelele de hand detection și fingerpose se încarcă programul și se testează că recunoaște eficient mâna și poziția, de asemenea că nu ia în calcul și alte poziții ale mâinii, care nu sunt definite. Toate modelele sunt testate în front-end pentru a verifica dacă imaginea video și imaginea recunoscută sunt aliniate (adică elementul ce a fost recunoscut trebuie să fie pe aceeași linie cu canvas-ul).

5.1.7 Testarea conexiunii fullstack (FE->DB, DB->FE)

Pentru această secțiune, prima dată s-a creat o pagină nouă de testare din interfața grafică pentru a nu interacționa din greșală cu alte componente din interfața aplicației. În pagina nou-creată s-au implementat componente pentru fiecare acțiune ce are de-a face cu baza de date (citirea tuturor notificărilor, citirea senzorilor, citirea acțiunii senzorilor, citirea utilizatorilor, editarea senzorilor, editarea utilizatorilor, adăugarea notificărilor, adăugarea senzorilor, adăugarea utilizatorilor, adăugarea acțiunilor senzorilor, ștergerea senzorilor și ștergerea utilizatorilor). Fiecare componentă s-a testat pentru a verifica dacă conexiunea fullstack funcționează conform așteptărilor.

5.2 Cazuri de testare

S-au pus în funcțiune metodele de testare descrise mai sus și chiar dacă în unele cazuri nu au fost rezultate promițătoare din prima (nu mergeau senzorii, nu mergea codul senzorilor, nu se afișau elemente în pagină, nu mergea conexiunea cu diferite baze de date, etc.), într-un final, după mai multe modificări și încercări, aplicația a fost adusă la un stadiu

funcțional, și testele au fost trecute cu brio. După ce s-a trecut de toate metodele de testare descrise în punctele anterioare, s-a mai făcut testarea în funcție de use cases (descrise în secțiunea Cazuri de utilizare). S-a navigat pe fiecare pagină în parte, s-a acționat asupra oricărui element care poate fi acționat, s-au verificat dacă datele sunt afișate corect, și așa mai departe. Rezultatele au fost în final satisfăcătoare, în sensul că aplicația funcționează corect și poate fi folosită ca și un demo al impelmentarii soluției de securitate aduse de această lucrare.

6 Concluzii

6.1 Rezumat contribuții

Având în vedere problema enumerată de câteva ori în cadrul acestei lucrări, și anume faptul că majoritatea soluțiilor de securitate constau în oferirea de servicii de instalare, monitorizare manuală și reparare senzori în cazul în care se semnalează probleme cu aceștia, soluția adusă poate fi văzută ca și o contribuție în domeniul de management al securității în clădiri. Această soluție conține unele tehnologii deja existente (câteva din modelele de recunoaștere video din tensorflow), însă ideea în sine e originală (lucru demonstrat în studiul efectuat asupra altor produse similare aflate pe piață), și de asemenea implementarea aplicației este totodată o contribuție personală.

6.2 Realizarea obiectivelor

Aplicația realizată prezintă o soluție îmbunătățită și mai completă decât multe soluții actuale de securitate care oferă doar servicii de instalare și reparare senzori. Cu această aplicație se poate monitoriza automat statusul tuturor senzorilor din clădire, sistematizat pe fiecare etaj prin planuri arhitecturale peste care sunt așezați senzorii în mod interactiv. De asemenea s-au implementat modele inteligente de recunoaștere video în timp real. Modelele pot fi selectate din aplicație pe camera de supraveghere video dorită de utilizator (de exemplu, camera 3 de la etajul 4).

În timpul testării, soluția propusă a demonstrat că monitorizarea sistemelor de securitate din clădire e centralizată și mult mai eficientă. Aplicația se încarcă rapid, se poate naviga ușor de la o pagină la alta. Request-urile trimise de client sunt preluate de server, acesta face apelul către baza de date, preia datele și trimite un response înapoi la client, clientul afișând datele cu succes.

6.3 Direcții de dezvoltare

Soluția prezentată în această lucrare este doar începutul unei soluții complete de securitate pentru clădiri. Un demo. Mai departe sunt mai multe aspecte ce pot fi cuprinse, ce vor fi enumerate și explicate pe scurt în continuare. În primul rând, trebuie făcută implementarea utilizatorilor cu roluri. Când se intră în aplicație, user-ul se poate loga, și doar dacă credențialele se potrivesc cu ce e înregistrat în baza de date de Users, se pot loga în aplicație și lucra în ea. Parolele vor fi salvate ca și hashed values, deci se va folosi un algoritm de criptare. De asemenea sesiunea va trebui să expire după o perioadă de timp. Numărul de attempts de logare trebuie să fie și el limitat, pentru a evita atacuri de Ddos. După ce e logat, userul poate vedea doar secțiunile de care e responsabil. De exemplu portarul vede doar parterul, în spațiul de care e responsabil și alarmele de

urgență generală. Cel care e responsabil de camera de control, are acces de admin, și așa mai departe.

Soluția actuală e în mare parte responsive, adică dacă se conectează cineva de pe PC, laptop sau telefon, încă sunt aliniate corespunzător elementele în pagina. Ca și dezvoltare ulterioară ar fi eficientizarea elementelor din pagină pe diferite dimensiuni ale ecranului. De exemplu planele arhitecturale să se poată redimensiona la alegere, fără a ieși din cadrul paginii și a fi nevoie de a trage ecranul mai departe (scroll stangă, sus, dreapta, jos).

Totodată, ar fi folositoare o funcționalitate de adăugare manuală a senzorilor. De exemplu presupunând că o companie client are deja instalată aplicația, dar mai târziu mai achiziționează niște senzori în anumite părți ale planului arhitectural de la un anumit etaj, și vrea să adauge acel senzor în poziția exactă în care e amplasat. În momentul de față nu este încă integrată această funcționalitate, dar ar fi de mare folos, cu condiția că valabilitatea acesteia să fie doar disponibilă adminilor. Adică nu vine orice angajat și pune niște senzori noi (asta ar reprezenta o cale de acces a atacatorilor).

În al doilea rând, planul 3D ar trebui integrat să afișeze și senzorii externi (camerele video de supraveghere din exterior, sau alți senzori externi). De asemenea, sistemul va fi extins să suporte mai multe tipuri diferite de senzori, intrând și cei de safety, nu doar de securitate, aducând pe piață o soluție completă de monitorizare automată a clădirilor (sistemul de securitate, detecție efracție, control acces, interfonie, supraveghere video, sistemul surselor de rezervă, anti-îngheț, siguranță, detecție incendiu, detecție și evacuare noxe, protecție și avertizare persoane, evacuare a fumului, sistemul de monitorizare lifturi, sistemul de iluminat, iluminat de evacuare, iluminat de siguranță, monitorizare instalații electrice, multimedia, comunicație, sistemul de încălzire/răcire ventilație și aer condițional, control al jaluzelelor, management al resurselor, distribuție de apă).

O altă direcție o reprezintă modelele de recunoaștere video în timp real. Acestea vor trebui antrenate mai bine și modificate pentru a folosi cât mai puține resurse. Astfel, vor recunoaște mai precis și nu va încetini aplicația. De asemenea, nu va dura atât de mult până se inițializează mecanismele de detecție (în prezent aproximativ 2 secunde). Un punct în plus este aducerea mai multor detecții. Aceste procese pot fi mult mai inteligente și astfel se reduce eroarea umană (dacă supervisorul camerelor video e prea obosit sau nu e atent la toate ecranele).

În continuare, se pot introduce mai multe metrics pe Dashboard, dar pentru mai multe metrice, va fi nevoie de mai multe funcționalități, iar dacă doar câteva din cele menționate mai sus vor fi implementate, atunci nu va fi nici o problemă în adăugarea mai multor metrice. Cu cât mai multe metrice generate automat, cu atât mai vizuală e performanța sistemului, și unde pot apărea probleme sau e nevoie de monitorizare mai în detaliu. Mai mult, comunicarea server-client poate fi îmbunătățită folosind un load-balancer al mesajelor (de exemplu Kafka), pentru a putea avea trafic mai mare, și răspuns mai rapid. Aplicația de front-end se va subscrie la topicuri, din topic server-ul va primi un request, care mai departe va lua datele de care e nevoie și va trimite un response înapoi la topic, de unde front-endul va citi și va afișa pe interfața grafică.

De asemenea, pentru a fi o soluție completă, aplicația trebuie să fie scalabilă la orice client. De aceea, se vor face pachete în funcție de nevoi. De la pachetul cel mai mic, ce va

conține soluții pentru case personale cu minimul de funcționalități, la Pachete mai mari pentru corporații, spitale, hoteluri, bănci, care vor avea nevoie de majoritatea funcționalităților. Totodată, după ce aplicația e finalizată, se poate crea o companie care oferă suport la aplicație, servicii de instalare sisteme, reparare, echipă de security în caz de furt sau amenințări, și de asemenea și propriile device-uri de detecție (CCTV-uri, sisteme deblocare efracție, și așa mai departe).

Se vor efectua mai multe teste precum: performance și load testing, pentru a vedea când începe sistemul să fie mai lent din cauza prea multor acțiuni pe aplicație. Va fi nevoie și de penetration tests pentru a verifica dacă sistemul este secure. Cum ar fi să se implementeze o soluție de securitate, care să fie insecure. Sunt deja destule aplicații inovative, dar insecure.

7 Bibliografie

- 1] M. C. P. H.-D. M. G. N. L. David J Brooks, „Building Automation & Control Systems - An Investigation into Vulnerabilities, Current Practice & Security Management Best Practice,” [Interactiv]. Available: https://www.securityindustry.org/wp-content/uploads/2018/08/BACS-Report_Final-Intelligent-Building-Management-Systems.pdf.
- 2] KrebsOnSecurity, „Target Hackers Broke in Via HVAC Company,” 5 February 2014. [Interactiv]. Available: <https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/>.
- 3] T. N. Y. Times, „Hackers Use New Tactic at Austrian Hotel: Locking the Doors,” [Interactiv]. Available: <https://www.nytimes.com/2017/01/30/world/europe/hotel-austria-bitcoin-ransom.html>.
- 4] Forbes, „Hackers Use DDoS Attack To Cut Heat To Apartments,” [Interactiv]. Available: <https://www.forbes.com/sites/leemathews/2016/11/07/ddos-attack-leaves-finnish-apartments-without-heat/>.
- 5] T. W. Post, „How a fish tank helped hack a casino,” [Interactiv]. Available: <https://www.washingtonpost.com/news/innovations/wp/2017/07/21/how-a-fish-tank-helped-hack-a-casino/>.
- 6] P. H. S. M. P. T. D. H. D. H. K. R. M.R. Brambley, „Advanced Sensors and Controls for Building Applications: Market Assessment and Potential R&D Pathways,” [Interactiv]. Available: https://www1.eere.energy.gov/buildings/publications/pdfs/corporate/pnnl-15149_market_assessment.pdf.
- 7] D. W. J. D. S. D. H. W. G. Kurt W. Roth, „Energy Consumption Characteristics of Commercial Building HVAC Systems Volume III: Energy Savings Potential,” [Interactiv]. Available: https://www1.eere.energy.gov/buildings/publications/pdfs/commercial_initiative/hvac_volume3_final_report.pdf.
- 8] Wikipedia, „Very large floating structure,” [Interactiv]. Available: https://en.wikipedia.org/wiki/Very_large_floating_structure.
- 9] Applied Risk, „I Own Your Building (Management System) - Building Management Systems Security Research,” [Interactiv]. Available: https://www.zeroscience.mk/files/ioybms_gk_2019.pdf.

- ABIREsearch, „Commercial Building Automation,” [Interactiv]. Available:
10] <https://www.abiresearch.com/market-research/product/1032116-commercial-building-automation/>.
- Kaspersky, „Industrial Control Systems Vulnerabilities Statistics,” [Interactiv]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/07/07190426/KL_REPORT_ICS_Statistic_vulnerabilities.pdf.
11]
- M. G. Y. E. C. U. Yisroel Mirsky, „HVACKer: Bridging the Air-Gap by Attacking the Air Conditioning System,” [Interactiv]. Available: <https://arxiv.org/abs/1703.10454>.
12]
- National Vulnerability Database, NIST, „CVE-2018-10660 Detail,” [Interactiv]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2018-10660>.
13]
- National Vulnerability Database, NIST, „CVE-2018-10661 Detail,” [Interactiv]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2018-10661>.
14]
- National Vulnerability Database, NIST, „CVE-2018-10662 Detail,” [Interactiv]. Available: <https://nvd.nist.gov/vuln/detail/CVE-2018-10662>.
15]
- Vdoo's security research teams, „Vdoo Discovers Significant Vulnerabilities in Axis Cameras,” [Interactiv]. Available: <https://www.vdoo.com/blog/vdoo-discovers-significant-vulnerabilities-in-axis-cameras>.
16]
- Metasploit, „Axis Network Camera - .srv to parhand Remote Code Execution (Metasploit),” [Interactiv]. Available: <https://www.exploit-db.com/exploits/45100>.
17]
- MongoDB, „Getting Started with Python and MongoSB,” [Interactiv]. Available: <https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.mongodb.com%2Fblog%2Fpost%2Fgetting-started-with-python-and-mongodb&psig=AOvVaw1HqFVPh479Ytdkp8lTRixd&ust=1625739699536000&source=images&cd=vfe&ved=0CAcQjRxqFwoTCLjNoLze0PECFQAAAAAdAAAAABA E>.
18]
- Farnell, „A000066 - Evaluation Board, MCU 8-Bit, Arduino Uno R3, AVR, ATmega328P,” [Interactiv]. Available: <https://be.farnell.com/arduino/a000066/arduino-uno-evaluation-board/dp/2075382>.
19]
- Bol, „Arduino Mega 2560,” [Interactiv]. Available: <https://www.bol.com/nl/nl/p/arduino-mega-2560/9200000049772089/>.
20]
- Tensorflow, „TensorFlow 2 meets the Object Detection API,” [Interactiv]. Available: <https://blog.tensorflow.org/2020/07/tensorflow-2-meets-object-detection-api.html>.
21]

- F. Berhane, „Tensorflow Tutorial,” [Interactiv]. Available:
22] https://datascience-enthusiast.com/DL/Tensorflow_Tutorial.html.