UNIVERSITATEA TEHNICĂ “Gheorghe Asachi,, din IAȘI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE

DOMENIUL: INGINERIA SISTEMELOR
SPECIALIZARE: AUTOMATICĂ ȘI INFORMATICĂ APLICATĂ

Sistem de monitorizare în proximitate a spațiilor publice de capacitate mare

LUCRARE DE DIPLOMĂ

Coordonator științific
Conf.dr.ing. Gabriela Varvara

Absolvent
Mihailov Emilian

DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII LUCRĂRII DE DIPLOMĂ

Subsemnatul(a) MIHAILOV EMILIAN,
legitimat(ă) cu PA seria AA nr. 0929927, CNP 2006042033411
autorul lucrării SISTEM DE MONITORIZARE ÎN PROXIMITATE A SPAȚIILOR PUBLICE DE CAPACITATE MARE

elaborată în vederea susținerii examenului de finalizare a studiilor de licență organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi" din Iași, sesiunea IULIE a anului universitar 2020-2021, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi" din Iași (Manualul Procedurilor, UTI.POM.02 – Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data
07.07.2021

Semnătura
Mihailov

Cuprins

Sistem de monitorizare în proximitate a spațiilor publice de capacitate mare

Mihailov Emilian

Rezumat

Proiectul constă în realizarea unui sistem care să asigure monitorizarea spațiilor publice de capacitate mare. Soluția generală este formată din două componente: elementul de măsurare a parcurgerii punctelor de acces de către clienți și aspectul vizual grafic reprezentat de către o aplicație web, care afișează diverse date relevante despre toate locațiile unui spațiu public concret. O componentă auxiliară care nu face parte din sistem, dar care își rezervă și ea un rol în suplinirea funcționalității sistemului, este utilizarea unei aplicații mobile. Aceasta oferă administratorului sistemului oportunitatea de a simula transmiterea de date a senzorilor, astfel testând integritatea și completitudinea sistemului. Pentru a asigura comunicația dintre ansamblul de elemente ale sistemului, se utilizează o rețea de tip client-server[1]. Conectarea se efectuează prin intermediul unui router LAN wireless, care oferă și capacitatea de securizare WPA[2].

Pentru a detecta accesarea unei locații, au fost luate în considerație cazurile în care locația respectivă poate avea un punct de acces cu un singur sens (intrarea separată de ieșire) sau cu două sensuri (intrare poate fi utilizată inclusiv ca și ieșire).

Componenta de măsurare este reprezentată de către o diodă laser, un receptor de diodă laser și o placă de dezvoltare la care sunt conectate receptorul și dioda. Pentru punctele de acces cu un singur sens, receptorul va aștepta două întreruperi ale diodei, iar apoi prin intermediul serviciilor API vor fi furnizate datele necesare către baza de date. În cazul punctelor de acces cu două sensuri, pentru a detecta direcția de mișcare a clienților (detectarea intrării sau a ieșirii din incintă) sunt aplicați doi senzori poziționați paralel unul față de celălalt, fiecare analogic cu cei de la intrarea cu un singur sens. Astfel, cunoscând deja care dintre cei doi senzori răspunde pentru detectarea intrării și care de ieșire, la momentul în care unul dintre aceștia este declanșat pentru prima oară, algoritmul de calcul percepe că în locația respectivă s-a produs o mișcare cu un sens bine stabilit și cunoscut și așteaptă cea de-a doua întrerupere. Dacă într-o perioadă limitată de timp s-a declanșat a doua oară același senzor, atunci se va utiliza un serviciu API de înștiințare a bazei de date precum că locația respectivă tocmai a fost parcursă de către un client. Timpul de așteptare a celei de-a doua întreruperi este esențial pentru evitarea unor calculări eronate a clienților care accesează o locație.

Algoritmul de mapare al pinilor pe plăcuța de dezvoltare este realizat astfel, încât să asigure o flexibilitate de utilizare cât mai înaltă. În baza de date este stabilit rolul fiecărui pin de pe o plăcuță concretă, ceea ce face suficient doar conectarea senzorilor necesari la plăcuță și conectarea plăcuței la rețea. Dacă aceasta va fi recunoscută de către baza de date, pași adiționali de configurare nu vor fi necesari. Sistemul va fi profilat conform structurii din baza de date.

Aplicația web a fost concepută pentru administratorul instituției publice, care va putea monitoriza datele, furnizate de către senzorii din locațiile curente, în timp real. Vizualizarea unor statistici, a istoricului de accesare a locațiilor, primirea alarmelor de înștiințare în cazul suprapopulării, reprezintă câteva dintre funcționalitățile site-ului web. Administratorul are dreptul de a crea, edita, șterge, vizualiza datele locațiilor din baza de date.

---

[1] Rețele client-server, în care un calculator îndeplinește rolul de server, în timp ce toate celelalte îndeplinesc rolul de client. A se vedea https://www.afahc.ro/ro/facultate/cursuri/retele_note_curs.pdf , vizitat la 28 iunie 2021.

[2] WPA înseamnă Wi-Fi Protected Access și este o tehnologie de securitate pentru rețelele Wi-Fi . A se vedea https://ro.eyewated.com/ce-inseamna-accesul-protejat-wi-fi/, vizitat la 28 iunie 2021.

Aplicația mobilă oferă oportunitatea de a simula transmiterea datelor de către senzori în timp real, conform mapării din baza de date. Acest lucru ar putea permite testarea corectitudinii mapării pinilor la locații, a dinamicii de lucru a site-ului web, dar și simularea diverselor scenarii de populare a instituției publice. În cadrul proiectului respectiv, aplicația mobilă este utilizată pentru a transmite date manual către locații, alegând un etaj, o locație și un punct de acces. Dar mai există și posibilitatea de a transmite date aleatorii către locațiile instituției.

Limbajul de programare utilizat pentru serviciile web este C#. Pe partea de platformă am folosit ASP.NET Core 3.1.[3] Modelul arhitectural aplicat în cadrul proiectului este MVC(Model-View-Controller) [4] care separă într-un mod inteligibil algoritmul aplicației de nivelul de prezentare. Pentru lucrul cu bazele de date am utilizat Entity Framework Core.[5]

Pentru programarea și rularea plăcuței de dezvoltare Node MCU V3 am utilizat Arduino, care aplică limbajul C/C++ în preponderență. Am utilizat și librăriile chipului ESP8266 de pe plăcuța de dezvoltare, pentru a apela serviciile API.

Pentru partea de interfață grafică am aplicat: HTML, CSS, JavaScript, jQuery, Ajax și librăria Bootstrap a cărei funcționalitate am modificat-o după necesitățile sistemului.

Pentru crearea aplicației mobile am utilizat Xamarin, iar mai exact Xamarin.Forms. Xamarin este o platformă destul de tânără, apărută pe piață de câțiva ani, dar totuși a fost utilizată în proiectul respectiv pentru mentenanța simplificată, flexibilitatea de lucru cu orice tip de sistem de operare mobil cum sunt Android, iOS sau Windows, dar și pentru că mediul de programare al sistemului este .NET, iar Xamarin este o platformă de aplicații mobile open-source pentru .NET.

Ca și sistem de gestionare a bazei de date a fost utilizat Microsoft SQL Server, care utilizează limbajul de interogare SQL.

---

[3] ASP.NET Core este o platformă de dezvoltare pentru aplicații web (site-uri, API-uri etc.) modernă, performantă, extensibilă, ușoară de învățat și de folosit. A se vedea https://agilehub.ro/introducere-in-asp-net-core/ , vizitat la 28 iunie 2021.

[4] Model-View-Controller (MVC) este un model arhitectural care separă funcționalitatea specifică domeniului pentru care este dezvoltat sistemul software de interfața grafică a aplicației, permițând dezvoltarea, întreținerea și testare separată a celor două părți. A se vedea http://www.aut.upt.ro/staff/diercan/data/PSSC/curs-11.pdf, vizitat la 28 iunie 2021.

[5] Entity Framework este un set de tehnologii in ADO.NET ce suporta dezvoltarea de aplicatii software cu baze de date, aplicatii orientate pe obiecte. A se vedea https://profs.info.uaic.ro/~iasimin/Special/Curs_EntityFramework.pdf , vizitat la 28 iunie 2021.

Capitolul 1. Introducere

1.1 Domeniul și contextul abordării temei

SARS-CoV-2, una dintre cele mai alarmante pandemii din ultimele decenii, a divizat omenirea în perioada de înainte și după. Cât de noi nu ar fi tehnologiile umane, am aflat că omenirea încă nu este invincibilă. În fiecare an apar tot mai multe cazuri de boli necunoscute, transmisibile de la om la om, sau și mai rău de la animal la om. Astfel, în tot acest context s-a dorit găsirea unor soluții care ar eficientiza lupta cu bolile transmisibile. Cea mai mare problemă este suprapopularea spațiilor publice. De fapt, însăși problema suprapopulării planetei noastre deja este o amenințare pentru viața și siguranța existenței umane.

Fiecare stat în contextul pandemie încearcă să limiteze riscurile de răspândire a virusurilor, dar și a bolilor contagioase în general. Astfel, una dintre soluțiile eficace de luptă este monitorizarea gradului de ocupare a unor zone publice. Informațiile monitorizării vor facilita luarea deciziilor adecvate și rapide în cazul aglomerării considerabile a unui spațiu.

1.2 Tema propusă

Pentru monitorizarea unor spații publice de capacitate mare, a fost ales drept obiect de studiu Iulius Mall din Iași, care zilnic primește în incinta sa un număr considerabil de oameni. În contextul pandemic, din motive de securizare a sănătății publice, administrația mall-ului a separat căile de acces a intrărilor de cele a ieșirilor. Astfel, în cadrul proiectului de licență a fost luat în considerație și acest aspect. Senzorii care marchează parcurgerea unei locații vor putea fi poziționați în conformitate cu planul de amenajare a punctelor de acces stabilit de către administrația mall-ului.

Datele brute primite de la senzori vor putea fi utilizate pentru interpretarea și analiza ulterioară a fluxului de parcurgere a locațiilor. Administratorul incintei va avea posibilitatea de a monitoriza în timp real numărul de oameni din fiecare locație, rata de ocupare a fiecărui nivel sau locații, istoricul vizitării fiecărei incinte. Din punct de vedere comercial, informația primită de la senzori ar putea pune bazele informaționale a unor strategii de marketing. Va fi posibilă urmărirea nivelului de popularitate a unei locații, orele de vârf când activitatea umană este cea mai activă etc..

Site-ul web creat în cadrul proiectului va demonstra cum anume se efectuează monitorizarea grafică a unui sistem dinamic de tip real. Administratorul va fi considerat de fapt unicul utilizator al sistemului. Acesta va putea monitoriza, analiza sistemul, însă nu se va putea implica direct în derularea acestuia. Datele furnizate către site vor depinde doar de fluxul de oameni care parcurge o locație.

Baza de date a fost ierarhizată într-un mod cât mai logic, pentru a putea fi utilizată și pentru alte instituții. Generalizarea logicii stabilește mobilitatea bazei de date. De altfel, fiecare clădire publică are etaje, fiecare etaj are minimum câte o locație și fiecare locație are măcar un punct de acces. Este o rânduială specifică construcțiilor umane, ceea ce permite crearea unui model aproape comun pentru fiecare instituție.

Toate componentele sistemului vor comunica printr-o rețea de tip wireless, astfel securizarea de transmitere și vizualizare a datelor va fi asigurată de către certificarea WPA (Wi-Fi Protected Access).

Capitolul 2. Fundamentarea teoretică

2.1 Soluții alternative

Urmărirea contactelor dintre oameni a devenit crucială, alături de vaccinarea masivă a populației, pentru a elimina focarele de Covid-19 și pentru a putea în același timp relaxa restricțiile de izolare socială din întreaga lume.

În România, Vodafone și-a informat abonații pe data de 14 mai că va furniza Comisiei Europene și altor agenții europene, precum și „autorităților publice din România, implicate în controlul și combaterea răspândirii infecției cu Covid19”, date care să arate mobilitatea populației.[1]

Rapoarte similare de mobilitate a populației sunt publicate și de Google, care își adună datele de la utilizatorii aplicațiilor sale. Rapoartele Google Maps sunt bazate exclusiv pe utilizarea gadgeturilor care au acces la GPS și evident rețea mobilă. Deși datele acestor rapoarte tind să fie cât mai apropiate de cele reale, totuși există o deviație considerabilă care nu poate fi negată. Nu toți utilizatorii utilizează GPS când sunt într-un spațiu public deschis, așa cum nu toți publică checkin-ul unei locații.

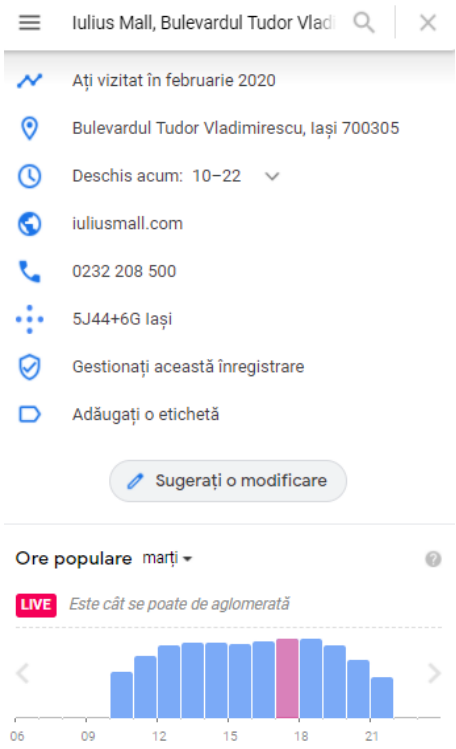

Figura 2.1 Graficul orelor de vârf pentru Iulius Mall pe Google Maps

Un sistem mai sofisticat și mai precis îl oferă compania canadiană Axper, care dezvoltă cele mai noi tehnologii de inteligență artificială pentru a asigura un nivel înalt de precizie în numărarea persoanelor din cadrul spațiilor publice de capacitate mare. Diferențierea dintre copii și adulți, femei și bărbați, excluderea din statistică a personalului, fără necesitate de a utiliza un server, toate datele fiind transmise prin Ethernet, sunt doar câteva din soluțiile pe care le oferă această companie. [2]

Deși Axper oferă o multitudine de soluții pentru monitorizare, totuși există și câteva minusuri. Cel mai mare ar fi mentenanța unui asemenea sistem. Cu cât un sistem este mai complicat, cu atât mai mult acesta depinde de serviciile de suport. Este destul de dificil de a menține în plină funcționalitate o organizare atât de amplă, fiind nevoie de actualizări de tip software regulat. Un alt minus unui asemenea sistem sunt evident costurile. Este destul de scump să obții, dar și să întreții un sistem atât de dinamic.

O altă soluție, care a fost propusă la începutul realizării acestui proiect a fost monitorizarea utilizatorilor de telefoane mobile din cadrul unei rețele. Ideea principală a fost utilizarea senzorilor interni ai unui smartphone. Aici pot fi enumerați: GPS, magnetometru, accelerometru și giroscop. În teorie, senzorii interni ar fi ajutat la numărarea pașilor, iar GPS-ul ar fi furnizat coordonatele terestre. Această soluție nu a fost implementată din cauza nerentabilității, dar și a irealizabilității. GPS-ul nu funcționează în spațiile închise, iar numărarea pașilor nu ar putea să prezică cu exactitate amplasarea unui smartphone în spațiu.

Încă o soluție încercată a fost triangularea telefoanelor mobile cu ajutorul routerelor Wi-Fi. Poziționarea telefonului în spațiu s-ar fi bazat pe compararea indicatorului de putere a semnalului smartphone-ului față de mai multe routere, astfel obținându-se poziția. În practică nu a fost posibilă aplicarea acestei ipoteze, deoarece semnalul Wi-Fi variază foarte mult și afectează funcționalitatea rețelei. Deși a fost implementat și un filtru Kalman pentru minimizarea perturbațiilor, stabilizarea sistemului nu a fost posibilă.

2.2 Servicii WEB

În primul rând, serviciile web sunt o tehnologie. Și ca orice altă tehnologie, au un mediu de aplicații destul de bine definit. O caracteristică cheie a acestui mediu este absența dependenței de caracteristicile și starea unui anumit computer, browser sau furnizor. Prin urmare, accesul la astfel de servicii este acceptat în orice stare. Singura condiție pentru utilizarea sistemului este o conexiune la Internet.

Până în prezent, cele mai frecvente sunt următoarele protocoale de implementare a unui serviciu web:

- SOAP (Simple Object Access Protocol) – care de fapt reprezintă un trio din standardele SOAP/WSDL/UDDI;
- REST (Representational State Transfer);
- XML-RPC (XML Remote Procedure Call).

Ca și exemplu putem considera o companie aeriană. Are multe zboruri, respectiv multe bilete. Transmite informațiile printr-un serviciu web către un site de procesare a călătoriilor. Un utilizator care vizitează site-ul web va putea cumpăra bilete pentru această companie aeriană chiar acolo.

Un alt exemplu de servicii web este un site de urmărire a vremii care conține informații despre condițiile meteorologice dintr-un anumit oraș sau țară în ansamblu. Aceste informații sunt adesea folosite de aplicații terțe.

Informațiile de pe internet sunt diverse. Site-urile web sunt gestionate de diferite sisteme. Sunt utilizate diferite protocoale de transmisie și criptare. Serviciile web facilitează schimbul de informații între diferite site-uri.

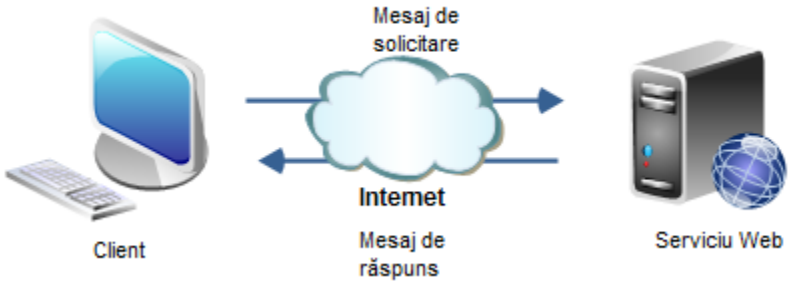

Figura 2.2: Ciclul de transfer de date în cadrul serviciului Web (imagine preluată de pe [3])

2.2.1 Servicii REST

REST este acronimul de la Representational State Transfer și reprezintă un model arhitectural pentru crearea serviciilor web. REST descrie o arhitectura orientată pe resurse, aceasta fiind prezentată în detaliu prima dată în teza de doctorat a lui Roy Fielding's. Motivul realizării unei astfel de arhitecturi a pornit de la faptul că serviciile de tip RPC și cele care în general folosesc SOAP aduc o oarecare complexitate.[4]

Ca și exemplificare, putem considera un calculator matematic de pe orice computer. Când facem clic pe butoane, dorind să obținem rezultate, funcțiile ascunse încep să interacționeze. Când serviciul primește rezultatul, este afișat pe ecran ca un număr gata calculat.

La serviciul REST, arhitectura lucrează într-un mod analogic. Când apăsăm un buton, sunt efectuate diverse operații pentru procesarea și transmiterea informațiilor. Nu numai că pot fi primite date dintr-o singură rețea, dar pot apela și accesa servere la dinstanță pentru a lua ceea ce au nevoie.

Beneficiile unui serviciu REST sunt:

- Componentele sistemului interacționează la o scară destul de mare;
- Toate interfețele sunt comune;
- Componentele pot fi implementate independent unul de celălalt;

- Există elemente intermediare care reduc procentul de latență și sporesc securitatea conexiunii.

Dezavantaje:

- Lipsa specificațiilor;
- Ambiguitatea metodelor de gestionare a datelor.

Există 4 tipuri principale de gestionare a informației:

- Get - primirea, doar transferul de date;
- Post – înregistrarea sau adăugarea a noi date;
- Update – actualizare, baza de date devine împrospătată;
- Delete - ștergere, sunt eliminate date din baza de date;

În calitate de pachete de transmitere a datelor se utilizează JSON. O matrice JSON este transmisă către adresa URL specificată, cum ar fi "*http://host/api/Locations*,,. Acolo se declanșează așa-numita funcție și, în dependență de date deja transmise și de cererea curentă, începe o anumită acțiune. Din exemplul de mai sus, se va apela funcția GET, care va returna datele corespunzătoare acestui URL în format JSON.

O altă caracteristică demnă de menționat este: nu contează de pe ce tip de dispozitiv sunt transmise informațiile, fie că e o aplicație mobilă sau un browser de computer.

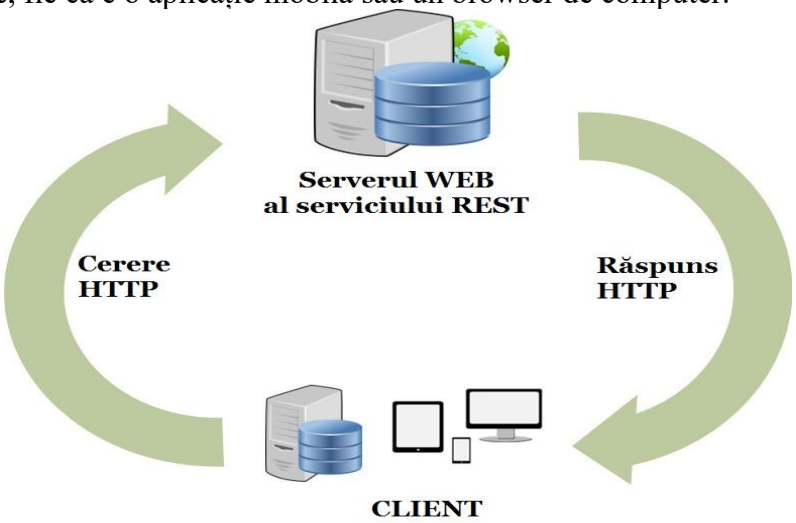

Figura 2.3 Reprezentarea circuitului de comunicație a unui serviciu REST (imagine preluată de pe [6])

2.2.2 Servicii API

API este o arhitectură, creată prin constrângeri și extensii. Dacă este utilizat, este obținut un stil care este optimizat pentru dorințele și potrivit pentru cazurile principale când este posibil să fie nevoie de el.

Sarcina sa este de a reprezenta starea transferului. Deja de la nume devine clar cum ar trebui să funcționeze un software bine conceput. Chiar și o rețea de pagini poate fi gândită ca o mașină de stat:

- utilizatorul vede primul mesaj index;
- navighează prin software făcând clic pe link;
- descoperă rezultatul pe ecran.

Pe Internet, API-urile iau două forme: partea client și partea server.

2.2.3 API client

Specificația HTML5 are mai multe API-uri pe care dezvoltatorii le pot utiliza, cum ar fi Fullscreen și Dialog API. Acestea sunt accesate prin JS, aceste interfețe extind funcționalitatea browserului și înlocuiesc aplicațiile standard. Această funcționalitate a avut un impact uriaș asupra site-urilor web bazate pe API, atât pentru aplicații web desktop cât și pentru aplicații mobile.

2.2.4 API server

Multe companii își lansează propriile API-uri: de exemplu, Netflix are un API care arată filme gratuite abonaților în timp ce caută. Există mii de API-uri diferite pe web, dintre care unele sunt utilizate în mod deschis, altele doar pe site-urile corporative. Toate datele sunt disponibile la cerere de la server.

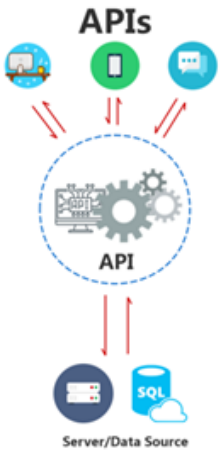

Figura 2.4: Comunicare dintre server și aplicații prin intermediul serviciului API. (imagine preluată de pe [7])

2.3 Tehnologii

Alegerea tehnologiilor potrivite pentru proiect reprezintă o etapă destul de importantă. Acestea stau la baza unei funcționalități fiabile, dacă sunt alese corect. În dependență de domeniul ales, tehnologiile diferă. Selectarea acestora necesită cunoștințe în domeniu, dar și timp. Uneori, alegerea unei tehnologii corecte poate rezolva foarte multe probleme într-o perioadă scurtă de timp, alte ori, ar putea fi unica soluție de soluționare a unor dificultăți.

2.3.1 .NET

.NET este un framework[6] de la Microsoft care permite utilizarea acelorași spații de nume, biblioteci și API-uri pentru diferite limbaje de programare. Cel mai adesea, acestea sunt patru limbi din familia .NET:

- C#;
- Visual Basic;
- Visual C++;
- F#.

În cadrul proiectului, din familia .NET am ales unul dintre limbajele de programare cu cea mai rapidă creștere, cerut pe piață, dar și "convenabil,, din punct de vedere al deținerii unor librării voluminoase, care aduc un raport considerabil în optimizarea codului. Limbajul ales este evident C#.

C# este o modificare a limbajului fundamental C de la Microsoft, concepută pentru a crea cel mai versatil instrument de dezvoltare software pentru un număr mare de dispozitive și sisteme de operare.

Utilizatorilor obișnuiți li se poate părea că acestea sunt un fel de lucruri ale programatorilor care nu le afectează viața în niciun fel. De fapt, are sens și pentru ei.

Dacă nu ar fi .NET, utilizatorii ar trebui să instaleze un mediu de rulare pentru programe în fiecare limbaj. Adică, pentru a rula o aplicație Visual Basic, trebuie să descărcați librăria Visual Basic. Dacă programul este scris în C #, atunci va trebui să descărcați mediul pentru acesta.

2.3.2 ASP.NET CORE

ASP.NET Core este o tehnologie de la Microsoft concepută pentru a crea diverse tipuri de aplicații web, de la site-uri web mici la portaluri web mari și servicii web.

[6] Framework este un software, sub forma unei platforme, necesare pentru facilitarea procesului de dezvoltare și de integrare a diferitor componente ale unui proiect software. A se vedea https://www.metawebart.com/ro/news/chto-takoe-framework1, vizitat la 30 iunie 2021.

Pe de o parte, ASP.NET Core este o continuare a dezvoltării platformei ASP.NET. Dar, pe de altă parte, aceasta nu este doar o altă versiune. Lansarea ASP.NET Core înseamnă de fapt o revoluție pentru întreaga platformă, o schimbare calitativă.

Datorită modularității framework-ului, toate componentele necesare unei aplicații web pot fi încărcate ca module separate prin intermediul managerului de pachete Nuget.

ASP.NET Core include și framework-ul MVC care integrează funcționalitatea MVC, API-ul web și paginile web.

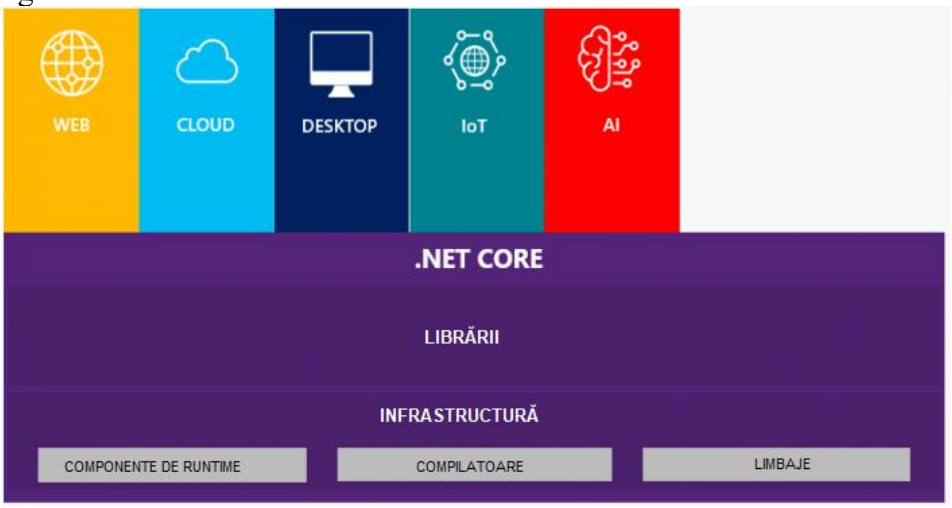

Figura 2.5 Schema de interacțiune dintre componentele de infrastructură, .NET Core și cele ale aplicațiilor (imagine preluată de pe [8])

2.3.3 Entity Framework

Entity Framework este o soluție pentru bazele de date utilizate în programarea limbajelor din familia .NET. Acesta permite interacțiunea cu SGBD utilizând entități, nu tabele. De asemenea, codul folosind EF este scris mult mai repede.

De exemplu, atunci când lucrează direct cu baze de date, un dezvoltator trebuie să-și facă griji cu privire la conectarea, pregătirea SQL și parametrii, trimiterea de interogări și tranzacții. În Entity Framework, toate acestea se fac automat - programatorul, pe de altă parte, lucrează direct cu entitățile și îi spune doar EF să salveze modificările.

Entity Framework oferă trei modalități posibile de interacțiune cu baza de date:

- Database first: Entity Framework creează un set de clase care reflectă modelul de bază de date specific
- Model first: dezvoltatorul creează mai întâi un model de bază de date, din care Entity Framework creează apoi o bază de date reală pe server.
- Code first: dezvoltatorul creează o clasă de model de date care va fi stocată în baza de date, iar apoi Entity Framework generează baza de date și tabelele sale pe baza acestui model

În cadrul proiectului a fost aleasă varianta database first, deoarece la momentul creării proiectului .NET, structura bazei de date deja era creată. Astfel, Entity Framework a fost acel instrument care a ajutat la migrarea structurii tabelelor din baza de date în proiect. În aplicația .NET, contextul bazei de date are ca și proprietăți câte o entitate pentru fiecare tabel. De facto, o entitate în .NET reprezintă o clasă câmpurile căreia se echivalează cu coloanele unui tabel.

Clasele migrate cu EF sunt numite modele sau clase modele. Acestea reprezintă forma proprietăților migrate din baza de date.

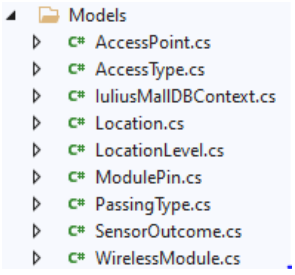

Figura 2.6 Exemplul unei structuri de modele migrate cu Entity Framework

Datorită Entity Framework, clasele generate pot fi modificate, editate sau șterse. Aplicarea acestor funcții se efectuează cu ajutorul comenzilor necesare din linia de comandă specială numită *Package Manager Console*, care se găsește în Visual Studio.

Un exemplu de migrare a tabelelor cu Entiti Framework *database first* utilizat și în cadrul proiectului:

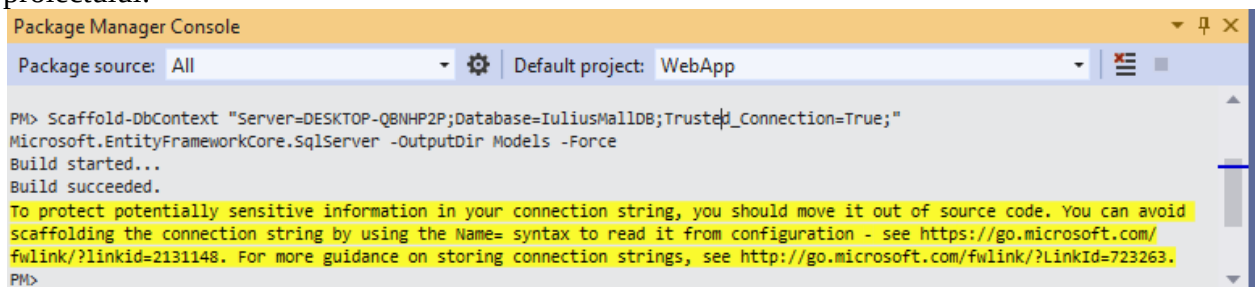

Figura 2.6 Migrarea tabelelor cu Entity Framework

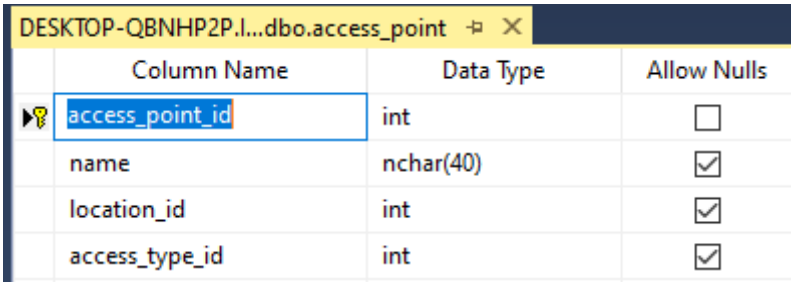

Figura 2.7 Migrarea tabelelor cu Entity Framework

Ca și rezultat se obține o întreagă structura de entități, create după structura din baza de date. Un exemplu al unei entități este tabelul corespunzător punctelor de acces (tabelul access_point, figura 2.7). După se poate observa în figura 2.8, translatarea tabelului într-o entitate a fost efectuată cu succes. Structura entității corespunde întocmai celei din baza de date. Analogic se procedează în mod automat și pentru celelalte tabele.

```
modelBuilder.Entity<AccessPoint>(entity =>
{
    entity.ToTable("access_point");

    entity.Property(e => e.AccessPointId).HasColumnName("access_point_id");

    entity.Property(e => e.AccessTypeId).HasColumnName("access_type_id");

    entity.Property(e => e.LocationId).HasColumnName("location_id");

    entity.Property(e => e.Name)
        .HasMaxLength(40)
        .HasColumnName("name")
        .IsFixedLength(true);

    entity.HasOne(d => d.AccessType)
        .WithMany(p => p.AccessPoints)
        .HasForeignKey(d => d.AccessTypeId)
        .HasConstraintName("FK_access_point_access_type");

    entity.HasOne(d => d.Location)
        .WithMany(p => p.AccessPoints)
        .HasForeignKey(d => d.LocationId)
        .HasConstraintName("FK_access_point_location");
});
```

Figura 2.8 Demonstrarea unei entități migrate în proiect cu Entity Framework, database first

2.3.4 POSTMAN

Postman este conceput pentru a verifica solicitările de la client la server și pentru a primi un răspuns de la partea de back-end[7]. Comunicarea Postman cu back-end-ul poate fi interpretată ca un dialog:

- Postman: “ Dă-mi informații despre locația cu id-ul x din cadrul mall-ului.,,
- Back-end: “Sigur, am verificat solicitarea și este corectă, primiți informațiile despre locația solicită.,,

Un astfel de dialog pozitiv apare dacă nu există erori pe partea back-end și dezvoltatorul a făcut totul conform documentației. Dar acest lucru nu se întâmplă întotdeauna într-o manieră atât de reușită. În practica executării proiectului, au avut loc următoarele dialoguri:

- Postman: “ Dă-mi informații despre locația cu id-ul x din cadrul mall-ului.,,
- Back-end: “Cine sunt eu?,,

sau

- Postman: “Dă-mi informații despre locația cu id-ul x din cadrul mall-ului.,,
- Back-end: “Locația cu id-ul x nu a fost găsită.,,

Răspunsurile de mai sus din back-end au propriile coduri de eroare care apar în răspuns.

În primul caz, acesta este un cod de eroare 500 (Internal Server Error), o eroare de server intern, care indică faptul că serverul a întâmpinat o condiție neașteptată care l-a împiedicat să îndeplinească cererea.

În al doilea - codul de răspuns HTTP de eroare 404 (Nu a fost găsit), serverul nu poate găsi date despre solicitarea primită de la client.

Exact pentru asta a fost conceput Postman - pentru a verifica solicitările clientului, serverului în raport cu cerințele necesare, pentru a asigura că totul funcționează pe partea de back-end.

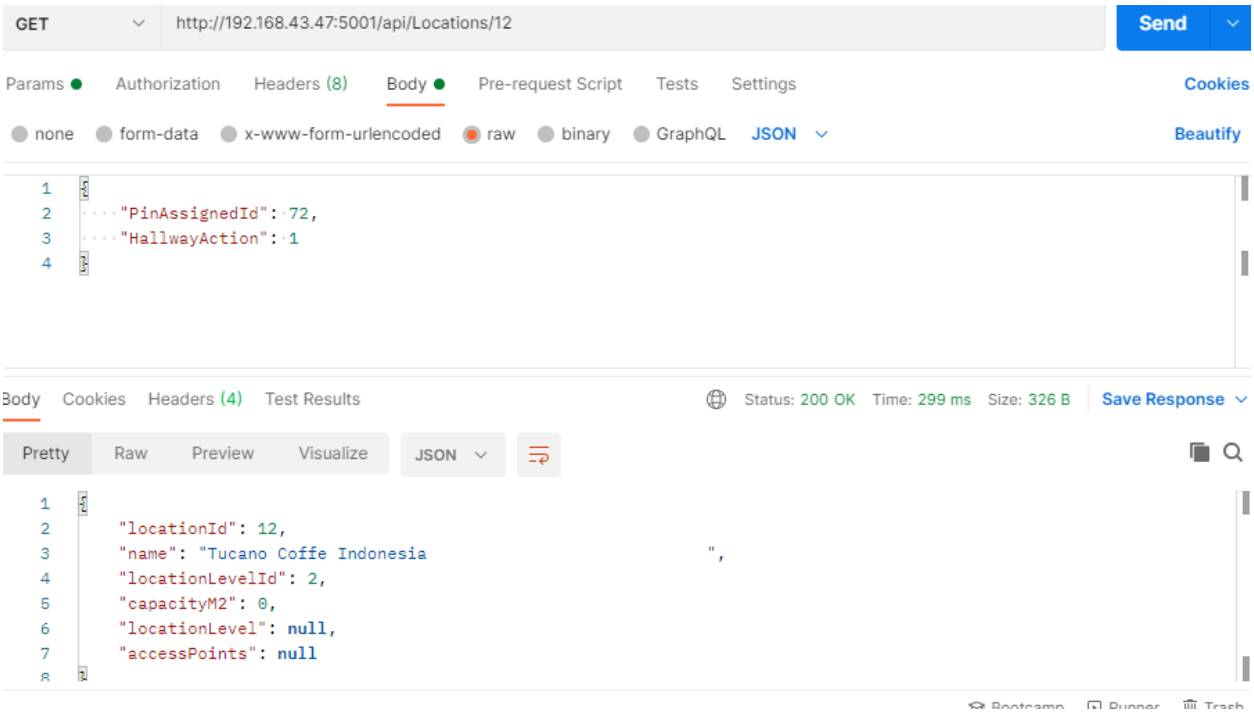

Figura 2.9 Exemplu de transmitere a unei solicitări HTTP de tip GET cu POSTMAN și primirea unui răspuns afirmativ.

[7] Back-end-ul reprezintă partea din spatele unei părți grafice. De obicei acesta este constituit din server, aplicație de interfață și baza de date.

POSTMAN a fost utilizat în cadrul proiectului pentru testarea integrității și corectitudinii de obținere, transmitere sau schimbare a datelor din baza de date. Fără acest instrument, exista un risc mare de corupere a datelor, de transmitere a unor cereri eronate, iar rezolvarea acestor probleme ar fi luat o parte excesivă de timp.

2.3.5 AJAX

AJAX (Asynchronous JavaScript și XML) este o tehnologie de comunicare flexibilă între client și server. Datorită utilizării sale, pot fi executate cereri asincrone către server fără a reîncărca întreaga pagină. Cu toate acestea, în zilele noastre, din ce în ce mai mult, în locul formatului XML, formatul JSON este utilizat pentru comunicarea dintre client și server.

Atunci când este aplicat în ASP.NET MVC, utilizarea AJAX a dus la un întreg concept numit „AJAX discret" și JavaScript discret (Ajax/JavaScript discret). Ideea din spatele acestui concept este că tot codul JavaScript necesar nu este utilizat pe pagina web, ci este plasat în fișiere separate cu extensia *.js. Și apoi, folosind eticheta <script>, ne conectăm la acest fișier de cod din pagina web.

Astfel decuplăm vizualizarea de logica aplicației. Care are avantajele sale. Deci, selectarea unui script într-un fișier descărcabil separat crește performanța site-ului, deoarece fișierul este salvat în cache și apoi încărcat de acolo.

În cadrul proiectului, AJAX a fost utilizat pentru a împrospăta asincron anumite secțiuni de pe pagina grafică, astfel nemaifiind nevoie de a reîncărca întreaga pagină. Acest lucru a fost posibil datorită proprietății AJAX de a transmite cereri HTTP de tip GET, POST.

2.3.6 Xamarin.Forms

Xamarin este un instrument pentru construirea aplicațiilor din familia de limbi .NET (C#, F#, Visual Basic) care permite creaea unui cod consistent care rulează pe diverse platforme mobile cum ar fi Android, iOS și Windows. Aceasta este o tehnologie asemănătoare XAML, adică interfața este descrisă în mod declarativ în format XML.

Un mare avantaj al Xamarinului este posibilitatea de vedea imediat cum sunt situate elementele pe formular și ce proprietăți au. Această abordare este foarte convenabilă, spre deosebire, de exemplu, de Windows.Forms, în care, dacă nu ar fi exista editorul grafic, ar fi extrem de dificil să dezvolți și să editezi interfețe de utilizator, deoarece toate elementele și proprietățile lor sunt create dinamic.

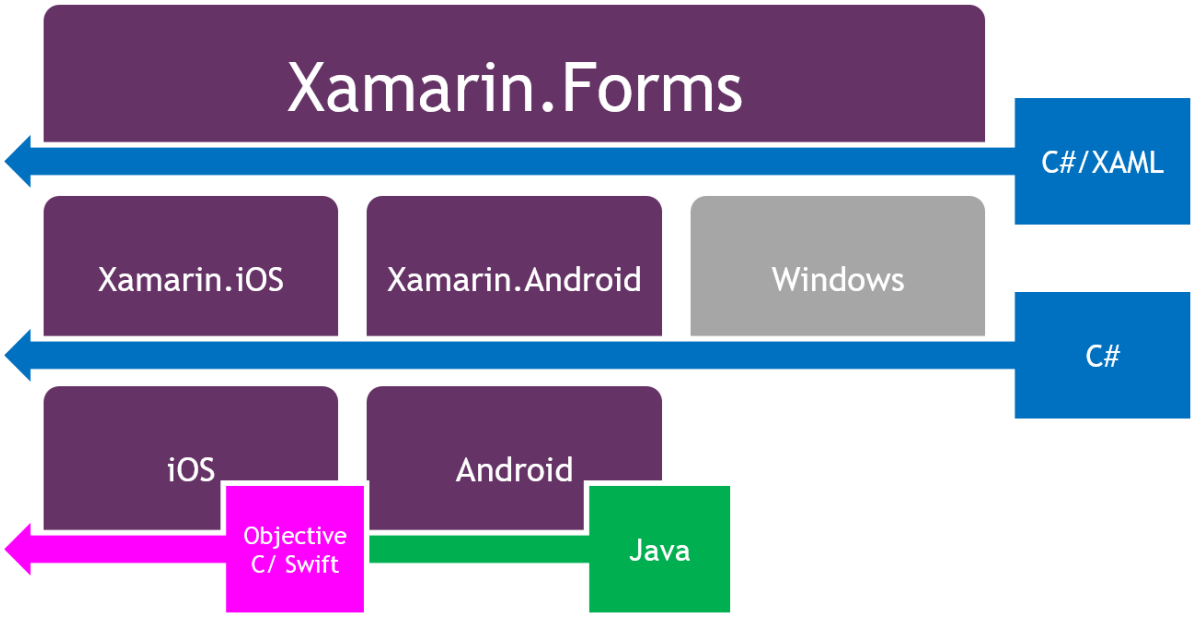

Figura 2.10 Platformele utilizate de către Xamarin.Forms și subnivelurile intermediare (imagine preluată de pe [9])

Xamarin.Forms este preferat pentru capacitatea sa de implementare multiplatformă. Dezvoltatorul scrie cod comun pentru iOS, Android și Windows, iar Xamarin află cum să conecteze codul la API-urile native pentru fiecare platformă. În plus, este posibilă nu numai scrierea generică multiplatformă, ci și specific unei platformei, iar Xamarin va înțelege ce și unde să fie apelat. Unul dintre principalele mecanisme pentru realizarea funcționalității pe mai multe platforme este prezența serviciilor „inteligente" capabile să efectueze apeluri interdependente, adică să abordeze una sau o altă implementare a unei anumite funcționalități, în funcție de platformă. Codul specific platformei poate fi scris nu numai în C#, ci și adăugat la marcarea XAML. În cazul proiectului, versiunea iOS a fost eliminată și o parte a interfeței grafice trebuia ascunsă, dimensiunile unor elemente depinzând și de platformă.

2.3.7 SkiaSharp

SkiaSharp este un API grafic 2D complet multiplatformă. Biblioteca de logo Skia susținută de biblioteca Skia de la Google, alimentează Google Chrome, Mozilla Firefox și stiva grafică Android. Acest API nu este doar puternic, ci și destul de ușor de utilizat și tot ce este necesar este instalarea SkiaSharp prin NuGet[8].

SkiaSharp poate fi utilizat pentru a reda imagini pe platforme, inclusiv iOS, macOS, tvOS, Android și Universal Windows Platform (UWP). SkiaSharp acceptă nu numai aplicațiile mobile native, ci și .NET Standard 1.3 și .NET Core pe macOS și Windows, precum și aplicațiile clasice de desktop Windows și WPF.

2.3.8 Arduino și librăria ESP8266WiFi

Limbajul de programare al dispozitivului Arduino se bazează pe C/C ++ și este legat de biblioteca AVR Libc[9] și vă permite să utilizați oricare dintre funcțiile sale. În același timp, este ușor de învățat, iar în acest moment Arduino este probabil cel mai convenabil mod de a programa dispozitive pe microcontrolere.

Compilatorul Arduino IDE simplifică foarte mult scrierea programelor pentru această platformă și crearea de dispozitive pe Arduino devine mult mai accesibilă chiar și pentru persoanele care nu cunosc prea mult limbajul C/C ++.

De asemenea, este demn de remarcat faptul că dezvoltatorii au dezacorduri cu privire la limbajul de programare Arduino - mai exact, unii insistă că nu este deloc un limbaj. Datorită similitudinii sale cu C ++, unii o numesc pur și simplu bibliotecă.

```
//StaticJsonDocument allocates memory on the stack
//The size of the pool in bytes was consulted with arduino.org/assistant
StaticJsonDocument < 1536 > doc;
DeserializationError error = deserializeJson(doc, wirelessModulePayload);

//Test if parsing succeeds
if (error) {
  return RETURN_RESPONSE_NOK;
}

JsonArray arr = doc.as <JsonArray>();

for (JsonObject repo : arr) {
  pinsMapping[parsingIndex].pinAssignedId = repo["pinAssignedId"];
  pinsMapping[parsingIndex].name = repo["name"];
  pinsMapping[parsingIndex].wirelessModuleId = repo["wirelessModuleId"];
  pinsMapping[parsingIndex].accessPointId = repo["accessPointId"];
  if (repo["pairId"] != NULL)
    pinsMapping[parsingIndex].pairId = repo["pairId"];
  parsingIndex = parsingIndex + 1;
}
```

Figura 2.11 Model de deserializare a unui apel HTTP cu ajutorul ESP8266WiFi

[8] NuGet este un mecanism de partajare a codului, care oferă diverse instrumente pentru anumite roluri.
[9] AVR Libc oferă un subset al bibliotecii standard a limbajului C pentru anumite microcontrolere.

Pentru comunicarea în rețeaua fără fir, configurarea echipamentelor fizice din cadrul proiectului a fost suplinită de către biblioteca ESP8266WiFi. Arduino IDE pentru ESP8266 permite scrierea unor schițe și încărcarea cu un singur clic pe ESP8266 într-un mediu familiar Arduino IDE. Cu ajutorul acestei librării au fost posibile transmiterea de apeluri HTTP către serviciul API și obținerea unor rezultate. Acest serviciu nu este utilizat doar la transmiterea datelor către server. În cadrul proiectului a fost posibilă, dar și necesară folosirea librăriei ESP8266WiFi pentru maparea modulelor echipamentelor fizice, structura cărora deja exista în baza de date. Un exemplu de „despachetare” a datelor furnizate din baza de date este prezentat în figura 2.11.

Capitolul 3. Dezvoltarea sistemului fizic

Studiul și proiectarea sistemului fizic poate crea anumite probleme rare întâlnite în sistemele software. De exemplu, în software timpul necesar pentru a finaliza o sarcină este doar un indicator al productivității, nu corectitudinea software-ului. În sistemele fizice, timpul joacă un rol mult mai important. Fiecare acțiune a unui sistem în timp real este calculată la nivel de milisecunde, iar întârzierile de timp pot să aducă consecințe majore cum ar fi pierderea sau coruperea datelor. Diferența fundamentală între sistemele informaționale și sistemele în timp real, constă din interpretarea parametrului „răspuns de intrare-ieșire”: „Răspunsul corect întârziat este un răspuns greșit".

Funcțiile sistemului fizic, ca orice sistem, fie că e un sistem biologic a unui organism, o societate umană sau un obiect creat artificial în orice scop - poate fi redus la o anumită sarcină de bază sau la un set de sarcini. Sistemul trebuie să aibă timp să proceseze modificările valorilor la intrările sale și să stabilească valorile reale ale ieșirilor de control în conformitate cu algoritmii stabiliți. Astfel, dinamica proceselor din mediu este controlată în conformitate cu principiul feedback-ului: prin măsurarea stării proceselor controlate și organizarea acțiunilor care afectează aceste procese prin intermediul unor mecanisme executive.

Primul pas, dar și cel mai important în stabilirea structurii sistemului fizic, este alegerea mediului de comunicație. Trebuie luată în considerație viteza de transmitere a datelor, securizarea și integritatea acestora.

3.1 Stabilirea mediului de comunicație

Comunicarea fără fir între dispozitivele electronice moderne devine din ce în ce mai relevantă în fiecare an, dată fiind dominanța crescândă a conceptului de *Internet al obiectelor*. Protocolul HTTP și limbajul HTML permit transferul de date în orice loc de pe Pământ unde există o rețea de internet.

Cel mai fiabil sistem de comunicație cu controlerele este cel Wi-Fi. Rețelele Wi-Fi joacă un rol important în lumea tehnologică actuală. Miliarde de dispozitive sunt conectate la rețelele Wi-Fi. În prezent, majoritatea conexiunilor la internet din lume sunt realizate prin intermediul rețelelor fără fir.

De fapt, Wi-Fi este standardul pentru conexiunea LAN fără fir. Pur și simplu, Wi-Fi este un adeziv capabil să conecteze mai multe dispozitive cu un router (router) care poate fi conectat la Internet.

După alegerea mediului, este necesară analizarea mediului de lucru. Necesitatea studiilor de mediu nu ar trebui privită ca o indicație că planificarea rețelei Wi-Fi este mai puțin importantă sau inutilă. Studiul mediului completează planificarea într-un mod pozitiv, indiferent dacă este efectuată înainte de implementare pentru a determina caracteristicile mediului sau după implementare pentru a confirma instalarea corectă. Dacă este omisă faza de planificare, costurile de implementare vor fi mult mai mari.

Este imposibil să se determine cu exactitate caracteristicile unui mediu de frecvență radio fără a ști ce se întâmplă în spectrul său. Sondajul pasiv permite utilizatorului să evalueze toate punctele de acces fără fir și utilizarea canalelor într-o locație pentru a optimiza performanța rețelei.

3.1.1 Examinarea pasivă

Examinarea pasivă se face de obicei înainte și după implementare. Interferențele canalelor adiacente pot fi o cauză majoră a lățimii de bandă reduse și a performanței slabe a aplicației. Prin efectuarea unui sondaj de pre-implementare pasiv, pot fi utilizate datele pentru a selecta în mod eficient canale pentru noi puncte de acces, pentru a evita interferențele între canale de la punctele de acces învecinate. Un sondaj pasiv după implementare ajută la asigurarea faptului că designul curent al rețelei protejează împotriva interferențelor de la un canal la altul.

3.1.2 Examinarea interferențelor

Interferențele care nu sunt asociate cu dispozitivele Wi-Fi măresc zgomotul general și poate face unele canale practic inutilizabile pentru o funcționare wireless fiabilă. Aceste dispozitive sunt omniprezente și, în unele cazuri, sursele de interferență pot fi o parte integrantă a altor soluții implementate în mediu.

Analiza interferențelor Wi-Fi este o componentă cheie a validării mediului, deși măsurătorile de semnal sau de zgomot sunt adesea în centrul unei examinări. Pentru o performanță optimă a rețelei, liniile directoare de verificare a implementării ar trebui să includă întotdeauna hărți de interferențe încorporate și rezultate ale analizei de interferență.

3.1.3 Studiul spectrului

Studiul spectrului folosește un analizor de spectru radio, care compară datele cheie cu planul etajului în timpul studiului mediului. Interogările de spectru sunt adesea efectuate înainte sau după desfășurare, dar dacă se fac înainte, este posibil să nu fie neapărat necesar după și invers. În cazul unui sondaj de pre-implementare, datele pot fi utilizate pentru a plasa mai eficient canalele punctelor de acces. În cazul unui sondaj post-implementare, datele despre spectru pot fi utilizate pentru a analiza impacturile potențiale și performanțele scăzute observate în timp ce se colectează date pentru un sondaj activ.

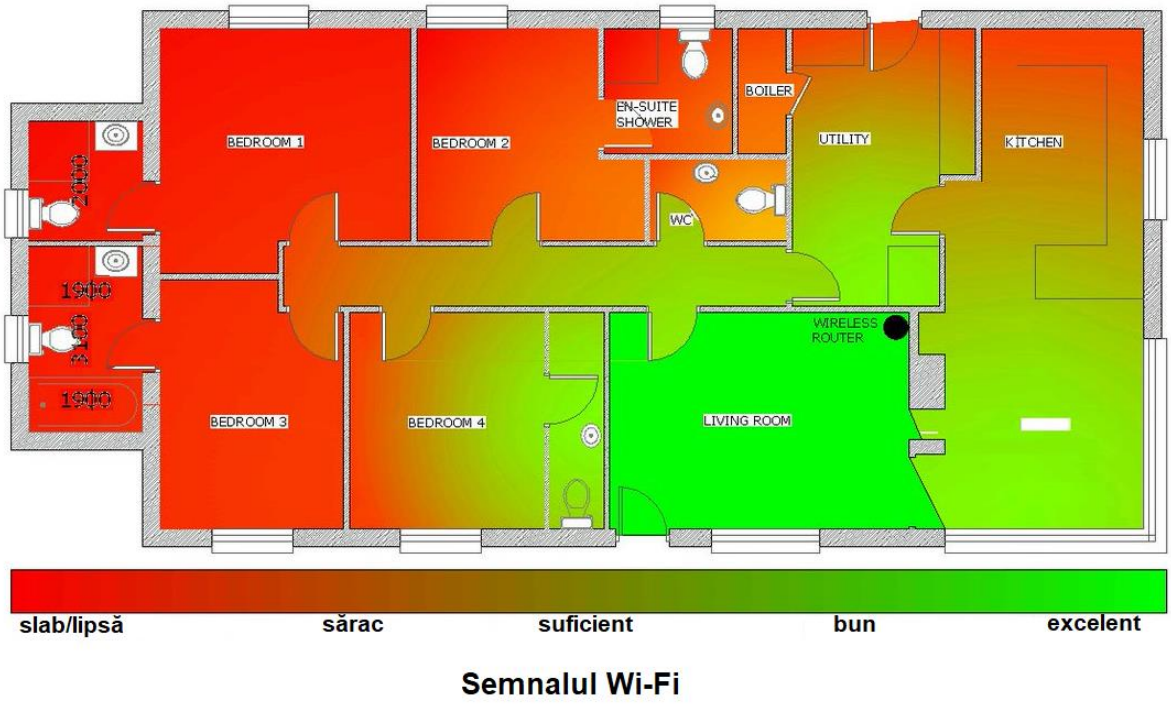

Figura 3.1: Exemplu de repartizare a puterii semnalului Wi-Fi (imagine preluată de pe [14])

3.2 Compoziția sistemului de monitorizare

Funcția de numărare este concepută pentru a îmbunătăți eficiența clădirii și nivelurile de servicii. Nu numai că ia în calcul numărul de persoane care intră în magazin, dar oferă și date critice pentru evaluarea eficienței comerțului și a comercializării și permite, de asemenea, ajustarea muncii zilnice pentru a îmbunătăți nivelul de servicii pentru clienți.

Funcția de numărare a persoanelor oferă următoarele avantaje:

- Determinarea intervalelor de timp cu un flux mare și mic de vizitatori.
- Determinarea prezenței în funcție de zilele săptămânii, orele zilei, compararea prezenței cu alte locații etc.
- Distribuirea eficientă a personalului.

Funcții mai avansate sunt disponibile într-un sistem care produce numărarea vizitatorilor pe baza analizei video. Datele dintr-un astfel de sistem pot fi prezentate sub forma unui grafic de prezență la lucru, care permite personalului magazinului să fie întărit cu un număr mare de asistenți de vânzări în timpul orelor de vârf, fără a fi nevoie ca un astfel de personal să fie prezent toată ziua, ceea ce reduce semnificativ costuri de personal.

Cea mai simplificată variantă este blocarea fasciculului de radiație la intrarea în magazin. Cu toate acestea, un astfel de sistem surprinde faptul în sine și nu oferă o înțelegere a faptului dacă un vizitator a intrat sau a ieșit. Totuși un plus al acestei posibilități este aplicarea ușoară, întreținerea tehnică rapidă, dar și costurile sunt considerabil mai mici față de un sistem video.

În cadrul proiectului a fost luată decizia de a utiliza partea simplificată, adică cea de monitorizare a blocării fasciculului diodei. Totuși problema cu deosebirea intrărilor de ieșiri este una reală. Pentru acest lucru a fost aplicată o îmbunătățire a sistemului.

După exemplul pe care l-am considerat drept model de studiu, adică mall-ul din Iași Iulius Mall, am considerat stabilirea unui raționament de divizare a tipului de intrări. S-a observat că incinta are intrările în clădirile separate de ieșiri, din motive de securitate a sănătății publice. Dar ar putea exista și intrări cu acces dublu, adică să poată fi utilizate pe post de intrare cât și de ieșire. În tot acest context, a fost necesară crearea unei strategii de deosebire a intrărilor de ieșiri pentru locațiile cu un singur sens, sau cu două sensuri.

3.2.1 Strategii de monitorizare

Baza strategiei de monitorizare a numărului de oameni care au parcurs o locație este formată în primul rând de către senzorii care indică un singur sens, fie că e doar intrare, fie că e doar ieșire. Logica este foarte simplă, la momentul în care s-a întrerupt fasciculul diodei, înseamnă că în locația respectivă tocmai s-a produs un eveniment.

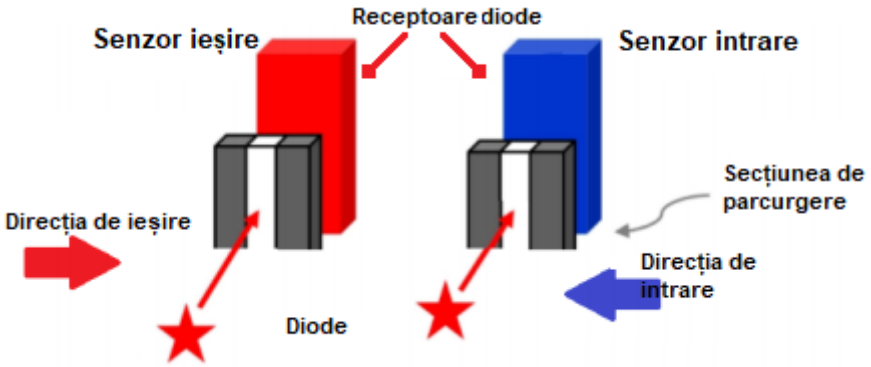

Figura 3.2: Demonstrația logicii dublu senzor (imagine preluată de pe [15])

Pentru locațiile care pot fi parcurse în ambele sensuri, logica senzorului unic nu lucrează, pentru că nu cunoaștem direcția. Soluția pentru determinarea direcției este adăugarea suplimentară a unui senzor, care să fie paralel cu primul, demonstrația poate fi văzută în figura 3.2. Astfel, la un moment dat, primul senzor care va fi întrerupt va stabili sensul de parcurgere a punctului de acces respectiv. Evident că pentru a diminua "accesările false,, se va adăuga un timer pentru declanșarea și celui de-al doilea senzor. În cazul în care timpul a expirat, iar cel de-al doilea senzor pereche nu a fost parcurs, acțiunea se anulează.

Deși ar părea că bazele logicii aplicației fizice au fost deja puse, nu este așa. O problemă majoră întâlnită este identificarea locației, a punctului de acces care tocmai a fost parcurs. Fiind ales drept obiect de studiu un spațiu public de capacitate mare, este evident că numărul locațiilor și a punctelor de acces din cadrul acestora va avea un număr considerabil de mare. Astfel, este vital de a menține o ordine de etichetare a senzorilor atașați fiecărui punct de acces, așa încât la momentul trimiterii unei informații către baza de date, aceasta să recunoască locația și punctul de acces din care tocmai a fost declanșată acțiunea. Mai multe detalii despre modul de mapare a senzorilor va fi prezentat în capitolul "*Implementarea aplicației*,,.

3.2.2 Stabilirea componentelor fizice

3.2.2.1 Plăcuța de dezvoltare

O variantă potrivită pentru proiectul propus a fost alegerea plăcuței de dezvoltare. Unul dintre criteriile importante a fost oportunitatea de conectivitate Wi-Fi cu mediul Arduino. Astfel, cea mai optimală a fost considerată plăcuța de dezvoltare *Node MCU V3 - LoLin (Wi-Fi),* la baza căreia se află chipul ESP8266. Plăcile V3 nu arată diferit de V2, ambele au o ieșire USB mai fiabilă. Placa V3 este fabricată de LoLin, diferența față de placa precedentă este că una dintre cele două ieșiri rezervate este utilizată pentru împământare suplimentară, iar cealaltă este utilizată pentru alimentarea cu energie USB. De asemenea, placa este mai mare decât tipurile anterioare.

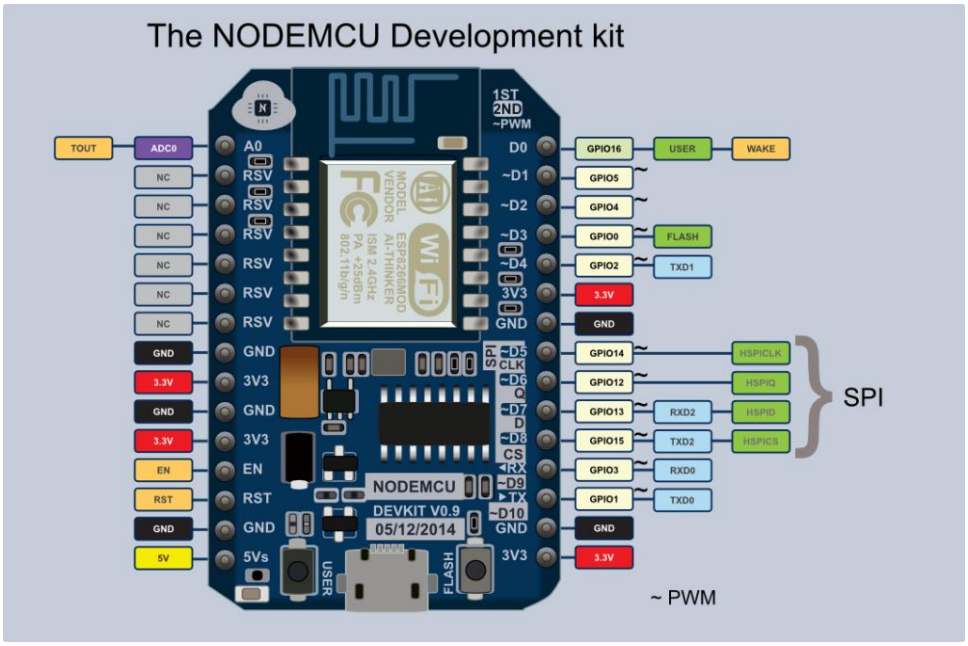

Figura 3.3 Plăcuța de dezvoltare *Node MCU V3 - LoLin (Wi-Fi) cu chipul ESP8266 (imagine preluată de pe [13])*

Specificațiile tehnice ale acestei plăcuțe sunt:

- Are 9 pini digitali (IO) din care 8 suporta PWM si unul analogic (doar intrare).
- La baza acestei placi sta chipul ESP8266 care rulează la 26 MHz, are 4 MB flash si are 160 KB RAM.
- Firmware-ul[10] implicit al chipului permite rularea de scripturi LUA dar poate fi adaptat pentru cod.[13]

Avantajele NodeMcu v3:

- Prezența unei interfețe UART-USB cu un conector micro USB facilitează conectarea plăcii la un computer.
- Memorie flash de 4 MB.
- Posibilitatea de a actualiza firmware-ul prin USB.
- Posibilitatea de a crea scripturi în LUA și de a le salva în sistemul de fișiere.

[10] Firmware-ul este un software de mici dimensiuni care face hardware-ul să funcţioneze şi să facă ce a intenționat producătorul său. A se vedea https://www.digitalcitizen.ro/intrebari-simple-ce-inseamna-firmware/, vizitat 3 iulie 2021.

3.2.2.2 Aplicarea senzorilor

Problema determinării prezenței unui obiect, în ciuda simplității aparente a soluției, rămâne importantă și adesea netrivială pentru multe industrii. La distanțe mici, senzorii inductivi și capacitivi fără contact, precum și diverse versiuni de comutatoare de contact, fac față acestui lucru. Cu toate acestea, există adesea cazuri în care sarcina necesită detectarea unui obiect la distanțe mari – până la câțiva metri. În acest caz, un alt tip de senzori de proximitate vine în salvare - senzori optici.

Senzorii optici, numiți și optoelectronici sau fotoelectrici, sunt denumirea colectivă a unei clase uriașe de dispozitive unite de un principiu comun de funcționare și de elemente structurale de bază. În funcție de sarcină, anumite elemente structurale, precum și dimensiunile senzorului, pot diferi dramatic - de la un cilindru mic de cel mult un milimetru în diametru la dispozitive voluminoase și grele capabile să funcționeze la distanțe de peste o sută de metri.

În general, fiecare senzor optoelectronic este format din două componente principale - un emițător și un receptor. La rândul său, emițătorul include de obicei:

- emițător (LED, laser sau altă opțiune);
- generator;
- element de reglare;
- indicator de lucru.

Receptorul este un dispozitiv mai complex și include:

- fotodiodă
- demodulator
- Declanșator Schmitt
- element de comutare electronic
- element de reglare (potențiometru / buton / șurub)
- funcționarea și indicatorii de funcționare

Pentru simplificarea aplicației, în cadrul proiectului, în calitate de emițător a fost utilizat un modul diodă laser similară celei din figura 3.4.

Specificațiile acestui modul sunt:

- Putere de ieșire: 5mW;
- Lungimea de undă: 650nm;
- Tensiune de lucru:3V;
- Curent de funcționare: mai mică de 20mA;
- Temperatura de funcționare: -10° C până la + 40° C;
- Materialul carcasei: cupru de înaltă calitate;
- Dimensiuni: 6 x 13 mm;
- Conexiunea: firul roșu la pozitiv, firul albastru la negativ; [12]

Figura 3.4 Modul diodă laser
(imagine preluată de pe [12])

Criteriul de alegere a receptorului a fost evident capacitatea de captare a radiației lase, dar nu mai puțin important, capacitatea de a oferi la ieșire un semnal digital care să poată fi perceput și procesat în programul de pe plăcuța de dezvoltare.

Modulul oferă un output digital. În mod normal, output-ul este pe LOW, iar când primește radiație laser, o sa treacă pe HIGH. De exemplu, cu acest produs, vă puteți construi o harpă cu lasere. Produsul vine și cu un circuit cu led indicator de alimentare, rezistențele necesare si condensatorul de filtraj, oferind o interfațare mai bună cu Arduino. Este recomandat sa fie folosit in spații întunecate, lumina soarelui sau alte surse de lumina pot activa receptorul. [11]

Specificațiile tehnice:

- Tensiune de alimentare: 2.7V - 5.5V (recomandat 5V);
- Curent: maxim 3mA;
- Frecvență ieșire: 200kHz - 300kHz;
- Temperatură de funcționare: -25°C - +70°C;
- Receptorul are 3 pini: 1: GND, 2: Output, 3: VCC.[11]

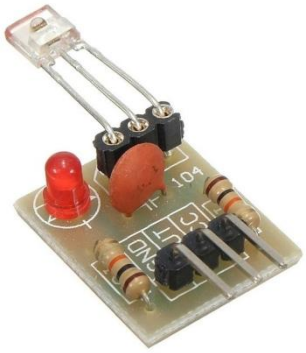

Figura 3.5 Fotodiodă receptoare
(imagine preluată de pe [11])

Capitolul 4. Proiectarea aplicației

Primul pas, dar și cel mai important în realizarea unui produs fie de tip hardware, fie software sau combinat, este analiza problemei. Trebuie stabilite niște obiective concrete, care să aibă în spate niște temeie bine motivate. Timpul la această etapă nu ar trebui să fie constrâns, deoarece tratarea superficială ar putea aduce la întârzieri mult mai mari, pierderi de costuri sau chiar anularea proiectului.

Înainte de proiectarea aplicației, este necesară estimarea posibilităților sistemului, cercetarea echipamentelor ce urmează să fie folosite dar și a resurselor în general. Fiabilitatea dar și realizabilitatea unui sistem trebuie abordată încă din primii pași ai proiectării.

La realizarea proiectului sunt implicați doi factori decisivi: arhitectul, adică persoana care proiectează aplicația și clientul, sau utilizatorul final care va folosi acea aplicație. În urma analizei cerințelor din partea celor două părți, se decid scopurile comune care urmează a fi implementate, la fel cum este reprezentat în figura 4.1.

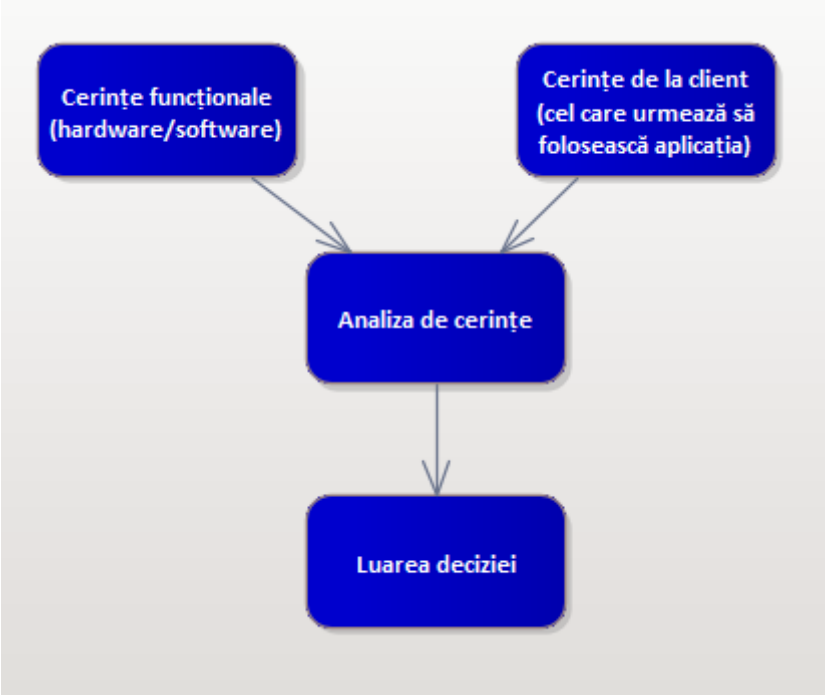

Figura 4.1 Procesul de prelucrare a unei decizii. (Diagrama realizată cu Enterprise Architect)

După luarea deciziei, în mod obișnuit aplicația trebuie să corespundă caracteristicilor:

- Componenta fizică (hardware) să fie cât mai optimizată pentru a asigura un timp scurt de răspuns;
- Componenta software să fie cât mai simplă, dar totodată să poată asigura furnizarea informațiilor necesare;
- Aplicația să fie pregătită de a lucra la o intensitate ridicată și în caz de urgență să existe scenarii de acționare;
- Minimizarea daunelor în cazul unui eșec cât de minor al sistemului (În cazul unor probleme tehnice, este nevoie de a avea la îndemână cel puțin un plan alternativ de acțiune)

Proiectarea organizațională a proiectului, dar și cea arhitecturală se produce în două etape. Prima este crearea unei perspective globale asupra subiectului, iar cealaltă este detalierea fiecărui aspect în parte și stabilirea unei strategii individuale pentru fiecare componentă.

4.1 Arhitectura aplicației

În general, nu există un termen universal acceptat „arhitectură software”. Cu toate acestea, când vine vorba de practică, majoritatea dezvoltatorilor înțeleg deja ce cod este bun și care este rău. O arhitectură bună este în primul rând o arhitectură benefică care face dezvoltarea și întreținerea unui program mai ușoară și mai eficientă. Un program bine arhitecturat este mai ușor de extins și modificat, precum și de testat, depanat și înțeles. Niște criterii considerate rezonabile pot fi:

Eficiența sistemului. În primul rând, programul, desigur, trebuie să rezolve sarcinile atribuite și să-și îndeplinească corect funcțiile chiar și în condiții diverse. Acestea includ caracteristici precum fiabilitatea, securitatea, performanța, capacitatea de a face față sarcinii crescute (scalabilitate) etc.

Flexibilitatea sistemului. Orice aplicație trebuie să se schimbe în timp - cerințele se schimbă, se adaugă altele noi. Cu cât este mai rapid și mai convenabil este să faci modificări asupra funcționalității existente, cu atât mai puține probleme și erori provoacă, iar sistemul este mai flexibil și mai competitiv.

Extensibilitatea sistemului. Abilitatea de a adăuga noi entități și funcții la sistem fără a încălca structura sa de bază. În faza inițială, este logic de a introduce în sistem doar funcționalitatea de bază și cea mai necesară. Dar, în același timp, arhitectura ar trebui să faciliteze creșterea funcționalității suplimentare, după cum este necesar. Și astfel încât efectuarea celor mai probabile schimbări necesită cel mai mic efort.

Testabilitate. Codul care este mai ușor de testat va avea mai puține erori și va rula mai sigur. Dar testele nu numai că îmbunătățesc calitatea codului. Mulți dezvoltatori ajung la concluzia că cerința unei „bune testabilități” este, de asemenea, o forță de ghidare care duce automat la o bună proiectare și, în același timp, unul dintre cele mai importante criterii pentru evaluarea calității acesteia.

Reutilizabilitatea. Este de dorit să proiectăm sistemul astfel încât fragmentele sale să poată fi reutilizate în alte sisteme.

Cod bine structurat, lizibil și ușor de înțeles. Întreținere. De regulă, o mulțime de oameni lucrează la program - unii pleacă, vin alții noi. După redactarea programului, de regulă, trebuie să fie menținut și de persoanele care nu au participat la dezvoltarea acestuia. Prin urmare, o arhitectură bună ar trebui să permită oamenilor noi să înțeleagă sistemul relativ ușor și rapid. Proiectul ar trebui să fie bine structurat, să nu conțină duplicate, să aibă un cod bine format și, de preferință, documentație. Și ori de câte ori este posibil, este mai bine să fie folosite soluții standard, general acceptate, familiare programatorilor din sistem. Cu cât sistemul este mai exotic, cu atât este mai dificil pentru alții să îl înțeleagă.

În general, există multe tipuri de arhitecturi care au o aplicabilitate de succes. Unul dintre cele mai utilizate este sistemul clasic pe trei niveluri, care implică împărțirea unei aplicații în trei niveluri.

Abordând arhitectura pe trei niveluri, atunci o aplicație n-tier ar putea fi împărțită în următoarele niveluri: un server de baze de date, o aplicație web pe un server web și browserul unui utilizator. Adică, fiecare strat ar reprezenta un proces fizic distinct distinct, chiar dacă atât serverul de baze de date, cât și serverul web și browserul utilizatorului ar fi pe același computer.

În cadrul proiectului a fost luată în considerație arhitectura pe trei niveluri. Nivelul client a fost combinat cu cel al aplicației, cele două interacționând cu serverul bazei de date.

Beneficiile unei arhitecturi pe mai multe niveluri sunt:

- Performanta ridicata
  - Încărcările de lucru extinse pe diferite servere oferă o performanță mai bună a implementărilor.
  - Abilitatea de a adăuga capacitate prin creșterea numărului de noduri din cluster.

- Disponibilitate ridicată
  - Toleranță la erori: dacă un nod este în jos, clusterul poate continua să funcționeze.
- Replicare
  - Setul de replici este configurat implicit în acele soluții în care replicarea este obligatorie.
- Securitate
  - Control de acces și securitate îmbunătățit prin separarea datelor de cod.
  - Configurat pentru a funcționa pe un cloud privat virtual (VPC).
  - Autentificare configurată pentru acces extern în majoritatea soluțiilor.[5]

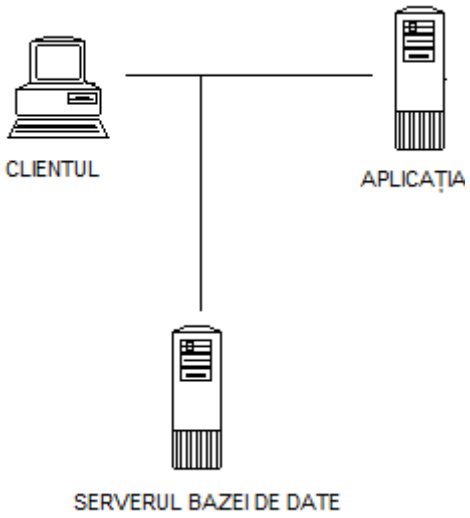

Figura 4.2 Arhitectura pe niveluri a aplicației

Nivelurile din figura 4.2 sunt justificate astfel:

- Secțiunea ce aparține clientului, acesta este nivelul cu care utilizatorul interacționează direct. Acest nivel include componentele interfeței cu utilizatorul, un mecanism pentru primirea de intrări de la utilizator. În ceea ce privește ASP.NET MVC, acest nivel conține vizualizări și toate acele componente care alcătuiesc interfața utilizatorului (stiluri, pagini HTML statice, JavaScript), precum și modele de vizualizare, controlere, obiecte contextuale de solicitare.
- Aplicația, conține un set de componente care sunt responsabile pentru procesarea datelor primite din stratul clientului, implementează toată logica aplicației necesare, toate calculele, interacționează cu baza de date și transferă rezultatul procesării în stratul de prezentare.
- Serverul bazei de date, stochează modele care descriu entitățile utilizate și găzduiește, de asemenea, clase specifice pentru lucrul cu diferite tehnologii de acces la date, de exemplu, clasa contextului de date Entity Framework. De asemenea, stochează depozitele prin care stratul aplicației interacționează cu baza de date.

4.2 Nivelurile aplicației

Aplicația proiectului a fost implementată utilizând arhitectura client-server pe 3 nivele. Ca și mod mai abstract de abordare a acestei arhitecturi, a fost luată decizia de a combina nivelul de interfață grafică(client) cu cel al aplicației, pentru a mări viteza de rulare a cererilor primite de la primul nivel, adică de la utilizator. Modelul structurii respective poate fi vizualizat în figura 4.2.

4.2.1 Nivelul de prezentare

Stratul de prezentare este singurul strat asociat direct cu utilizatorul. În această privință, este foarte important în scopuri de marketing. Stratul de prezentare este utilizat pentru a primi date de la utilizator și pentru a le transmite stratului de logică de afaceri (business-logic) pentru prelucrare ulterioară, iar atunci când datele sunt primite în obiectul de valorificare, este responsabil pentru prezentarea obiectului într-o formă adecvată, care este ușor de înțeles de către utilizatorul. Acest strat poate rula într-un browser web sau ca o interfață grafică pentru un computer sau o aplicație mobilă. Straturile de prezentare ale aplicațiilor web sunt de obicei dezvoltate folosind HTML, CSS și JavaScript.

Pentru fiecare acțiune a utilizatorului există o formă de răspuns în formă grafică. Fie că utilizatorul apasă un buton, fie ca accesează un link de navigare pe altă pagină, fie că nu efectuează nici o acțiune, iar în secțiunea de notificări primește vești.

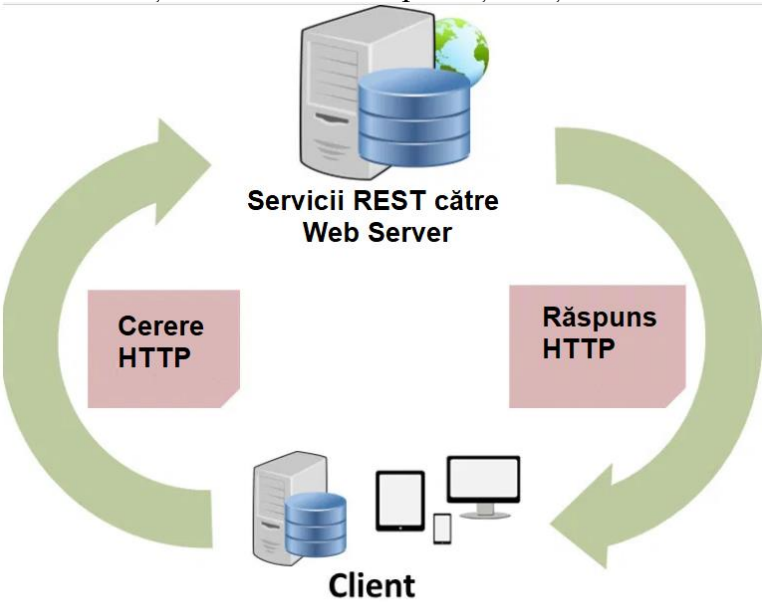

Figura 4.3 Varianta simplificată de interacțiune dintre client și server (imagine preluată de pe [10])

În figura 4.3. observăm cum se produce trimiterea de la client a unor solicitări în contextul utilizării unor funcționalități și primirea răspunsul, ca și consecință a acelor manipulări. Prin intermediul serviciilor de la nivelul aplicației cererile HTTP ajung către server, care furnizează date ce spre final ajung tot la client, inițiatorul procesului. Solicitările HTTP respective nu sunt în mod obligatoriu transmise de către client explicit. În cazul aplicației din proiect, datele sunt transmise în mod automat, ciclic către baza de date prin intermediul unor controlere. Implicarea utilizatorului nu este necesară, pagina își reîmprospătează resursele datorită apelurilor HTTP repetate din JavaScript cu ajutorul AJAX.

4.2.2 Nivelul logic

Nivelul logic sau mai numit și bussines-logic este un nivel, care nu ar putea fi numit întocmai superior celorlalte, dar oricum are o importanță deosebită. Acesta conține un set de componente care sunt responsabile pentru procesarea datelor primite din stratul de prezentare, implementează toate logica aplicației necesare, toate calculele, interacționează cu baza de date și transferă rezultatul procesării în stratul de prezentare.

Unul dintre avantajele distinctive ale platformei ASP.NET Core este utilizarea modelului MVC. Conceptul MVC implică împărțirea unei aplicații în trei componente.

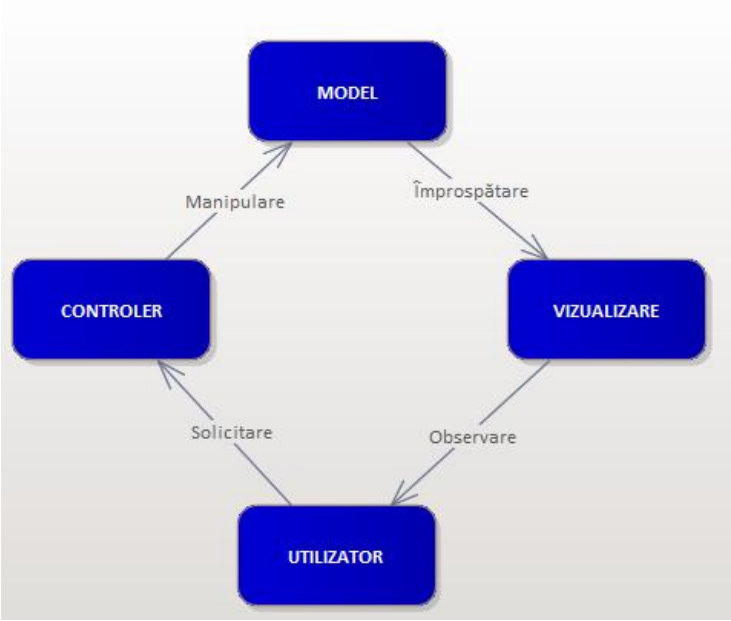

Figura 4.4 Circuitul de parcurgere a șablonului MVC la trimiterea unei solicitări din partea utilizatorului

4.2.3 Nivelul fizic

Nivelul respectiv este răspunzător de comunicarea cu aplicația prin intermediul nivelului logic. Furnizarea datelor de către senzori este stocată în baza de date utilizând nivelele intermediare. Controlerele ar putea fi numite "coridoarele,, prin care trece informația brută primită de la senzori. Acele resurse senzoriale sunt atribuite unui model al aplicației, care prin intermediului resurselor de control, sunt trimise către baza de date.

4.2.3.1 Modelul

Model descrie datele utilizate în aplicație, precum și logica care este direct legată de date, de exemplu, logica de validare a datelor. De obicei, obiectele model sunt stocate într-o bază de date.

În MVC, modelele sunt reprezentate de două tipuri principale: modele de vizualizare, care sunt utilizate de vizualizări pentru a afișa și transfera date și modele de domeniu, care descriu logica de manipulare a datelor.

Modelul poate conține date, poate stoca logica pentru gestionarea acestor date. În același timp, modelul nu ar trebui să conțină logica interacțiunii utilizatorului și nu ar trebui să definească mecanismul de procesare a cererii. În plus, modelul nu ar trebui să conțină nici o logică pentru afișarea datelor în vizualizare.

Pentru încapsularea modelelor am utilizat obiecte de tip DTO (data transfer object). Acest pas a ajutat la includerea în modelele DTO doar a proprietăților necesare pentru efectuarea unor acțiuni, aici referindu-ne la manipularea controlerelor. În situația în care se dorește combinarea mai multor câmpuri din diferite modele, sau crearea a noi entități pentru implementarea lor ulterioară în cadrul controlerelor, la fel se utilizează obiectele de tip DTO. Conversia de la un obiect obișnuit către unul DTO s-a făcut manual în modul următor:

```
public async Task<ActionResult<AccessPointDTO>> DeleteAccessPoint(int id)
{
    var accessPoint = await _context.AccessPoints.FindAsync(id);
    if (accessPoint == null)
    {
        return null;
    }

    _context.AccessPoints.Remove(accessPoint);
    await _context.SaveChangesAsync();

    return new AccessPointDTO
    {
        AccessPointId = accessPoint.AccessPointId,
        Name = accessPoint.Name,
        LocationId = accessPoint.LocationId,
        AccessTypeId = accessPoint.AccessTypeId
    };
}
```

Figura 4.5. Exemplu de mapare manuală DTO.

4.2.3.2 Vizualizare (View)

View-ul este responsabil pentru partea vizuală sau interfața cu utilizatorul, adesea o pagină html, prin care utilizatorul interacționează cu aplicația. O vizualizare poate conține, de asemenea, logică legată de afișarea datelor. În același timp, vizualizarea nu trebuie să conțină logică pentru gestionarea solicitărilor utilizatorilor sau a manipulării datelor. Pentru fiecare View a fost creat câte un controler, care să răspundă de logica manipulării, analizării și furnizării datelor de back-end.

4.2.3.3 Controler (Controller)

Reprezintă componenta centrală MVC care asigură comunicarea între utilizator sau sistemul fizic și aplicație, vizualizare și baza de date. Conține logica pentru gestionarea cererii utilizatorului. Controlerul primește datele utilizatorului și le procesează. Și în funcție de rezultatele procesării, acesta trimite utilizatorului o anumită ieșire, de exemplu, sub forma unei vizualizări umplute cu date de model.

Pentru a menține controlerele într-o structură cât mai fiabilă și de a facilita testabilitatea codului, în cadrul proiectului a fost implementată *Dependency Injection(Injectarea dependențelor)*. A fost mutată crearea interfeței de comunicare cu baza de date și a fost executată legarea obiectelor dependente în afara clasei controlerului. Cu alte cuvinte, contextele bazei de date au fost mutate într-o nouă secțiune numită *Servicii,* care puteau fi apelate prin intermediul unor interfețe create în acest scop.

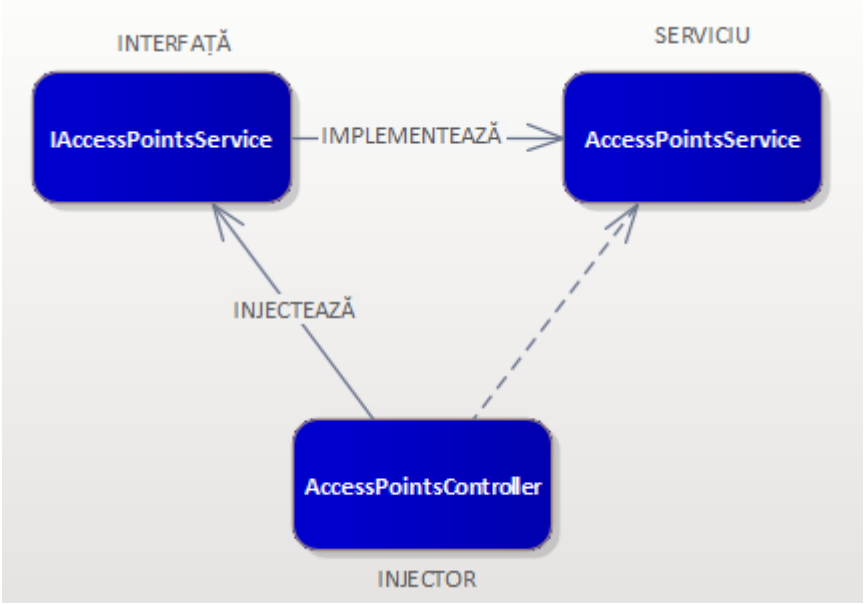

Figura 4.6: Exemplu de implementare Dependency Injection

Nivelul logic poate fi numit pe bună dreptate "inima aplicației", deoarece aici se găsesc mecanismele funcționale de conversație cu baza de date și cu interfața clientului. Totodată procesarea datelor și reinterpretarea acestora are loc tot pe acest nivel.

Pentru a fi interogat acest nivel, este suficient ca utilizatorul(inclusiv sistemul fizic, care poate fi interpretat ca un utilizator al aplicației) să facă o solicitare, care va fi interpretată ca un apel HTTP. Controlerul va prelua comanda, va trimite o solicitare către baza de date, iar în caz de succes vor fi efectuate manipulările necesare. Spre sfârșit, pachetul de date gata procesat este transmis către interfața grafică, adică către utilizator.

Un mare plus al combinării nivelului logic cu cel de utilizator îl constituie posibilitatea de a transmite interogări SQL din controler către baza de date, direct utilizând contextul bazei de date și funcționalitățile LINQ[11].

Încă un plus al .NET-ului care nu a fost remarcat anterior este LINQ-ul. LINQ (**L**anguage-**In**tegrated **Q**uery) este un limbaj simplu și convenabil pentru interogarea unei surse de date. Un obiect care implementează interfața *IEnumerable* (de exemplu, colecții standard, matrice), un set de date, un document XML poate acționa ca sursă de date. Dar, indiferent de tipul sursei, LINQ permite aplicarea acelorași abordări de preluarea datelor. Un exemplu folosit în proiect este figura 4.7. Interogarea sub forma unui șir de caractere, este întocmită într-un format recunoscut de către SQL. Datorită LINQ, comanda este transmisă direct către baza de date, iar rezultatul este stocat într-o variabilă de tip listă.

```
void UpdateSensorData()
{
    var queryString = "SELECT * FROM sensor_outcome WHERE cast(sensor_outcome.time_stamp as Date) = cast(getdate() as Date)";
    todaySensOutcomes = dbContext.SensorOutcomes.FromSqlRaw(queryString).ToList();
}
```

Figura 4.7: Trimiterea unei interogări către baza de date SQL utilizând funcționalitățile LINQ

4.2.4 Nivelul bazei de date

Nivelul bazei de date este utilizat pentru stocarea și gestionarea informațiilor procesate de către aplicație. Rolul său poate fi jucat de un sistem de gestionare a bazelor de date relaționale, cum ar fi Oracle, MySQL sau cum este în cazul proiectului respectiv, Microsoft SQL Server. În aplicația pe trei niveluri, comunicarea cu nivelul de acces la date este realizat exclusiv prin intermediul stratului aplicației.

Pentru a asigura o structură optimă a acestui nivel au fost utilizate legături relaționale. O bază de date relațională este o colecție de date cu relații predefinite între ele. Aceste date sunt organizate ca un set de tabele formate din coloane și rânduri. Tabelele stochează informații despre obiectele reprezentate în baza de date. Fiecare coloană a tabelului stochează un anumit tip de date, fiecare celulă conține valoarea unui atribut. Fiecare secțiune a tabelei este un set de valori conexe care se referă la un singur obiect sau entitate. Fiecare rând dintr-un tabel poate fi etichetat cu un identificator unic numit cheie primară, iar rândurile din mai multe tabele pot fi legate folosind chei străine. Aceste date pot fi accesate în mai multe moduri, fără a fi necesară reorganizarea tabelelor bazei de date.

Toate datele proiectului sunt determinate de către structura specifică bazelor de date relaționale. Astfel, se va observa fenomenul relațional când se va dori obținerea anumitor date care nu au o legătură directă cu un alt tabel, ci se va parcurge structura logică relațională dintre tabele, din legătură în legătură și se va obține rezultatul solicitat. Acest raționament se va folosi de mai multe ori în cadrul proiectului, demonstrând astfel eficiența bazelor de date relaționale.

Pentru a exemplifica în practică stilul bazelor de date relaționale, vom examina figura 4.8. După cum se poate observa, tabelul *module_pin* nu are o relație directă cu tabelul *location*. Dar să presupunem că dorim să aflăm lista de *module_pin* care face parte din *location*. Pentru

[11] LINQ un mecanism pentru manipularea colecțiilor de obiecte (selectare, ordonare, filtrare, agregarea pentru afișarea, modificarea și durabilitatea datelor). A se vedea http://blog.zeltera.eu/?p=896, vizitat la 1 iulie 2021.

acest lucru, în primul rând se vor lua toate datele tip *access_point* din acea locație dată, iar apoi pentru fiecare *access_point*,datorita relației cu *module_pin*, se vor extrage datele din *module_pin* care corespund fiecărui punct de acces. Raționamente analogice se vor utiliza pe tot parcursul proiectului.

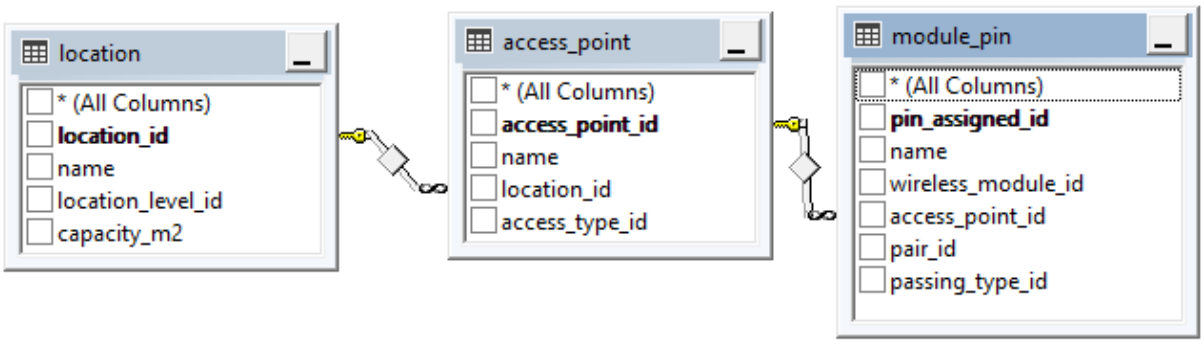

Figura 4.8 Exemplu de baze de date relaționale

4.3 Schema logică a aplicației

După cum a fost menționat și anterior, aplicația proiectului este structurată pe 3 niveluri pentru a sigura un grad de flexibilitate și reutilizabilitate. Se observă că o asemenea structură oferă posibilitatea de a adăuga nivele suplimentare, fără a mai fi necesar de a modifica întreaga structură. O arhitectură pe trei niveluri clasică constă de obicei dintr-un nivel de prezentare, un nivel de domeniu logic și un nivel de stocare.

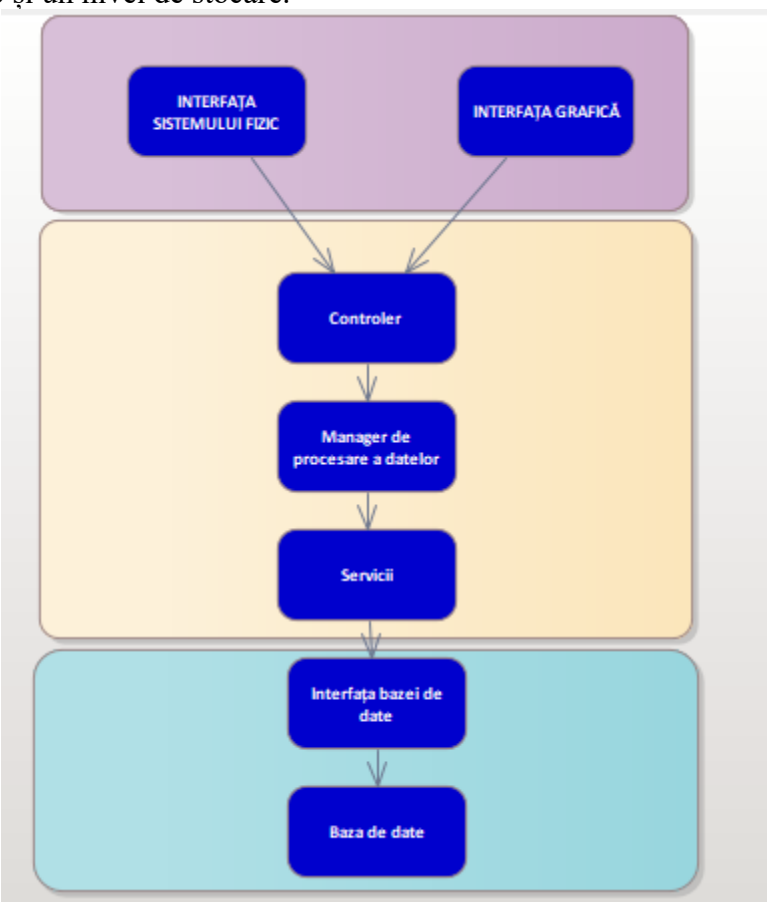

Figura 4.9: Straturile aplicației

Interfața grafică

Mai numită și interfața de comunicare cu utilizatorul, este predestinată unui singur utilizator, administratorului instituției publice. Acesta va avea privilegiile de gestionare a

interfeței grafice. Un lucru foarte important de remarcat este faptul că autentificarea explicită pentru unicul tip de utilizator nu a fost creată din motivul eliminării redundanței.

Interfața sistemului fizic

Acest bloc în esență este foarte asemănător cu cel al interfeței grafice. Unica deosebire este că solicitările făcute de către interfața grafică sunt declanșate de către utilizator la acțiunea de clic sau în mod automat ciclic pentru împrospătarea datelor. Interfața sistemului fizic însă transmite date numai când senzorii aplicației fizice detectează o acțiune. Dar în esență, ambele blocuri furnizează date la producerea unui eveniment. Mai mult de atât, interfața de procesare a solicitărilor nu deosebește o cerere din partea sistemului grafic sau a sistemului fizic. Ipotetic, dacă administratorul sistemului ar avea în fereastra grafică o secțiune de transmiterea unor date "simulate,, sau "false" către baza de date, controlerele aplicației nu vor putea deosebi vreo diferență față de trimiterea fizică a datelor (acest lucru va fi explicat mai detaliat în secțiunea de testare a aplicației). În practică transmiterea datelor false de către administrator nu este posibilă, dar nici necesară. Nimeni nu ar trebui să influențeze blocul de date generate de către sistemul fizic.

Interfața aplicației

- Controlerele care fie sunt apelate în mod automat ciclic, fie la solicitarea utilizatorului.
- Partea de procesare a logicii. Este partea funcțională care răspunde de primirea și prelucrarea cererilor recepționate de la client sau apelurile solicitate bazei de date.
- Serviciile care asigură legătura cu baza de date prin intermediul protocolului HTTP.

Interfața și blocul bazei de date

Această secțiune răspunde de interfațarea conexiunii dintre baza de date și aplicație. Trimiterea interogărilor către baza de date, primirea datelor, stabilirea și verificarea conexiunii cu aceasta asigură nucleul blocului respectiv. Interogările procesate de către Microsoft SQL server sunt foarte rapide. Acest aspect este important, deoarece sistemul aplicativ este dinamic, fiind transmise solicitări multiple simultane către baza de date într-o perioadă scurtă de timp.

Capitolul 5. Implementarea aplicației

Etapa de implementare include procesul care definește aplicația. Tehnologiile, instrumentele, echipamentele fizice utilizate în acest proces trebuie stabilite astfel, încât să asigure o ergonomie optimală și o viteză de lucru relativ rapidă.

Alegerea C# ca limbaj principal al proiectului a fost motivată de structura optimală a acestuia. Multitudinea de componente diverse pe care le deține .NET, dar și organizarea bine stabilită a acestora, pot fi comparate cu o bibliotecă universală, componentele căreia își au propria funcționalitate unicală. Competența oficială în limbajul C# permite dezvoltarea aplicațiilor pentru orice platformă și sistem de operare.

Un alt motiv ar fi Entity Framework, instrumentul principal de comunicare între aplicațiile .NET și bazele de date relaționale. Simplifică maparea obiectelor din software cu tabele și coloane dintr-o bază de date relațională. Îmbunătățește productivitatea prin reducerea sarcinilor de păstrare a datelor. Microsoft dezvoltă activ acest instrument și, în general, câștigă concurența împotriva majorității omologilor săi pentru dezvoltatorul C#.

C# este dominant în construirea de aplicații web. Pentru implementarea aplicației, a fost utilizat ASP.NET Core, care permite dezvoltarea de produse puternice și fiabile, cu o interfață de utilizator avansată. Simplifică serios munca programatorilor, construiește o arhitectură de cod curat.

Microsoft Visual Studio utilizat ca un mediu de dezvoltare integrat, Microsoft SQL Server, folosit pentru crearea și administrația bazelor de date, celelalte componente enumerate mai sus, dar și altele, constituie un întreg sistem unitar dinamic, care lucrează ca un singur organism.

Deși implementarea aplicației fizice nu implică direct sistemul Microsoft, ci mai curând funcționalitățile Arduino, compatibilitatea cu acesta a fost posibilă fără a crea careva probleme. A fost suficientă librăria Arduino ESP8266WiFi pentru a realiza conexiunea cu mediul de lucru al aplicației. Modul de transmitere al informației fiind wireless, prin intermediul modulelor de comunicație a controlerelor.

5.1 Nivelul bazei de date

Este importantă planificarea modulului în care structura de stocare logică a bazei de date va afecta performanța sistemului și diverse operațiuni de gestionare a acesteia. De exemplu, înainte de a crea tabelele pentru baza de date, ar trebui de știut câte câmpuri de date vor compune spațiul de tabel, ce informații vor fi stocate în fiecare spațiu de tabel.

Baza de date ar putea fi considerată scheletul de date a aplicației. Anume de la proiectarea și implementarea bazei de date începe construirea proiectului.

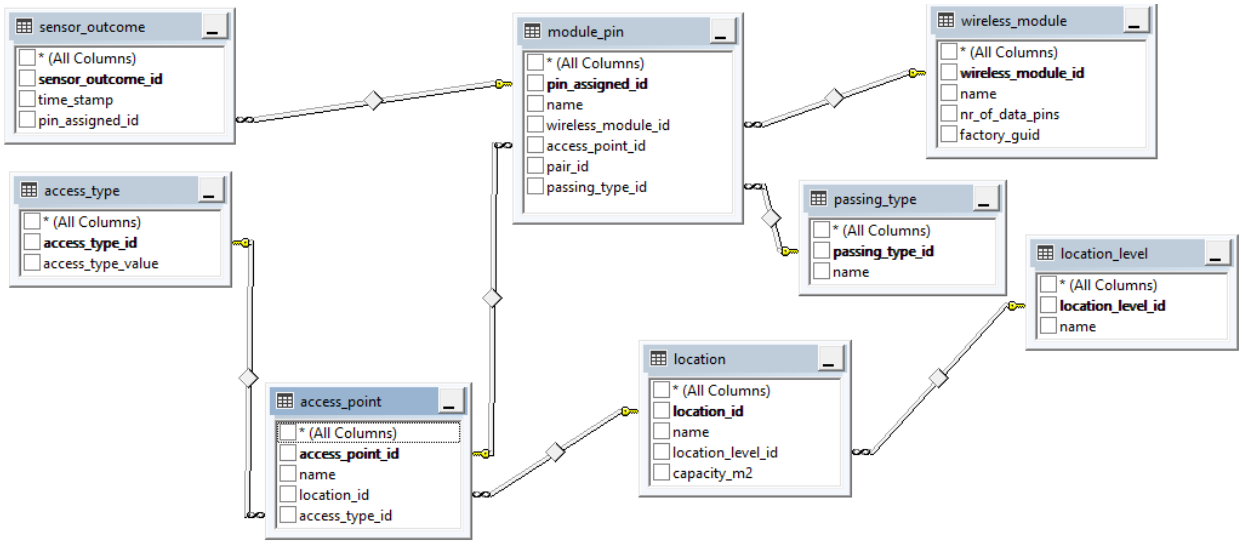

Figura 5.1: Structura bazei de date a aplicației

Dat fiind obiectul de studiu monitorizarea unui spațiu public de capacitate mare, structura clasică a unei clădiri poate fi considerată o ierarhie de nivele, în cadrul cărora sunt locații. Iar fiecare locație are câte unul sau mai multe puncte de acces, cum este demonstrat în figura 5.2.

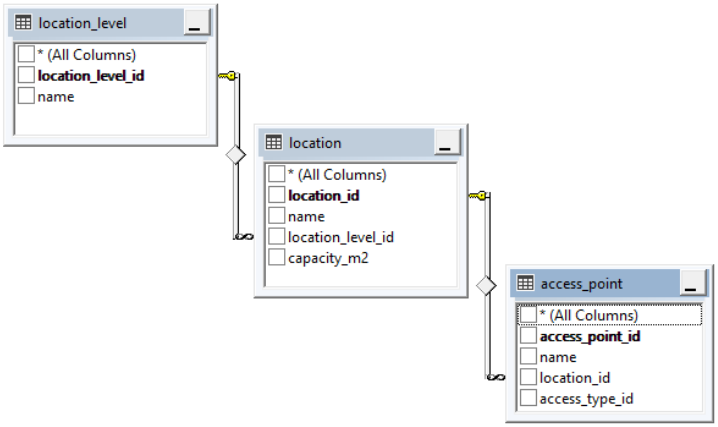

Figura 5.2 Structura clasică a unei clădiri multietajate în baza de date

Dar structura din figura 5.2 nu este suficientă pentru a construi un sistem de monitorizare în timp real cu senzori. Pentru lucrul cu echipamentele fizice, au mai fost adăugate câteva tabele suplimentare, după cum este prezentat în figura 5.3

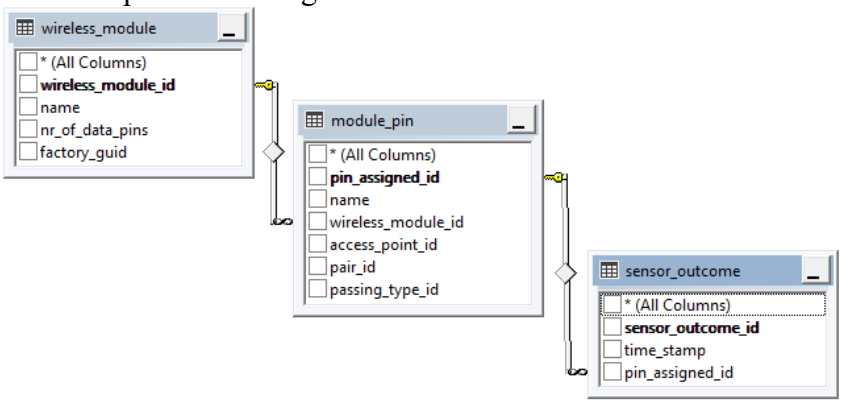

Figura 5.3 Tabelele necesare realizării sistemului fizic

Pentru modulul Node MCU V3, dar de fapt această structură ar putea fi implementată pentru aproape orice plăcuță de dezvoltare, a fost creat tabelul *wireless_module*. Structura

acestuia este simplă. Pe lângă datele clasice cum este id-ul plăcuței în baza de date sau numele, este necesar de a ști câți pini de comunicație are plăcuța respectivă, dar mai ales numărul de identificare din fabrică al acesteia. Anume identificatorul chip-ului plăcuței îi va permite să se conecteze la rețea și să transmită date. Dacă plăcuța nu va fi găsită în baza de date, aceasta nu va fi funcționabilă, adică datele transmise de către senzorii acesteia nu vor fi găsite în baza de date.

Tabelul *module_pin* este subordonatul lui *wireless_module*. Tabelul respectiv răspunde de maparea pinilor. Este un punct critic în care introducerea datelor eronate ar putea aduce la incapacitatea unui senzor de a transmite date către baza de date. Un câmp din acest tabel care ar putea atrage atenția este *pair_id*. Acesta este utilizat în cazul în care punctul de acces poate fi parcurs în ambele sensuri. Astfel este necesar de a cunoaște pin-ul pereche al acestuia, pentru ca aplicația fizică să poată organiza sincronizarea de lucru a celor doi.

Și ultimul din această structură, dar nu și cel mai puțin important este tabelul *sensor_outcome*. Când într-una din locații s-a produs un eveniment, anume în acest tabel sunt stocate datele despre pin-ul care a fost declanșat și ora exactă. Datorită legăturii relaționale dintre tabele, la momentul în care în *sensor_outcome* a fost introdusă o înregistrare, putem cu ușurință să spunem din ce modul face parte acel pin, din ce punct de acces a fost luată proba și respectiv locația. Astfel, aplicația fizică transmite doar pin-ul declanșat, iar aplicația în continuare procesează aceste date.

Ca și tabele ajutătoare, a fost utilizat tabelul *passing_type* și *access_type*. Sunt destul de importante și aceste tabele. Spre exemplu, *access_type* a fost utilizat pentru a stabili tipul unui punct de acces. Doar sunt puncte care nu au nici un punct de acces propriu, acestea folosesc holul clădirii. Sunt puncte cu un singur sens, fie că e doar intrare, fie că este doar ieșire. Evident că s-au luat în considerație și punctele de acces cu ambele sensuri. Este vital de a cunoaște această informație, pentru că știind tipul de acces, vom putea cu siguranță să asigurăm acea locație cu numărul corect de senzori. Funcția tabelului *passing_type* este una foarte simplă, de a indica sensul de parcurgere a punctului de acces. Această informație este atribuită fiecărui pin și este extrem de importantă. Știind sensul de parcurgere vom putea cunoaște numărul de oameni care au intrat sau au ieșit dintr-o locație.

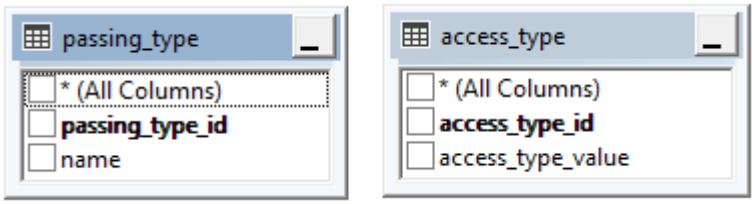

Figure 5.4: Tabelele de suport

5.3 Nivelul fizic

În capitolul 3, a fost abordată tema de stabilire a componentelor fizice. Achiziționarea și conectarea echipamentelor nu este suficientă ca sistemul fizic să lucreze. Este necesară crearea unei logici de receptare și procesare a datelor primite de către acei senzori.

Arduino este mediul în care putem să dezvoltăm logica de comportare a echipamentelor fizice. Am enunțat anterior că plăcuța de dezvoltare Node MCU V3 are incorporat chip-ul ESP8266 Wi-Fi, care permite transmiterea datelor către server prin rețeaua fără fir. Pentru a fi posibil acest lucru, comunicația cu baza de date se va înfăptui prin intermediul controlerelor, despre care vom discuta mai detaliat în secțiunea *Nivelul logic*.

5.3.1 Autentificarea în rețea

Primul pas de configurare a plăcuței îl constituie realizarea conexiunii la rețeaua fără fir. Codul aplicației ar trebui în mod automat să se autorizeze în rețea. În cazul în care datele rețelei nu sunt corecte sau numărul de identificare din fabrică a chipului nu au fost recunoscute de către aplicație, funcționalitatea dispozitivului respectiv va fi corupta.

Pentru autentificarea în rețea se va utiliza librăria ESP8266WiFi. Datele despre numele rețelei și parola vor fi deja cunoscute în aplicație. Dacă administratorul rețelei decide să schimbe parola de la rețea din anumite motive, atunci niciunul din dispozitive nu va mai fi funcțional.

5.3.2 Autorizarea chip-ului

Odată conectați la rețea, dacă serverul este activ, putem să realizăm conexiunea cu acesta. Următorul pas după conectarea în rețea este evident verificarea dacă chip-ul curent este autorizat în baza de date. În tabelul *wireless_module*, există câmpul *factory_guid*. Aici sunt stocate toate codurile de identificare a plăcuțelor din sistem. Utilizând controlerul dedicat acestui tabel, se execută o solicitare către baza de date. În caz de succes, serviciul HTTP GET trimite un pachet de date în format JSON, care va conține datele despre modulul activat, după cum este prezentat în figura 5.5.

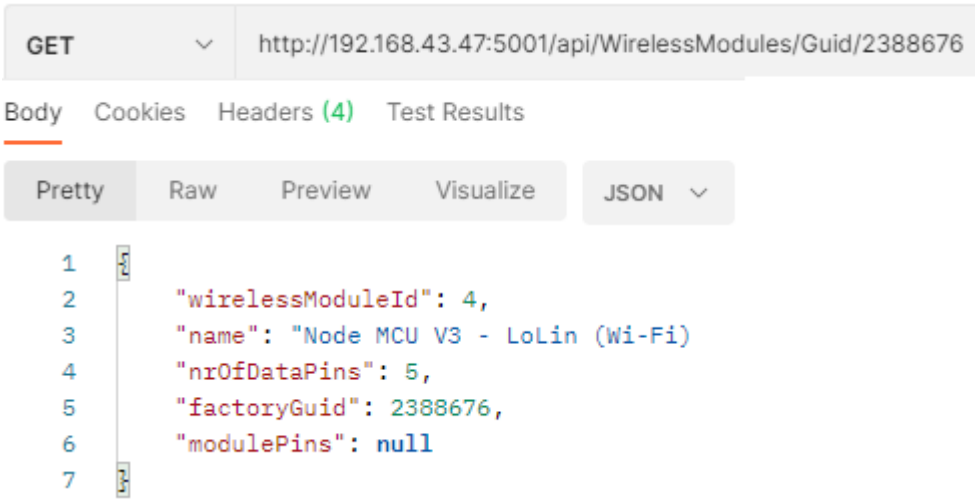

Figura 5.5: Răspunsul pozitiv pentru modului wireless cu id-ul 2388676, realizat cu POSTMAN

Cu ajutorul librăriei ESP8266WiFi, obținem acest pachet de date și îl *deserializăm*[12] după cum este prezentat în figura 5.6.

```
wirelessModulePayload = httpGETRequest(requestDestination);
//StaticJsonDocument allocates memory on the stack
//The size of the pool in bytes was consulted with https://arduinojson.org/v6/assistant/
StaticJsonDocument<16384> doc;

DeserializationError error = deserializeJson(doc, wirelessModulePayload);
//Test if parsing succeeds
if (error) {
  return RETURN_RESPONSE_NOK;
}

localWirelessModule.wirelessModuleId = doc["wirelessModuleId"];
localWirelessModule.name = doc["name"];
localWirelessModule.nrOfDataPins = doc["nrOfDataPins"];
localWirelessModule.factoryGuid = doc["factoryGuid"];
moduleExists = RETURN_RESPONSE_OK;
Serial.println("Checking wireless module succeed!");
return moduleExists;
}
```

Figura 5.6: Procesul de deserializare efectuat cu ajutorul ESP8266WiFi

Dacă verificarea chip-ului a fost realizată cu succes, în continuare poate să fie realizată maparea pinilor.

[12] Procesul invers, de citire a unui obiect serializat pentru a-i reface starea originală, se numește deserializare. A se vedea https://profs.info.uaic.ro/~acf/java/slides/extra/serializare_slide.pdf. Vizitat pe 3 iulie, 2021.

5.3.3 Maparea pinilor

Tabelul *module_pin* a fost realizat cu scopul de a stabili rolul fiecărui pin în raport cu un punct de acces. În baza de date deja se va cunoaște pentru fiecare plăcuță de dezvoltare rolul fiecărui pin. Astfel, nu senzorii conectați la plăcuță stabilesc identitatea unui punct de acces, ci pinii. Putem să comutăm cu locul senzorii, de la un pin la altul, absolut nimic nu se va schimba. Pinii sunt cei care stabilesc în care locație tocmai a fost declanșată o acțiune.

În secțiunea precedentă, am executat autorizarea chip-ului. Controlerul a trimis solicitarea către baza de date. În cazul în care operațiunea a decurs cu succes, va începe procesul de mapare a pinilor. Se trimite o solicitare similară cu cea de la pasul precedent, doar că acum se va folosi controlerul tabelului *module_pin.* Ca și număr de identificare va fi folosit de data aceasta nu id-ul din fabrică a chip-ului de pe plăcuță, ci id-ul local din baza de date corespunzător plăcuței. Această măsură a fost implementată ca un pas minor de securizare datelor, în cazul coruperii acestora. Rezultatul cererii controlerului vor fi toți pinii aferenți plăcuței solicitate în format JSON, după cum este prezentat în figura 5.7.

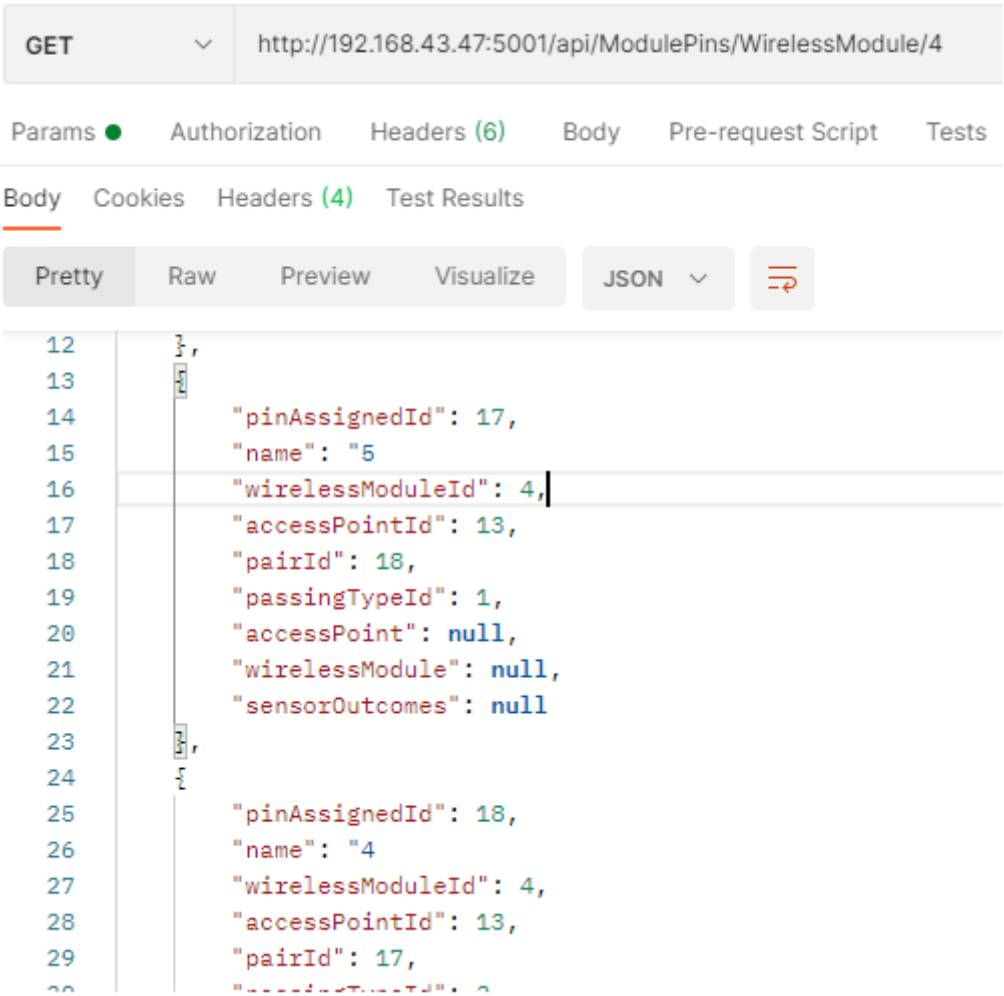

Figura 5.7: Cererea către serviciul API de obținere a pinilor care corespund modulului cu id-ul intern 4

Analogic pasului precedent, aceste date vor fi deserializate și mapate într-o structură de tip vector.

5.3.4 Setarea pinilor ca ieșire

Maparea ne permite în continuare să realizăm operațiuni cu fiecare pin în parte. Astfel, pentru a putea transmite date pe fiecare pin, este necesar parcurge structura de mapări pentru a seta fiecare pin ca *ieșire*.

Până la această etapă a rulat configurarea de inițializare și pregătire a plăcuței de lucru. Fiind conectați la rețea, plăcuța fiind autorizată, iar pinii mapați, poate începe procesul de monitorizare a spațiului public respectiv.

5.3.5 Monitorizarea

Deși această etapă depinde complet de precedentele, ar putea fi considerată corpul aplicației fizice. Codul monitorizării rulează într-o buclă infinită, astfel viteza de rulare a aplicației depinde de optimalitatea codului. O funcționalitate simplă și bine gândită scade numărul de erori în sistem. Aplicația cunoaște deja că avem două tipuri de puncte de acces pentru monitorizare, cu un singur sens, sau cu două.

Logica de interceptare a unei acțiune într-un punct de acces este foarte simplă. Ciclic, aplicația verifică dacă fascicolul de lumină a fost întrerupt la vreun pin. Pentru a înțelege cum s-a produs acest eveniment, trebuie să cunoaștem că la momentul primirii radiației, foto-dioda receptoare este pe HIGH, iar la momentul întreruperii trece în LOW. Astfel, pentru a urmări o intrare, este suficient să cunoaștem starea precedentă a foto-diodei. Comparând cu starea actuală, dacă s-a produs trecerea din HIGH în LOW, putem considera că tocmai a fost parcurs punctul de acces.

Dacă trecerea este confirmată, se verifică dacă senzorul are o pereche sau nu. Cel mai simplu caz este cel în care senzorul nu are o pereche, atunci se efectuează trimiterea unui apel HTTP de tip POST cu ajutorul controlerului corespunzător tabelului *sensor_outcome*. Către server este transmisă doar informația despre id-ul pin-ului, pe care îl găsim în maparea efectuată anterior. Serverul primește datele și le stochează în tabelul *sensor_outcome*.

Dacă punctul de acces are două sensuri de parcurgere, este necesar ne a găsi perechea pin-ului și de a verifica dacă acesta a fost declanșat primul sau nu.

Dacă pin-ul pereche nu a fost declanșat, se consideră că cel curent este declanșat primul, deci el va stabili sensul de parcurgere. Se memorează timpul întreruperii primului senzor și se așteaptă o perioadă prestabilită de timp pentru ca senzorul pereche să fie declanșat. Dacă timpul expiră, intrarea se anulează. În cazul în care senzorul pereche a fost solicitat în timp util, se efectuează exact aceeași pași ca la senzorul cu un singur sens. Unica diferență fiind că la momentul declanșării celui de-al doilea senzor, datele trimise către baza de date îi aparțin primului senzor declanșat, căci el stabilește sensul de parcurgere. Datele sunt transmise către server și memorate.

Dacă senzorul pereche a fost deja declanșat și timpul de declanșare al senzorului curent nu a expirat, se transmite automat identificatorul pinului pereche și se salvează în baza de date.

5.4 Nivelul logic

Acest nivel realizează legătura dintre client și baza de date. Oricine solicită o comunicare cu baza de date poate fi considerat client. Astfel, administratorul aplicației este cel care utilizează interfața grafică, care la rândul său acționează nivelul logic pentru a transmite sau solicita date de la baza de date. Dar totodată, sistemul fizic poate fi considerat și el client. Acesta primește datele necesare mapării pinilor, trimite datele colectate de către pini. Și toate aceste procese au loc nemijlocit prin intermediul nivelului logic.

Primul pas realizat în crearea nivelului logic a fost realizarea conexiuni cu baza de date utilizând Entity Framework și importarea modelelor din baza de date, structura acestora fiind prezentată în figura 5.1.

Al doilea pas, dar și cel mai important a fost crearea controlerelor pentru fiecare tabel. Acestea au fost aplicate la nivelul fizic cât și la nivelul de interfață grafică. Este de remarcat faptul că există două tipuri de controlere în cadrul aplicației.

Primul tip de controlere sunt utilizate exclusiv pentru serviciile REST necesare nivelului fizic. Acest tip de controlere returnează fie un răspuns HTTP de succes, fie corpul modelului solicitat.

Al doilea tip este utilizat exclusiv pentru modelul MVC, adică pentru interfața grafică. Pe lângă controlerele REST clasice (GET, POST, DELETE etc.), cele din MVC pot fi personalizate la modul extrem. Astfel, în proiect pot fi găsite controlere care să returneze numărul de oameni din clădire, rata de ocupare a clădirii, numărul de clienți din fiecare locație etc. Toate aceste date sunt transmise în format JSON, similar celor REST, cum este în exemplul din figura 5.8.

```
public IActionResult LocGuestsOrderDescByGuests()
{
    ManageDashboardData(todaySensOutcomes, dashboardFunctionality.eachLocationNumberOfGuests);
    return new JsonResult(dashboard.locationsGuestsNumber.OrderByDescending(x => x.Value.actualGuestsNumber));
}
```

Figura 5.8: Controler MVC care returnează în format JSON numărul de oameni din fiecare locație

Diferența dintre un controler MVC și unul REST este determinată de obligativitatea utilizării conexiunii cu baza de date în cazul celor REST. Adică orice operațiune ce implică acest tip de controler, obligatoriu va face o solicitare către baza de date, fie că în mod direct, fie că prin intermediul serviciilor. Cel MVC, fiind utilizat pentru partea grafică, poate evident să fie utilizat și pentru obținerea datelor din baza de date, însă cel mai des sunt niște apeluri care declanșează în spate o logică de calculare a unui algoritm, care mai apoi este utilizată în interfața grafică drept sursă de informație.

Se poate constata faptul că utilizarea controlerelor MVC implică pe fundal aplicarea unor algoritmi de rezolvare a solicitărilor parvenite de pe partea grafică. Pentru a fi mai ușor de citit și procesat în cazul unor date mai complexe, au fost utilizate structuri de date de tip dicționar. Cheia primară fiind obligatoriu de tip șir de caractere, altfel JavaScript, care e pe nivelul grafic, interpretează greșit datele.

5.5 Nivelul grafic

I se mai spune și nivelul de prezentare, este dedicat exclusiv interfeței utilizatorului. Ca și instrumente de manipulare a datelor grafice sunt utilizate HTML, CSS, JavaScript, JQuery și nemijlocit Ajax.

Fiind un sistem de monitorizare în timp real, a fost necesară găsirea unor instrumente de împrospătare a datelor ciclic. Adică apelarea repetată a unor controlere, care de fiecare dată fac o cerere către baza de date pentru a reîmprospăta datele. Pentru acest lucru am folosit Ajax. Datele furnizate de către controlere sunt deserializate, iar apoi parcurse element cu element și incluse în contextul necesar.

Pe lângă diverse statistici, s-a simțit nevoia de a crea un sistem simplificat de notificări instantanee pentru cazul în care o locație devine suprapopulată. Rata de ocupare a fiecărei locații se împrospătează ciclic, astfel se stochează într-un dicționar numai datele despre locațiile care au un grad periculos de populare, celelalte nu sunt importante în acest context. Dacă o locație atinge un grad de ocupare considerat între 60% și 89%, atunci administratorul va primi o notificare instantă care îl va înștiința că locația respectivă începe să devină aglomerată. Se reține când ultima dată locația a fost aglomerată și dacă timp de o minută starea locației nu se schimbă, atunci notificare va reapărea. În cazul în care gradul de ocupare depășește 90%, sunt declanșate notificările de alertă. Acestea sunt reverificate cu o frecvență de 30 de secunde pentru ca administratorul să observe cu siguranță notificarea și să ia măsuri. Este necesar de a puncta faptul că dacă o locație are gradul de ocupare cuprins între 60% și 89%, iar gradul precedent era peste 90%, atunci sistemul de alarme nu înștiințează administratorul. Se consideră că au fost luate măsuri, din această cauză se reverifică starea locație peste încă o minută. Daca e în aceeași stare de pericol, se activează alarma.

Capitolul 6. Testarea aplicației

Testarea aplicației va parcurge în două etape. Prima etapă va constitui testarea aplicației web, iar a doua testarea aplicației fizice. Scopul principal al oricărei testări, inclusiv testarea aplicațiilor web sau fizice, este de a detecta toate erorile din software sau respectiv hardware și de a elabora recomandări pentru prevenirea lor în viitor.

6.1 Testarea aplicației grafice

Testarea este un termen foarte general. Testarea aplicației grafice poate fi de mai multe feluri:

- Testare funcțională;
- Testarea compatibilităților;
- Testarea performanței.

Testarea funcțională este procesul de evaluare a comportamentului unei aplicații pentru a determina dacă toate caracteristicile dezvoltate se comportă conform așteptărilor. Pentru ca produsul să funcționeze corect, toate procesele trebuie să funcționeze conform unor cerințe prestabilite. Testarea funcțională poate fi efectuată folosind scripturi de testare pre-pregătite sau metode de testare exploratorii.

Testarea compatibilității este procesul de evaluare a comportamentului unei aplicații în diferite browsere, sisteme de operare și dispozitive cu rezoluții diferite de ecran. Verificarea compatibilității produsului cu toate cele mai recente versiuni de Firefox, Microsoft Edge, Google Chrome este un exemplu de acest tip de testare.

Testarea performanței este un set de verificări care vizează determinarea limitelor de performanță ale unei aplicații.

6.1.1 Testarea funcțională a aplicației web

Cea mai dificilă, dar și totodată complexă testare a aplicației grafice o reprezintă testarea funcțională. Pentru a asigura un grad înalt de testabilitate, este necesar de a simula parcurgerea tuturor locațiilor din baza de date a aplicației. Pentru a executa acest lucru, va fi necesară crearea unei aplicații mobile care să utilizeze controlerele REST pentru a simula ieșirile senzorilor.

În .NET, cea mai potrivită platformă de creare a aplicațiilor mobile este Xamarin.Forms.

Etapele de proiectare a aplicației mobile nu diferă prea mult de cea web. O excepție fiind controlerele. Varianta mobilă nu are acces direct către contextul bazei de date, cum a fost la aplicația web. Astfel, va fi necesar de a utiliza explicit controlere de tip REST, similare după logică celor din aplicația fizică a proiectului.

6.1.1.1 Funcționalitatea de randomizare

Pentru a simula parcurgerea locațiilor unei clădiri cu etaje, trebuie stabilite câteva scenarii de acțiune a potențialilor "clienți,,, fiecare având câte un grad de probabilitate de executare stabilit explicit în aplicație:

1. Un client intră în clădire. Pentru a fi posibil acest lucru, sunt găsite toate punctele de acces în clădire, se alege una aleatoriu. Tocmai a fost simulat pin-ul care răspunde de intrarea în clădire. Însă acest pas nu este suficient din cauza algoritmului de numărare a clienților din fiecare locație. Acesta constă dintr-o formulă comună pentru toate locațiile. Numărul total este egal cu numărul de oameni care au intrat în acea locație minus numărul care părăsit incinta. Însă există o problemă când vine vorba de holuri. Numărul de pe un hol este epuizat nu doar de către ieșirile acestuia, dar și de locațiile de pe acel nivel. Pentru a nu crea încă un algoritm separat de calcul pentru holuri, a fost luată decizia de a introduce în baza de date a unor puncte de acces virtuale, cum sunt cele din figura 5.9. La momentul în care un om iese de pe hol sau intră pe hol în

favoarea unei locații de pe acel nivel, se mai transmite către server încă o înregistrare, doar că virtuală, care să determine că holul tocmai a fost părăsit. Într-o asemenea manieră, dacă ar fi să calculăm numărul de oameni care au părăsit unul din holuri, se vor scădea senzorii punctelor de acces obișnuite, iar în plus în ecuație vor fi considerate și intrările/ieșirile virtuale.

	access_point_id	name	location_id	access_type_id
1	108	Common Entry Lvl 1	55	4
2	110	Common Entry Lvl 2	78	4
3	112	Common Entry Lvl -1	10	4
4	114	Common Entry Lvl 0	32	4

Figura 5.9: Punctele de acces virtuale pentru holuri din baza de date

Pentru a spori dinamismul acestei secțiuni, după intrarea în clădire, adică fiind pe hol, oaspetele va alege aleatoriu una din locațiile de pe același nivel și o va accesa, părăsind holul. Dar de fapt, există o șansă ca locația aleasă să fie însuși holul, în asemenea caz, oaspetele rămâne pe loc.

2. Schimbă locația de pe același nivel. Se caută nivelele cu oaspeți și se alege unul aleatoriu. Pe nivelul respectiv se caută locațiile populate, se alege o locație aleatoare, se alege un punct de acces aleatoriu al acestei locații și se părăsește incinta. Apoi pe același nivel, în mod analogic se alege un punct de acces și un client accesează locația.
3. Schimbă nivelul, schimbă locația. Acest scenariu este similar cu precedentul, doar că de data aceasta trebuie de luat în considerație faptul că oaspetele va părăsi holul pe ieșirea sa reală, nu una virtuală și va intra pe holul altui nivel într-un mod analogic. Deci la alegerea punctelor de acces a holurilor altui nivel, cele virtuale vor fi excluse.
4. Părăsește clădirea. Se alege o locație aleatoare a unui nivel oarecare populat și dacă nu este pe holul care duce direct spre ieșire, clientul va părăsi holul de pe acel nivel, va accesa holul nivelului cu ieșirea din clădire, va alege una din ieșiri și va părăsi incinta.

Aplicând aceste scenarii de testare, au fost depistate și câteva vulnerabilități a codului. După cum a fost menționat și anterior, formula de calculare a numărului de oameni nu asigura o funcționalitate completă. Astfel, formula a rămas neschimbată, însă au fost executate câteva modificări în baza de date. Rolul simulării a avut un mare impact asupra testării și a celorlalte formule: de calculare a ratei de ocupare a clădirii sau a locațiilor, a orei de vârf etc.

Chiar și dacă am presupune că nu ar fi fost găsite vulnerabilități în urma testării, acest lucru nu înseamnă că testarea a fost făcută rău sau invers, că aplicația ar avea un algoritm ideal. Fiecare testare își are scopurile sale, însă pentru a ridica siguranța optimalității, este necesară de a efectua o testare pe aceeași componentă, însă sub un alt unghi, adică utilizând altă metodă.

6.1.1.2 Funcționalitatea de transmitere a datelor manual

Trimiterea randomizată a datelor permite simularea diferitelor scenarii de parcurgere a locațiilor unui spațiu public. Dar mai există o metodă de testare a aplicației, cea de simulare manuală a senzorilor fiecărui punct de acces în parte. A fost necesară crearea unei interfețe grafice mobile și de a găsi o metodă ca utilizatorul să poată transmite date către locații într-un mod cât de cât similar celui real. Decizia luată a fost crearea a două pagini grafice, una pentru punctele de acces cu un singur sens și alta pentru cele cu sens dublu. Pentru desen, a fost utilizat instrumentul Skia Sharp, principiul paginilor fiind următorul: în cazul punctelor de acces cu un singur sens, după cum este prezentat în figura 5.10(*a*), pe ecran va apărea o singură linie vizibilă, care de fapt va fi analogia liniei luminoase a diodei din sistemul fizic. La momentul în care utilizatorul va glisa cu un deget pe ecran, sau cu două, la parcurgerea acelei linii de două ori se va lua în considerație în locația respectivă tocmai a avut loc o acțiune.

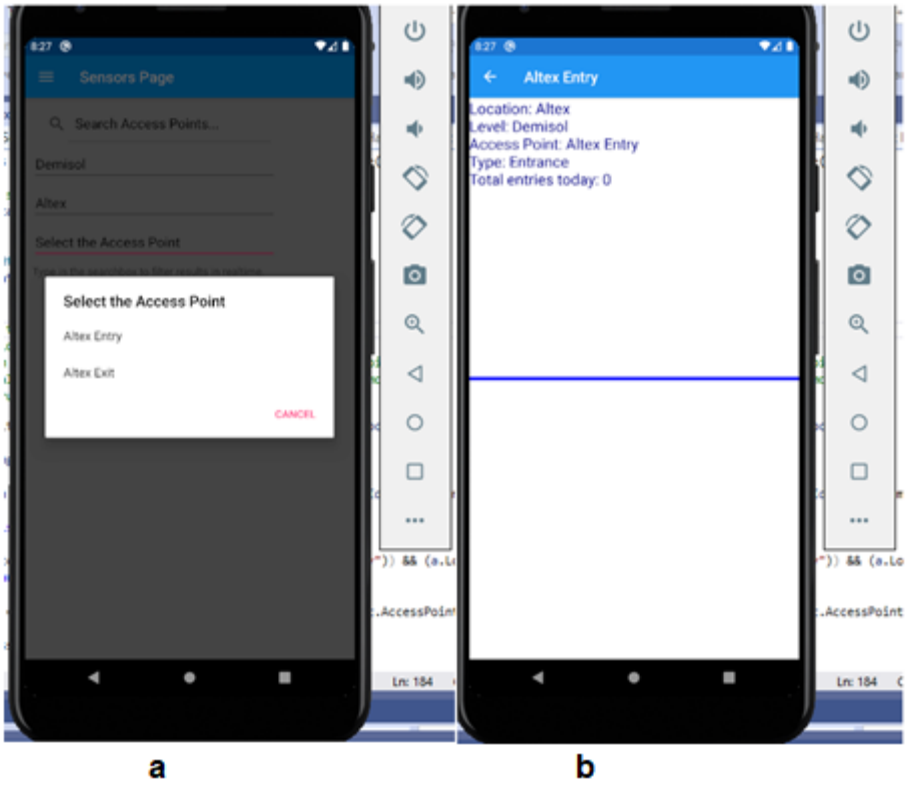

Figura 5.10: Pașii de accesare a simulării unui punct de acces concret
a - alegerea punctului de acces; b – interfața simulării de senzor

Dacă la senzorii cu o singură direcție sensul de parcurgere nu avea importantă, deoarece acțiunea executată era de fiecare dată aceeași, sau intrare, sau ieșire, la simularea senzorului dublu direcția de manipulare are sens. După cum a fost indicat în logica sistemului fizic, primul senzor întrerupt este cel care dictează sensul de parcurgere a unei locații. Astfel, la acest tip de puncte de acces, vor exista două puncte de acces, respectiv două linii desenate în interfața grafică (figura 5.11). Culoarea liniilor vor sugera sensul de parcurgere, dacă prima va fi întreruptă linia roșie, atunci sensul de parcurgere va însemna ieșirea locației, și invers.

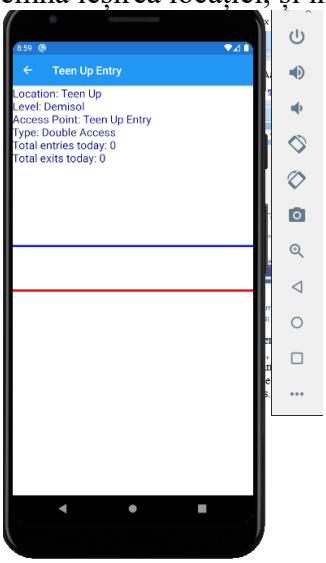

Figura 5.11: Simularea unui punct de acces cu sens dublu

După fiecare parcurgere a unui punct de acces cu succes, utilizatorul va primi o notificare în partea de jos a ecranului, iar în antet se va observa că datele s-au împrospătat, iar acțiunea realizată a fost introdusă în baza de date cu succes(figura 5.12).

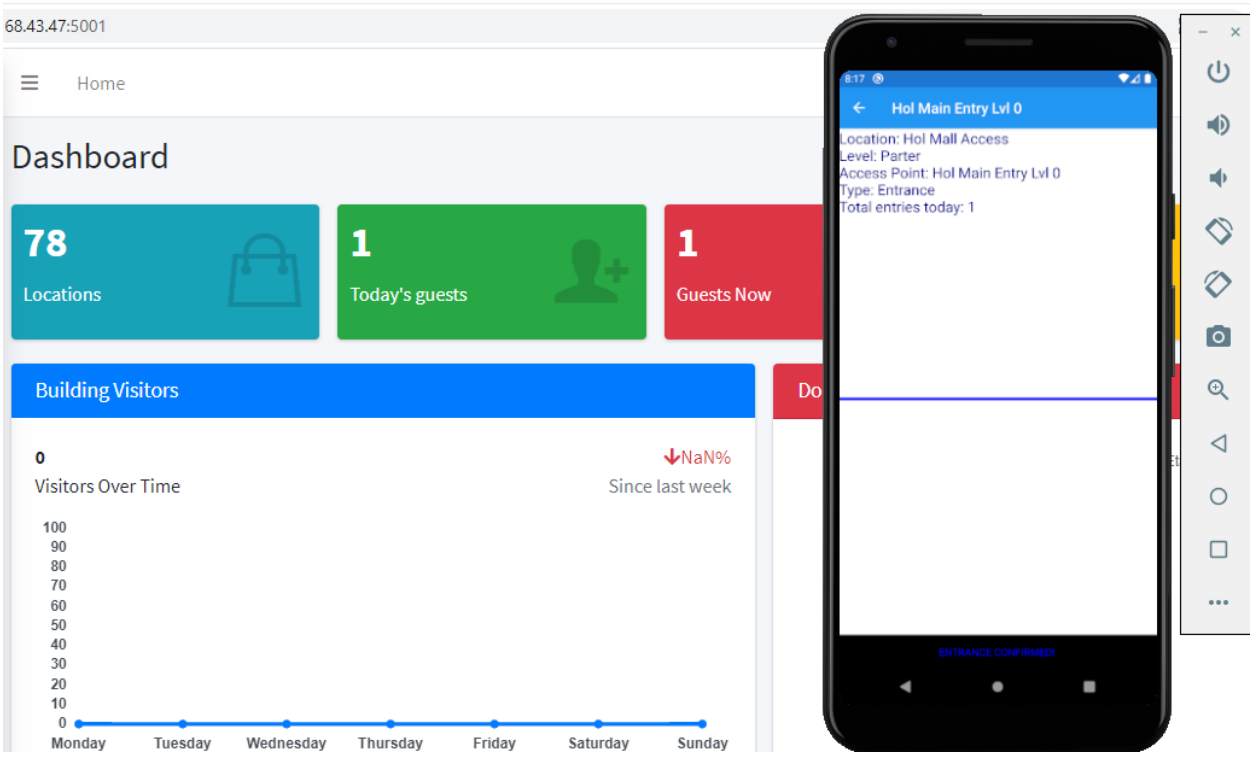

Figura 5.12: Demonstrația trimiterii unei simulări de intrare în clădire prin punctul de acces a holului principal.

Eficiența testării individuale a fiecărei locații permite găsirea unor vulnerabilități de cod la nivel de locație. Spre exemplu, în timpul testării s-a depistat faptul că magazinul *H&M,* care este amplasat pe două nivele în Iulius Mall și are pe fiecare nivel câte un punct de acces, avea scris în baza de date două puncte de acces cu același nume "*H&M,,,* în loc să fie etichetate suplimentar conform nivelului pe care se află, spre exemplu "*H&M Nivel 1,,*. Din cauza acestei probleme, la momentul în care din aplicația mobilă era selectat un punct de acces al respectivului magazin, aceasta se bloca. Cu o depanare rapidă problema a fost rezolvată instant prin modificarea în baza de date a numelui punctelor de acces.

Majoritatea erorilor de testare au fost depistate în baza de date, deoarece fiind dedicată unui spațiu public de capacitate mare, există o probabilitate ca factorul uman să greșească la un moment dat populând o asemenea cantitate mare de informații. Însă într-un asemenea sistem, greșelile nu pot fi permisibile, deoarece sigur vor aduce la erori de monitorizare sau chiar vor dăuna întregului sistem.

6.2 Testarea aplicației fizice

Pentru a executa acest tip de testare, în primul rând ne vom asigura că avem la îndemână o plăcuță de dezvoltare Node MCU V3, un senzor receptor de diodă și respectiv o sursă de lumină. Descrierea completă a echipamentelor poate fi găsită în capitolul 3, tema "*Compoziția sistemului de monitorizare,,*.

Un alt pas obligatoriu este introducerea în baza de date informația corespunzătoare plăcuței.

	wireless_module_id	name	nr_of_data_pins	factory_guid
1	4	Node MCU V3 - LoLin (Wi-Fi)	5	2388676

Figura 5.13: Introducerea în tabelul din baza de date *wireless_module* a plăcuței cu id-ul din fabric 2388676

Maparea pinilor se execută în tabelul *module_pin (*figura 5.14*)*. Datele se introduc cu mare grijă, deoarece pinii fizici vor fi configurați exclusiv pe baza acestor informații. Este necesar de a puncta faptul că numele pin-ului joacă și el un rol important. De facto, numele

pinilor va fi preluat de către programul plăcuței și datorită funcției Arduino *pinMode(număr_pin, mod),* aceștia vor fi configurați pentru a putea primi datele de la senzori.

	pin_assigned_id	name	wireless_module_id	access_point_id	pair_id	passing_type_id
1	16	16	4	NULL	NULL	NULL
2	17	5	4	13	18	1
3	18	4	4	13	17	2
4	19	0	4	13	NULL	NULL
5	20	2	4	16	NULL	2

Figura 5.14: Maparea pinilor în tabelul *module_pin*

Pentru testarea fizică vom conecta senzorii la porturile D1 D2, care au etichetele GPIO5 și respectiv GPIO4[13]. După cum observăm în figura 5.14 există coeficienții celor doi pini GPIO în coloana *name.*

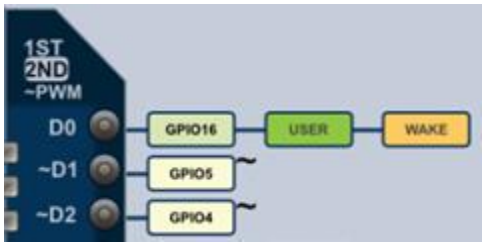

Figura 5.15: Notarea pinilor pe plăcuța de dezvoltare Node MCU V3

Conectarea fizică fiind realizată cu succes, pe interfața serială vor apărea mesajele de confirmare a conectării în rețeaua locală, găsirea perechilor de pini în caz că există. La momentul în care unul punctul de acces va fi parcurs, pe serială va apărea un mesaj cu informația aferentă ultimelor tranzacții. În figura 5.16 este prezentat un exemplu în care un oaspete va intra în locație, iar apoi va ieși.

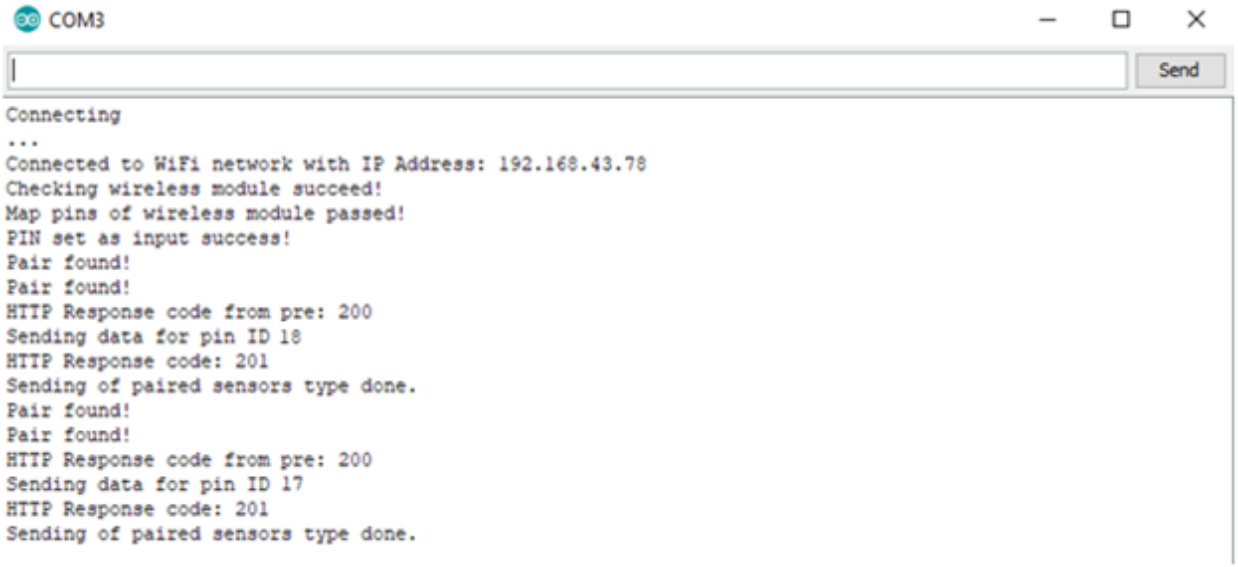

Figura 5.16: Afișajul pe interfața serială

Pentru a verifica dacă într-adevăr s-au stocat datele, vom utiliza controlerul de la adresa HTTP "*http://192.168.43.47:5001/api/SensorOutcomes/Today*,,, creat cu intenția de a obține datele trimise de către senzori în ziua curentă. În figura 5.17 vom putea observa aceleași id-uri

[13] GPIO sau **g**eneral-**p**urpose **i**nput/**o**utput este un pin de semnal digital pe un circuit integrat sau pe o placă de circuite electronice care poate fi utilizat ca intrare sau ieșire, sau ambele, și care poate fi controlat de utilizator în timpul rulării. A se vedea https://en.wikipedia.org/wiki/General-purpose_input/output, vizitat la 4 iulie 2021.

indicate și pe interfața serială, însă mai există un detaliu minor, care îi va scăpa unui observator ce nu are cunoștințele necesare. Pentru fiecare pin, se observă în figura 5.17 că aproape instant s-a mai declanșat încă un pin, care nu aparține plăcuței respective de dezvoltare


```
{
  "sensorOutcomeId": 159921,
  "timeStamp": "2021-07-04T10:08:21.03",
  "pinAssignedId": 17,
  "pinAssigned": null
},
{
  "sensorOutcomeId": 159922,
  "timeStamp": "2021-07-04T10:11:43.74",
  "pinAssignedId": 112,
  "pinAssigned": null
},
{
  "sensorOutcomeId": 159923,
  "timeStamp": "2021-07-04T10:11:43.893",
  "pinAssignedId": 19,
  "pinAssigned": null
},
{
  "sensorOutcomeId": 159924,
  "timeStamp": "2021-07-04T10:11:43.957",
  "pinAssignedId": 113,
  "pinAssigned": null
},
```


Figura 5.17: Datele primate de la senzorii sistemului fizic

Dacă vom căuta în tabelul *module_pin* însemnătatea acestor pini (figura 5.18), vom observa că aceștia aparțin unui punct de acces virtual. Într-un asemenea mod se remarcă faptul că un client tocmai a ieșit dintr-o incintă și a intrat pe hol, sau invers, că a intrat într-o locație părăsind holul.

	pin_assigned_id	name	wireless_module_id	access_point_id	pair_id	passing_type_id
1	112	Virt. pin IN lvl -1	101	112	NULL	1
2	113	Virt. pin OUT lvl -1	101	112	NULL	2

Figura 5.18: Pinii virtuali utilizați pentru a marca ieșirea de pe hol în favoarea unei locații și intrarea pe hol ieșind dintr-o locație

Așteptările nu corespund întotdeauna realității. Din această cauză necesitatea testării aplicației fizice este evidentă și necesară, chiar obligatorie. Pentru a obține un mode de lucru optimal, aceasta trebuie testată în diferite condiții, sub diferite scenarii. Deși în teorie programul ar părea optimizat, nu trebuie să uităm faptul că lucrăm cu senzori care cu ușurință pot fi perturbați de către factorii mediului. Testarea în sine este un proces foarte important care scoate la iveală diverse probleme fie la nivel fizic, fie la nivel logic, fie la nivel grafic.

Capitolul 7. Concluzii și direcții viitoare de dezvoltare

7.1 Concluzii

Fiind un sistem real de monitorizare, acest proiect a înglobat în sine foarte multe componente și principii de organizare. Toate succesele, dar mai ales eșecurile au stat la baza creării unei aplicații stabile care să poată fi capabilă să înregistreze numărul de oameni dintr-o clădire de capacitate mare.

Primii pași probabili au fost cei mai grei. Până a ajunge la un plan de monitorizare funcțional, au fost testate și probate mai multe variante alternative, fiecare având câte un avantaj. Însă în practică acele plusuri deveneau inutile. După un anumit număr de încercări, totuși a fost considerată drept cea mai simplă, dar și optimală, varianta de optimizare a spațiilor publice cu ajutorul senzorilor optoelectronici. Selectând echipamentele necesare, pe baza structurii acestora se putea crea mediul de programare.

În alegerea platformei de programare a fost depusă destul de multă perseverență. Alegerea mediului corect și compatibil cu echipamentele fizice garantează câștigul multor ore de muncă. Astfel a fost ales limbajul care înglobează în sine un număr foarte mare de funcționalități, librării, concepte, este C#.

Gândirea și realizarea bazei de date a fost a doua cea mai mare problemă, după găsirea unei soluții optime de monitorizare. Aceasta trebuia să asigure o funcționalitate completă pentru utilizator și mai ales pentru programator. Anume baza de date trebuie să fie cea compatibilă cu sistemul fizic. Altfel partea de cod va părea un haos, care va tinde cu orice preț să compenseze imperfecțiunile din baza de date, prin utilizarea diverselor funcționalități de reglare. Completarea datelor în baza de date deși pare un lucru care nu necesită prea multă cunoaștere, nu este așa. Fiind un sistem de tip real, fiecare greșeală fie că e în baza de date, fie că e în aplicația codului, își pune amprenta pe instabilitatea sistemului. Neglijarea creării unei baze de date corect structurate, dar și cu atenția populată, poate aduce la erori greu de perceput în sistem.

Găsirea unui mod cât mai eficient de structurare a aplicației, este și aceasta o etapă importantă. Alegerea arhitecturii pe trei niveluri a stat la baza unui cod curat, bine stabilit și optimizat.

Realizând aplicația, optimalitatea acesteia nu putea fi asigurată până nu a fost testată. Astfel au fost necesare găsirea unor strategii de testare a sistemului real. Pentru acest lucru a fost gândită aplicația mobilă, care să poate fi conectată la rețea și să simuleze datele trimise către senzori. Testarea fizică a fost la fel de importantă, de fapt, anume acest tip de testare ne poate asigura dacă aplicația creată poate fi pusă în practică. Toate etapele create pas cu pas, în ansamblu au putut fi diagnosticate prin intermediul testării fizice.

Este evident faptul că această aplicație poate și trebuie să fie îmbunătățită. Scopul acestui proiect fiind găsirea și demonstrarea modului de abordare a unei monitorizări în timp real. În principiu, un sistem ce rulează în timp real mereu necesită îmbunătățiri. Lumea este în dezvoltare continuă, apar noi tehnologii, apar noi probleme.

7.2 Direcții viitoare de dezvoltare

Soluțiile de optimizare pot fi aplicate fie asupra sistemului fizic, aplicând careva corecții, fie asupra codului care stă în spatele proiectului. Îmbunătățirile unui sistem este un proces continuu, care are drept scop eliminarea unor erori, optimizarea lucrului componentelor, găsirea a noi soluții și idei de abordare.

Astfel, în cadrul proiectului respectiv pot fi efectuate îmbunătățiri la toate nivelele, începând cu cel fizic și terminând cu interfața grafică.

Sistemul de securizare este unul dintre cele mai importante aspecte ale unui sistem. Deși lucrul senzorilor în rețea este securizat de către certificarea Wi-Fi WPA, este necesară de a adăuga nivele suplimentare de securizare. Spre exemplu odată conectat la rețea, senzorul ar

trebui să trimită către server niște certificate de autorizare. Serverul le va procesa și în cazul unui răspuns pozitiv, sistemul fizic respectiv să fie autorizat.

Un alt aspect de viitor ar fi adăugarea unui algoritm de inteligență artificială care pe baza studierii orelor de activitate, va seta senzorii în regim de lucru pasiv, pentru a nu consuma energie electrică excesiv.

Dacă ar fi să abordăm o îmbunătățire mai radicală, atunci am alege să schimbăm senzorii optoelectronic pe un sistem de monitorizare video cu inteligență artificială integrată. Astfel, se va putea face diferența dintre maturi și copii, dintre femei și bărbați. La fel nu va mai fi necesar de a include personalul clădirii în lista de oaspeți care parcurg locațiile.

La nivel de arhitectură, pentru a asigura un grad mai înalt de securizare, va fi necesar de a separa în module diferite interfața grafică de cea logică.

La nivel de cod, cea mai mare dar și necesară modificare ar fi utilizarea cât mai des posibil a interogărilor de tip SQL cu ajutorul Linq sau dacă va fi separată partea grafică de cea logică, să fie create noi controlere care să implementeze o logică similară. La momentul actual, dacă este nevoie de a calcula numărul de oameni dintr-o locație, se stochează în program toate datele de la senzori din acea zi, iar apoi se parcurge fiecare înregistrare și se verifică. Un asemenea proces durează foarte mult. Dar cu ajutorul interogărilor SQL, numărul de oameni dintr-o locație poate fi determinat din doi pași (primul este obținerea numărului de intrări în locație și al doilea numărul de ieșiri) sau chiar un pas. Cunoașterea funcționalităților SQL a demonstrat că viteza aplicației poate fi optimizată de câteva ori.

Pentru aplicația web ar fi preferabil de a adăuga și un panou de autentificare. Deși doar administratorul are posibilitatea de a se conecta la rețeaua securizată, un nivel suplimentar de securizare niciodată nu dăunează sistemului. Ca și remarcă pe partea grafică, ar fi un plus de a crea și o aplicație mobilă care să permită administratorului să vizualizeze datele în timp real. În starea actuală a aplicației, administratorul primește notificări instantanee, însă ar fi o idee foarte bună de a crea un meniu separat de notificări, care să afișeze data și ora declanșării anumitor alarme. Înștiințarea administratorului ar putea fi făcută în mod automat pe email, sau pe telefonul mobil, într-un format de raport care să conțină toate datele necesare.

Sistemul actual poate fi lărgit și de a adăuga componente fizice noi. La momentul în care una din locații are un grad prea mare de oameni, pe un panou din acea locație să apară o avertizare vizibilă pentru toată lumea.

Pentru utilizatorii obișnuiți care vizitează locația, s-ar putea de creat o aplicație mobilă care să îi informeze pe aceștia care este perioada optimală de vizitare a acesteia. Interfața cu utilizatorul simplu ar putea fi destul de dinamizată. S-ar putea crea o hartă a clădirii și să permită utilizatorilor de a găsi locații și de a urma drumul spre ele din poziția curentă în timp real.

După cum a fost menționat anterior, proiectele care vizează monitorizarea în timp real mereu vor crea oportunități pentru dezvoltare și îmbunătățire continuă. Întrebarea este… cât de departe vor ajunge?

Bibliografie

[1] Ron Synovitz, Andrei Luca Popescu, Pandemia și viața privată. Tehnologia de urmărire a coronavirusului a declanșat bătălia pe datele personale [Online], Disponibil la adresa: https://romania.europalibera.org/a/pandemia-viata-privata-tehnologia-urmarire-coronavirus-date-personale/30626102.html, Accesat 29 iunie 2021.

[2] Necunoscut, Axper Analytics [Online], Disponibil la adresa https://axper.com/wp-content/themes/axper_responsive/pdf/Software-mall-EN.pdf, Accesat 29 iunie 2021.

[3] Necunoscut, Web Service Development and Composition [Online], Disponibil la adresa: http://www.ebtps.com/development/web-service-development-composition.html, Accesat 30 iunie 2021.

[4] Olteanu Ana-Cristina, Arhitectura serviciilor web [Online], Disponibil la adresa http://webcache.googleusercontent.com/search?q=cache:MXr9e3ChBB4J:stst.elia.pub.ro/ric/teme_ric_2008_9/anaolteanu/proiectfinal.doc+&cd=6&hl=ro&ct=clnk&gl=ro, Accesat 29 iunie 2021.

[5] Necunoscut, Single-Tier Vs. Multi-Tier Architecture: Choosing The Right Bitnami Package [Online], Disponibil la adresa: https://docs.bitnami.com/azure-templates/singletier-vs-multitier/, Accesat 1 iulie 2021.

[6] Necunoscut, How to define a RESTful Web Services [Online], Disponibil la adresa: https://www.jobtrack.com.au/site/JT/blog/article/rest-define/, Accesat 30 iunie 2021.

[7] Necunoscut, Correctly distinguish the difference and connection between API & REST API & RESTful API & Web Service [Online] Disponibil la adresa: https://developpaper.com/correctly-distinguish-the-difference-and-connection-between-api-rest-api-restful-api-web-service/, Accesat 29 iunie 2021.

[8] Necunoscut, Introduction to ASP.NET Core [Online], Disponibil la adresa: http://www.corevaluetechnologies.com/blog/introduction-aspnet-core, Accesat 30 iunie 2021.

[9] Necunocut, Native Mobile Platform Breakdown: A Guide to Xamarin, iOS, and Android [Online], Disponibil la adresa: https://www.grapecity.com/blogs/native-mobile-platform-breakdown-a-guide-to-xamarin-ios-and-android/, Accesat 30 iunie 2021.

[10] Necunoscut, Necunoscut [Online], Disponibil la adresa https://i2.wp.com/restfulapi.net/wp-content/uploads/rest-arch.jpg?fit=782%2C670&ssl=1, Accesat 1 iulie 2021.

[11] Necunoscut, Receptor pentru Dioda Laser [Online], Disponibil la adresa: https://ardushop.ro/ro/home/895-receptor-pentru-dioda-laser.html, Accesat 3 iulie 2021.

[12] Necunoscut, Modul diodă laser (roșu) 5mW [Online], Disponibil la adresa: https://ardushop.ro/ro/electronica/262-modul-dioda-laser-rou-5mw.html?search_query=dioda&results=22, Accesat 3 iulie 2021.

[13] Necunoscut, Node MCU V3 - LoLin (Wi-Fi) [Online], Disponibil la adresa: https://ardushop.ro/ro/home/108-node-mcu.html?search_query=esp8266&results=12, Accesat 3 iulie 2021.

[14] Necunoscut, Got poor #WiFi? Its the walls! The signal through one concrete block wall is ok but not two. Here's a common scenario that doesn't work. [Online], Disponibil la adresa: https://twitter.com/ntsprk/status/926084826322669569/photo/1, Accesat 3 iulie 2021.

[15] Necunoscut, Position Sensors – Choosing the right sensor [Online], Disponibil la adresa https://www.celeramotion.com/zettlex/support/technical-papers/position-sensors/, Accesat 3 iulie 2021.

Anexe

Anexa 1: Codul sursă al sistemului fizic

```
#include <ArduinoJson.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include "Wire.h"

#define HTTP_REQUEST_OK 200
#define HTTP_REQUEST_INIT - 1
#define RETURN_RESPONSE_OK 1
#define RETURN_RESPONSE_NOK 0
#define DEBOUNCE_DELAY 1000 //120 ms
const char * ssid = "BuildingNetwork";
const char * password = "VeryStrongPassword";
uint32_t chipId = 0;
const char * postPinController =
"http://192.168.43.47:5001/api/SensorOutcomes";
String wirelessModulePayload;
unsigned long lastTime = 0;
unsigned char setupReady = 0;

/*structure for extracting data with HTTP call GET */
struct ModulePins {
  int pinAssignedId;
  const char * name;
  int wirelessModuleId;
  int accessPointId; /* can be NULL*/
  int pairId;
  /*-----*/
  unsigned char actualSensorState; /* Used to record the signal edge */
  unsigned char lastSensorState;
  unsigned long millisPrevious;
}* pinsMapping;

struct WirelessModule {
  int wirelessModuleId;
  const char * name;
  int nrOfDataPins;
  int factoryGuid;
} localWirelessModule;

char * concat(const char * s1,
              const char * s2) {
  char * result = (char * ) malloc((strlen(s1) + strlen(s2) + 100) *
sizeof(char)); // +1 for the null-terminator
  if (result == NULL) {
    printf("Out of memory\n");
    exit(-1);
  }

  strcpy(result, s1);
  strcat(result, s2);

  return result;
}

char * uint32ToString(uint32_t value) {
  char * str = (char * ) malloc(sizeof(char) * 11); /*11 bytes = 10 for the
digits and 1 for the null character */
```

```
  //snprintf(str, sizeof str, "%lu", (unsigned long)value);
  sprintf(str, "%lu", value);

  return str;
}

String httpGETRequest(const char * servName) {
  HTTPClient http;

  // Your IP address with path or Domain name with URL path
  http.begin(servName);

  // Send HTTP POST request
  int httpResponseCode = http.GET();

  String payload = "{}";

  if (httpResponseCode >= 200 && httpResponseCode <= 299) {
    payload = http.getString();
  } else {
    /*Do nothing */
  }

  // Free resources
  http.end();

  return payload;
}

unsigned char checkWirelessModule() {
  HTTPClient http;
  unsigned char moduleExists = 0;
  const char * moduleGuid;
  const char * httpGetModReqByGuid =
"http://192.168.43.47:5001/api/WirelessModules/Guid/";
  const char * requestDestination;

  moduleGuid = uint32ToString(chipId);
  requestDestination = concat(httpGetModReqByGuid, moduleGuid);
  wirelessModulePayload = httpGETRequest(requestDestination);
  //StaticJsonDocument allocates memory on the stack
  //The size of the pool in bytes was consulted with
https://arduinojson.org/v6/assistant/
  StaticJsonDocument<16384> doc;

  DeserializationError error = deserializeJson(doc, wirelessModulePayload);
  //Test if parsing succeeds
  if (error) {
    return RETURN_RESPONSE_NOK;
  }

  localWirelessModule.wirelessModuleId = doc["wirelessModuleId"];
  localWirelessModule.name = doc["name"];
  localWirelessModule.nrOfDataPins = doc["nrOfDataPins"];
  localWirelessModule.factoryGuid = doc["factoryGuid"];
  moduleExists = RETURN_RESPONSE_OK;
  Serial.println("Checking wireless module succeed!");
  return moduleExists;
}

void initPinsMappingData()
{
  unsigned char parsingIndex = 0u;
```

```
  for (parsingIndex = 0; parsingIndex < localWirelessModule.nrOfDataPins;
parsingIndex++) {
    pinsMapping[parsingIndex].pinAssignedId = 0;
    pinsMapping[parsingIndex].name = "";
    pinsMapping[parsingIndex].wirelessModuleId = 0;
    pinsMapping[parsingIndex].accessPointId = 0;
  }
}

unsigned char mapPinsByWirelessModule() {
  HTTPClient http;
  unsigned char pinsMapped = 0u;
  const char * httpGetModReqByGuid =
"http://192.168.43.47:5001/api/ModulePins/WirelessModule/";
  const char * requestDestination;
  const char * moduleId;
  unsigned char parsingIndex = 0u;

  moduleId = uint32ToString((unsigned long)
localWirelessModule.wirelessModuleId);
  pinsMapping = (ModulePins * ) malloc(localWirelessModule.nrOfDataPins *
sizeof(ModulePins));

  initPinsMappingData();

  requestDestination = concat(httpGetModReqByGuid, moduleId);

  wirelessModulePayload = httpGETRequest(requestDestination);

  //StaticJsonDocument allocates memory on the stack
  //The size of the pool in bytes was consulted with arduino.org/assistant
  StaticJsonDocument < 1536 > doc;
  DeserializationError error = deserializeJson(doc, wirelessModulePayload);

  //Test if parsing succeeds
  if (error) {
    return RETURN_RESPONSE_NOK;
  }

  JsonArray arr = doc.as <JsonArray>();

  for (JsonObject repo : arr) {
    pinsMapping[parsingIndex].pinAssignedId = repo["pinAssignedId"];
    pinsMapping[parsingIndex].name = repo["name"];
    pinsMapping[parsingIndex].wirelessModuleId = repo["wirelessModuleId"];
    pinsMapping[parsingIndex].accessPointId = repo["accessPointId"];
    if (repo["pairId"] != NULL)
      pinsMapping[parsingIndex].pairId = repo["pairId"];
    parsingIndex = parsingIndex + 1;
  }

  pinsMapped = RETURN_RESPONSE_OK;
  Serial.println("Map pins of wireless module passed!");
  return pinsMapped;
}

/* This function is used in order to prepare the "virtual" hallway for the
user that parse a location.
   E.g. if a user is leaving a location, he is entering the hallway. For the
leaving location we use
   the sendHttpPost function, but for hallway we also must handle a function.
That's why we use preHttpPost. */
```

```
void preHttpPost(const char * pinAssignedId)
{
  HTTPClient http;
  char * postData;
  const char * hallPostController =
"http://192.168.43.47:5001/api/SensorOutcomes/ManHall";
  postData = concat("{\"pinAssignedId\":", pinAssignedId);
  postData = concat(postData, "}");

  http.begin(hallPostController);

  http.addHeader("Content-Type", "application/json");
  int httpResponseCode = http.POST(postData);
  Serial.print("HTTP Response code from pre: ");
  Serial.println(httpResponseCode);

  // Free resources
  http.end();
}
void sendHttpPost(const char * pinAssignedId) {
  HTTPClient http;
  char * postData;
  postData = concat("{\"pinAssignedId\":", pinAssignedId);
  postData = concat(postData, "}");

  // The Domain name with URL path or IP address with path
  http.begin(postPinController);
  http.addHeader("Content-Type", "application/json");

  Serial.print("Sending data for pin ID ");
  Serial.println(pinAssignedId);
  int httpResponseCode = http.POST(postData);
  Serial.print("HTTP Response code: ");
  Serial.println(httpResponseCode);

  // Free resources
  http.end();
}

int convertFromCharToInt(const char * data) {
  return atoi(data);
}

void setPinsAsInput() {
  unsigned char parsingIndex = 0u;
  int pinInput;

  for (parsingIndex = 0; parsingIndex < localWirelessModule.nrOfDataPins;
parsingIndex++) {
    pinInput = convertFromCharToInt(pinsMapping[parsingIndex].name);
    pinMode(pinInput, INPUT);
  }
  Serial.println("PIN set as input success!");
}

ModulePins * checkPairExists(int pairId) {
  for (int parsingIndex = 0; parsingIndex < localWirelessModule.nrOfDataPins;
parsingIndex++) {
    if (pinsMapping[parsingIndex].pinAssignedId == pairId)
    {
      Serial.println("Pair found!");
      return &pinsMapping[parsingIndex];
    }
```

```
  }

  return NULL;
}
void setup() {
  Serial.begin(115200);

  WiFi.begin(ssid, password);
  Serial.println("Connecting");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.print("Connected to WiFi network with IP Address: ");
  Serial.println(WiFi.localIP());

  chipId = ESP.getChipId();
  /*Check if connected device can be found in Data Base */
  if (checkWirelessModule()) {
    if (mapPinsByWirelessModule()) {
      // initialize the pins as input
      setPinsAsInput();
      setupReady = 1;
    }
  }
}

void loop() {
  //Check WiFi connection status
  if (WiFi.status() == WL_CONNECTED) {
    if (setupReady == 1) {
      unsigned long now = millis();
      int pinInput;

      ModulePins * pairPin;

      for (int parsingIndex = 0; parsingIndex <
localWirelessModule.nrOfDataPins; parsingIndex++) {
        pinInput = convertFromCharToInt(pinsMapping[parsingIndex].name);
        pinsMapping[parsingIndex].actualSensorState = digitalRead(pinInput);
/* read data*/
        if (now - pinsMapping[parsingIndex].millisPrevious > DEBOUNCE_DELAY)
{
          pinsMapping[parsingIndex].millisPrevious = 0;
          /* When the receiver get the laser radiation, the output will be on
HIGH.
             Then, when a guest is interrupting the sensor, the output is
LOW. */
          if ((pinsMapping[parsingIndex].actualSensorState == LOW) &&
              (pinsMapping[parsingIndex].lastSensorState == HIGH)) {
            pairPin = checkPairExists(pinsMapping[parsingIndex].pairId);
            if (pairPin) {
              /* check if the pair sensor is not triggered already */
              if (pairPin->millisPrevious) {
                /* The second (pair) sensor was triggered, that means we cant
send this data to the server. */
                char* assignedPin =
uint32ToString(pinsMapping[parsingIndex].pairId);

                preHttpPost(assignedPin);
                sendHttpPost(assignedPin);
```

```
                Serial.println("Sending of paired sensors type done.");
              }
            } else {
              char* assignedPin =
uint32ToString(pinsMapping[parsingIndex].pinAssignedId);
              sendHttpPost(assignedPin);
              Serial.println("Sending of single sensors type done.");
            }
            pinsMapping[parsingIndex].millisPrevious = now; /* Record the
time to calculate later the debounce. */
          }
        }
        if (pairPin) pairPin->millisPrevious = 0; // reset that pair was not
triggered
        pinsMapping[parsingIndex].lastSensorState =
pinsMapping[parsingIndex].actualSensorState;
      }
    }
    else {
      Serial.println("Setup failed");
    }
  }
  else {
    Serial.println("WiFi Disconnected");
  }
}
```

Anexa 2: Codul sursă al aplicației grafice

Controlerele

AccessPointsTableController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApp.Models;

namespace WebApp.Controllers.AdminDashboard
{
    [Route("[controller]")]
    public class AccessPointsTableController : Controller
    {
        IuliusMallDBContext _context = new IuliusMallDBContext();

        public IActionResult Index()
        {
            return View(_context.AccessPoints);
        }

        [HttpPost("[action]")]
        public ActionResult AddAccessPoint(AccessPoint accessPoint)
        {
            _context.AccessPoints.Add(accessPoint);
            _context.SaveChanges();

            return RedirectToAction("Index", "AccessPointsTable", "Add");
        }

        [HttpPost("[action]")]
        public ActionResult EditAccessPoint([FromForm] AccessPoint
accessPoint)
        {
            _context.Entry(accessPoint).State = EntityState.Modified;
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "AccessPointsTable", "Edit");
        }

        [HttpPost("[action]/{LocationToDeleteId}")]
        public ActionResult DeleteAccessPoint(int LocationToDeleteId)
        {
            var accessPoint = _context.AccessPoints.Where(a =>
a.AccessPointId == LocationToDeleteId).SingleOrDefault();
            _context.AccessPoints.Remove(accessPoint);
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "AccessPointsTable", "Delete");
        }

        [HttpGet("[action]/{id}")]
        public IActionResult GetAccessPointById(int id)
        {
            return new JsonResult(_context.AccessPoints.SingleOrDefault(b =>
b.AccessPointId == id));
        }
```

```
    }
}
```

LocationLevelsTableController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApp.Models;

namespace WebApp.Controllers.AdminDashboard
{
    [Route("[controller]")]
    public class LocationLevelsTableController : Controller
    {
        IuliusMallDBContext _context = new IuliusMallDBContext();

        public IActionResult Index()
        {
            return View(_context.LocationLevels);
        }

        [HttpPost("[action]")]
        public ActionResult AddLocationLevel(LocationLevel locationLevel)
        {
            _context.LocationLevels.Add(locationLevel);
            _context.SaveChanges();

            return RedirectToAction("Index", "LocationLevelsTable", "Add");
        }

        [HttpPost("[action]")]
        public ActionResult EditLocationLevel([FromForm] LocationLevel
locationLevel)
        {
            _context.Entry(locationLevel).State = EntityState.Modified;
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "LocationLevelsTable", "Edit");
        }

        [HttpPost("[action]/{locationLevelToDeleteId}")]
        public ActionResult DeleteLocationLevel(int locationLevelToDeleteId)
        {
            var locationLevel = _context.LocationLevels.Where(a =>
a.LocationLevelId == locationLevelToDeleteId).SingleOrDefault();
            _context.LocationLevels.Remove(locationLevel);
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "LocationLevelsTable",
"Delete");
        }

        [HttpGet("[action]/{id}")]
        public IActionResult GetLocationLevelById(int id)
        {
            return new JsonResult(_context.LocationLevels.SingleOrDefault(b
=> b.LocationLevelId == id));
```

```
        }

    }
}
```

LocationsTableController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApp.Models;

namespace WebApp.Controllers.AdminDashboard
{
    [Route("[controller]")]
    public class LocationsTableController : Controller
    {
        IuliusMallDBContext _context = new IuliusMallDBContext();

        public IActionResult Index()
        {
            return View(_context.Locations);
        }

        [HttpPost("[action]")]
        public ActionResult AddLocation(Location location)
        {
            _context.Locations.Add(location);
            _context.SaveChanges();

            return RedirectToAction("Index", "LocationsTable", "Add");
        }

        [HttpPost("[action]")]
        public ActionResult EditLocation([FromForm] Location location)
        {
            _context.Entry(location).State = EntityState.Modified;
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "LocationsTable", "Edit");
        }

        [HttpPost("[action]/{locationToDeleteId}")]
        public ActionResult DeleteLocation(int locationToDeleteId)
        {
            var location = _context.Locations.Where(a => a.LocationId ==
locationToDeleteId).SingleOrDefault();
            _context.Locations.Remove(location);
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "LocationsTable", "Delete");
        }

        [HttpGet("[action]/{id}")]
        public IActionResult GetLocationById(int id)
        {
            return new JsonResult(_context.Locations.SingleOrDefault(b =>
b.LocationId == id));
        }
```

```
    }
}
```

ModulePinsTableController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApp.Models;

namespace WebApp.Controllers.AdminDashboard
{
    [Route("[controller]")]
    public class ModulePinsTableController : Controller
    {
        IuliusMallDBContext _context = new IuliusMallDBContext();

        public IActionResult Index()
        {
            return View(_context.ModulePins);
        }

        [HttpPost("[action]")]
        public ActionResult AddModulePin(ModulePin modulePin)
        {
            _context.ModulePins.Add(modulePin);
            _context.SaveChanges();

            return RedirectToAction("Index", "ModulePinsTable", "Add");
        }

        [HttpPost("[action]")]
        public ActionResult EditModulePin([FromForm] ModulePin modulePin)
        {
            _context.Entry(modulePin).State = EntityState.Modified;
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "ModulePinsTable", "Edit");
        }

        [HttpPost("[action]/{pinToDeleteId}")]
        public ActionResult DeleteModulePin(int pinToDeleteId)
        {
            var modulePin = _context.ModulePins.Where(a => a.PinAssignedId ==
pinToDeleteId).SingleOrDefault();
            _context.ModulePins.Remove(modulePin);
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "ModulePinsTable", "Delete");
        }

        [HttpGet("[action]/{id}")]
        public IActionResult GetModulePinById(int id)
        {
            return new JsonResult(_context.ModulePins.SingleOrDefault(b =>
b.PinAssignedId == id));
        }
```

```
    }
}
```

WirelessModulesTableController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using WebApp.Models;

namespace WebApp.Controllers.AdminDashboard
{
    [Route("[controller]")]
    public class WirelessModulesTableController : Controller
    {
        IuliusMallDBContext _context = new IuliusMallDBContext();

        public IActionResult Index()
        {
            return View(_context.WirelessModules);
        }

        [HttpPost("[action]")]
        public ActionResult AddWirelessModule(WirelessModule wirelessModule)
        {
            _context.WirelessModules.Add(wirelessModule);
            _context.SaveChanges();

            return RedirectToAction("Index", "WirelessModulesTable", "Add");
        }

        [HttpPost("[action]")]
        public ActionResult EditWirelessModule([FromForm] WirelessModule
wirelessModule)
        {
            _context.Entry(wirelessModule).State = EntityState.Modified;
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "WirelessModulesTable", "Edit");
        }

        [HttpPost("[action]/{wirelessModuleToDeleteId}")]
        public ActionResult DeleteWirelessModule(int
wirelessModuleToDeleteId)
        {
            var wirelessModule = _context.WirelessModules.Where(a =>
a.WirelessModuleId == wirelessModuleToDeleteId).SingleOrDefault();
            _context.WirelessModules.Remove(wirelessModule);
            _context.SaveChangesAsync();

            return RedirectToAction("Index", "WirelessModulesTable",
"Delete");
        }

        [HttpGet("[action]/{id}")]
        public IActionResult GetWirelessModuleById(int id)
        {
            return new JsonResult(_context.WirelessModules.SingleOrDefault(b
=> b.WirelessModuleId == id));
```

```
        }

    }
}
```

HomeController.cs

```
using WebApp.Models;
using WebApp.ViewModels;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.EntityFrameworkCore;
using System.Diagnostics.CodeAnalysis;
using System.Globalization;
using Microsoft.AspNetCore.Authorization;

namespace WebApp.Controllers.AdminDashboard
{
    public class HomeController : Controller
    {
        IuliusMallDBContext dbContext;
        AdminDashboardViewModel dashboard = new AdminDashboardViewModel();
        static List<ModulePin> modPins = new List<ModulePin>();
        static List<AccessPoint> accPoints = new List<AccessPoint>();
        static List<Location> locations = new List<Location>();
        static List<LocationLevel> locationLevels = new
List<LocationLevel>();
        static List<SensorOutcome> todaySensOutcomes = new
List<SensorOutcome>();
        static Dictionary<string, locationPeopleRecord>
locationDayPeopleRecord = new Dictionary<string, locationPeopleRecord>();
        const int passingTypeEntrance = 1;
        const int passingTypeExit = 2;
        const double mallFloorSpaceFactor = 0.7;
        const int daysInTwoWeeks = 14;
        const int hoursInDay = 24;

        public IActionResult Index()
        {
            return View();
        }

        public HomeController()
        {
            InitData();
            UpdateSensorData();
            LastTwoWeeksGuests();
        }

        void UpdateSensorData()
        {
            var queryString = "SELECT * FROM sensor_outcome WHERE
cast(sensor_outcome.time_stamp as Date) = cast(getdate() as Date)";
            todaySensOutcomes =
dbContext.SensorOutcomes.FromSqlRaw(queryString).ToList();
        }

        void InitData()
        {
            dbContext = new IuliusMallDBContext();
```

```
            modPins = dbContext.ModulePins.ToList();
            accPoints = dbContext.AccessPoints.ToList();
            locations = dbContext.Locations.ToList();
            locationLevels = dbContext.LocationLevels.ToList();
        }

        public IActionResult ShowBoxes()
        {
            /* Get the count of the locations. */
            dashboard.locationsCount = locations.Count();

            /* Calculate the entered guests number. */
            dashboard.currentDayGuestsCount = 0;
            dashboard.currentDayGuestsCount =
GuestsEnteredCount(todaySensOutcomes.ToList());

            /* Calculate the current guests number in the building. */
            int currentGuestsEntered = dashboard.currentDayGuestsCount;
            int currentGuestsLeft =
GuestsExitsCount(todaySensOutcomes.ToList());
            int currentGuestsCount = currentGuestsEntered -
currentGuestsLeft;

            dashboard.actualGuestsCount = currentGuestsCount;

            /* Get the whole building occupacy rate. */
            dashboard.maxPeopleBuildingCapacity = getBuildOccRate();
            /* According to
https://www.designingbuildings.co.uk/w/images/6/6d/Floor_space_factors.jpg
                a shopping mall's floor space factor is 0.7 m^2 for a person
*/
            dashboard.buildingOccupancyRate =
Math.Round((double)((currentGuestsCount * 100) /
dashboard.maxPeopleBuildingCapacity), 2, MidpointRounding.ToEven);

            return new JsonResult(dashboard);
        }

        public uint locationGuestsCapacity(Location location)
        {
            return (uint)(location.CapacityM2 / mallFloorSpaceFactor);
        }


        public int getBuildOccRate()
        {
            int? mallTotalCapacityM2 = 0;

            mallTotalCapacityM2 = locations.Select(x => x.CapacityM2).Sum();

            return (int)(mallTotalCapacityM2 / mallFloorSpaceFactor);
        }


        public IActionResult GetBuildingOccRate()
        {
            return new JsonResult(getBuildOccRate());
        }

        public IActionResult GetLevelsOccupacy()
        {
            dashboard.levelsOccupancyRate.Clear();
```

```
            ManageDashboardData(todaySensOutcomes,
dashboardFunctionality.eachLevelNumberOfGuests);

            foreach (var levelWithGuests in dashboard.levelsGuestsNumber)
            {
                foreach (var level in locationLevels)
                {
                    if (levelWithGuests.Key == level.LocationLevelId)
                    {
                        if
(dashboard.levelsOccupancyRate.ContainsKey(level.Name)) /* Get the level's
name */
                            dashboard.levelsOccupancyRate[level.Name] =
levelWithGuests.Value; /* Update instance */
                        else
                            dashboard.levelsOccupancyRate.Add(level.Name,
levelWithGuests.Value); /* Create new record */
                    }
                }
            }

            if (dashboard.levelsOccupancyRate.Any() == false)
            {
                foreach (var level in locationLevels)
                {
                    dashboard.levelsOccupancyRate.Add(level.Name, 0);
                }
            }

            return new JsonResult(dashboard.levelsOccupancyRate);
        }

        public IActionResult GetLocationsOccupacyRate()
        {
            dashboard.eachLocationOccupacyRate.Clear();

            ManageDashboardData(todaySensOutcomes,
dashboardFunctionality.eachLocationNumberOfGuests);

            if (dashboard.locationsGuestsNumber.Any())
            {
                foreach (var dictLocation in dashboard.locationsGuestsNumber)
                {
                    LocationOccRateData locationOccRateData = new
LocationOccRateData
                    {
                        locationOccupacyRate = 0,
                        locationName = "",
                        timeStampInSeconds = 0,
                        locationOccupacyState = ""
                    };

                    var location = locations.Where(x => x.Name ==
dictLocation.Key).SingleOrDefault();
                    if (location.CapacityM2 == 0 || location == null)
continue;

                    /* Calculate the occupacy rate for this location. */
                    var maxLocPeopleCapacity = (int)(location.CapacityM2 /
mallFloorSpaceFactor);
                    var locOccRate =
(int)(dictLocation.Value.actualGuestsNumber * 100) / maxLocPeopleCapacity; /*
cast to get an integer value */
```

```
                    locationOccRateData.locationName = location.Name;
                    locationOccRateData.locationOccupacyRate = locOccRate;
                    locationOccRateData.locationOccupacyState =
GetLocationOccupacyState(locOccRate).ToString();
                    /* Get the current time relative to a set time index.
That will be used for triggering the alarms. */
                    locationOccRateData.timeStampInSeconds =
(int)((DateTime.Now.ToUniversalTime() - new DateTime(2021, 1,
1)).TotalSeconds);


dashboard.eachLocationOccupacyRate.Add(location.LocationId.ToString(),
locationOccRateData);
                }
            }

            return new
JsonResult(dashboard.eachLocationOccupacyRate.OrderByDescending(x =>
x.Value.locationOccupacyRate));
        }

        OccupacyStates GetLocationOccupacyState(int locationOccRate)
        {
            OccupacyStates state = (locationOccRate >= 0 && locationOccRate
<= 39) ? OccupacyStates.optimal :
                                    (locationOccRate >= 40 && locationOccRate
<= 59) ? OccupacyStates.busy :
                                    (locationOccRate >= 60 && locationOccRate
<= 89) ? OccupacyStates.danger :
                                    (locationOccRate >= 90) ?
OccupacyStates.alert : OccupacyStates.invalid;

            return state;
        }

        void ManageDashboardData(List<SensorOutcome> daySensorOutcomes,
dashboardFunctionality functionality)
        {
            List<ModulePin> locationModulePins = new List<ModulePin>();
            dashboard.levelsGuestsNumber.Clear();
            dashboard.locationsGuestsNumber.Clear();
            dashboard.dayLocFlow.Clear();

            LastTwoWeeksGuests();

            if (daySensorOutcomes.Any())
            {
                foreach (var location in locations.Where(l => l.CapacityM2 !=
0).ToList())
                {
                    var locationAccPoints = accPoints.Where(x => x.LocationId
== location.LocationId).ToList();

                    var numberOfEntraces = 0u;
                    var numberOfExits = 0u;
                    var locGuestsCount = 0u;

                    foreach (var accessPoint in locationAccPoints)
                    {
                        foreach (var modPin in modPins)
                        {
                            if (modPin.AccessPointId ==
accessPoint.AccessPointId)
```

```
                            {
                                locationModulePins.Add(modPin);
                            }
                        }
                    }

                    foreach (var locModPin in locationModulePins)
                    {
                        if (locModPin.PassingTypeId == passingTypeEntrance)
                        {
                            numberOfEntraces +=
(uint)daySensorOutcomes.Where(x => x.PinAssignedId ==
locModPin.PinAssignedId).Count();
                        }

                        if (locModPin.PassingTypeId == passingTypeExit)
                            numberOfExits += (uint)daySensorOutcomes.Where(x
=> x.PinAssignedId == locModPin.PinAssignedId).Count();
                    }

                    locGuestsCount = numberOfEntraces - numberOfExits;
                    if (functionality ==
dashboardFunctionality.eachLocationNumberOfGuests)
                    {
                        LocationCapacity locCapData = new LocationCapacity();
                        locCapData.actualGuestsNumber = locGuestsCount;
                        locCapData.maxGuestsCapacity =
locationGuestsCapacity(location);
                        dashboard.locationsGuestsNumber.Add(location.Name,
locCapData);
                    }

                    if (functionality ==
dashboardFunctionality.eachLocationFlowData)
                    {
                        LocationsFlow flowData = new LocationsFlow();
                        flowData.entrances = numberOfEntraces;
                        flowData.exits = numberOfExits;

                        dashboard.dayLocFlow.Add(location.Name, flowData);
                    }

                    if (functionality ==
dashboardFunctionality.eachLevelNumberOfGuests)
                    {
                        if
(dashboard.levelsGuestsNumber.ContainsKey(location.LocationLevelId))

dashboard.levelsGuestsNumber[location.LocationLevelId] += locGuestsCount;
                        else

dashboard.levelsGuestsNumber.Add(location.LocationLevelId, locGuestsCount);
                    }

                    locationModulePins.Clear();
                    locGuestsCount = 0u;
                    numberOfEntraces = 0u;
                    numberOfExits = 0u;
                }
            }
        }
        public IActionResult LocGuestsOrderDescByGuests()
        {
```

```
            ManageDashboardData(todaySensOutcomes,
dashboardFunctionality.eachLocationNumberOfGuests);
            return new
JsonResult(dashboard.locationsGuestsNumber.OrderByDescending(x =>
x.Value.actualGuestsNumber));
        }

        public IActionResult LastTwoWeeksGuests()
        {
            int dayNumberOfGuests = 0;
            DayData selectedDay = new DayData
            {
                dayGuestsCount = 0,
                dayOfWeekName = ""
            };

            DateTime endTodayTime = DateTime.Today.AddDays(1).AddTicks(-1);
//Today at 23:59:59
            DateTime twoWeeksAgoDate = DateTime.Today.AddDays(-14);
            string selectTwoWeeksQuery = "SELECT * FROM sensor_outcome WHERE
cast(sensor_outcome.time_stamp as Date) <= '"
                                         +
endTodayTime.ToString("MM/dd/yyyy") + "' AND cast(sensor_outcome.time_stamp
as Date) > '"
                                         +
twoWeeksAgoDate.ToString("MM/dd/yyyy") + "'";

            List<SensorOutcome> lastTwoWeeksSensorData =
dbContext.SensorOutcomes.FromSqlRaw(selectTwoWeeksQuery).ToList();
            var groupSensorsDataByDay = lastTwoWeeksSensorData.GroupBy(x =>
((DateTime)x.TimeStamp).Date)
                                                                .Select(group =>
new { Day = group.Key, Items = group.ToList() }).ToList();

            /* Init dictionary data */
            dashboard.lastTwoWeeksDict.Clear();

            /* Populate firstly the dictionary with the date of last two
weeks and also each name of the day of week. */
            for (int dayCounter = -daysInTwoWeeks + 1; dayCounter < 1;
dayCounter++)
            {
                var day =
DateTime.Today.AddDays(dayCounter).ToString("MM/dd/yyyy");
                selectedDay.dayOfWeekName =
DateTime.Today.AddDays(dayCounter).DayOfWeek.ToString();

                if (dayCounter == 0)
                {

dashboard.lastTwoWeeksDict.Add(endTodayTime.ToString("MM/dd/yyyy"),
selectedDay); /* Till today 23:59:59*/
                }
                else
                {
                    dashboard.lastTwoWeeksDict.Add(day, selectedDay);
                }
            }

            selectedDay.dayGuestsCount = 0;
            selectedDay.dayOfWeekName = "";
```

```
            /* Now we are able to insert the guests number for each day for
last two weeks. */
            foreach (var day in groupSensorsDataByDay)
            {
                string formatDayDate = day.Day.ToString("MM/dd/yyyy");

                dayNumberOfGuests = GuestsEnteredCount(day.Items);
                selectedDay.dayGuestsCount = dayNumberOfGuests;
                selectedDay.dayOfWeekName = day.Day.DayOfWeek.ToString();

                dashboard.lastTwoWeeksDict[formatDayDate] = selectedDay;
                dayNumberOfGuests = 0;
            }

            return new JsonResult(dashboard.lastTwoWeeksDict);
        }

        int GuestsEnteredCount(List<SensorOutcome> daySensorData)
        {
            int guestsCounter = 0;
            if (daySensorData.Any())
            {
                foreach (var sensOutcome in daySensorData)
                {
                    foreach (var pin in modPins)
                    {
                        if ((pin.PinAssignedId == sensOutcome.PinAssignedId)
&& (pin.PassingTypeId == passingTypeEntrance))
                        {
                            foreach (var accPoint in accPoints.ToList())
                            {
                                if ((accPoint.AccessPointId ==
pin.AccessPointId) && (accPoint.Name.Contains("Hol Main")) &&
(accPoint.AccessTypeId == passingTypeEntrance))
                                {
                                    guestsCounter = guestsCounter + 1;
                                }
                            }
                        }
                    }
                }
            }

            return guestsCounter;
        }

        int GuestsExitsCount(List<SensorOutcome> daySensorData)
        {

            int guestsCounter = 0;
            if (daySensorData.Any())
            {
                foreach (var sensOutcome in daySensorData)
                {
                    foreach (var pin in modPins)
                    {
                        if ((pin.PinAssignedId == sensOutcome.PinAssignedId)
&& (pin.PassingTypeId == passingTypeExit))
                        {
                            foreach (var accPoint in accPoints.ToList())
                            {
```

```
                                if ((accPoint.AccessPointId ==
pin.AccessPointId) && (accPoint.Name.Contains("Hol Main")) &&
(accPoint.AccessTypeId == passingTypeExit))
                                {
                                    guestsCounter = guestsCounter + 1;
                                }
                            }
                        }
                    }
                }

                return guestsCounter;
            }

            [HttpGet]
            public IActionResult OnDateSelect(DateTime? date = null)
            {
                if (date != null)
                {
                    string sendQuery = "SELECT * FROM sensor_outcome WHERE
cast(sensor_outcome.time_stamp as Date) = '"
                                        +
((DateTime)date).Date.ToString("MM/dd/yyyy") + "'";
                    var selectedDaySensOutcomes =
dbContext.SensorOutcomes.FromSqlRaw(sendQuery).ToList();

                    if (selectedDaySensOutcomes.Any())
                    {
                        ManageDashboardData(selectedDaySensOutcomes,
dashboardFunctionality.eachLocationFlowData);
                        /* Now the dashboard.dayLocFlow is available. We can get
from here the entrances count for each location. */

                        LocationsPeakHour((DateTime)date);
                        /* Now the locationDayPeopleRecord is available Is needed
to get the
                         * maximum occupacy rate from each location and the time
of this event. */

                        dashboard.calendarProperties =
dashboard.dayLocFlow.Zip(locationDayPeopleRecord, (n, w) =>
                        new
                        {
                            Name = n.Key,
                            DayGuestsCount = n.Value.entrances,
                            OccRateRecord = w.Value.occupacyRateRecord,
                            PeakHour = w.Value.recordTime
                        }).ToDictionary(x => x.Name, x => new
CalendarLocationsProps
                        {
                            selDayGuestsCount = (int)x.DayGuestsCount,
                            locationPeakHour = x.PeakHour,
                            occupacyRateRecord = (int)x.OccRateRecord
                        });
                    }
                }

                return new
JsonResult(dashboard.calendarProperties.OrderByDescending(x =>
x.Value.occupacyRateRecord));
            }
```

```
        public IActionResult GetEachLocationFlowToday()
        {
            dashboard.dayLocFlow.Clear();

            ManageDashboardData(todaySensOutcomes,
dashboardFunctionality.eachLocationFlowData);

            return new JsonResult(dashboard.dayLocFlow);
        }

        public void LocationsPeakHour(DateTime date)
        {
            locationDayPeopleRecord.Clear();

            foreach (var location in locations.ToList())
            {
                if (location.CapacityM2 != 0)
                {
                    locationPeopleRecord currentLocationData = new
locationPeopleRecord();
                    uint peopleRecordCounter = 0u;

                    for (int hour = 1; hour < hoursInDay; hour++)
                    {
                        DateTime parseHoursInDay = new DateTime();
                        if (hour != hoursInDay - 1)
                            parseHoursInDay = date.AddHours(hour);
                        if (hour == hoursInDay - 1)
                            parseHoursInDay = date.AddDays(1).AddSeconds(-1);
/* Till 23:59:59*/

                        var daySensorLocationEntries =
dbContext.SensorOutcomes.FromSqlRaw("SELECT * FROM sensor_outcome WHERE
sensor_outcome.pin_assigned_id "

+ "IN (SELECT pin_assigned_id FROM module_pin WHERE
module_pin.access_point_id "

+ "IN (SELECT access_point_id FROM access_point WHERE
access_point.location_id "

+ $"= {location.LocationId}) AND module_pin.passing_type_id =
{passingTypeEntrance})"

+ " AND cast(sensor_outcome.time_stamp as DATETIME) between
CONVERT(DATETIME,"

+ $"'{(date.AddSeconds(1)).ToString("dd/MM/yyyy HH:mm:ss",
CultureInfo.InvariantCulture)}"

+ $"',103) AND CONVERT(DATETIME,'{parseHoursInDay.ToString("dd/MM/yyyy
HH:mm:ss", CultureInfo.InvariantCulture)}"

+ $"',103)").Count();
                        var daySensorLocationExits =
dbContext.SensorOutcomes.FromSqlRaw("SELECT * FROM sensor_outcome WHERE
sensor_outcome.pin_assigned_id "

+ "IN (SELECT pin_assigned_id FROM module_pin WHERE
module_pin.access_point_id "

+ "IN (SELECT access_point_id FROM access_point WHERE
access_point.location_id "
```

```
+ $"= {location.LocationId}) AND module_pin.passing_type_id =
{passingTypeExit})"

+ " AND cast(sensor_outcome.time_stamp as DATETIME) between
CONVERT(DATETIME,"

+ $"'{(date.AddSeconds(1)).ToString("dd/MM/yyyy HH:mm:ss",
CultureInfo.InvariantCulture)}"

+ $"',103) AND CONVERT(DATETIME,'{parseHoursInDay.ToString("dd/MM/yyyy
HH:mm:ss", CultureInfo.InvariantCulture)}"

+ $"',103)").Count();

                        /* Calculate the maximum of people in a hour period.
*/
                        if (peopleRecordCounter < daySensorLocationEntries -
daySensorLocationExits)
                        {
                            peopleRecordCounter =
(uint)(daySensorLocationEntries - daySensorLocationExits);
                            currentLocationData.recordTime =
parseHoursInDay.ToString("HH:mm");
                        }
                    }

                    /* Calculate the occupacy rate for this location. */
                    var maxLocPeopleCapacity = (int)(location.CapacityM2 /
mallFloorSpaceFactor);
                    var locRecordOccRate = (int)(peopleRecordCounter * 100) /
maxLocPeopleCapacity; /* cast to get an integer value */

                    currentLocationData.occupacyRateRecord =
(uint)locRecordOccRate;

                    if (locationDayPeopleRecord.ContainsKey(location.Name))
                    {
                        locationDayPeopleRecord[location.Name] =
currentLocationData;
                    }
                    else
                    {
                        locationDayPeopleRecord.Add(location.Name,
currentLocationData);
                    }
                }
            }
        }
    }
}
```

Fişierele C# HTML

_Calendar.cshtml

```
<div id="calendar-container" class="card bg-gradient-success">
    <div class="card-header border-0">

        <h3 class="card-title">
            <i class="far fa-calendar-alt"></i>
```

```
            Calendar
        </h3>
    </div>
    <!-- /.card-header -->
    <div class="card-body pt-0">
        <!--The calendar -->
        <div id="calendar" style="width: 100%"></div>
    </div>
    <!-- /.card-body -->
</div>

<div id="calendar-table" class="card card-success" style="display: none;">
    <div class="card-header">
        <button id="back-button" type="button" class="bt btn-success bt-
sm">&#60;&#60; Back</button>
        <h3 id="calendar-tab-title" class="card-title"></h3>
    </div>
    <!-- /.card-header -->
    <div class="card-body">
        <table id="calendar-data-table" class="table table-bordered">
            <thead>
                <tr>
                    <th>#</th>
                    <th>Location</th>
                    <th>Accessed (Times)</th>
                    <th>Rate record</th>
                    <th>Peak hour</th>
                </tr>
            </thead>
            <tbody id="calendar-day-table-body">
            </tbody>
        </table>
        <div class="d-flex justify-content-center">
            <div id="loading" class="spinner-border text-success"
role="status" style="display: none;">
                <span class="sr-only">Loading...</span>
            </div>
        </div>
    </div>
    <!-- /.card-body -->
</div>
<!-- /.card -->
<!-- Tempusdominus Bootstrap 4 -->
<link rel="stylesheet" href="plugins/tempusdominus-bootstrap-
4/css/tempusdominus-bootstrap-4.min.css">
<script>
    $(function () {
        $('#calendar').datetimepicker({
            format: 'L',
            inline: true,
            locale: moment.locale('en', {
                week: { dow: 1 }
            }),
        })

        $('div').on('click', 'tr > .day', function () {
            var clickedDate = $(this).data().day;
            var date = new Date(clickedDate);
            $('div tr > .day.active').removeClass('active');
            var options = { year: 'numeric', month: 'long', day: 'numeric' };

            document.getElementById("calendar-tab-title").innerHTML = "Show
data for " + date.toLocaleDateString("en-GB", options);
```

```
            $(this).addClass('active');
            $("#calendar-container").hide("slow");
            $("#calendar-table").show();
            $("#loading").show();

            $.ajax({
                type: "GET",
                url: '/Home/OnDateSelect' + '?date=' + clickedDate,
                success: function (data) {
                    if (Object.keys(data).length > 0) {
                        var toString = JSON.stringify(data);
                        var dictioData = JSON.parse(toString);
                        var calendarTableRows = "";
                        var counter = 1;
                        for (const key in dictioData) {
                            if (dictioData.hasOwnProperty(key)) {
                                var cardNominal = "bg-primary";
                                if (dictioData[key].value.occupacyRateRecord
>= 0 && dictioData[key].value.occupacyRateRecord <= 39) cardNominal = "bg-
primary";
                                if (dictioData[key].value.occupacyRateRecord
>= 40 && dictioData[key].value.occupacyRateRecord <= 59) cardNominal = "bg-
success";
                                if (dictioData[key].value.occupacyRateRecord
>= 60 && dictioData[key].value.occupacyRateRecord <= 89) cardNominal = "bg-
warning";
                                if (dictioData[key].value.occupacyRateRecord
>= 90) cardNominal = "bg-danger";
                                calendarTableRows += "<tr><td>" + counter +
"</td><td>" + dictioData[key].key + "</td><td>" +
dictioData[key].value.selDayGuestsCount + "</td><td><span class=\"badge " +
cardNominal + "\">" + dictioData[key].value.occupacyRateRecord +
"%</span></td><td><span class=\"badge " + cardNominal + "\">" +
dictioData[key].value.locationPeakHour + "</span></td></tr>"
                                counter = counter + 1;
                            }
                        }
                    }

                    document.getElementById("calendar-day-table-
body").innerHTML = calendarTableRows;

                    $('#calendar-data-table').DataTable({
                        "paging": true,
                        "searching": true,
                        "ordering": true,
                        "info": false,
                        "responsive": true,
                        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
                        "destroy": true,
                    });

                    $("#loading").hide();
                    $("#calendar-data-table").show();
                },
                error: function () {
                    $('#calendar-data-table').DataTable({
                        "paging": true,
                        "searching": true,
                        "ordering": true,
                        "info": false,
                        "responsive": true,
                        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
```

```
                        "destroy": true,
                    });

                    $("#loading").hide();
                    $("#calendar-data-table").show();
                }

            });

            return false;
        })
    })
</script>
<script>
    $("#back-button").click(function () {
        $("#calendar-table").hide();
        var calendarTable = $("#calendar-data-table");
        calendarTable.DataTable().destroy();
        $("#calendar-day-table-body tr").remove();
        $("#calendar-container").show(500);
        document.getElementById("calendar-tab-title").innerHTML = "";
    });
</script>
```

_CurrentDayGuestsFlow.cshtml

```
<!-- location_150x150.png image source https://blog.mailtag.io/how-to-track-gmail-location-email-location-tracking-guide/ -->
<div class="card card-warning">
    <div class="card-header">
        <button id="refresh-command-flow" type="button" class="bt btn-warning bt-sm toastrDefaultInfo">
            <i class="fas fa-sync-alt"></i>
            Refresh
        </button>
        <h3 class="card-title">Locations Parsed Today</h3>
    </div>
    <div class="card-body">
        <table id="guestsFlowTable" class="table table-bordered">
            <thead id ="guestsFlowTableHead">
                <tr>
                    <th>Location</th>
                    <th>
                        <small class="text-success mr-1">
                            <i class="fas fa-arrow-up"></i>
                        </small>
                        Entered (Times)
                    </th>
                    <th>
                        <small class="text-danger mr-1">
                            <i class="fas fa-arrow-down"></i>
                        </small>
                        Left (Times)
                    </th>
                </tr>
            </thead>
            <tbody id="locations-data-flow">
            </tbody>
        </table>
        <div class="d-flex justify-content-center">
            <div id="loading-flow-table" class="spinner-border text-success" role="status" style="display: none;">
                <span class="sr-only">Loading...</span>
```

```
            </div>
        </div>
    </div>
</div>

<script>
    var locationFlowTableData = "";
    var GuestsFlowTable;
    function drawWeekFlowData() {
        $.ajax({
            url: "/Home/GetEachLocationFlowToday",
            type: "GET",
            success: function (result) {
                locationFlowTableData = "";
                var toString = JSON.stringify(result);
                var locationsFlowData = JSON.parse(toString);
                for (const key in locationsFlowData) {
                    if (locationsFlowData.hasOwnProperty(key)) {
                        locationFlowTableData += "<tr><td><img
src=\"dist/img/location_150x150.png\" alt=\"Product 1\" class=\"img-circle
img-size-32 mr-2\">" +
                        key + "</td><td>" + locationsFlowData[key].entrances
+ "</td><td>" + locationsFlowData[key].exits + "</td></tr>";
                    }
                }

                if ($("#guestsFlowTable").is(':hidden')) {
                    $("#guestsFlowTable").show();
                    $("#guestsFlowTable_paginate").show();
                    $("#loading-flow-table").hide();
                }

                if (GuestsFlowTable != null) {
                    GuestsFlowTable.clear();
                    GuestsFlowTable.destroy();
                }
                var tableId = "#guestsFlowTable";

                $(tableId + " tbody").empty();
                $(tableId + " thead").empty();
                var constructHead = "<tr><th>Location</th><th> <small
class=\"text-success mr-1\"><i class=\"fas fa-arrow-up\"></i>" +
                    "</small>Entered (Times)</th><th><small class=\"text-
danger mr-1\"><i class=\"fas fa-arrow-down\">" +
                    "</i></small>Left (Times)</th> </tr>";

                document.getElementById("guestsFlowTableHead").innerHTML =
constructHead;
                document.getElementById("locations-data-flow").innerHTML =
locationFlowTableData;

                GuestsFlowTable = $('#guestsFlowTable').DataTable({
                    "paging": true,
                    "searching": true,
                    "ordering": true,
                    "info": false,
                    "responsive": true,
                    "lengthMenu": [[10, 20, 50, -1], [10, 20, 50, "All"]],
                    "destroy": true,
                });
            }
        });
    }
```

```
    drawWeekFlowData();
</script>

<script>
    $("#refresh-command-flow").click(function () {
        $("#guestsFlowTable").hide();
        $("#guestsFlowTable_paginate").hide();
        $("#loading-flow-table").show();
        drawWeekFlowData();
        setTimeout(function () { toastr.info('Successfully refreshed!'); },
1700);
    });

</script>
```

_Donut.cshtml

```
<!-- DONUT CHART -->
<div class="card card-danger">
    <div class="card-header">
        <h3 class="card-title">Donut Chart</h3>
    </div>
    <div class="card-body">
        <canvas id="donutChart" style="min-height: 250px; height: 250px; max-
height: 250px; max-width: 100%;"></canvas>
    </div>
    <!-- /.card-body -->
</div>
<!-- /.card -->

    <script>
        function updateDonutChart() {

            var locationsNameArray = [];
            var locationsDataArray = [];
            var colorForEachLocation = [];
            var donutChart;
            var colorsList = ['#f56954', '#00a65a', '#f39c12', '#00c0ef',
'#3c8dbc', '#d2d6de'];
            setInterval(drawDonutChart, 10000);
            drawDonutChart();
        function drawDonutChart() {
            $.ajax({
                url: "/Home/GetLevelsOccupacy",
                type: "GET",
                success: function (result) {
                    locationsNameArray = [];
                    locationsDataArray = [];
                    colorForEachLocation = [];

                    var toString = JSON.stringify(result);
                    var dictioData = JSON.parse(toString);
                    for (const key in dictioData) {
                        if (dictioData.hasOwnProperty(key)) {
                            locationsNameArray.push(key);
                            locationsDataArray.push(dictioData[key]);
                        }
                    }

                    for (let i = 0; i < locationsNameArray.length; i++) {
                        colorForEachLocation[i] = colorsList[i];
                    }
```

```
                    var donutChartCanvas =
$('#donutChart').get(0).getContext('2d');
                    var donutData = {
                        labels: locationsNameArray,
                        datasets: [
                            {
                                data: locationsDataArray,
                                backgroundColor: colorForEachLocation,
                            }
                        ]
                    }
                    var donutOptions = {
                        maintainAspectRatio: false,
                        responsive: true,
                        animation: {
                            duration: 0
                        },
                    }
                    if (typeof donutChart !== "undefined")
donutChart.destroy();

                    donutChart = new Chart(donutChartCanvas, {
                        type: 'doughnut',
                        data: donutData,
                        options: donutOptions,
                    });
                }
            });
            }
        }
    updateDonutChart();
</script>
```

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Admin | Dashboard</title>

    <!-- Google Font: Source Sans Pro -->
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,70
0&display=fallback">

    <!-- Site style -->
    <link rel="stylesheet" href="~/css/site.css">
    <!-- Font Awesome -->
    <link rel="stylesheet" href="plugins/fontawesome-free/css/all.min.css">
    <!-- Ionicons -->
    <link rel="stylesheet"
href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
    <!-- Theme style -->
    <link rel="stylesheet" href="dist/css/adminlte.min.css">
    <!-- Toastr -->
    <link rel="stylesheet" href="plugins/toastr/toastr.min.css">
    <!-- DataTables -->
    <link rel="stylesheet" href="plugins/datatables-
bs4/css/dataTables.bootstrap4.min.css">
```

```
    <link rel="stylesheet" href="plugins/datatables-
responsive/css/responsive.bootstrap4.min.css">
    <link rel="stylesheet" href="plugins/datatables-
buttons/css/buttons.bootstrap4.min.css">

    <!-- JS -->
    <!-- AdminLTE dashboard demo (This is only for demo purposes) -->
    <script
src="http://cdnjs.cloudflare.com/ajax/libs/summernote/0.8.9/summernote.js"
defer></script>

    <script src="~/plugins/jquery/jquery.min.js"></script>

    <script
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.10.3/jquery-
ui.min.js"></script>

    <script src="~/plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
    <script src="~/plugins/chart.js/Chart.min.js"></script>


    <!-- daterangepicker -->
    <script src="~/plugins/moment/moment.min.js"></script>
    <script src="plugins/daterangepicker/daterangepicker.js"></script>
    <!-- Tempusdominus Bootstrap 4 -->
    <script src="~/plugins/tempusdominus-bootstrap-4/js/tempusdominus-
bootstrap-4.min.js"></script>
    <script src="~/dist/js/adminlte.min.js"></script>
    <script src="dist/js/pages/dashboard.js"></script>

    <script src="plugins/toastr/toastr.min.js"></script>

    <!-- DataTables  & Plugins -->
    <script src="~/plugins/datatables/jquery.dataTables.min.js"></script>
    <script src="~/plugins/datatables-
bs4/js/dataTables.bootstrap4.min.js"></script>
</head>
<body class="hold-transition sidebar-mini layout-fixed">
    <div class="wrapper">
        <!-- Navbar -->
        <nav class="main-header navbar navbar-expand navbar-white navbar-
light">
            <!-- Left navbar links -->
            <ul class="navbar-nav">
                <li class="nav-item">
                    <a class="nav-link" data-widget="pushmenu" href="#"
role="button"><i class="fas fa-bars"></i></a>
                </li>
                <li class="nav-item d-none d-sm-inline-block">
                    <a asp-action="Index" asp-controller="Home" class="nav-
link">Home</a>
                </li>
            </ul>
        </nav>
        <!-- /.navbar -->
        <!-- Sidebar Menu -->
        @await Html.PartialAsync("_MainMenu")
        <!-- /.sidebar-menu -->
        <!-- Content Wrapper. Contains page content -->
        <div class="content-wrapper">
            <!-- Content Header (Page header) -->
            <div id = "catchHeader" class="content-header">
                <div class="container-fluid">
```

```
                    <div class="row mb-2">
                        <div class="col-sm-6">
                            <h1 class="m-0">Dashboard</h1>
                        </div><!-- /.col -->
                        <div class="col-sm-6">
                            <ol class="breadcrumb float-sm-right">
                                <li class="breadcrumb-item"><a asp-
action="Index" asp-controller="Home">Home</a></li>
                                <li class="breadcrumb-item active">Index</li>
                            </ol>
                        </div><!-- /.col -->
                    </div><!-- /.row -->
                </div><!-- /.container-fluid -->
            </div>
            <!-- /.content-header -->
            <!-- Dashboard Content -->
            @RenderBody()
            <!-- /.content -->
        </div>
        <!-- /.content-wrapper -->
        <footer class="main-footer">
            <strong>Released &copy; 2021 <a href="#">Mihailov</a>.</strong>
            <div class="float-right d-none d-sm-inline-block">
                <b>Version</b> 1.0.0
            </div>
        </footer>

        <!-- Control Sidebar -->
        <aside class="control-sidebar control-sidebar-dark">
            <!-- Control sidebar content goes here -->
        </aside>
        <!-- /.control-sidebar -->
    </div>
    <!-- ./wrapper -->
    <script>
        var controllerName =
'@ViewContext.RouteData.Values["Controller"].ToString()';
        if (controllerName != "Home") {
            document.getElementById("catchHeader").style.display = "none";
        }
        else {
            document.getElementById("catchHeader").style.display = "block";
        }
    </script>
</body>
</html>
```

_LocationsTable.cshtml

```
<div class="card card-success">
    <div class="card-header">
        <button id="refresh-command" type="button" class="bt btn-success bt-
sm toastrDefaultInfo">
            <i class="fas fa-sync-alt"></i>
            Refresh
        </button>
        <h3 class="card-title">Occupacy Rate</h3>
    </div>
    <!-- card-body -->
    <div class="card-body">
        <table id="locationsTable" class="table table-bordered">
            <thead>
                <tr>
```

```
                    <th style="width: 10px">#</th>
                    <th>Location</th>
                    <th>Progress</th>
                    <th style="width: 40px">Occupacy</th>
                </tr>
            </thead>
            <tbody id="locations-table-body">
            </tbody>
        </table>
        <div class="d-flex justify-content-center">
            <div id="loading-location-table" class="spinner-border text-
success" role="status" style="display: none;">
                <span class="sr-only">Loading...</span>
            </div>
        </div>
    </div>
    <!-- /.card-body -->
</div>

<script>
    var locationsIds = [];
    var locationsNames = [];
    var locationsOccRate = [];
    var LocationDataTable;
    function LocationsOccRateTable() {
        $.ajax({
            url: "/Home/GetLocationsOccupacyRate",
            type: "GET",
            success: function (result) {
                locationsIds = [];
                locationsNames = [];
                locationsOccRate = [];

                var toString = JSON.stringify(result);
                var dictioData = JSON.parse(toString);
                for (const key in dictioData) {
                    if (dictioData.hasOwnProperty(key)) {
                        locationsIds.push(key);

locationsNames.push(dictioData[key].value.locationName);

locationsOccRate.push(dictioData[key].value.locationOccupacyRate);
                    }
                }

                var locOccRateRaws = "";
                var cardNominal = "";
                for (let i = 0; i < locationsIds.length; i++) {
                    if (locationsOccRate[i] >= 0 && locationsOccRate[i] <=
39) cardNominal = "bg-primary";
                    if (locationsOccRate[i] >= 40 && locationsOccRate[i] <=
59) cardNominal = "bg-success";
                    if (locationsOccRate[i] >= 60 && locationsOccRate[i] <=
89) cardNominal = "bg-warning";
                    if (locationsOccRate[i] >= 90) cardNominal = "bg-danger";

                    locOccRateRaws += "<tr><td>" + locationsIds[i] +
"</td><td>" + locationsNames[i] + "</td><td><div class=\"progress progress-
xs\"><div class=\"progress-bar " + cardNominal + "\" style=\"width:" +
locationsOccRate[i] + "%\"></div></div></td><td><span class=\"badge " +
cardNominal + "\">" + locationsOccRate[i] + "%</span></td></tr>"
                }
```

```
                if ($("#locationsTable").is(':hidden')) {
                    $("#locationsTable").show();
                    $("#locationsTable_paginate").show();
                    $("#loading-location-table").hide();
                }

                if (typeof LocationDataTable !== "undefined") {
                    LocationDataTable.clear();
                    LocationDataTable.destroy();
                }

                document.getElementById("locations-table-body").innerHTML =
locOccRateRaws;

                LocationDataTable = $('#locationsTable').DataTable({
                    "paging": true,
                    "searching": true,
                    "ordering": true,
                    "info": false,
                    "responsive": true,
                    "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
                    "destroy": true,
                });
            }
        });
    }

    LocationsOccRateTable();
</script>

<script>
    $("#refresh-command").click(function () {
        $("#locationsTable").hide();
        $("#locationsTable_paginate").hide();
        $("#loading-location-table").show();
        LocationsOccRateTable();
        setTimeout(function () { toastr.info('Successfully refreshed!'); },
1700);
    });
</script>
```

_MainMenu.cshtml

```
  <!-- Main Sidebar Container -->
<aside class="main-sidebar sidebar-dark-primary elevation-4">
    <!-- Sidebar -->
    <div class="sidebar">
        <!-- Sidebar Menu -->
        <nav class="mt-2">
            <ul class="nav nav-pills nav-sidebar flex-column" data-
widget="treeview" role="menu" data-accordion="false">
                <li class="nav-item">
                    <a id="Home" asp-action="Index" asp-controller="Home"
class="nav-link">
                        <i class="nav-icon fas fa-tachometer-alt"></i>
                        <p>
                            Dashboard
                        </p>
                    </a>
                </li>
                <li class="nav-item">
                    <a id="AccessPointsTable" asp-action="index" asp-
controller="AccessPointsTable" class="nav-link">
```

```
                        <i class="nav-icon fas fa-exchange-alt"></i>
                        <p>
                            Access Points
                        </p>
                    </a>
                </li>
                <li class="nav-item">
                    <a id="LocationsTable" asp-action="index" asp-
controller="LocationsTable" class="nav-link">
                        <i class="nav-icon fas fa-building"></i>
                        <p>
                            Locations
                        </p>
                    </a>
                </li>
                <li class="nav-item">
                    <a id="LocationLevelsTable" asp-action="index" asp-
controller="LocationLevelsTable" class="nav-link">
                        <i class="nav-icon fas fa-server"></i>
                        <p>
                            Levels
                        </p>
                    </a>
                </li>
                <li class="nav-item">
                    <a id="WirelessModulesTable" asp-action="index" asp-
controller="WirelessModulesTable" class="nav-link">
                        <i class="nav-icon fas fa-microchip"></i>
                        <p>
                            Wireless Modules
                        </p>
                    </a>
                </li>
                <li class="nav-item">
                    <a id="ModulePinsTable" asp-action="index" asp-
controller="ModulePinsTable" class="nav-link">
                        <i class="nav-icon fas fa-circle"></i>
                        <p>
                            Module's Pins
                        </p>
                    </a>
                </li>
            </ul>
        </nav>
        <!-- /.sidebar-menu -->


    </div>
    <!-- /.sidebar -->
</aside>

<script>
    var controllerName =
'@ViewContext.RouteData.Values["Controller"].ToString()';

    var element = document.getElementById(controllerName);
    element.classList.add("active");
</script>
```

_TopLocationsLive.cshtml

```
<div class="card card-blue">
    <div class="card-header border-transparent">
        <span class="badge-live bg-danger">LIVE</span>
```

```
        <h3 class="card-title">Top guests locations</h3>
    </div>
    <!-- /.card-header -->
    <div class="card-body">
        <div id="locationsTable_wrapper" class="dataTables_wrapper dt-
bootstrap4 no-footer">

            <div class="table-responsive">

                <div class="dataTables_length">
                <label class="lab-prop">Show
                    <select id="locationsTableLength" class="custom-select
custom-select-sm form-control form-control-sm">
                    <option value="8" selected="selected">8</option>
                    <option value="10">10</option>
                    <option value="20">20</option>
                    <option value="-1">All</option>
                        </select> entries</label></div>

                <table class="table m-0">
                    <thead>
                        <tr>
                            <th>Name</th>
                            <th>Guests Number</th>
                            <th>Max Guests Capacity</th>
                            <th>Status</th>
                        </tr>
                    </thead>
                    <tbody id="top-lcs_live-body">
                    </tbody>
                </table>
            </div>
            <!-- /.table-responsive -->
        </div>
    </div>
        <!-- /.card-body -->
    </div>
<!-- /.card -->
<script>
    /* Dictionary to store last state of each location(only for danger and
     * alert states). This will be used to trigger the notifications alarm.
     * If a location got "danger" state then the state will be checked once
     * in a minute. If the current state is "alert", then the state will be
checked
     * once in 30 seconds. If last state was "alert" and now is "danger",
     * pass and check again after one minute. */
    var oldOccupacyRateStates = [];
    function updateTopLocations() {
        setInterval(findTopLocations, 3000);
        findTopLocations();
        function findTopLocations() {
            $.ajax({
                url: "/Home/LocGuestsOrderDescByGuests",
                type: "GET",
                success: function (result) {
                    var toString = JSON.stringify(result);
                    var topLocations = JSON.parse(toString);
                    $.ajax({
                        url: "/Home/GetLocationsOccupacyRate",
                        type: "GET",
                        success: function (data) {
                            var cardNominal = "";
                            var locationStatus = "";
```

```
                            var topLocationsBody = "";
                            var toString = JSON.stringify(data);
                            var locationFound = false;

                            var locTableLength =
document.getElementById('locationsTableLength').value;
                            if (locTableLength == -1) locTableLength =
Object.keys(topLocations).length;
                            var dictioData = JSON.parse(toString);
                            if ((Object.keys(dictioData).length != 0) &&
(Object.keys(topLocations).length != 0)) {
                                for (const topLocationsKey in topLocations) {
                                    for (const dictioDataKey in dictioData) {
                                        if (topLocations[topLocationsKey].key
== dictioData[dictioDataKey].value.locationName) {
                                            if
(dictioData[dictioDataKey].value.locationOccupacyState.localeCompare("optimal
") == 0) {
                                                cardNominal = "badge-
success"; locationStatus = "optimal"; locationFound = true;
                                            }
                                            if
(dictioData[dictioDataKey].value.locationOccupacyState.localeCompare("busy")
== 0) {
                                                cardNominal = "badge-
primary"; locationStatus = "busy"; locationFound = true;
                                            }

                                            if
(dictioData[dictioDataKey].value.locationOccupacyState.localeCompare("danger"
) == 0) {
                                                var locFoundInOldData =
false;

                                                cardNominal = "badge-
warning"; locationStatus = "danger"; locationFound = true;

                                                /* Check if old Occupacy Rate
States Dictionary contains already the locations. */
                                                for (const locatKey in
oldOccupacyRateStates) {
                                                    if
(oldOccupacyRateStates[locatKey].key ==
dictioData[dictioDataKey].value.locationName) { /* If true, this location was
already triggered with danger state or alert*/
                                                        if
(oldOccupacyRateStates[locatKey].value.State == "danger") { /* If true, check
if 1 minute left and trigger the notification. */
                                                            if
(dictioData[dictioDataKey].value.timeStampInSeconds -
oldOccupacyRateStates[locatKey].value.TimeStampInSeconds >= 60) {

toastr.warning('Warning! The ' + dictioData[dictioDataKey].value.locationName
+ ' is starting to get crowded.');
                                                                delete
oldOccupacyRateStates[locatKey];
                                                            }
                                                        }
                                                        else if
(oldOccupacyRateStates[locatKey].value.State == "alert") { /* If true,
comparing to last state, the current state is better. Don't trigger the
alarm. Wait for one more minute and check again. */
```

```
                                                    if
(dictioData[dictioDataKey].value.timeStampInSeconds -
oldOccupacyRateStates[locatKey].value.TimeStampInSeconds >= 120) {

toastr.warning('Warning! The ' + dictioData[dictioDataKey].value.locationName
+ ' is starting to get crowded.');
                                                        delete
oldOccupacyRateStates[locatKey];
                                                    }
                                                }
                                                locFoundInOldData =
true;
                                            }
                                        }

                                        if (locFoundInOldData ==
false) { /* Here are no records at all or it's the first time in a period
when this location is with this status. */

oldOccupacyRateStates.push({
                                                key:
dictioData[dictioDataKey].value.locationName,
                                                value: {
                                                    State:
dictioData[dictioDataKey].value.locationOccupacyState,

TimeStampInSeconds: dictioData[dictioDataKey].value.timeStampInSeconds,
                                                }
                                            });
                                        }
                                    }

                                    if
(dictioData[dictioDataKey].value.locationOccupacyState.localeCompare("alert")
== 0) {
                                        var locFoundInOldData =
false;

                                        cardNominal = "badge-danger";
locationStatus = "alert"; locationFound = true;
                                        for (const locatKey in
oldOccupacyRateStates) {
                                            if
(oldOccupacyRateStates[locatKey].key ==
dictioData[dictioDataKey].value.locationName) {
                                                if
(dictioData[dictioDataKey].value.timeStampInSeconds -
oldOccupacyRateStates[locatKey].value.TimeStampInSeconds >= 30) {

toastr.error('Alert! The ' + dictioData[dictioDataKey].value.locationName + "
location is almost full!");
                                                    delete
oldOccupacyRateStates[locatKey];
                                                }
                                                locFoundInOldData =
true;
                                            }
                                        }

                                        if (locFoundInOldData ==
false) { /* Here are no records at all or it's the first time in a period
when this location is with this status. */
```

```
oldOccupacyRateStates.push({
                                                        key:
dictioData[dictioDataKey].value.locationName,
                                                        value: {
                                                            State:
dictioData[dictioDataKey].value.locationOccupacyState,

TimeStampInSeconds: dictioData[dictioDataKey].value.timeStampInSeconds,
                                                        }
                                                    });
                                                }
                                            }

                                            if (locTableLength != 0) { /*
Show just wanted number of record. */
                                                if (locationFound == true) {
                                                    topLocationsBody +=
"<tr><td>" + dictioData[dictioDataKey].value.locationName + "</td><td>" +
topLocations[topLocationsKey].value.actualGuestsNumber + "</td><td>" +
topLocations[topLocationsKey].value.maxGuestsCapacity + "</td><td><span
class=\"badge " + cardNominal + "\">" + locationStatus + "</span></td></tr>";
                                                    locationFound = false;
                                                }

                                                locTableLength =
locTableLength - 1;
                                            }
                                        }
                                    }
                                }
                                document.getElementById("top-lcs_live-
body").innerHTML = topLocationsBody;
                                (function blink() {
                                    $('.badge-
live').fadeOut(600).fadeIn(1600, blink);
                                })();
                            }
                            else {
                                var emptyBody = "<tr class=\"odd\"><td
valign=\"top\" colspan=\"4\" class=\"dataTables_empty no-elem\">No data
available in table</td></tr>";
                                document.getElementById("top-lcs_live-
body").innerHTML = emptyBody;
                            }
                        },
                        error: function(){
                            $('.badge-live').mouseover(
                                function () {
                                    if ($(this).is(':animated')) {
                                        $(this).stop().animate({ opacity:
'100' });
                                    }
                                }
                            );
                        }
                    });
                }
            });
        }
    }

    updateTopLocations();
```

```
</script>
```

_WeekVisitors.cshtml

```
<div class="card card-blue">
    <div class="card-header border-0">
        <div class="d-flex justify-content-between">
            <h3 class="card-title">Building Visitors</h3>
        </div>
    </div>
    <div class="card-body">
        <div class="d-flex">
            <p id ="two-weeks-count" class="d-flex flex-column"></p>
            <p id = "percentage-Data" class="ml-auto d-flex flex-column text-
right"></p>
        </div>
        <!-- /.d-flex -->

        <div class="position-relative mb-4">
            <canvas id="visitors-chart" height="200"></canvas>
        </div>

        <div class="d-flex flex-row justify-content-end">
            <span class="mr-2">
                <i class="fas fa-square text-primary"></i> This Week
            </span>

            <span>
                <i class="fas fa-square text-gray"></i> Last Week
            </span>
        </div>
    </div>
</div>
<!-- /.card -->
<!-- /.content -->

<script>
    function drawTwoWeeksGuestsData() {
        var dayGuestsCountArray = [];
        var dayOfWeekNamesArray = [];
        var maxCapacity = 0;
        $.ajax({
        url: "/Home/LastTwoWeeksGuests",
        type: "GET",
        success: function (result) {
            dayGuestsCountArray = [];
            dayOfWeekNamesArray = [];

            var toString = JSON.stringify(result);
            var dictioData = JSON.parse(toString);
            for (const key in dictioData) {
                if (dictioData.hasOwnProperty(key)) {
                    dayGuestsCountArray.push(dictioData[key].dayGuestsCount);
                    dayOfWeekNamesArray.push(dictioData[key].dayOfWeekName);
                }
            }
            var days = dayOfWeekNamesArray.slice(0, 7);
            var lastWeek = dayGuestsCountArray.slice(0, 7);
            var thisWeek = dayGuestsCountArray.slice(7, 14);
            var lastWeekGuestsCount = lastWeek.reduce((a, b) => a + b, 0);
            var thisWeekGuestsCount = thisWeek.reduce((a, b) => a + b, 0);
            var percentageCompare = 0;
            var percentageDetails = "";
```

```
            if (lastWeekGuestsCount >= thisWeekGuestsCount) {
                percentageCompare = 100 - parseInt((thisWeekGuestsCount *
100) / lastWeekGuestsCount);
                percentageDetails = "<span class=\"text-red\"><i class=\"fas
fa-arrow-down\"></i >" + percentageCompare + "% </span ><span class=\"text-
muted\">Since last week</span>";

            }
            else {
                percentageCompare = 100 - parseInt((lastWeekGuestsCount *
100) / thisWeekGuestsCount);
                percentageDetails = "<span class=\"text-success\"><i
class=\"fas fa-arrow-up\"></i >" + percentageCompare + "% </span ><span
class=\"text-muted\">Since last week</span>";
            }

            var guestsCount = "<span class=\"text-bold text - lg\">" +
(thisWeekGuestsCount + lastWeekGuestsCount) + "</span> <span> Visitors Over
Time</span >";

            var ticksStyle = {
                fontColor: '#495057',
                fontStyle: 'bold'
            }

            $('#two-weeks-count').html(guestsCount);

            $('#percentage-Data').html(percentageDetails);

            $.ajax({
                url: "/Home/GetBuildingOccRate",
                type: "GET",
                success: function (resultData) {
                    maxCapacity = resultData;
                }
            });

            var mode = 'index'
            var intersect = true

            var $visitorsChart = $('#visitors-chart')
            // eslint-disable-next-line no-unused-vars
            var visitorsChart = new Chart($visitorsChart, {
                data: {
                    labels: days,
                    datasets: [{
                        type: 'line',
                        data: thisWeek,
                        backgroundColor: 'transparent',
                        borderColor: '#007bff',
                        pointBorderColor: '#007bff',
                        pointBackgroundColor: '#007bff',
                        fill: false
                    },
                    {
                        type: 'line',
                        data: lastWeek,
                        backgroundColor: 'tansparent',
                        borderColor: '#ced4da',
                        pointBorderColor: '#ced4da',
                        pointBackgroundColor: '#ced4da',
                        fill: false
```

```
                    }]
                },
                options: {
                    maintainAspectRatio: false,
                    tooltips: {
                        mode: mode,
                        intersect: intersect
                    },
                    hover: {
                        mode: mode,
                        intersect: intersect
                    },
                    legend: {
                        display: false
                    },
                    scales: {
                        yAxes: [{
                            gridLines: {
                                display: true,
                                lineWidth: '4px',
                                color: 'rgba(0, 0, 0, .2)',
                                zeroLineColor: 'transparent'
                            },
                            ticks: $.extend({
                                beginAtZero: true,
                                suggestedMax: maxCapacity + 100
                            }, ticksStyle)
                        }],
                        xAxes: [{
                            display: true,
                            gridLines: {
                                display: false
                            },
                            ticks: ticksStyle
                        }]
                    }
                }
            })
        }
    });
    }

    drawTwoWeeksGuestsData();
</script>
```

WirelessModulesTable -> Index.cshtml

```
@model IEnumerable<WebApp.Models.WirelessModule>

<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></
script>
<link rel="stylesheet" href="~/css/crudTables.css">

<!-- Main content -->
<section class="content">
    <div class="container-xl">
        <div class="table-responsive">
            <div class="table-wrapper">
                <div class="table-title">
                    <div class="row">
```

```
                        <div class="col-sm-6">
                            <h2>Manage <b>Wireless Modules</b></h2>
                        </div>
                        <div class="col-sm-6">
                            <a href="#AddWirelessModule" class="btn btn-
success" data-toggle="modal"><i class="material-icons">&#xE147;</i> <span>Add
Wireless Module</span></a>
                        </div>
                    </div>
                </div>
                <table id="crudWirelessModulesTable" class="table table-
striped table-hover">
                    <thead>
                        <tr>
                            <th>Wireless Module Id</th>
                            <th>Name</th>
                            <th>Nr Of Data Pins</th>
                            <th>Factory Guid</th>
                            <th>Options</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var item in Model)
                        {
                        <tr>
                            <td>@item.WirelessModuleId</td>
                            <td>@item.Name</td>
                            <td>@item.NrOfDataPins</td>
                            <td>@item.FactoryGuid</td>
                            <td>
                                <a href="#EditWirelessModule" class="edit"
data-toggle="modal"><i class="material-icons" data-toggle="tooltip"
title="Edit"
onclick="EditWirelessModule('@item.WirelessModuleId')">&#xE254;</i></a>
                                <a href="#DeleteWirelessModule"
class="delete" data-toggle="modal"><i class="material-icons" data-
toggle="tooltip" title="Delete"
onclick="DeleteWirelessModule('@item.WirelessModuleId')">&#xE872;</i></a>
                            </td>
                        </tr>
                        }
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <!-- Add Modal HTML -->
    <div id="AddWirelessModule" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post"
action="WirelessModulesTable/AddWirelessModule">
                    <div class="modal-header">
                        <h4 class="modal-title">Add Wireless Module</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name" placeholder="Name"
class="form-control" required>
                        </div>
```

```
                        <div class="form-group">
                            <label>Nr Of Data Pins</label>
                            <input type="number" name="NrOfDataPins"
placeholder="NrOfDataPins" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Factory Guid</label>
                            <input type="number" name="FactoryGuid"
placeholder="FactoryGuid" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-success"
value="Add">
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Edit Modal HTML -->
    <div id="EditWirelessModule" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post"
action="WirelessModulesTable/EditWirelessModule">
                    <div class="modal-header">
                        <h4 class="modal-title">Edit Wireless Module</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Wireless Module Id</label>
                            <input type="number" name="WirelessModuleId"
id="EditWirelessModuleId" class="form-control" readonly>
                        </div>
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name"
id="EditWirelessModuleName" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Nr Of Data Pins</label>
                            <input type="number" name="NrOfDataPins"
id="EditNrOfDataPins" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Factory Guid</label>
                            <input type="number" name="FactoryGuid"
id="EditFactoryGuid" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-info"
value="Save">
                    </div>
                </form>
            </div>
        </div>
```

```
    </div>
    <!-- Delete Modal HTML -->
    <div id="DeleteWirelessModule" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form id="DeleteWirelessModuleForm" method="post">
                    <div class="modal-header">
                        <h4 class="modal-title">Delete Wireless Module</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <p>Are you sure you want to delete these Records?</p>
                        <p class="text-warning"><small>This action cannot be
undone.</small></p>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-danger"
value="Delete">
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>

<script>
    $('#crudWirelessModulesTable').DataTable({
        "paging": true,
        "searching": true,
        "ordering": true,
        "info": false,
        "responsive": true,
        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
        "destroy": true,
    });
</script>

<script>
    function EditWirelessModule(id) {
        $.ajax({
            url: "/WirelessModulesTable/GetWirelessModuleById/" + id,
            type: "GET",
            success: function (result) {
                document.getElementById("EditWirelessModuleId").value =
result.wirelessModuleId;
                document.getElementById("EditWirelessModuleName").value =
result.name;
                document.getElementById("EditNrOfDataPins").value =
result.nrOfDataPins;
                document.getElementById("EditFactoryGuid").value =
result.factoryGuid;
            }
        });
    }

    function DeleteWirelessModule(id) {
        document.getElementById('DeleteWirelessModuleForm').action =
"WirelessModulesTable/DeleteWirelessModule/" + id;
    }
</script>
```

```
<script>
    (function () {
        if (window.location.hash.substr(1) == 'Edit')
            toastr.info("Congratulations! Your data has been successfully
edited.");
        if (window.location.hash.substr(1) == 'Delete')
            toastr.info("Congratulations! Your data has been successfully
deleted.");
        if (window.location.hash.substr(1) == 'Add')
            toastr.info("Congratulations! Your data has been successfully
added.");
    })();
</script>
```

ModulePinsTable -> Index.cshtml

```
@model IEnumerable<WebApp.Models.ModulePin>

<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></
script>
<link rel="stylesheet" href="~/css/crudTables.css">

<!-- Main content -->
<section class="content">
    <div class="container-xl">
        <div class="table-responsive">
            <div class="table-wrapper">
                <div class="table-title">
                    <div class="row">
                        <div class="col-sm-6">
                            <h2>Manage <b>Wireless Modules</b></h2>
                        </div>
                        <div class="col-sm-6">
                            <a href="#AddModulePin" class="btn btn-success"
data-toggle="modal"><i class="material-icons">&#xE147;</i> <span>Add Wireless
Module</span></a>
                        </div>
                    </div>
                </div>
                <table id="crudModulePinsTable" class="table table-striped
table-hover">
                    <thead>
                        <tr>
                            <th>Pin Assigned Id</th>
                            <th>Name</th>
                            <th>Wireless Module Id</th>
                            <th>Access Point Id</th>
                            <th>Pair Id</th>
                            <th>Passing Type Id<br>(1-IN/2-OUT)</th>
                            <th>Options</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var item in Model)
                        {
                        <tr>
                            <td>@item.PinAssignedId</td>
                            <td>@item.Name</td>
                            <td>@item.WirelessModuleId</td>
```

```
                            <td>@item.AccessPointId</td>
                            <td>@item.PairId</td>
                            <td>@item.PassingTypeId</td>

                            <td>
                                <a href="#EditModulePin" class="edit" data-
toggle="modal"><i class="material-icons" data-toggle="tooltip" title="Edit"
onclick="EditModulePin('@item.PinAssignedId')">&#xE254;</i></a>
                                <a href="#DeleteModulePin" class="delete"
data-toggle="modal"><i class="material-icons" data-toggle="tooltip"
title="Delete"
onclick="DeleteModulePin('@item.PinAssignedId')">&#xE872;</i></a>
                            </td>
                        </tr>
                        }
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <!-- Add Modal HTML -->
    <div id="AddModulePin" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post" action="ModulePinsTable/AddModulePin">
                    <div class="modal-header">
                        <h4 class="modal-title">Add Wireless Module</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name" placeholder="Name"
class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Wireless Module Id</label>
                            <input type="number" name="WirelessModuleId"
placeholder="WirelessModuleId" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Access Point Id</label>
                            <input type="number" name="AccessPointId"
placeholder="AccessPointId" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Pair Id (not mandatory)</label>
                            <input type="number" name="PairId"
placeholder="PairId" class="form-control">
                        </div>
                        <div class="form-group">
                            <label>Passing Type Id (1-IN/2-OUT)</label>
                            <input type="number" name="PassingTypeId"
placeholder="PassingTypeId" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-success"
value="Add">
                    </div>
```

```
                </form>
            </div>
        </div>
    </div>
    <!-- Edit Modal HTML -->
    <div id="EditModulePin" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post" action="ModulePinsTable/EditModulePin">
                    <div class="modal-header">
                        <h4 class="modal-title">Edit Wireless Module</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Pin Assigned Id</label>
                            <input type="number" name="PinAssignedId"
id="EditPinAssignedId" class="form-control" readonly>
                        </div>
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name"
id="EditModulePinName" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Wireless Module Id</label>
                            <input type="number" name="WirelessModuleId"
id="EditWirelessModuleId" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Access Point Id</label>
                            <input type="number" name="AccessPointId"
id="EditAccessPointId" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Pair Id</label>
                            <input type="number" name="PairId"
id="EditPairId" class="form-control">
                        </div>
                        <div class="form-group">
                            <label>Passing Type Id</label>
                            <input type="number" name="PassingTypeId"
id="EditPassingTypeId" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-info"
value="Save">
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Delete Modal HTML -->
    <div id="DeleteModulePin" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form id="DeleteModulePinForm" method="post">
                    <div class="modal-header">
                        <h4 class="modal-title">Delete Wireless Module</h4>
```

```
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <p>Are you sure you want to delete these Records?</p>
                        <p class="text-warning"><small>This action cannot be
undone.</small></p>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-danger"
value="Delete">
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>

<script>
    $('#crudModulePinsTable').DataTable({
        "paging": true,
        "searching": true,
        "ordering": true,
        "info": false,
        "responsive": true,
        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
        "destroy": true,
    });
</script>

<script>
    function EditModulePin(id) {
        $.ajax({
            url: "/ModulePinsTable/GetModulePinById/" + id,
            type: "GET",
            success: function (result) {
                document.getElementById("EditPinAssignedId").value =
result.pinAssignedId;
                document.getElementById("EditWirelessModuleId").value =
result.wirelessModuleId;
                document.getElementById("EditModulePinName").value =
result.name;
                document.getElementById("EditAccessPointId").value =
result.accessPointId;
                document.getElementById("EditPairId").value = result.pairId;
                document.getElementById("EditPassingTypeId").value =
result.passingTypeId;
            }
        });
    }

    function DeleteModulePin(id) {
        document.getElementById('DeleteModulePinForm').action =
"ModulePinsTable/DeleteModulePin/" + id;
    }
</script>

<script>
    (function () {
        if (window.location.hash.substr(1) == 'Edit')
```

```
            toastr.info("Congratulations! Your data has been successfully
edited.");
        if (window.location.hash.substr(1) == 'Delete')
            toastr.info("Congratulations! Your data has been successfully
deleted.");
        if (window.location.hash.substr(1) == 'Add')
            toastr.info("Congratulations! Your data has been successfully
added.");
    })();
</script>
```

LocationsTable -> Index.cshtml

```
@model IEnumerable<WebApp.Models.Location>

<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></
script>
<link rel="stylesheet" href="~/css/crudTables.css">

<!-- Main content -->
<section class="content">
    <div class="container-xl">
        <div class="table-responsive">
            <div class="table-wrapper">
                <div class="table-title">
                    <div class="row">
                        <div class="col-sm-6">
                            <h2>Manage <b>Locations</b></h2>
                        </div>
                        <div class="col-sm-6">
                            <a href="#AddLocation" class="btn btn-success"
data-toggle="modal"><i class="material-icons">&#xE147;</i> <span>Add New
Location</span></a>
                        </div>
                    </div>
                </div>
                <table id="crudLocationsTable" class="table table-striped
table-hover">
                    <thead>
                        <tr>
                            <th>Location Id</th>
                            <th>Name</th>
                            <th>Location LevelId</th>
                            <th>Capacity m2</th>
                            <th>Options</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var item in Model)
                        {
                            <tr>
                                <td>@item.LocationId</td>
                                <td>@item.Name</td>
                                <td>@item.LocationLevelId</td>
                                <td>@item.CapacityM2</td>
                                <td>
                                    <a href="#EditLocation" class="edit"
data-toggle="modal"><i class="material-icons" data-toggle="tooltip"
title="Edit" onclick="EditLocation('@item.LocationId')">&#xE254;</i></a>
```

```
                                    <a href="#DeleteLocation" class="delete"
data-toggle="modal"><i class="material-icons" data-toggle="tooltip"
title="Delete" onclick="DeleteLocation('@item.LocationId')">&#xE872;</i></a>
                                </td>
                            </tr>
                        }
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <!-- Add Modal HTML -->
    <div id="AddLocation" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post" action="LocationsTable/CreateNewAdmin">
                    <div class="modal-header">
                        <h4 class="modal-title">Add Location</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name" placeholder="Name"
class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Location Level Id</label>
                            <input type="number" name="LocationLevelId"
placeholder="Location Level Id" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Capacity m2</label>
                            <input type="number" name="CapacityM2"
placeholder="Capacity m2" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-success"
value="Add">
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Edit Modal HTML -->
    <div id="EditLocation" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post" action="LocationsTable/EditLocation">
                    <div class="modal-header">
                        <h4 class="modal-title">Edit Location</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Location Id</label>
                            <input type="number" name="LocationId"
id="EditLocationId" class="form-control" readonly>
```

```
                        </div>
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name"
id="EditLocationName" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Location Level Id</label>
                            <input type="number" name="LocationLevelId"
id="EditLocationLevelId" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Capacity m2</label>
                            <input type="number" name="CapacityM2"
id="EditCapacityM2" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-info"
value="Save">
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Delete Modal HTML -->
    <div id="DeleteLocation" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form id="DeleteLocationForm" method="post">
                    <div class="modal-header">
                        <h4 class="modal-title">Delete Location</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <p>Are you sure you want to delete these Records?</p>
                        <p class="text-warning"><small>This action cannot be
undone.</small></p>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-danger"
value="Delete">
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>

<script>
    $('#crudLocationsTable').DataTable({
        "paging": true,
        "searching": true,
        "ordering": true,
        "info": false,
        "responsive": true,
        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
        "destroy": true,
```

```
    });
</script>

<script>
    function EditLocation(id) {
        $.ajax({
            url: "/LocationsTable/GetLocationById/" + id,
            type: "GET",
            success: function (result) {
                document.getElementById("EditLocationId").value =
result.locationId;
                document.getElementById("EditLocationName").value =
result.name;
                document.getElementById("EditLocationLevelId").value =
result.locationLevelId;
                document.getElementById("EditCapacityM2").value =
result.capacityM2;
            }
        });
    }

    function DeleteLocation(id) {
        document.getElementById('DeleteLocationForm').action =
"LocationsTable/DeleteLocation/" + id;
    }
</script>

<script>
    (function () {
        if (window.location.hash.substr(1) == 'Edit')
            toastr.info("Congratulations! Your data has been successfully
edited.");
        if (window.location.hash.substr(1) == 'Delete')
            toastr.info("Congratulations! Your data has been successfully
deleted.");
        if (window.location.hash.substr(1) == 'Add')
            toastr.info("Congratulations! Your data has been successfully
added.");
    })();
</script>
```

LocationLevelsTable -> Index.cshtml

```
@model IEnumerable<WebApp.Models.LocationLevel>

<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></
script>
<link rel="stylesheet" href="~/css/crudTables.css">

<!-- Main content -->
<section class="content">
    <div class="container-xl">
        <div class="table-responsive">
            <div class="table-wrapper">
                <div class="table-title">
                    <div class="row">
                        <div class="col-sm-6">
                            <h2>Manage <b>Location Levels</b></h2>
                        </div>
                        <div class="col-sm-6">
```

```
                            <a href="#AddLocationLevel" class="btn btn-
success" data-toggle="modal"><i class="material-icons">&#xE147;</i> <span>Add
New Level</span></a>
                        </div>
                    </div>
                </div>
                <table id="crudLocationLevelsTable" class="table table-
striped table-hover">
                    <thead>
                        <tr>
                            <th>Level Id</th>
                            <th>Name</th>
                            <th>Options</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var item in Model)
                        {
                            <tr>
                                <td>@item.LocationLevelId</td>
                                <td>@item.Name</td>
                                <td>
                                    <a href="#EditLocationLevel" class="edit"
data-toggle="modal"><i class="material-icons" data-toggle="tooltip"
title="Edit"
onclick="EditLocationLevel('@item.LocationLevelId')">&#xE254;</i></a>
                                    <a href="#DeleteLocationLevel"
class="delete" data-toggle="modal"><i class="material-icons" data-
toggle="tooltip" title="Delete"
onclick="DeleteLocationLevel('@item.LocationLevelId')">&#xE872;</i></a>
                                </td>
                            </tr>
                        }
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <!-- Add Modal HTML -->
    <div id="AddLocationLevel" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post"
action="LocationLevelsTable/AddLocationLevel">
                    <div class="modal-header">
                        <h4 class="modal-title">Add Level</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name" placeholder="Name"
class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-success"
value="Add">
                    </div>
                </form>
```

```
            </div>
        </div>
    </div>
    <!-- Edit Modal HTML -->
    <div id="EditLocationLevel" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post"
action="LocationLevelsTable/EditLocationLevel">
                    <div class="modal-header">
                        <h4 class="modal-title">Edit Level</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Level Id</label>
                            <input type="number" name="LocationLevelId"
id="EditLocationLevelId" class="form-control" readonly>
                        </div>
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name"
id="EditLocationLevelName" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-info"
value="Save">
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Delete Modal HTML -->
    <div id="DeleteLocationLevel" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form id="DeleteLocationLevelForm" method="post">
                    <div class="modal-header">
                        <h4 class="modal-title">Delete Level</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <p>Are you sure you want to delete these Records?</p>
                        <p class="text-warning"><small>This action cannot be
undone.</small></p>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-danger"
value="Delete">
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>
```

```
<script>
    $('#crudLocationLevelsTable').DataTable({
        "paging": true,
        "searching": true,
        "ordering": true,
        "info": false,
        "responsive": true,
        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
        "destroy": true,
    });
</script>

<script>
    function EditLocationLevel(id) {
        $.ajax({
            url: "/LocationLevelsTable/GetLocationLevelById/" + id,
            type: "GET",
            success: function (result) {
                document.getElementById("EditLocationLevelId").value =
result.locationLevelId;
                document.getElementById("EditLocationLevelName").value =
result.name;
            }
        });
    }

    function DeleteLocationLevel(id) {
        document.getElementById('DeleteLocationLevelForm').action =
"LocationLevelsTable/DeleteLocationLevel/" + id;
    }
</script>

<script>
    (function () {
        if (window.location.hash.substr(1) == 'Edit')
            toastr.info("Congratulations! Your data has been successfully
edited.");
        if (window.location.hash.substr(1) == 'Delete')
            toastr.info("Congratulations! Your data has been successfully
deleted.");
        if (window.location.hash.substr(1) == 'Add')
            toastr.info("Congratulations! Your data has been successfully
added.");
    })();
</script>
```

AccessPointsTable -> Index.cshtml

```
@model IEnumerable<WebApp.Models.AccessPoint>

<link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
<script
src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"></
script>
<link rel="stylesheet" href="~/css/crudTables.css">

<!-- Main content -->
<section class="content">
    <div class="container-xl">
        <div class="table-responsive">
            <div class="table-wrapper">
                <div class="table-title">
```

```
                    <div class="row">
                        <div class="col-sm-6">
                            <h2>Manage <b>Access Points</b></h2>
                        </div>
                        <div class="col-sm-6">
                            <a href="#AddAccessPoint" class="btn btn-success"
data-toggle="modal"><i class="material-icons">&#xE147;</i> <span>Add New
Access Point</span></a>
                        </div>
                    </div>
                </div>
                <table id="crudAccessPointsTable" class="table table-striped
table-hover">
                    <thead>
                        <tr>
                            <th>Access Point Id</th>
                            <th>Name</th>
                            <th>Location Id</th>
                            <th>Access Type Id <br><p class="info-col">1-
IN/2-OUT<br>4-DBL/5-NON</p></th>
                            <th>Options</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach (var item in Model)
                        {
                            <tr>
                                <td>@item.AccessPointId</td>
                                <td>@item.Name</td>
                                <td>@item.LocationId</td>
                                <td>@item.AccessTypeId</td>
                                <td>
                                    <a href="#EditAccessPoint" class="edit"
data-toggle="modal"><i class="material-icons" data-toggle="tooltip"
title="Edit"
onclick="EditAccessPoint('@item.AccessPointId')">&#xE254;</i></a>
                                    <a href="#DeleteAccessPoint"
class="delete" data-toggle="modal"><i class="material-icons" data-
toggle="tooltip" title="Delete"
onclick="DeleteAccessPoint('@item.AccessPointId')">&#xE872;</i></a>
                                </td>
                            </tr>
                        }
                    </tbody>
                </table>
            </div>
        </div>
    </div>
    <!-- Add Modal HTML -->
    <div id="AddAccessPoint" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post"
action="AccessPointsTable/AddAccessPoint">
                    <div class="modal-header">
                        <h4 class="modal-title">Add Access Point</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Name</label>
```

```
                            <input type="text" name="Name" placeholder="Name"
class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Location Id</label>
                            <input type="number" name="LocationId"
placeholder="Location Id" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Access Type Id</label>
                            <label>(1-IN/2-OUT/4-DBL/5-NON)</label>
                            <input type="number" name="AccessTypeId"
placeholder="Access Type Id" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-success"
value="Add">
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Edit Modal HTML -->
    <div id="EditAccessPoint" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form method="post"
action="AccessPointsTable/EditAccessPoint">
                    <div class="modal-header">
                        <h4 class="modal-title">Edit Access Point</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <div class="form-group">
                            <label>Access Point Id</label>
                            <input type="number" name="AccessPointId"
id="EditAccessPointId" class="form-control" readonly>
                        </div>
                        <div class="form-group">
                            <label>Name</label>
                            <input type="text" name="Name"
id="EditAccessPointName" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Location Id</label>
                            <input type="number" name="LocationId"
id="EditAccPointLocationId" class="form-control" required>
                        </div>
                        <div class="form-group">
                            <label>Access Type Id</label>
                            <input type="number" name="AccessTypeId"
id="EditAccessTypeId" class="form-control" required>
                        </div>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-info"
value="Save">
```

```
                    </div>
                </form>
            </div>
        </div>
    </div>
    <!-- Delete Modal HTML -->
    <div id="DeleteAccessPoint" class="modal fade">
        <div class="modal-dialog">
            <div class="modal-content">
                <form id="DeleteAccessPointForm" method="post">
                    <div class="modal-header">
                        <h4 class="modal-title">Delete Access Point</h4>
                        <button type="button" class="close" data-
dismiss="modal" aria-hidden="true">&times;</button>
                    </div>
                    <div class="modal-body">
                        <p>Are you sure you want to delete these Records?</p>
                        <p class="text-warning"><small>This action cannot be
undone.</small></p>
                    </div>
                    <div class="modal-footer">
                        <input type="button" class="btn btn-default" data-
dismiss="modal" value="Cancel">
                        <input type="submit" class="btn btn-danger"
value="Delete">
                    </div>
                </form>
            </div>
        </div>
    </div>
</section>

<script>
    $('#crudAccessPointsTable').DataTable({
        "paging": true,
        "searching": true,
        "ordering": true,
        "info": false,
        "responsive": true,
        "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
        "destroy": true,
    });
</script>

<script>
    function EditAccessPoint(id) {
        $.ajax({
            url: "/AccessPointsTable/GetAccessPointById/" + id,
            type: "GET",
            success: function (result) {
                document.getElementById("EditAccessPointId").value =
result.accessPointId;
                document.getElementById("EditAccessPointName").value =
result.name;
                document.getElementById("EditAccPointLocationId").value =
result.locationId;
                document.getElementById("EditAccessTypeId").value =
result.accessTypeId;
            }
        });
    }

    function DeleteAccessPoint(id) {
```

```
        document.getElementById('DeleteAccessPointForm').action =
"AccessPointsTable/DeleteAccessPoint/" + id;
    }
</script>

<script>
    (function () {
        if (window.location.hash.substr(1) == 'Edit')
            toastr.info("Congratulations! Your data has been successfully
edited.");
        if (window.location.hash.substr(1) == 'Delete')
            toastr.info("Congratulations! Your data has been successfully
deleted.");
        if (window.location.hash.substr(1) == 'Add')
            toastr.info("Congratulations! Your data has been successfully
added.");
    })();
</script>
```

Home -> Index.cshtml

```
<!-- Main content -->
<section class="content">
    <div class="container-fluid">
        <!-- Small boxes (Stat box) -->
        <div class="row" id="infoBoxes">
        </div>
        <!-- /.row (main row) -->
        <!-- Main row -->
        <div class="row">
            <!-- Left col -->
            <section class="col-lg-7 connectedSortable">
                @await Html.PartialAsync("_WeekVisitors")
                @await Html.PartialAsync("_Calendar")
                @await Html.PartialAsync("_CurrentDayGuestsFlow")
            </section>
            <!-- /.Left col -->
            <!-- right col -->
            <section class="col-lg-5 connectedSortable">
                @await Html.PartialAsync("_Donut")
                @await Html.PartialAsync("_TopLocationsLive")
                @await Html.PartialAsync("_LocationsTable")
            </section>
            <!-- /.right col -->
        </div>
    </div><!-- /.container-fluid -->
</section>

<script type="text/javascript">
    function updateUsers() {
        setInterval(findUsers, 3000);
        function findUsers() {
            $.ajax({
                url: "/Home/ShowBoxes",
                type: "GET",
                success: function (result) {
                    var colSec = '<div class="col-lg-3 col-6">';
                    var locationsBox =
                        colSec +
                        '<div class="small-box bg-info">' +
                        '<div class="inner">' +
                        '<h3>' + result.locationsCount + '</h3>' +
                        '<p>Locations</p>' +
```

```
                        '</div>' +
                        '<div class="icon">' +
                        '<i class="ion ion-bag"></i>' +
                        '</div>' +
                        '</div>' +
                        '</div>';
                    var currentDayGuestsBox =
                        colSec +
                        '<div class="small-box bg-success">' +
                        '<div class="inner">' +
                        '<h3>' + result.currentDayGuestsCount + '</h3>' +
                        '<p>Today\'s guests</p>' +
                        '</div>' +
                        '<div class="icon">' +
                        '<i class="ion ion-person-add"></i>' +
                        '</div>' +
                        '</div>' +
                        '</div>';
                    var actualGuestsBox =
                        colSec +
                        '<div class="small-box bg-danger">' +
                        '<div class="inner">' +
                        '<h3>' + result.actualGuestsCount + '</h3>' +
                        '<p>Guests Now</p>' +
                        '</div>' +
                        '<div class="icon">' +
                        '<i class="ion ion-stats-bars"></i>' +
                        '</div>' +
                        '</div>' +
                        '</div>';
                    var OccupancyRateBox =
                        colSec +
                        '<div class="small-box bg-warning">' +
                        '<div class="inner">' +
                        '<h3>' + result.buildingOccupancyRate +
                        '<sup style="font-size: 20px">%</sup></h3>' +
                        '<p>Occupacy Rate</p>' +
                        '</div>' +
                        '<div class="icon">' +
                        '<i class="ion ion-pie-graph"></i>' +
                        '</div>' +
                        '</div>' +
                        '</div>';

                    var boxesCode = locationsBox + currentDayGuestsBox +
actualGuestsBox + OccupancyRateBox;
                    $('#infoBoxes').html(boxesCode);
                }
            });
        }
    }
    updateUsers();
</script>
```

Anexa 3: Codul sursă al aplicației mobile

AboutPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="SensorSimulation.Views.AboutPage"
             Title="About">

    <ContentPage.Resources>
        <ResourceDictionary>
            <Color x:Key="Accent">#96d1ff</Color>
        </ResourceDictionary>
    </ContentPage.Resources>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <StackLayout BackgroundColor="White" VerticalOptions="FillAndExpand"
HorizontalOptions="Fill">
            <StackLayout Orientation="Horizontal" HorizontalOptions="Center"
VerticalOptions="Center">
                <ContentView Padding="0,2,0,5"
VerticalOptions="FillAndExpand">
                    <Image Source="SensorsSimulationLogo.png"
VerticalOptions="Center" HeightRequest="100" WidthRequest="430"/>
                </ContentView>
            </StackLayout>
        </StackLayout>
        <ScrollView Grid.Row="1">
            <StackLayout Orientation="Vertical" Padding="30,24,30,24"
Spacing="10">
                <Label Text="Application description" FontSize="Title"/>
                <Label Text="This application was created in order to
simulate the monitoring sensors of people in the building. The data are sent
in real time and can be monitored in the dashboard of the web application."
FontSize="16" Padding="0,0,0,0"/>
                <Label FontSize="16" Padding="0,24,0,0">
                    <Label.FormattedText>
                        <FormattedString>
                            <FormattedString.Spans>
                                <Span Text="With this app you can send random
data to the random locations in real time. Also don't hesitate to use the
manual data sending, where you can choose a concrete sensor to send data."/>
                            </FormattedString.Spans>
                        </FormattedString>
                    </Label.FormattedText>
                </Label>
                <Label Text="Feel free to try each functionality. And of
course... Enjoy the using!" FontSize="16" Padding="0,0,0,0"/>
            </StackLayout>
        </ScrollView>
    </Grid>

</ContentPage>
```

DoubleSensorPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:tt="clr-namespace:TouchTracking"
             xmlns:skia="clr-
namespace:SkiaSharp.Views.Forms;assembly=SkiaSharp.Views.Forms"
             x:Class="SensorSimulation.Views.DoubleSensorPage"
             Title="{Binding Title}">
    <Grid>
        <Label x:Name ="AccessPointData" Text="NULL"/>
        <skia:SKCanvasView x:Name="canvasView"
PaintSurface="OnCanvasViewPaintSurface" />
        <Grid.Effects>
            <tt:TouchEffect Capture="True"
                            TouchAction="OnTouchEffectAction" />
        </Grid.Effects>
    </Grid>

</ContentPage>
```

DoubleSensorPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;
using SkiaSharp;
using SkiaSharp.Views.Forms;
using TouchTracking;
using System.Threading;
using SensorSimulation.Models;
using SensorSimulation.ViewModels;
using Acr.UserDialogs;
using WebApp.Services.DTOModels;

namespace SensorSimulation.Views
{
    [Obsolete]
    public partial class DoubleSensorPage : ContentPage
    {
        Dictionary<long, SKPath> inProgressPaths = new Dictionary<long,
SKPath>();
        List<SKPath> completedPaths = new List<SKPath>();
        SKBitmap saveBitmap;
        static double numberOfCrossingLine = 0;
        SKPoint sensorInLeft;
        SKPoint sensorInRight;
        int waitingForEntranceFlag = 0;
        int waitingForExitFlag = 0;
        SKPoint sensorOutLeft;
        SKPoint sensorOutRight;
        string indentOption = "\t\t\t\t\t\t\t\t\t\t\t\t\t\t\t";
        AccessPointDTO accessPointHandler;
        FormattedString formatedData = new FormattedString();
        Span passingData;

        SKPaint paint = new SKPaint
        {
            Style = SKPaintStyle.Stroke,
```

```
            StrokeWidth = 1,
            StrokeCap = SKStrokeCap.Square,
            StrokeJoin = SKStrokeJoin.Round,
            Color = SKColors.Black
        };

        SKPaint sensorOut = new SKPaint()
        {
            Style = SKPaintStyle.Fill,
            Color = SKColors.Red,
            StrokeWidth = 10
        };

        SKPaint sensorIn = new SKPaint()
        {
            Style = SKPaintStyle.Fill,
            Color = SKColors.Blue,
            StrokeWidth = 10
        };

        public DoubleSensorPage(AccessPointDTO accPoint)
        {
            InitializeComponent();
            accessPointHandler = accPoint;
            Title = accessPointHandler.Name;

            InitAccessPointData();
            GetCurrentDaySensorCrossed();
        }

        void InitAccessPointData()
        {
            var locationName = GlobalData.locViewModel.GetLocations().Where(x
=> x.LocationId == accessPointHandler.LocationId).Single();
            string accessType = "Double Access";

            var locationLevelName =
GlobalData.locLevelViewModel.GetLocationLevel(locationName.LocationLevelId).N
ame;
            formatedData.Spans.Add(new Span { Text = "Location: " +
locationName.Name + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });
            formatedData.Spans.Add(new Span { Text = "Level: " +
locationLevelName + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });
            formatedData.Spans.Add(new Span { Text = "Access Point: " +
accessPointHandler.Name + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });
            formatedData.Spans.Add(new Span { Text = "Type: " + accessType +
"\n", ForegroundColor = Color.DarkBlue, Font = Font.SystemFontOfSize(20) });

            AccessPointData.FormattedText = formatedData;
        }

        void OnCanvasViewPaintSurface(object sender, SKPaintSurfaceEventArgs
args)
        {
            SKImageInfo info = args.Info;
            SKSurface surface = args.Surface;
            SKCanvas canvas = surface.Canvas;

            // Create bitmap the size of the display surface
            if (saveBitmap == null)
```

```
            {
                saveBitmap = new SKBitmap(info.Width, info.Height);
            }
            // Or create new bitmap for a new size of display surface
            else if (saveBitmap.Width < info.Width || saveBitmap.Height <
info.Height)
            {
                SKBitmap newBitmap = new SKBitmap(Math.Max(saveBitmap.Width,
info.Width),
                                                  Math.Max(saveBitmap.Height,
info.Height));

                using (SKCanvas newCanvas = new SKCanvas(newBitmap))
                {
                    newCanvas.Clear();
                    newCanvas.DrawBitmap(saveBitmap, 0, 0);
                }

                saveBitmap = newBitmap;

            }

            // Render the bitmap
            canvas.Clear();
            canvas.DrawBitmap(saveBitmap, 0, 0);

            sensorInLeft = new SKPoint
            {
                X = -info.Width, //1080
                Y = info.Height / 2 //990
            };

            sensorInRight = new SKPoint
            {
                X = info.Width,
                Y = info.Height / 2
            };

            canvas.DrawLine(sensorInLeft, sensorInRight, sensorOut);

            sensorOutLeft = new SKPoint
            {
                X = -info.Width,
                Y = info.Height / 2 - 200
            };

            sensorOutRight = new SKPoint
            {
                X = info.Width,
                Y = info.Height / 2 - 200
            };

            canvas.DrawLine(sensorOutLeft, sensorOutRight, sensorIn);

        }

        void GetCurrentDaySensorCrossed()
        {
            var sensorData =
GlobalData.sensOutcomeViewModel.GetCurrentDaySensorOutcome().Where(x => (x !=
null)).ToList();

            /* For double sensor, we will have two pins assigned. */
```

```
            var modulePins =
GlobalData.modulePinViewModel.GetModulePins().Where(x => x.AccessPointId ==
accessPointHandler.AccessPointId);

            string message = "";

            /* Remove to update the upcomming line crossing */
            if (formatedData.Spans.Count > 4)
            {
                formatedData.Spans.RemoveAt(5);
                formatedData.Spans.RemoveAt(4);
            }

            if (modulePins.Count() == 2)
            {
                foreach(var mdPin in modulePins)
                {
                    var results = sensorData.Where(x => x.PinAssignedId ==
mdPin.PinAssignedId).Count();

                    if (mdPin.PassingTypeId == GlobalData.passingTypeExit)
                        message = "Total exits today: ";
                    if (mdPin.PassingTypeId ==
GlobalData.passingTypeEntrance)
                        message = "Total entries today: ";

                    passingData = new Span { Text = message +
results.ToString() + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) };
                    formatedData.Spans.Add(passingData);
                }

                AccessPointData.FormattedText = formatedData;
            }
        }

        void OnTouchEffectAction(object sender, TouchActionEventArgs args)
        {
            switch (args.Type)
            {
                case TouchActionType.Pressed:
                    if (!inProgressPaths.ContainsKey(args.Id))
                    {
                        SKPath path = new SKPath();

                        path.MoveTo(ConvertToPixel(args.Location));
                        inProgressPaths.Add(args.Id, path);
                        UpdateBitmap();
                    }
                    break;

                case TouchActionType.Moved:
                    if (inProgressPaths.ContainsKey(args.Id))
                    {
                        SKPath path = inProgressPaths[args.Id];
                        path.LineTo(ConvertToPixel(args.Location));
                        SKPoint currentPath = ConvertToPixel(args.Location);

                        /* Check the direction of path */
                        if (sensorOutLeft.Y - 10 <= (int)currentPath.Y &&
sensorOutLeft.Y + 10 >= (int)currentPath.Y)
                        {
                            /* Check the entrance */
```

```
                            if (sensorOutLeft.X <= currentPath.X &&
sensorOutRight.X >= currentPath.X)
                            {
                                numberOfCrossingLine += 0.5;

                                inProgressPaths.Remove(args.Id);
                                UpdateBitmap();
                                if (numberOfCrossingLine % 1 == 0 &&
numberOfCrossingLine != 0 && waitingForEntranceFlag == 1)
                                {

HandleAccessPointTransit(GlobalData.passingTypeEntrance);
                                    GetCurrentDaySensorCrossed();
                                    waitingForEntranceFlag = 0;
                                    numberOfCrossingLine = 0; // reset
crossing flag
                                }

                                if (Math.Abs(numberOfCrossingLine) % 1 != 0
&& waitingForEntranceFlag == 0)
                                {
                                    waitingForEntranceFlag = 1;
                                }
                            }
                        }
                        else if (sensorInLeft.Y - 10 <= (int)currentPath.Y &&
sensorInLeft.Y + 10 >= (int)currentPath.Y)
                        {
                            /* Check the exit */
                            if (sensorInLeft.X <= currentPath.X &&
sensorInRight.X >= currentPath.X)
                            {
                                numberOfCrossingLine -= 0.5;
                                inProgressPaths.Remove(args.Id);
                                if (Math.Abs(numberOfCrossingLine) % 1 == 0
&& numberOfCrossingLine != 0 && waitingForExitFlag == 1)
                                {

HandleAccessPointTransit(GlobalData.passingTypeExit);
                                    GetCurrentDaySensorCrossed();
                                    waitingForExitFlag = 0;
                                    numberOfCrossingLine = 0; // reset
crossing flag
                                }

                                if (Math.Abs(numberOfCrossingLine) % 1 != 0
&& waitingForExitFlag == 0)
                                {
                                    waitingForExitFlag = 1;
                                }
                            }
                        }
                        else
                        {
                            UpdateBitmap();
                        }
                    }
                    break;

                case TouchActionType.Released:
                    if (inProgressPaths.ContainsKey(args.Id))
                    {
                        completedPaths.Add(inProgressPaths[args.Id]);
```

```
                        inProgressPaths.Remove(args.Id);
                        UpdateBitmap();
                    }
                    break;

                case TouchActionType.Cancelled:
                    if (inProgressPaths.ContainsKey(args.Id))
                    {
                        inProgressPaths.Remove(args.Id);
                        UpdateBitmap();
                    }
                    break;
            }
        }

        void HandleAccessPointTransit(int passingType)
        {
            ConfirmNotification(passingType);

            var sensOut = new SensorOutcomeDTO();

            // get the module pin that matches the access point id of the
current acc point
            GlobalData.modulePinViewModel.GetModulePins().Select(x =>
x.AccessPointId == accessPointHandler.AccessPointId);
            sensOut.PinAssignedId =
GlobalData.modulePinViewModel.GetModulePins()
                                            .Where(x => (x.AccessPointId ==
accessPointHandler.AccessPointId)
                                            && (x.PassingTypeId ==
passingType)) /*check the passing type*/
                                            .Single()
                                            .PinAssignedId;

            /* First of all, if we want to exit(enter) a location, we need to
enter(exit) the hallway. */
            ManageVirtualHallway(passingType, sensOut);

            GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);
        }

        void ManageVirtualHallway(int passingType, SensorOutcomeDTO sensOut)
        {
            var hallwayAction = passingType == GlobalData.passingTypeEntrance
? GlobalData.passingTypeExit : GlobalData.passingTypeEntrance;

            ManageHallwayDTO manHallway = new ManageHallwayDTO
            {
                PinAssignedId = sensOut.PinAssignedId,
                HallwayAction = hallwayAction
            };


GlobalData.sensOutcomeViewModel.ManHallBeforePostSensorOutcome(manHallway);
        }

        void ConfirmNotification(int passingType)
        {
            double SecondsDuration = 1;
            string Message = "";
            Color textColor = Color.White;

            if (passingType == GlobalData.passingTypeExit)
```

```
            {
                Message = indentOption + "EXIT CONFIRMED!";
                textColor = Color.Red;
            }

            if (passingType == GlobalData.passingTypeEntrance)
            {
                Message = indentOption + "ENTRANCE CONFIRMED!";
                textColor = Color.Blue;
            }

            ToastConfig.DefaultBackgroundColor =
System.Drawing.Color.AliceBlue;
            ToastConfig.DefaultMessageTextColor = System.Drawing.Color.Red;
            ToastConfig.DefaultActionTextColor =
System.Drawing.Color.DarkRed;
            UserDialogs.Instance.Toast(new ToastConfig(Message)
                        .SetBackgroundColor(Color.Black)
                        .SetMessageTextColor(textColor)
                        .SetDuration(TimeSpan.FromSeconds(SecondsDuration))
                        .SetPosition(ToastPosition.Bottom));
        }

        void UpdateBitmap()
        {
            using (SKCanvas saveBitmapCanvas = new SKCanvas(saveBitmap))
            {
                saveBitmapCanvas.Clear();

                foreach (SKPath path in completedPaths)
                {
                    saveBitmapCanvas.DrawPath(path, paint);
                }

                foreach (SKPath path in inProgressPaths.Values)
                {
                    saveBitmapCanvas.DrawPath(path, paint);
                }
            }

            canvasView.InvalidateSurface();
        }

        SKPoint ConvertToPixel(Point pt)
        {
            return new SKPoint((float)(canvasView.CanvasSize.Width * pt.X /
canvasView.Width),
                               (float)(canvasView.CanvasSize.Height * pt.Y /
canvasView.Height));
        }
    }
}
```

RandomPostData.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="SensorSimulation.Views.RandomPostData"
             Title="Random Post Data">

    <Grid>
        <Grid Grid.Column="1"></Grid>
```

```
        <Grid Grid.Row="5"></Grid>
        <Label Text="Sent Data Counter : "  FontSize="Title"
TextColor="Black"/>
        <Label x:Name ="SentDataCounter"  Grid.Column="1" Grid.Row="0"
Text="0" FontSize="Title" TextColor="Black"/>
        <Button x:Name="StartButton" Grid.Column="0" Grid.Row="2"
Text="START" BackgroundColor="Blue" Clicked="OnStartButtonClicked"/>
        <Button x:Name="StopButton" Grid.Column="1" Grid.Row="2" Text="STOP"
BackgroundColor="Red" Clicked="OnStopButtonClicked" IsEnabled="False"/>
    </Grid>

</ContentPage>
```

RandomPostData.xaml.cs

```
using SensorSimulation.Models;
using SensorSimulation.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace SensorSimulation.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class RandomPostData : ContentPage
    {
        int dataCounter = 0;
        readonly List<LocationDTO> locations = new List<LocationDTO>();
        readonly List<AccessPointDTO> accessPoints = new
List<AccessPointDTO>();
        readonly List<ModulePinDTO> modulePins = new List<ModulePinDTO>();
        readonly List<LocationLevelDTO> levels = new
List<LocationLevelDTO>();
        Dictionary<int, uint> locationsGuestsNumber = new Dictionary<int,
uint>(); /* <int, int>, first int is for location's id and second for guest's
number */
        Dictionary<int?, uint> levelsGuestsNumber = new Dictionary<int?,
uint>(); /* <int, int>, first int is for level's id and second for guest's
number */
        List<SensorOutcomeDTO> sensorsInOutData = new
List<SensorOutcomeDTO>();
        SensorOutcomeDTO sensOut = new SensorOutcomeDTO();
        LocationDTO mainBuildingHallway = new LocationDTO();
        private CancellationTokenSource _canceller;

        const int arbitraryGuestsNumber = 10;
        const int arbitraryGuestsFromLvlNumber = 5;

        enum RandomCases
        {
            FirstTimeEntering = 0,
            ChangeLocationSameLevel,
            ChangeLevelChangeLocation,
            Exit,
            Default
        }
```

```
        public RandomPostData()
        {
            InitializeComponent();

            locations = GlobalData.locViewModel.GetLocations().ToList();
            accessPoints =
GlobalData.accPointViewModel.GetAccessPoints().ToList();
            modulePins =
GlobalData.modulePinViewModel.GetModulePins().ToList();
            levels =
GlobalData.locLevelViewModel.GetLocationLevels().ToList();
            mainBuildingHallway = locations.Where(x => x.Name.Contains("Mall
Access")).Single();
        }

        async void OnStartButtonClicked(object sender, EventArgs args)
        {
            StartButton.IsEnabled = false;
            StopButton.IsEnabled = true;
            StopButton.BackgroundColor = Color.Red;

            _canceller = new CancellationTokenSource();

            await Task.Run(() =>
           {
               do
               {
                   if (_canceller.Token.IsCancellationRequested)
                       break;
                   /* Update sensor's receiving data each time*/
                   sensorsInOutData =
GlobalData.sensOutcomeViewModel.GetCurrentDaySensorOutcome().Where(x => (x !=
null)).ToList();
                   var postResponse = RandomiseAction();

                   if ((postResponse >= HttpStatusCode.OK) && (postResponse
<= GlobalData.httpLastPositiveResponse))
                   {
                       /* Got positive response, send another one */
                       dataCounter = dataCounter + 1;
                       Device.BeginInvokeOnMainThread(() => /* To be able to
write message in UI */
                       {
                           SentDataCounter.Text = dataCounter.ToString(); //
update Data Counter
                       });
                   }
               } while (true);
           });

            _canceller.Dispose();
            StartButton.IsEnabled = true;
            StartButton.BackgroundColor = Color.Blue;
            StopButton.IsEnabled = false;
        }

        void OnStopButtonClicked(object sender, EventArgs args)
        {
            _canceller.Cancel();
        }

        void GetEachLocationNumberOfGuests()
```

```
        {
            List<ModulePinDTO> locationModulePins = new List<ModulePinDTO>();
            levelsGuestsNumber.Clear();
            locationsGuestsNumber.Clear();

            foreach (var location in locations)
            {
                var locationAccPoints = accessPoints.Where(x => x.LocationId
== location.LocationId).ToList();
                var numberOfEntraces = 0u;
                var numberOfExits = 0u;
                var locGuestsCount = 0u;

                foreach (var accessPoint in locationAccPoints)
                {
                    foreach (var modPin in modulePins)
                    {
                        if (modPin.AccessPointId ==
accessPoint.AccessPointId)
                        {
                            locationModulePins.Add(modPin);
                        }
                    }
                }

                foreach (var locModPin in locationModulePins)
                {
                    if (locModPin.PassingTypeId ==
GlobalData.passingTypeEntrance)
                    {
                        numberOfEntraces += (uint)sensorsInOutData.Where(x =>
x.PinAssignedId == locModPin.PinAssignedId).Count();
                    }

                    if (locModPin.PassingTypeId ==
GlobalData.passingTypeExit)
                        numberOfExits += (uint)sensorsInOutData.Where(x =>
x.PinAssignedId == locModPin.PinAssignedId).Count();
                }

                locGuestsCount = numberOfEntraces - numberOfExits;
                locationsGuestsNumber.Add(location.LocationId,
locGuestsCount);


                /* Also calculate guests from each level */
                if (levelsGuestsNumber.ContainsKey(location.LocationLevelId))
                    levelsGuestsNumber[location.LocationLevelId] +=
locGuestsCount;
                else
                    levelsGuestsNumber.Add(location.LocationLevelId,
locGuestsCount);

                locationModulePins.Clear();
                locGuestsCount = 0u;
                numberOfEntraces = 0u;
                numberOfExits = 0u;
            }
        }

        HttpStatusCode PostRandomAccPntData(int locationId, int
passingTypeId)
        {
```

```
            List<int> locWithPassTypePinsId = new List<int>();
            HttpStatusCode sendingDataStatus = 0;

            var location = locations.Where(x => x.LocationId ==
locationId).Single();
            int pinRandomId = -1;

            /* Get pins of location that corresponde with passing type id*/
            var locationAccessPoints = accessPoints.Where(x => (x.LocationId
== location.LocationId) && (!x.Name.Contains("Common Entry")) &&
(!x.Name.Contains("Hol Main")));

            foreach (var accessPoint in locationAccessPoints)
            {
                foreach (var modPin in modulePins)
                {
                    if ((modPin.AccessPointId == accessPoint.AccessPointId)
&& (modPin.PassingTypeId == passingTypeId))
                    {
                        locWithPassTypePinsId.Add(modPin.PinAssignedId);
                    }
                }
            }

            /* Chose a random access point */
            if (locWithPassTypePinsId.Count() > 0)
            {
                pinRandomId =
locWithPassTypePinsId.Shuffle().Take(1).Single();
                sensOut.PinAssignedId = pinRandomId;
                sendingDataStatus =
GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);

                /* Update sensor's receiving data each time*/
                return sendingDataStatus;
            }

            sendingDataStatus = HttpStatusCode.NoContent;
            return sendingDataStatus;
        }

        /* isLocation dissociate the cases when a guest make operations
inside
         * a level(enter/exit a location and access hallway) or he enters or
exits
         * the current level from another level(enter or exit throw hallway
access points).
         * If isLocation is on false, then the scope is to make operations
using Common Entries.
         * If isLocation is on true, then the real hallway entrances are
used. */
        HttpStatusCode EnterOrExitHallOfLevel(int levelId, int passingType,
bool isLocation)
        {
            HttpStatusCode sendingDataStatus = 0;

            var HallLevelLocation = locations.Where(x => (x.LocationLevelId
== levelId) && (x.Name.Contains("Hol"))).Single();

            if (isLocation == false)
            {
```

```
                var accPoint = accessPoints.Where(a =>
(a.Name.Contains("Common Entry")) && (a.LocationId ==
HallLevelLocation.LocationId)).Single();
                if (accPoint != null)
                {
                    var enterPin = modulePins.Where(m => (m.AccessPointId ==
accPoint.AccessPointId) && (m.PassingTypeId == passingType)).Single();

                    sensOut.PinAssignedId = enterPin.PinAssignedId;

                    sendingDataStatus =
GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);
                }
            }

            if (isLocation == true)
            {
                var availableAccPoints = new List<AccessPointDTO>();

                foreach (var accPoint in accessPoints.Where(a =>
(a.AccessTypeId == passingType) || (a.AccessTypeId ==
GlobalData.locationWithDoubleSensor)).ToList())
                {
                    if ((accPoint.LocationId == HallLevelLocation.LocationId)
&& (!accPoint.Name.Contains("Common Entry") && (!accPoint.Name.Contains("Hol
Main"))))
                    {
                        availableAccPoints.Add(accPoint);
                    }
                }

                if (availableAccPoints.Any())
                {
                    var randAvailableAccPoint =
availableAccPoints.Shuffle().Take(1).Single();
                    var enterPin = modulePins.Where(m => (m.AccessPointId ==
randAvailableAccPoint.AccessPointId) && (m.PassingTypeId ==
passingType)).Single();

                    sensOut.PinAssignedId = enterPin.PinAssignedId;
                    sendingDataStatus =
GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);
                }
            }

            return sendingDataStatus;
        }

        HttpStatusCode PassLocationFromLevel(int level, int passingTypeId)
        {
            /* Get the location ids with minimum 1 guest */
            List<int> populatedLocationsId = new List<int>();
            HttpStatusCode sendingDataStatus = 0;
            int locationRandomId = -1;

            foreach (var entry in locationsGuestsNumber)
            {
                if (entry.Value != 0)
                    populatedLocationsId.Add(entry.Key);
            }

            var hallFromSelectedLevel = locations.Where(x =>
(x.Name.Contains("Hol")) && (x.LocationLevelId == level)).Single();
```

```
            var locationsFromSelectedLevel = locations.Where(x =>
(x.LocationLevelId == level) && (hallFromSelectedLevel.LocationId !=
x.LocationId) && (x.CapacityM2 != 0)).ToList();

            var selectedLocations = new List<LocationDTO>();

            if (passingTypeId == GlobalData.passingTypeExit)
            {
                foreach (var location in locationsFromSelectedLevel)
                {
                    if (populatedLocationsId.Contains(location.LocationId))
                    {
                        selectedLocations.Add(location); /* Get location DTO
object */
                    }
                }
            }
            else /* the passing type is Entrance */
            {
                selectedLocations = locationsFromSelectedLevel.ToList();
            }

            /* Choose a random location on current level from where a guest
is able to leave excepting hall. For this we handle apart case. */
            if (selectedLocations.Count() > 0)
            {
                locationRandomId = selectedLocations.Select(x =>
x.LocationId).Shuffle().Take(1).Single();
                /* Chose a random access point to leave or enter the location
*/
                sendingDataStatus = PostRandomAccPntData(locationRandomId,
passingTypeId);
            }

            return sendingDataStatus;
        }

        HttpStatusCode GuestFirstTimeEnter()
        {
            HttpStatusCode sendingDataStatus = 0;
            int pinIdAssignedToEntrie = -1;

            /* Here can be more building's doors from outside, chose one
randomly */
            var entrieAccessPoint = accessPoints.Where(x => (x.LocationId ==
mainBuildingHallway.LocationId) &&
                                                            (x.Name.Contains("Hol
Main")) &&
                                                           ((x.AccessTypeId ==
GlobalData.locationWithDoubleSensor) ||
                                                           (x.AccessTypeId ==
GlobalData.locationWithEntranceSensor))).Select(x => x.AccessPointId);

            var randomEntrieId =
entrieAccessPoint.Shuffle().Take(1).Single();

            foreach (var modPin in modulePins)
            {
                if ((modPin.AccessPointId == randomEntrieId) &&
(modPin.PassingTypeId == GlobalData.passingTypeEntrance))
                {
                    pinIdAssignedToEntrie = modPin.PinAssignedId;
```

```
                    /* Now a new guest entered in the building. */
                    sensOut.PinAssignedId = pinIdAssignedToEntrie;
                    sendingDataStatus =
GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);
                    return sendingDataStatus;
                }
            }
            return sendingDataStatus;
        }

        HttpStatusCode ExitBuilding()
        {
            HttpStatusCode sendingDataStatus = 0;
            List<int> populatedLocationsId = new List<int>();
            var selectedLocations = new List<int>();
            int idOfExitLevel = 0;

            /* Ger a random populated level, a random populated location and
get out a guest from building. */
            if (levelsGuestsNumber.Where(k => (k.Value >
arbitraryGuestsNumber)).Any())
            {
                foreach (var entry in locationsGuestsNumber)
                {
                    if (entry.Value != 0)
                        populatedLocationsId.Add(entry.Key);
                }

                idOfExitLevel = (int)levelsGuestsNumber.Where(k => k.Value !=
0).Select(k => k.Key).ToList().Shuffle().Take(1).Single();

                foreach(var loc in locations)
                {
                    if(loc.LocationLevelId == idOfExitLevel &&
populatedLocationsId.Contains(loc.LocationId) && (!loc.Name.Contains("Hol")))
                    {
                        selectedLocations.Add((int)loc.LocationId);
                    }
                }

                if(selectedLocations.Any())
                {
                    var randPopulatedLocation =
selectedLocations.Shuffle().Take(1).Single();

                    if (randPopulatedLocation != 0) /* Was selected a valid
location id */
                    {
                        /* Chose a random access point of the location and
leave it. */
                        sendingDataStatus =
PostRandomAccPntData(randPopulatedLocation, GlobalData.passingTypeExit);

                        /* Enter the hallway by exiting the location */
                        EnterOrExitHallOfLevel(idOfExitLevel,
GlobalData.passingTypeEntrance, false);

                        if (idOfExitLevel !=
mainBuildingHallway.LocationLevelId)
                        {
                            /* Exit the hallway from current level to get on
the main level and to leave the building */
```

```
                            EnterOrExitHallOfLevel(idOfExitLevel,
GlobalData.passingTypeExit, true);

                            /* Enter the main hallway to leave the building
*/

EnterOrExitHallOfLevel((int)mainBuildingHallway.LocationLevelId,
GlobalData.passingTypeEntrance, true);
                        }

                        var availableAccPoints = accessPoints.Where(a =>
((a.AccessTypeId == GlobalData.passingTypeExit) || (a.AccessTypeId ==
GlobalData.locationWithDoubleSensor)) && (a.Name.Contains("Hol
Main"))).ToList();

                        if (availableAccPoints.Any())
                        {
                            var randAvailableAccPoint =
availableAccPoints.Shuffle().Take(1).Single();
                            var enterPin = modulePins.Where(m =>
(m.AccessPointId == randAvailableAccPoint.AccessPointId) && (m.PassingTypeId
== GlobalData.passingTypeExit)).Single();

                            sensOut.PinAssignedId = enterPin.PinAssignedId;
                            /* Leave the building */
                            sendingDataStatus =
GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);
                        }
                    }
                }
            }

            return sendingDataStatus;
        }

        HttpStatusCode LeaveMainHallwayEnterLocation()
        {
            HttpStatusCode sendingDataStatus = 0;

            /* Get a random level and bring here a guest. */
            if (levelsGuestsNumber.Where(k =>
(mainBuildingHallway.LocationLevelId == k.Key) && (k.Value >
arbitraryGuestsFromLvlNumber)).Any())
            {
                var idOfRandomLevel = levels.Select(l =>
l.LocationLevelId).Shuffle().Take(1).Single();

                if (idOfRandomLevel != mainBuildingHallway.LocationLevelId)
                {
                    /* Leave hallway from main level. */

EnterOrExitHallOfLevel((int)mainBuildingHallway.LocationLevelId,
GlobalData.passingTypeExit, true);

                    /*Enter the real entrance of the new level.*/
                    EnterOrExitHallOfLevel(idOfRandomLevel,
GlobalData.passingTypeEntrance, true);

                    /* Leave the current hallway to enter a location. Now the
isLocation flag will be on false. */
                    EnterOrExitHallOfLevel(idOfRandomLevel,
GlobalData.passingTypeExit, false);
```

```
                    sendingDataStatus =
PassLocationFromLevel(idOfRandomLevel, GlobalData.passingTypeEntrance);
                }
                else                    /* If the choosed is the main level,
just enter a location from the level. */
                {
                    /* Leave hallway from main level. */

EnterOrExitHallOfLevel((int)mainBuildingHallway.LocationLevelId,
GlobalData.passingTypeExit, false);
                    sendingDataStatus =
PassLocationFromLevel((int)mainBuildingHallway.LocationLevelId,
GlobalData.passingTypeEntrance);
                }
            }

            return sendingDataStatus;
        }

        HttpStatusCode RandomiseAction()
        {
            int[] probabilityRange = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
            HttpStatusCode sendingDataStatus = 0;
            var randomAction = probabilityRange.Shuffle().Take(1).Single();
            var probabilityDistribution = (randomAction >= 0 && randomAction
<= 3) ? RandomCases.FirstTimeEntering :
                                          (randomAction >= 4 && randomAction
<= 5) ? RandomCases.ChangeLocationSameLevel :
                                          (randomAction >= 6 && randomAction
<= 8) ? RandomCases.ChangeLevelChangeLocation :
                                          (randomAction == 9) ?
RandomCases.Exit : RandomCases.Default;

            GetEachLocationNumberOfGuests();

            var levelsIdWithGuests = levelsGuestsNumber.Where(x => (x.Value
!= 0));

            switch (probabilityDistribution)
            {
                case RandomCases.FirstTimeEntering:  // 40% probability to
reach
                    sendingDataStatus = GuestFirstTimeEnter();
                    //case RandomCases.PopulateLocations:
                    LeaveMainHallwayEnterLocation();
                    break;
                case RandomCases.ChangeLocationSameLevel: // 20% probability
to reach
                    /* Chose a random level, from where a guest can leave */
                    if (levelsIdWithGuests.Count() != 0)
                    {
                        var levelsWithNumberOfGuests =
levelsIdWithGuests.Where(x => x.Value > 5).ToList();
                        if (levelsWithNumberOfGuests.Count != 0)
                        {
                            int currentLevel =
levelsWithNumberOfGuests.Select(x => (int)x.Key).Shuffle().Take(1).Single();
                            if (currentLevel != 0)
                            {
                                HttpStatusCode enterHallStatus = 0;
                                /* Change location from same level.*/
                                sendingDataStatus =
PassLocationFromLevel(currentLevel, GlobalData.passingTypeExit);
```

```
                                /* That means we got a location from the
level where the guest can leave. */
                                if (sendingDataStatus >= HttpStatusCode.OK &&
sendingDataStatus <= GlobalData.httpLastPositiveResponse)
                                {
                                    /* Enter the hallway by exiting a
location. */
                                    enterHallStatus =
EnterOrExitHallOfLevel(currentLevel, GlobalData.passingTypeEntrance, false);
                                }

                                if (((sendingDataStatus >= HttpStatusCode.OK
&& sendingDataStatus <= GlobalData.httpLastPositiveResponse) &&
                                    (enterHallStatus >= HttpStatusCode.OK &&
enterHallStatus <= GlobalData.httpLastPositiveResponse)) ||
                                    (sendingDataStatus == 0)) /* Appart case
is when the sendingDataStatus is 0. That means that locations from this level
are empty. We can directly leave the hall.*/
                                {
                                    EnterOrExitHallOfLevel(currentLevel,
GlobalData.passingTypeExit, false);
                                    sendingDataStatus =
PassLocationFromLevel(currentLevel, GlobalData.passingTypeEntrance);
                                }
                            }
                        }
                    }

                    break;
                case RandomCases.ChangeLevelChangeLocation: // 30%
probability to reach
                    if (levelsIdWithGuests.Count() != 0)
                    {
                        /* Get a level with guests where anyone can leave. */
                        int currentLevel = levelsIdWithGuests.Select(x =>
(int)x.Key).Shuffle().Take(1).Single();

                        sendingDataStatus =
PassLocationFromLevel(currentLevel, GlobalData.passingTypeExit);

                        if (sendingDataStatus >= HttpStatusCode.OK &&
sendingDataStatus <= GlobalData.httpLastPositiveResponse)
                        {
                            /* Enter the hallway from a current location
leaving. */
                            /* If this line of code is not executed, that
means here are no locations with guests on this level. */
                            EnterOrExitHallOfLevel(currentLevel,
GlobalData.passingTypeEntrance, false);

                            /* Here we don't need to track the current level,
will be set a random one */
                            /* Get a random location level */
                            var randLocationLevel = levels.Where(l =>
l.LocationLevelId != currentLevel).Select(x =>
x.LocationLevelId).Shuffle().Take(1).Single();

                            /* Get out of the hall on the previous level to
enter again a hallway from another level. */
                            /* Exit the hallway only if previously a guest
entered it by exit a location. */
                            EnterOrExitHallOfLevel(currentLevel,
GlobalData.passingTypeExit, true);
```

```
                            /* Enter hallway from the another level. */
                            EnterOrExitHallOfLevel(randLocationLevel,
GlobalData.passingTypeEntrance, true);

                            EnterOrExitHallOfLevel(randLocationLevel,
GlobalData.passingTypeExit, false);
                            sendingDataStatus =
PassLocationFromLevel(randLocationLevel, GlobalData.passingTypeEntrance);
                        }
                    }

                    break;
                case RandomCases.Exit: // 10% probability to reach
                    if (levelsIdWithGuests.Count() != 0)
                    {
                        sendingDataStatus = ExitBuilding();
                    }
                    break;
                default:
                    break;
            }

            return sendingDataStatus;
        }
    }
}
```

SensorPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage
    x:Class="SensorSimulation.Views.SensorsPage"
    xmlns="http://xamarin.com/schemas/2014/forms"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    Title="Sensors Page">
    <ContentPage.Content>
        <Grid BackgroundColor="DarkGray">
            <StackLayout Margin="10"
                         VerticalOptions="Start"
                         HorizontalOptions="Start">
                <SearchBar x:Name="searchBar"
                   HorizontalOptions="Fill"
                   VerticalOptions="StartAndExpand"
                   Placeholder="Search Access Points..."
                   CancelButtonColor="Black"
                   PlaceholderColor="Black"
                   TextTransform="Lowercase"
                   HorizontalTextAlignment="Start"
                   TextChanged="OnTextChanged"/>

                <Picker x:Name="picker_level"
                        Title="Select a level"
                        TitleColor="Black"
                        ItemsSource="{Binding Items}"
                        ItemDisplayBinding="{Binding Name}"

SelectedIndexChanged="Picker_SelectedIndexChangedLcLevel"
                        IsVisible="True"/>
                <Picker x:Name="picker_location"
                        TitleColor="Black"
                        IsEnabled="False"
                        ItemDisplayBinding="{Binding Name}"
```

```
SelectedIndexChanged="Picker_SelectedIndexChangedLocation"
                        IsVisible="True"/>
                <Picker x:Name="picker_accessPoint"
                        TitleColor="Black"
                        IsEnabled="False"
                        ItemDisplayBinding="{Binding Name}"

SelectedIndexChanged="Picker_SelectedIndexChangedAccPoint"
                        IsVisible="True"/>

                <Label Text="Type in the searchbox to filter results in
realtime."
               HorizontalOptions="Fill"
               VerticalOptions="CenterAndExpand" />
                <ListView x:Name="searchResults"
                  HorizontalOptions="Fill"
                  VerticalOptions="CenterAndExpand"
                          IsVisible="False"
                          ItemTapped="GetTappedItem"/>
            </StackLayout>
        </Grid>
    </ContentPage.Content>

</ContentPage>
```

SensorPage.xaml.cs

```
using SensorSimulation.Models;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;
using SensorSimulation.Services;
using SensorSimulation.ViewModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;
using Acr.UserDialogs;

namespace SensorSimulation.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    public partial class SensorsPage : ContentPage
    {
        public ICommand NavigateCommand { get; set; }
        LocationLevelViewModel locLevelViewModel = new
LocationLevelViewModel();

        LocationDTO selectedLocationFromPicker;
        string tappedItem = "";

        public SensorsPage()
        {
            InitializeComponent();
            BindingContext = locLevelViewModel;
            searchResults.ItemsSource =
DataService.GetSearchResults(searchBar.Text);
        }
```

```
        private void Picker_SelectedIndexChangedLcLevel(object sender,
EventArgs e)
        {
            var _picker = sender as Picker;
            var locLvlCollection = locLevelViewModel.Items;
            var index = _picker.SelectedIndex;

            if (picker_accessPoint.IsEnabled && picker_location.IsEnabled) /*
We want to select another, reset other pickers */
            {
                ResetLocationAndAccessPointPicker();
            }

            if (index != -1)
            {
                if (!picker_location.IsEnabled)
                {
                    picker_location.IsEnabled = true;
                    picker_location.IsVisible = true;
                    picker_location.Title = "Select Location";
                }

                picker_location.ItemsSource = new
ObservableCollection<LocationDTO>(locLvlCollection[index].Locations.Where(l
=> (!l.Name.Contains("Hol") || (l.Name.Contains("Hol Mall Access")))));

            }
        }

        private void Picker_SelectedIndexChangedLocation(object sender,
EventArgs e)
        {
            var _picker = sender as Picker;
            var loca = new LocationViewModel();
            var locationCollection = loca.Items;
            var index = _picker.SelectedIndex;

            if (index != -1)
            {
                var nameOfLocationFromPicker = _picker.Items[index];
                selectedLocationFromPicker =
loca.GetLocationByName(nameOfLocationFromPicker);

                List<AccessPointDTO> locWithSensors = new
List<AccessPointDTO>();

                if (selectedLocationFromPicker.Name.Contains("Hol Mall
Access"))
                {
                    locWithSensors =
selectedLocationFromPicker.AccessPoints.Where(x => x.Name.Contains("Hol
Main")).ToList();
                }
                else
                {
                    locWithSensors =
selectedLocationFromPicker.AccessPoints.Where(x => (x.AccessTypeId !=
GlobalData.locationWithNoSensors)).ToList();
                }

                if (!locWithSensors.Any())
                {
                    picker_accessPoint.IsEnabled = false;
```

```
                    picker_accessPoint.IsVisible = false;
                    string Message = "This location does not support any
sensors!";
                    RiseNotification(Message);
                }
                else
                {
                    if (!picker_accessPoint.IsEnabled)
                    {
                        picker_accessPoint.IsEnabled = true;
                        picker_accessPoint.Title = "Select the Access Point";
                        picker_accessPoint.IsVisible = true;
                    }

                    picker_accessPoint.ItemsSource = new
ObservableCollection<AccessPointDTO>(locWithSensors);
                }
            }
        }

        [Obsolete]
        private void Picker_SelectedIndexChangedAccPoint(object sender,
EventArgs e)
        {
            var _picker = sender as Picker;
            var index = _picker.SelectedIndex;
            if (index != -1)
            {
                var getSelectedPickerItem =
picker_accessPoint.Items[index].ToString();
                var accPoint =
((ObservableCollection<AccessPointDTO>)picker_accessPoint.ItemsSource).Where(
x => x.Name == getSelectedPickerItem).Single();

                RedirectToSensorPage(accPoint);
            }
        }

        void ResetLocationAndAccessPointPicker()
        {
            picker_location.Title = "";
            picker_location.IsVisible = false;
            picker_location.IsEnabled = false;
            picker_location.SelectedIndex = -1;

            picker_accessPoint.Title = "";
            picker_accessPoint.IsVisible = false;
            picker_accessPoint.IsEnabled = false;
            picker_accessPoint.SelectedIndex = -1;
        }

        void ResetPickerParameters()
        {
            picker_level.Title = "Select a level";
            picker_level.IsVisible = true;
            picker_level.IsEnabled = true;
            picker_level.SelectedIndex = -1;

            ResetLocationAndAccessPointPicker();

        }

        [Obsolete]
```

```
        async void RedirectToSensorPage(AccessPointDTO accessPoint)
        {
            if (accessPoint.AccessTypeId ==
GlobalData.locationWithDoubleSensor)
            {
                await Navigation.PushAsync(new
DoubleSensorPage(accessPoint));
                ResetPickerParameters();
            }

            if (accessPoint.AccessTypeId ==
GlobalData.locationWithEntranceSensor || accessPoint.AccessTypeId ==
GlobalData.locationWithExitSensor)
            {
                await Navigation.PushAsync(new
SingleSensorPage(accessPoint));
                ResetPickerParameters();
            }

            if (accessPoint.AccessTypeId == GlobalData.locationWithNoSensors)
            {
                string Message = "This location does not support any
sensors!";
                RiseNotification(Message);
                ResetPickerParameters();
            }
        }

        void RiseNotification(string Message)
        {
            double SecondsDuration = 1;
            ToastConfig.DefaultBackgroundColor =
System.Drawing.Color.AliceBlue;
            ToastConfig.DefaultMessageTextColor = System.Drawing.Color.Red;
            ToastConfig.DefaultActionTextColor =
System.Drawing.Color.DarkRed;
            UserDialogs.Instance.Toast(new ToastConfig(Message)
                        .SetBackgroundColor(Color.Red)
                        .SetMessageTextColor(Color.LightGoldenrodYellow)
                        .SetDuration(TimeSpan.FromSeconds(SecondsDuration))
                        .SetPosition(ToastPosition.Bottom));
        }

        [Obsolete]
        void GetTappedItem(object sender, ItemTappedEventArgs e)
        {
            tappedItem = e.Item.ToString();
            var acu = new AccessPointViewModel();
            if (tappedItem.Any())
            {
                var findAccessPoint = acu.GetAccessPoints().Where(x => x.Name
== tappedItem).ToList();
                if (findAccessPoint.Count == 1)
                {
                    searchResults.IsVisible = false;
                    searchResults.SelectedItem = "None";
                    searchBar.Text = "";
                    RedirectToSensorPage(findAccessPoint.Single());
                }
            }
        }
        void OnTextChanged(object sender, TextChangedEventArgs e)
        {
```

```
            ResetPickerParameters();

            picker_level.IsVisible = false;
            picker_level.IsEnabled = false;

            searchResults.IsVisible = true;
            searchResults.ItemsSource =
DataService.GetSearchResults(e.NewTextValue);

            if (e.NewTextValue == string.Empty) /* When was deleted all
characters from search bar */
            {
                ResetPickerParameters();
                searchResults.IsVisible = false;
            }
        }
    }
}
```

SingleSensorPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:tt="clr-namespace:TouchTracking"
             xmlns:skia="clr-
namespace:SkiaSharp.Views.Forms;assembly=SkiaSharp.Views.Forms"
             x:Class="SensorSimulation.Views.SingleSensorPage"
             Title="{Binding Title}">
    <Grid>
        <Label x:Name ="AccessPointData" Text="NULL"/>

        <skia:SKCanvasView x:Name="canvasView"
PaintSurface="OnCanvasViewPaintSurface" />
        <Grid.Effects>
            <tt:TouchEffect Capture="True"
                            TouchAction="OnTouchEffectAction" />
        </Grid.Effects>
    </Grid>
</ContentPage>
```

SingleSensorPage.xaml.cs

```
using Acr.UserDialogs;
using SensorSimulation.Models;
using SensorSimulation.ViewModels;
using SkiaSharp;
using SkiaSharp.Views.Forms;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using TouchTracking;
using WebApp.Services.DTOModels;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace SensorSimulation.Views
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    [Obsolete]
```

```
    public partial class SingleSensorPage : ContentPage
    {
        static AccessPointDTO accessPointHandler;
        string indentOption = "\t\t\t\t\t\t\t\t\t\t\t\t\t\t";
        Dictionary<long, SKPath> inProgressPaths = new Dictionary<long,
SKPath>();
        static double numberOfCrossingLine = 0;
        FormattedString formatedData = new FormattedString();
        List<SKPath> completedPaths = new List<SKPath>();
        SKBitmap saveBitmap;
        SKPoint sensorLeft;
        SKPoint sensorRight;

        const int infoRows = 4;

        public SingleSensorPage(AccessPointDTO accPoint)
        {
            InitializeComponent();
            accessPointHandler = accPoint;
            Title = accessPointHandler.Name;

            InitAccessPointData();

            GetCurrentDaySensorCrossed();
        }

        SKPaint sensorDraw = new SKPaint()
        {
            Style = SKPaintStyle.Fill,
            Color = SKColors.Black,
            StrokeWidth = 10
        };

        SKPaint paint = new SKPaint
        {
            Style = SKPaintStyle.Stroke,
            StrokeWidth = 1,
            StrokeCap = SKStrokeCap.Round,
            StrokeJoin = SKStrokeJoin.Round,
            Color = SKColors.Black
        };

        void InitAccessPointData()
        {
            var locationName = GlobalData.locViewModel.GetLocations().Where(x
=> x.LocationId == accessPointHandler.LocationId).Single();

            string accessType = GetAccessTypeById();

            var locationLevelName =
GlobalData.locLevelViewModel.GetLocationLevel(locationName.LocationLevelId).N
ame;

            formatedData.Spans.Add(new Span { Text = "Location: " +
locationName.Name + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });
            formatedData.Spans.Add(new Span { Text = "Level: " +
locationLevelName + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });
            formatedData.Spans.Add(new Span { Text = "Access Point: " +
accessPointHandler.Name + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });
```

```
            formatedData.Spans.Add(new Span { Text = "Type: " + accessType +
"\n", ForegroundColor = Color.DarkBlue, Font = Font.SystemFontOfSize(20) });

            AccessPointData.FormattedText = formatedData;
        }

        string GetAccessTypeById()
        {
            switch (accessPointHandler.AccessTypeId)
            {
                case GlobalData.locationWithEntranceSensor:
                    return "Entrance";
                case GlobalData.locationWithExitSensor:
                    return "Exit";
                default:
                    return "";
            }
        }
        void SetSensorColorByAccessType()
        {
            if (accessPointHandler.AccessTypeId ==
GlobalData.passingTypeExit)
                sensorDraw.Color = SKColors.Red;
            if (accessPointHandler.AccessTypeId ==
GlobalData.passingTypeEntrance)
                sensorDraw.Color = SKColors.Blue;
        }

        void GetCurrentDaySensorCrossed()
        {
            var sensorData =
GlobalData.sensOutcomeViewModel.GetCurrentDaySensorOutcome().Where(x => (x !=
null)).ToList();

            /*get module pin, to have acces to the pin's sensor outcome*/
            var modulePin =
GlobalData.modulePinViewModel.GetModulePins().Where(x => x.AccessPointId ==
accessPointHandler.AccessPointId).Single();
            var results = sensorData.Where(x => x.PinAssignedId ==
modulePin.PinAssignedId).Count();

            string message = "";

            if (formatedData.Spans.Count > infoRows)
            {
                formatedData.Spans.RemoveAt(infoRows);
            }

            if (accessPointHandler.AccessTypeId ==
GlobalData.passingTypeExit)
                message = "Total exits today: ";
            if (accessPointHandler.AccessTypeId ==
GlobalData.passingTypeEntrance)
                message = "Total entries today: ";

            formatedData.Spans.Add(new Span { Text = message +
results.ToString() + "\n", ForegroundColor = Color.DarkBlue, Font =
Font.SystemFontOfSize(20) });

            AccessPointData.FormattedText = formatedData;
        }
```

```
        void OnCanvasViewPaintSurface(object sender, SKPaintSurfaceEventArgs
args)
        {
            SKImageInfo info = args.Info;
            SKSurface surface = args.Surface;
            SKCanvas canvas = surface.Canvas;

            // Create bitmap the size of the display surface
            if (saveBitmap == null)
            {
                saveBitmap = new SKBitmap(info.Width, info.Height);
            }
            // Or create new bitmap for a new size of display surface
            else if (saveBitmap.Width < info.Width || saveBitmap.Height <
info.Height)
            {
                SKBitmap newBitmap = new SKBitmap(Math.Max(saveBitmap.Width,
info.Width),
                                                  Math.Max(saveBitmap.Height,
info.Height));

                using (SKCanvas newCanvas = new SKCanvas(newBitmap))
                {
                    newCanvas.Clear();
                    newCanvas.DrawBitmap(saveBitmap, 0, 0);
                }

                saveBitmap = newBitmap;

            }

            // Render the bitmap
            canvas.Clear();
            canvas.DrawBitmap(saveBitmap, 0, 0);

            sensorLeft = new SKPoint
            {
                X = -info.Width, //1080
                Y = info.Height / 2 //990
            };

            sensorRight = new SKPoint
            {
                X = info.Width,
                Y = info.Height / 2
            };

            SetSensorColorByAccessType();

            canvas.DrawLine(sensorLeft, sensorRight, sensorDraw);

        }

        void OnTouchEffectAction(object sender, TouchActionEventArgs args)
        {
            switch (args.Type)
            {
                case TouchActionType.Pressed:
                    if (!inProgressPaths.ContainsKey(args.Id))
                    {
                        SKPath path = new SKPath();

                        path.MoveTo(ConvertToPixel(args.Location));
```

```
                        inProgressPaths.Add(args.Id, path);
                        UpdateBitmap();
                    }
                    break;

                case TouchActionType.Moved:
                    if (inProgressPaths.ContainsKey(args.Id))
                    {
                        SKPath path = inProgressPaths[args.Id];
                        path.LineTo(ConvertToPixel(args.Location));
                        SKPoint currentPath = ConvertToPixel(args.Location);

                        if ((sensorLeft.Y - 10 <= (int)currentPath.Y) &&
(sensorLeft.Y + 10 >= (int)currentPath.Y))
                        {
                            if (sensorLeft.X <= currentPath.X &&
sensorRight.X >= currentPath.X)
                            {
                                numberOfCrossingLine += 0.5;

                                inProgressPaths.Remove(args.Id);
                                UpdateBitmap();
                                if (numberOfCrossingLine % 1 == 0 &&
numberOfCrossingLine != 0)
                                {
                                    HandleAccessPointTransit();
                                    GetCurrentDaySensorCrossed();
                                    numberOfCrossingLine = 0; // reset
crossing flag
                                }
                            }
                        }
                        else
                        {
                            UpdateBitmap();
                        }
                    }
                    break;

                case TouchActionType.Released:
                    if (inProgressPaths.ContainsKey(args.Id))
                    {
                        completedPaths.Add(inProgressPaths[args.Id]);
                        inProgressPaths.Remove(args.Id);
                        UpdateBitmap();
                    }
                    break;

                case TouchActionType.Cancelled:
                    if (inProgressPaths.ContainsKey(args.Id))
                    {
                        inProgressPaths.Remove(args.Id);
                        UpdateBitmap();
                    }
                    break;
            }
        }

        SKPoint ConvertToPixel(Point pt)
        {
            return new SKPoint((float)(canvasView.CanvasSize.Width * pt.X /
canvasView.Width),
```

```
                                    (float)(canvasView.CanvasSize.Height * pt.Y /
canvasView.Height));
        }

        void UpdateBitmap()
        {
            using (SKCanvas saveBitmapCanvas = new SKCanvas(saveBitmap))
            {
                saveBitmapCanvas.Clear();

                foreach (SKPath path in completedPaths)
                {
                    saveBitmapCanvas.DrawPath(path, paint);
                }

                foreach (SKPath path in inProgressPaths.Values)
                {
                    saveBitmapCanvas.DrawPath(path, paint);
                }
            }

            canvasView.InvalidateSurface();
        }

        void HandleAccessPointTransit()
        {
            ConfirmNotification();
            var sensOut = new SensorOutcomeDTO();

            // get the module pin that matches the access point id of the
current acc point
            GlobalData.modulePinViewModel.GetModulePins().Select(x =>
x.AccessPointId == accessPointHandler.AccessPointId);
            sensOut.PinAssignedId =
GlobalData.modulePinViewModel.GetModulePins()
                                            .Where(x => (x.AccessPointId ==
accessPointHandler.AccessPointId))
                                            .Single()
                                            .PinAssignedId;

            /* First of all, if we want to exit(enter) a location, we need to
enter(exit) the hallway. */
            ManageVirtualHallway((int)accessPointHandler.AccessTypeId,
sensOut);

            /* Now we can exit or enter the location. */
            GlobalData.sensOutcomeViewModel.PostSensorOutcome(sensOut);
        }

        void ManageVirtualHallway(int passingType, SensorOutcomeDTO sensOut)
        {
            var hallwayAction = passingType == GlobalData.passingTypeEntrance
? GlobalData.passingTypeExit : GlobalData.passingTypeEntrance;

            ManageHallwayDTO manHallway = new ManageHallwayDTO
            {
                PinAssignedId = sensOut.PinAssignedId,
                HallwayAction = hallwayAction
            };


GlobalData.sensOutcomeViewModel.ManHallBeforePostSensorOutcome(manHallway);
        }
```

```
        void ConfirmNotification()
        {
            double SecondsDuration = 1;
            string Message = "";
            Color textColor = Color.White;

            if (accessPointHandler.AccessTypeId ==
GlobalData.passingTypeExit)
            {
                Message = indentOption + "EXIT CONFIRMED!";
                textColor = Color.Red;
            }

            if (accessPointHandler.AccessTypeId ==
GlobalData.passingTypeEntrance)
            {
                Message = indentOption + "ENTRANCE CONFIRMED!";
                textColor = Color.Blue;
            }

            ToastConfig.DefaultBackgroundColor =
System.Drawing.Color.AliceBlue;
            ToastConfig.DefaultMessageTextColor = System.Drawing.Color.Red;
            ToastConfig.DefaultActionTextColor =
System.Drawing.Color.DarkRed;
            UserDialogs.Instance.Toast(new ToastConfig(Message)
                        .SetBackgroundColor(Color.Black)
                        .SetMessageTextColor(textColor)
                        .SetDuration(TimeSpan.FromSeconds(SecondsDuration))
                        .SetPosition(ToastPosition.Bottom));
        }
    }
}
```