**PROIECTARE**

1. Arhitectura programului

Programul este compus din un sistem de gestiune a angajaților dintr-o firmă care va permite evidența prezenței angajaților, monitorizarea accesului în parcarea firmei și recunoașterea angajatului prin cod QR la intrarea în firmă. Interacțiunea dintre module se va realiza astfel:

* La intrarea în parcare, o cameră conectată la Raspberry Pi 4 Model B va citi numărul de înmatriculare al automobilului și va trimite informația către un server/API.
* Serverul/API va ține evidența numărului de înmatriculare și a datei și orei de intrare în parcare.
* În funcție de informațiile primite de la server/API, un modul Arduino va semnaliza accesul autorizat cu ajutorul unui led verde timp de 7 secunde, iar accesul neautorizat cu un led roșu timp de 7 secunde. În cazul în care sistemul este funcțional, un led albastru va fi aprins.
* La ieșirea din parcare, utilizatorul va apăsa un buton care va deschide bariera și va semnaliza ieșirea cu un led verde timp de 7 secunde. Locurile de parcare disponibile vor fi actualizate în funcție de informațiile primite de la server/API.
* La intrarea în firmă, utilizatorul va genera un cod QR pe telefonul acestuia și îl va scana cu o cameră aflată la intrare. Informațiile din codul QR vor fi trimise către server/API pentru autentificare.
* Dacă angajatul este recunoscut în sistem, un modul Arduino va semnaliza accesul autorizat cu ajutorul unui led verde timp de 7 secunde, iar dacă nu este recunoscut un led roșu va fi aprins timp de 5 secunde. În cazul în care sistemul este funcțional, un led albastru va fi aprins.
* După ce angajatul primește acces la intrarea în firmă, serverul/API va monitoriza cât timp acesta a fost prezent la locul de muncă.
* La ieșirea din firmă, angajatul va apăsa un buton pentru a porni o altă cameră destinată scanării codului QR de identificare a persoanei. Monitorizarea timpului petrecut la locul de muncă se va opri.
* Administratorul va putea să administreze lista angajaților din firmă, precum și să monitorizeze locul unde se găsesc aceștia în birou. Prin intermediul unei interfețe grafice, administratorul va putea să-i așeze la un birou cu alți colegi și să observe în mod grafic care sunt colegii de birou ai acestuia.
* Sistemul va permite administratorului să genereze rapoarte grafice cu diferiți parametri măsurați pentru unul sau mai mulți angajați, care vor fi accesibile prin interfața web a serverului/API.

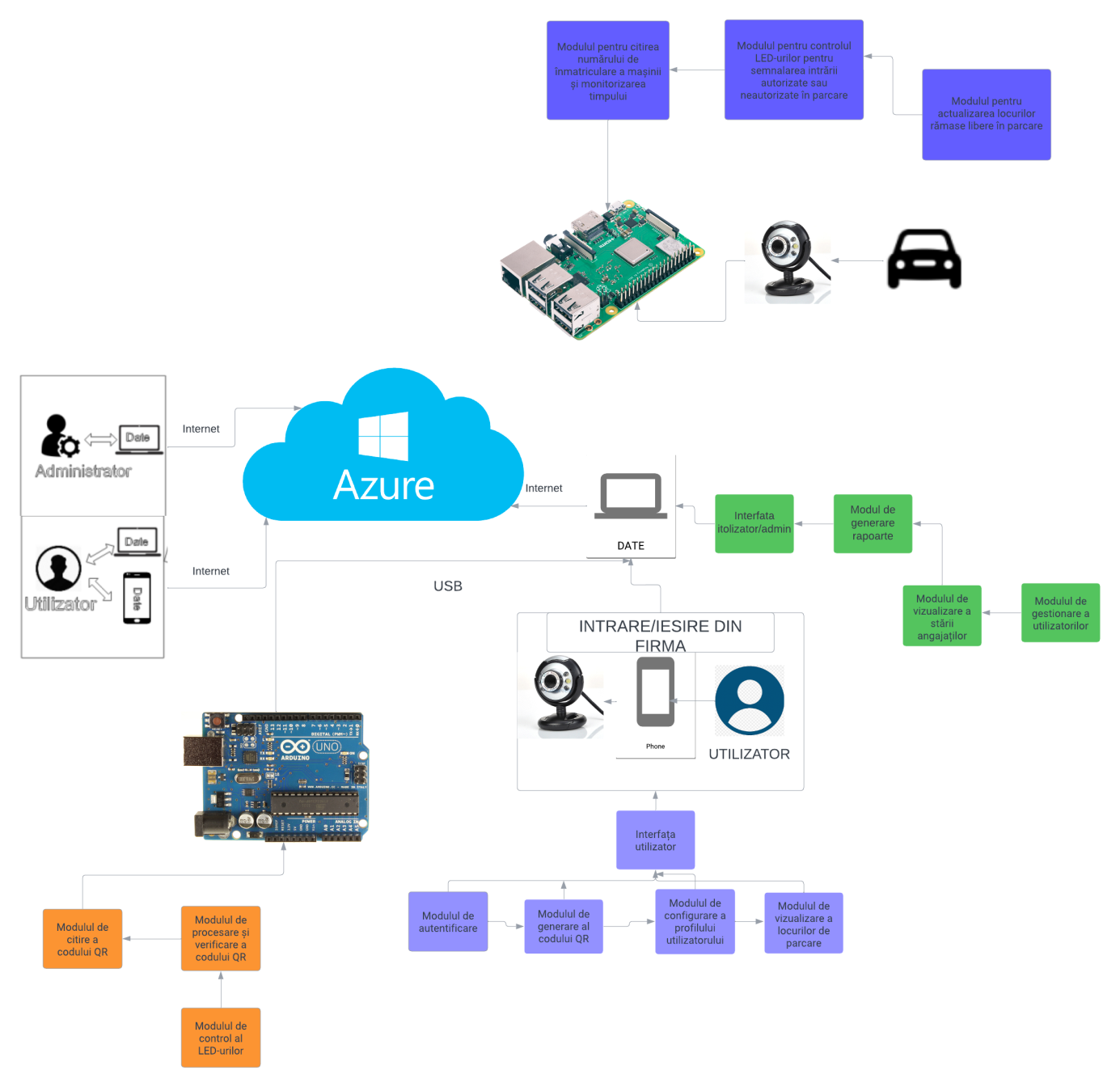


Fig. 1

In figura 1 este prezentata arhitectura programului nostru bazata pe o diagrama cu module si submodule care explica părțile de care avem nevoie pentru implementare.

1. Diagrama de secvență a programului

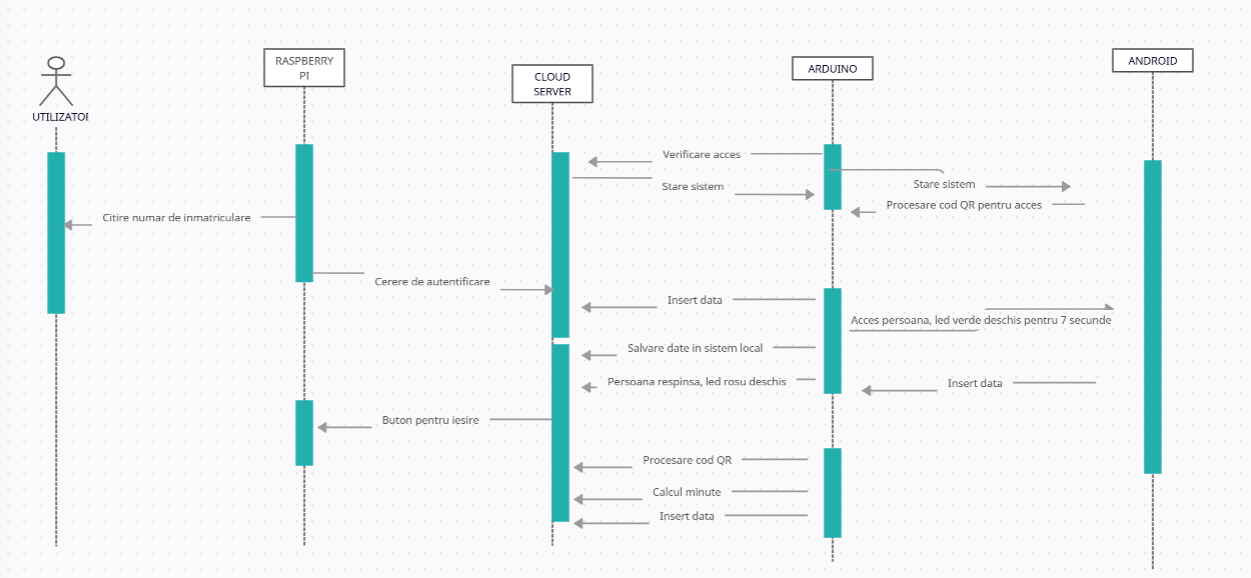


Fig. 2

Diagrama de secvență de mai sus(fig. 2) este o diagramă de modelare a comportamentului dinamic al sistemului. Aceasta descrie interacțiunea între obiectele din sistem și ordinea mesajelor schimbate între acestea.

În cazul acestei diagrame de secvența pentru procesul descris anterior, se poate observa următoarele:

* Se începe cu apăsarea butonului de către persoana care dorește să intre în firmă.
* Apoi, se scanează codul QR generat de pe telefonul acesteia cu ajutorul modulului de scanare QR.
* Programul verifică dacă persoana este în baza de date și, dacă da, se deschide ledul verde timp de 7 secunde și se înregistrează intrarea în firmă în baza de date prin internet.
* Dacă persoana nu este în baza de date, se deschide ledul roșu și nu se permite accesul în firmă.
* După ce persoana petrece o perioadă de timp în firmă, aceasta apasă butonul și scanează un alt cod QR pentru a înregistra ieșirea.
* Programul calculează durata petrecută de persoana în firmă și o înregistrează în baza de date pentru pontaj.
* În cazul în care nu există conexiune la internet, datele sunt stocate local timp de 24 de ore.
* După înregistrarea ieșirii, se afișează starea sistemului prin deschiderea ledului albastru.

Această diagramă de secvențe ilustrează interacțiunea dintre obiectele din sistem și ordinea mesajelor schimbate între acestea, astfel încât să se realizeze procesul de înregistrare a intrărilor și ieșirilor în firmă.

1. Descrierea componentelor (modulelor)

Descrierea componentelor unui modul este necesară pentru a asigura o înțelegere clară și completă a funcționalităților, responsabilităților și interacțiunilor dintre diferitele părți ale sistemului. Acest lucru este important din mai multe motive:

1. Planificarea și proiectarea: Descrierea componentelor ajută la planificarea și proiectarea sistemului, permițând dezvoltatorilor să identifice și să analizeze cerințele și să determine cum ar trebui să interacționeze diferitele module și componente.
2. Divizarea muncii: O descriere detaliată a componentelor permite echipei de dezvoltare să împartă munca în mod eficient între membrii săi, astfel încât fiecare să se concentreze pe o anumită componentă sau funcționalitate.
3. Comunicarea și colaborarea: Descrierea componentelor facilitează comunicarea și colaborarea între membrii echipei, precum și cu alte părți interesate, cum ar fi clienții sau stakeholderii. Acest lucru asigură că toată lumea are o înțelegere comună a scopului și funcționalității sistemului.
4. Testarea și depistarea erorilor: Descrierea componentelor facilitează procesul de testare și depistare a erorilor, deoarece dezvoltatorii pot identifica în mod clar responsabilitățile și interacțiunile dintre diferitele părți ale sistemului.
5. Documentația și întreținerea: Descrierea componentelor ajută la crearea unei documentații solide și complete pentru sistem, ceea ce este esențial pentru întreținerea și dezvoltarea ulterioară a proiectului. Acest lucru permite altor dezvoltatori să înțeleagă rapid modul în care funcționează sistemul și să contribuie la îmbunătățirea sau extinderea acestuia.
6. Reutilizarea și adaptarea: O descriere clară a componentelor permite reutilizarea sau adaptarea lor în alte proiecte sau contexte, economisind timp și resurse.

Prin descrierea componentelor modulelor, putem asigura o proiectare eficientă și eficace, o comunicare clară între membrii echipei și o documentație solidă care facilitează dezvoltarea și întreținerea sistemului.

**Descrierea componentei modulului RaspberryPi:**

Pentru modulul RaspberryPi am identificat următoarele componente și s-a realizat descrierea acestora împreună cu implementarea unor scheme logice pentru o mai bună înțelegere:

1. *Componenta de autentificare:*

Intrări:

* + Numărul de înmatriculare al mașinii

Prelucrări:

* + Verificarea validității numărului de înmatriculare.
  + Trimiterea datelor către server/API pentru accesul în parcare.

Ieșiri:

* + Starea autentificării (autentificare reușită/eșuată).

Interacțiuni:

* + Comunică cu baza de date din Cloud pentru a obține accesul în parcare

O imagine care conține diagramă

Descriere generată automat

Fig. 3.1

În figura de mai sus se prezintă arhitectura componentei de autentificare a numărului de înmatriculare.

1. *Componenta de prelucrare a numărului de înmatriculare:*

Intrări:

* + Numărul de înmatriculare al mașinii.

Prelucrări:

* + Trimiterea unei cereri către baza de date pentru a obține informațiile despre parcare.

Ieșiri:

* + Informații despre accesul în parcare

Interacțiuni:

* + Comunică cu baza de date din Cloud pentru a obține accesul în parcare.

1. *Componenta de vizualizare a locurilor disponibile în parcare:*

Intrări:

* + Numărul de înmatriculare al mașinii.

Prelucrări:

* + Trimiterea unei cereri către baza de date din Cloud pentru a obține numărul de locuri disponibile în parcarea firmei.

Ieșiri:

* + Numărul de locuri disponibile în parcarea firmei.
  + Existența numărului de înmatriculare a mașinii în baza de date

Interacțiuni:

* + Comunică cu baza de date pentru a obține informațiile despre locurile disponibile în parcare.

O imagine care conține diagramă

Descriere generată automat

Fig. 3.2

În figura de mai sus este o schemă vizuală simplificată pentru componenta de vizualizare a locurilor disponibile în parcare și a existenței numărului de înmatriculare a mașinii.

**Descrierea componentei modulului Android:**

Pentru modulul Android am identificat următoarele componente si s-a realizat descrierea acestora împreuna cu implementarea unor scheme logice pentru o mai buna înțelegere:

1. *Componenta de autentificare:*

Intrări:

* + Email/username și parolă introduse de angajat.

Prelucrări:

* + Verificarea validității email-ului și parolei.
  + Trimiterea datelor către server/API pentru autentificare.

Ieșiri:

* + Starea autentificării (autentificare reușită/eșuată).
  + Token de acces pentru utilizare ulterioară în alte cereri către server/API.

Interacțiuni:

* + Comunică cu serverul/API pentru a obține starea autentificării și tokenul de acces.

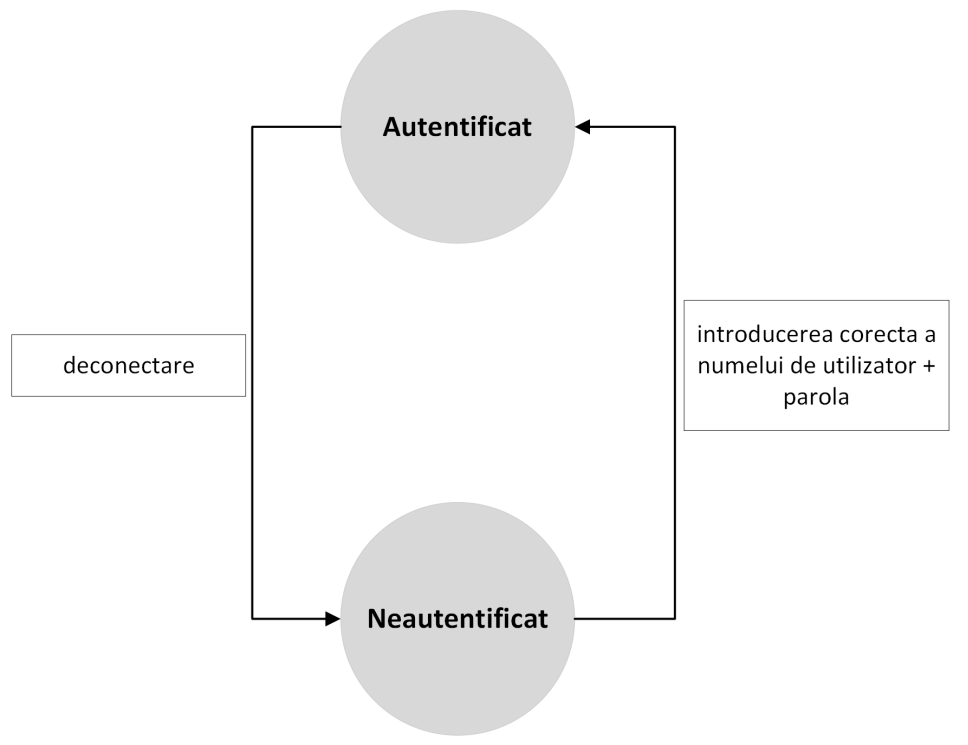


Fig. 3.3

În figura 3.3 se prezintă arhitectura componentei de autentificare a aplicației.

2. *Componenta de prelucrare a profilului angajatului:*

Intrări:

* + Token de acces al angajatului autentificat.

Prelucrări:

* + Trimiterea unei cereri către server/API pentru a obține informațiile despre profilul angajatului.

Ieșiri:

* + Informații despre profilul angajatului (nume, prenume, CNP, poza de profil).

Interacțiuni:

* + Comunică cu serverul/API pentru a obține informațiile despre profilul angajatului.

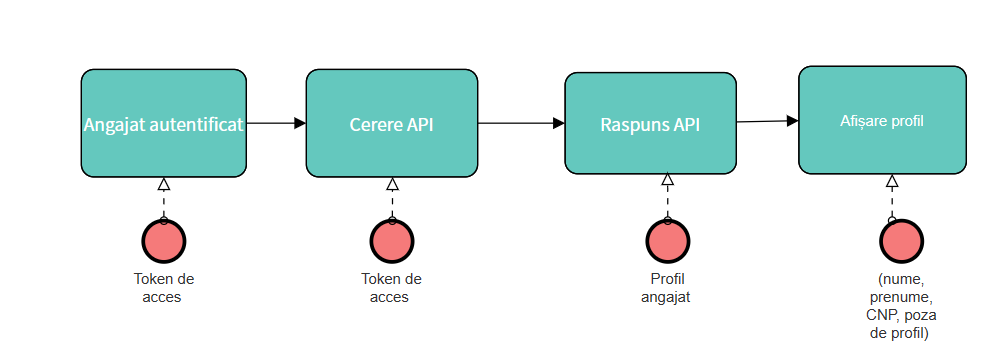


Fig. 4

In figura 4 este prezentata componenta de prelucrare a profilului angajatului - partea aplicației Android care gestionează informațiile despre profilul angajatului autentificat. Aceasta se ocupă de obținerea și afișarea acestor informații în aplicație.

3.*Componenta de generare a codului QR:*

Intrări:

* + CNP-ul angajatului

Prelucrări:

* + Aplicarea algoritmului de codificare al CNP-ului pentru a crea un șir de caractere unic.
  + Generarea codului QR pe baza șirului de caractere unic.

Ieșiri:

* + Codul QR generat, afișat pe ecranul telefonului.

Interacțiuni:

* + Folosește informațiile despre profilul angajatului pentru a obține CNP-ul.

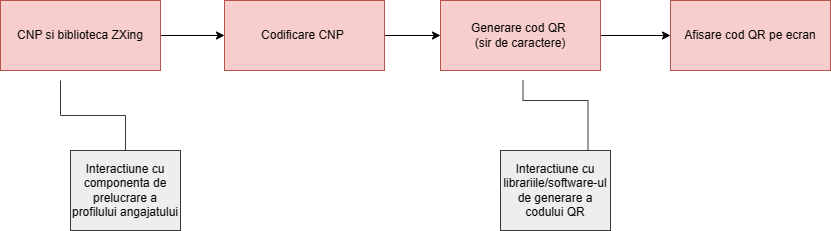


Fig. 5

Schema de mai sus(fig. 5) reprezintă o reprezentare vizuală a componentei de generare a codului QR din aplicația Android. Schema arată procesul de intrare, prelucrare, ieșire și interacțiune cu alte componente sau librării. Acesta oferă o înțelegere mai bună a modului în care componenta de generare a codului QR funcționează și se integrează cu alte părți ale aplicației.

4.*Componenta de vizualizare a locurilor disponibile în parcare:*

Intrări:

* + Token de acces al angajatului autentificat.

Prelucrări:

* + Trimiterea unei cereri către server/API pentru a obține numărul de locuri disponibile în parcarea firmei.

Ieșiri:

* + Numărul de locuri disponibile în parcarea firmei.

Interacțiuni:

* + Comunică cu serverul/API pentru a obține informațiile despre locurile disponibile în parcare.

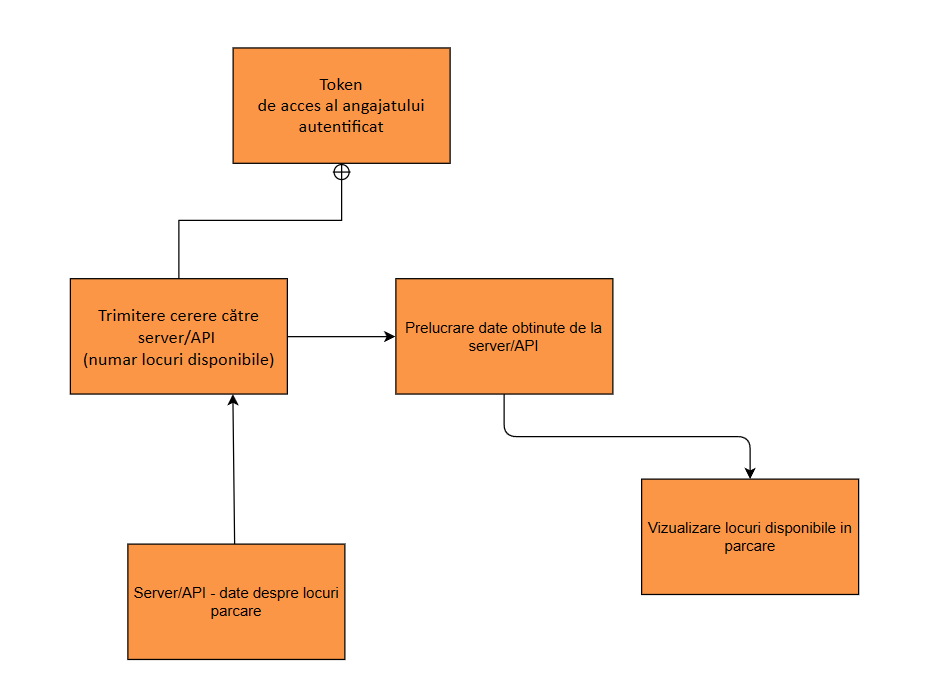


Fig. 6

In figura 6 este o schemă vizuală simplificată pentru componenta de vizualizare a locurilor disponibile în parcare.

Daca vorbim de interacțiunea componentei aplicației de generare a codului QR cu alte module ale procesului, identificam următoarele:

**Comunicarea complexă între aplicația Android și componenta cloud**

se realizează prin intermediul unor cereri HTTP/HTTPS și un set de API-uri, utilizând protocoale de autentificare și de transfer de date securizate. Acest proces implică următoarele etape:

* Autentificarea utilizatorului:
  + Aplicația Android trimite o cerere HTTP/HTTPS de autentificare către serverul cloud, incluzând datele de autentificare ale utilizatorului (email și parolă).
  + Serverul cloud verifică validitatea datelor de autentificare și răspunde cu un token de acces care va fi utilizat pentru a autoriza cererile ulterioare.
* Încărcarea datelor despre profilul angajatului:
  + Aplicația Android trimite o cerere HTTP/HTTPS către serverul cloud, incluzând tokenul de acces, pentru a solicita informații despre profilul angajatului.
  + Serverul cloud verifică tokenul de acces și răspunde cu datele profilului angajatului (nume, prenume, CNP, poza de profil, etc.).
* Vizualizarea locurilor de parcare disponibile:
  + Aplicația Android trimite o cerere HTTP/HTTPS către serverul cloud, incluzând tokenul de acces, pentru a solicita numărul de locuri de parcare disponibile în parcarea firmei.
  + Serverul cloud verifică tokenul de acces și răspunde cu informațiile despre locurile de parcare disponibile.

Pe parcursul acestei comunicări, aplicația Android și serverul cloud vor folosi protocoale criptate, cum ar fi HTTPS și mecanisme de autentificare securizate (OAuth2, JWT, etc.) pentru a asigura confidențialitatea și integritatea datelor transmise între cele două componente.

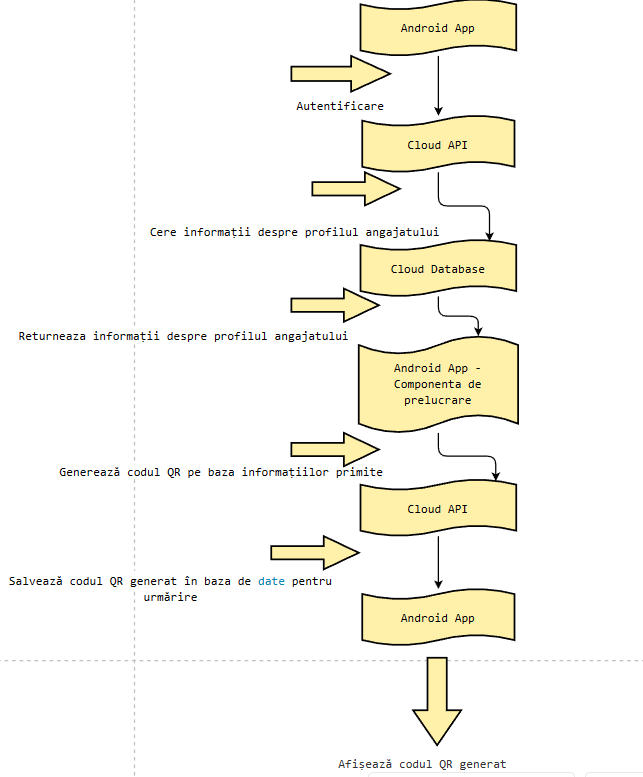


Fig. 7

Diagrama de secvență(fig. 7) prezintă interacțiunile între aplicația Android și modulul cloud în procesul de generare a codului QR. Mai întâi, utilizatorul se autentifică în aplicație și solicită generarea codului QR. Aplicația trimite o cerere către API-ul cloud-ului pentru a obține informațiile despre profilul angajatului, iar cloud-ul răspunde cu informațiile respective. Componenta de prelucrare din aplicația Android folosește informațiile primite pentru a genera codul QR, iar apoi aplicația trimite codul QR înapoi la cloud pentru a fi salvat în baza de date. La final, aplicația afișează codul QR generat pentru utilizator.

**Interacțiunea dintre aplicația Android și componenta Arduino**

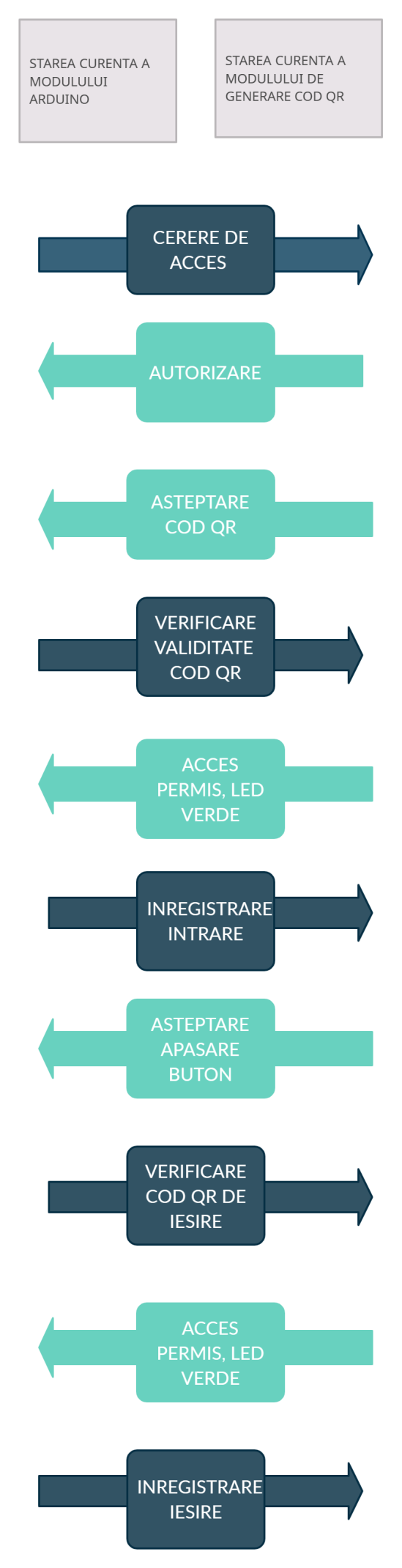
Se realizează prin intermediul serverului/API care conectează cele două componente.

Atunci când un angajat dorește să intre în firmă, acesta solicită generarea unui cod QR prin intermediul aplicației Android. Componenta de generare a codului QR primește CNP-ul angajatului și data și ora curente, aplică algoritmul de codificare al CNP-ului și generează un cod QR valid pentru o perioadă de 5 minute.

Angajatul scanează codul QR cu ajutorul modulului Arduino, care procesează informațiile și le transmite prin intermediul serverului/API către baza de date a firmei. Datele sunt stocate în baza de date și se realizează înregistrarea orei de intrare a angajatului în firmă.

La plecarea angajatului din firmă, acesta apasă un buton și scanează un alt cod QR generat pe telefonul său prin intermediul aplicației Android. Modulul Arduino procesează informațiile și le transmite prin intermediul serverului/API către baza de date a firmei, înregistrând ora de ieșire a angajatului și calculând timpul petrecut în firmă.

În cazul în care nu există conexiune la internet, datele sunt stocate local pe modulul Arduino timp de 24 de ore, după care se încearcă trimiterea lor către server/API când conexiunea este restabilită.

 Fig. 8

**3.Descrierea comunicării între module**

Modulul aplicației Android:

Pentru a dezvolta aplicația Android pentru angajați, vom folosi Android Studio și limbajul de programare Java. Aplicația va avea următoarele componente și canale de comunicație:

1. Interfața de autentificare: Angajații se vor autentifica în aplicație folosind contul lor din firmă sau email-ul personal. Canalele de comunicație vor include transmiterea datelor de autentificare către serverul de autentificare și primirea unui token de acces în cazul unei autentificări reușite.

2. Profilul angajatului: Acesta va permite vizualizarea și editarea informațiilor personale ale angajatului, inclusiv stabilirea unui avatar. Comunicarea cu serverul va include transmiterea și primirea datelor profilului, precum și încărcarea și descărcarea imaginilor.

3. Generarea codului QR: La solicitarea angajatului, aplicația va genera un cod QR pe baza CNP-ului acestuia. Acest cod QR va fi valid timp de 5 minute. Generarea codului QR va fi realizată local, fără a necesita comunicare cu serverul.

4. Vizualizarea locurilor disponibile în parcare: Aplicația va comunica cu serverul pentru a obține numărul de locuri disponibile în parcarea firmei. Acesta va fi actualizat în timp real, pentru a se asigura că informațiile sunt corecte.

Toate aceste canale de comunicație vor fi securizate, utilizând criptare și autentificare pentru a proteja confidențialitatea și integritatea datelor.

Comunicarea între aplicația Android și server va fi realizată prin intermediul unui API RESTful. Pentru fiecare canal de comunicație, vom avea următorii parametri și restricții:

1. **Autentificare:**

* Modul de comunicare: HTTP(S) request (POST) către server/API
* Parametrii: email, parola
* Semnificații: email-ul și parola angajatului pentru autentificare
* Restricții: email valid, parola cu minim 8 caractere, cu cel puțin o literă mare, o literă mică și un număr

1. **Profilul angajatului:**

* Modul de comunicare: HTTP(S) request (GET, PUT) către server/API
* Parametrii: token de acces, id angajat, nume, prenume, CNP, poza de profil (avatar)
* Semnificații: informații despre profilul angajatului, inclusiv CNP pentru generarea codului QR
* Restricții: nume și prenume nu pot fi goale, CNP valid, dimensiunea pozei de profil limitată (ex: 1 MB)

1. **Generarea codului QR:**

* Modul de comunicare: intern în aplicație, fără conexiune la server/API
* Parametrii: CNP, timestamp curent
* Semnificații: CNP-ul angajatului și momentul generării codului QR
* Restricții: codul QR expiră în 5 minute de la generare

1. **Vizualizarea locurilor disponibile în parcare:**

* Modul de comunicare: HTTP(S) request (GET) către server/API
* Parametri: token de acces
* Semnificații: numărul de locuri disponibile în parcarea firmei
* Restricții: necesită conexiune la internet și autentificare validă

1. **Autentificare API**

* Modul de comunicare: HTTP(S) request (POST) către server
* Parametrii: email, parola
* Semnificații: email-ul și parola angajatului pentru autentificare
* Restricții: autentificare eșuată în cazul unor date incorecte

1. **Profil angajat API:**

* Modul de comunicare: HTTP(S) request (GET, PUT) către server
* Parametrii: token de acces, id angajat, nume, prenume, CNP, poza de profil (avatar)
* Semnificații: informații despre profilul angajatului și actualizări ale acestuia
* Restricții: acces restricționat la profilul angajatului conectat

1. **Locuri parcare API:**

* Modul de comunicare: HTTP(S) request (GET) către server
* Parametrii: token de acces
* Semnificații: numărul de locuri disponibile în parcarea firmei
* Restricții: necesită autentificare validă pentru a accesa informațiile

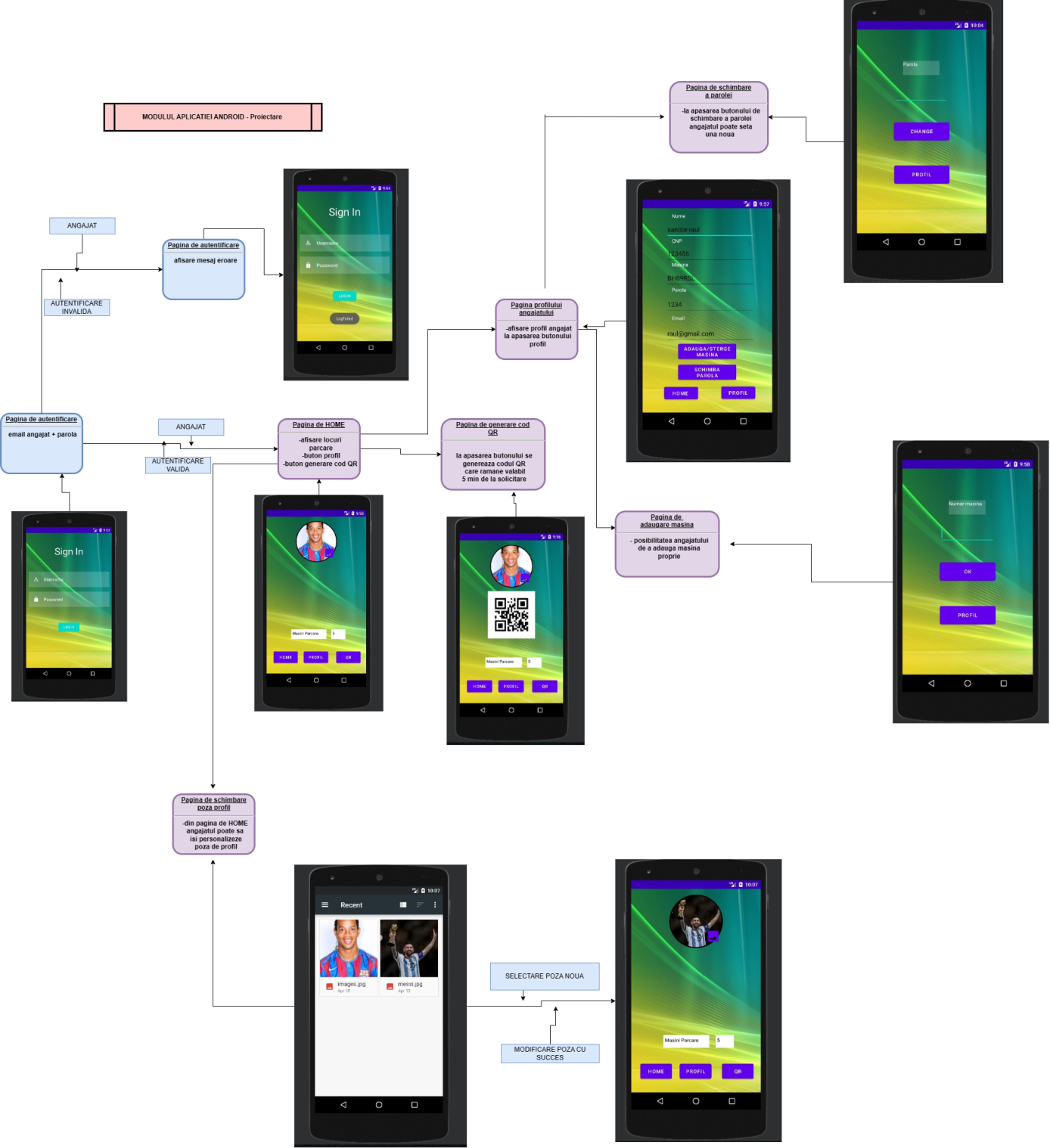


Fig. 9

Figura 9 reprezintă un model de proiectare pentru modulul aplicației Android de generare a codului QR.

În plus, aplicația va trebui să gestioneze conexiunea la internet și să afișeze mesaje de eroare corespunzătoare în cazul în care conexiunea nu este disponibilă.

Pentru a asigura o experiență fluidă și eficientă a utilizatorului, aplicația va utiliza tehnici de optimizare precum gestionarea resurselor, reducerea utilizării memoriei și asigurarea unei bune performanțe în diferite condiții de rețea. De asemenea, se va acorda o atenție deosebită aspectelor vizuale și de interacțiune, pentru a crea o interfață intuitivă și ușor de utilizat.

În concluzie, aplicația Android pentru angajați va permite generarea codurilor QR, vizualizarea și editarea profilului, precum și afișarea numărului de locuri disponibile în parcarea firmei. Comunicarea între aplicație și server se va realiza prin intermediul unui API RESTful, utilizând canale de comunicație securizate și criptate. Implementarea tehnicilor de optimizare va asigura o performanță bună și o experiență plăcută pentru utilizator.

Schema logica privind componentele si modul de comunicare din acestea pentru Aplicația Android:

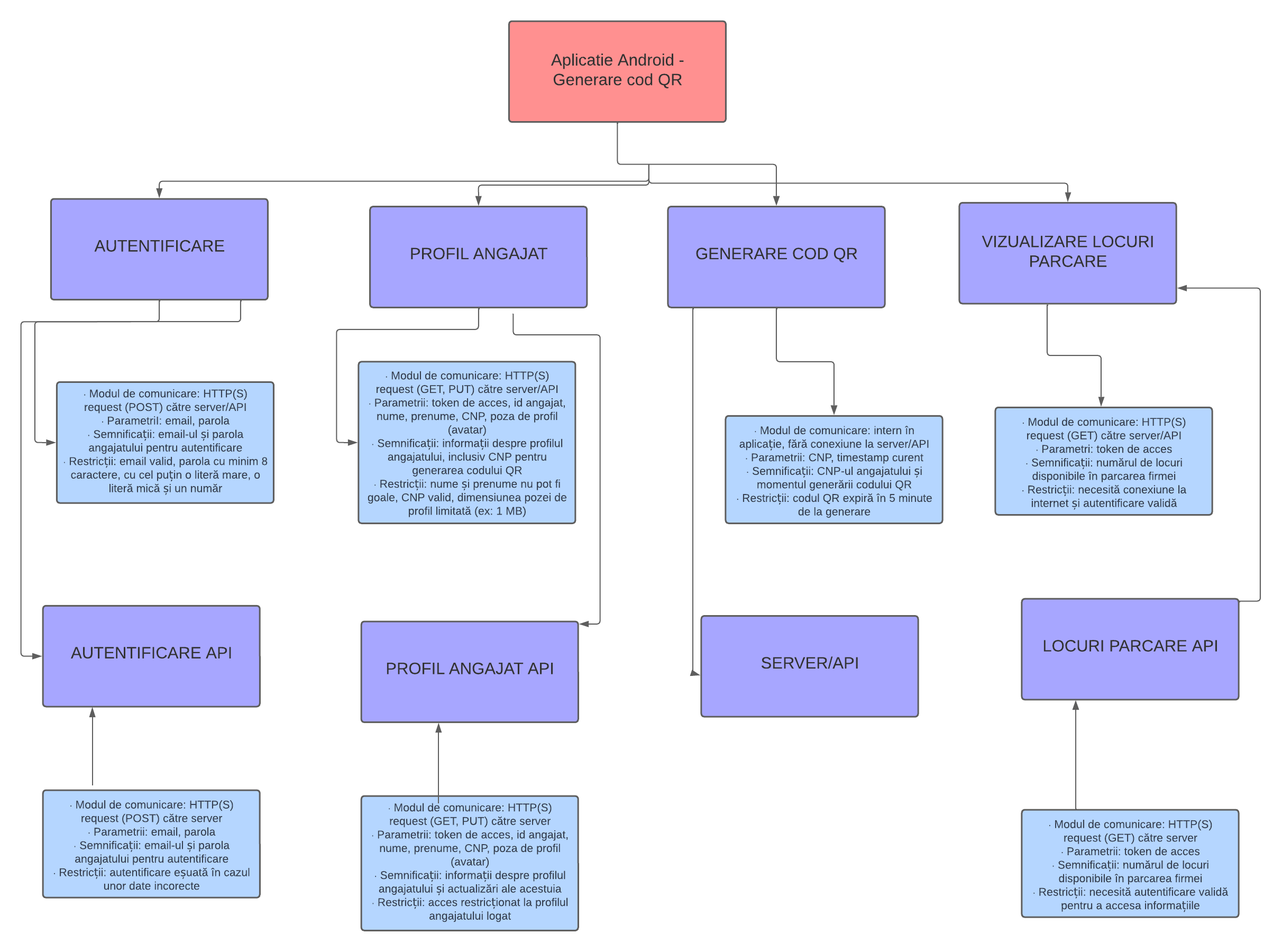
****

Fig. 10

Această schemă(fig. 10) prezintă o structură mai complexă și detaliată, care include sub-componentele aplicației Android și interacțiunile cu serverul API. În plus, sunt reprezentate și conexiunile dintre interfața grafică și diferitele funcționalități ale aplicației. Acest tip de schemă oferă o imagine de ansamblu a interacțiunilor dintre componentele aplicației și poate fi utilizată pentru a ghida procesul de dezvoltare și implementare.

1. Diagrama de clase

Putem să ne gândim la următoarele clase pentru a modela starea aplicației:

1. Clasa **Angajat**:
   * Atribute: CNP, nume, email, poziție, avatar (URL sau referință la resursă), parola(vizualizare sau schimbare), mașina (adăugare de mașina)
   * Metode: constructor, getteri și setteri pentru fiecare atribut
2. Clasa **CodQR**:
   * Atribute: CNP, data de intrare, ora de intrare, , codQR (valoarea codului QR), expirat (boolean)
   * Metode: constructor, getteri și setteri pentru fiecare atribut, o metodă pentru a verifica dacă codul QR a expirat
3. Clasa **Parcare**:
   * Atribute: locuriLibere (numărul de locuri disponibile),contrar ne afișează 0 daca nu exista niciun loc liber in parcare
   * Metode: constructor, getteri și setteri pentru fiecare atribut, metode pentru a actualiza numărul locuri libere
4. Clasa **Autentificare**:

* + Atribute: angajat (o instanță a clasei Angajat), autentificat (boolean)
  + Metode: constructor, getteri și setteri pentru fiecare atribut, o metodă pentru autentificare (care verifică email-ul și parola angajatului și setează atributul **autentificat**)
  + Prin metodele enumerate mai sus, se va face accesul in baza de date din cloud si se va verifica daca angajatul exista, iar pe baza verificării se va putea intra in aplicație, contrar se va genera un mesaj de eroare la conectare

1. Clasa **Aplicație**:
   * Atribute: angajat (o instanță a clasei Angajat), codQR (o instanță a clasei CodQR), parcare (o instanță a clasei Parcare), autentificare (o instanță a clasei Autentificare)
   * Metode: constructor, getteri și setteri pentru fiecare atribut, metode pentru a interacționa cu clasele Autentificare, CodQR și Parcare (de exemplu, autentificarea angajatului, generarea codului QR, verificarea disponibilității locurilor de parcare)
   * Pentru a implementa aceste clase în Java, vom crea fișierele corespunzătoare pentru fiecare clasă, cu atributele și metodele specificate. Apoi, vom utiliza aceste clase în activitățile și fragmentele aplicației noastre pentru a gestiona starea aplicației în timpul rulării.

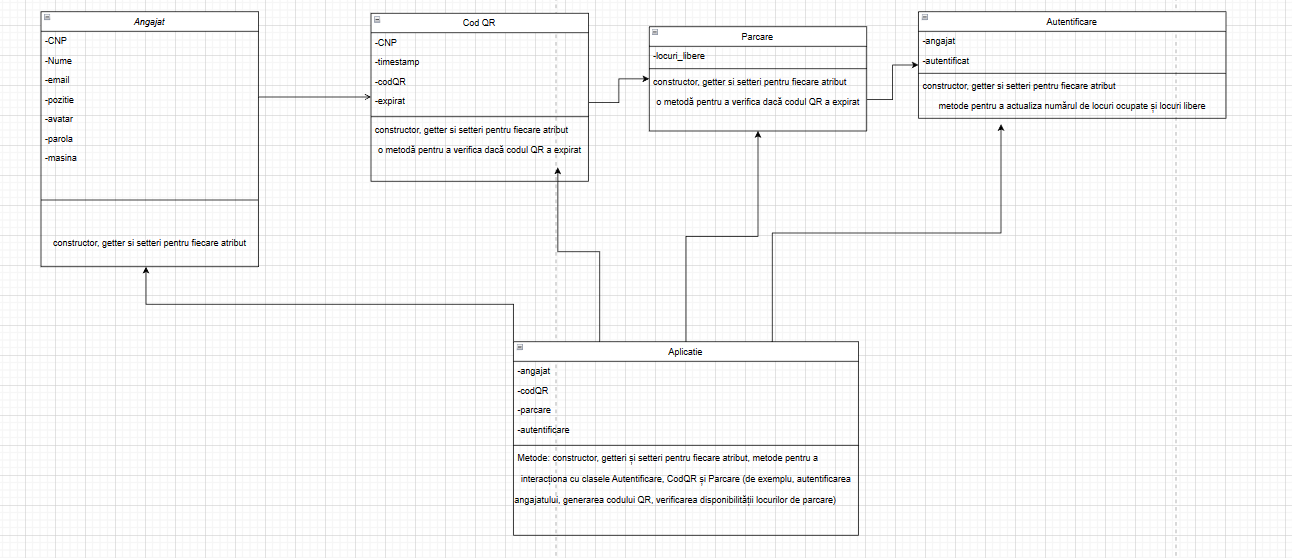


Fig. 11

Acest model de stare ne va ajuta să gestionam informațiile despre angajați, codurile QR, parcarea și autentificarea într-un mod organizat și eficient, facilitând dezvoltarea aplicației și menținerea codului.

Unul din algoritmii reprezentativi ai modulului aplicației Android de generare cod QR este algoritmul de codificare al CNP-ului angajatului. CNP-ul este o informație sensibilă și ar trebui să fie manipulată cu precauție, astfel încât să nu fie expusă în mod inadecvat. Astfel, algoritmul de codificare al CNP-ului poate fi realizat în mai mulți pași pentru a asigura un nivel adecvat de securitate și unicitate. Acesta poate fi folosit în combinație cu alte date si librarii, spre exemplu: librăria ZXing pentru a genera codul QR.

Indicații pentru realizarea unui astfel de algoritm:

* + - Obținerea CNP-ul angajatului din baza de date
    - Folosirea librăriei ZXing pentru a asigura unicitatea șirului de caractere. Se poate utiliza o funcție de hashing pentru a combina CNP-ul și utilitatea librăriei mai sus menționată, cum ar fi SHA-256 sau alți algoritmi de hashing disponibili.
    - Hash-ul rezultat poate fi folosit ca bază pentru generarea codului QR.

ZXing (pronunțat "Zebra Crossing") este o bibliotecă open source pentru decodarea și generarea de coduri de bare, inclusiv coduri QR.

Algoritmul de generare a codului QR în interiorul bibliotecii ZXing folosește o serie de pași pentru a transforma datele introduse într-un cod QR.

Acești pași includ:

* + - Encodarea datelor - datele de intrare (de exemplu, un URL sau un text) sunt convertite într-un format binar, folosind o schemă specifică (de exemplu, UTF-8).
    - Adăugarea de date de corecție a erorilor - pentru a asigura că codul QR poate fi decodat corect chiar și în cazul unor erori minore, se adaugă date suplimentare de corecție a erorilor.
    - Generarea matricei codului QR - datele encodate sunt reprezentate sub formă de matrice de celule negre și albe (sau colorate) pentru a forma codul QR propriu-zis.
    - Adăugarea de informații de formatare - matricea este completată cu anumite informații de formatare, cum ar fi margini sau tipuri de cod de bare.
    - Aplicarea masării - pentru a asigura o scanare mai ușoară a codului QR, se aplică un algoritm de mascare care modifică anumite celule ale matricei în funcție de o cheie specifică.
    - Adăugarea de patru colțuri de detectare - pentru a ajuta la detectarea codului QR și la corectarea distorsiunilor, se adaugă patru colțuri de detectare în matrice.
    - Adăugarea de coduri de corecție a formatului - pentru a asigura că codul QR poate fi citit corect chiar și în cazul unor erori de format, se adaugă coduri de corecție a formatului.
    - În final, matricea codului QR este afișată ca o imagine, care poate fi salvată sau afișată direct pe ecran.

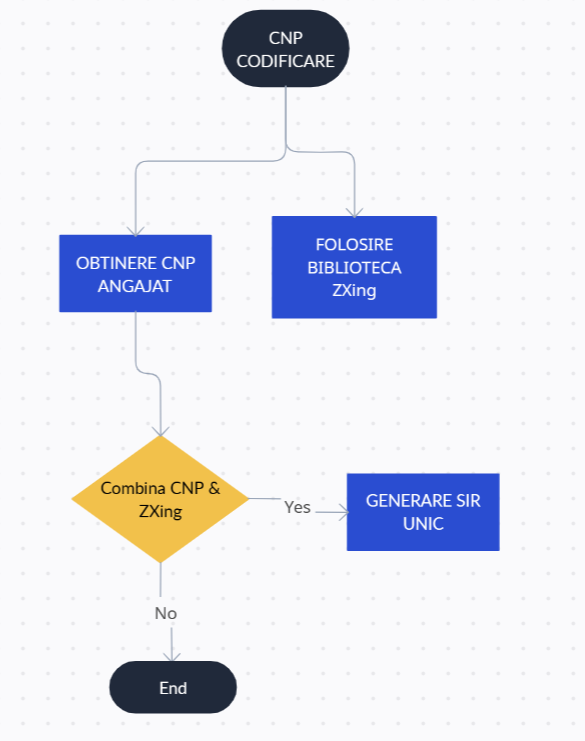


Fig. 12

Această schemă logică(fig. 12) prezintă pașii necesari pentru a realiza codificarea CNP-ului, combinând CNP-ul și facilitățile oferite de biblioteca ZXing pentru a genera un șir de caractere unic. Acest șir de caractere unic poate fi folosit ulterior pentru a crea codul QR în aplicația Android. Schema poate fi folosită ca bază pentru proiectarea și dezvoltarea algoritmului în limbajul de programare ales.

1. Scheme cu interacțiune in interfețe

*Interacțiunea intre interfețe la modulul WEB:*

Pagina cu Raportul prezentei in firma pentru toți angajații

Pagina cu alocare birou pentru angajați

Pagina cu alocare poziție la birou pentru angajati

Pagina cu Raportul prezentei in firma

Pagina Utilizator

Pagina de Login

Pagina Administrator

Fig. 13

In figura 13 este ilustrata schema aplicației noastre WEB cu interfață grafica intuitivă, cu navigare simplă spre toate funcționalitățile aplicației.

Prima pagina este cea de Login, in care utilizatorul se poate conecta cu contul primit de la firma prin intermediul adresei de email si al parolei. Se poate conecta, in funcție de tipul de privilegiul pe care il are, ca administrator sau ca simplu angajat.

A doua pagina este dedicata administratorului care gestionează conturile tuturor angajaților din firma. Poate realiza următoarele acțiuni:

* va putea crea utilizatori noi, pentru fiecare dintre aceștia va înregistra: nume, prenume, CNP si număr de înmatriculare dacă acesta are mașină
* setarea numărului total de locuri de parcare ale firmei
* prin intermediul butonului „Generate Reports For All Employees” se navighează la alta pagina ce conține urmărirea prezenței în firmă a tuturor angajaților într-un raport, prin înregistrarea orelor de intrare/ieșire pentru fiecare angajat în parte si daca este cu mașina sau nu.
* prin intermediul butonului „Office Allocation” se navighează la pagina cu toate birourile din firma distribuite grafic in funcție de etaj si departament.
* poate da click pe pozițiile de la fiecare birou, îl redirecționează in alta pagina unde poate aloca cate un angajat la fiecare poziție.
* poate vizualiza starea fiecărui angajat in mod grafic, culoarea roșie - ocupat si verde - disponibil. De exemplu, la o masa de birou sunt 6 scaune din care 5 sunt ocupate.
* prin intermediul butonului „Log out”, este redirecționat la pagina de Login

In Pagina Utilizator, angajatul:

* își poate vedea datele personale( nume, prenume, adresa de email, CNP, numărul de înmatriculare al mașinii daca are, etaj, departament, birou si poziție la birou
* prin intermediul butonului „Generate Report” se navighează la alta pagina care conține raportul personal pe zile, sub forma unui tabel, cu ora de intrare respectiv ieșire din firma, timpul petrecut si daca este cu mașina sau nu
* prin intermediul butonului „Log out”, este redirecționat la pagina de Login

*Interacțiunea între interfețe la modulul Arduino:*

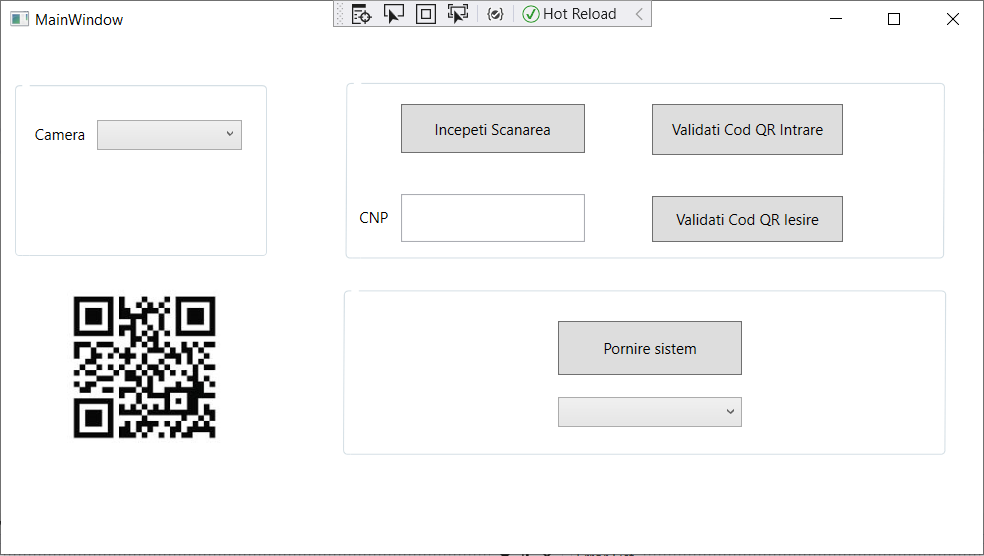


Fig. 14

1. Descrierea modulelor din punct de vedere al funcționalității

*Componenta Cloud*

Pentru a realiza stocarea și accesarea datelor fișei unui angajat, este necesară crearea unei baze de date relaționale care să conțină tabele pentru a stoca informațiile despre angajați și locurile de parcare. Ca limbaj pentru gestionarea bazei de date utilizăm PostgreSQL (Structured Query Language), iar serviciul de găzduire online, pentru integrarea acesteia în cloud, este Microsoft Azure.

Azure este o platformă profesională de servicii în cloud, care îți permite să construiești rapid, să implementezi și să administrezi în siguranță aplicații scalabile. Microsoft Azure își pune resursele la dispoziția prin intermediul internetului. Considerăm că este potrivit proiectului nostru deoarece acesta susține orice sistem de operare și o gamă largă de limbaje de programare, oferind acces la servicii integrate astfel încât utilizatorii să poată gestiona infrastructura mult mai rapid prin executarea unui număr ridicat de sarcini.

PostgreSQL este un sistem de gestiune a bazelor de date relaționale (RDBMS) puternic și open-source, care oferă o gama largă de caracteristici și capacități pentru stocarea, gestionarea și analizarea datelor. PostgreSQL stochează datele în tabele, care pot fi organizate în scheme. Fiecare tabela constă din rânduri și coloane, unde fiecare coloană reprezintă un tip specific de date, iar fiecare rând reprezintă o înregistrare unică. PostgreSQL suportă o gama largă de tipuri de date, inclusiv numerice, text, dată/oră, boolean și altele.

PostgreSQL suportă SQL (Structured Query Language), care este un limbaj standard folosit pentru interogarea și manipularea bazelor de date relaționale. Utilizatorii pot folosi SQL pentru a crea, citi, actualiza și șterge date în PostgreSQL. De asemenea, PostgreSQL suportă funcții SQL avansate, cum ar fi vizualizări, proceduri stocate, declanșatoare și altele.

*Componenta Arduino*

Modulul Arduino Uno va avea conectate la portul de ieșire 3 led-uri de culori diferite: led-ul albastru deschis va indica funcționalitatea sistemului, cel de culoare roșie semnalizează că persoana nu a fost identificată cu succes și nu i se permite accesul în firmă, iar cel de-al treilea led, de culoare verde, indică că angajatul a fost identificat în baza de date și are acces la locul de muncă. Astfel, programul din modulul Arduino indică starea sistemului, astfel că utilizatorul poate vedea dacă sistemul este pornit în cazul în care se poate observa un led albastru aprins. Sistemul poate fi oprit dacă persoana s-a identificat și are drepturi de administrator. În cazul în care o persoană neautorizată încearcă să oprească sistemul, această încercare va fi tratată ca atare de program, neavând niciun efect vizibil. Oprirea sistemului corespunde opririi ledului albastru și incapacității unei persoane de a mai scana codul unic. Pornirea sistemului se poate face selectând un port respectiv pentru configurația hardware. Acest lucru se va face din bara de selecție prezentă în interfața grafică dedesubt de butonul de oprire/pornire a sistemului . În cazul în care nu se alege un port, sistemul va alege un port implicit.

Dacă nu va fi posibilă găsirea unei conexiuni USB cu un port, sistemul nu va putea fi pornit, acest lucru fiind tratat drept o excepție. O soluție pentru a rezolva această neplăcere ar fi aceea de a vă asigura că ați conectat cablul USB în device-ul pe care îl utilizați pentru a comunica cu Modulul Arduino.

La apăsarea butonului „Începeți Scanarea” din cadrul interfeței cu utilizatorul, va fi selectată prima cameră detectată în cazul în care nu a fost selectată deja altă cameră pe care o dorește utilizatorul. Pe ecran se va observa în „real-time” ceea ce captează camera și se va opri când aceasta va capta un cod QR. În urma captării unui cod QR , în caseta de text cu eticheta „CNP” va fi afișat rezultatul decriptării codului QR. În cazul în care textul decriptat nu va fi format din 13 cifre, ci din alte caractere sau un șir de o altă lungime, în caseta text veți putea observa un text sugestiv. În cazul în care utilizatorul a încercat scanarea unui cod QR fără reușită, nefiind focalizat sau identificat codul respectiv, scanarea va continua până când se reușește identificarea unui cod QR valid.

La intrarea unei persoane în sediu, după scanarea codului QR, aceasta va apasă butonul de „Validați Cod QR Intrare”. În urma acestei acțiuni, persoana va fi căutată în baza de date în funcție de CNP-ul extras în urma decriptării codului unic. În cazul în care persoana va fi găsită în baza de date, va avea acces în sediu, fiind semnalizat acest lucru prin aprinderea unui led verde pentru 7 secunde. În același timp, în baza de date se vor salva persoana, ora și data la care aceasta a intrat in sediu. În caz contrar, dacă persoana nu va fi găsită în baza de date, accesul acesteia va fi restricționat. Acest lucru este semnalizat prin aprinderea unui led roșu timp de 5 secunde. În situația în care se încearcă validarea fără a se fi scanat un cod QR, acest lucru va fi tratat ca o excepție si va fi semnalizat printr-un mesaj sugestiv, neavând niciun efect asupra ledurilor de acces și, implicit, nicio schimbare în baza de date nu va avea loc.

La ieșirea unei persoane din sediu, aceasta va trebui să apese un buton conectat la modulul Arduino Uno, la portul de intrare. Apăsarea acestuia va fi urmată de scanarea codului unic din aplicația mobilă. Scanarea codului va rezulta un CNP în caseta de text, iar pe baza CNP-ului, persoana va fi căutată în baza de date. Dacă persoana va fi găsită în baza de date, accesul acesteia va fi permis, fapt semnalizat prin aprinderea ledului verde timp de 7 secunde. În timpul garantării accesului, va fi calculat și numărul de minute pe care persoana le-a petrecut în firmă pe baza înregistrării orei intrării în firmă, urmând să se facă un update în baza de date pentru a introduce numărul de minute petrecute de persoana respectivă pentru pontaj. Pentru ca acest lucru să fie posibil, modulul Arduino va comunica cu componenta Cloud și va insera în baza de date informațiile prin intermediul Internetului. În cazul în care codul rezultat în urma decodificării codului QR nu este găsit în baza de date, accesul persoanei va fi restricționat. Acest lucru va fi semnalizat prin aprinderea unui led de culoare roșie timp de 5 secunde . Similar ca în cazul în care persoana ar intra în sediu, programul nu va permite validarea unei persoane care nu a finalizat cu succes scanarea codului QR unic. Acest lucru va fi tratat ca o excepție și va fi semnalizat printr-un mesaj sugestiv, neavând niciun efect asupra ledurilor de acces și, implicit, nicio schimbare în baza de date nu va avea loc.

Dacă în cadrul inserărilor, respectiv actualizărilor, în baza de date intervine o problema cauzată de lipsa unei conexiuni la Internet, această situație va fi rezolvată prin salvare locală a datelor pe o durată de 24h.

*Componenta RaspberryPi*

Modulul RaspberryPi va avea conectate la portul de ieșire o cameră și 3 led-uri de culori diferite: led-ul albastru deschis va indica funcționalitatea sistemului, cel de culoare roșie semnalizează că persoana nu a fost identificată cu succes și nu i se permite accesul în firmă, iar cel de-al treilea led, de culoare verde, indică că angajatul a fost identificat în baza de date și are acces în parcare. Astfel, programul din modulul RaspberryPi indică starea sistemului, astfel că utilizatorul poate vedea dacă sistemul este pornit în cazul în care se poate observa un led albastru aprins și este oprit în caz contrar. Oprirea sistemului corespunde opririi ledului albastru și incapacității unei persoane de a mai scana numărul de înmatriculare a mașinii.

La intrarea unei persoane în parcare, se va scana numărului de înmatriculare. În urma acestei acțiuni, persoana va fi căutată în baza de date. În cazul în care persoana va fi găsită în baza de date și numărul maxim de locuri de parcare nu a fost atins, va avea acces în sediu, fiind semnalizat acest lucru prin aprinderea unui led verde pentru 7 secunde. În același timp, în baza de date se vor salva persoana, ora și data la care aceasta a intrat in sediu. În caz contrar, dacă persoana nu va fi găsită în baza de date sau a fost atins numărul maxim de locuri de parcare, accesul acesteia va fi restricționat. Acest lucru este semnalizat prin aprinderea unui led roșu timp de 5 secunde. În situația

La ieșirea unei persoane din parcare, aceasta va trebui să apese un buton conectat la modulul RaspberryPi, la portul de intrare. În urma apăsării butonului, se va aprinde un led verde timp de 7 secunde.

Dacă în cadrul inserărilor, respectiv actualizărilor, în baza de date intervine o problema cauzată de lipsa unei conexiuni la Internet, această situație va fi rezolvată prin salvare locală a datelor pe o durată de 24h.

In cod se utilizează următoarele tehnologii și biblioteci:

* Python: Un limbaj de programare interpretat de nivel înalt, popular pentru dezvoltarea de aplicații diverse, inclusiv scripting, web scraping, automatizare și AI.
* OpenCV (cv2): O bibliotecă Python utilizată pentru procesarea imaginilor și a videoclipurilor. În acest caz, este folosită pentru a prelucra imaginea și a o pregăti pentru OCR.
* PyTesseract: O bibliotecă Python pentru recunoașterea optică a caracterelor (OCR) care identifică și citește textul din imagini.
* PiCamera: O bibliotecă Python pentru controlul camerei RaspberryPi. Este folosită aici pentru a realiza și a salva capturi de ecran.
* RPi.GPIO: O bibliotecă Python pentru controlul pinilor GPIO (General Purpose Input Output) ai RaspberryPi. În acest caz, este folosită pentru a controla LED-urile și butoanele.
* mysql.connector: Un driver Python pentru interacționarea cu baze de date MySQL. În acest caz, este folosit pentru a extrage și a modifica datele dintr-o bază de date MySQL.
* time: O bibliotecă standard Python folosită pentru a manipula și a opera cu timpul. Aici este folosită pentru a gestiona pauzele de timp (sleep).
* Raspberry Pi: Un microcomputer de dimensiuni reduse, utilizat pentru a rula codul.

Pe lângă acestea, codul implică și lucrul cu o bază de date MySQL, folosind comenzi SQL pentru a extrage și a actualiza datele. Acest lucru implică cunoașterea limbajului SQL.

*Componenta Web*

Aplicația noastră WEB responsive este construita din mai multe pagini cu legături intre ele. Astfel, in fișierul App.js sunt importate fișierele .js corespunzătoare fiecărei pagini (*LoginPage, UserProfilePage, AdminProfilePage, UserReportPage, AllUsersReports, OfficeAllocationPage, ChairAllocationPage*), librăria *react-router-dom* pentru *routes* si apoi sunt definite path-urile dintre pagini.

1. *Pagina de Login.* Utilizatorul trebuie sa introducă datele contului primit de la firma, adresa de email si parola care sunt stocate într-o baza de date. La introducerea acestor date pot apărea următoarele doua validări de input:

* daca emailul si parola nu corespund cu cele din baza de date a firmei, se va afișa mesajul “Invalid email or password”
* daca la adresa de email nu se scrie semnul “@” se va afișa mesajul “Please include an “@” in the email address

La introducerea parolei exista opțiunea de a ascunde parola sau nu prin iconița din dreapta cu un ochi.

Aplicația noastră permite memorarea in browser a conturilor cu care utilizatorul a fost conectat cu succes pentru accesarea mai rapida ulterior. Când începi sa tastezi apare ca sugestie si daca dai click pe el, se completează automat datele.

Navigarea din pagina de login către profil se face cu ajutorul metodei specifice useNavigate.

1. *Pagina Utilizator*. Fiecare angajat are afișate datele personale pe care nu le poate edita. Acestea sunt preluate din baza de date, doar administratorul le poate modifica si apoi se actualizează automat pe pagina utilizatorului respectiv. Trecerea din pagina de utilizator in cea cu raportul prezentei in firma se face prin intermediul evenimentului *handleGenerateReportClick* din spatele butonului *Generate Report.* De asemenea, cu *navigateToLoginPage* ne întoarcem lapagina de login.

Folosim *useState* pentru a tine minte si in alte pagini informațiile asociate utilizatorului conectat.

1. *Pagina Raport Utilizator.* Datele personale ale utilizatorului pe zile cu ora de intrare respectiv ieșire din firma, timpul petrecut si daca este cu mașina sau nusunt preluate din baza de date si afișate sub forma tabelara. Angajatul le poate doar vizualiza, nu si modifica.
2. *Pagina Administrator.* Administratorulgestionează conturile tuturor angajaților si numărul total de locuri de parcare din firma astfel:

* Poate crea utilizatori noi introducând datele care se stochează in baza de date. Funcția *fetch* face o solicitare HTTP POST la /api/users cu datele utilizatorului trimise ca corpul cererii, convertite în format JSON. Funcția procesează răspunsul primit de la server, transformându-l din format JSON într-un obiect JavaScript cu metoda response.json(). Acest obiect este apoi afișat în consolă cu mesajul "New user added:". În cazul în care apare o eroare în timpul procesării cererii, aceasta este afișată în consolă cu mesajul "Error adding new user:".

Folosim metoda preventDefault() pentru a opri comportamentul implicit al formularului, astfel încât pagina să nu fie încărcată din nou în momentul trimiterii formularului.

* Poate seta numărul total de locuri de parcare. Folosim tot funcția *fetch* pentru a prelua din baza de date numărul actual de locuri de parcare, îl setam pe alta valoare si apoi îl salvam in baza de date din nou cu ajutorul evenimentului din spatele butonului *Save*.
* Poate vizualiza raportul tuturor angajaților din firma.

1. *Pagina Raportul Tuturor Utilizatorilor*. Sunt afișate sunt forma tabelara datele ce cuprind prezenta tuturor angajaților in firma. Trecerea din pagina de administrator in cea cu raportul prezentei tuturor angajaților in firma se face prin intermediul evenimentului *handleGenerateAllReportClick* din spatele butonului *Generate For All Employees.*

Pentru aspectul paginilor web am folosit Css-ul (**Cascading Style Sheets)** care este un **limbaj de stilizare** al elementelor si al tagurilor html pus la dispoziție de Visual Studio Code. Astfel, pentru fiecare pagina din aplicația WEB am creat cate un fișier cu extensia .css si am scris linii de cod corespunzătoare pentru editarea textului si fundalului.