Universitatea Tehnica Cluj
Facultatea de Automatica si Calculatoare
Specializarea Calculatoare

Documentatie Assignment 1

Student: Ciuciuc Bogdan Adrian
Indrumator de laborator:
Antonesi Mircea Gabriel
Grupa 1 (la laborator 4)
An: 4

Cuprins

Descrierea aplicatiei....3
Diagramele asociate aplicatiei....5
Detalii tehnice – back-end....6
Detalii tehnice – front-end....7
Deployment....8
Bibliografie....9

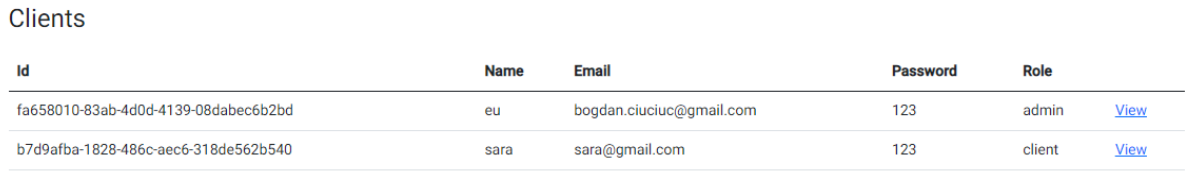
Descrierea aplicatiei

Aceasta aplicatie este o aplicatie web de tip client-server pentru gestionarea clientilor si a device-urilor lor. Pentru partea de Client (front-end) s-a folosit Angular 14, pentru partea de server (back-end) s-a folosit .NET 6.0 (C#), iar pentru partea de baza de date s-a folosit MSSQL. In aceasta aplicatie, exista doua tipuri de utilizatori – admin si client. Pentru ca un utilizator sa primeasca un rol, acesta trebuie sa se autentifice. Actiunile sau operatiile pe care acesta le poate face depind de rolul pe care il primeste printr-un JWT.

Dupa ce un utilizator se autentifica, acesta este redirectat catre Home page.

Adminul poate sa vada orice pagina prin acel JWT primit, iar clientul poate vedea doar o parte din paginile de web posibile (ex.: clientul nu poate vedea lista clientilor, iar cand intra pe acea pagina se afiseaza un mesaj - “No clients found or you don’t have access”).

Fereastra adminului:

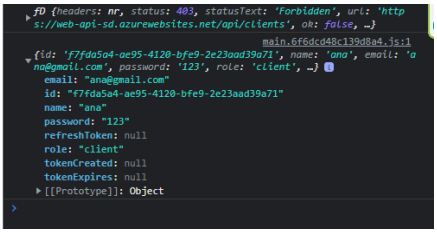


Fereastra clientului:

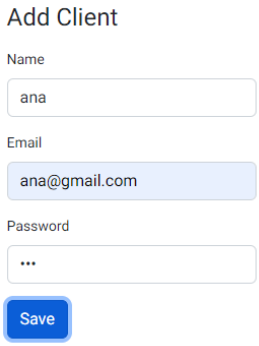
De asemenea, clientul nu are dreptul la anumite operatii – spre exemplu adaugarea unui nou client. Utilizatorul cu rol de admin poate face operatii de create, update, delete pentru utilizatori si pentu dispozitive, iar cel cu rol de client nu poate, primind o eroare in consola. Cu toate ca pagina i se deschide si poate adauga in campuri valori, acesta nu va putea niciodata salva acele valori – salvarea va da o eroare 403 (forbidden access).

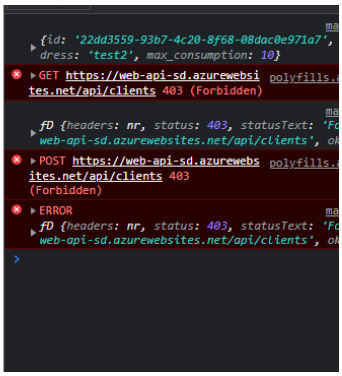
Consola (si rezultatul operatiei) pentru admin:

f7fda5a4-ae95-4120-bfe9-2e23aad39a71	ana	ana@gmail.com	123	client	View



Consola pentru client:





Diagramele asociate aplicatiei

Diagrama, in mare, a acestei aplicatii web:

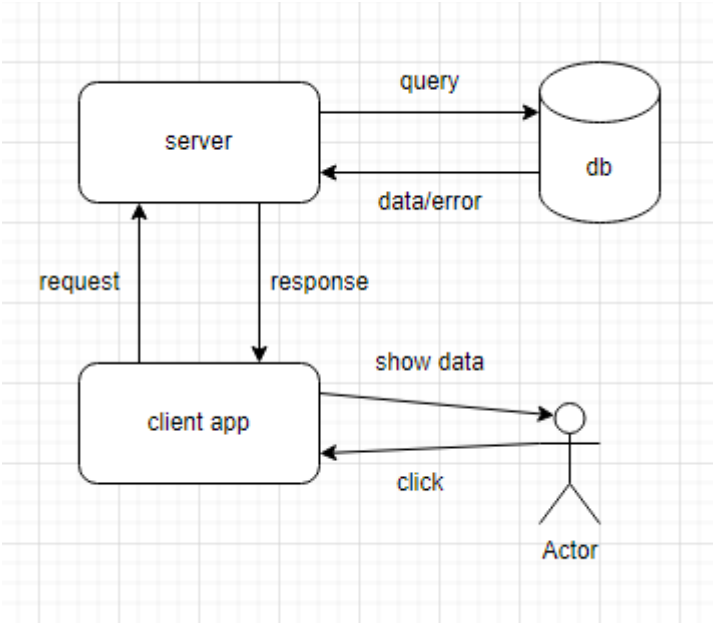
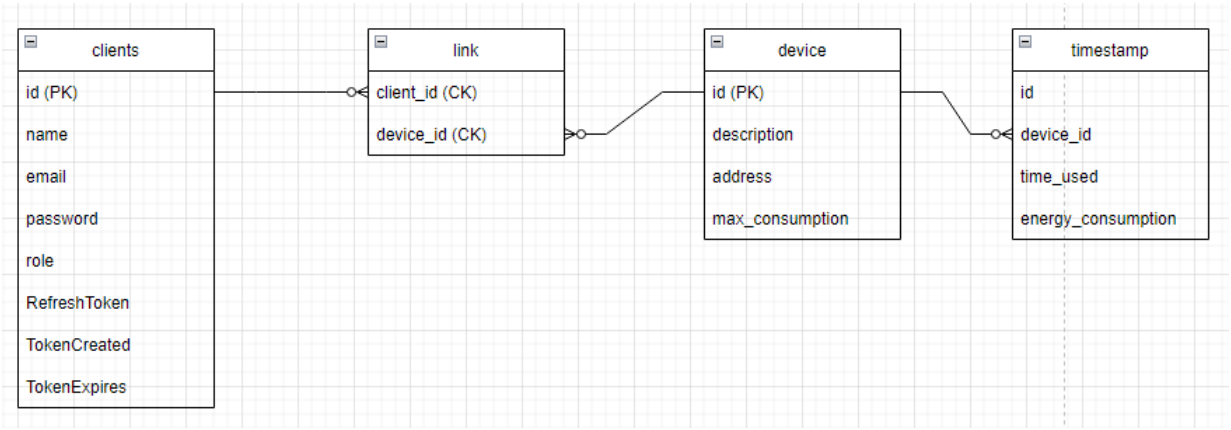


Diagrama bazei de date:



Detalii tehnice – back-end

In partea de back-end s-au declarat modelele bazelor de date dupa care s-a realizat un migration catre baza de date. Cu ajutorul framework-ului .NET 6.0 si namespace-ului Microsoft.EntityFrameworkCore, s-a creat in mod global un context al bazei de date pentru usurinta accesarii si crearii bazei de date.

Pentru modelele Client si Device, s-au creat controllere care sunt folosite pentru a pune sau a aduce date legate de clients/devices din sau in baza de date. Din nou, pentru usurinta, controllerele sunt adnotate cu [ApiController] pentru a specifica faptul ca acele clase vor fi folosite pentru a asista partea de front-end. Ele sunt adnotate si cu [Route(“api/[controller]”)] pentru a specifica ruta pe care front-end-ul poate accesa aceste controllere (ex.: localhost:7203/api/clients). Controllerul pentru clienti a fost adnotat in plus cu [Authorize(Roles = “admin”)] pentru a specifica faptul ca doar utilizatorii cu rolul de admin are acces la acest controller si la functiile din el.

Metodele controllerelor care asista front-end-ul sunt si ele adnotate cu [HttpGet(“path”)]/[HttpPost(“path2”]/etc. pentru a specifica ce tip de request asteapta si la ce adresa (ex.: localhost:7203/api/clients/path). In cazul in care asteapta valori de la front-end, metodele sunt adnotate si cu [Route(“{id:Guid}”)], iar parametrii metodelor sunt adnotati cu [FromRoute]. Daca este necesar un id (de tip Guid) ce ar veni primul sau [FromBody] in cazul in care se asteapta un format .json. Valorile din formatul .json urmeaza sa fie puse de catre .NET 6.0 in locul in care trebuie in variabila declarata ca parametru. Metodele sunt de tip async Task<IActionResult> pentru a putea rula bucati de cod care pot fi executate in mod asincron pe procesoare diferite si pentru a arata faptul ca va returna un IActionResult care in acest caz este un cod http (200 – OK, 401 – Not authorized, 403 – Forbidden Access, 405 – Bad Request, 409 – Conflict).

In continuare, s-a creat un controller special pentru autentificare care are doar adnotarile de [ApiController] si [Route(“api/[controller]”)]. In acest controller se realizeaza logica de log in, sign up si generare a token-ilor JWT. Token-urile sunt ulterior date la utilizatori in primul rand pentru a diferentia rolurile, iar in al doilea rand pentru a-i autentifica. Token-urile dureaza cate o zi, iar in cazul in care un utilizator se logheaza pe un alt cont intainte de termenul expirarii, tokenul lui este sters.

Pentru metodele din acest controller sunt folosite din nou adnotarile ce au fost folosite pentru controllerele de client/device, modificate pentru a corespunde cu solutia problemei.

Detalii tehnice – front-end

In partea de front-end s-au declarat modelele pentru client/device pentru a putea trimite datele in acelasi format dintr-o parte in alta. Fiind folosit angular, fiecare component este format dintr-un fisier HTML, un fisier TypeScript pentru logica din spatele datelor trimise catre fisierul HTML, un fisier TypeScript pentru specificatii si un fisier CSS pentru stilizarea afisarii (care momentan nu a fost folosita).

Pentru inceput, fiecare component/modul importat este declarat in fisierul app.module.ts pentru ca aplicatia sa stie de existenta lui. Apoi, pentru ca paginile web sa fie luate in considerare de catre aplicatie, in fisierul app-routing.module.ts toate componentele au primit un path. In continuare, paginile web au fost initializate cu un navigation bar in care au fost adaugate link-uri catre ele (ex.: lista clientilor, lista device-urilor, autentificare etc.).

Clients/devices au 3 componente care sunt aproape identice (difera numele variabilelor si functiilor) deoarece operatiile sunt aceleasi (view all, add, edit/delete). Acestea doua au fiecare atasat un fisier de serviciu care face legatura intre front-end si back-end, trimitand back-end-ului ceea ce asteapta si asteptand un mesaj. Acel mesaj este ulterior preluat de component si este fie afisat in consola in caz de eroare, fie se executa o redirectionare.

Clients are 2 componente in plus fata de devices, dar ambele fac referinta si la devices. Este vorba de componentele client-devices si link. Client-devices este o pagina accesata din pagina client-list. In aceasta pagina, utilizatorul cu rolul de admin poate vizualiza toate device-urile care sunt legate de acel utilizator. Din aceasta pagina, adminul poate ulterior sa lege mai multe device-uri la acel utilizator accesand componentul link.

Pe langa aceste categorii de componente, mai exista o categorie care, la fel ca si la partea de back-end, este pusa separat fata de logica modelelor – acesta este componentul de autentificare cu serviciul aferent. Acesta se ocupa de logica de vizualizare a paginii de autentificare care are doua butoane – log in si sign up. In functie de butonul apasat, acest component acceseaza metode diferite ale serviciului lui pentru a face legatura cu back-end-ul. In cazul in care se apasa butonul de sign up, acesta doar trimite un request catre back-end cu datele noului utilizator. In cazul in care este apasat butonul de log in, pe langa requestul trimis catre back-end, mai salveaza token-ul primit de la back-end intr-un local storage.

Deployment

Pentru partea de deploy a fost aleasa partea de deploy pe Azure deoarece este mai simpla, mai rapida, nu necesita fisiere in plus fata de codul sursa al aplicatiilor si nici memorie RAM extra. A fost creat un resource group pe site-ul portal.azure.com pentru a uni cele doua aplicatii si baza de date.

Primul element caruia i s-a facut deploy a fost baza de date. Pentru aceasta a fost nevoie de crearea unui server SQL. In acest server s-a creat baza de date. Connection string-ul serverului a fost mai apoi pus in locul connection string-ului bazei de date locale din aplicatia de back-end. Avand acel connection string, din Visual Studio 2022, s-a putut face conexiunea catre acel SQL server in care s-au creat tabelele dupa acelasi sql query. Pentru permiterea aplicatiei de back-end de a avea acces la server, s-a adaugat o regula de firewall pentru adresa IP a back-end-ului.

Al doilea element caruia i s-a facut deploy a fost back-end-ul. Pentru acesta s-a creat pe site-ul Azure un Web App caruia i s-a downloadat publish profile-ul. Din Visual Studio 2022 s-a dat publish la aplicatie si s-a importat profilul downloadat. Pentru a face posibila conexiunea dintre back-end si front-end, s-a adaugat un CORS pentru permiterea front-end-ului de a accesa aplicatia.

Ultimul element caruia i s-a facut deploy a fost front-end-ul. Pentru acest lucru, s-a facut un nou Web App pe site-ul Azure, s-a downloadat extensia Azure in Visual Studio code, s-a facut build la solutie si s-a dat deploy doar build-ul facut anterior (din folderul dist).

Bibliografie

1. https://stackoverflow.com/
2. https://www.youtube.com/
3. https://learn.microsoft.com/
4. https://portal.azure.com/
5. https://www.google.com/