UNIVERSITATEA “LUCIAN BLAGA” DIN SIBIU

FACULTATEA DE INGINERIE

DEPARTAMENTUL DE CALCULATOARE ŞI INGINERIE ELECTRICĂ

**Internet of Things (IoT)**

Îndrumar de proiect

Ingineria Sistemelor Multimedia

An IV

Prof. Dr. Ing. [Adrian Florea](mailto:adrian.florea@ulbsibiu.ro) Ing. [Daniel POPA](mailto:idaniel.popa@ulbsibiu.ro)

Ing. [Claudia Andreea BANCIU](mailto:claudia.banciu@ulbsibiu.ro)

**Cuprins:**

[**0. Format Proiect 3**](#_gtho7lxk2d1y)

[**1. git vs. Github 5**](#_4iu2geuexuf7)

[1.1 Aspecte teoretice 5](#_288p1zfgpj3d)

[1.2 Noțiuni practice 5](#_e3is5tuwaexw)

[**2. Introducere în Internet of Things 11**](#_hp6hjx7mvyn7)

[2.1 Aspecte teoretice - IoT 11](#_8xccb48nckg7)

[2.2 Noțiuni practice 13](#_1m5ukf8d0rsp)

[2.2.1 Setup - NodeMCU 13](#_g88dtlhk6151)

[2.2.2 Structura unui program Arduino 17](#_ktmn8d1jidn2)

[2.2.3 Notiuni de baza ale limbajului C/C++ Arduino 17](#_drqrzr30hqcs)

[Functiile Setup si Loop 17](#_uamyfsvfquj9)

[Serial Print 17](#_t5q7q6l0t3g)

[Tipuri de Date + Variabile 18](#_5v97dr6oap0)

[Condiționale: If, Else 18](#_6avncj9ph6u)

[While 18](#_5isvxye7r8zm)

[For 19](#_qqlfqmvbkxth)

[Functii 19](#_elgpx4848thg)

[Librarii 19](#_7tg49vvqnp8q)

[2.2.4 Exerciții 20](#_kgug6mba079u)

[2.2.5 Challenges 23](#_pkmtfxhv9ux)

[2.2.6 Useful Stuff 23](#_3vhcch8ezcx0)

[**3. Colectare Date Pubela 25**](#_kxsmbm9oa3td)

[3.1 Aspecte teoretice - Webserver vs API 25](#_mv48ddao4pd)

[3.2 Noțiuni practice 29](#_tecyunbxxink)

[**4. Aplicatie web 31**](#_93fdbhugzxb8)

[4.1 Aspecte teoretice 32](#_3w9b68y5xy8k)

[ASP.NET 32](#_k1neh6iwob7w)

[SQLite 33](#_to7wisuqhjqh)

[Avantaje SQLite 33](#_e1jg011rrg3i)

[Dezavantaje SQLite 33](#_xwcqjqa4np8b)

[Avantaje SQL Server 33](#_nsxw4tinwq7k)

[Dezavantaje SQL Server 33](#_3nbetox65igp)

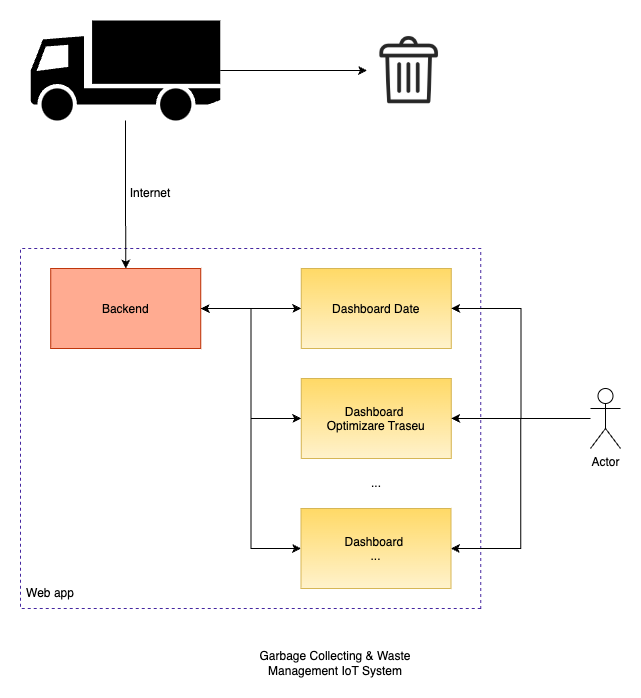
[4.2 Notiuni practice 34](#_tynrjxb8pbqw)

[Cerinte 37](#_56mn94a0x8y7)

[**5. Functionalitati 39**](#_3d455kpy9rh0)

# **0. Format Proiect**

Acest proiect va provoacă să proiectati și să construiti un sistem inteligent IoT pentru serviciul public de salubrizare al unui oras. Scopul proiectului este de a optimiza procesul de colectare al deseurilor, combinand cunostiinte din domeniile IoT, web development si algoritmi de optimizare pentru abordarea problemelor din lumea reala.



# 

# **1. git vs. Github**

Vom folosi git si Github pentru a pastra istoricul proiectului. Toate modificarile le vom salva intr-un repository de Github.

## **1.1 Aspecte teoretice**

Introducere: [git vs. Github](https://docs.google.com/presentation/d/1nD5QJUJgQ_gMjyymWIQ2gYSlvN1_BFK45Ut43A4GA1Q/edit?usp=sharing)

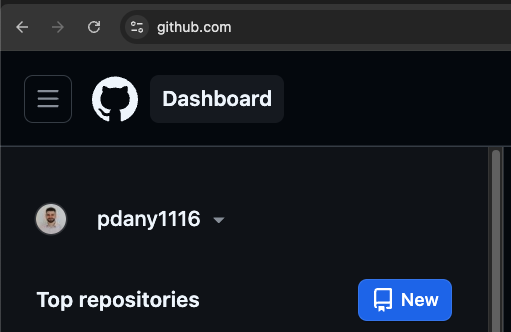
## **1.2 Noțiuni practice**

**Primul Github Repository**

**Pasul 1: Crearea unui cont pe GitHub**

1. **Accesează** <https://github.com>.
2. **Creaza un cont** (cu email personal)

**Pasul 2: Crearea primului repository**

1. **Creează un repository nou**
2. **Numeste repository-ul** first-test-repository
3. **Apasă pe butonul Create repository**.

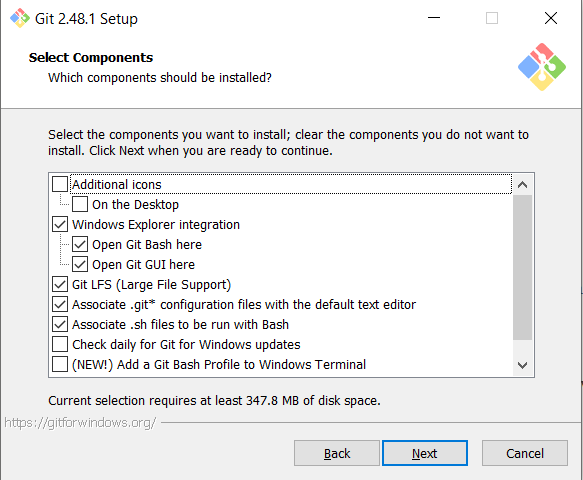
**Pasul 3: Instalarea Git pe Windows**

1. **Accesează site-ul oficial Git** <https://git-scm.com/downloads/win>.
2. **Descarca versiunea 32b sau 64b (depinde de sistemul tau de operare**)

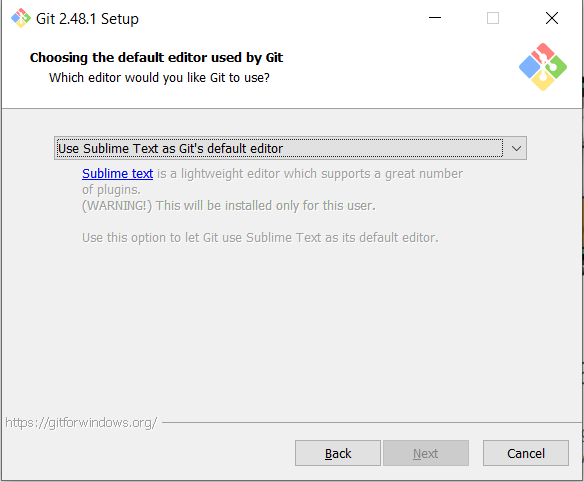


**Pasul 2: Instalează Git**

1. **Deschide installer-ul** .exe
2. **Alege urmatoarele configurari:**

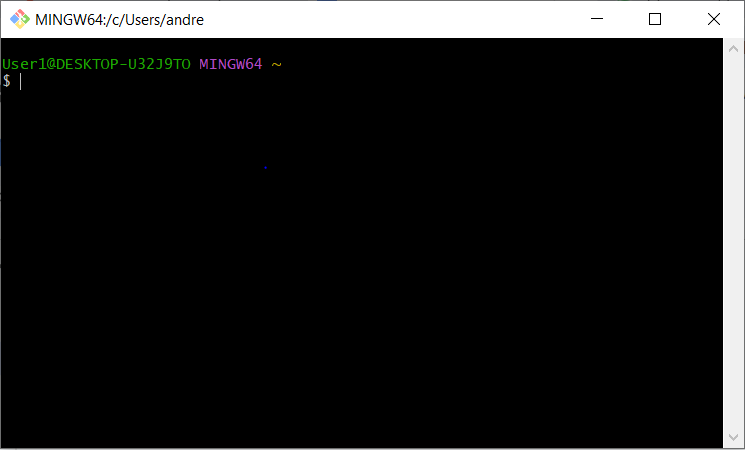
****

1. **Selectează editorul de text default**: Poți alege editorul de text preferat (recomand Notepad++ sau Sublime, trebuie instalat de pe link-ul afisat). Dupa instalare Notepad++/Sublime, inchideti procesul de instalare si porniti-l din nou.



4. **Next, Next, Next…**

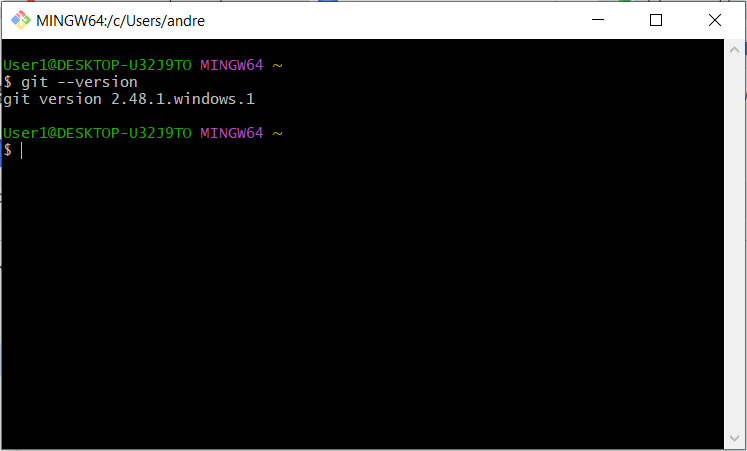
5. **Launch Git Bash**

****

**Pasul 3: Deschide Git Bash**

1. **Deschide Git Bash**După instalare, poți accesa Git Bash fie din meniul Start, ori cu click dreapta **Git Bash Here** din orice director.
2. **Verifică versiunea Git**

git --version



**Pasul 4: Configurarea Git**

Înainte de a începe să folosești Git, trebuie să configurezi numele tău și adresa ta de email. Aceste informații vor fi folosite pentru a-ți semna commit-urile.

**Configurează numele utilizatorului**  
git config --global user.name "Numele Tău"

**Configurează adresa de email**  
git config --global user.email "email@exemplu.com"

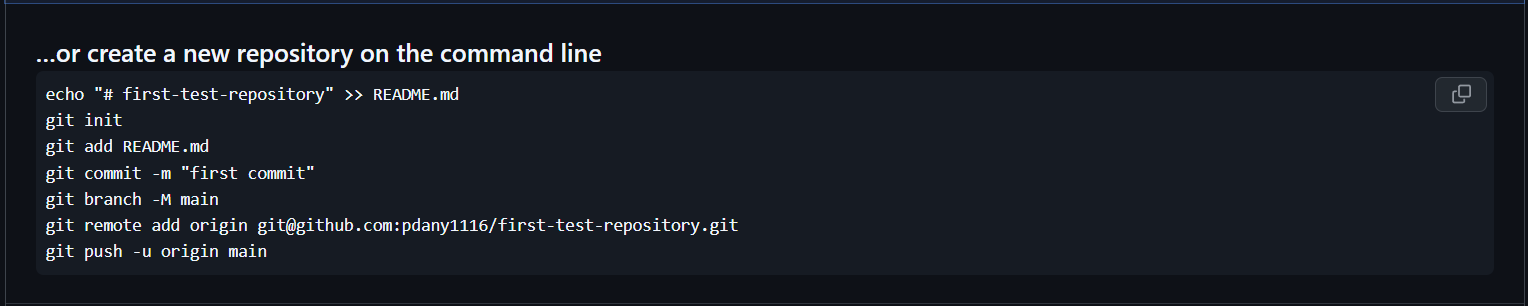
**Pasul 5: Folosirea Git**

După ce ai terminat instalarea și configurarea, poți începe să folosești Git pentru a gestiona proiectul pe care l-ai creat mai devreme.

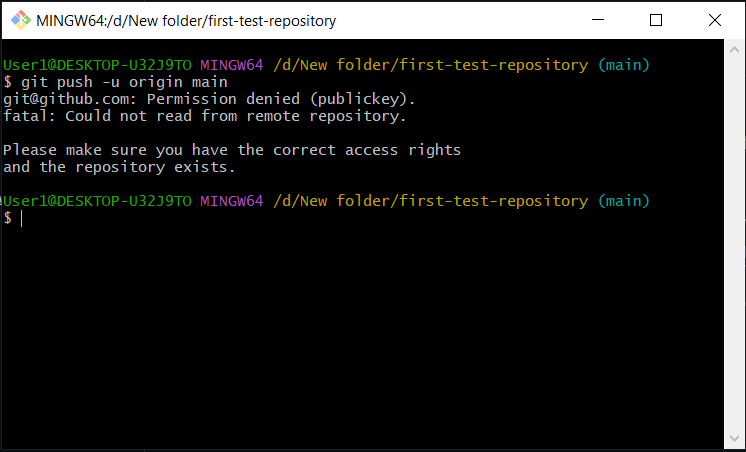
Trebuie sa cream un director nou cu numele **first-test-repository** (acelasi nume cu repository). Intrati in repository.

Deschideti o consola Git Bash (click dreapta **Git Bash Here).**

Accesati repository-ul vostru [https://github.com/<numele-tau>/first-test-repository](https://github.com/%3Cnumele-tau%3E/first-test-repository) si urmariti pasii pentru initializarea unui repository nou.



La ultima comanda, veti intalni o eroare:



Cauza erorii este faptul ca Github nu permite oricui necunoscut sa incarce cod pe platforma lor. Deci ne trebuie o metoda sa ne autenitifcam la Github ca el sa stie cine a incarcat codul si daca are permisiile necesare. Ne vom crea un set de chei ssh (1 cheie privata si 1 cheie publica).

**Deschide Git Bash**

**Generareza setul de chei SSH:**ssh-keygen -t ed25519 -C "[emailul.tau@example.com](mailto:emailul.tau@example.com)"

O sa iti ceara path-uri unde sa instalese cheile, apasa Enter de 3 ori.

**Adăugarea cheii publice pe GitHub:**  
cat ~/.ssh/id\_ed25519.pub

Cheia are formatul ssh-ed25519 AAAAC3NzaC…..

Mergi în **GitHub > Settings > SSH and GPG keys** și adaug-o.

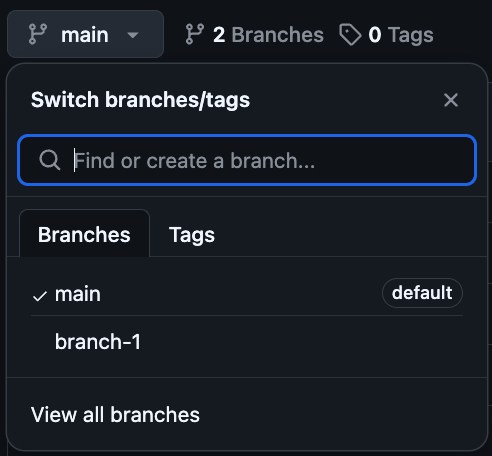
**Incearca din nou ultima comanda**

git push -u origin main

Accesati repository-ul vostru [https://github.com/<numele-tau>/first-test-repository](https://github.com/%3Cnumele-tau%3E/first-test-repository) si observati ce s-a schimbat.

**Cerinte**:

* Local, creaza un branch nou branch-1. Muta-te pe noul branch. Adauga un fisier example.txt care sa contina numele tau. Creaza un commit. Incarca schimbarile pe repository-ul de pe Github.
  + Rezultat: pe Github trebuie sa apara noul branch si sa contina noul commit



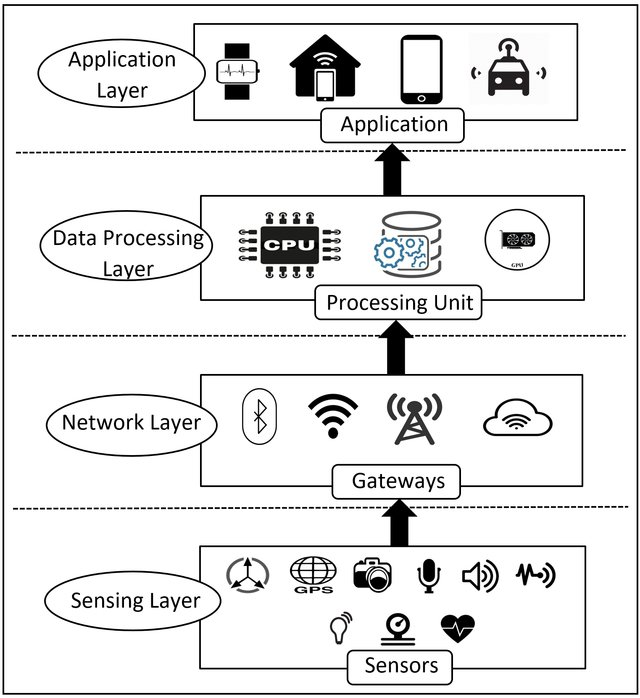
* Pe Github, creaza un Pull Request din branch-1 in main si fa-i merge.
* Local, muta-te inapoi pe branch-ul main si adu-ti local schimbarile de pe local.
  + Rezultat: pe laptop ar trebui sa ai in directorul curent fisierul example.txt

# **2. Introducere în Internet of Things**

## **2.1 Aspecte teoretice - IoT**

**Internet of Things (IoT)** se referă la rețeaua colectivă de dispozitive conectate și la tehnologia care facilitează comunicarea între dispozitive și cloud, precum și între dispozitive.

Un **obiect (thing)** poate fi un automobil care are senzori încorporați, un smartwatch, o persoană cu un implant de monitorizare cardiacă, un animal de fermă cu un biocip, sau orice alt obiect care **este capabil să transfere date printr-o rețea**.



Un sistem de tip IoT este bazat pe o arhitectura din 4 nivele:

**A. Stratul de detecție (Sensing Layer)**

Principalul scop al stratului de detecție este de a obține date din lumea reală. Este format din mai mulți senzori. Utilizarea mai multor senzori pentru aplicații este una dintre caracteristicile principale ale dispozitivelor IoT. Senzorii sunt conectati la untitatea de procesare (microcontroller). Senzorii pot fi clasificați în trei mari categorii:

1. *Senzori de mișcare*: Măsoară modificarea mișcării, precum și orientarea dispozitivelor.
2. *Senzori de mediu*: Sunt încorporați în dispozitivele IoT pentru a detecta modificarea parametrilor de mediu în proximitatea dispozitivului.
3. *Senzori de poziție*: Se ocupă de poziția și locația fizică a dispozitivului. Cei mai comuni senzori de poziție utilizați în dispozitivele IoT sunt senzorii magnetici și senzorii GPS.

**B. Stratul de rețea**

Stratul de rețea acționează ca un canal de comunicare pentru a transfera datele, colectate în stratul de detecție, către alte dispozitive conectate. În dispozitivele IoT, stratul de rețea este implementat prin utilizarea diverselor tehnologii de comunicare (de exemplu, Wi-Fi, Bluetooth, Zigbee, Z-Wave, LoRa, rețea celulară etc.) pentru a permite fluxul de date între alte dispozitive din cadrul aceleiași rețele.

**C. Stratul de procesare a datelor**

Stratul de procesare a datelor constă în unitatea principală de procesare a datelor a dispozitivelor IoT. Stratul de procesare a datelor preia datele colectate în stratul de detecție și analizează datele pentru a lua decizii pe baza rezultatului. În unele dispozitive IoT stratul de procesare a datelor salvează, rezultatul analizei anterioare pentru a îmbunătăți experiența utilizatorului. Acest strat poate partaja rezultatul prelucrării datelor cu alte dispozitive conectate prin intermediul stratului de rețea.

**D. Stratul de aplicații**

Stratul de aplicații implementează și prezintă rezultatele stratului de procesare a datelor intr-o interfata grafica utilizatorilor. Deobicei, aceasta interfata este o aplicatie mobila sau un site web, unde utilizatorii pot analiza datele colectate sau pot interactiona cu sistemul.

## **2.2 Noțiuni practice**

**NodeMCU** este un mediu de dezvoltare software și hardware open-source, construit în jurul unui System-on-a-Chip foarte ieftin numit ESP8266. Placa de dezvoltare NodeMCU este format din cipul wif ESP8266 dezvoltat de Espressif Systems cu protocol TCP/IP.

### 2.2.1 Setup - NodeMCU

1. Descarcare Arduino IDE <https://www.arduino.cc/en/software>

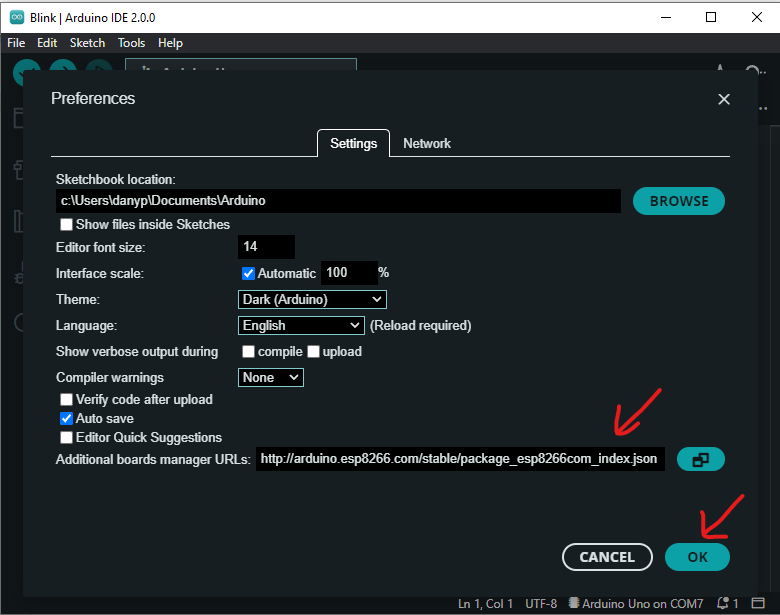
2. Instalare driver:

* (**NODEMCU cu SPATE NEGRU**) Descarcati arhiva, dezarhivati si instalati driver-ul CH341SER pentru sistemul vostru de operare (Windows, Linux, Mac) <https://github.com/nodemcu/nodemcu-devkit/tree/master/Drivers>
* (**NODEMCU cu SPATE ALB**) Descarcati si instalati CP2102 driver pentru Windows <https://www.pololu.com/docs/0J7/all#2>

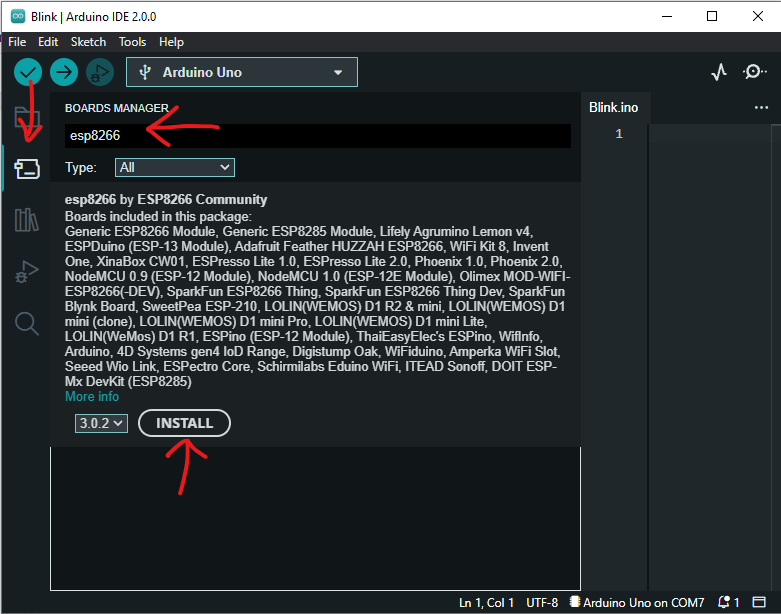
3. Pentru a putea folosi placa de dezvoltare NodeMCU, următoarele configurări trebuie făcute.

1. Deschideti Arduino IDE, accesati File -> Preferences
2. Adauga linkul de mai jos în Additional Boards Manager.

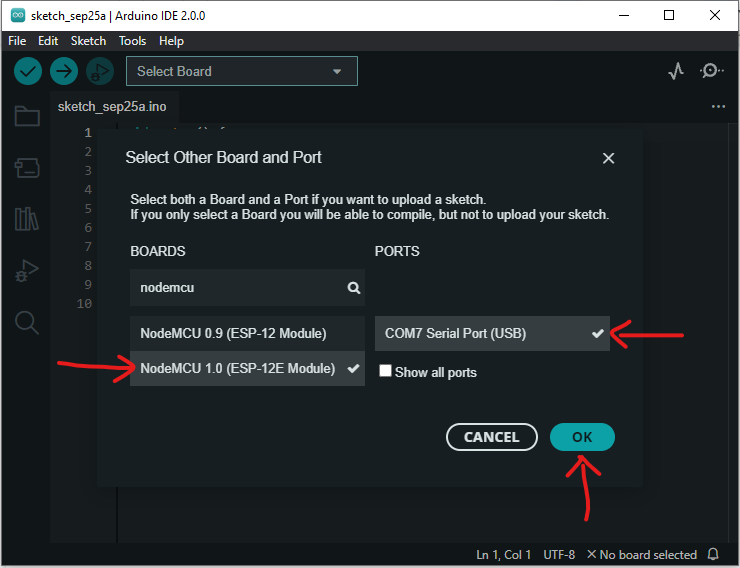
<http://arduino.esp8266.com/stable/package_esp8266com_index.json>



1. Instalează pachetul de placi dezvoltare bazate pe ESP8266.

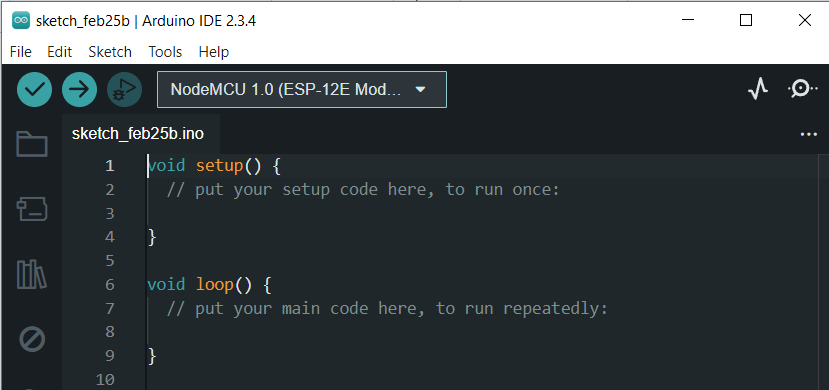
****

1. Selectează placa de dezvoltare NodeMCU 1.0



### 2.2.2 Structura unui program Arduino

Un program pe care il scriem si il incarcam pe placa de dezvoltare pentru a rula este scris in limbajul de programare **C/C++**. Programul se salveaza intr-un fisier de tip **.ino** si se numeste **Sketch**. Programul are 2 functii principale **setup** si **loop**.



### 2.2.3 Notiuni de baza ale limbajului C/C++ Arduino

#### Functiile Setup si Loop

Funcțiile setup și loop sunt esențiale în orice program Arduino. Funcția setup rulează o singură dată la începutul programului și este folosită pentru inițializări, în timp ce funcția loop rulează în mod continuu, repetându-se la infinit și este locul în care puneți codul principal al programului.

| void setup() {  // Inițializări aici  }  void loop() {  // Cod principal aici  } |
| --- |

#### Serial Print

Descriere: Funcția Serial.print este folosită pentru a afișa date pe portul serial pentru a le monitoriza și depana.

| void setup() {  Serial.begin(9600); // Inițializare port serial  }  void loop() {  int val = 42;  Serial.print("Valoarea este: ");  Serial.println(val); // Afișează valoarea și trece la o nouă linie in consola  delay(1000); // Așteaptă 1 secundă  } |
| --- |

#### Tipuri de Date + Variabile

Descriere: Arduino suportă mai multe tipuri de date, cum ar fi int, float, și char. Variabilele sunt folosite pentru a stoca și manipula date.

Exemple comune de tipuri de date includ:

* Integer (int): Folosit pentru a stoca numere întregi, cum ar fi -1, 0, 1, 2, etc.
* Float: Folosit pentru a stoca numere zecimale, cum ar fi 3.14 sau 0.5.
* String: Folosit pentru a stoca șiruri de caractere, cum ar fi "Hello, World!".
* Char: Folosit pentru a stoca un singur caracter, cum ar fi 'A'.
* Boolean (bool): Folosit pentru a stoca valori true sau false.

| void setup() {  int a = 5;  float b = 3.14;  char c = 'A';  }  void loop() {  int x = 10;  x = x + 5; // Modificarea variabilei  } |
| --- |

#### Condiționale: If, Else

Descriere: Condițiile permit programului să ia decizii bazate pe evaluarea unei expresii.

| int sensorValue = analogRead(A0);  if (sensorValue > 500) {  digitalWrite(13, HIGH);  } else {  digitalWrite(13, LOW);  } |
| --- |

#### While

Descriere: O buclă while execută un bloc de cod repetat atât timp cât o anumită condiție este adevărată. Buclele while sunt folosite pentru a executa un bloc de cod atâta timp cât o anumită condiție este adevărată. De exemplu:

| // va executa codul atâta timp cât variabila count este mai mică decât 5  int count = 0;  while (count < 5) {  Serial.println(count);  count++;  } |
| --- |

#### For

Descriere: Bucla for este utilizată pentru a executa un bloc de cod de un număr specific de ori. Buclele for sunt utilizate pentru a executa un bloc de cod de un număr specific de ori. Ele includ o inițializare, o condiție de oprire și o expresie de iterare. De exemplu

| // codul din interiorul buclei va fi executat de 5 ori.  for (int i = 0; i < 5; i++) {  Serial.println(i);  } |
| --- |

#### Functii

Descriere: Funcțiile sunt blocuri de cod reutilizabile care pot fi apelate de mai multe ori în program care efectuează o anumită acțiune. Funcțiile permit organizarea și structurarea codului în bucăți mai mici și independente. Ele pot primi parametri și pot returna valori.

| int add(int a, int b) {  return a + b;  }  void setup() {  int result = add(3, 4);  Serial.println(result); // Va afișa 7  } |
| --- |

#### Librarii

Descriere: Bibliotecile Arduino sunt seturi de funcții predefinite pe care le puteți utiliza pentru a adăuga funcționalități suplimentare la proiectele dvs.

| #include <Servo.h>  Servo myservo; // Crearea unei instanțe de servo  void setup() {  myservo.attach(9); // Atribuirea pinului 9 la servo  }  void loop() {  myservo.write(90); // Mută servo la 90 de grade  delay(1000);  myservo.write(0); // Mută servo la 0 grade  delay(1000);  } |
| --- |

### 2.2.4 Exerciții

**Blink v1 cu LED intern**

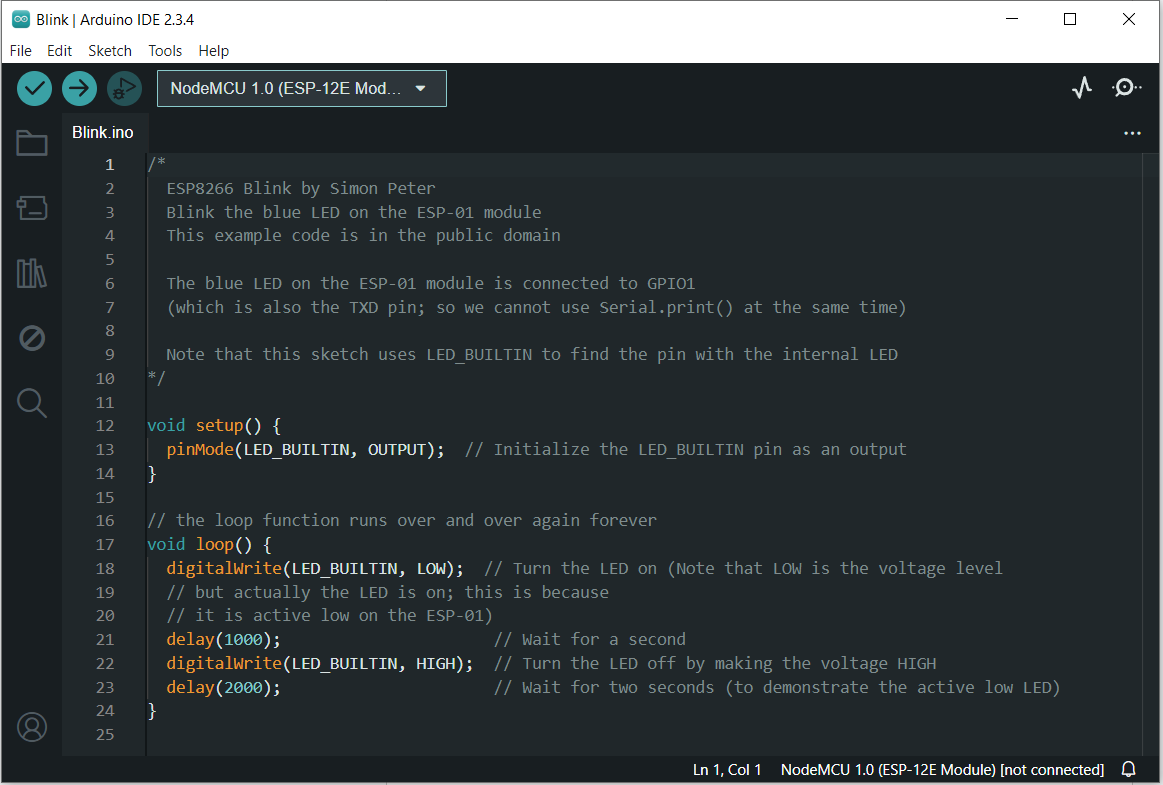
Componente necesare:

1. NodeMCU + Cablu USB

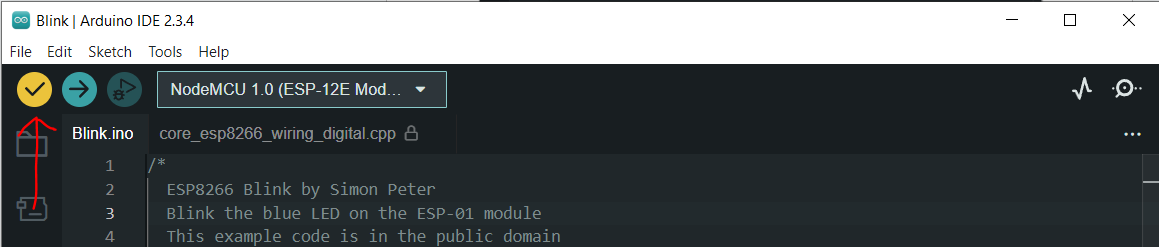
Aplicatia Blink este echivalentul programului ‘Hello World’ pe care il implementezi prima data in orice limbaj de programare. Arduino Uno si NodeMCU au un LED direct integrat pe placa. Exemplul acesta va aprinde si stinge acest LED la un interval de o secunda. Exemplul exista deja in Arduino IDE:

Pentru **Node MCU** il gasiti la: File -> Examples -> ESP8266 -> Blink.

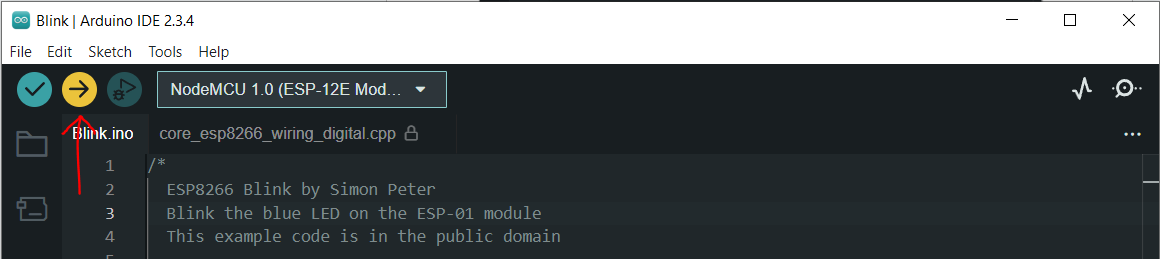
Pentru **Node MCU** acest led se aprinde atunci cand **pinul D4 (GPIO2)** digital este pus in HIGH si se stinge atunci cand pinul D4 este pus in LOW.



Dupa ce am inteles ce face codul putem apasa **Verifiy** pentru a compila codul.



Daca nu a aparaut nici o eroare, putem apasa **Upload** pentru a incarca programul compilat pe NodeMCU.



Ar trebui sa vezi in consola mesajul “Done Uploading”, iar ledul de pe placa NodeMCU sa se aprinda la interval de 1 secunda.

**Gata!**

**De ce nu merge? 🤬**

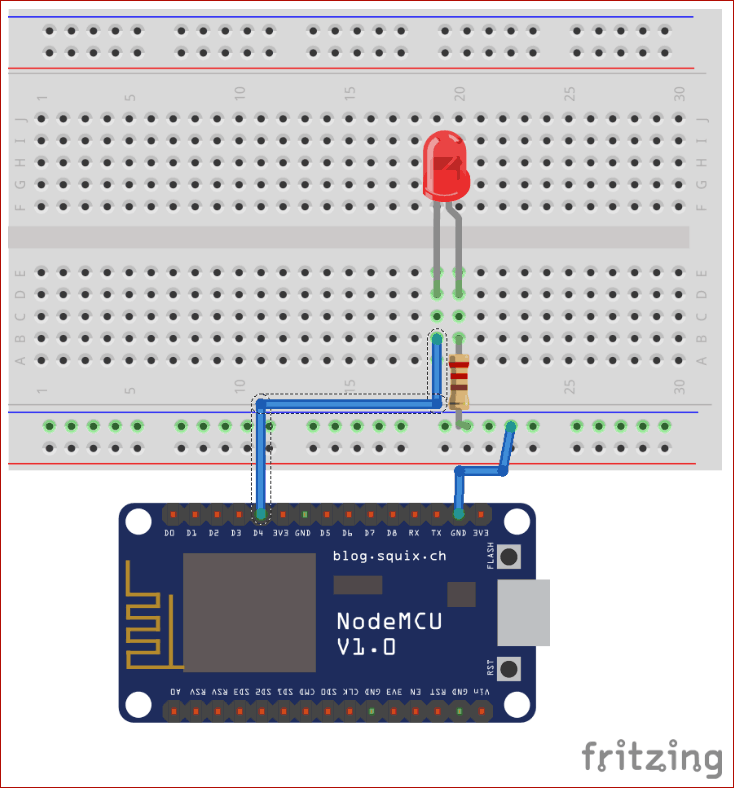
* sigur programul s-a incarcat pe **NodeMCU** (ai văzut mesajul "Done Uploading")?

**Blink v2 cu LED extern**

Componente necesare:

1. NodeMCU + Cablu USB
2. LED
3. Rezistor 220 Ohmi
4. Fire
5. Breadboard

Opriți alimentarea plăcii NodeMCU. Conectati componentele ca în schemele de mai jos. Alimentati din nou placa NodeMCU.

****

**Intrebari:**

1. De ce se aprinde LED-ul extern dacă nu am incarcat programul pe placa?
2. De ce se aprind ambele LED-uri (cel extern și cel integrat)?
3. De ce am ales un rezistor de 220 Ohm?

Resistance = (Supply voltage - LED voltage drop) / Current

= (5.0 volts - 1.4 volts) / 0.015 amps

= 3.6 volts / 0.015 amps

= 240 Ohms ~ 220 Ohms (nearest standard resistor value)

**De ce nu merge? 🤬**

* sigur ai conecta firele ca în schema?
* sigur rezistorul este de 220 Ohmi?
* sigur nu ai pus LED-ul invers?

### 2.2.5 Challenges

1. Foloseste pinul D1 (GPIO5) în loc de D4 (GPIO2) pentru aplicația **Blink v2**.
2. Aprinde ledul pentru 4 secunde și stinge-l pentru 2 secunde pentru aplicația **Blink v2**.
3. Transmite in codul Morse mesajul ‘Salut’ cu ajutorul ledului.

### 2.2.6 Useful Stuff

1. Get Started with Arduino IDE: <https://docs.arduino.cc/software/ide-v2>
2. NodeMCU Pinout: <https://i0.wp.com/randomnerdtutorials.com/wp-content/uploads/2019/05/ESP8266-NodeMCU-kit-12-E-pinout-gpio-pin.png?quality=100&strip=all&ssl=1>
3. Calculare rezistenta <https://upload.wikimedia.org/wikipedia/commons/4/4b/Resistor_Color_Code.svg>
4. Funcții des folosite în programele Arduino:

| **Funcție** | **Descriere** |
| --- | --- |
| pinMode(PIN, STATE); | Aceasta functie seteaza modul unui pin. Pinul este setat pe INPUT când dorim sa citim valori sau pe OUTPUT cand dorim sa trimitem valori. Mai multe detalii despre toate modurile gasiti aici: <https://docs.arduino.cc/learn/microcontrollers/digital-pins> |
| digitalRead(PIN); | Această funcție este utilizată pentru a citi starea unui pin digital. Valoarea este fie LOW, fie HIGH și poate fi stocată într-o variabilă de tip bool. |
| digitalWrite(PIN, SIGNAL); | Utilizați această funcție pentru a trimite un semnal digital LOW sau HIGH pe un anumit pin. |
| analogRead(PIN); | Această funcție este utilizată pentru a citi intrarea analogică de la pinii analogici. Valoarea de intrare poate fi stocată într-o variabilă de tip int.  Intrarea analogică are intervalul de la 0 la 5 volți, iar valoarea este convertită într-un număr binar de la 0 la 1023. 2,5 volți = număr binar 511. |
| analogWrite(PIN, SIGNAL); | Utilizați această funcție pentru a trimite un semnal analogic pe un anumit pin. Valoarea semnalului poate fi specificată ca un număr întreg între 0 și 255. |
| delay(MS); | Această funcție este utilizată pentru oprirea execuției programului pentru numărul specificat de milisecunde. |
| Serial.begin(BAUD-RATE); | Această funcție setează rata de transmisie pentru comunicarea seriala. |
| Serial.print(MESSAGE); | Această metodă este utilizată pentru a emite un mesaj de tip șir de caractere prin comunicarea serioala. |
| Serial.println(MESSAGE); | Folosește metoda print pentru a emite mesajul, iar la sfarsit adauga și caracterele “\r\n” care semnifica un rand liber după mesaj. |

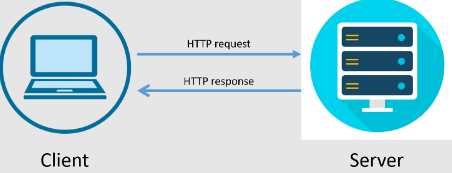
# 

# 

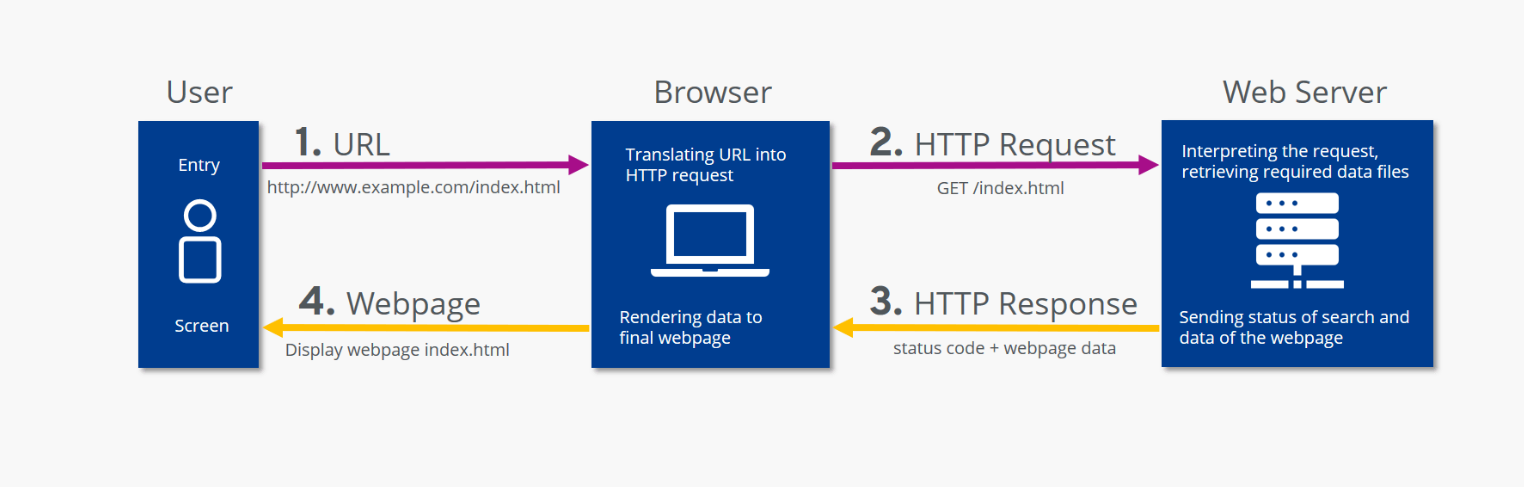
# **3. Colectare Date Pubela**

## **3.1 Aspecte teoretice - Webserver vs API**

Un **server web** stochează și livrează conținutul unui site web - cum ar fi text, imagini, etc. (in general date) - clienților care îl solicită. Cel mai frecvent tip de client este un web browser, care solicită date de la site-ul web atunci când un utilizator il acceseaza. Un server web comunică cu un browser web folosind protocolul HTTP. Conținutul majorității paginilor web este codificat în HTML.



**Cum functioneaza protocolul HTTP?**



1. Utilizatorul tastează *example.com* în bara de adrese a browserului de internet.
2. Browserul trimite cererea respectivă (adică solicitarea HTTP) către serverul web care gestionează domeniul *example.com*.
3. Serverul web primește cererea HTTP, caută fișierul dorit (în acest exemplu, pagina de pornire *example.com*, adică fișierul *index.html*) și începe prin a trimite inapoi un raspuns care informează clientul despre rezultatul căutării cu un status code.
4. În cazul în care fișierul a fost găsit, serverul trimite corpul mesajului după header (adică conținutul propriu-zis). În exemplul nostru, acesta este fișierul *index.html*.
5. Browserul primește fișierul, interpreteaza structura HTML a fisierului si afiseaza elementele site-ului web (paragrafe, tabele, butoane, etc.).

Un **HTTP Request** contine:

1. O **metoda HTTP** (GET, POST, PUT, DELETE, OPTIONS, etc.) care descrie actiunea requestului.
2. **Destinatia** requestului, de obicei o adresă URL, sau adresa IP a web serverului + portul.
3. **Headere**, care permit clientului să transmită informații suplimentare catre server.
4. Un corp (**body**) folosit pentru a trimite date catre server.

Un **HTTP Response** contine:

1. Un **cod de status**, care indică succesul sau eșecul solicitării. Codurile de stare obișnuite sunt 200, 404. Pe langa cod, mai este si un mesaj care sa detalieze statusul (OK, NOT FOUND).
2. **Headere**, care permit serverului să transmită informații suplimentare catre client.
3. Un corp (**body**) folosit pentru a trimite date catre client.

**Diferența dintre un Web Server și un Backend API**

Deși termenii „web server” și „backend API” sunt adesea folosiți împreună, ei au roluri diferite:

* **Un web server** trimite fișiere statice, cum ar fi pagini HTML, către utilizatori.
* **Un backend API** gestionează date dinamice și răspunde la cereri HTTP cu informații structurate de obicei JSON sau XML.

**Exemplu**

Să presupunem că avem un website simplu care afișează o listă de utilizatori.

1. **Web Server:**

Când utilizatorul accesează https://example.com, browserul face o cerere:

| GET / HTTP/1.1  Host: example.com |
| --- |

Serverul răspunde cu o pagină HTML simplă:

| HTTP/1.1 200 OK  Content-Type: text/html  <html>  <body>  <h1>Lista utilizatorilor</h1>  <p>Popa Daniel</p>  <p>Banciu Claudia</p>  <p>Florea Adrian</p>  </body>  </html> |
| --- |

1. **Backend API:**

Dacă pagina trebuie să afișeze utilizatori reali, browserul ar face o cerere către API pentru date:

| GET /api/users HTTP/1.1  Host: example.com |
| --- |

API-ul răspunde cu date JSON:

| HTTP/1.1 200 OK  Content-Type: application/json  [  { "id": 1, "nume": "Popa Daniel" },  { "id": 2, "nume": "Banciu Claudia" },  { "id": 3, "nume": "Florea Adrian" }  ] |
| --- |

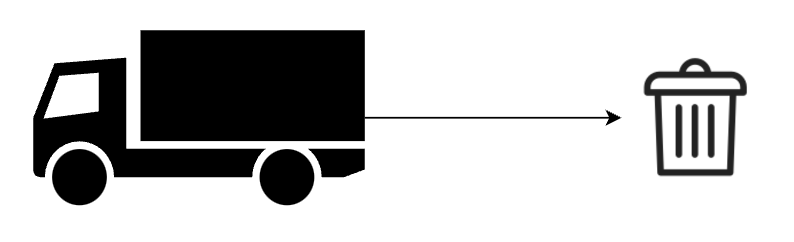
**Diferența cheie**

* Web serverul raspunde cu un fișier HTML, pe care browserul îl afișează.
* Backend API-ul oferă doar date (JSON/XML), pe care un program sau o aplicație le poate folosi dinamic.

## 

## **3.2 Noțiuni practice**

Problema pe care dorim sa o rezolvam in aceasta etapa este sa gasim o metoda sa colectam date despre pubelele cetatenilor pentru a monitoriza in ce zile au scos cetatenii pubelele. O sa ne concentram pe partea de colectare a datelor.

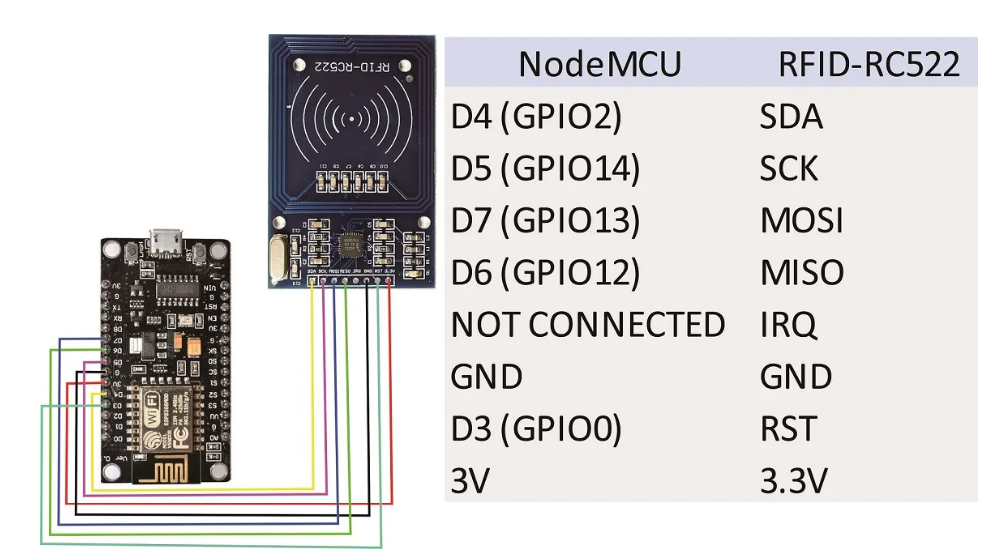


Cerinte:

* Definiti ce date puteti colecta despre evenimentul de colectare a pubelei care ne vor ajuta pe viitor
* Creati o aplicatie Arduino pentru NodeMCU care va afisa pe consola datele colectarii. Configurati schema hardware de mai jos.

Componente:

* NodeMCU
* Cititor RFID RC522
* RFID Tag



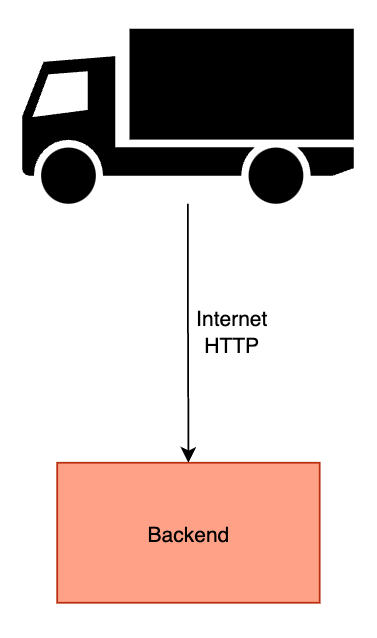
* Functionalitate: Afisati pe consola id-ul la fiecare citire
* Functionalitate: La fiecare citire, trimiteti datele catre webserver-ul aflat la <http://10.14.10.113:3000/api/data>.
  + Ca sa trimiteti datele catre API, puteti sa va ajutati de exemplul pus la dispozitie de libraria pentru ESP8266.
    - Accesati **File -> Examples -> ESP8266HttpClient -> PostHttpClient**
  + Deschideti <https://editor-next.swagger.io/> si introduceti documentatia OpenAPI de mai jos pentru a intelege cum trebuie facut request-ul catre API. Schimbati parametrul “name” sa contina numele vostru.

| openapi: 3.1.1  info:  contact:  name: Popa Daniel  email: idaniel.popa@ulbsibiu.ro  title: Mock API  description: API pentru primirea datelor de la NodeMCU  version: 1.0.0  servers:  - url: http://10.14.10.113:3000  description: Local development webserver  tags:  - name: nodemcu  paths:  /api/data:  post:  description: Endpoint unde NodeMCU trimite date  operationId: post  tags: [nodemcu]  requestBody:  required: true  content:  application/json:  schema:  type: object  properties:  id:  type: string  example: "E0040100019D0CA1"  name:  type: string  example: "Popa Daniel"  responses:  "200":  description: Request primit cu succes  content:  application/json:  schema:  type: object  properties:  message:  type: string  example: "Request primit!" |
| --- |

* Challenge: Daca o trimitere de date esueaza din cauza serverului, incercati din nou
* Challenge: Daca NodeMCU ramane fara conexiune la internet, salvati ultimele citiri pentru a le trimite cand revine conexiunea la internet

# **4. Aplicatie web**

Problema pe care dorim sa o rezolvam in aceasta etapa este sa stocam toate colectarile de pubele. Datele de la masina de colectare sunt trimise din NodeMCU prin internet prin protocolul HTTP. Trebuie implementat un endpoint unde NodeMCU sa trimita datele, iar pentru asta avem nevoie de o aplicatie web.

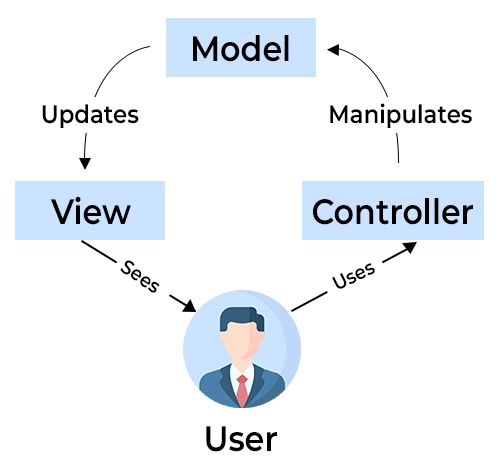


## **4.1 Aspecte teoretice**

### ASP.NET

Documentatie: <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-9.0&tabs=visual-studio>

Modelul arhitectural MVC (Model-View-Controller) separă o aplicație în trei componente principale: Model, View și Controller. Modelul MVC vă ajută să creați aplicații care sunt mai ușor de testat și mai ușor de actualizat decât aplicațiile monolitice tradiționale.



Aplicațiile bazate pe MVC conțin:

1. Models: Clase care reprezintă datele aplicației. În mod obișnuit, obiectele model recuperează și stochează starea modelului într-o bază de date.
2. Views: Views sunt componentele care afișează interfața cu utilizatorul (UI) a aplicației. În general, această interfață utilizator afișează datele modelului.
3. Controllers: Clase care:

* Gestionează cererile browserului.
* Recuperează datele modelului.
* apelează modele de vizualizare care returnează un răspuns.

### SQLite

Ca serviciu de baza de date vom folosi SQLite <https://www.sqlite.org/>. SQLite ne va ajuta sa construim o baza de date fara batai de cap care o sa serveasca perfect nevoile aplicatiei noastre.

### **Avantaje SQLite**

**Super simplu** – Nu trebuie instalat, e doar un fișier.  
**Performant pentru aplicații mici** – Nu are overhead de rețea.  
**Portabil** – Aplicația poate funcționa doar cu fișierul .db.

### **Dezavantaje SQLite**

**Nu suportă bine multi-user** – Dacă mai multe persoane scriu simultan în baza de date, devine lent.  
**Nu este scalabil** – Dacă aplicația crește, trebuie migrată pe un alt sistem.

### **Avantaje SQL Server**

**Ideal pentru aplicații mari** – Suportă multe conexiuni simultane și volume mari de date.  
**Securitate avansată** – Permite control avansat al accesului.

### **Dezavantaje SQL Server**

**Necesită instalare și configurare** – Mai greu de folosit pentru începători.  
**Resurse mai mari** – Necesită mai multă memorie și procesor decât SQLite.  
**Nu este portabil** – Dacă muți aplicația, trebuie configurat serverul SQL din nou.

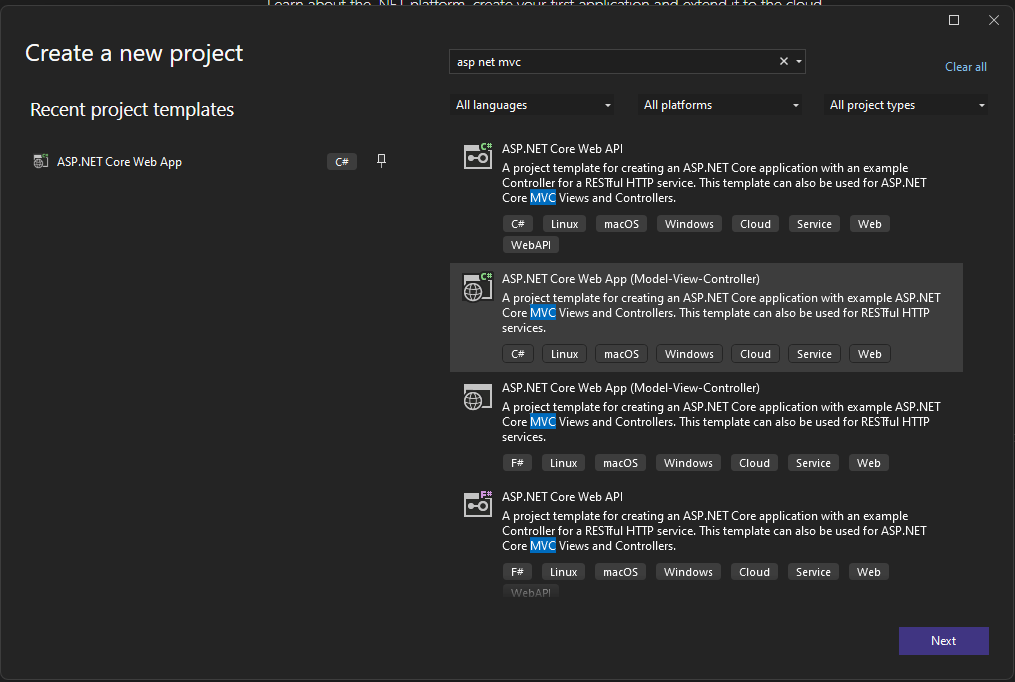
## 

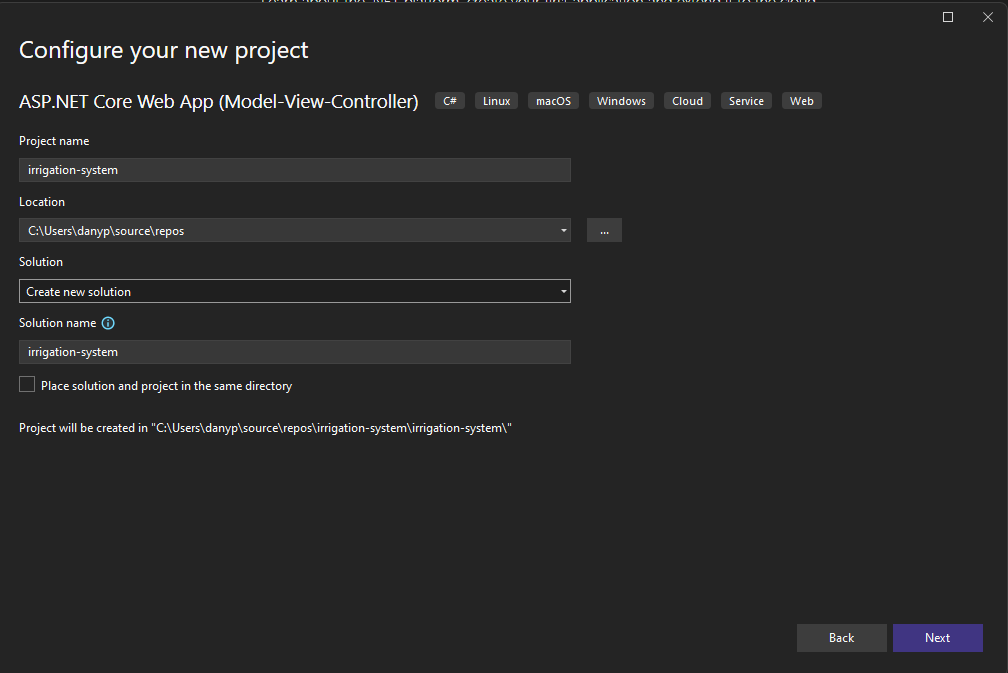
## **4.2 Notiuni practice**

Pregatim setup pentru o aplicate ASP.NET Core MVC cu .NET 9.0 (ultima versiune din 2024 Noiembrie) cu o baza de date SQLite.

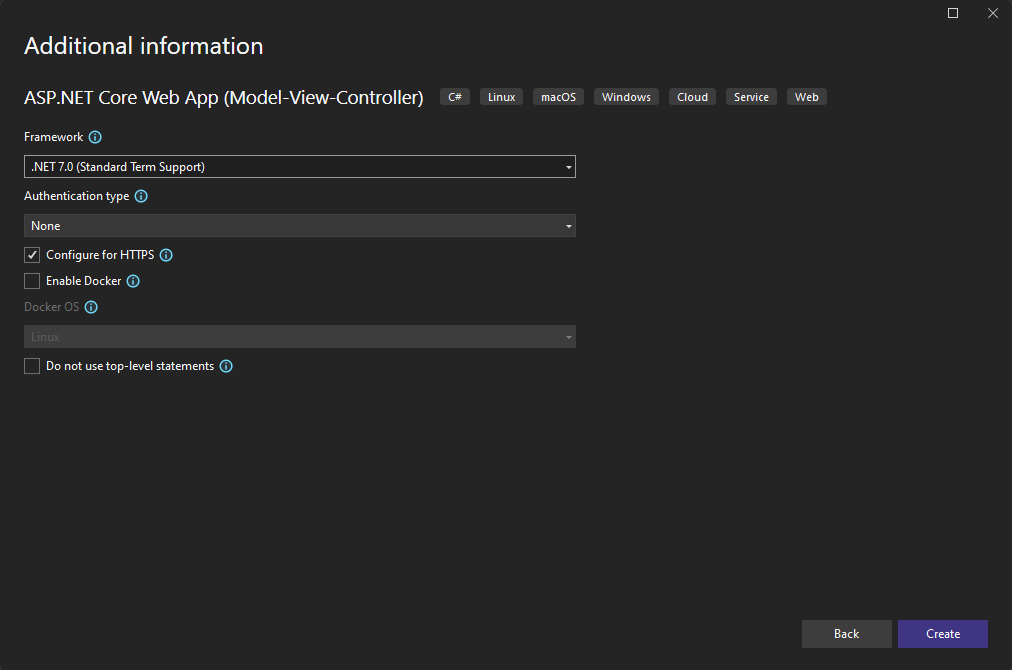
Pasi:

1. Pornim Visual Studio 2022 IDE si ne cream o aplicatie web de tip ASP.NET Core Web App (Model-View-Controller) cu **.NET 9.0**

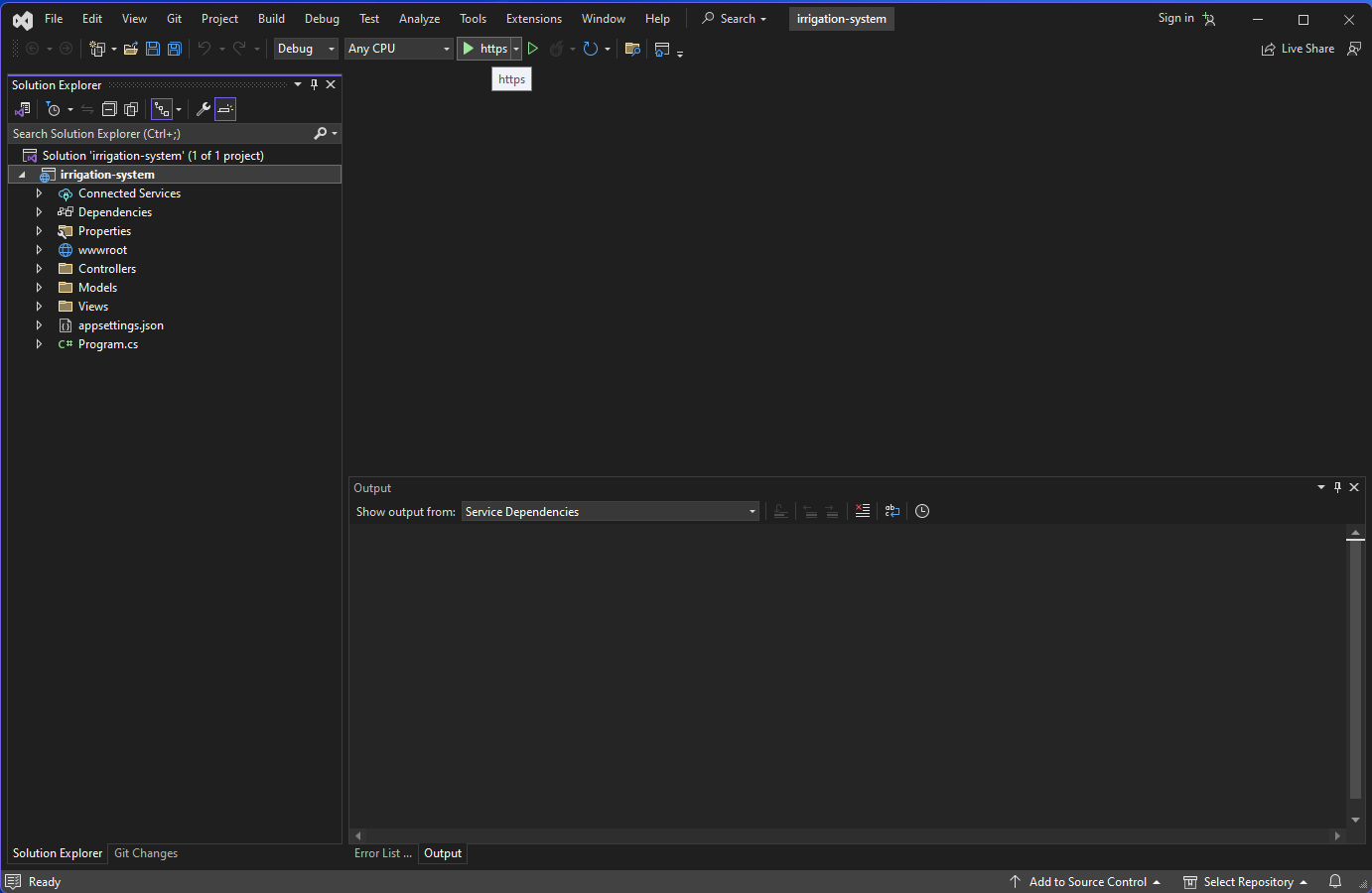




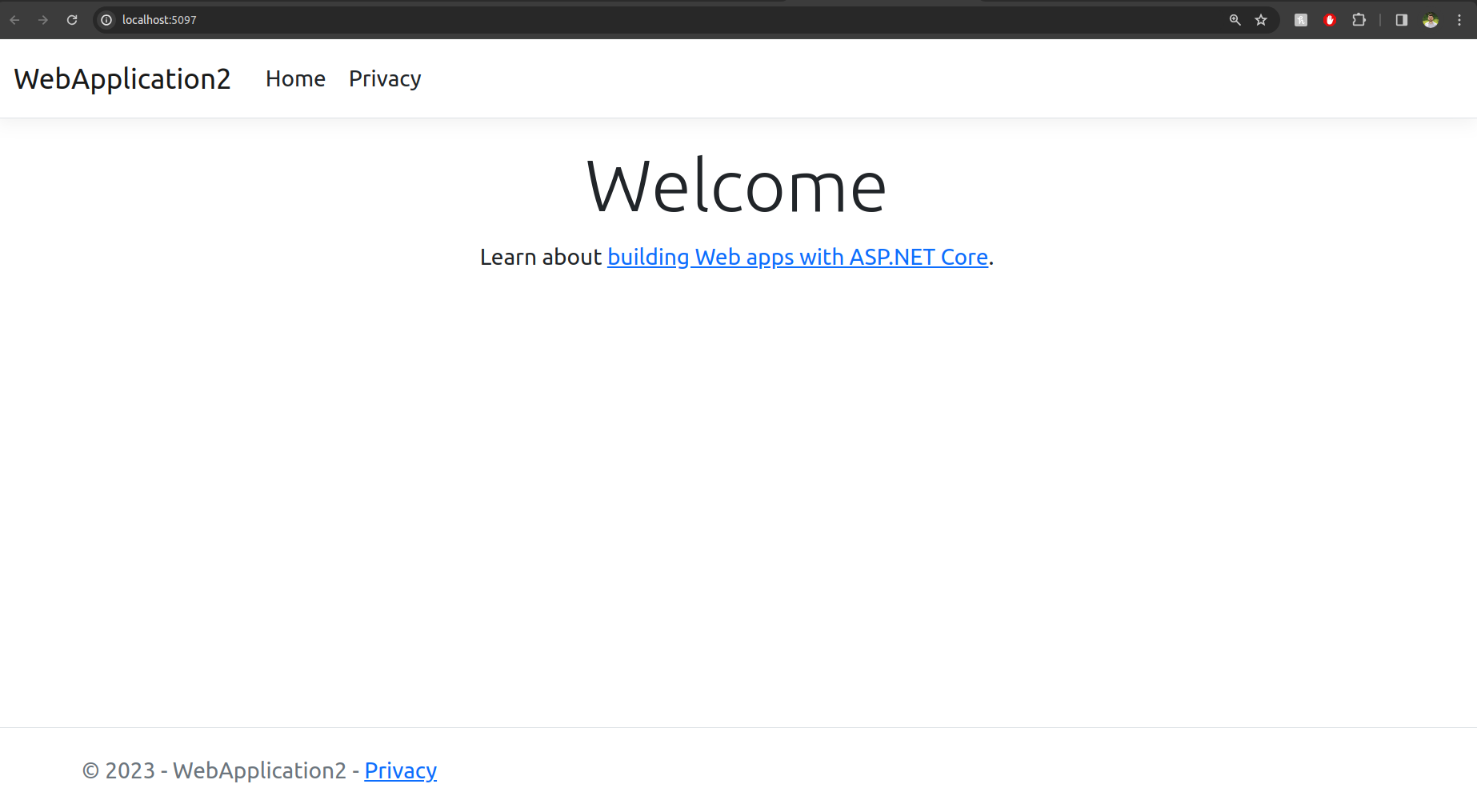
**Selectati .NET 9.0**



1. Dupa crearea aplicatiei si dupa ce IDE-ul s-a deschis, trebuie sa rulam aplicatia



1. Aplicatia web se va deschide in browser.



1. Trebuie sa instalam Entity Framework pentru a interactiona cu baza de date. Instalati urmatoarele NuGet Packages din Package Manager Console sau NuGet Package Manager:

* **Microsoft.EntityFrameworkCore.Sqlite**
* **Microsoft.EntityFrameworkCore.Tools**

| dotnet add <PROJECT> package Microsoft.EntityFrameworkCore.Sqlite |
| --- |

| dotnet add <PROJECT> package Microsoft.EntityFrameworkCore.Tools |
| --- |

Fisierul *appsettings.json* trebuie arate astfel, aici specificam locatia bazei de date:

| {  "Logging": {  "LogLevel": {  "Default": "Information",  "Microsoft.AspNetCore": "Warning"  }  },  "AllowedHosts": "\*",  "ConnectionStrings": {  "DefaultConnection": "Data Source=Data/development.db"  }  } |
| --- |

Creati directorul *Data,* iar in acesta creati fisierul *AppDbContext.cs*:

| using Microsoft.EntityFrameworkCore;  namespace YourNamespace.Data  {  public class AppDbContext : DbContext  {  public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) { }  }  } |
| --- |

Adaugati pe prima linie a fisierului *Program.cs*:

| using Microsoft.EntityFrameworkCore; |
| --- |

Tot in fisierul *Program.cs,* adaugati dupa var builder = WebApplication.CreateBuilder(args);;:

| builder.Services.AddDbContext<AppDbContext>(options =>  options.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"))); |
| --- |

Asigurati-va ca nu aveti erori in fisierul *Program.cs*, cel mai probabil trebuie importat fisierul *AppDbContext.cs*.

### Cerinte

La sfarsitul laboratorului ne dorim sa avem un API endpoint unde NodeMCU trebuie sa trimita datele fiecarei colectari. Ca si indrumare, va las urmatoarele sfaturi:

1. Creati un fisier .gitignore pentru o aplicatie ASP NET si incarcati fisierele pe Github.
2. Creati un model care va reprezenta structura datelor trimise de NodeMCU.
3. Creati baza de date SQLite care contine tabela unde veti stoca datele ([aveti nevoie de migrari](https://learn.microsoft.com/en-us/ef/core/managing-schemas/migrations/?tabs=vs#create-your-first-migration)). Atentie, orice modificare a modelului trebuie acoperita de modificarea bazei de date, de exemplu daca adaugati un atribut nou in model dupa ce ati rulat migrarea intiala, atributul trebuie sa fie legat de o coloana in baza de date.
4. Creati un API Endpoint care va primi datele de la NodeMCU ca input si le va incarca in baza de date.
5. Folositi un client http, [Postman](https://www.postman.com/downloads/) sau [Insomnia](https://insomnia.rest/download) sau [curl](https://curl.se/docs/tutorial.html), pentru a testa daca endpoint-ul functioneaza corect.
6. Porniti NodeMCU si verificati daca datele ajung in baza de date.
7. Challenge: Scrieti un test care verifica functionalitatea endpoint-ului creat.

Pentru a vizualiza baza de date creata, instalati **SQLite and SQL Server Compact Toolbox** din **Extensions -> Manage Extensions.** Ca sa se instaleze, trebuie sa inchideti Visual Studio si sa il reporniti.

# 

# **5. Dashboard Cetateni & Pubele**

### 

### Cerinte

La sfarsitul laboratorului dorim sa avem un Pull Request cu urmatoarele functionalitati:

1. Ca admin, doresc sa pot adauga cetateni cu urmatoarele informatii: nume, prenume, email, CNP.
2. Ca admin, doresc sa atribui pubele cetatenilor. Un cetatean poate avea mai multe pubele la adrese diferite. O pubela poate fi a unui singur cetatean.
3. Ca admin, doresc sa vad toate colectarile de pubele ale unui cetatean.
4. Challenge: Implementati un mecanism care sa alerteze adminii daca o pubela este colectata de la alta adresa fata de cea inscrisa in contract.
5. Challenge: Securizati aplicatia si permiteti acces-ul doar utilizatorilor admini.

# 

# **6. Optimizare Rute**

**⚠️⚠️⚠️**

**Disclaimer:** Pentru realizarea urmatoarelor cerinte, vom folosi API-uri oferite de Google Maps si Mapbox care ofera free plan pana intr-un anumit numar de requesturi. Request-urile facute peste limita pot aduce costuri aditionale. Fiti foarte atenti si codati optim.

⚠️⚠️⚠️

Masinile de gunoi SB42ULB, au parcurs traseele de mai jos in data de 15.10.2024. Importati datele dintr-un traseu la alerege in baza voastra de date. Toate functionalitatile din acest capitol se vor baza set-ul de date ales.

[Traseu 1 - SB42ULB](https://docs.google.com/spreadsheets/d/1YovN1Ux6ojzx6YvjOurqicTX328NV_va48l_xxpz6k0/edit?usp=sharing)

[Traseu 2 - SB77ULB](https://docs.google.com/spreadsheets/d/130hHcjw1X8hRvfLTpsd1BS2EkiacANh4KK6LT6e_0N4/edit?usp=sharing)

[Traseu 3 - SB99ULB](https://docs.google.com/spreadsheets/d/1TNGMzxn3foXckHT4uc06xm36oeRH7L-wVTxmirNydPw/edit?usp=sharing)

Locatia de plecare (garaj): Strada Șelimbărului 90, Cisnadie, Romania

Locatia de oprire (groapa gunoi): TRACON SRL DEDMI - Cristian, Sibiu, Romania

### Cerinte

La sfarsitul laboratorului dorim sa avem un Pull Request cu urmatoarele functionalitati:

1. Ca admin, doresc sa vizualizez harta cu colectarile din ziua 2024-10-15.
   1. Sugerez sa folositi [Leaflet JS](https://leafletjs.com/), Google Maps API este contra cost dupa ce depasiti limita de request-uri gratis [Overview | Google Maps API](https://developers.google.com/maps/documentation/javascript/overview) si are o limita destul de mica.
   2. Pentru acuratete maxima, folositi coordonate pentru a adauga puncte pe harta. Pentru a afla coordonatele pentru locatile pubelelor, sugerez a folosi [Google Geocoding API](https://developers.google.com/maps/documentation/geocoding). Creati un cont la [console.cloud.google.com](https://console.cloud.google.com/welcome/new?inv=1&invt=AbtA3w&project=black-copilot-454818-t0) cu free tier plan.
2. Ca admin, doresc ca ruta din ziua 2024-10-15 sa fie optimizata (UI la alegere). Ca sa optimizam ruta, ne lovim de problema Travel Salesman Problem.
   1. Avem nevoie de distantele dintre locatii. Distantele le putem lua prin [Mapbox Matrix API](https://docs.mapbox.com/api/navigation/matrix/?utm_medium=pricing#response-retrieve-a-matrix). Creati un cont la [account.mapbox.com/auth/signup](https://account.mapbox.com/auth/signup/) cu free tier plan.
   2. Salvati raspunsurile de la API, pentru ca Mapbox are o limita de 100.000 de request-uri. Daca nu le salvati, riscati sa treceti peste limita. ⚠️Monitorizati in permanenta sa nu treceti peste limita pentru ca veti fi taxati.
3. Ca admin, doresc sa vizualizez harta cu ruta optimizata.
4. Ca admin, doresc sa vizualizez care este diferenta de distanta parcursa dintre traseul parcurs de masina si noul traseu optimizat.
5. Ca admin, doresc sa vizualizez care este timpul de parcurgere al noului traseu optimizat, presupunand ca o colectare dureaza 1min.

# 

# **7. Optimizare Cost Clienti**