SPRC 2019-2020 - Tema 3 - Platformă IoT folosind Microservicii

Responsabil: Dorinel Filip

Titular curs: Florin Pop

1 Obiectivele temei

În cadrul temei veți dezvolta o platformă pentru colectarea, stocarea și vizualizarea de date numerice provenite de dispozitive IoT. Prin rezolvarea ei ei, vă veți familiariza cu utilizarea protocolului MQTT în contextul dezvoltării unei aplicații ce procesează (pe bază de evenimente) mesajele primite de la un astfel de broker și veți interacționa cu baze de date specializate în lucrul cu serii temporale de date.

Soluția va fi implementată folosind microservicii, plecând de la o arhitectură potrivită pentru o aplicație, ce urmează a fi rulată într-un mediu cloud, oferindu-vă oportunitatea descoperirii avantajelor unei astfel de abordări.

2 Descrierea soluției cerute

Pentru această temă ne propunem implementarea unei platforme pentru colectarea, stocarea și vizualizarea datelor numerice provenite de la un număr mare de dispozitive Internet of Things.

Pentru implementarea acesteia, vom adopta o arhitectură simplificată (dar eficientă) inspirată din modul de funcționare a celor mai cunoscute servicii cloud publice de acest tip. În mod uzual, o astfel de soluție cuprinde următoare componente:

- Dispozitive cu senzori conectate la Internet;
- Un broker de mesaje [MQTT Broker];
- O bază de date de tipul TSDB [InfluxDB];
- Un adaptor ce parsează datele din mesajele primite de la device-urile IoT și le adaugă în baza de date;
- O interfață pentru vizualizarea datelor [Grafana].

Fiecare dintre aceste componente, cu excepția dispozitivelor cu senzori (ce vor fi simulate, la momentul evaluării temei, cu ajutorul unui script dezvoltat de echipa de asistenți), va apărea sub forma unei soluții software incluse în rezolvarea temei voastre.

Brokerul de mesaje, baza de date TSDB și interfața pentru vizualizarea datelor sunt componente software complexe, dar generale și **nu** vor fi implementate ca parte a rezolvării temei.

Pentru baza de date și interfața de vizualizare a datelor se vor folosi InfluxDB, respectiv Grafana, iar pentru broker-ul de mesaje veți alege o implementare open-source pentru un broker de mesaje MQTT.

Întreaga soluție va fi livrată sub forma unui **stack Docker Swarm**, în care fiecare componentă apare ca un serviciu distinct descris în cadrul fișierului .yml.

Cerințele referitoare la funcționarea fiecăreia dintre componente sunt descrise în secțiunile următoare.

3 Componentele soluției

3.1 Broker-ul de mesaje

Pentru brokerul de mesaje se va folosi o soluție Open-Source (dintre cele pe care au disponibile imagini oficiale pe hub.docker.com) ce implementează cel puțin versiunea 3.0 a protocolului MQTT[1].

Acesta este un protocol deschis de tipul publicare/abonare (în engleză publish/subscribe) foarte utilizat în mediul Internet of Things (IoT)/Machine-to-Machine (M2M) datorită modului foarte eficient în care exploatează resursele de rețea și de calcul (CPU și memorie) a dispozitivelor conectate.

Brokerul va fi poarta de comunicație către platformă a device-urilor IoT și, de obicei, este servit ca un serviciu post-autentificare (și autorizare), acceptând și livrând mesaje doar mesaje de la/către clienții autorizați, conform unei liste de acces (ACL).

Pentru această temă, serviciul broker MQTT trebuie să:

- expună portul TCP default (1883) pe toate adresele IP ale mașinii pe care rulează;
- să permită oricărui client să publice mesaje pe orice subiect (eng. topic), respectiv să se aboneze la orice subiect (inclusiv la toate, folosind caracterul de wild-card #).

Alegerea implementării Open-Source pentru acest serviciu este la libera voastră alegere, atât timp cât aceasta are o imagine oficială în Docker Hub [2].

3.2 Dispozitivele IoT și formatul datelor în payload-ul MQTT

Deși este adesea folosit în scenarii de telemetrie, protocolul MQTT nu specifică modul de formatare a datelor în payload-ul mesajelor trimise, pentru utilizarea lui fiind necesare convenții suplimentare asupra modului de formatare a subiectelor (topics) și a încărcăturii (payload-ului) mesajelor.

În cadrul protocolului MQTT, subiectele sunt șiruri de caractere fără spații, ce formează o structură ierarhică prin separarea unui nivel de altul cu ajutorul caracterului slash (/). În mod similar, pentru InfluxDB, identificatorii seriilor de timp sunt dați de șiruri de caractere separate prin puncte.

În această secțiune stabilim atât formatul subiectelor și mesajelor MQTT, cât și corespondența lor cu seriile de timp din baza de date.

Subiectele mesajelor vor respecta formatul `<locație>/<stație>`, unde atât `<locație>` cât și `<stație>` sunt șiruri de caractere ce nu includ caracterul special /. Un exemplu valid de astfel de subiect ar fi `UPB/RPi_1`.

Datele trimise de senzori vor fi incluse în payload sub formă de dicționare formatate ca string-uri de tipul JSON[3]. Aceste dicționare vor apărea câte unul într-un mesaj MQTT și vor avea un singur nivel de imbricare. Vor fi adăugate în baza de date doar valorile numerice din dicționar. Valorile de alt tip decât numeric (float sau int) vor fi ignorate.

O cheie specială în acest format este cheia `timestamp`. Aceasta va fi folosită pentru a specifica momentul realizării măsurătorilor din dicționar.

Dacă un mesaj este invalid, atunci acesta va fi ignorat.

[1] https://mqtt.org
[2] https://hub.docker.com
[3] https://www.json.org

Exemple de mesaje MQTT

Mai jos sunt prezentate două exemple de mesaje MQTT și datele ce vor trebui adăugate, drept consecință a lor, în baza de date.

Exemplu			Date adăugate în baza de date		
#	Topic	Payload	Serie de timp	Valoare	Timestamp
1	UPB/RPi_1	Listing 1.	UPB.RPi_1.BAT	99	26 noiembrie 2019, ora 3:54:20 am
			UPB.RPi_1.HUMID	40	
			UPB.RPi_1.TMP	25.3	
2	Dorinel/Zeus	Listing 2	Dorinel.Zeus.Alarm	0	Timpul curent, preluat de pe componenta adaptor.
			Dorinel.Zeus.RSSI	1500	
			Dorinel.Zeus.AQI	12	

Tabela 1: Exemple de mesaje.

```
1 {
2       "BAT": 99,
3       "HUMID": 40,
4       "PRJ": "SPRC",
5       "TMP": 25.3,
6       "status": "OK",
7       "timestamp": "2019-11-26T03:54:20+03:00"
8 }
```

Listing 1: Payload pentru exemplul 1.

```
1 {
2       "Alarm": 0,
3       "AQI": 12,
4       "RSSI": 1500
5 }
```

Listing 2: Payload pentru exemplul 2.

3.3 Adaptor pentru introducerea datelor în baza de date

Această componentă se va conecta persistent la brokerul MQTT, se va abona la toate mesajele ce sunt trimise prin acesta (topicul wildcard - `#`) și va introduce în baza de date, conform specificațiilor din Secțiunea 3.2.

Pentru această componentă, procesarea mesajelor MQTT se va face event-based (atunci când acestea ajung la broker), iar trimiterea datelor către TSDB se va face în cel mai scurt timp posibil.

3.3.1 Mesaje de logging pentru adaptor

Componenta adaptor trebuie să afișeze mesaje de logging potrivite care să permită urmărirea datelor ce sunt adăugate.

Mesajele se vor afișa (la `stdout`, `stderr` sau prin intermediul unui logger), astfel încât să fie vizibile folosind comanda `docker service logs` apelată pentru serviciul corespunzător, **dacă și numai dacă** variabila de mediu `DEBUG_DATA_FLOW` are valoarea `"true"`. Absența acestei variabile de mediu nu trebuie să afecteze funcționarea programului, însă mesajele de debug nu vor fi generate dacă aceasta lipsește.

```
2019-11-27 18:38:14 Received a message by topic [UPB/RPi_1]
2019-11-27 18:38:14 Data timestamp is: 2019-11-27 20:35:57+02:00
2019-11-27 18:38:14 UPB.RPi_1.BAT 99
2019-11-27 18:38:14 UPB.RPi_1.HUMID 40
2019-11-27 18:38:14 UPB.RPi_1.TEMP 25.3

2019-11-27 18:38:17 Received a message by topic [Dorinel/Zeus]
2019-11-27 18:38:17 Data timestamp is NOW
2019-11-27 18:38:17 Dorinel.Zeus.Alarm 0
2019-11-27 18:38:17 Dorinel.Zeus.RSSI 1500
2019-11-27 18:38:17 Dorinel.Zeus.AQI 12
```

Listing 3: Exemplu de mesaje de logging pentru componenta adaptor.

Listing-ul 3 prezintă un exemplu de format de logging. Utilizarea lui nu este o cerință, dar mesajele afișate trebuie să conțină aceleași informații.

3.4 Baza de date TSDB

Baza de date a sistemului va fi reprezentată de o instanță a InfluxDB. Aceasta va fi expusă doar în interior-ul stack-ului soluției voastre și va permite, fără autentificare, adăugarea, citirea și manipularea de serii de timp.

Aceasta trebuie configurată astfel încât:

- **Retenția datelor să fie nelimitată** (să stocheze, pentru fiecare serie de timp, o istorie oricât de lungă, fără a face automat ștergerea/agregarea datelor cu vechime mai mare);
- **Rezoluția de timp a datelor să fie maximă** (în baza de date să se stocheze timestamp-ul cu precizie minim o secundă a datelor);
- **Datele să fie stocate persistent** în unul sau mai multe volume Docker, astfel încât ștergerea și repornirea stack-ului să nu conducă la dispariția datelor.

Pentru realizarea configurațiilor precizate, se pot face modificări în fișierul `stack.yml`, se pot utiliza comportamente default ale imaginii de Docker sau se pot seta dinamic (folosind interfața CLI sau Chronograf[4]), cu condiția ca acestea să fie realizate persistent de către scriptul ce asigură rularea temei.

4 Componenta de vizualizare a datelor

Pentru vizualizarea datelor, vom folosi o instanță a ultimei versiuni a Grafana[5]. Aceasta se va conecta la baza de date și va permite vizualizarea grafică a acestora.

În cadrul temei, aceasta va expune (prin portul 80 al tuturor adreselor IP ale mașinii pe care rulează), interfața web specifică și va permite unui utilizator autentificat cu username-ul `asistent` și parola `grafanaSPRC2020` să vizualizeze și să editeze două dashboard-uri definite, conform specificațiilor din paragrafele următoare.

Datele trebuie să fie provenite din baza de date a stack-ului.

4.1 Dashboard pentru vizualizarea datelor IoT din locația UPB

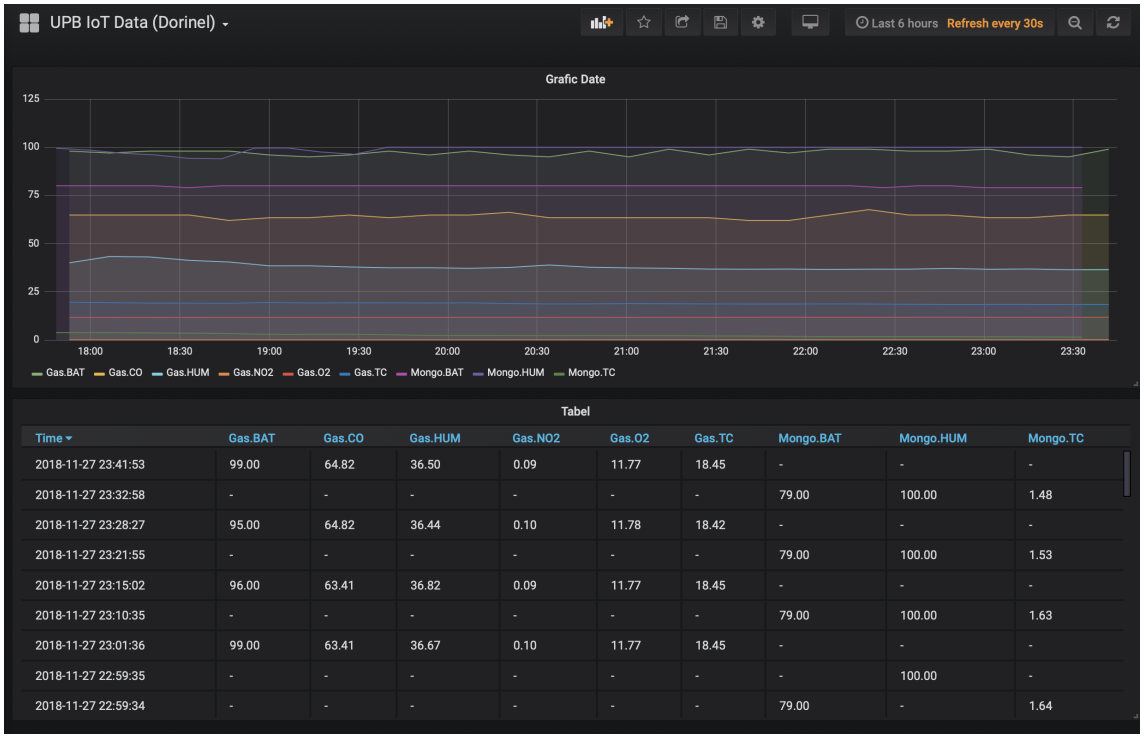
Primul dashboard prestabilit va include 2 panel-uri (un grafic și un tabel) pentru toate seriile de date, generate de toate stațiile din locația UPB. Un rând al tabelului va conține toate măsurătorile realizate la un anumit moment de timp (după aplicarea politicii de grupare). În cazul în care stațiile măsoară la momente de timp diferite, un rând poate fi incomplet.

[4] https://www.influxdata.com/time-series-platform/chronograf/
[5] https://grafana.com

Mai jos prezentăm detaliile de construcție a dashboard-ului:

- Numele dashboard-ului: UPB IoT Data;
- Formatul numelor seriilor de timp: `STAȚIE.METRICĂ` (exemplu: RPi_1.TEMP);
- Perioada de afișare: ultimele 6 ore;
- Interval de grupare a datelor: 1 secundă;
- Politica de grupare a datelor: medie aritmetică;
- Perioada de refresh automat al dashboard-ului: 30 de secunde

În Figura 1 este prezentat un exemplu de dashboard realizat conform cu specificațiile de mai sus. La momentul rulării testului, pentru locația UPB erau conectate două stații, una numită `Gas` și una numită `Mongo`.



Time	Gas.BAT	Gas.CO	Gas.HUM	Gas.NO2	Gas.O2	Gas.TC	Mongo.BAT	Mongo.HUM	Mongo.TC
2018-11-27 23:41:53	99.00	64.82	36.50	0.09	11.77	18.45	-	-	-
2018-11-27 23:32:58	-	-	-	-	-	-	79.00	100.00	1.48
2018-11-27 23:28:27	95.00	64.82	36.44	0.10	11.78	18.42	-	-	-
2018-11-27 23:21:55	-	-	-	-	-	-	79.00	100.00	1.53
2018-11-27 23:15:02	96.00	63.41	36.82	0.09	11.77	18.45	-	-	-
2018-11-27 23:10:35	-	-	-	-	-	-	79.00	100.00	1.63
2018-11-27 23:01:36	99.00	63.41	36.67	0.10	11.77	18.45	-	-	-
2018-11-27 22:59:35	-	-	-	-	-	-	-	100.00	-
2018-11-27 22:59:34	-	-	-	-	-	-	79.00	-	1.64

Figura 1: Exemplu de dashboard cu datele din UPB.

Dashboard-ul trebuie să includă automat noile metrici apărute în baza de date. Pentru realizarea acestui feature se recomandă studierea modului de lucru cu expresii regulate în interacțiunea cu InfluxDB[6].

4.2 Dashboard pentru monitorizarea nivelului bateriilor

Al doilea dashboard va cuprinde o evidență a nivelului bateriilor pentru toate device-urile ce au trimis date în platformă în ultimele 48 de ore. Acesta va cuprinde 2 paneluri (un grafic similar celui din dashboard-ul `UPB IoT Data` și un tabel cu agregări statistice).

Prin convenție, vom considera că, dacă sunt capabile de acest lucru, toate device-urile raportează nivelul bateriei ca valoare a unei chei numite `BAT` în cadrul dicționarului cu date.

[6] http://docs.grafana.org/v3.1/datasources/influxdb/

Detaliile de construcție a dashboard-ului sunt următoarele:

- Numele dashboard-ului: Battery Dashboard;
- Formatul numelor seriilor de timp: `NUME_STAȚIE` (exemplu: RPi_1;
- Interval de grupare a datelor: 1 secundă;
- Politica de grupare a datelor: medie aritmetică;
- Perioada de refresh automat al dashboard-ului: 30 de minute;
- Capăt de tabel pentru tabelul statistic:

  Serie de timp (`STAȚIE`);

  Valoare curentă (ultima disponibilă);

  Valoarea minimă, Valoarea maximă, Media valorilor.

În Figura 2 este prezentat un exemplu de dashboard realizat conform cu specificațiile de mai sus.

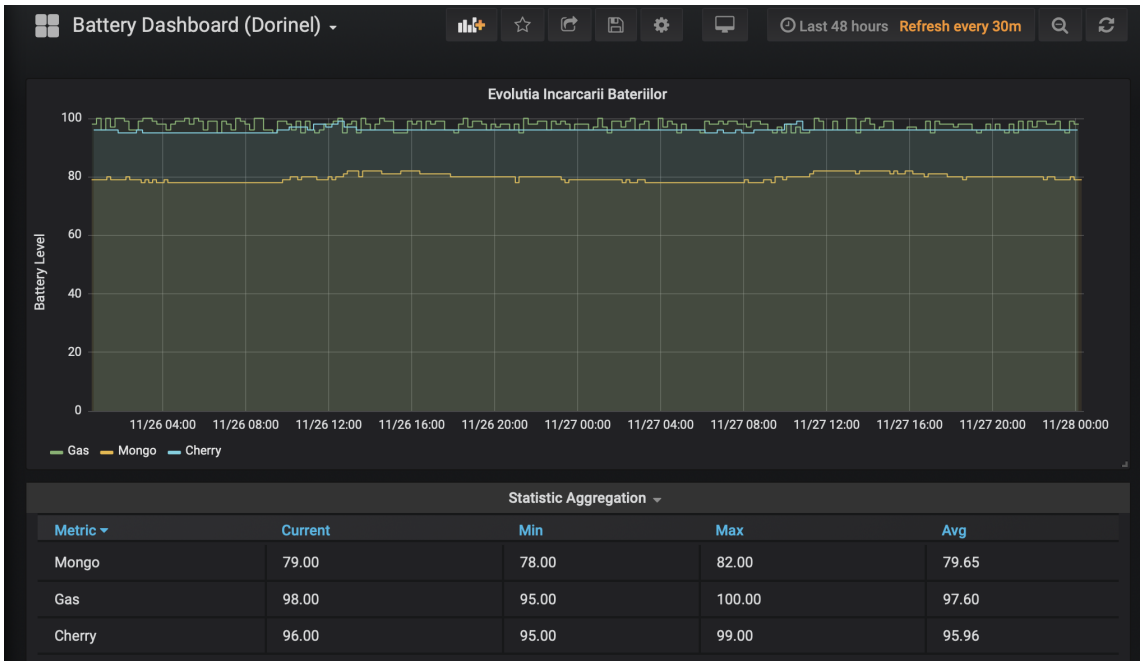


Figura 2: Exemplu de dashboard pentru monitorizarea nivelului bateriilor.

Observați că în acest dashboard apar și stațiile care nu au locația UPB. La fel ca la dashboard-ul anterior, noile device-uri apărute trebuie să fie adăugate automat în dashboard.

5 Comunicația în cadrul aplicației

Conform descrierii, întreaga aplicație va expune doar 2 porturi: portul 1883 pentru brokerul MQTT și portul 80 pentru interfața web a modulului de vizualizare (Grafana).

Toate comunicațiile backend-to-backend se vor realiza prin intermediul IP-urilor private din cadrul stack-ului Docker Swarm.

Pentru limitarea impactului diverselor atacuri informatice asupra aplicației și pentru întărirea (în engleză, hardening[7]) securității ei, se vor utiliza mai multe rețele Docker astfel încât să se limiteze comunicația container-container între containerele ce nu necesită, în mod normal, acest lucru.

[7] https://en.wikipedia.org/wiki/Hardening_(computing)

Astfel:

- Brokerul MQTT va comunica doar cu adaptorul ce adaugă informații în baza de date;
- Baza de date va comunica doar cu adaptorul și interfața de vizualizare (Grafana);
- Adaptorul va comunica doar cu broker-ul MQTT și baza de date;
- Interfața de vizualizare (Grafana) va comunica doar cu baza de date.

6 Împachetarea și prezentarea soluției

Rezolvarea temei va fi trimisă pe acs.curs.pub.ro sub forma unei arhive .zip ce va conține:

1. Un fișier `stack.yml` cu descrierea stack-ului de Docker cu soluția;
2. Fișierele sursă (însoțite de Dockerfile) pentru componentele software dezvoltate de voi;
3. Un script bash (`run.sh`) ce va pregăti și realiza prima rulare (inclusiv build-ul eventualelor imagini create de voi), folosind Docker Swarm, a aplicației;
4. Un fișier `README` în care se vor explica deciziile de implementare ale soluției și modul de lansare și utilizare a acesteia;
5. Orice alte fișiere utilizate de scriptul `run.sh` pentru pornirea inițială a aplicației.

Pentru restabilirea inițială a eventualelor configurații păstrate sub formă de fișiere, înainte de rularea scriptului va fi setată o variabilă de mediu ce va indica directorul în care sunt stocate volumele Docker cu driver local (exemplu: `export SPRC_DVP=/var/lib/docker/volumes`).

Soluția trebuie să includă în mod obligatoriu un fișier stack.yml și să poată fi rulată folosind Docker Swarm. Mai mult, după prima rulare, tema trebuie să poată fi oprită și repornită folosind comenzile `docker stack rm sprc3` respectiv `docker stack deploy -c stack.yml sprc3`.

În urma acestui proces, datele stocate de serviciul de bază de date, respectiv modificările de configurație efectuate asupra serviciului de vizualizare **NU** trebuie să se piardă.

În realizarea temei este permisă și chiar încurajată folosirea imaginilor publice oficiale[8] de pe hub.docker.com.

7 Bonus

Dacă ați rezolvat corect cerința temei, aveți un sistem funcțional, dar care nu prezintă niciun mecanism de autentificare.

Pentru obținerea unui punctaj bonus, se cere modificarea funcționării serviciului ce asigură brokerul MQTT astfel încât acesta să realizeze autentificarea dispozitivelor IoT. Crearea de conturi, respectiv oferirea sau retragerea de drepturi se va face dintr-un serviciu web dezvoltat de voi și expus la portul 8080. Interfața acestui serviciu trebuie să permită crearea unui cont și administrarea (diferențiată) a drepturilor de publicare, respectiv abonare pentru fiecare topic și pentru fiecare utilizator.

Configurația inițială brokerul trebuie să fie de așa natură încât să nu afecteze funcționarea elementelor din platformă, iar orice modificare ulterioară trebuie să fie stocată persistent.

Interfața Web de configurare va rula într-un container diferit față de brokerul de mesaje. Pentru rezolvarea acestui task este permisă adăugarea (justificată) a cel mult încă 2 containere.

8 Punctarea temelor

8.1 Observații generale

Temele vor fi evaluate sub un sistem de operare Linux din familia Ubuntu/Debian ce rulează cel puțin versiunea 18.06.1-ce a Docker.

[8] prin imagine oficială înțelegem o imagine foarte cunoscută, publicată de membrii unui proiect Open-Source cu vizibilitate.

8.2 Acordarea punctajului

O temă care nu compilează, nu poate fi rulată, nu implementează nicio funcționalitate cerută sau nu respectă modelul de implementare din enunț va fi punctată automat cu 0 puncte.

Fiecare temă va fi notată pe o scală de la 0 la 120, punctajul fiind împărțit astfel:

- 20 de puncte - integrarea în stack a soluțiilor open-source[9];
- 40 de puncte - dezvoltarea adaptorului și integrarea componentelor având ca rezultat stocarea valorilor în baza de date[10];
- 20 de puncte - configurarea interfeței de vizualizare având ca rezultat cel puțin un dashboard funcțional:
- 10 puncte - realizarea corectă a dashboard-urilor default;
- 10 puncte separarea corectă a traficului între containere;
- 20 de puncte - rezolvarea task-ului bonus.

8.3 Verificarea anti-plagiat

Toate temele vor fi verificate anti-plagiat folosind atât metode automate, cât și manuale (evaluarea acestui aspect de către echipa de asistenți). Mai mult, se interzice utilizarea de imagini (chiar și de pe hub.docker.com) care aparțin altor colegi, unor companii/persoane ce le-au dezvoltat doar pentru uz propriu sau care au o vizibilitate redusă.

În cazul în care nu sunteți siguri dacă este permisă utilizarea unei imagini, vă recomandăm să transformați această problemă într-o întrebare pentru responsabilii de temă.

[9] se oferă doar dacă în opțiunile folosite se reflectă cerințele temei și serviciile sunt folosite pentru implementarea a cel puțin o funcționalitate cerută

[10] se acordă doar dacă resursele din arhiva temei dovedesc implementarea funcționalității