Componentele sistemului informatic – Etapa 2

Universitatea POLITEHNICA din București, Facultatea de Automatică și Calculatoare,
Departamentul Calculatoare

CUPRINS

1 Introducere

Proiectul are ca scop realizarea unui model de laborator pentru un sistem de tip CPSS (Cyber-Physical-Social System) pentru dezvoltare sustenabilă în domeniul furnizării de apă în mediul urban prin implicarea activă a comunității. În implementarea acestuia intervin mai multe scenarii determinate de factori ce țin de infrastructura inteligentă precum și de interacțiunea cu cetățeanul.

Fiecare componentă poate fi modelată printr-un set de cazuri de utilizare specifice, iar ansamblul de componente este considerat pentru etapa de integrare, ca un metasistem.

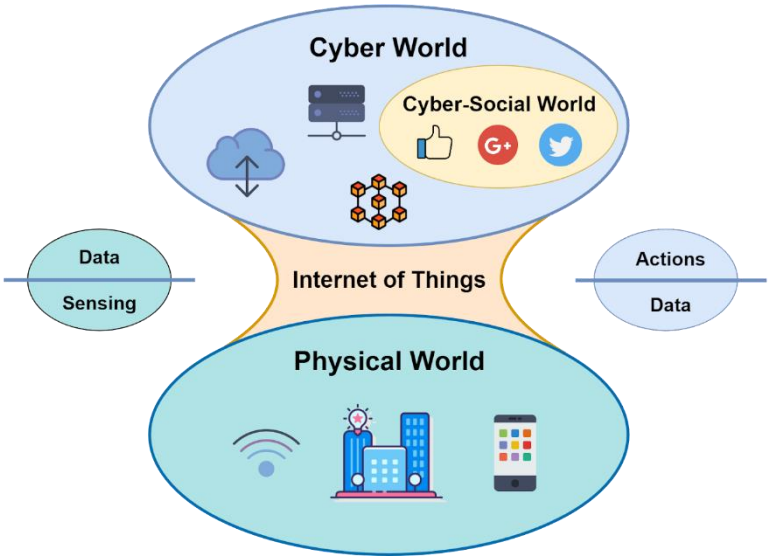


Fig. 1 Arhitectura generală de tip CPSS

Pentru a defini scenariile este important de avut în vedere contextul general stabilit în urma analizei stadiului actual și a posibilităților de dezvoltare.

Măsurile de protecție a mediului și de combatere a fenomenelor asociate încălzirii globale implică și managementul infrastructurilor de furnizare a apei potabile. Măsurile de gestionare eficientă a distribuției de apă și cele care vizează reducerea pierderilor și a consumurilor neraționale, contribuie la menținerea unui echilibru stabil între producție și consum, iar la nivel global contribuie la menținerea unui echilibru în ecosistemul unei regiuni.

Sistemele informatice destinate pentru monitorizarea și optimizarea consumurilor de apă pot contribui semnificativ la menținerea acestui echilibru și asigurarea sustenabilității pe termen mediu și lung a unei anumite regiuni. Cantitatea și calitatea apei furnizate de o infrastructură de furnizare a apei potabile sau industriale influențează calitatea vieții pentru populația unei regiuni și asigură suportul pentru dezvoltarea unor activități economice productive din domeniul agricol și industrial.

Colectarea datelor privind consumurile de apă și privind calitatea apei trebuie să se facă la o rezoluție temporală și spațială care să permită identificarea anumitor profile de comportament pentru furnizorii și consumatorii de apă. Dezvoltarea unor sisteme pilot care să identifice consumurile

de apă din interiorul unei gospodarii/apartament și monitorizarea mai frecventă în mai multe puncte de consum asigură suportul pentru identificarea mult mai rapidă a unor incidente în rețeaua de apă și permite predicția cu o precizie mai mare a consumurilor de apă.

Un management optimal al rețelei de furnizare a apei presupune, pe lângă partea de monitorizare și o parte de analiză și interpretare automată a datelor colectate în vederea detectării unor comportamente anormale (detecție de anomalii) și predicția consumurilor viitoare. Rezultatele obținute asigură suportul decizional necesar în managementul în timp-real al rețelelor de apă.

Pentru validarea și testarea metodelor propuse și implementate s-au folosit seturi de date oferite ca date de referință de către proiecte anterioare:

- set pentru analiza consumurilor de apă pe un set de apartamente/gospodării și pe o anumită perioadă de timp;
- set pentru detectarea de anomalii cu privire la calitatea apei.

Eficiența implementării unui sistem de management a unei rețele de apă depinde în egală măsură de factorii decizionali, dar și de contribuția și implicarea directă a consumatorilor. În acest scop proiectul prezintă un sistem colaborativ de colectare a informațiilor bazat pe două concepte: crowdsensing (colectare de date prin contribuția utilizatorilor) și Serious Gaming (introducerea de recompense pentru contribuțiile individuale, asemănător cu tehnicile utilizate în jocuri). În acest mod utilizatorii devin factori activi în optimizarea consumurilor de apă și în detectarea în timp util a incidentelor din rețea.

Arhitectura sistemului evidențiază componentele funcționale și intercondiționările dintre ele. Proiectarea detaliată a componentelor sistemului și a interacțiunilor dintre acestea au fost concepute sub forma unor servicii accesibile prin interfețe standardizate pentru asigurarea scalabilității și extensibilității sistemului. Detaliile de implementare a componentelor prezintă sistemele hardware și metodele software utilizate.

2 Arhitectura generală a sistemului

Arhitectură generală a sistemului este prezentată în Fig. 2.

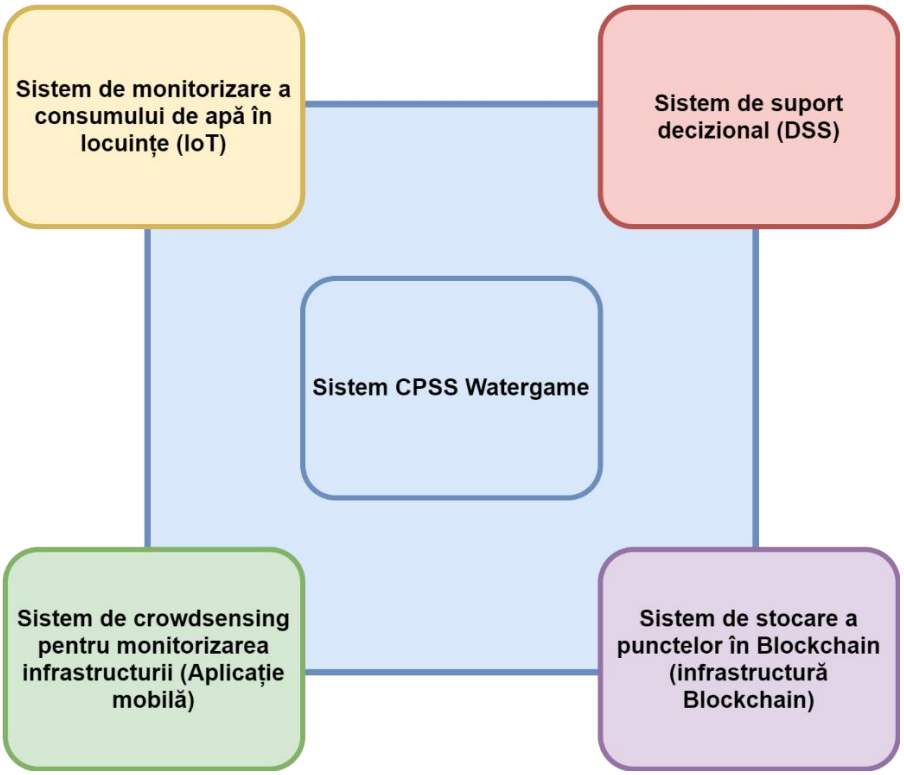


Fig. 2 Arhitectura sistemului CPSS

Sistemul este conceput ca un ansamblu de subsisteme independente, care pot interacționa în contextul unei arhitecturi decuplate. Fiind un sistem complex de tip CPSS, poate fi văzut ca un meta-sistem alcătuit din mai multe componente fundamentale, precum:

- sistemul de monitorizare a consumului de apă în locuințe prin intermediul unei arhitecturi de tip IoT;
- sistemul de suport decizional (DSS) ce oferă funcții inteligente de procesare a datelor;
- sistemul de crowdsensing pentru monitorizarea infrastructurii, prin care este implicat factorul social;
- sistemul de stocare a punctelor în Blockchain.

Integrarea acestor componente este posibilă la nivel de cazuri de utilizare, prin sistemul de puncte ce pot fi obținute din raportări sau prin eficientizarea consumului de apă în gospodărie. Sistemul de suport decizional utilizează datele colectate de la senzori pentru a oferi o imagine de ansamblu pentru furnizori și recomandări personalizate pentru consumatori. Sistemul de crowdsensing are la bază o aplicație mobilă prin care se pot raporta incidente în furnizarea apei sau în contextul infrastructurii de apă la nivel urban.

În contextul acestei etape, sunt prezentate în continuare aceste sub-sisteme din perspectiva arhitecturii și a tehnologiilor folosite, fiecare sub-sistem fiind reprezentat printr-o arhitectură și un set de tehnologii specifice.

2.1 Arhitectura componentei de monitorizare a consumului (WMS)

2.1.1 Arhitectura de colectare și procesare a datelor la UTCN (UTCN.C)

În varianta gândită de UTCN (**UTCN.C**), arhitectura sistemului de colectare și pre-procesare a datelor de consum de apă este de tip multi-nivel, cu 3 nivele (Fig. 3): Monitorizare, Edge și Cloud.

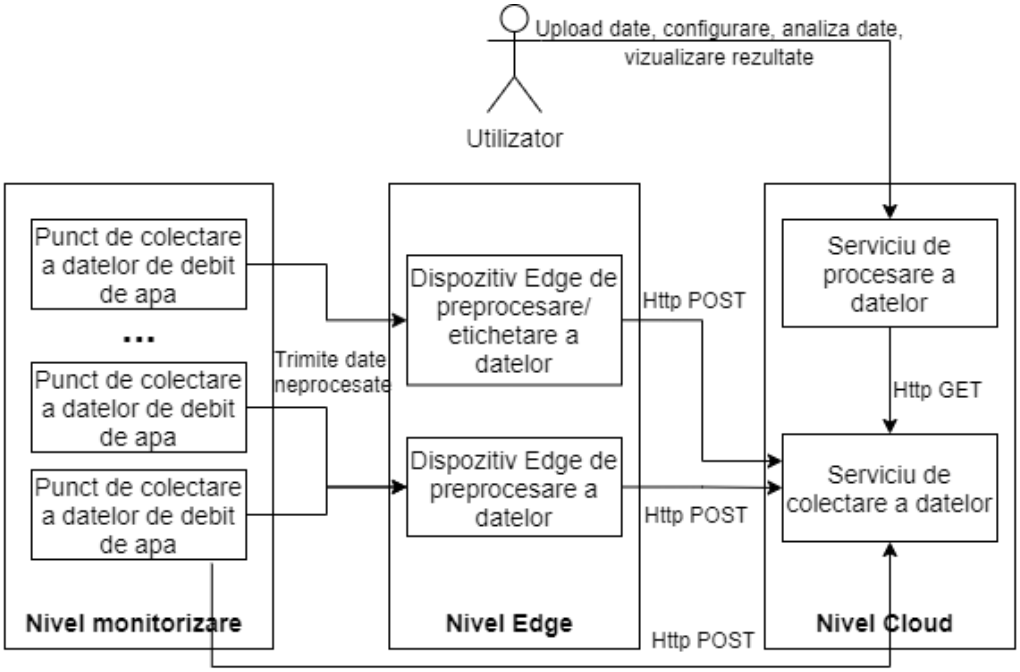


Fig. 3 Arhitectura sistemului de colectare și procesare a datelor de consum de apă (UTCN.C)

La baza arhitecturii se află **nivelul de monitorizare**, care cuprinde punctele în care se colectează datele de debit de apă instalate la locația consumatorului. La o locație pot fi instalate mai multe astfel de puncte de colectare a datelor. Datele de debit, neprocesate, se trimit periodic, în batch-uri dispozitivelor de la nivelul edge.

Nivelul edge este compus din dispozitive cu capacitate de stocare și procesare redusă, care aplică proceduri de pre-procesare a datelor primite de la punctele de colectare a datelor de debit. Datele pre-procesate sunt trimise apoi serviciului de colectare a datelor de la nivelul cloud. La nivelul edge sunt păstrate datele dintr-o fereastră de timp configurată în funcție de capacitatea de stocare a dispozitivului folosit.

La **nivelul cloud** se află un serviciu de colectare, stocare de lungă durată a datelor și un serviciu de procesare avansată (offline) a datelor. Serviciul de colectare primește date preprocesate de la dispozitivele edge și le stochează într-o bază de date în format standard OGC SensorThings[1] sau poate primi cereri prin care să servească datele stocate altor aplicații.

Serviciul de procesare a datelor poate aplica proceduri complexe de preprocesare și analiză, folosind algoritmi de învățare automată pe datele stocate de serviciul de colectare a datelor de consum sau pe seturi de date provenite din alte surse. Serviciul de procesare a datelor are atașată

[1] https://www.ogc.org/standards/sensorthings

o interfață utilizator care permite utilizatorului să încarce seturi de date, să configureze și să execute pipeline-uri de procesare a datelor și să vizualizeze rezultatele analizei datelor.

2.1.2 Arhitectura de colectare a datelor la UPB (UPB.C)

Varianta UPB pentru colectarea eficientă a datelor legate de consumul de apă (**UPB.C**) descrisă în Fig. 4 presupune că rețeaua de senzori este integrată într-o arhitectură bazată pe servicii Cloud ce permite evitarea nivelului edge din sistemul de colectare gândit de UTCN (Fig. 3).

Astfel, achiziția datelor se realizează cu ajutorul unei plăci de dezvoltare NodeMCU bazată pe microcontrolerul ESP8266 care dispune de comunicare WiFi, pini GPIO (General-purpose input / output) și poate fi programat în mediul Arduino (Parihar, 2019). O astfel de placă de dezvoltare poate fi conectată la mai multe debitmetre, monitorizând astfel debitul de apă prin mai multe conducte. Modulul de achiziție astfel proiectat, este pre-programat și dispune de o interfață de configurare ce permite conectarea la rețeaua locală și atașarea unui număr variabil de debitmetre. Interfața este expusă printr-un server găzduit direct pe microcontrolerul ESP8266 configurat în același timp în modul stație și client WiFi.

Colectarea datelor se realizează printr-un broker de mesaje MQTT, ce permite o arhitectură decuplată printr-un protocol standard de comunicație, adaptat pentru domeniul IoT (Mishra & Kertesz, 2020). MQTT este un protocol de comunicație de tip publish-subscribe ce suportă conexiune bi-direcțională peste TCP / IP, fiind ideal pentru schimbul de mesaje între dispozitive IoT (IBM, 2021). Dispozitivele IoT se pot astfel conecta la broker-ul de mesaje ce are rol de rutare a mesajelor primite de la clienți, și pot publica mesaje pe un topic sau se pot abona la un topic pentru a primi mesaje.

Server-ul aplicației se conectează la broker-ul MQTT pentru a recepționa datele trimise de senzori și realizează interfața cu baza de date MySQL[2] (Bell, 2016). Datele colectate sunt salvate periodic în baza de date, conform cu intervalul configurat la nivel de dispozitiv IoT. Există două topice pentru transmiterea datelor: date de consum instantaneu (monitorizare în timp real, cu frecvență ridicată), date de volum (înregistrate în baza de date, cu o frecvență mai scăzută). Datele pot fi accesate în timp real folosind un client MQTT (e.g. aplicație desktop, module IoT), conectarea fiind permisă pe bază de credențiale de acces la nivel de broker.

Pentru disponibilitatea datelor pentru vizualizare în timp real, este adăugat un modul REDIS[3] pentru stocarea temporară a ultimelor măsurători colectate pe topicul de consum instantaneu (Chen et al., 2016).

Datele legate de consum și starea senzorilor sunt prezentate către client cu ajutorul unei aplicații mobile (Android). Autentificarea și gestionarea conturilor pentru utilizatori se realizează prin serviciul Firebase[4], iar încărcarea datelor de consum se realizează prin cereri HTTP de la server-ul principal.

[2] https://www.mysql.com/

[3] https://redis.io/

[4] https://firebase.google.com/

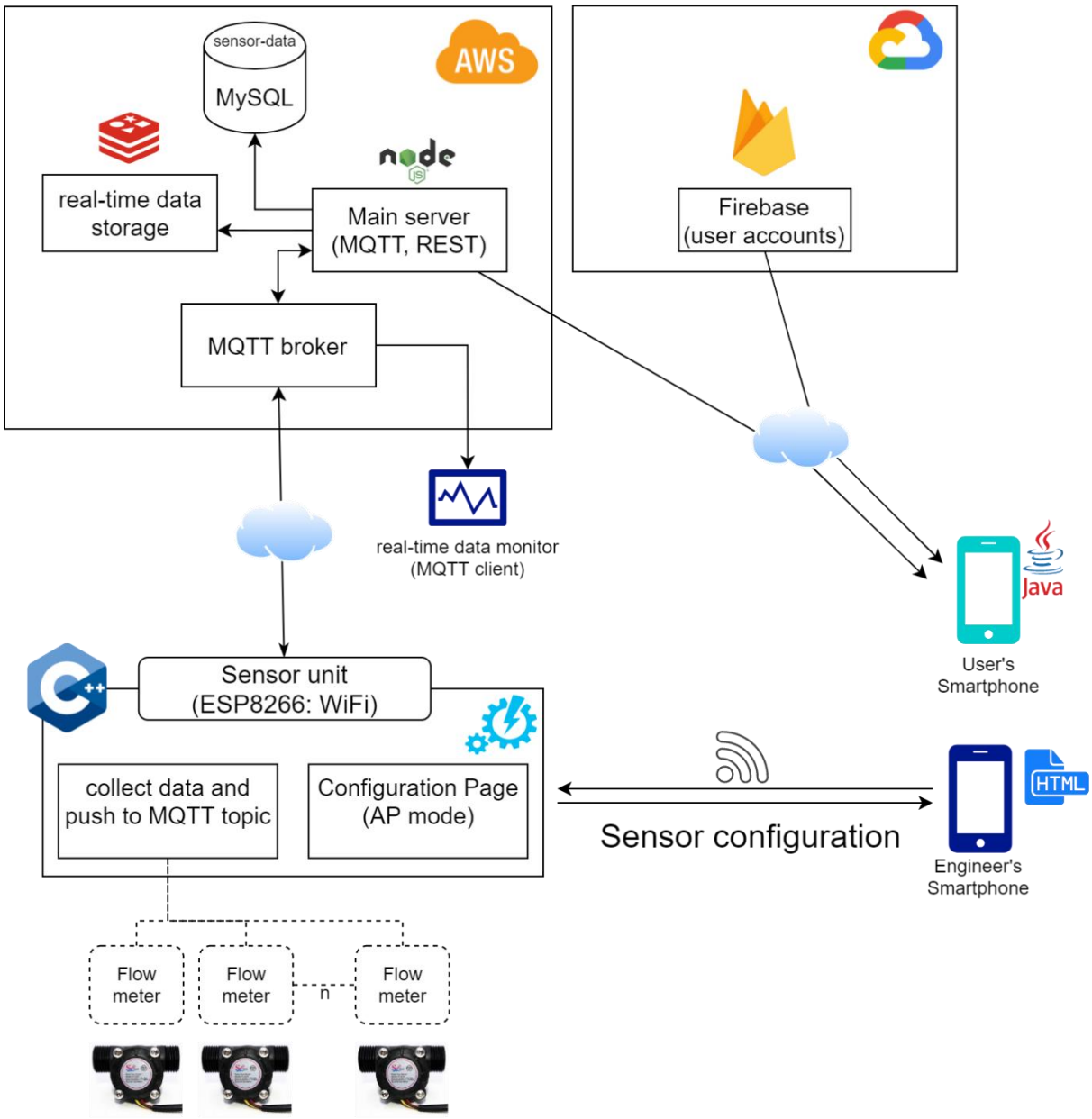


Fig. 4 Arhitectura sistemului de monitorizare a consumului de apă în locuințe (UPB.C)

2.2 Arhitectura componentei de suport decizional (DSS)

Componenta de suport decizional reprezintă o extensie a sistemului de monitorizare a consumului, realizând analiza datelor și apoi oferind rapoarte și recomandări pentru furnizori sau consumatori, în vederea eficientizării consumului de apă.

Sistemul de tip DSS integrează algoritmi de ML într-un serviciu implementat în Python care preia datele de consum de la server-ul aplicației și implementează scenariile de procesare a datelor. Aplicația mobilă prezintă recomandările individuale generate în urma procesării pentru consumator.

2.3 Arhitectura componentei de Crowdsensing (CS)

Arhitectura componentei de crowdsensing (Fig. 5) are la bază servicii Cloud și o aplicație mobilă interactivă. Pentru a asigura infrastructura tehnică necesară unei aplicații la standardele actuale din industrie, sunt considerate integrări cu servicii de aplicație precum: Firebase[5], Google Sign In, OneSignal[6], Apple Login.

Spre deosebire de celelalte componente, factorul uman este central în funcționarea aplicației pentru raportarea incidentelor pe hartă. Aplicația permite plasarea de indicatori pe hartă în mod interactiv, pe bază de geolocație și realitate mixtă, combinate cu elemente de joc pentru stimularea participării.

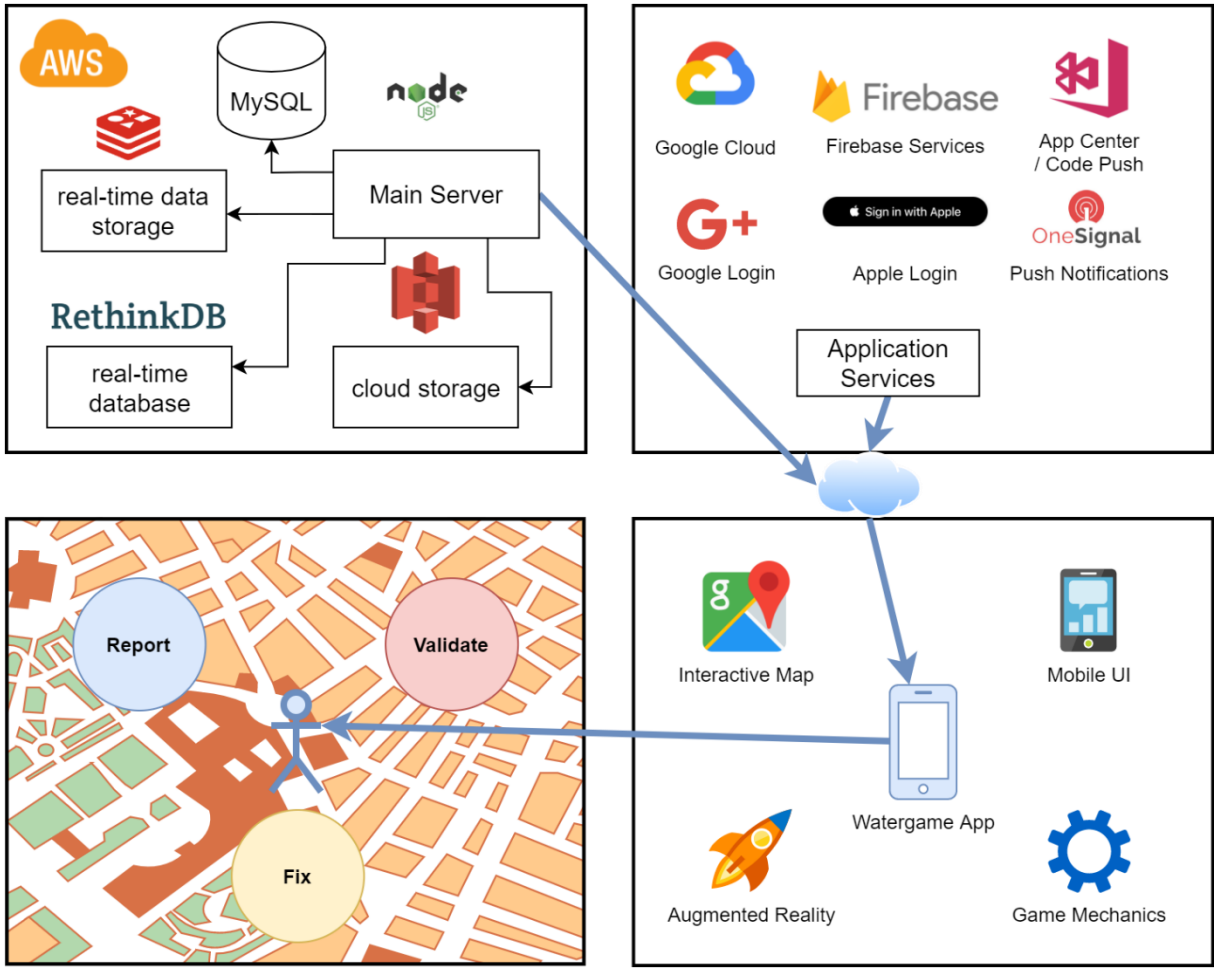


Fig. 5 Arhitectura sistemului de raportare a incidentelor pe bază de crowdsensing

[5] https://firebase.google.com/

[6] https://onesignal.com/

Pentru operabilitate în condiții reale sunt considerate metode variate de stimulare a participării precum:

- modele matematice pentru alocarea dinamică a activităților de raportare;
- gamificare și elemente de joc serios (Serious Gaming);
- aspect vizual atractiv și interfață intuitivă.

Infrastructura Cloud este găzduită pe AWS (Amazon Web Services) reprezentată printr-un server de aplicație care gestionează cererile HTTP de la aplicația mobilă precum și logica de business pentru accesarea bazei de date și a sistemelor de stocare a datelor temporare. Sistemul operează cu o bază de date MySQL ce asigură persistența datelor și o structură bine definită pentru operabilitate. Pentru scalabilitate, sistemul incorporează și o memorie temporară de tip REDIS. Datele în timp real sunt procesate folosind o bază de date de timp real RethinkDB[7] ce permite funcții avansate de interogare a datelor (Wingerath, 2017) în timp ce datele multimedia sunt stocate într-un serviciu de stocare cloud (Amazon S3).

Arhitectura jocului este reprezentată de cele 3 scenarii specifice fiecărui tip de utilizator (Finder, Fixer, Validator) și are la bază harta interactivă configurată pentru a integra elementele de joc propuse.

2.4 Arhitectura componentei Blockchain (B)

Arhitectura componentei Blockchain (Fig. 6) are la bază o rețea de test Ethereum și o aplicație web funcțională în condiții de laborator. Interacțiunea cu rețeaua blockchain se realizează prin intermediul aplicației sau prin intermediul unui wallet conectat direct (e.g. MetaMask (MetaMask, 2021)). Pentru a sincroniza wallet-urile cu conturile de utilizator din sistem, sistemul va fi conectat indirect (prin server-ul de aplicație, folosind cereri HTTP) la baza de date MySQL de pe AWS.

Rețeaua de test este pusă la dispoziție de platforma Ethereum și presupune o serie de noduri conectate între ele în mod peer-top-peer, care oferă posibilitatea introducerii datelor în blockchain, validării tranzacțiilor și interogării datelor (Neto, 2020).

Aplicația web prin intermediul căreia se va putea tranzacționa token-ul (WTG – Water Game Token) va fi instalată folosind NPM Lite Server[8]. Soluția va oferi și diverse validări pentru operațiunile ce se doresc a fi efectuate (donare de token, transfer token) prin intermediul JavaScript și web3.js. De asemenea, prin intermediul Web3.js se va realiza conexiunea la blockchain și interacțiunea cu Smart Contracts (Ethereum Revision a57dd3c5, 2016).

Pentru efectuarea diferitelor operațiuni pe blockchain (e.g. oferirea de token drept recompensă pentru utilizatori), ce vor fi declanșate din exteriorul rețelei, se va pune la dispoziție un REST (Representational state transfer) API prin intermediul căruia se vor realiza respectivele cereri.

[7] https://rethinkdb.com/

[8] https://www.npmjs.com/package/lite-server

Arhitectura soluției va oferi posibilitatea tranzacționării token-ului generat și din afara soluției, prin intermediul alternativelor precum: exchange-uri sau platforme de tranzacționare de tokeni, dacă se dorește trecerea monedei într-un mediu de producție.

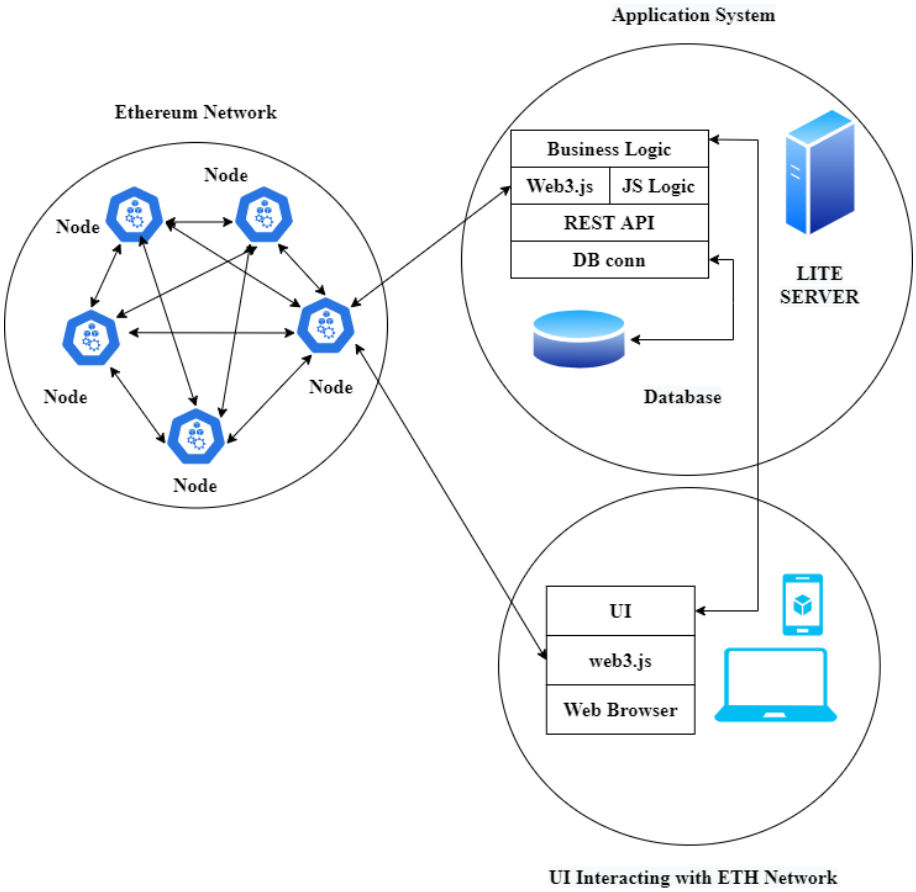


Fig. 6 Arhitectura componentei blockchain

3 Componentele sistemului

Componentele sistemului sunt proiectate separat și sunt definite prin interacțiuni specifice din perspectiva cazurilor de utilizare. Definirea interacțiunilor la nivel de sistem CPSS urmează a fi tratată în faza de integrare.

3.1 Monitorizare consum (WMS)

Componenta de monitorizare a consumului are la bază un sistem IoT de colectare a datelor de la consumatori, pe baza unor module programabile disponibile pe piață. În proiectarea modulelor de achiziție de date, am avut în vedere simplitatea soluției pentru o instalare rapidă și costuri minime.

Etapele proiectării au fost centrate pe următoarele obiective:

- Realizarea schemei electrice a modulului IoT pentru achiziția datelor;
- Proiectarea modului de conectare a modulelor IoT;
- Proiectarea arhitecturii Edge de pre-procesare a datelor;
- Proiectarea arhitecturii Cloud pentru colectarea datelor;
- Proiectarea aplicației de monitorizare și raportare pentru consumatori.

3.1.1 Componenta de monitorizare debit de apă (UTCN.C)

În nivelul de monitorizare este nevoie de unul sau mai multe debitmetre conectate la unul sau mai multe noduri de colectare, în funcție de specificul fiecărei locații de monitorizare. Specificul locației constă în faptul că pot exista unul sau mai multe puncte de măsurare, iar dacă există multiple puncte de măsurare atunci, în funcție de locația acestora (pe unul sau mai multe niveluri, de exemplu), poate fi necesară prezența mai multor noduri. Trebuie luată în considerare în faza de proiectare a sistemului de colectare a datelor dintr-o locație faptul că avem de a face cu debitmetre care se conectează cablat la noduri și din cauza aceasta există limitări de distanță, alte limitări fiind date de poziția fizică a locurilor unde pot fi montate debitmetrele și de unde pot fi accesate sursele de alimentare de tip AC sau DC pentru noduri.

În plus, de la nivelul nodurilor de colectare la care sunt conectate debitmetrele trebuie asigurată o poziționare care să permită comunicarea nodului cu punctul local de acces wireless.

Se folosește la un nivel mai ridicat puterea de procesare de la nivelul nodurilor dacă debitmetrele sunt conectate direct pentru obținerea datelor de consum în litri / minut sau litri / oră și a minimului de procesare legat de acestea.

Dacă nodurile la care sunt legate debitmetrele sunt degrevate de procesarea datelor, acestea sunt transmise direct sub forma măsurată spre un SBC (single board computer) aflat tot în cadrul rețelei de senzori, la marginea acesteia. Avantajele legate de utilizarea unui SBC țin de puterea de procesare mai ridicată, resursele de memorie mai mari și de posibilitatea de stocare locală a unei cantități considerabile de date pentru crearea și menținerea unui istoric de consum.

Partea de achiziție a datelor prin intermediul nodurilor la care sunt conectate debitmetrele rămâne neschimbată față de cazul anterior, iar aici la marginea rețelei se utilizează un dispozitiv embedded care utilizează un SoC (System on Chip) și integrează în componenta centrală de procesare o arhitectură de tip ARM.

Un astfel de sistem are cerințe de consum energetic foarte mici și este capabil să accelereze aplicațiile de ML.

3.1.2 Proiectarea modulelor de achiziție (UPB.C)

Schema electrică a modulului de achiziție a datelor este prezentată în Fig. 7. Conectarea debitmetrelor YF-S201[9] se realizează prin:

- cablul de date (galben), conectat la un pin digital de pe modulul NodeMCU;
- cablul de alimentare (roșu), conectat la pinul Vin de pe modulul NodeMCU;
- cablul de masă (negru), conectat la pinul GND de pe modulul NodeMCU.

Dintre acestea, cablurile de alimentare (și masă) sunt unite pe un singur pin la intrarea modulului, fapt ce necesită lipirea manuală a conectorilor terminali și presupune o mare parte din efortul necesar pentru instalare.

Modulele NodeMCU sunt alimentate prin cablu USB de la un alimentator de rețea. Deoarece acestea sunt proiectate să transmită date cu o frecvență ridicată, nu este posibilă funcționarea pe baterie. Au fost achiziționate cabluri USB lungi de 5 m pentru a permite alimentarea modulelor în locuințe, unde prizele de alimentare pot fi la distanțe mai mari de locul instalării senzorilor.

Procesul de instalare constă în pre-programarea modulelor și configurarea acestora la locul instalării, pentru a se conecta la rețeaua locală de WiFi și a transmite conform cu senzorii conectați la pinii digitali.

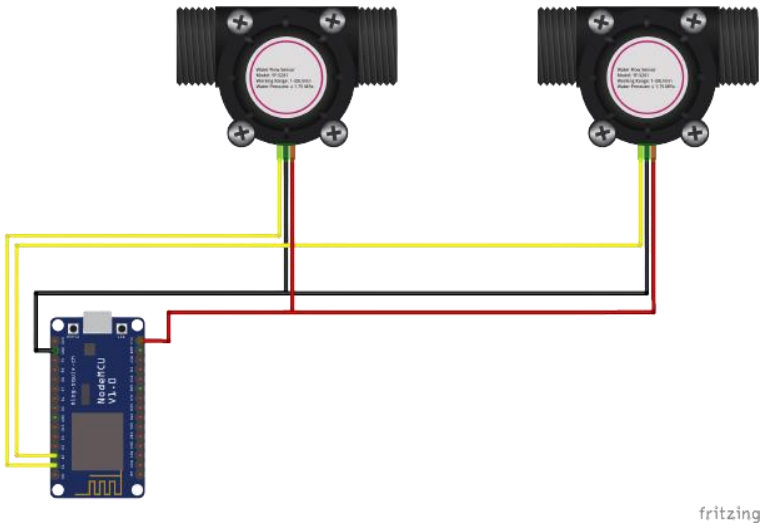


Fig. 7 Schema electrică a modulului de achiziție a datelor

[9] https://www.optimusdigital.ro/ro/senzori-senzori-hall/3078-debitmetru-yf-s201-1-30-l-min.html

3.1.3 Componenta Edge de preprocesare a datelor (UTCN.C)

Pe nivelul edge, datele trec printr-o etapă de screening prin care sunt eliminate toate valorile invalide apărute din cauza calibrării debitmetrelor, a degradării capacității de măsurare cu acuratețe sau datorate întreruperilor în alimentarea cu energie electrică a acestora.

După aceasta se trece la etapa de filtrare a citirilor debitmetrelor de tip outlier cauzate de factori incontrolabili, cum sunt impuritățile de la nivelul fluxurilor de apă măsurate, care influențează funcționarea componentelor în rotație din cadrul debitmetrelor.

Pentru debitmetrele comerciale disponibile pentru măsurarea consumului rezidențial/casnic, acestea sunt capabile sa înregistreze valori ale debitului în intervalul de 1 litru – 40 litri / minut. Pentru eventualele valori pe care debitmetrele le transmit în afara acestui interval, nodurile de tip NodeMCU efectuează filtrarea valorilor considerate în afara intervalului de referință.

De asemenea, în ceea ce privește eșantioanele lipsă în cadrul fluxului de măsurare pe un debitmetru la nivelului nodului se aplică funcția de aproximare a valorilor lipsă din cadrul secvenței de valori măsurate. Pentru a realiza acest lucru este posibilă utilizarea unei funcții de interpolare care realizează o medie aritmetică a *n* valori ale eșantioanelor primite înainte de eșantionul lipsă și a *n* valori primite după eșantionul lipsă, unde *n* este un factor configurabil la nivelul fiecărui nod în parte în funcție și de valorile obținute pentru debit în cadrul etapei de calibrare.

Tot la nivelul unui nod are loc etapa de agregare a tuturor eșantioanelor primite de la debitmetrele conectate local la un nod. În cazul punctelor de consum apropiate fizic cum sunt cele din încăperile de mici dimensiuni se realizează o agregare a tuturor valorilor de consum și se realizează o singură raportare către componenta aflată la marginea edge-ului pentru o transmisie mai departe în cloud.

Tot la nivelul acestei componente se realizează în funcție de specificul aplicației și etapa de reducere a valorilor redundante.

3.1.4 Serviciul de colectare a datelor (UTCN.C)

Serviciul de colectare a datelor are două componente (Fig. 8):

- API de colectare / regăsire a datelor;
- Baza de date.

API-ul poate fi accesat de dispozitivele de monitorizare / edge și de alte aplicații care procesează datele prin cereri HTTP de tip GET și POST. Cererile trimise înspre API și răspunsurile acestuia folosesc date împachetate în format JSON.

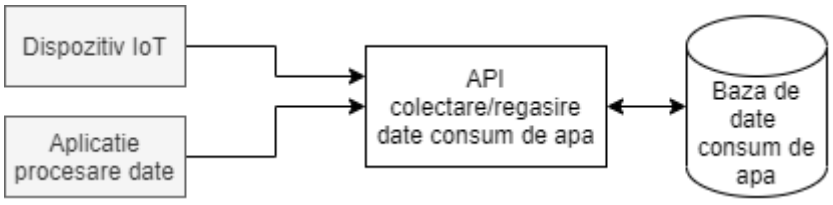


Fig. 8 Serviciul de colectare a datelor

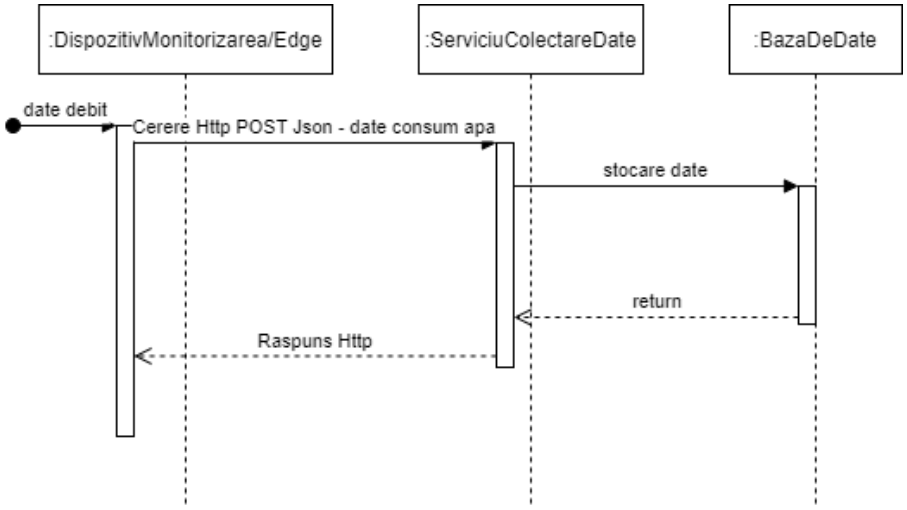


Fig. 9 Interacțiunea între un dispozitiv de monitorizare/edge și serviciul de colectare a datelor

Interacțiunea între dispozitivele de monitorizare / edge și serviciul de colectare a datelor este descrisă în Fig. 9. Dispozitivele trimit periodic datele încapsulate într-o cerere HTTP de tip POST, în format JSON. Datele preluate în acest fel sunt stocate în baza de date.

Datele colectate pot fi accesate de aplicații client, cum ar fi, de exemplu, una care face procesarea / analiza datelor. Aplicația de procesare trimite o cerere HTTP de tip GET, după care primește datele ca răspuns, în format standard SensorThings. Interacțiunea între aplicația de procesare a datelor și serviciul de colectare a datelor este descrisă în Fig. 10.

Modelul folosit pentru preluarea, stocarea și servirea datelor este cel specificat în standardul OGS SensorThings și descris în Fig. 11. Datele transmise între clienți și API sunt structurate ca *datastreams*. Datele transmise vor conține atâtea datastreams câte puncte de măsurare sunt instalate la o locație. Dacă la o locație există un singur punct de măsurare al consumului de apă, datele vor fi structurate într-un datastream, ca în exemplul următor.

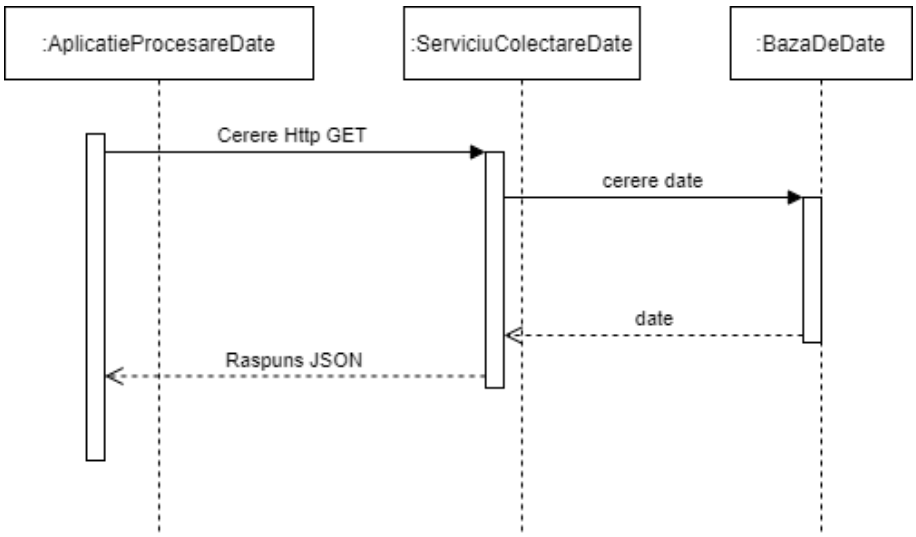


Fig. 10 Interacțiunea între o aplicație care face procesarea datelor și serviciul de colectare a datelor

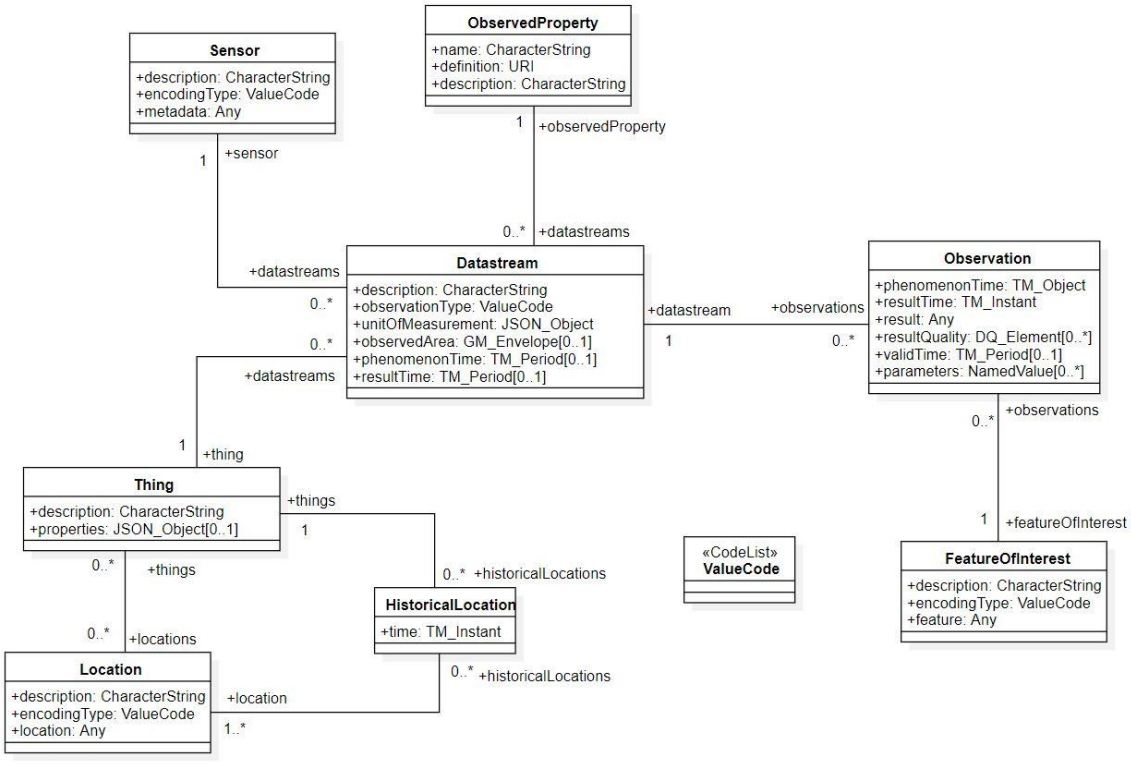


Fig. 11 Modelul de date OGC SensorThings

ExempluConsum.JSON

```
{
	"name": "Lab-test-debitmeter",
	"description": "lab-test",
	"properties": {
		"style": "lab"

	},
	"Locations": [
		{
			"name": "lab-C04",
			"description": "lab baritiu 28",
			"encodingType": "application/vnd.geo+json",
			"location": {
				"type": "Point",
				"coordinates": [23.586384084911145, 46.77302048938485]
			}
		}
	],
```

```
	"Datastreams": [
		{
			"name": "test-debitmeter",
			"description": "consum apa rece",
			"observationType": "http://www.opengis.net/def/observationType/OGC-OM/2.0/OM_Measurement",
			"unitOfMeasurement": {
				"name": "litre",
				"symbol": "L",
				"definition": "http://www.qudt.org/qudt/owl/1.0.0/unit/Instances.html#Liter"
			},
			"Sensor": {
				"name": "YF-S201",
				"description": "Debit meter-water",
				"encodingType": "application/html",
				"metadata": "https://www.optimusdigital.ro/ro/senzori-senzori-hall/3078-debitmetru-yf-s201-1-30-l-min.html"
			},
			"ObservedProperty": {
				"name": " water consumption ",
				"definition": "http://www.qudt.org/qudt/owl/1.0.0/quantity/Instances.html#Volume",
				"description": "consum de apa"
			},
			"Observations": [
				{
					"phenomenonTime": "2021-09-14T10:04:00Z",
					"result": 2
				},
				{
					"phenomenonTime": "2021-09-14T10:05:00Z",
					"result": 2.5
				}
			]
		}
	]
}
```

3.1.5 Aplicația mobilă (UPB.WMS)

Aplicația mobilă permite furnizorilor și consumatorilor să vizualizeze informații despre consumul de apă din locuință. Pentru furnizor, aplicația permite monitorizarea senzorilor instalați, adăugarea unui nou senzor și asocierea cu un consumator pe baza unui cod de client, debranșarea, precum și vizualizarea notificărilor transmise de consumatori cu privire la problemele ce pot apărea în furnizarea apei.

Consumatorul beneficiază de un set de funcționalități mai restrânse pentru monitorizarea senzorilor asociați, precum și actualizarea datelor personale și transmiterea notificărilor pentru a semnala anumite probleme.

Aplicația se conectează la server-ul aplicației pentru sincronizarea datelor de la senzorii instalați, apoi se abonează la brokerul MQTT pentru actualizarea și vizualizarea acestora în timp real.

Datele de identificare ale utilizatorilor precum și funcționalitățile ce presupun autentificarea și vizualizarea datelor de consum sunt gestionate de serviciul Firebase, separat de server-ul aplicației. Firebase Realtime Database (Google, 2021a) este o bază de date ne-relațională, având o structură arborescentă de tip JSON (JavaScript Object Notation) și o interfață de acces programat cu suport optimizat pentru tehnologii mobile (Moroney, 2017).

Date de consum

Pentru vizualizarea datelor istorice, se realizează o cerere HTTP iar server-ul aplicației răspunde cu datele înregistrate în baza de date MySQL folosind un set de filtre (e.g. ID-ul modulului, canalul de măsurare, numărul de înregistrări returnate sau intervalul de timp solicitat). În mod normal datele sunt actualizate în baza de date la intervale egale de timp, ceea ce permite afișarea lor pe grafic, fără o prelucrare intermediară.

Notificări

Prin funcția de notificare, furnizorul poate primi de la consumatori notificări despre problemele raportate de aceștia. Acestea pot fi:

- reclamații;
- solicitări intervenții;
- instalare de senzori.

Prin reclamații, consumatorul poate semnaliza anumite probleme în furnizarea apei precum: oprirea alimentării cu apă, probleme de facturare, branșamente ilegale, etc.

Prin solicitări de intervenție, consumatorul poate solicita intervenția unei echipe autorizate pentru operații de mentenanță sau reparații în sistemul de distribuție.

Notificarea de tip instalare de senzori este specifică aplicației de monitorizare a consumului, prin care un nou client poate solicita configurarea modulelor instalate, fiind transmise date de identificare ale solicitantului în vederea actualizării datelor din aplicație.

3.2 Suport decizional (DSS)

În contextul DSS propus, scenariile de evaluare sunt independente, iar metodele de analiză propuse pot fi utilizate la cerere de către părțile interesate. Având în vedere rolurile multiple și părțile interesate într-o rețea de distribuție a apei, soluția propusă poate oferi suport decizional și recomandări după cum urmează:

- Atât consumatorii, cât și furnizorii și operatorii de rețea pot primi alerte în momentul în care se detectează anomalii în datele culese de serviciul de colectare a datelor, alerte pe baza cărora să se poată lua diverse decizii (de ex. înlocuirea unor echipamente sau investigarea integrității implementării).
- Furnizorii pot primii estimări ale consumului ce urmează a fi generat de diverși consumatori în parte și să ia decizii pe baza acestor estimări (de ex. debranșarea în cazul în care datoriile ar depăși anumite limite maxime).
- Furnizorii și operatorii de rețea pot beneficia de o mai bună înțelegere a comportamentului consumatorului în scopul dezvoltării de soluții stimulative și centrate pe consumatori pentru optimizarea resurselor de apă.
- Furnizorii și operatorii de rețea pot beneficia de suport în procesul de luare a deciziilor, analizând datele extrase și profilurile consumatorilor.
- Consumatorii pot beneficia de monitorizarea propriului consum, învățând să se autoeduce și să utilizeze în mod responsabil resurselor de apă.
- Consumatorii pot beneficia de recomandări specifice în ceea ce privește zona lor geografică, prin comparație cu vecinii lor.
- Furnizorii pot beneficia de recomandări specifice cu privire la întreaga rețea, care arată o comparație cu clustere similare de consumatori din orice zonă geografică.
- Furnizorii și operatorii de rețea pot beneficia de recomandări generale pentru zonele geografice, subliniind efectul general asupra rețelei de distribuție a apei.

În plus, DSS poate oferi recomandări bazate pe nivelurile de deviație standard ale comportamentului utilizatorilor după cum urmează:

- recomandări generale pentru zonele geografice;
- liniile directoare specifice pentru grupurile de consumatori cu privire la întreaga rețea;
- recomandări specifice pentru grupurile de consumatori în ceea ce privește zona geografică a acestora.

3.2.1 Serviciile de detecție a anomaliilor (UTCN.A1) și de predicție (UTCN.A2)

Serviciile de detecție a anomaliilor și de predicție (UTCN.A1 și UTCN.A2) sunt oferite prin intermediul platformei numite AutomaticAI (Czako, Sebestyen & Hangan, 2021). Dezvoltarea acestei platforme a fost realizată în cadrul unui proiect anterior și cuprinde toate metodele și algoritmii necesari pentru a configura și regla un algoritm de detectare a anomaliilor de la zero. Platforma conține algoritmi de preprocesare, cum ar fi diferite tipuri de transformare a datelor sau metode de scalare a datelor, algoritmi de selectare a caracteristicilor și de reducere a dimensionalității, cum ar fi PCA, LDA și alții, algoritmi de clasificare și regresie și, de asemenea, o mare varietate de metode

de vizualizare a datelor și a rezultatelor. Arhitectura de nivel înalt al acestei platforme se regăsește în Fig. 12.

Pentru procesarea și analiza datelor de consum de apă și pentru detecția anomaliilor în calitatea apei s-a extins platforma existentă prin adăugarea unei funcționalități extrem de utile legate de posibilitatea de a crea și configura diferite pipeline-uri de procesare și analiză a datelor de intrare. Folosind această opțiune a fost creată soluția pentru detecția anomaliilor în calitatea apei, construind un pipeline care conține mai mulți pași de preprocesare și un model de AI pentru clasificare.

Pe lângă acest pipeline dinamic, pentru a avea funcționalități de analiză și procesare a seriilor de timp, au fost adăugate module noi pentru vizualizarea datelor de tip serii de timp, pentru configurarea modelelor de predicție, vizualizarea rezultatelor și evaluarea modelelor antrenate. Pentru predicție a fost adăugat algoritmul Auto-ARIMA (Yermal & Balasubramanian, 2017) și posibilitatea de a configura diferite arhitecturi de rețele neuronale recurente de tip LSTM (Lu et al., 2020), (Xingjian et al., 2015).

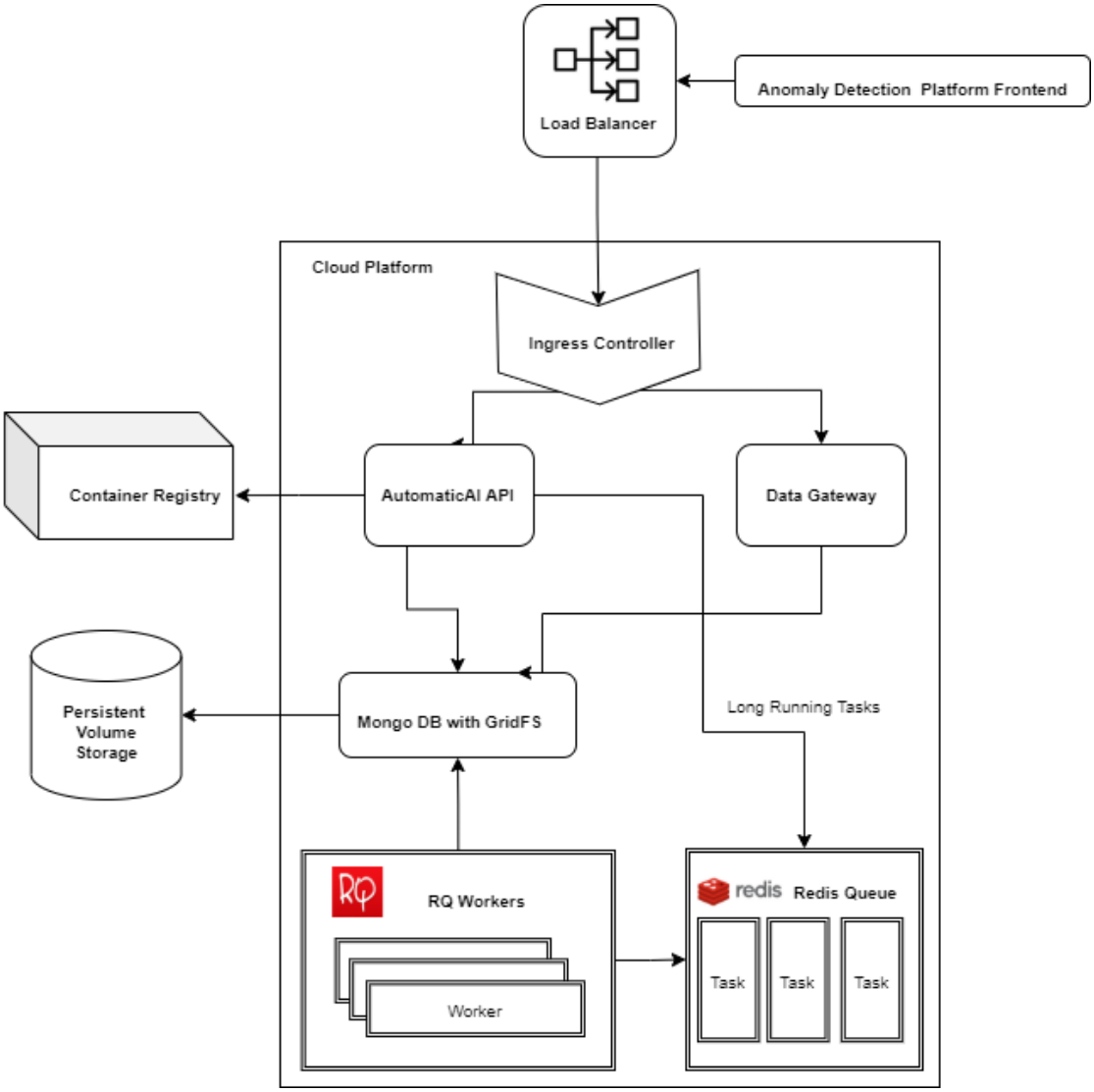


Fig. 12 Arhitectura de nivel înalt a platformei AutomaticAI

Toate aceste module și funcționalități noi au fost adăugate în serviciul AutomaticAI și a fost creată și o parte de interfață cu utilizatorul (UI) pentru folosirea cât mai ușoară a acestora.

3.2.2 Clustering pe bază de date de consum folosind K-Means (UPB.A1)

Metoda de clustering K-Means a fost utilizată pe setul de date Helix (Chatzigeorgakidis, 2018) pentru a împărți consumatorii în grupuri similare în funcție de consumul de resurse de apă, în timp ce diferite metode de pre-procesare au fost utilizate pentru a crește nivelul de detaliu în ceea ce privește comportamentele identificate ale acestora. Pipeline-ul de clusterizare este prezentat în Fig. 13Fig. 13.

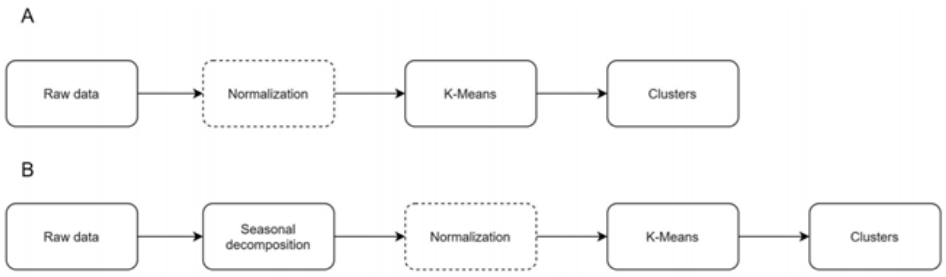


Fig. 13 Pipeline clusterizare

3.2.3 Clustering pe bază de coordonate geografice folosind OPTICS (UPB.A2)

Pentru a clasifica tipurile de consumatori pentru diferite zone geografice, identificate prin geo-locația acestora s-a utilizat clusterizarea OPTICS pornind de la setul de date Helix (Chatzigeorgakidis, 2018), rezultatele evidențiind 7 clustere bazate pe coordonatele geografice.

OPTICS (Ordering Points To Identify the Clustering Structure) are multe asemănări cu algoritmul DBSCAN și poate fi considerat o generalizare a DBSCAN, cu diferența că păstrează ierarhia clusterului pentru o rază variabilă de vecinătate. Diferența cheie dintre DBSCAN și OPTICS este faptul că algoritmul OPTICS construiește un graf de accesibilitate, care atribuie fiecărui eșantion atât o distanță de accesibilitate, cât și un punct în cadrul atributului ordering. În plus, OPTICS este mai potrivit pentru utilizarea pe seturi de date mari decât implementarea DBSCAN.

Pentru fiecare grup, s-a efectuat o analiză a seriei temporale pentru a extrage componenta sezonieră din care a fost calculată deviația standard. DSS se bazează pe multiple strategii de clasificare care implică evaluarea nivelurilor de deviație standard în contextul unei rețele de distribuție a apei. Etapa de clasificare a implicat reatribuirea fiecărui cluster pe baza nivelurilor identificate în termeni de deviație standard, pentru a evalua tiparele de consum dintr-o perspectivă numerică.

Prin urmare, modelele de suport al deciziilor sunt definite pe baza nivelurilor de deviație standard care pot fi fixate pentru întreaga rețea sau pot varia în funcție de zonă, pentru a oferi recomandări pentru diferite tipuri de părți interesate. În cele din urmă, rezultatele au fost prezentate sub formă de hartă, subliniind tipurile de consumatori, pentru a oferi suport decizional spre obținerea unor comportamente mai sustenabile ale consumatorilor.

3.2.4 Metode de clasificare (UPB.A3)

În procesul de colectare a datelor, seturile de date pot conține și valori nule (spații libere, NaN, nedefinit etc.). Acest lucru este incompatibil cu estimatorii scikit-learn, deoarece toate valorile matricei ar trebui să fie numerice, fără spații. Una dintre soluțiile la acest inconvenient este imputarea valorilor lipsă, cu alte cuvinte, încercând-se încearcă prezicerea lor folosind date cunoscute. Clasa SimpleImputer a fost folosită pentru a codifica valorile lipsă. Pentru a oferi o evaluare relativă în ceea ce privește modelele de consum, a fost utilizată normalizarea datelor. În procesul de normalizare, datele au fost scalate la intervale stabilite, folosind clasa MinMaxScaler.

Pipeline-ul de procesare folosit în scenariile UPB.A1 - UPB.A3 (Fig. 14) a fost implementat în Python folosind pachetul scikit-learn (Pedregosa et al., 2011; scikit-learn developers (BSD License)., n.d.) pentru metodele de clustering (e.g. K-Means și OPTICS), în timp ce pre-procesarea implică pachetul statsmodels[10] pentru analiza seriilor temporale (adică descompunerea sezonieră) și *numpy* pentru normalizarea datelor.

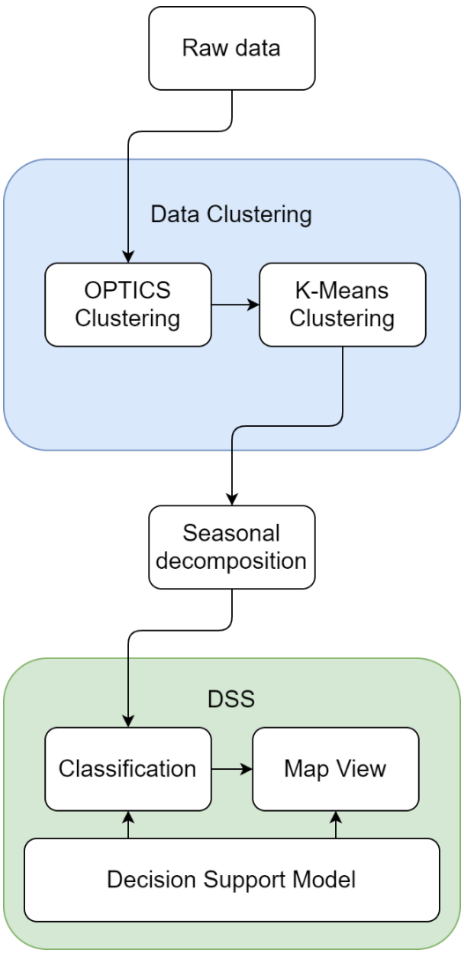


Fig. 14 Proiectarea componentei de suport decizional folosită în scenariile UPB.A1- UPB.A3

[10] https://www.statsmodels.org/stable/index.html

3.3 Crowdsensing (MCS)

În proiectarea componentei Crowdsensing, principalul obiectiv a fost de modelare a problemei într-o formă standard. Problema raportărilor în contextul MCS necesită o abordare teoretică pentru a stabili modul de alocare a sarcinilor în funcție de tipul de utilizator. În condiții reale, o astfel de aplicație presupune participarea utilizatorilor, fapt pentru care proiectarea componentei nu este doar o problemă tehnică.

Este necesar de luat în calcul modul în care sunt alocate activitățile de raportare în funcție de profilul utilizatorilor, pentru a asigura buna funcționare a soluției în ansamblu. În acest sens, pot exista probleme de ordin tehnic: scalabilitate, complexitate, tehnologii emergente, dar și non-tehnic: motivație, stimularea participării, raportări false.

Din perspectivă teoretică, o astfel de problemă se poate reduce la un scenariu de optimizare, pentru a obține un echilibru dintre priorități și reputația utilizatorilor. Alocarea activităților de raportare (identificare probleme, soluționare) poate fi modelată ca o problemă a rutării vehiculelor (VRP) adaptată la scenariul de joc, pentru o abordare generală centrată pe utilitatea platformei. Din perspectiva motivației și a costurilor individuale, un mecanism pe bază de licitații Vickrey-Clarke-Groves (VCG) reprezintă o abordare centrată pe utilizator (Hanum et al., 2018; Makowski & Ostroy, 1987).

O astfel de soluție nu poate fi evaluată decât în contextul unei aplicații pe scară largă, cu implicarea unui număr mare de utilizatori pentru a obține rezultate relevante. De aceea, pentru un model de laborator, aducerea problemei la o formă standard (VRP, VCG) permite proiectarea componentei pe baza unor soluții existente, deja validate în alte studii, și reprezintă principalul element de inovație în contextul MCS.

Problema raportărilor prin crowdsensing poate fi abordată ca o problemă de rutare în două etape. Alocarea sarcinilor este tradusă într-o problemă de rutare a vehiculelor (VRP) cu constrângeri de capacitate, în timp ce mecanismul de stimulare este perfecționat în continuare folosind un model de licitație Vickrey-Clarke-Groves (VCG) pentru raportare veridică.

3.3.1 Model de alocare a raportărilor bazat pe VRP

Modelul VRP reprezintă abordarea centrată pe platformă pentru MCS. Deși există mai multe variații ale VRP utilizate pentru rezolvarea diferitelor probleme, modelul este adaptat la scenariul MCS propus în arhitectura componentei de Crowdsensing, prin definirea contextului și a constrângerilor necesare.

Modelul de alocare a sarcinilor bazat pe VRP este definit după cum urmează: participanții sunt reprezentați de vehicule, cu capacități în funcție de nivelurile de reputație, în timp ce constrângerile de distanță definesc, de exemplu, distanța maximă de parcurs. Sarcinile sunt reprezentate de clienți, cu locații fixe și cerințe stabilite în funcție de prioritățile atribuite. Fiecare vehicul are un punct de plecare, care corespunde unui depozit, în timp ce celelalte noduri sunt asociate clienților (Hanum et al., 2018).

Pentru un scenariu de tip joc serios pe bază de geolocație, în care un set de provocări sunt distribuite pe hartă, proiectarea jocului pentru realizarea unei distribuții optime a participanților poate

fi definită ca o problemă de optimizare a expedierii combinată cu un mecanism de stimulare bazat pe recompense. Sarcinile implică raportarea problemelor în contextul apei urbane și pot fi extinse pentru multe cazuri de utilizare.

Cu toate acestea, pentru a obține o utilitate generală a platformei, stimulentul bazat pe recompense trebuie să fie contrabalansat de o strategie de reglementare. Scorul de reputație este moneda principală în joc și trebuie definit pentru a obține stimulente optime pentru raportări, reducând în același timp riscul de supra-raportare sau raportare falsă.

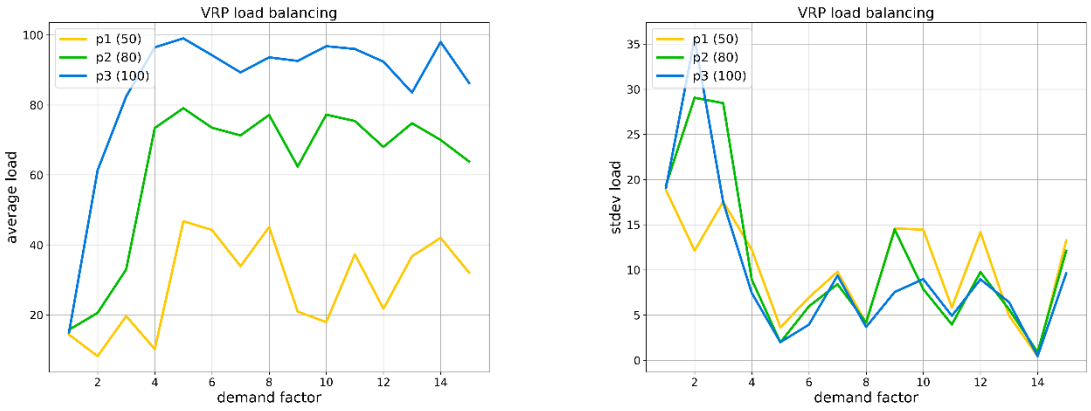
În acest sens, algoritmul VRP poate fi configurat prin constrângerile de capacitate pentru fiecare participant în funcție de reputația acestuia. La fiecare iterație a algoritmului de optimizare, capacitățile sunt actualizate. În cazul raportării veridice, capacitatea este mărită în funcție de recompensă, ceea ce permite direcționarea către sarcinile următoare. Am luat în considerare trei tipuri de participanți: potrivire 100% (raportare veridică și exactă), potrivire 80% (raportare veridică și în mare parte exactă) și potrivire 50% (raportare falsă / inexactă).

Pentru a evalua modelul și diferitele tipuri de comportament, rulăm problema de optimizare pe întreaga gamă de distribuții în ceea ce privește coordonatele geografice ale participanților. Am luat în considerare coordonatele fixe pentru sarcini și coordonatele variabile de pornire pentru participanți, pentru a evalua costul mediu al sistemului (distanța totală parcursă) în funcție de comportamentul fiecăruia și, prin urmare, pentru a evalua utilitatea platformei pentru încurajarea participării sincere.

Modelul de optimizare este implementat în Python, bazat pe framework-ul Google OR-Tools, care oferă o formulare generală a problemei VRP cu mai multe aspecte și constrângeri (Google, 2021b). Pentru a evalua echilibrarea încărcării VRP, adică alocarea sarcinilor, am luat în considerare un set de 10 locații din București și trei participanți care corespund comportamentului simulat: raportare precisă, raportare în mare parte exactă și raportare falsă. Modulul de rutare este inițializat cu matricea distanțelor, numărul de participanți și locația depozitelor (locațiile de pornire).

Pentru experimentul prezentat în Fig. 15, am evaluat efectul acestui factor de recompensă în intervalul de la 1 la 15, unde un factor de 5 are ca rezultat o distribuție globală echilibrată (recompense totale vs reputația totală). Deoarece locația de pornire a participanților poate afecta acuratețea rezultatelor, am considerat o inițializare aleatorie folosind un cerc de delimitare. Algoritmul VRP este evaluat pentru 1000 de iterații, iar sarcina medie (alocarea resurselor) este prezentată pentru fiecare tip de participant. Conform rezultatelor, în timp ce problema de optimizare este rezolvată în orice mod, factorul optim de recompensă ar trebui definit pentru a obține rezultatul dorit pentru mecanismul de stimulare. Prin urmare, ar trebui să existe un echilibru între reputația participanților și recompensele alocate pentru fiecare sarcină disponibilă.

Prin urmare, geo-localizarea și alte constrângeri pot servi drept filtru pentru trimiterea sarcinilor către participanți pentru a realiza o strategie de rutare rentabilă pentru o utilitate sporită a platformei în ceea ce privește raportarea veridică și calitatea raportărilor.



(a) (b)

Fig. 15 Alocarea activităților prin VRP (a) Alocare medie; (b) Deviație standard.

3.3.2 Model de licitație bazat pe VCG pentru raportare veridică

Abordarea centrată pe utilizator bazată pe VCG este definită în contextul platformei MCS propuse. În licitația VCG, există mai multe articole și agenți, fiecare agent plasând o ofertă pentru fiecare articol, fără a cunoaște evaluările altora. Sistemul de licitație atribuie articolele în funcție de evaluările lor reale, în timp ce fiecare agent plătește doar valoarea marginală în comparație cu ceilalți agenți, încurajând licitarea veridică (Makowski & Ostroy, 1987; Sessa et al., 2017; Tao & Song, 2018).

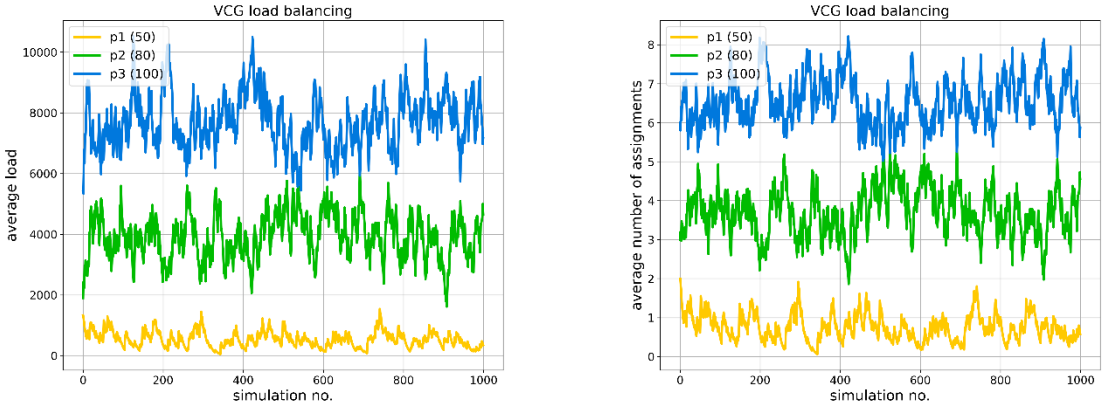
Mecanismul promovează licitarea sinceră prin compatibilitatea stimulentelor (licitarea sinceră este o strategie dominantă), raționalitatea individuală (fiecare agent sincer primește o plată), eficiența (licitarea sinceră maximizează bunăstarea socială).

Modelul de licitație VCG se bazează pe o implementare Python care primește datele de intrare dintr-un fișier text, descriind articolele și ofertele disponibile pentru fiecare participant și returnează alocarea articolelor în conformitate cu strategia de licitare veridică. Matricea distanțelor este tradusă în formatul de intrare după cum urmează: sumele licitate pentru fiecare sarcină sunt definite în funcție de distanță (costul raportat) și reputația normalizată a participantului. Beneficiul total și numărul de articole pentru fiecare participant sunt extrase pentru a compara rezultatele.

Folosind o metodă similară cu cea prezentată în cazul VRP, modelul VCG a fost integrat în simulator și rezultatele au fost evaluate pentru configurații aleatorii în ceea ce privește pozițiile de start ale participanților. Același număr de iterații (1000) a fost folosit pentru a compara rezultatele în ceea ce privește echilibrarea optimă a sarcinii într-o distribuție variabilă a participanților pe hartă.

Pentru inspecția vizuală, rezultatele sunt post-procesate folosind un filtru de medie alunecătoare (sliding window), așa cum se arată în Fig. 16 pentru fiecare tip de participant. Am luat în considerare aceleași tipuri de participanți cu reputație normalizată: 1 (raportare veridică și exactă), 0.8 (raportare veridică și în cea mai mare parte exactă) și 0.5 (raportare falsă/inexactă).

Rezultatele arată că atât un număr mai mare de sarcini, cât și o recompensă totală mai mare sunt atribuite pentru licitarea sinceră pe întreaga gamă de trasee simulate, care este direct influențată de reputația relativă a participanților.



(a) (b)

Fig. 16 Alocarea activităților prin VCG (a) Recompensă medie; (b) Sarcini alocate în medie.

3.4 Blockchain (B)

Componenta blockchain este construită pe baza tehnologiei Ethereum și a unora dintre cele mai noi platforme de dezvoltare, folosind Truffle, Ganache, Metamask și Web3.js.

3.4.1 Truffle

Pentru implementarea soluției blockchain bazată pe tehnologia Ethereum s-a folosit Truffle (ConsenSys Software Inc., 2021) ca mediu de dezvoltare și testare. Acesta oferă numeroase facilități pentru realizarea unei aplicații blockchain printre care: compilarea smart contract-urilor, testarea acestora, linie de comandă, pipeline-uri configurabile, interogarea rețelei blockchain pentru vizualizarea detaliilor fiecărei adrese (ETH disponibil pentru fiecare nod, cantitate deținută - moneda WGT), etc.

3.4.2 Ganache

Ganache[11] este o componentă a suitei Truffle și oferă posibilitatea realizării on-click a unei rețele blockchain locală pe baza căreia se pot dezvolta smart contracts. Prin intermediul Ganache avem acces atât la o interfață UI prin care se pot vedea în timp real wallet-urile / nodurile conectate la rețeaua locală, cât și la un utilitar de tip linie de comandă (Fig. 17).

Ganache poate fi folosit pe tot parcursul dezvoltării, pentru dezvoltare, deploy și testarea aplicației implementate.

[11] https://www.trufflesuite.com/docs/ganache/overview

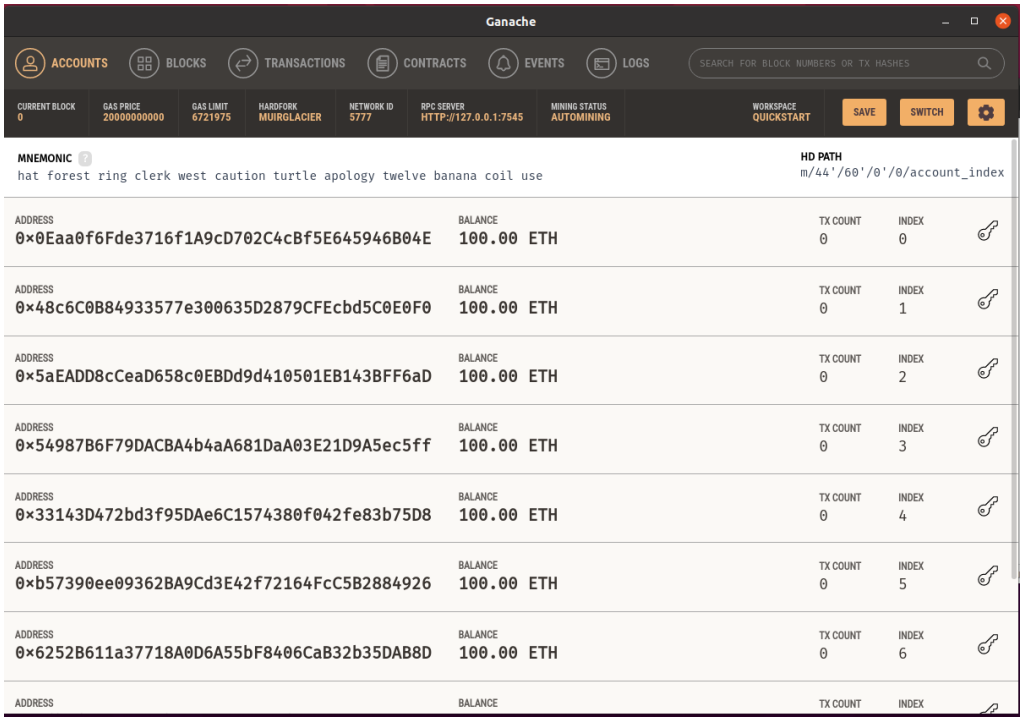


Fig. 17 Interfata Ganache

3.4.3 Smart Contracts

Pentru realizarea funcționalităților soluției blockchain este necesară implementarea contractelor inteligente. Astfel au fost dezvoltate următoarele: contract pentru realizarea monedei virtuale (WGT – Water Game Token), conform standardului ERC-20 (ethereum.org, 2021), monedă ce va fi oferită ca recompensa pentru clienți în urma acțiunilor acestora, dar care va putea fi și tranzacționată separat ca orice altă criptomonedă; contracte pentru facilitarea următoarelor acțiuni: cumpărare de WGT, transfer de WGT dintr-un portofel în altul, sau donare de WGT.

Implementarea contractelor inteligente a fost realizată folosind limbajul de programare Solidity[12], limbaj orientat pe obiect, specific pentru a interacționa cu tehnologia blockchain.

3.4.4 Metamask si Web3.js

Pentru facilitarea interacțiunii utilizatorului cu soluția blockchain s-a folosit Web3.js[13] împreună cu Metamask (MetaMask, 2021) (vezi Fig. 18). Metamask va fi folosit ca un gateway și oferă posibilitatea comunicării cu aplicații de tip blockchain, dar poate fi folosit și ca wallet. Metamask oferă posibilitatea de administra mai multe conturi / chei de conturi, de a tranzacționa criptomonede într-un mod securizat și a accesa aplicații descentralizate prin intermediul unei extensii ce poate fi instalată la nivelul unui browser web sau prin intermediul unei aplicații mobile.

[12] https://docs.soliditylang.org/en/v0.8.10/

[13] https://web3js.readthedocs.io/en/v1.5.2/

Web3.js reprezintă o colecție de biblioteci JavaScript prin intermediul cărora se facilitează interacțiunea cu un nod din rețeaua blockchain în scopul efectuării diferitelor operațiuni folosind o conexiune HTTP sau IPC (Inter-Process Communication).

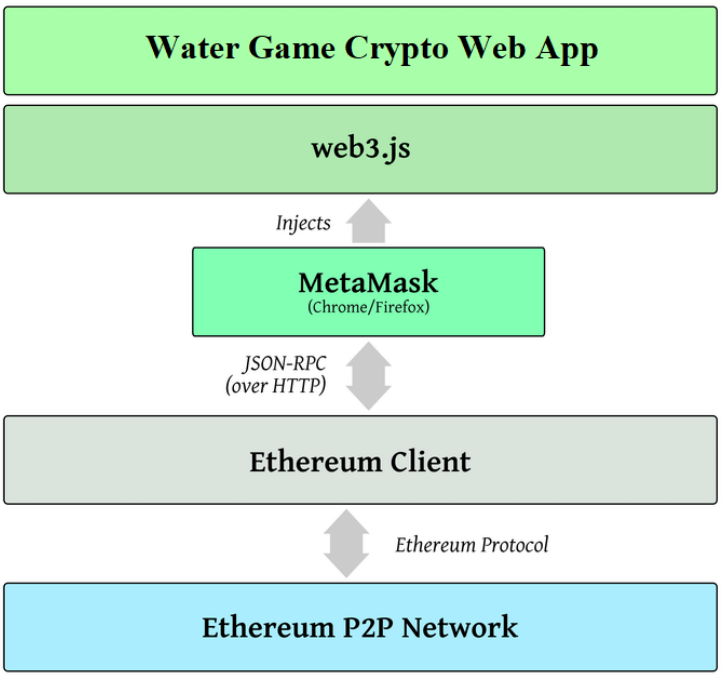


Fig. 18 Proiectarea componentei blockchain

4 Detalii de implementare

Implementarea componentelor presupune metode și tehnologii eterogene, de la componenta hardware de colectare a datelor de consum, la sistemul colaborativ pentru interacțiunea cu utilizatorul, la sistemul de suport decizional și până la infrastructura blockchain pentru validarea tranzacțiilor de stimulare a participării. Arhitectura decuplată la nivel de sistem va permite integrarea acestor componente în următoarea etapă.

4.1 Monitorizare consum (WMS)

Implementarea componentei de monitorizare a consumului presupune conectarea debitmetrelor YF-S201 la modulele WiFi de tip NodeMCU (Karray et al., 2018). Majoritatea debitmetrelor disponibile comercial sunt debitmetre construite și care funcționează pe baza efectului Hall. Acest lucru determină ca în construcția senzorilor de măsurare din cadrul debitmetrelor să existe un senzor magnetic de efect Hall integrat care, folosind o rotiță, produce un impuls electric la fiecare rotație a acesteia. Astfel valoarea primară pe care un debitmetru o furnizează este frecvența impulsurilor sau numărul de impulsuri pe unitatea de timp. Pe baza acestei valori a frecvenței impulsurilor se calculează valoarea debitului de apă în litri / minut sau litri pe oră. Formula de calcul utilizată pentru modelele de debitmetre de acest tip presupune o împărțire a valorii frecvenței impulsurilor la un factor numeric supraunitar.

În fiecare locuință, au fost instalate debitmetre și un număr minim de module WiFi în funcție de configurația locală. În general, un singur modul a fost instalat în fiecare locație (e.g. baie, bucătărie), la care au fost conectate debitmetrele prin cabluri (de ex.: chiuvetă apă caldă / rece, duș combinat, toaletă, mașină de spălat).

4.1.1 Componentele de monitorizare debit de apă și nivel edge (UTCN.C)

Pentru sistemul de monitorizare a debitului de apă s-au realizat 3 modele de montaj.

În primul caz (UTCN.C1) s-au folosit în locații debitmetre de tipul YF-S201 conectate la noduri NodeMCU cu chip wireless de tip ESP8266[14]. Alimentarea acestora la punctele de măsurare s-a făcut cu surse de tip AC. Aceste surse au fost înlocuite în situațiile în care spațiul disponibil pentru montaj și sursele de alimentare cu curent alternativ nu sunt posibile, cu surse de alimentare bazate pe acumulatori externi conform montajului din Fig. 19. Acest tip de alimentare este posibilă doar pe perioade limitate de timp, cu necesitatea înlocuirii sursei de alimentare o dată la 96 ore. Mai departe acestea au fost configurate să se conecteze la punctul local de acces din fiecare locație unde sunt instalate, pentru a putea comunica cu serviciul web din cloud și pentru a transmite acestuia datele de consum.

La nivelul nodului NodeMCU se utilizează un sketch care programează partea de configurare a nodului inclusiv din punctul de vedere al conectivității, execută partea de preprocesare a datelor, calculează consumul de apă, iar apoi face postarea datelor formatate cu câmpurile de identificare spre serviciul din cloud.

[14] https://usermanual.wiki/Document/YFS201manual.662398285/view

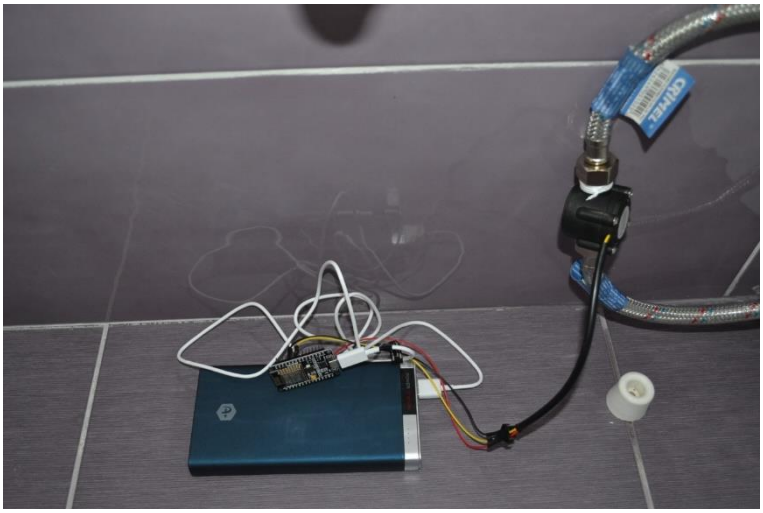


Fig. 19 Montaj debitmetru conectat la NodeMCU alimentat cu sursă externă

Modelul de sketch folosit local pe nodul NodeMCU pentru testarea modului de calcul a debitului de apă este următorul:

```
void setup()
{
   pinMode(flowmeter, INPUT);
   Serial.begin(9600);
   attachInterrupt(0, flow, RISING);
   sei();
   currentTime = millis();
   cloopTime = currentTime;
}
void loop ()
{
   currentTime = millis();
   if(currentTime >= (cloopTime + 1000))
   {
      cloopTime = currentTime;
      l_hour = (flow_frequency * 60 / 7.5);
      flow_frequency = 0;
      Serial.print(l_hour, DEC);
      Serial.println(" L/hour");
   }
}
```

Conform specificațiilor producătorului debitmetrului YF-S201, formula de calcul pentru debitul de apă măsurat în litri / oră este: Debit (l / h) = Frecvența_impulsurilor * 60 / 7.5.

În al doilea caz (UTCN.C2) sunt montate multiple debitmetre la mai multe noduri NodeMCU aflate în puncte diferite în cadrul aceleiași locații. Tot în cadrul locației este instalat un SBC de tip Raspberry PI Model 3+[15]. În Fig. 20 se poate observa un montaj cu debitmetru și NodeMCU alimentat la sursa de tip AC / DC.

Fig. 20 Montaj debitmetru conectat la NodeMCU alimentat cu sursa AC / DC

În acest caz, la nivelul nodului NodeMCU s-a testat funcționarea comunicării cu SBC-ul Raspberry PI prin intermediul sketch-ului următor:

```
unsigned long timerDelay = 5000;

void setup() {

Serial.begin(9600);

WiFi.begin(ssid, password);

Serial.println("Connecting");

      while(WiFi.status() != WL_CONNECTED) {

          delay(500);

          Serial.print(".");

        }
```

15 https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf

```
        Serial.println("");
        Serial.print("Connected to WiFi network with IP Address: ");
        Serial.println(WiFi.localIP());
      }
void loop() {
      if ((millis() - lastTime) > timerDelay) {
        if(WiFi.status()== WL_CONNECTED){
          HTTPClient http;
          http.begin(serverName);
          http.addHeader("Content-Type","application/x-www-form-urlencoded");
          http.addHeader("Content-Type", "application/json");
          int httpResponseCode = http.POST("\"12\"");
          Serial.print("HTTP Response code: ");
          Serial.println(httpResponseCode);
          http.end();
          }
          else {
            Serial.println("WiFi Disconnected");
          }
          lastTime = millis();
        }
      }
```

La nivelul nodurilor NodeMCU se calculează, bazat pe valoarea frecvenței impulsurilor, doar consumul în litri / oră, care este transmis formatat cu datele de identificare ale punctului de măsurare către SBC. La nivelul SBC-ului se face pre-procesarea datelor de consum, care ulterior sunt transmise în cloud pentru vizualizare și analiză mai avansată.

În cazul al treilea (UTCN.C3), similar cu ceea ce se întâmplă în cazul 2, se conectează debitmetrele la nodurile NodeMCU corespunzătoare în funcție de dispunerea punctelor de măsurare în cadrul locației după care acestea execută citirile datelor de consum, calculează consumul în litri / minut, pre-procesează datele și le transmit mai departe unui dispozitiv embedded cum este cel de la Nvidia Jetson Nano. La nivelul acestui dispozitiv se face partea de procesare a datelor.

4.1.2 Serviciul de colectare a datelor (UTCN.C)

Soluția aleasă pentru serviciul de colectare a datelor este o implementare open source a standardului OGC SensorThings, Frost server[16]. Implementarea este una bazată pe containere

[16] https://fraunhoferiosb.github.io/FROST-Server/

Docker. Aplicația conține două containere, unul pentru serviciul de colectare a datelor și altul pentru baza de date PostgreSQL cu extensie Postgis. Aplicația poate primi atât cereri HTTP cât și cereri MQTT. În prezent, componentele sistemului dezvoltat în proiect utilizează doar cereri HTTP.

Pentru instalare în cloud s-a folosit următorul fișier Docker compose:

```
version: '2'


services:
  web:
    image: fraunhoferiosb/frost-server:latest
    environment:
      - serviceRootUrl=http://10.20.1.47/:8080/FROST-Server
      - http_cors_enable=true
      - http_cors_allowed.origins=*
      - persistence_db_driver=org.postgresql.Driver
      - persistence_db_url=jdbc:postgresql://database:5432/sensorthings
      - persistence_db_username=sensorthings
      - persistence_db_password=xxx
      - persistence_autoUpdateDatabase=true
    ports:
      - 8080:8080
      - 1883:1883
    depends_on:
      - database


  database:
    image: postgis/postgis:11-2.5-alpine
    environment:
      - POSTGRES_DB=sensorthings
      - POSTGRES_USER=sensorthings
      - POSTGRES_PASSWORD=xxx
    volumes:
      - postgis_volume:/var/lib/postgresql/data
volumes:
    postgis_volume:
```

În Fig. 21 se poate observa răspunsul serverului Frost la o interogare despre Observații.

Fig. 21 Răspunsul serverului Frost la o interogare despre Observații

API-ul Frost poate să răspundă la interogări legate de toate entitățile specificate în modelul de date (Senzori, Locații, Datastreams etc.).

4.1.3 Configurarea modulelor de achiziție (UPB.C)

Modulele NodeMCU (NodeMcu Team, 2018) sunt inițial programate în Arduino IDE și expun anumite configurări necesare pentru instalarea în locația consumatorului (Bento, 2018).

Configurarea plăcilor se realizează prin intermediul WiFi, acestea funcționând atât în modul de stație, cât și în modul AP (access point). Pagina de configurare a modulelor este găzduită local și poate fi accesată prin conectarea la punctul de acces propriu la adresa 192.168.4.1 (Fig. 22). Conexiunea este securizată la nivel de protocol WiFi, prin WPA-PSK (Khasawneh et al., 2014).

Se inițializează parametrii precum: configurația pinilor la care sunt conectate debitmetrele și parametrii de conectare (numele și parola rețelei WiFi locale). Pagina de configurare oferă și informații suplimentare legate de măsurătorile realizate ușurând astfel instalarea și depanarea la fața locului.

Configurările sunt salvate în memoria EEPROM și pot fi modificate doar prin conectarea la modul, caz în care este solicitată parola de WiFi, sau prin comenzi de la distanță, prin intermediul mesajelor transmise prin MQTT, caz în care este solicitată o parolă pentru conectare.

4.1.4 Interfață de comunicare (UPB.WSN)

Interfața de comunicare este proiectată pentru a facilita transmiterea datelor în rețeaua de senzori wireless (WSN). Prin intermediul conexiunii WiFi locale, modulele se conectează la brokerul MQTT din Cloud și trimit pachete de date la diferite intervale:

- date de debit - frecvență ridicată (5 s), pe topicul wsn/watergame/realtime;
- date de volum - frecvență scăzută (10 min), pe topicul wsn/watergame/data;
- date de control - configurare de la distanță, pe topicul wsn/watergame/cmd.

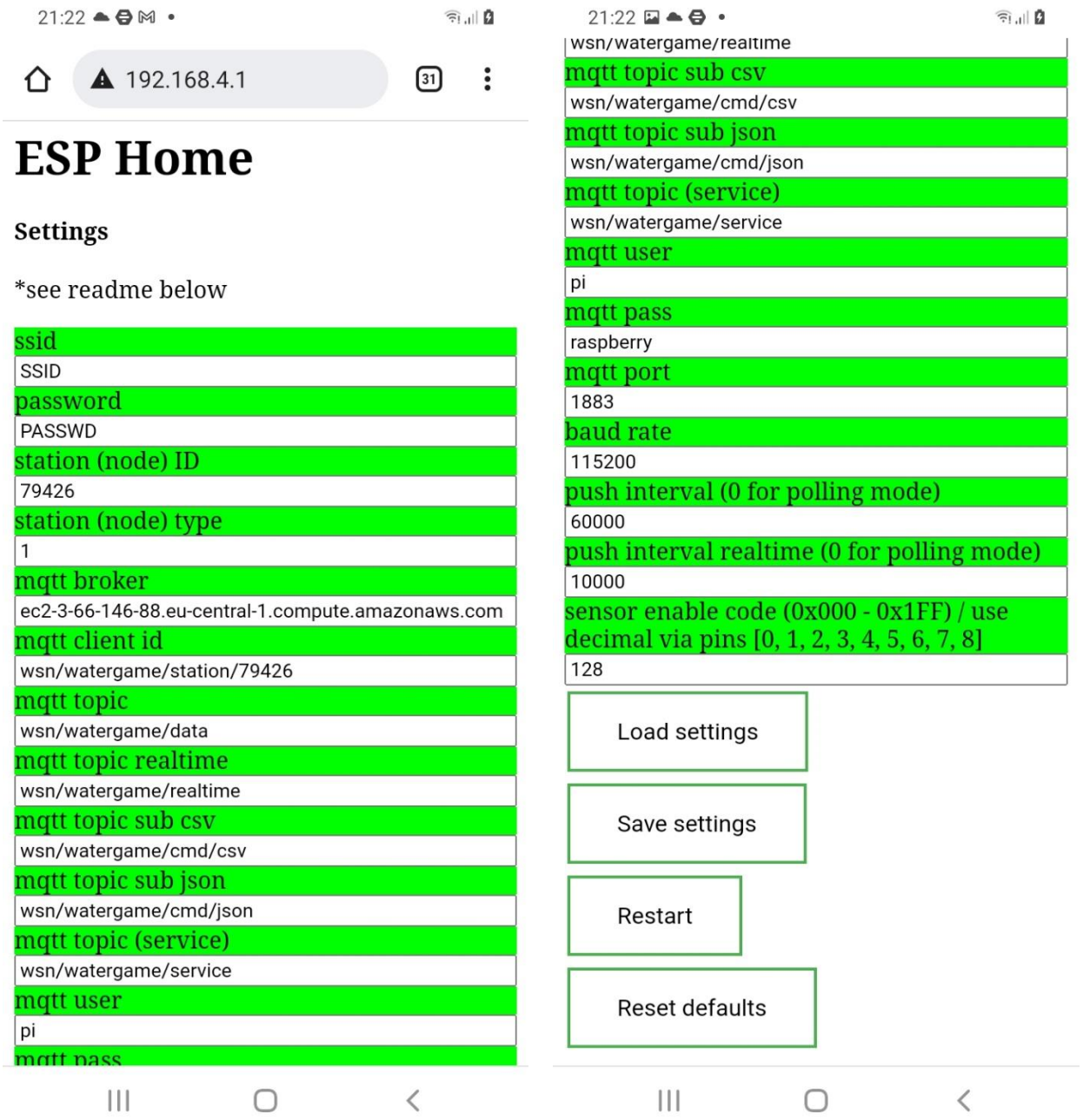


Fig. 22 Interfața integrată de configurare a modulului de achiziție

Datele sunt colectate de către server, acesta fiind abonat la brokerul de MQTT, astfel:

- Datele cu frecvență redusă sunt stocate în baza de date MySQL pentru păstrarea istoricului de consum pe termen lung.
- Datele cu frecvență ridicată sunt utilizate pentru afișarea în timp real și sunt stocate temporar în memoria REDIS.

Prin aplicația client MQTTBox[17] se pot vizualiza datele care se transmit în timp real. Atunci când modulele trimit datele către brokerul de MQTT, se realizează operația de "publish" ce poate fi monitorizată prin abonarea la acel topic (Fig. 23). Atunci când comenzile sunt recepționate de module prin brokerul de MQTT, se realizează operația de "subscribe", iar comenzile pot fi trimise prin operația de “publish” din aplicația client (Mishra & Kertesz, 2020).

Pot exista astfel mai mulți abonați, precum server-ul NodeJS aflat în cloud pe Amazon EC2 (Amazon Web Services, 2021), care preia datele și le stochează în baza de date MySQL sau aplicația mobilă ce afișează datele în timp real.

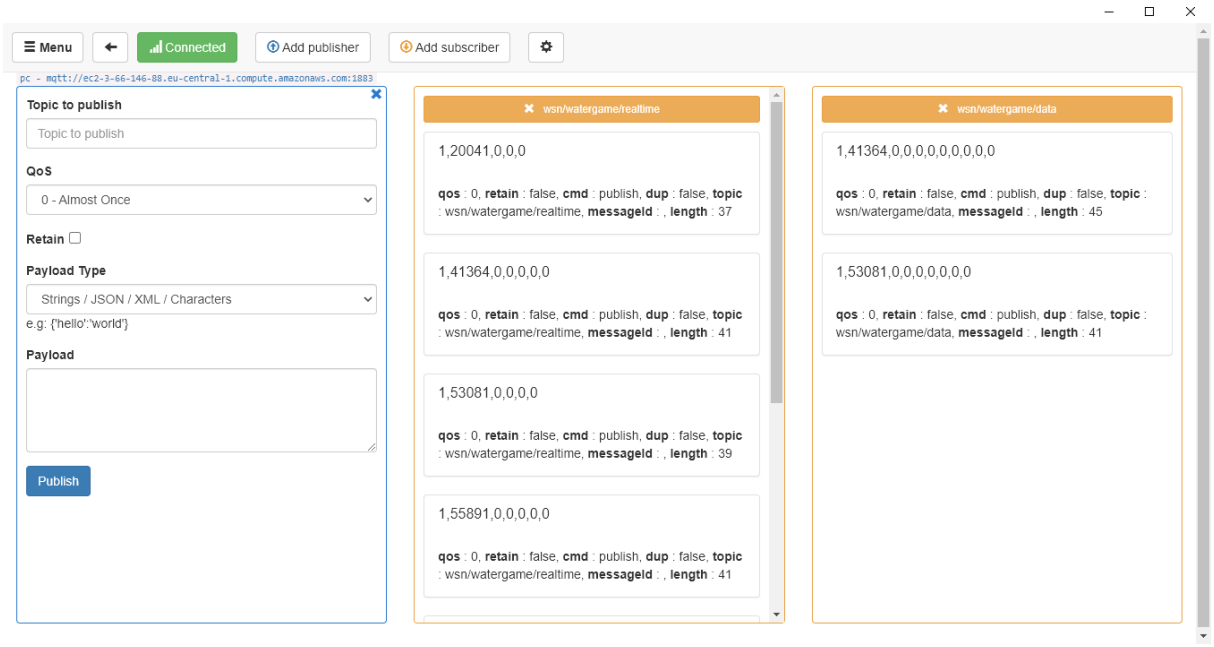


Fig. 23 Vizualizarea datelor cu MQTTBox

Pentru identificarea senzorilor, datele trimise de un modul sunt împachetate în format CSV (comma-separated values), având următoarea structură: "node_type, node_id, recv_command, channel_0_value, channel_1_value, ...", unde:

- **node_type** - tipul elementului care a trimis datele (în acest caz avem doar senzori de debit, având codificarea 1);
- **node_id** - id-ul modulului de achiziție;
- **recv_command** - tipul comenzii (100 pentru identificarea pachetului de date, 110 pentru identificarea comenzilor);
- **channel_0_value, channel_1_value, …** - valorile în format CSV corespunzătoare canalelor de măsurare configurate pentru modulul de achiziție.

4.1.5 Procesul de instalare (UPB)

Instalarea constă în montarea debitmetrelor și a modulelor de comunicație. Debitmetrele se instalează prin racord flexibil de 1/2” cu garnitură de etanșare. Pentru situațiile în care se folosește

[17] https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf

racord de 3/8” se adaugă un racord sau adaptor 1/2” la 3/8”. În cazul bateriei de la duș, montarea senzorului se realizează prin racord fix de 1/2” iar furtunul de la duș se conectează direct la celălalt capăt al debitmetrului (Fig. 24).

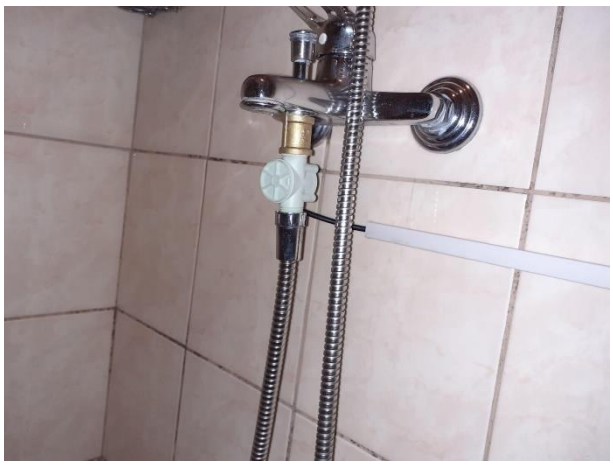


Fig. 24 Senzor de debit instalat la bateria de duș

Modulele de achiziție sunt amplasate în locații mai greu accesibile, precum masca de la chiuvetă. Debitmetrele se conectează prin cabluri la un modul de achiziție, fapt pentru care este necesară o evaluare a locației în vederea dimensionării și amplasării corespunzătoare.

Configurarea modulelor se realizează după ce au fost montate debitmetrele, prin conectarea la pagina de configurare, apoi se verifică datele colectate.

În următoarele imagini sunt prezentate câteva module instalate în locuințe. În Fig. 25 este prezentat un debitmetru conectat la bateria de duș. Cablurile sunt instalate într-un traseu de cabluri pentru o conexiune robustă. Distanța dintre module și debitmetre nu a reprezentat un impediment pentru captarea impulsurilor generate de debitmetre, fiind instalate debitmetre la distanțe de la 20 cm până la 10 m.

În Fig. 25 este prezentat un modul de achiziție montat sub masca de la chiuvetă și conexiunile cu debitmetrele atașate la alimentarea cu apă caldă și rece.

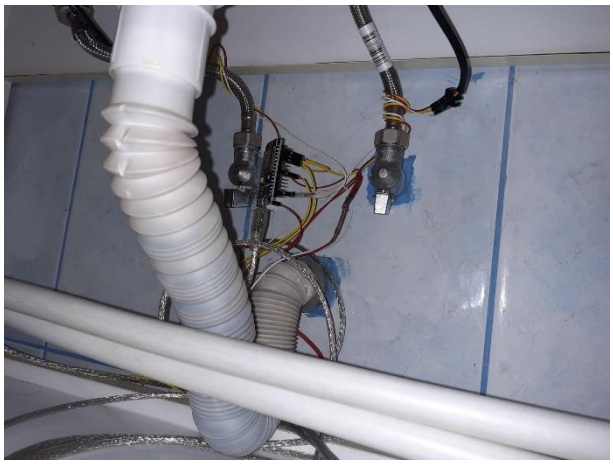


Fig. 25 Modul de achiziție instalat sub chiuvetă

În Fig. 26 este prezentat un debitmetru instalat la alimentarea cu apă a toaletei, conectat prin cabluri la modulul de achiziție aflat pe peretele opus, sub masca de la chiuvetă. Cablurile sunt instalate prin traseul de cabluri atașat de perete.

Fig. 26 Senzor de debit instalat la alimentarea cu apă a toaletei

În primă fază, am instalat 32 de senzori de debit conectați la 11 module de achiziție conform tabelului de mai jos:

Tabel 1 Module de achiziție instalate pentru colectarea datelor de consum din locuințe

modul				D1	D2	D3	D5	D6	D7	localizare	
no.	id	locație	spațiu	chiuvetă rece	chiuvetă caldă	toaletă	duș	mașină spălat	mașină spălat vase	lat	lng
1	97074	DA	bucătărie	x	x					44.48634	26.04781
2	20041	CC	bucătărie	x	x					44.411789	26.1273017
3	62706	CC	baie_1	x	x	x				44.411789	26.1273017
4	41364	CC	baie_2	x	x	x	x			44.411789	26.1273017
5	33383	CC	baie_3	x	x	x	x			44.411789	26.1273017
6	53081	CC	baie_4	x	x	x				44.411789	26.1273017
7	55891	CC	baie_5	x	x	x	x			44.411789	26.1273017
8	16102	AT	bucătărie	x	x					44.414563	26.0335787

9	14945	AT	baie			x				44.414563	26.0335787
10	89776	AB	bucătărie	x	x					44.377512	26.1381178
11	65506	AB	baie	x	x	x		x	x	44.377512	26.1381178

Fiecare modul este reprezentat printr-un ID unic generat la instalare și este configurat pentru colectarea datelor de la senzorii de debit atașați. Astfel, pentru fiecare intrare de date (canal de măsurare), au fost atașați senzorii conform tabelului: primele 2 canale (D1, D2) sunt folosite pentru chiuvetă (apă rece, apă caldă), al treilea (D3) pentru toaletă, al cincilea (D5) pentru duș, iar ultimele 2 canale (D6, D7) pentru mașina de spălat, respectiv mașina de spălat vase.

Modulele de achiziție au fost programate anterior instalării și permit configurarea intrărilor de date din interfața de configurare. În urma testelor, pinul D4 a cauzat probleme de stabilitate, fiind identificată ulterior corespondența cu funcția de inițializare a modulului ESP8266 în modul de programare. Astfel, pinul D4 nu a mai fost conectat, iar senzorul pentru duș a fost conectat la pinul D5.

4.1.6 Aplicație mobilă (UPB.WMS)

Aplicația mobilă WaterMonitoringSystem (WMS) disponibilă pentru Android permite monitorizarea consumului de apă din perspectiva consumatorilor sau a furnizorilor, oferind o interfață interactivă pentru sistemul de monitorizare a consumului (Fig. 27). Datele de consum sunt preluate de la server-ul aplicației, iar autentificarea se realizează prin Firebase pentru o securitate sporită și decuplarea serviciilor de date (datele colectate de la senzori sunt gestionate separat de cele referitoare la conturile utilizatorilor).

Aplicația mobilă implementează cazurile de utilizare identificate în etapa de proiectare a arhitecturii sistemului, precum: vizualizarea senzorilor pe hartă, vizualizarea graficelor, cont furnizori / consumatori, raportare probleme.

Implementarea aplicației este realizată în Android Studio, folosind limbajul Java. Pentru conectarea la serviciul Firebase, se configurează un proiect în Firebase Console și o aplicație mobilă (Khawas & Shah, 2018). Platforma generează un fișier de configurare care este adăugat în proiectul Android, preluat de SDK-ul Firebase pentru conectare la baza de date.

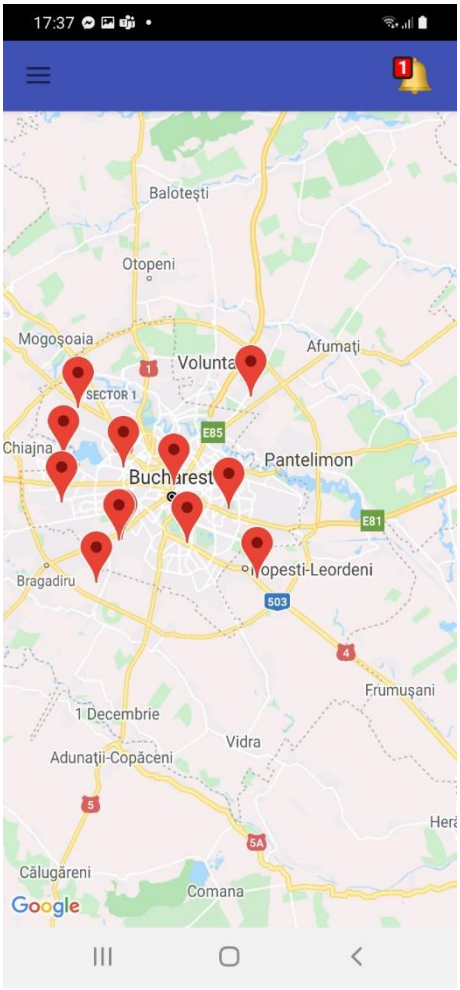


Fig. 27 Aplicația WaterMonitoringSystem (mod furnizor). Vizualizarea pe hartă a modulelor instalate.

Aplicația este formată din următoarele pachete Java:

- **activities** – pachetul în care se găsesc clasele responsabile pentru ferestrele aplicației, structurate în funcție de rol;
- **adapters** – clase responsabile cu funcționalități mai complexe, precum actualizarea listelor cu date în timp real;
- **api** – clase responsabile de conectarea la server-ul NodeJS prin cereri HTTP;
- **authentication**: clase responsabile cu procesul de autentificare și înregistrare a utilizatorilor în aplicație;
- **models** – clase ce modelează datele și mapările necesare pentru interacțiunea cu server-ul aplicației și cu serviciul Firebase;
- **mqtt** – clase ce implementează comunicația prin MQTT folosind biblioteca Paho[18] și un set de metode pentru conectarea la fluxul de date în timp real;
- **utils** – funcții de suport pentru operații mai complexe.

[18] https://www.eclipse.org/paho/index.php?page=downloads.php

4.1.7 Serviciul de date (UPB.REST)

Serviciul de date este implementat pe back end-ul principal al sistemului de monitorizare a consumului. Aplicația mobilă folosește serviciile disponibile sub formă de REST API în cadrul server-ului principal (Masse, 2011). Acesta expune un set de metode pentru accesarea datelor colectate și a informațiilor despre modulele de achiziție înregistrate conform tabelului de mai jos:

Tabel 2 REST API

endpoint	**descriere**	**parametri**
GET /sensors/manager/get-registered-sensors	vizualizarea modulelor înregistrate în baza de date	N/A
GET sensors/data/get-current-data-snapshot	vizualizarea ultimelor date transmise de un anumit modul	sensorId
GET sensors/data/get-stored-data	vizualizarea datelor înregistrate în baza de date, în format JSON	sensorId chan limit mode
GET /sensors/data/get-stored-data-processed	vizualizarea datelor grupate după ID-ul modulului de achiziție, în format JSON	sensorId chan limit mode
GET /sensors/data/get-stored-data-csv	vizualizarea datelor grupate după ID-ul modulului de achiziție, în format CSV	sensorId chan limit mode
POST /sensors/manager/set-coords	asocierea coordonatelor GPS pentru un modul de achiziție	sensorId lat lng

În următorul tabel este prezentat modul în care endpoint-urile sunt corelate cu funcționalitățile aplicației:

Tabel 3 Utilizarea endpoint-urilor în aplicație

endpoint	**descriere**	**funcționalitate**
GET /sensors/manager/get-registered-sensors	vizualizarea modulelor înregistrate în baza de date	afișarea pe hartă a modulelor instalate
GET sensors/data/get-current-data-snapshot	vizualizarea ultimelor date transmise de un anumit modul	inițializarea datelor afișate pentru fiecare senzor

GET sensors/data/get-stored-data	vizualizarea datelor înregistrate în baza de date, în format JSON	afișarea datelor colectate de un modul de achiziție în format grafic (un singur senzor)
GET /sensors/data/get-stored-data-processed	vizualizarea datelor grupate după ID-ul modulului de achiziție, în format JSON	afișarea datelor colectate de un modul de achiziție în format grafic (mai mulți senzori)
GET /sensors/data/get-stored-data-csv	vizualizarea datelor grupate după ID-ul modulului de achiziție, în format CSV	export date în fișier CSV
POST /sensors/manager/set-coords	asocierea coordonatelor GPS pentru un modul de achiziție	adăugarea sau actualizarea coordonatelor GPS pentru un modul de achiziție

4.2 Suport decizional (DSS)

4.2.1 Serviciul de procesare/analiză a datelor (UTCN.A)

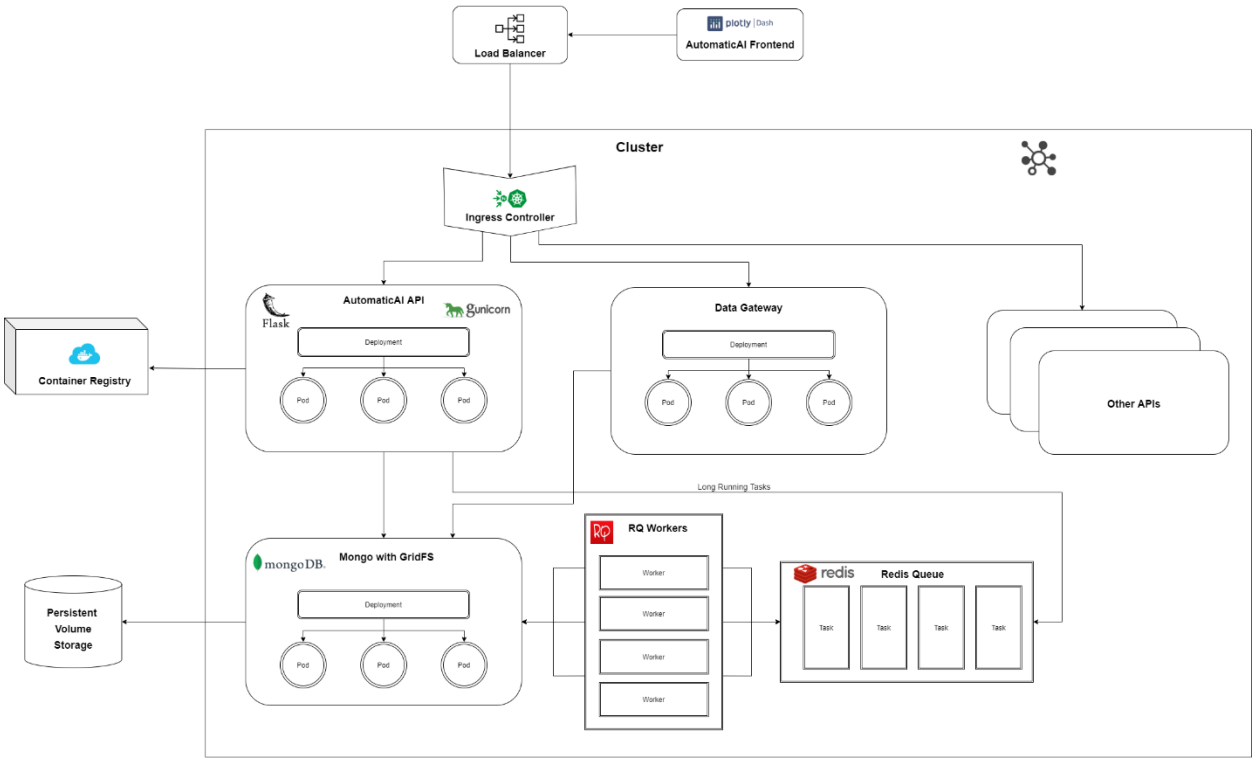


Fig. 28 Arhitectura detaliată a platformei extinse AutomaticAI

După cum se poate vedea în Fig. 28, există un site web implementat în Dash Plotly care comunică cu aplicația bazată pe microservicii printr-un load balancer. Folosind acest load balancer,

se decuplează clientul de clustere și se expune un IP public prin care serviciile din cluster pot fi apelate de clienți. Deoarece request-urile diferite pot ajunge la diferite servicii pe diferite noduri sau mașini virtuale, a fost folosit un controler de intrare (ingress controller) NGINX, care funcționează ca un proxy invers ascunzând diferitele mașini virtuale de la client și, de asemenea, rutează cererile pe baza sub-domeniului furnizat în adresa URL a request-ului.

În Fig. 28, unul dintre cele mai importante microservicii este AutomaticAI API. Acest serviciu conține logica pentru configurarea unui pipeline pentru rezolvarea problemelor de clasificare, regresie sau de detectare a anomaliilor. Pipeline-ul poate fi configurat pe baza unui json de configurare. Acest serviciu conține, de asemenea, algoritmul PSO-SA, care poate selecta automat un tip de algoritm AI și poate regla hiper-parametrii acestuia doar pe baza datelor de intrare, fără intervenție umană.

Selectarea, reglarea și antrenarea unui algoritm AI este un proces consumator de timp și de lungă durată. Pentru a preveni blocarea API-ului, se rulează toate procesele de lungă durată în paralel și asincron. Pentru a obține o paralelizare adevărată și o procesare asincronă, a fost aplicat modelul producător-consumator, în care se folosește o coadă Redis pentru a programa sarcinile care trebuie procesate offline de către procesele de lucru numite workers.

Pentru stocarea fișierelor de date de intrare și a modelelor antrenate se folosește MongoDB cu GridFS. GridFS este utilizat pentru stocarea și preluarea fișierelor mari. Aceste fișiere sunt împărțite în bucăți și fiecare bucată este stocată ca document separat. De asemenea, se stochează metadate despre bucăți, astfel fișierul original putând fi reconstruit oricând.

Serviciul numit AutomaticAI API are structura prezentată în Fig. 29. El are mai multe module, fiecare având responsabilități diferite. Modulul principal este numit App, care face legătura între diferitele componente și module ale aplicației și definește rutele prin care request-urile pot ajunge la componenta care poate să le trateze.

Aplicația conține mai multe controllere:

1) UsersAPI – conține endpointuri de CRUD pentru utilizatori (adică metode de GET, POST, PUT și DELETE);
2) DatasetsAPI – conține endpointuri de CRUD pentru seturi de date;
3) AIModelAPI – endpoint pentru a citi modelul antrenat și salvat în baza de date;
4) TrainAPI – endpoint pentru declanșarea antrenării modelului selectat;
5) JobsAPI – pentru procesele de lungă durată se generează un job care se rulează în spate. Acest modul este folosit pentru a vizualiza job-urile și statusurile lor (e util pentru depanare);
6) ModelSelectionAPI – conține un endpoint care declanșează rularea algoritmului de selecție și optimizare automată a modelelor AI, numit PSO-SA;
7) PipelineAPI- modulul cel mai complex, folosit pentru a crea un pipeline configurabil, așa cum este descris în secțiunea următoare;
8) LstmAPI – conține un endpoint pentru crearea și configurarea unui model LSTM;
9) AutoArimaAPI - conține un endpoint pentru crearea și configurarea algoritmului Auto-Arima;
10) EvaluateAPI – conține diferite funcții pentru evaluarea algoritmilor de clasificare (acuratețe, precizie, scor f1, matrice de confuzie etc.) și de regresie (MAE, RMSE etc.);

11) TimeSeriesEvaluationAPI – conține diferite funcții pentru evaluarea algoritmilor folosiți pentru analiza, procesarea seriilor de timp și de predicție a valorilor următoare.

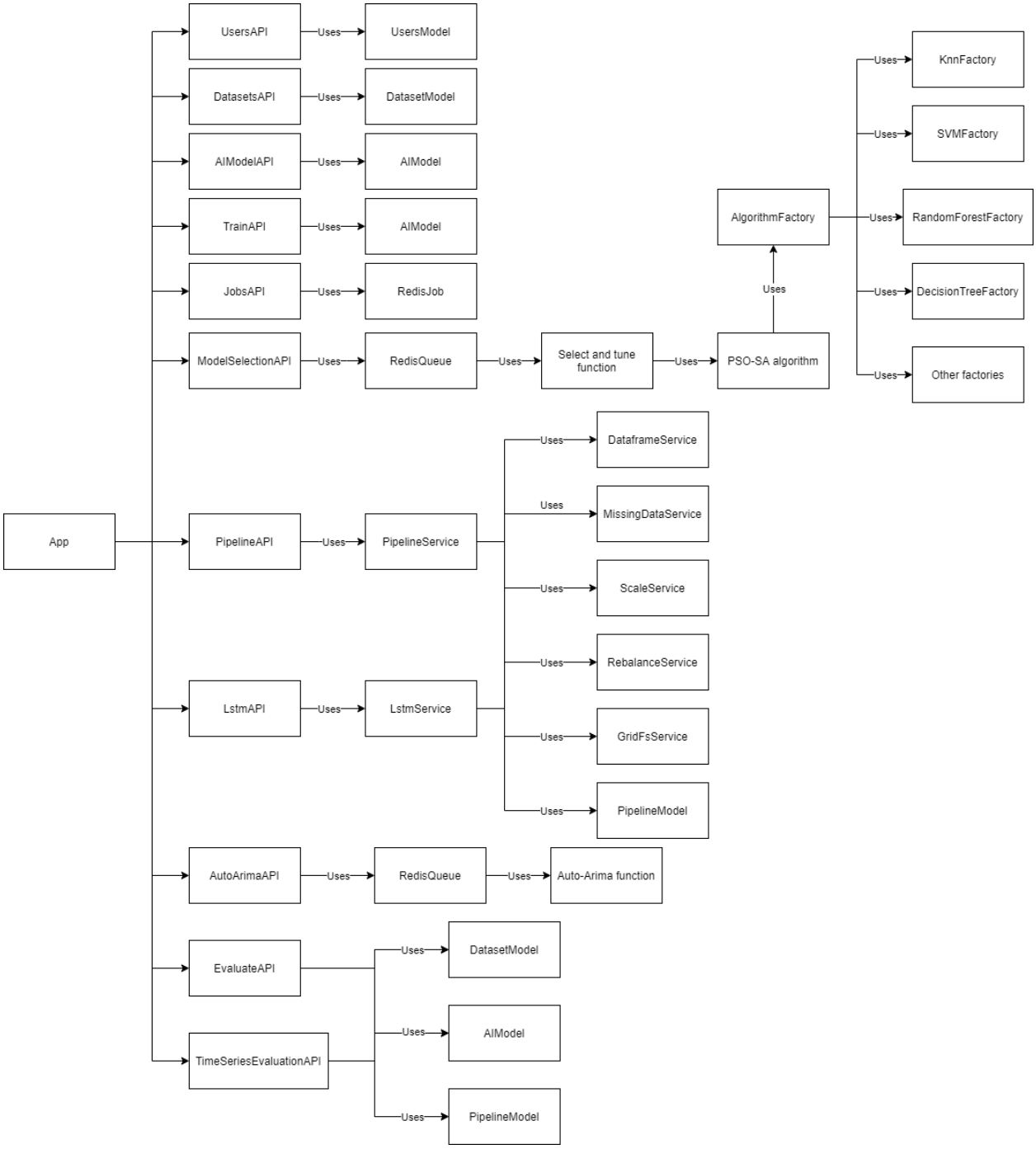


Fig. 29 Componentele serviciului AutomaticAI API

Pipeline dinamic

Pipeline-ul dinamic se configurează folosind un json de configurare. În mod obligatoriu, pipeline-ul trebuie să aibă un nume, trebuie să se specifice numele modelului care va fi salvat în urma rulării acestui pipeline, un nume de utilizator (astfel fiecare utilizator va avea acces doar la

pipeline-urile create de el), numele fișierului de intrare, folosit ca date de antrenare, cantitatea datelor folosite din acest fișier exprimat în procente, și elementele pipeline-ului. Proprietatea ”pipelineElements” e o listă care conține niște obiecte având ca proprietăți tipul algoritmului și parametrii. Ca să se poată crea o listă dinamică de parametri a fost utilizat un dicționar, unde cheia este numele parametrilor. Un exemplu pentru configurarea unui pipeline care conține un pas de scalare folosind algoritmul MinMax și un pas de re-echilibrarea datelor folosind algoritmul SMOTE (Chawla et al., 2002) este prezentat în Fig. 30.

```
{
    "name": "test-pipeline",
    "modelName": "test-model",
    "step": "first-step",
    "username": "test1",
    "trainDatasetName": "test.csv",
    "sampleSizeInPercentage": 100,
    "pipelineElement":
    [
        {
            "algorithmType": "Scale",
            "parameters":
            {
                "scaleType": "MinMax"
            }
        },
        {
            "algorithmType": "Rebalance",
            "parameters":
            {
                "rebalanceType": "SMOTE"
            }
        }
    ]
}
```

Fig. 30 Exemplu pentru configurarea pipeline-ului dinamic

Interfața grafică de tip web

Această interfață ajută în utilizarea mai ușoară a sistemului de procesare și analiză de date (Fig. 31). Setul de date de analizat poate fi încărcat în platformă prin drag-and-drop sau prin selecția fișierului în fereastra browse.

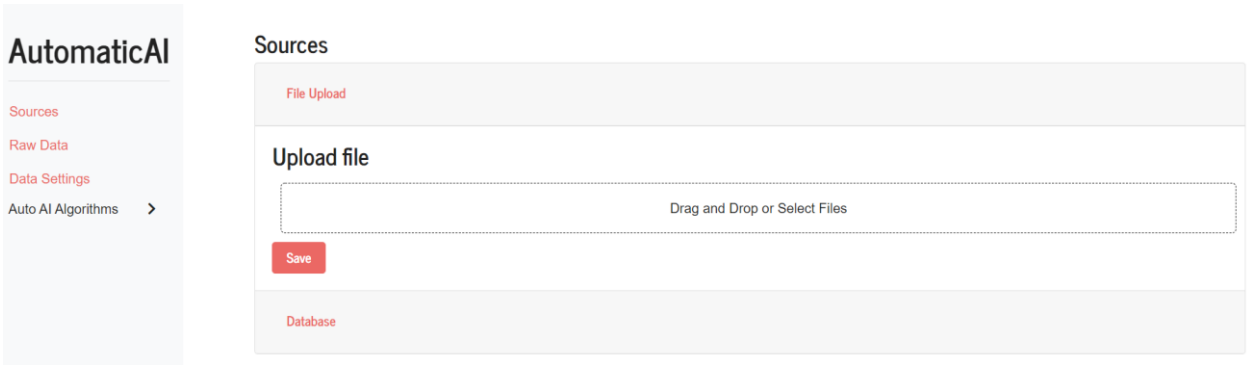


Fig. 31 Încărcarea fișierelor

În acest moment platforma suportă doar fișiere de tip .csv. După încărcarea fișierului, în secțiunea Raw Data se pot vizualiza toate fișierele încărcate de către un utilizator (Fig. 32).

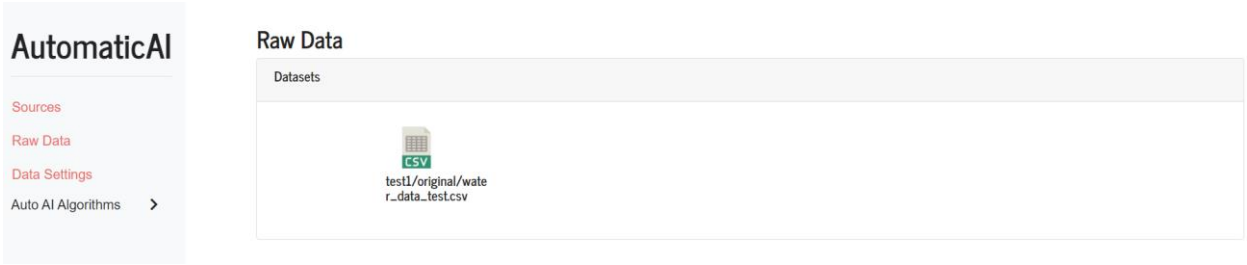


Fig. 32 Vizualizarea listei fișierelor încărcate

Pentru a specifica care este coloana țintă sau coloana care conține etichetele (în cazul problemelor de clasificare) există pagina de Data Settings, unde utilizatorul poate selecta coloana dorită, apăsând butonul numit Set as target (Fig. 33).

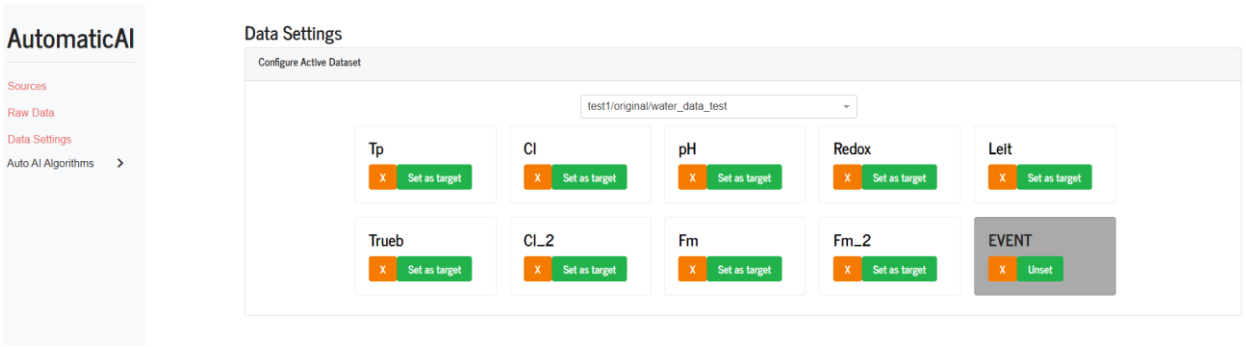


Fig. 33 Configurarea setului de date

După configurarea datelor de intrare, urmează crearea pipeline-ului dinamic. Primul pas este selecția datelor de intrare utilizând drop-down-ul din Fig. 34.

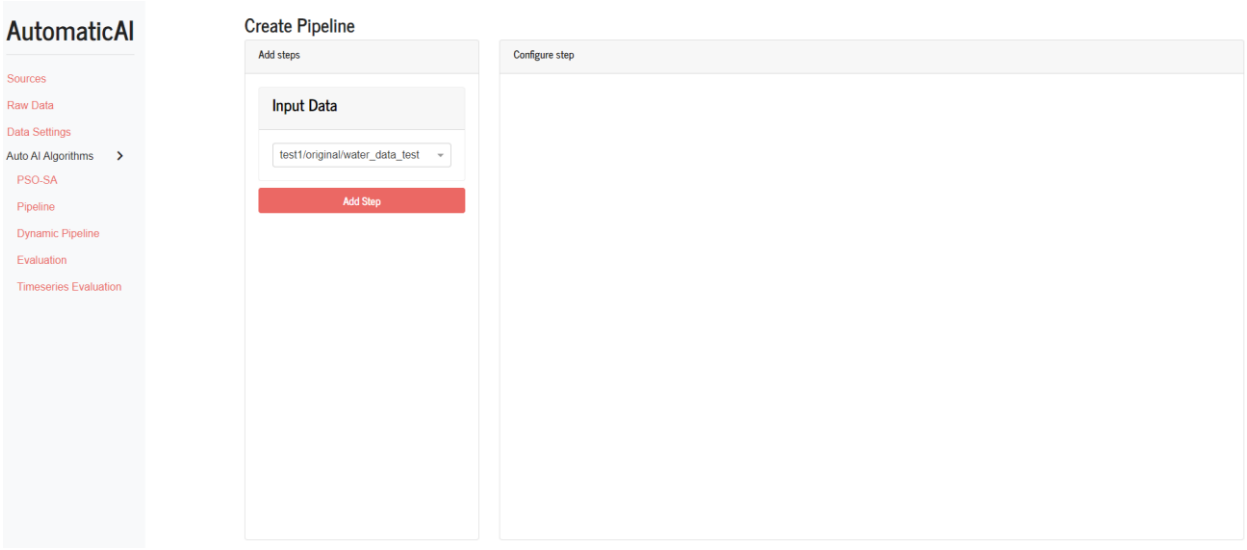


Fig. 34 Alegerea setului de date în configurarea pipeline-ului dinamic

Pentru a adăuga diferiți pași de pre-procesare, se poate apăsa butonul Add Step, apoi se alege categoria (cum ar fi transformarea datelor, scalare, normalizare etc.) și numele algoritmului (de exemplu MinMax, SMOTE, RUID etc.) (Fig. 35).

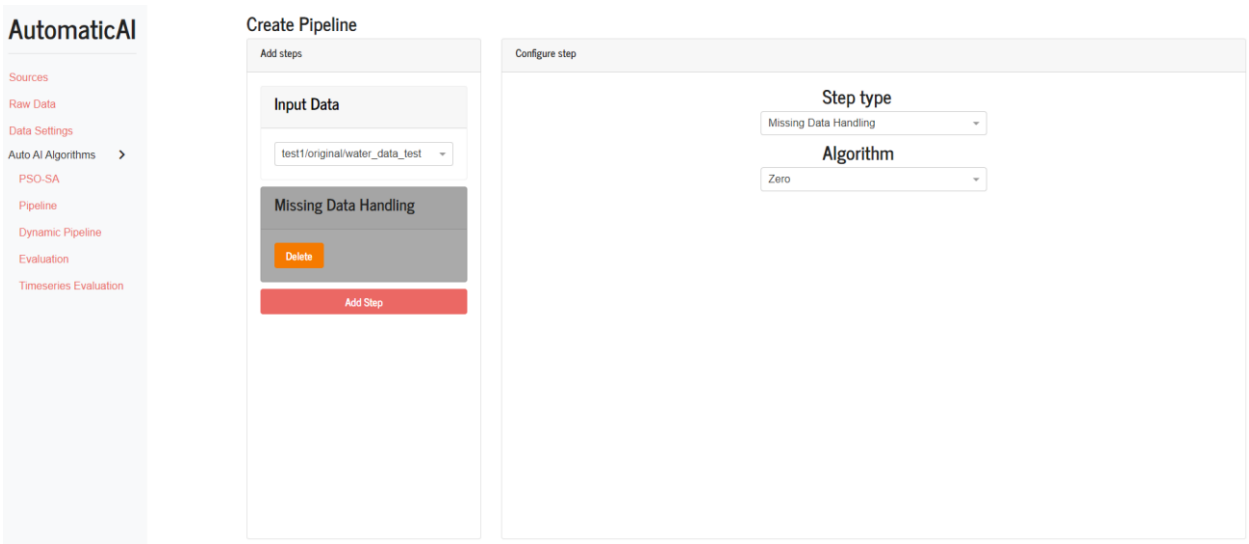


Fig. 35 Adăugarea unui pas de preprocesare în pipeline-ul dinamic

Ultimul pas este selecția algoritmului de analiză a datelor, cum ar fi algoritmi de clasificare, de analiză de serii de timp etc. Pentru fiecare algoritm se deschide un panou de control cu parametrii specifici algoritmului. Prin apăsarea butonului de Train începe procesul de antrenare, progresul fiind afișat folosind un progress bar (Fig. 36).

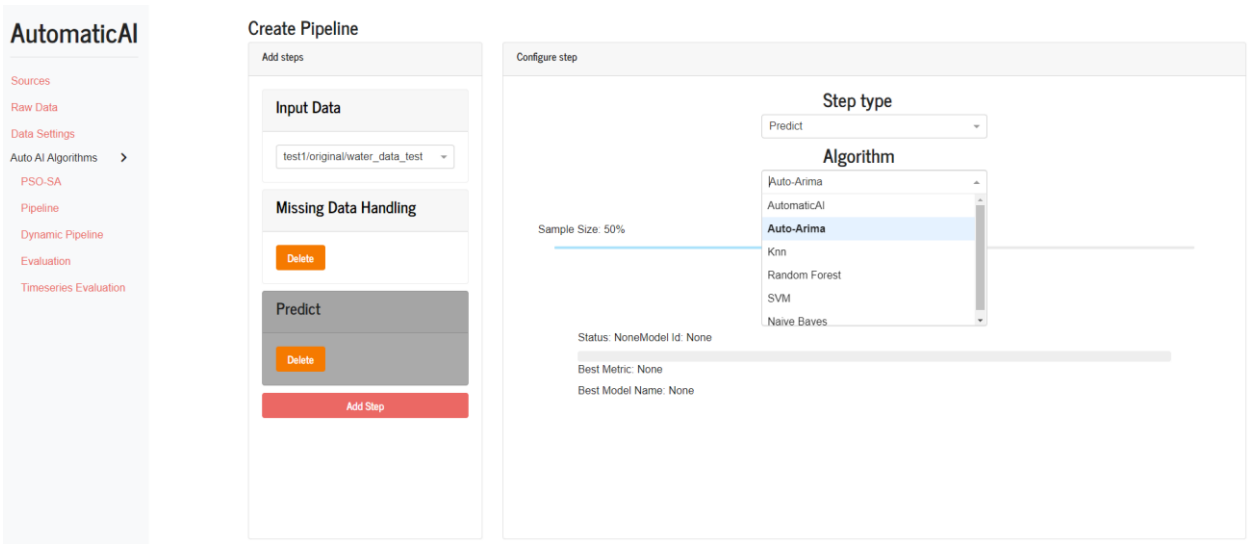


Fig. 36 Alegerea algoritmului de analiză de date

După antrenare, modelul poate fi aplicat pe date noi, pe date de test și automat se generează rapoarte și metrici de evaluare (cum ar fi precizie, acuratețe, matricea de confuzie etc.) (Fig. 37).

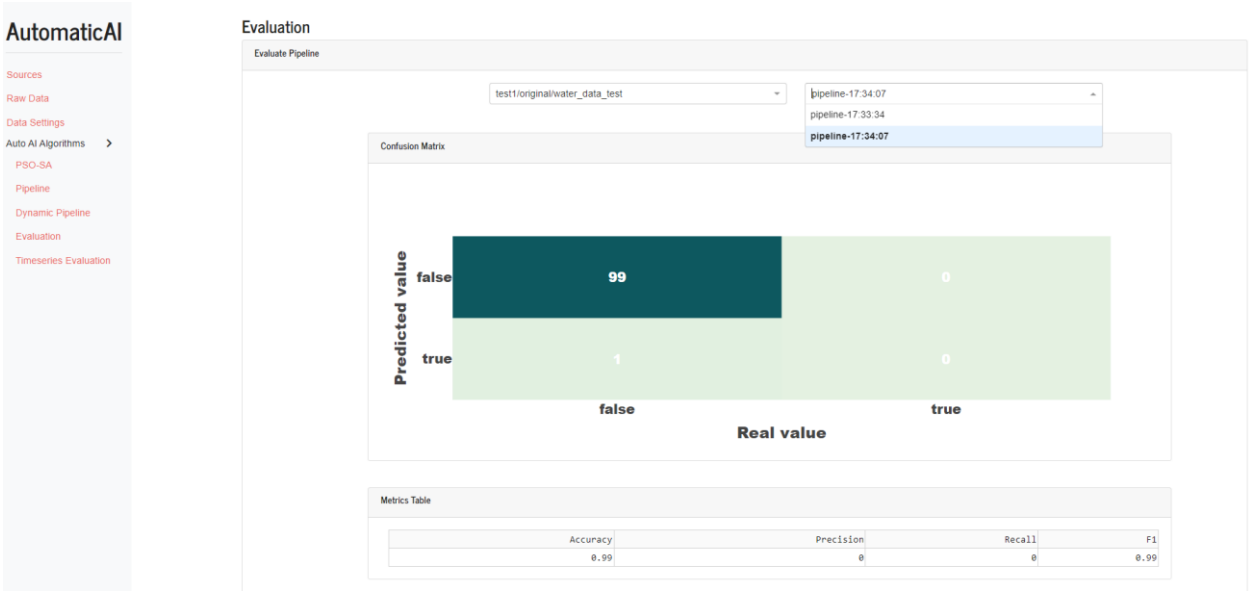


Fig. 37 Evaluarea pipeline-ului de clasificare

În cazul seriilor de timp, rezultatele sunt afișate folosind o diagramă de tip linechart (Fig. 38).

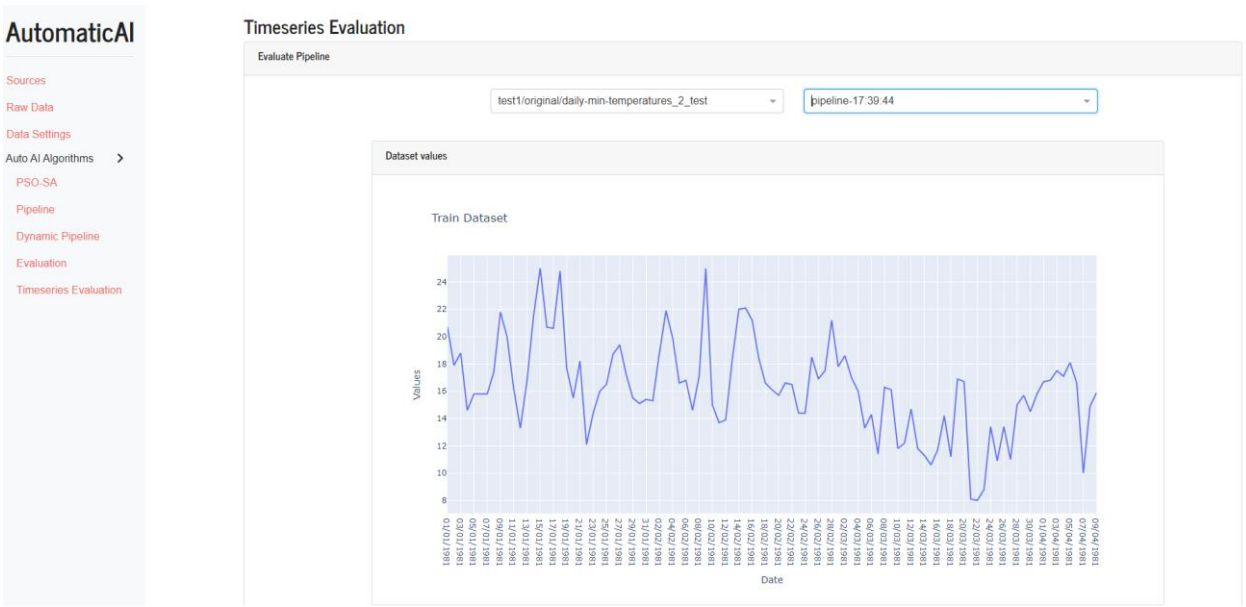


Fig. 38 Evaluarea pipeline-ului de procesare a seriilor de timp

4.2.2 Serviciul de analiză a datelor (UPB.A)

Serviciul de analiză a datelor este compus din următoarele componente la nivel de implementare:

- Implementare Python:
  - Script-uri Python;
  - Date preluate din fișiere CSV;
- Componente:
  - Proiect clusterizare;

  - Proiect clasificare;
- Biblioteci și framework-uri:
  - scikit-learn, Keras, Tensorflow;
  - Matplotlib.

Metode clusterizare

Deoarece datele reprezintă consumul mediu săptămânal pe oră, pe întreaga perioadă de timp, pentru fiecare nod de măsurare, a fost utilizată o metodă de clustering K-Means pentru a identifica tiparele medii săptămânale de consum.

Pentru a identifica numărul optim de clustere pentru setul de date disponibil, am aplicat metoda Elbow pe baza WCSS. Datele colectate de la senzorii IoT au fost stocate în fișiere de tip CSV, pentru a efectua cu ușurință procesarea acestora, cu scopul final de a oferi un rezultat cât mai apropiat de cel real.

Gruparea datelor ne-etichetate este efectuată folosind modulul *sklearn.cluster* din biblioteca scikit-learn. Fiecare algoritm de grupare vine în două variante: o clasă, care implementează metoda *fit* pentru antrenarea modelului și o funcție, care returnează etichetele identificate, corespunzătoare diferitelor clustere, ce pot fi găsite în atributul labels_.

Algoritmul K-Means grupează datele încercând să separe eșantioanele în n grupuri de varianță egală, minimizând WCSS. Acest algoritm necesită specificarea numărului de clustere. Este scalabil pentru un număr mare de date și a fost utilizat în multe domenii diferite.

Metode clasificare

Metodele de clasificare se împart în 2 clase, folosind biblioteci și framework-uri diferite.

Scikit-learn

În prima categorie intră arborii de decizie (DT) și Random Forest, framework-ul folosit pentru acestea fiind scikit-learn.

Arborii de decizie (DT) reprezintă o metodă de învățare supervizată neparametrică utilizată pentru clasificare și regresie. Scopul este de a crea un model care prezice valoarea unei variabile țintă prin învățarea unor reguli simple de decizie deduse din caracteristicile datelor.

Ca implementare, *DecisionTreeClassifier* este utilizat pentru clasificarea multi-clasă pe setul de date etichetat. Ca și în cazul altor clasificatoare, *DecisionTreeClassifier* primește ca intrare două matrici: X, de formă (n_samples, n_features) care conține datele de antrenare și Y, de formă (n_samples,), care conține etichetele clasei pentru datele de antrenare. După antrenarea modelului prin metoda *fit*, acesta este folosit pentru predicția claselor, folosind metoda *predict* [19].

Pentru a îmbunătăți robustețea, se folosesc metode ansamblu, prin care se combină predicțiile mai multor estimatori. O astfel de metodă este reprezentată de Random Forest (*RandomForestClassifier*). Fiecare arbore decizional din ansamblu este construit dintr-un eșantion extras din setul de antrenare. Algoritmul Random Forest realizează o variație redusă prin combinarea diverșilor arbori de decizie, rezultând astfel un model general mai bun. Dimensiunea

[19] https://scikit-learn.org/stable/modules/tree.html

eșantionului este controlată cu parametrul max_samples dacă bootstrap=True (implicit); în caz contrar, întregul set de date este folosit pentru a construi fiecare arbore.

În plus, la împărțirea fiecărui nod în timpul construcției unui arbore, cea mai bună împărțire este găsită fie din toate caracteristicile de intrare, fie dintr-un subset aleatoriu de dimensiune max_features. (Consultați instrucțiunile de reglare a parametrilor pentru mai multe detalii [20]).

TensorFlow. Keras

Ceilalți algoritmi folosiți se regăsesc în framework-ul Keras construit peste TensorFlow. În timp ce TensorFlow este un strat de infrastructură pentru programare diferențiabilă, care se ocupă de tensori, variabile și gradienți, Keras este o interfață cu utilizatorul pentru învățare profundă, care se ocupă de straturi, modele, optimizatori, funcții de pierdere, metrici și multe altele.

Keras servește ca API de nivel înalt pentru TensorFlow. Clasa Layer este abstractizarea fundamentală în Keras. Un Layer încapsulează o stare (ponderi) și unele calcule (definite în metoda de apelare). În primă fază este construit modelul rețelei neuronale:

```
# add input layer
model.add(Dense(hsize, input_dim=input_dim, activation='relu'))

# add dropout to hidden layers to prevent overfitting
model.add(Dropout(0.5))
model.add(Dense(hsize, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(hsize, activation='relu'))
model.add(Dropout(0.5))

# add output layer
model.add(Dense(output_dim, activation=activation_fn))
```

Apoi modelul este antrenat pe setul de date, distribuit în date de antrenare și de validare:

```
early_stopping_monitor = EarlyStopping(patience=50)

# fit the keras model on the dataset
h = model.fit(X, y, validation_split=0.2, epochs=epochs,
 batch_size=batch_size, verbose=1, callbacks=[early_stopping_monitor])
```

Acuratețea modelului este evaluată pe setul de date de test:

```
_, accuracy = model.evaluate(X, y, batch_size=batch_size, verbose=1)
```

Predicția se realizează pe baza datelor de intrare, care pot fi neetichetate:

[20] https://scikit-learn.org/stable/modules/ensemble.html#random-forest-parameters

```
predictions = model.predict(X)
```

Vizualizare

Matplotlib

Matplotlib este o bibliotecă de plotare pentru limbajul de programare Python, ce oferă un API orientat pe obiecte pentru încorporarea graficelor în aplicații de vizualizare a datelor. Pyplot este un modul Matplotlib care oferă o interfață asemănătoare MATLAB. Matplotlib este conceput pentru a fi la fel de utilizabil ca MATLAB, cu posibilitatea de a folosi Python și avantajul de a fi gratuit și open-source. Matplotlib este folosit în cadrul serviciului de analiză a datelor pentru:

- reprezentarea seriilor de timp;
- vizualizarea datelor procesate în coordonate 2D;
- vizualizarea datelor procesate sub formă de hartă a culorilor:
- analiză comparativă prin grafic cu bare.

Integrare

În urma integrării serviciilor oferite de UPB și UTCN, partea legată de imputarea datelor și scalarea acestora nu va mai fi realizată la nivelul UPB, ci va fi preluată de la serviciul AutomaticAI, care, pe lângă imputarea datelor, se mai ocupă și de detecția anomaliilor și de predicțiile ce trebuie făcute.

4.3 Crowdsensing (CS)

Platforma de crowdsensing este implementată de o aplicație mobilă, bazată pe o platformă de explorare urbană prin gamificare, dezvoltată în colaborare cu LEPLACE GLOBAL, un startup românesc din domeniul aplicațiilor de realitate augmentată. Participanții / cetățenii pot raporta probleme legate de utilitățile de apă și le pot plasa pe hartă. Gamificarea este folosită pentru a crește nivelul de participare și interacțiune cu mediul, în timp ce design-ul fluid și aspectul vizual asigură un mediu de joc atractiv, cu implicații în lumea reală. Pagina principală este prezentată în Fig. 39, cu locația problemelor și a participanților conform configurației experimentale.

Deși există diferite tipuri de probleme care pot fi găsite în infrastructura urbană de apă, vizualizarea hărții dezvăluie probleme din apropiere, așa cum sunt raportate de participanți: inundații, probleme de alimentare cu apă (e.g. scurgeri sau întreruperi ale serviciului), poluare și alte pericole. Locațiile participanților sunt afișate pe hartă în scop experimental, în timp ce în aplicațiile din lumea reală, colectarea datelor de locație în timp real poate fi interzisă de legile de confidențialitate, conform cazului de utilizare.

Modelul crowdsensing prezentat în etapa de proiectare poate fi utilizat pentru a defini alocarea sarcinilor pentru fiecare participant și pentru a oferi mecanismul de stimulare. Problemele raportate pot fi înregistrate în blockchain, pentru un nivel sporit de încredere pentru părțile interesate, prin stocarea securizată a datelor și transparență.

Fig. 39 Aplicația Watergame pentru raportarea incidentelor

Componentele propuse abordează problemele existente în aplicațiile mobile de crowdsensing, precum problemele legate de motivație, încredere și acoperire, din trei perspective diferite. Prin urmare, utilitatea globală a platformei, adică acoperirea și costul asociat cu raportările, este influențată de: (1) modelul crowdsensing; (2) un design de joc atractiv pentru a crește nivelul de participare; (3) nivelul de încredere perceput, asigurat de tehnologia blockchain.

Pentru a asigura performanța și scalabilitatea soluției, server-ul aplicației este găzduit în cloud (AWS) și au fost proiectate o serie de componente și optimizări de arhitectură precum integrarea unei memorii cache REDIS, optimizări la nivel de bază de date (indexarea tabelelor, refactorizarea bazei de date în funcție de cazurile de utilizare), optimizări în transferul de date (stocare temporară, compactarea structurilor de date arborescente). Aceste optimizări precum și integrarea cu platforma de explorare urbană vor fi prezentate mai pe larg în etapa de integrare.

4.4 Blockchain (B)

4.4.1 Aplicatie Web – Oferta de moneda inițială

Aplicația web dezvoltată permite clienților achiziționarea de criptomonede înainte de lansarea soluției, așa cum este prezentat în Fig. 40. Monedele achiziționate vor putea fi folosite pentru încheierea de contracte cu beneficii suplimentare sau ca investiție financiară, datorită faptului că în timp valoarea monedelor se va aprecia, ulterior acestea putând fi vândute, prin intermediul soluțiilor de exchange ce vor adopta moneda WGT.

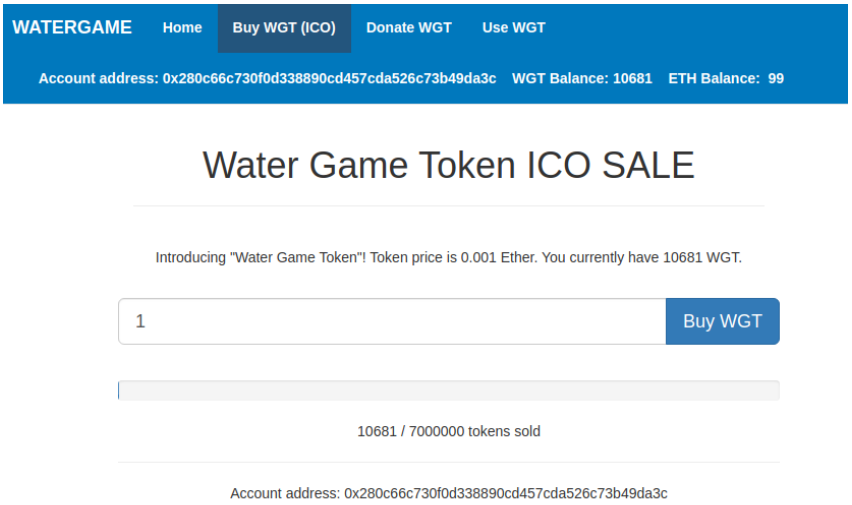


Fig. 40 Interfața sistemului blockchain. ICO

4.4.2 Aplicație Web – Transfer criptomonedă

Aplicația web dezvoltată permite clienților sa transfere o cantitate de criptomonedă din portofelul electronic propriu la adresa unui prieten așa cum este prezentat în Fig. 41. Astfel soluția va crea o comunitate de utilizatori ce interacționează între ei prin token.

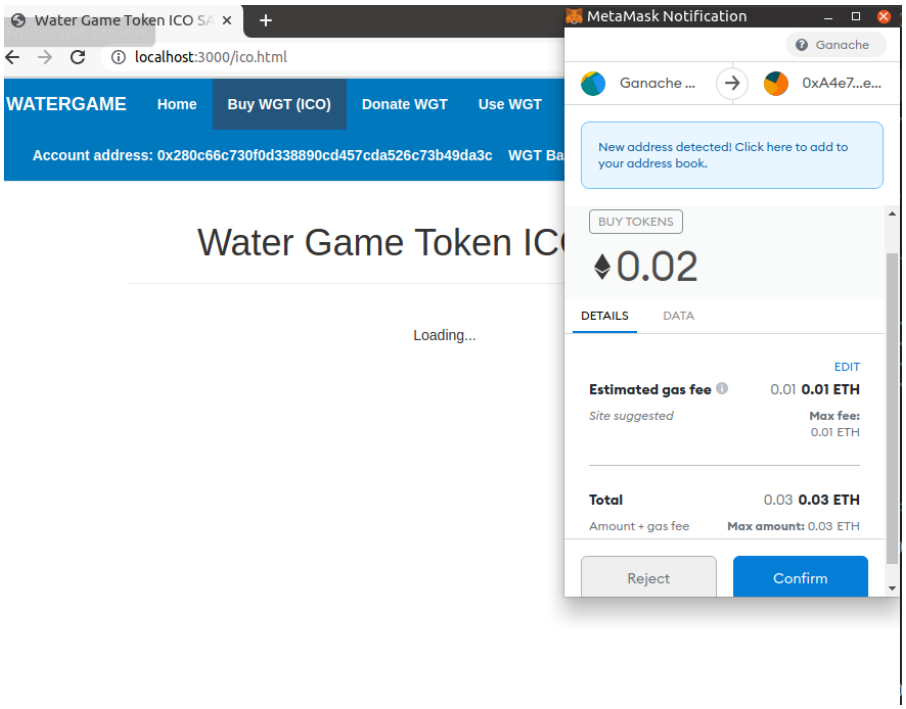


Fig. 41 Interfața sistemului Blockchain. Integrare cu Ethereum

În Fig. 41 este prezentat modul în care se realizează o tranzacție cu criptomoneda WGT folosind MetaMask, împreună cu mesajul de succes aferent acestei tranzacții (Fig. 42).

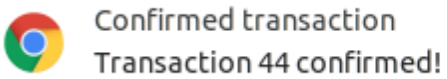


Fig. 42 Interfața sistemului Blockchain. Confirmarea tranzacției.

4.4.3 Funcționalitate REST

Pentru integrarea tuturor componentelor sistemului cu zona de blockchain este necesară implementarea unei modalități de comunicare bazate pe un REST API prin care se vor putea declanșa diverse acțiuni / tranzacții în funcție de output-ul celorlalte sisteme (de ex.: acordarea de recompense).

Astfel, în următoarea etapă, această componentă va permite inițializarea unei tranzacții prin apeluri REST care să transfere o cantitate de criptomonede din portofelul electronic al furnizorului (sau administratorului) către un utilizator drept recompensă (de ex.: realizarea unui consum optim pentru o lună sau sesizarea unui incident valid în aplicația de crowdsensing).

Bibliografie

Alzen, J. L., Langdon, L. S., & Otero, V. K. (2018). A logistic regression investigation of the relationship between the Learning Assistant model and failure rates in introductory STEM courses. International journal of STEM education, 5(1), 1-12, doi 10.1186/s40594-018-0152-1.

Amazon Web Services. (2021). Overview of Amazon Web Services AWS Whitepaper. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/aws-overview.pdf

Arsene, D., Predescu, A., Truică, C.-O., Apostol, E.-S., Mocanu, M., & Chiru, C. (2021). Profiling consumers in a water distribution network using K-Means clustering and multiple pre-processing methods. 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 1–6. https://doi.org/10.1109/ECAI52376.2021.9515193

Austin Water. (2021), Austin Water - Residential Water Consumption, https://data.austintexas.gov/Utilitiesand-City-Services/Austin-Water-Residential-WaterConsumption/sxk7-7k6z

Bell, C. (2016). MySQL for the Internet of Things. https://doi.org/10.1007/978-1-4842-1293-6

Bento, A. (2018). IoT: NodeMCU 12e X Arduino Uno, Results of an experimental and comparative survey. 6, 45–56.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32, DOI: 10.1023/A:1010933404324.

Chandrasekaran, S., Friese, M., Stork, J., Rebolledo, M. & Bartz-Beielstein, T. (2017) Gecco Challange 2017, https://www.spotseven.de/gecco/gecco-challenge/gecco-challenge-2017/

Chatzigeorgakidis, G. (2018). Household Water Consumption (Average), HELIX, 2018. https://data.hellenicdataservice.gr/dataset/236a0883-91b6-4f57-af9a-8d54ae7b154e (accessed Apr. 20, 2021).

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357, DOI: 10.1613/jair.953.

Chen, S., Tang, X., Wang, H., Zhao, H., & Guo, M. (2016). Towards Scalable and Reliable In-Memory Storage System: A Case Study with Redis. https://doi.org/10.1109/TrustCom.2016.0255

ConsenSys Software Inc. (2021). Truffle | Overview | Documentation | Truffle Suite. https://www.trufflesuite.com/docs/truffle/overview

Cunningham, P., & Delany, S. J. (2007). k-Nearest neighbour classifiers. Mult Classif Syst., 54, DOI: 10.1145/3459665.

Czako, Z., Sebestyen, G., & Hangan, A. (2021). AutomaticAI-A Hybrid Approach for Automatic Artificial Intelligence Algorithm Selection and Hyperparameter Tuning. Expert Systems with Applications, 115225, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2021.115225.

Evgeniou, T., & Pontil, M. (1999, July). Support vector machines: Theory and applications. In Advanced Course on Artificial Intelligence (pp. 249-257). Springer, Berlin, Heidelberg, DOI: 10.1007/3-540-44673-7_12.

ethereum.org. (2021). Token ERC-20 Standard. https://ethereum.org/ro/developers/docs/standards/tokens/erc-20/

Ethereum Revision a57dd3c5. (2016). Getting Started - web3.js 1.0.0 documentation. https://web3js.readthedocs.io/en/v1.5.2/getting-started.html

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63(1), 3-42, DOI: 10.1007/s10994-006-6226-1.

Google. (2021a). Firebase Realtime Database. https://firebase.google.com/docs/database

Google. (2021b). Vehicle Routing Problem | OR-Tools | Google Developers. https://developers.google.com/optimization/routing/vrp

Hanum, F., Hartono, A. P., & Bakhtiar, T. (2018). On the multiple depots vehicle routing problem with heterogeneous fleet capacity and velocity. IOP Conference Series: Materials Science and Engineering, 332, 12052. https://doi.org/10.1088/1757-899x/332/1/012052

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

IBM. (2021). Getting to know MQTT. https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/

Karray, F., Triki, M., Jmal, W., Abid, M., & Obeid, A. (2018). WiRoTip: an IoT-based Wireless Sensor Network for Water Pipeline Monitoring. International Journal of Electrical and Computer Engineering (IJECE), 8, 3250. https://doi.org/10.11591/ijece.v8i5.pp3250-3258

Khasawneh, M., Kajman, I., Alkhudaidy, R., & Althubyani, A. (2014). A Survey on Wi-Fi Protocols: WPA and WPA2 (Vol. 420). https://doi.org/10.1007/978-3-642-54525-2_44

Khawas, C., & Shah, P. (2018). Application of Firebase in Android App Development-A Study. International Journal of Computer Applications, 179, 49–53. https://doi.org/10.5120/ijca2018917200

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. Complexity, 2020.

Makowski, L., & Ostroy, J. M. (1987). Vickrey-Clarke-Groves mechanisms and perfect competition. Journal of Economic Theory, 42(2), 244–261. https://doi.org/10.1016/0022-0531(87)90087-1

Masse, M. (2011). REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces. " O'Reilly Media, Inc."

MetaMask. (2021). MetaMask - A crypto wallet & gateway to blockchain apps. https://metamask.io/index.html

Mishra, B., & Kertesz, A. (2020). The Use of MQTT in M2M and IoT Systems: A Survey. IEEE Access, 8, 201071–201086. https://doi.org/10.1109/ACCESS.2020.3035849

Moroney, L. (2017). The Firebase Realtime Database (pp. 51–71). https://doi.org/10.1007/978-1-4842-2943-9_3

Murtagh, F. (1991). Multilayer perceptrons for classification and regression. Neurocomputing, 2(5-6), 183-197, ISSN 0925-2312, https://doi.org/10.1016/0925-2312(91)90023-5.

Neto, A. (2020). Developing for Ethereum, Test networks, Standalone Nodes, Solidity, and the Remix IDE. Medium. https://medium.com/@arlindont/developing-for-ethereum-9a0551f2b9d9

NodeMcu Team. (2018). NodeMcu -- An open-source firmware based on ESP8266 wifi-soc. https://www.nodemcu.com/index_en.html

Parihar, Y. S. (2019). Internet of Things and Nodemcu A review of use of Nodemcu ESP8266 in IoT products. 6, 1085.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Portet, S. (2020). A primer on model selection using the Akaike Information Criterion. Infectious Disease Modelling, 5, 111-128, doi 10.1016/j.idm.2019.12.010.

Predescu, A., & Mocanu, M. (2019). Increasing Collaboration and Participation Through Serious Gaming for Improving the Quality of Service in Urban Water Infrastructure. In Lecture Notes in Business Information Processing: Vol. 373 LNBIP. https://doi.org/10.1007/978-3-030-36691-9_49

Predescu, A., Arsene, D., Pahonțu, B., Mocanu, M., & Chiru, C. (2021). A Serious Gaming Approach for Crowdsensing in Urban Water Infrastructure with Blockchain Support. Applied Sciences, 11(4). https://doi.org/10.3390/app11041449

Prusa, J., Khoshgoftaar, T. M., Dittman, D. J., & Napolitano, A. (2015, August). Using random undersampling to alleviate class imbalance on tweet sentiment data. In 2015 IEEE international conference on information reuse and integration (pp. 197-202). DOI: 10.1109/IRI.2015.39.

scikit-learn developers (BSD License). (n.d.). scikit-learn. http://scikit-learn.org/stable/

Sessa, P. G., Walton, N., & Kamgarpour, M. (2017). Exploring the Vickrey-Clarke-Groves Mechanism for Electricity Markets. IFAC-PapersOnLine, 50(1), 189–194. https://doi.org/10.1016/j.ifacol.2017.08.032

Sherstinsky, A. (2020). Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306, DOI: 10.1016/j.physd.2019.132306.

Tao, X., & Song, W. (2018). Efficient Path Planning and Truthful Incentive Mechanism Design for Mobile Crowdsensing. Sensors (Basel, Switzerland), 18(12), 4408. https://doi.org/10.3390/s18124408

Topîrceanu, A., & Grosseck, G. (2017). Decision tree learning used for the classification of student archetypes in online courses. Procedia Computer Science, 112, 51-60, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2017.08.021.

Wingerath, W. (2017). A Real-Time Database Survey: The Architecture of Meteor, RethinkDB, Parse & Firebase (Medium (ed.)).

Xingjian, S. H. I., Chen, Z., Wang, H., Yeung, D. Y., Wong, W. K., & Woo, W. C. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In Advances in neural information processing systems (pp. 802-810).

Yermal, L., & Balasubramanian, P. (2017, December). Application of auto arima model for forecasting returns on minute wise amalgamated data in nse. In 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC) (pp. 1-5). IEEE