Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.42:004.738.12
DOI: https://doi.org/10.24867/15BE16Radovic

UPOTREBA WOLKABOUT PLATFORME ZA REALIZACIJU KUĆNOG IoT

USE OF WOLKABOUT PLATFORM FOR REALIZATION OF HOME IoT

Luka Radović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *Zadatak rada predstavlja aplikaciju koja u saradnji sa Wolkabout platformom realizuje Home automation sistem. Serverski deo aplikacije je realizovan u programskom jeziku Java (Spring), a klijentski deo u najnovijoj verziji Angular okruženja. Aplikacija omogućava korišćenje Wolkabout platforme za CRUD operacije nad uređajima, senzorima i aktuatorima, ali predstavlja i brokera za slanje podataka (očitanih vrednosti senzora povezanog na Raspberry Pi) preko MQTT protokola na Wolkabout platformu koja je prikazuje stanje svega i menja stanja uređaja pod jasno definisanim pravilima. Na osnovu ranije očitanih temperaturnih vrednosti senzora, aplikacija obezbeđuje predikciju trenutne temperature.*

Ključne reči: *Internet of Things, Home automation, Wolkabout, Raspberry Pi*

Abstract – *This paper describes one implementation of Wolkabout platform for IoT Home automation system, with devices, sensors and actuators. Frontend is implemented in Angular and backend is implemented in Spring (Java). Application uses a real sensor for temperature and humidity connected to Raspberry Pi. Based on the previously read temperature values of the sensor, the application provides a prediction of the current temperature.*

Keywords: *Internet of things, Home automation, Wolkabout, Raspberry Pi*

1. UVOD

Zadatak rada predstavlja razvoj aplikacije koja u saradnji sa Internet of Things [1] Wolkabout platformom realizuje Home automation [2] sistem.

Internet of Things ili skraćeno IoT predstavlja tehnologiju za umrežavanje više uređaja različitog tipa, koji šalju podatke, na osnovu njih sprovode određene zadatke i to samostalno ili uz malu ljudsku intervenciju, kao i "razgovaraju" preko raznih komunikacionih protokola.

S obzirom da se radi o veb aplikaciji, ona se sastoji iz dva dela: klijentskog koji je implementiran u Angular [3] okruženju, kao i serverskog, za koji sam odabrao Java [4] programski jezik (Spring [5] okruženje). Aplikacija (LR Simulator) omogućava korišćenje Wolkabout platforme za CRUD [6] operacije nad uređajima, senzorima i aktuatorima, ali predstavlja i brokera za slanje podataka preko MQTT [7] protokola na Wolkabout platformu. Platforma služi za kreiranje uređaja, senzora, aktuatora, prikazivanje njihovih stanja, kao i za konfiguraciju pravila za automatizovan rad platforme nad uređajima, senzorima i aktuatorima. Podaci se čitaju iz Sqlite [13] baze podataka, a nastaju uz pomoć Python [14] programa koji očitava vrednosti DHT22 senzora (temperature i vlažnosti vazduha) koji je nakačen na Raspberry Pi 4 [15]. Takođe, moguće je i predvideti trenutnu temperaturu na osnovu ranije očitanih vrednosti senzora.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Milan Vidaković, red. prof.

2. OPIS KORIŠĆENIH TEHNOLOGIJA

2.1. Spring okruženje

Trenutno najpopularnije okruženje za razvoj Java veb aplikacija je Spring. Dependency injection [8] mehanizam za povezivanje objekata je jedna od njegovih glavnih karakteristika.

REST (Representational State Transfer) [9] je stil softverske arhitekture koji obezbeđuje standarde i olakšava komunikaciju sistemima na vebu. Glavne karakteristike su mu što je stateless (serveri koji ga implementiraju ne moraju ništa da znaju o klijentu, jer se on zasniva na postojanju resursa i uniformom upravljanju njima putem skupa predefinisanih operacija) i što je odvojena totalno implementacija na klijentu i serveru, tako da obe mogu biti menjane stalno, bez posledica, sve dok šalju poruke u dogovorenom formatu. Zahvaljujući REST – u, različiti klijenti mogu da "gađaju" iste servise, da koriste iste operacije i dobijaju iste odgovore.

Resursi su "imenice" veba - predstavljaju podatke ili funkcionalnosti identifikovane jedinstvenim identifikatorom (URI – Uniform Resource Identifier). Isti resurs može biti predstavljen u različitim formatima.

Komunikacija između klijenta i servisa u REST – u se zasniva na slanju request – a i response – a. Request treba da se sastoji od HTTP [10] metode (GET, POST, PUT, DELETE), zaglavlja koja daje više informacija o samom zahtevu, putanje do REST servisa i, po potrebi, telo poruke koja sadrži neke podatke. Zahvaljujući REST – u ti podaci mogu biti u raznim formatima, neki od njih su: image/png, audio/wav, application/json...

Spring Data JPA [11] (Java Persitence API) uvodi koncept repozitorijuma koji izbacuje potrebu za pisanjem ponavljajućeg koda. Repozitorijum je u stvari interfejs, kojim se upravlja podacima. Nije potrebno praviti klasu koja implementira interfejs, samim njegovim postojanjem je moguće koristiti operacije predviđene nasleđenim repozitorijumskim interfejsom (primer metode –

findById, koja pretražuje bazu po id – ju i vraća traženi objekat ili null vrednost ukoliko ne postoji). Postoje različiti tipovi repozitorijuma: CrudRepository, PagingAndSortingRepository, kao i JpaRepository koji sadrži standardne CRUD operacije, paginaciju i sortiranje, kao i mogućnost definisanja nestandardnih operacija nad podacima korišćenjem query anotacije napisanih u JPQL (Java Persistence Query Language). I dalje nema potrebe za implementacijom metode, samo je potrebno da se anotira željenim upitom. JpaRepository može biti bilo koja implementacija JPA. U mom projektu je to Hibernate [12], dok je za konkretan Data Source korišćena eksterna Sqlite baza podataka. Hibernate je okruženje koje pruža mogućnost mapiranja objektno orijentisanog modela na relacionu bazu podataka. Njegova glavna uloga je mapiranje javinih klasa na tabele u bazi.

2.2. Angular okruženje

Angular je okruženje za razvoj klijentskih aplikacija, koje su tipa Single Page Application, sastoje se iz komponenti koje se menjaju. Svaka komponenta je zadužena za deo funkcionalnosti. Rutiranje omogućava prelazak između njih.

Komponente objedinjuje element prikaza (HTML template), stilove koji se primenjuju na HTML i aplikativnu logiku (TS klasu) koja ga kontroliše. Nakon kreiranja komponente, ona mora biti uvrštena u deklaraciju NgModule – a. On predstavlja korenski modul aplikacije i obavezan je u svakoj.

S obzirom da svaka komponenta ima svoj prikaz, potrebno je imati navigaciju među njima, za tu svrhu postoji rutiranje, gde mi određujemo putanju i komponentu (prikaz) koji se otvara (primer: {path: "devices", component: DeviceComponent}).

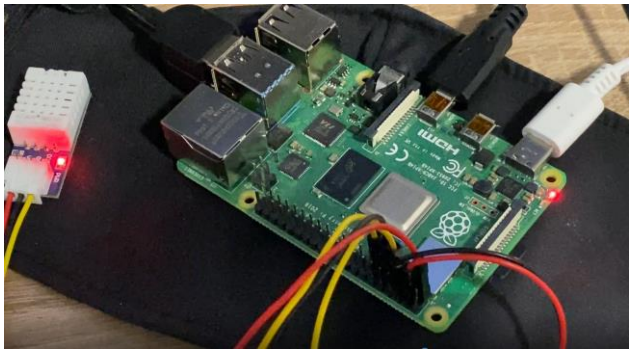
Još jedna od bitnih stvari Angular – a su servisi koji su spona između klijenta i servera. Njihova komunikacija se vrši preko Http protokola (Http Client biblioteke).

2.3. Raspberry Pi i DHT22

Raspberry Pi je mali kompjuter, veličine kreditne kartice koji može stati u džep. Cena mu je pristupačna, lako je prenosiv i može da se priključi na TV/monitor i da koristi standardnu tastaturu i miš. Nastao je u Velikoj Britaniji sa idejom da ljudima svih godina omogući da uče da programiraju na lak način. Zahvaljujući svojim mogućnostima, za kratko vreme postaje vrlo popularan te se ubacuje u različite grane privrede, jedna od njih je i IoT (Home automation).

Za ovaj projekat je korišćen Raspberry Pi 4 Model B sa 4 GB RAM - a. Kao što vidimo na slici 1, opremljen je sa: USB, micro HDMI, USB - C, ethernet, audio, 40 GPIO [16] pin - ova, WiFi - jem, Bluethooth - om, microSD karticom i tako dalje.

DHT22 je digitalni senzor temperature i vlažnosti vazduha (slika 1). Mana mu je što može da očitava vrednosti jednom u 2 sekunde, ne može u manjem intervalu. Ima 3 žice za povezivanje - power, digital out i ground, te se lako povezuje na Raspberry Pi. U mom slučaju, power žica je nakačena na 3.3v (1. pin), ground žica na ground (9. pin), a digital out na GPIO4 (7. pin). DHT22 očitava temperature od -40 do 80 stepeni Celzijusa, sa greškom od pola stepena, a od 0 do 100 procenata vlažnosti vazduha sa greškom od 2 do 5%.



Slika 1. *Raspberry Pi i DHT22*

2.4. Python

Python je jedan od najpopularnijih programskih jezika opšte namene. Jako je čitljiv zbog svoje sintakse, samim tim lako se uči i održavanje nije toliko skupo. Podržava korišćenje modula i paketa, te podstiče modularnost i ponovnu primenu istog koda. Pronalaženje i ispravljanje grešaka, popularnih "bugova", je jednostavno, ne postoji "segmentation fault", već kad interpreter pronađe grešku, baci izuzetak. Sam "debager" je napisan u Python - u.

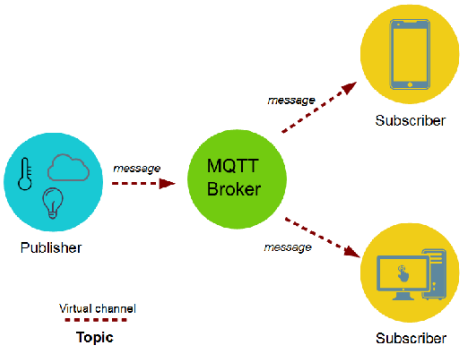
U projektu, Python je korišćen za rad sa senzorom, za dobijanje očitanih vrednosti i njihovih upisivanja u Sqlite bazu podataka. Za samo čitanje podataka mi je pomogla Adafruit_DHT [17] biblioteka i to svojom metodom read_retry, koja od parametara prima vrstu senzora i broj GPIO pin - a. Primer očitavanja vlažnosti vazduha i temperature upotrebom pomenute metode: h,t = dht.read_retry(dht.DHT22, DHT).

Python sam, takođe, koristio za predikciju trenutne temperature na osnovu ranije očitanih vrednosti senzora. Zahvaljujući klasi SensorData, imao sam sve što mi je potrebno za učenje, a to je timestamp i value - kada i šta je senzor očitao. Za rad sa bazom podataka, korišćena je biblioteka sqlite3. Predikciju sam realizovao pomoću polinomske regresije [18] i biblioteke sklearn, a da bih mogao da napravim REST servis, bila mi je potrebna flask biblioteka.

2.5. MQTT protokol

MQTT predstavlja publish/subscribe, extremno jednostavan i lak protokol za poruke, dizajniran za ograničene uređaje, visoka kašnjenja i ne toliko pouzdane mreže. Dizajnerski principi su da smanji količinu podataka koja se prenosi i resurse koje uređaji moraju da imaju, a takođe, pokušava i da osigura sigurnost dostave informacije.

Takvi principi su idealni za M2M ("machine-to-machine") ili "Internet of Things" svet pun uređaja, a i mobilnih aplikacija gde je količina podataka koja može da se prenese vrlo bitna, kao i trajanje baterije. Kao što vidimo na slici 2, MQTT protokol ima publishera koji šalje/objavljuje poruku na neku temu (Topic), a broker onda ima zadatak da je prosledi svim subscriberima te teme.

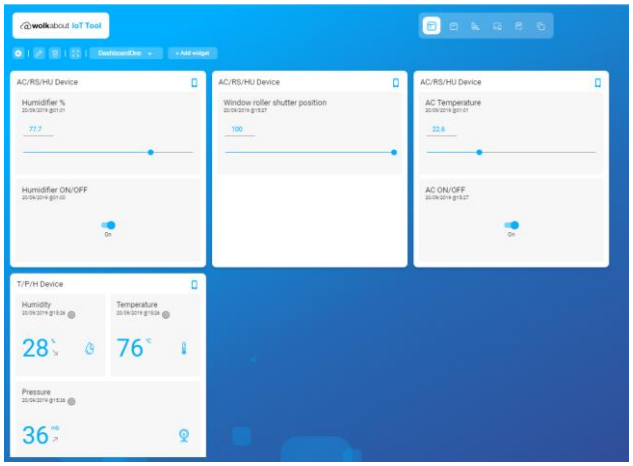


Slika 2. *MQTT protokol*

2.6. Wolkabout platforma

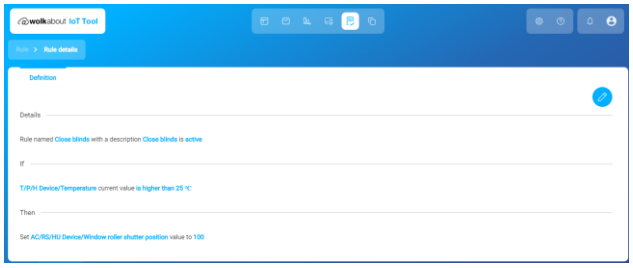
Na samoj Wolkabout platformi koja se nalazi na adresi (https://demo.wolkabout.com) korisniku je omogućeno da iskonfiguriše Dashboardove, kao i widgete i vidi stanje svojih uređaja, senzora i aktuatora.

Na slici 3 vidimo dva uređaja, jedan koji sadrži sve aktuatore i trenutne njihove vrednosti, kao i drugi koji sadrži sve senzore i vrši očitavanja, tačnije dobija podatke sa LR Simulatora.



Slika 3. *Wolkabout platforma*

Korisniku je omogućeno i da kreira pravila, po kojima određuje ponašanje uređaja u raznim situacijama. Na slici 4, vidimo pravilo Close binds. Ono podrazumeva spuštanje roletni ako temperatura bude iznad 25 stepeni. Moguće je i kombinovati više uslova.



Slika 4. *Definisanje pravila na platformi*

3. SPECIFIKACIJA APLIKACIJE

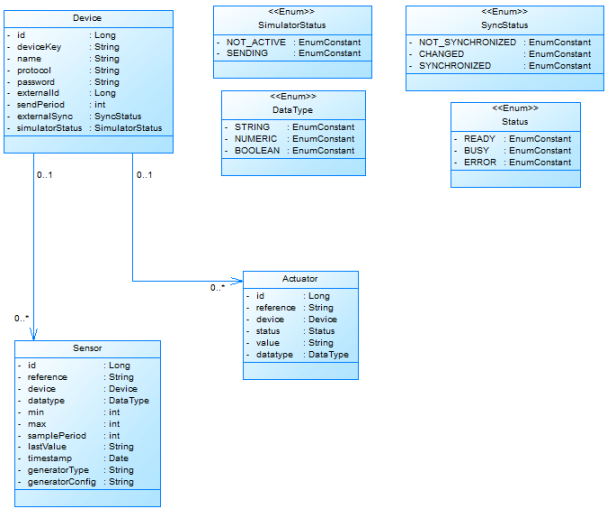
3.1. Dijagram klasa

Glavna klasa je Device, koja je sastavljena iz sledećih atributa: deviceKey (oznaka uređaja), name (ime), protocol (format poruke), externalId (id platforme), sendPeriod (na koliko vremenski šalje podatke), password (šifra), externalSync (da li je syncovana platforma sa našom bazom), kao i simulatorStatus (da li je pušteno slanje podataka).

Sledeća klasa je Sensor, neki od njenih atributa su: reference (jedinstveni identifikator senzora), device (kom uređaju pripada), dataType (enumeracija koja određuje tip podatka), min i max (vrednosti), lastValue (poslednja vrednost), timestamp (vreme očitavanja)...

SensorData klasa služi da opiše podatke koje očitava senzor, obuhvata: sensor (koji pokazuje koji senzor je očitao vrednost), device (kom uređaju taj senzor pripada), value (očitana vrednost) i timestamp (datum i vreme očitavanja).

Poslednja klasa modela je Actuator, koja je vrlo slična senzoru, a sastoji se iz: reference (jedinstveni identifikator), device (uređaj kome pripada), status (enumeracija u kom je stanju – READY, BUSY, ERROR), dataType (tip podataka sa kojima radi), value (vrednost sa kojom radi). Dijagram klasa prikazan je na slici 5.



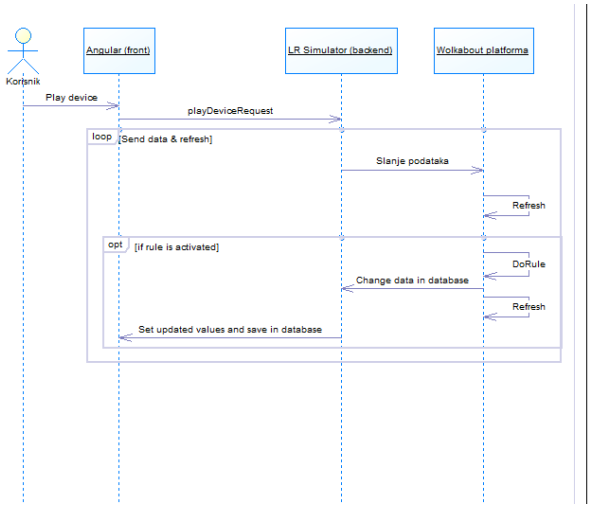
Slika 5. *Dijagram klasa*

3.2. Dijagram sekvence

Dijagram sekvence, kao što vidimo na slici 6, prikazuje pokretanje uređaja za slanje podataka. Ono, takođe, kreće od korisnika koji aktivira LR Simulator za slanje podataka određenog uređaja na platformu.

Wolkabout platforma prima podatke, obrađuje ih i osvežava vrednosti.

Ukoliko se prilikom nekog očitavanja ispuni uslov za pokretanje nekog pravila, platforma ga primenjuje i postavlja nove vrednosti.



Slika 6. *Dijagram sekvence*

4. ZAKLJUČAK

Internet of things je tehnologija koja je još uvek u razvitku i trebala bi tek da zablista u budućnosti. Home automation je sve češći i češći u domovima. Određeni proizvođači sve više ubacuju IoT funkcionalnosti u svoje uređaje, tako recimo LG frižideri javljaju svojim vlasnicima šta nedostaje i šta od namirnica treba da se kupi. To je samo jedna od sve učestalijih primena.

Budućnost IoT se ogleda u umrežavanju velikog broja uređaja u industriji i svakodnevnom životu, kao i njihovoj lakšoj kontroli i upravljanju sa udaljenih pozicija, kako bi se olakšao ubrzani život čovečanstva.

Što se tiče mog projekta, trebalo bi povezati još neki realan senzor/aktuator na Raspberry Pi i još više automatizovati svoj dom.

5. LITERATURA

[1] Internet of things
https://www.gkmit.co/blog/internet-of-things-iot-introduction-applications-and-future-scope
[2] Home automation
https://dzone.com/articles/home-automation-using-iot
[3] Angular okruženje
https://angular.io/
[4] Java
https://en.wikipedia.org/wiki/Java_(programming_language)
[5] Spring okruženje
https://spring.io/
[6] CRUD
https://www.codecademy.com/articles/what-is-crud
[7] MQTT protokol
http://mqtt.org/faq
[8] Dependecy injection
https://en.wikipedia.org/wiki/Dependency_injection
[9] REST
https://www.codecademy.com/articles/what-is-rest
[10]HTTP
https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[11] Spring Data
https://spring.io/projects/spring-data
[12] Hibernate
http://hibernate.org/
[13] Sqlite
https://www.sqlite.org/index.html
[14] Python
https://www.python.org/
[15] Raspberry Pi
https://www.raspberrypi.org/
[16] GPIO
https://en.wikipedia.org/wiki/General-purpose_input/output
[17] Adafruit
https://www.adafruit.com/
[18] Polinomska regresija
https://machinelearningmastery.com/polynomial-features-transforms-for-machine-learning/?fbclid=IwAR2pdyhj3CK5fkHzM9WuiKldOAMcIIBexXJXcU0E1Aa1tvCNMuYFSYqkbCA

Kratka biografija:

Luka Radović rođen je u Novom Sadu 1996. god. Diplomski rad na Fakultetu tehničkih nauka iz oblasti Elektrotehnike i računarstva odbranio je 2019. godine. Od tada je zaposlen na FTN kao saradnik u nastavi.
kontakt: lukaradovic96@gmail.com