Implementacija koncepta digitalnog dvojnika korištenjem Microsoft Azure platforme

Bazina, Domagoj

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA FORENZIČNE ZNANOSTI

FORENZIKA I NACIONALNA SIGURNOST

DIPLOMSKI RAD

IMPLEMENTACIJA KONCEPTA DIGITALNOG DVOJNIKA KORIŠTENJEM MICROSOFT AZURE PLATFORME

Mentor: izv. prof. dr. sc. Toni Perković

DOMAGOJ BAZINA

701/2021.

Split, srpanj 2023. godine

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA FORENZIČNE ZNANOSTI

FORENZIKA I NACIONALNA SIGURNOST

DIPLOMSKI RAD

IMPLEMENTACIJA KONCEPTA DIGITALNOG DVOJNIKA KORIŠTENJEM MICROSOFT AZURE PLATFORME

Mentor: izv. prof. dr. sc. Toni Perković

DOMAGOJ BAZINA

701/2021.

Split, srpanj 2023. godine

Ovaj rad je napravljen u suradnji sa ETK Istraživačkim odjelom, Ericsson Nikola Tesla, Split, Hrvatska. This work was done in a collaboration with ETK Research department, Ericsson Nikola Tesla, Split, Croatia.

Želim iskazati zahvalu dr.sc. Toniju Masteliću na podršci i stručnom vodstvu pri izradi ovog rada.

Rad je izrađen u Splitu pod nadzorom mentora izv. prof. dr. sc. Tonija Perkovića u vremenskom razdoblju od 1. travnja 2023. godine do 3. srpnja 2023. godine.

Datum predaje diplomskog rada: 3. srpnja 2023.
Datum prihvaćanja rada: 6. srpnja 2023.
Datum usmenog polaganja: 11.srpnja 2023.

Ispitno povjerenstvo:

1. Doc.dr.sc. Nina Mišić Radanović
2. Prof.dr.sc. Josip Kasum
3. Izv.prof.dr.sc Toni Perković

Sadržaj

1. Uvod

U današnjem svijetu, umjetna inteligencija sve više dolazi do izražaja, bilo u poslovnom ili društvenom kontekstu. Posebno se ističe njezina primjena u području obrade prirodnog jezika, kao što je slučaj s ChatGPT-om, koji omogućuje komunikaciju između ljudi i umjetne inteligencije. No, ovaj revolucionarni alat predstavlja samo jedan od mnogobrojnih primjera korištenja umjetne inteligencije koja već neko vrijeme rapidno napreduje i koristi se u različitim područjima poput obrade slika, autonomnih vozila, optimizacije sustava i mnogih drugih.

Jedan od zanimljivih primjera primjena umjetne inteligencije svakako uključuje i digitalne dvojnike, riječ je o tehnološkom konceptu koji omogućuje stvaranje virtualnih kopija fizičkih objekata, procesa ili sustava. Spomenute virtualne kopije ili modeli integriraju podatke iz stvarnog svijeta uz pomoć senzora i drugih uređaja, koristeći ih za optimizaciju, analizu i predviđanje putem umjetne inteligencije i strojnog učenja. Digitalni dvojnici imaju primjene u brojnim granama, a posebno u industriji, gdje omogućuju praćenje i optimizaciju proizvodnih procesa u stvarnom vremenu. Dakle, isti mogu simulirati različite scenarije i otkriti potencijalne probleme prije nego što se isti pojave u stvarnosti, a ova mogućnost je izuzetno korisna u sustavima kritične infrastrukture poput nuklearnih elektrana ili naftnih postrojenja. Kako bih digitalni dvojnik pružao optimalne performanse, neophodno je osigurati pouzdanu i sigurnu komunikaciju između senzora stvarne tvornice i samog dvojnika, a jedan od načina za postizanje toga je korištenje računalne infrastrukture u oblaku, odnosno cloud platformi.

Računalna infrastruktura u oblaku pruža korisnicima efikasan način za optimalno korištenje resursa, istovremeno pružajući skalabilnost, pouzdanost i potrebne protokole kako bih komunikaciju između uređaja i digitalnog dvojnika učinila sigurnom. Microsoft Azure je jedna od vodećih cloud platformi koja ne samo da pruža navedene mogućnosti, već nudi i dodatne značajke koje su ključne za implementaciju i upravljanje digitalnim dvojnicima. Osobito je važno istaknuti resurse koji omogućuju obradu velike količine podataka kao i analizu istih u stvarnom vremenu, što značajno olakšava proces implementacije digitalnih dvojnika.

2. Cilj rada

Cilj ovog rada je implementirati koncept digitalnog dvojnika na temelju tvornice za proizvodnju matičnih ploča uz korištenje gotovih rješenja dostupnih unutar Microsoft Azure platforme. Prije početka samog projekta, istražili smo postojeća rješenja u ovom području i odlučili kako ćemo neke od njih implementirati zajedno sa našim idejama i konceptima. Kako bih sam proces implementacije bio efikasniji, odlučili smo ga podijeliti u nekoliko koraka:

- **Prikupljanje informacija** – potrebno je prikupiti informacije i domensko znanje o sustavu, odnosno procesu na temelju kojeg želimo implementirati digitalnog dvojnika. U našem slučaju, radi se o tvornici, odnosno procesu proizvodnje matičnih ploča kojeg smo dizajnirali uz pomoć alata za izradu video igrica. Zbog tehničkih ograničenja, proces implementacije digitalnog dvojnika smo proveli na virtualnoj simulaciji tvornice koja posjeduje većinu značajki kao i stvarna tvornica.
- **Dizajniranje modela** – nakon što smo prikupili potrebne informacije o sustavu, iste je potrebno pretočiti u virtualne modele koje opisuju sam sustav, odnosno njegove značajke i svojstva. Ovo smo napravili uz pomoć gotovog Microsoft Azure servisa nazvanog Azure Digital Twins.
- **Povezivanje digitalnog dvojnika s tvornicom** – poslije definiranja modela, odnosno izrade digitalnog dvojnika, istog je potrebno povezati s podatcima iz stvarne tvornice. Kako bih omogućili uspješnu i sigurnu komunikaciju između stvarne tvornice i digitalnog dvojnika, stvorili smo komunikacijski kanal sastavljen od nekoliko različitih Microsoft Azure servisa.
- **Izrada korisničkog sučelja** – korisničko sučelje je jedan od jednostavnijih način pomoću kojeg korisnici mogu stupiti u interakciju s računalnim sustavom. U ovu svrhu ćemo dizajnirali jednostavno korisničko sučelje korištenjem Angular programskog okvira, čija će uloga biti upravljanje i nadzor nad digitalnim dvojnikom, odnosno stvarnim sustavom.
- **Validacija i provođenje simulacija** - po završetku prethodno spomenutih koraka, potrebno je testirati performanse sustava i zatim provesti nekoliko simulacija, odnosno scenarija kako bih ispitali/dokazali značajke i mogućnosti koje nam digitalni dvojnik pruža.

3. Izvor podataka i metode

Ovaj diplomski rad je sastavljen od dva dijela, teorijskog i empirijskog, odnosno praktičnog dijela. U teorijskom dijelu rada smo ukratko objasnili sve tehnologije i koncepte korištene u implementaciji praktičnog dijela rada, a kao izvore podataka smo uglavnom koristili literaturu i projekte otvorenog tipa (eng. *open-source*). Korištena literatura se uglavnom fokusirala na znanstvene članke, članke iz časopisa, konferencijske radove i internetske izvore, a s obzirom na to da smo uglavnom koristili rješenja definirana unutar Microsoft Azure platforme, oslonili smo se na njihovu dokumentaciju i izvorni kod. Neke od metoda korištene u teorijskom dijelu uključuju deskriptivnu metodu, metodu analize i sinteze i induktivno deduktivnu metodu.

Kao što sam spomenuo u prethodnom poglavlju, cilj ovog rada je implementacija digitalnog dvojnika u Microsoft Azure platformu, stoga smo se fokusirali na njihova postojeća rješenja i kombinirali ih s našim idejama i konceptima vezanim uz digitalnog dvojnika. Druge tehnologije korištene u ovom radu su isto otvorenog tipa, stoga svu njihovu dokumentaciju i izvorni kod (eng. s*ource code*) je relativno jednostavno pronaći na internetu. U ovom dijelu rada smo koristili eksperimentalnu metodu te modeliranje i simulaciju s obzirom na to da smo dizajnirali softver, prikupljali podatke, provodili testiranja te u konačnici donosili zaključke.

4. Digitalni dvojnici

Digitalnim dvojnicima ili blizancima smatramo virtualne modele fizičkih objekata, procesa i sustava temeljenih na podatcima u stvarnom vremenu, a stvorenih u svrhu optimizacije i analize istih. Koncept digitalnog dvojnika se može primijeniti na različite objekte i scenarije, od jednostavnih predmeta poput strojeva, do složenih sustava poput tvornice ili nekog proizvodnog segmenta. Temelj na kojem svaki digitalni dvojnik počiva su podatci i informacije o sustavu na temelju kojeg je isti napravljen, stoga informacije o mogućnostima i svojstvima sustava definiramo prilikom izrade digitalnih modela, a podatke o trenutnom stanju sustava šaljemo putem IoT uređaja i senzora. Sama svrha digitalnih dvojnika je optimizacija i poboljšanje performansi određenog sustava, a da bih to mogli napraviti potrebno je provesti analize i simulacije koje u stvarnim uvjetima riskantno provesti, posebice ako se radi o kritičnoj infrastrukturi poput nuklearnih elektrana ili naftnih platformi. Osim što simulacije mogu biti riskantne, iste mogu izazvati značajan gubitak vremena i resursa, što se u konačnici može negativno odraziti na krajnjeg korisnika. Slijedom navedenog, digitalni dvojnici su izvrsna platforma za provođenje spomenutih analiza i simulacija s obzirom na to da je riječ o „virtualnom svijetu“, a ujedno mogu predvidjeti i potencijalne probleme, odnosno ugroze koje mogu imati negativne posljedice na sam sustav (1).

Prvi spomen na koncept digitalnih dvojnika možemo pronaći 2002. godine od strane Johana Vickersa, koji je bio zaposlen kao inženjer u tvrtki General Electric. Tijekom prezentacije na godišnjoj konferenciji tvrtke General Electric, Vickers je iznio ideju da se senzori mogu iskoristiti za skupljanje podataka o raznim objektima u stvarnom vremenu i da se na temelju spomenutih podataka mogu izraditi virtualne replike samih objekata. Osim načina rada, Vickers je iznio i područjima u kojima bi se digitalni dvojnici mogli primijeniti, a ujedno je i predvidio napredak senzorne tehnologije kao i umjetne inteligencije koja će dodatno poboljšati primjenu digitalnih dvojnika. Osim Johana Vickersa, važno je spomenuti i Michael Grieves-a, profesor na sveučilištu Florida Institute of Technology. Grieves je razvio koncept digitalnih dvojnika za proizvodne procese i opisao ga u svojoj knjizi "Virtual Enterprise, The Executive's Guide to the Enterprise Amalgamation", koja je objavljena 2002. godine. Johan Vickers i Michael Grieves se smatraju pionirima u razvoju digitalnih dvojnika, temeljem njihovih ideja, isti su postali popularni kako industriji tako i u istraživanjima (2). S razvojem tehnologija, koncept

digitalnih dvojnika je stekao primjenu u brojnim granama i tvrtkama, od kojih posebno ističemo:

- Industriju i proizvodnju – digitalni dvojnici se koriste za modeliranje proizvodnih postrojenja i strojeva, optimizaciju performansi opreme i otkrivanje potencijalnih problema prije nego što dođe do kvara. Primjer za to je tvrtka General Electric koja koristi digitalne dvojnike u proizvodnji avionskih motora kako bi optimizirali performanse i produžili vijek trajanja motora (3). Ova grana nam je posebno zanimljiva s obzirom na to da je rad temeljen na procesu implementacije digitalnih dvojnika na primjeru tvornice za proizvodnju matičnih ploča.
- Zdravstvo – u medicini i zdravstvu digitalni dvojnici se koriste za modeliranje ljudskog tijela u svrhu poboljšanja dijagnostike i plana liječenja. Sim&Cure je francuska tvrtka koja se bavi razvojem softvera za planiranje operacija mozga i simulaciju krvnih žila (4). Njihova tehnologija koristi digitalne dvojnike kako bi omogućila precizniju i personaliziranu operaciju za svakog pacijenta. Uz pomoć digitalnog dvojnika, Sim&Cure može modelirati anatomske karakteristike pacijentovog mozga i simulirati cirkulaciju krvi kroz krvne žile. Na temelju simulacija, kirurzi mogu planirati operaciju s visokom preciznošću, minimizirajući rizike i poboljšavajući ishode same operacije.
- Transport – digitalni dvojnici se mogu iskoristiti za optimizaciju logistike, transporta i urbanih sustava. Primjer za to je grad Singapur koji je razvio "Virtual Singapore", riječ je o digitalnom dvojniku cijelog grada koji se koristi za planiranje gradskih projekata, upravljanje rizicima i gradskim resursima (5). Ovaj digitalni dvojni omogućuje gradskim vlastima da modeliraju prometne tokove u stvarnom vremenu, predviđaju prometne gužve te planiraju nove prometne rute kako bi se poboljšala prometna situacija u gradu .
- Energetiku – u energetici, digitalni dvojnici se koriste za praćenje i upravljanje energetskim sustavima, uključujući obnovljive izvore energije. Primjer upotrebe digitalnih dvojnika u energetskoj industriji je tvrtka Siemens Energy (6) koja koristi digitalne dvojnike za optimizaciju rada svojih turbina. Koristeći senzore, podatci o radu turbina se prikupljaju i analiziraju, a zatim se stvara digitalni model turbine. Korištenjem digitalnog modela, moguće je simulirati različite scenarije i izvršiti testiranja bez utjecaja na stvarne turbine, što pomaže u optimizaciji rada i održavanju istih, a ujedno smanjuje vrijeme nedostupnosti (eng. *downtime*) te troškove održavanja.

4.1. Proces integracije digitalnih dvojnika

Proces implementacije, odnosno način izrade i primjene digitalnih dvojnika ovisi o području, odnosno sustavu na temelju kojeg želimo izraditi digitalnog dvojnika. Unatoč činjenici što sam proces nije standardiziran, sastoji se od nekoliko koraka koji su primjenjivi u većini slučaja (7), a navedeni koraci uključuju:

- **Prikupljanje podataka i analiza** - početni korak u izradi digitalnih dvojnika je prikupljanje podataka o stvarnom sustavu ili objektu na temelju kojeg želimo napraviti digitalnog dvojnika. Spomenuti podatci mogu uključivati razne izvore i formate poput fotografija, IoT uređaja, informacija o materijalima, performansama, mogućnostima, a mogu uključivati CAD crteže i ostale formate. Navedeni podatci su ključni za drugi korak, odnosno proces izrade modela digitalnih dvojnika.
- **Izrada 3D modela** – nakon prikupljanja relevantnih podataka i informacija, možemo se usmjeriti na sam proces izrade 3D modela. Ovo možemo napraviti uz pomoć računalnih programa za izradu video igrica, poput Unreal Engine-a ili Unity-a. Temelj za izradu ovih 3D objekata može uključivati skeniranje stvarnih objekata, korištenje CAD crteža ili korištenje gotovih modela koje možemo pronaći na internetu.
- **Povezivanje 3D modela s podatcima** - Treći dio procesa uključuje povezivanja prikupljenih informacija s prethodno izrađenim 3D modelom, dakle nakon što smo izradili objekt, moramo ga povezati sa svim relevantnim informacijama iz stvarnog svijeta za taj objekt, poput materijala, performansi, mogućnosti, senzornih vrijednosti i ostalih podataka. Rezultat ovog koraka će biti virtualni model koji sadrži sve relevantne informacije za stvarni sustav, a ujedno posjeduje i 3D prikaz tog sustava, što olakšava vizualizaciju i upravljanje istim.
- **Validacija i simulacija –** Nakon što smo izradili virtualne modele, na temelju istih možemo napraviti digitalne dvojnike te provesti procese validacije i simulacije kako bi provjerili njegovu valjanost i funkcionalnost. U ovoj fazi se koriste računalne simulacije za testiranje performansi digitalnog dvojnika u različitim uvjetima i situacijama, kako bi se osiguralo da digitalni dvojni odražava ponašanje stvarnog objekta ili procesa.
- **Kontinuirano ažuriranje i učenje –** Posljednji korak se koristi u svrhu analize i optimizacije stvarnog objekta odnosno sustava. Nakon što smo ustvrdili da je digitalni dvojni validan i funkcionalan, možemo započeti s brojnim procesima učenja i

simulacija, sve u svrhu optimizacije stvarnog sustava. Međutim, kako bih digitalni dvojni bio uspješan, potrebno ga je konstantno ažurirati i prilagođavati u skladu s promjenama u stvarnom sustavu. Uz kontinuiran dotok podataka, i vođenje evidencije o svim promjenama unutar stvarnog sustava, digitalni dvojni može provoditi simulacije koje mogu biti korisne u prevenciji potencijalnih problema, ali može i provoditi optimizaciju nad stvarnim sustavom te pridonijeti njegovoj učinkovitosti.

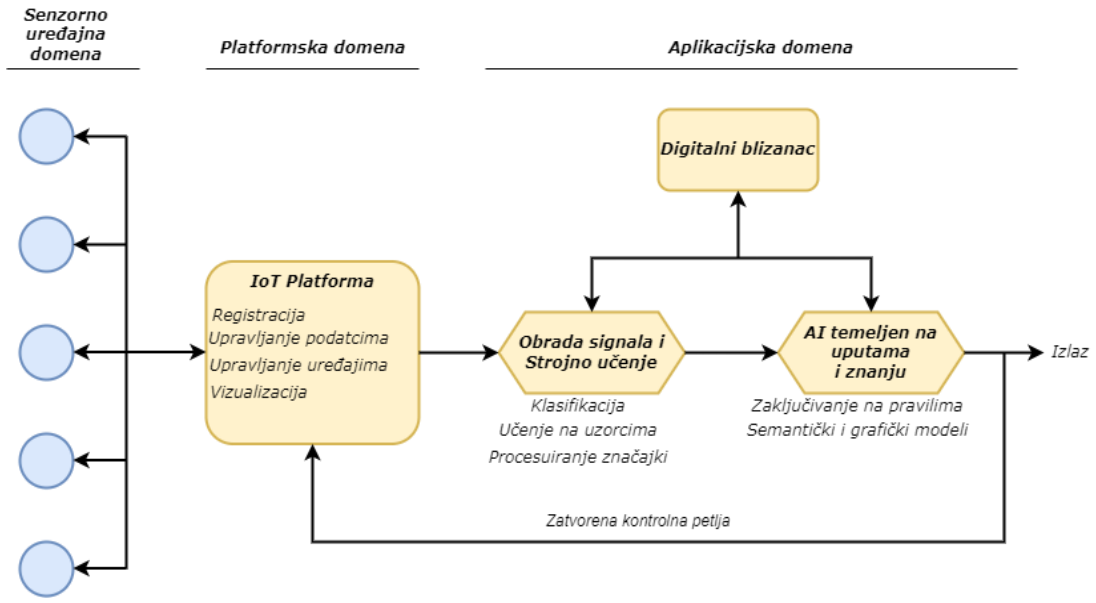
4.2. Umjetna inteligencija i digitalni dvojnici

Uz pojam digitalnog dvojnika, najčešće pronalazimo termine poput interneta stvari, umjetne inteligencije i strojnog učenja, a glavni razlog je to što uz pomoć spomenutih tehnologija, odnosno računalnih grana, digitalni dvojni može provoditi simulacije i optimizacije stvarnog sustava. Kako bih se bolje upoznali s načinom na koji digitalni dvojnik funkcionira, kratko ćemo objasniti ulogu svakog od navedenih termina u životnom ciklusu (eng. lifecycle) jednog digitalnog dvojnika (Slika 1).

- **Internet stvari** - Internetom stvari (eng. *Internet of Things - IoT*) smatramo mrežu međusobno povezanih uređaja putem interneta koji su sposobni prikupljati i razmjenjivati podatke bez ljudske potpore. Ovi uređaji mogu biti senzori, pametni telefoni, kućanski aparati, računala i slično, a glavna njihova prednost je to što omogućuju umrežavanje uređaja različitih proizvođača te njihovu integraciju. Internet stvari se koristi u brojnim industrijama poput proizvodnje, zdravstva, energetike, a njegovu primjenu možemo pronaći u pametnim kućama, uređajima za upravljanjem prometom i slično. Uloga IoT uređaja kod digitalnih dvojnika se odnosi na prikupljanje podataka o stvarnim objektima koje želimo modelirati, a prikupljeni podatci se koriste za stvaranje i ažuriranje digitalnih dvojnika (8). Osobito je važno da spomenuti podatci stižu u stvarnom vremenu što omogućuje donošenje odluka i predviđanje potencijalnih problema. IoT senzori se najčešće koriste za mjerenje temperature, vlažnosti, potrošnje energije, vibracija, smjera vjetra i slično.
- **Umjetna inteligencija** – Umjetna inteligencija (eng. *Aritifical intelligence - AI*) područje unutar računalne znanosti čiji je fokus razvoj inteligentnih računalnih sustava koji su sposobni izvršavati zadaće koji inače zahtijevaju ljudsku inteligenciju. Neke od sposobnosti umjetne inteligencije uključuju zaključivanje, rasuđivanje, učenje,

prepoznavanje uzoraka i slično. Umjetna inteligencija koristi različite tehnike kako bi omogućila računalima da razumiju i analiziraju podatke, a temeljem tih uvida postiže inteligentne odluke ili ponašanje. Najčešće korištene tehnike su **strojno učenje**, duboko učenje, obrada prirodnog jezika i evolucijski algoritmi. Uloga umjetne inteligencije za digitalne dvojnike je izuzetno ključna zbog toga što pospješuje analitičke i prediktivne sposobnosti što u konačnici omogućuje bolje upravljanje sustavom (9). Glavna primjena je to što AI sustavima omogućuje predviđanje problema prije nego se dogode, a što je produkt analize velike količine podataka prikupljenih od strane IoT uređaja.

- **Strojno učenje** – strojno učenje (eng. *Machine Learning - ML*) je područje umjetne inteligencije koja se bavi razvojem modela i algoritama koji omogućuju računalima učenje iz podataka i izvršavanje naredbi bez definiranih programskih uputa. Obično računala izvršavaju zadatke prema strogo definiranim i jasno strukturiranim pravilima, međutim algoritmi strojnog učenja omogućuju da računala analiziraju obrasce i značajke u podatcima. Temeljem navedenog računala se mogu samostalno razvijati i prilagođavati modele kako bih usavršila svoju izvedbu. Neki od najčešćih primjena strojnog učenja su prepoznavanje govora ili preporučivanje sadržaja temeljenog na ponašanju korisnika. Uloga strojnog učenja kod digitalnih dvojnika se posebno koristi za razvoj i optimizaciju algoritama koji se koriste za analizu podataka, provođenje simulacija i predviđanje ponašanja stvarnog objekta ili procesa. Strojno učenje omogućuje digitalnom dvojniku da se prilagodi novim podacima i promjenama u stvarnom objektu ili procesu (10).



Slika 1. Prikaz i uloga interneta stvari (IoT), strojnog učenja(ML) i umjetne inteligencije (AI) unutar digitalnog dvojnika

4.3. Prednosti i nedostatci digitalnih dvojnika

Kao i većina tehnoloških koncepata, digitalni dvojnici imaju svoje prednosti i nedostatke (7) koje treba dobro proučiti prije nego što se tvrtka odluči primijeniti navedenu tehnologiju. Neke od glavnih prednosti digitalnog dvojnika uključuju:

- Poboljšanje učinkovitosti i produktivnosti – digitalni dvojnici mogu se iskoristiti za optimiziranje operacija, smanjivanje vremena nedostupnosti tako da uoče problem prije nego što on nastane.
- Osnažuje donošenje odluka – s obzirom na to da digitalni dvojnici pružaju podatke u stvarnom vremenu o performansama fizičkog objekta, možemo donositi pravovremene odluke i samim tim pospješiti poslovne procese.
- Smanjenje operativnih troškova – digitalni dvojni je sposoban identificirati i optimizirati neučinkovite procese i samim tim smanjiti operativne troškove.
- Povećanje sigurnosti – digitalne dvojnike možemo koristiti za provođenje simulacija i testova unutar okruženja koja su potencijalno nesigurna i za koje bi stvarni testovi mogli imati katastrofalne posljedice. Na ovaj način možemo predvidjeti ponašanje sustava u ekstremnim situacijama i tako povećati sigurnost zaposlenika, sustava i okoliša.

S druge strane imamo i nedostatke koji mogu imati značajan utjecaj na krajnjeg korisnika, a oni uključuju :

- Visok trošak implementacije – troškovi za implementaciju i razvoj digitalnih dvojnika mogu biti visoki s obzirom na to da uključuju hardverska i softverska rješenja kao i domensko znanje o samom konceptu.
- Sigurnosni rizik – s obzirom na to da unutar samog digitalnog dvojnika baratamo s velikom osjetljivih podataka i informacija, moramo voditi brigu o sigurnosti i privatnosti istih.
- Integracija i kvaliteta podataka – s obzirom na to da se digitalni dvojnici oslanjaju na podatke koji mogu biti kompleksni i dolaziti iz više izvora, integracija istih može biti izazovan proces.
- Ograničena skalabilnost – digitalni dvojnici mogu biti kompleksni i zahtjevni za skalirati, posebice kada se radi o velikim sustavima, posebice ako tome pridodamo nedostatak standardizacije kod digitalnih dvojnika i njihove izrade.

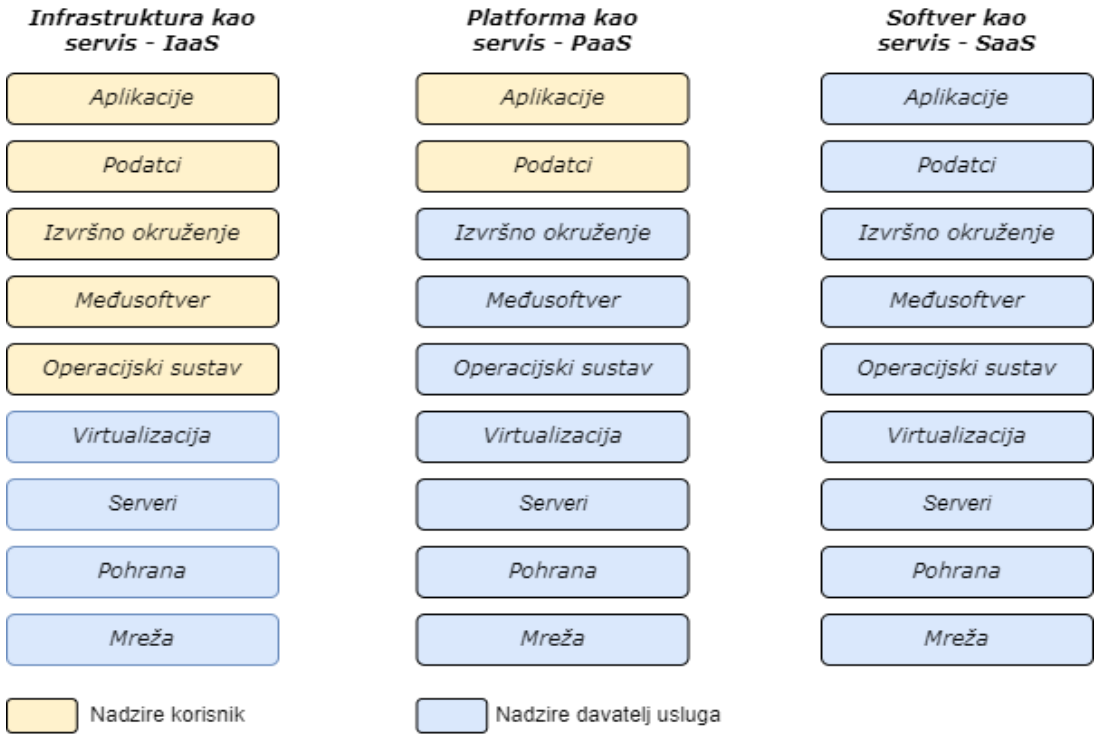
5. Microsoft Azure platforma

Cloud platformama smatramo računalne platforme koje pružaju pristup računalnim resursima poput pohrane, servera i aplikacija putem interneta. Navedene resurse upravlja i održava davatelj usluga (eng. *provider*) tako da korisnici mogu samostalno skalirati infrastrukturu i resurse prema vlastitim potrebama bez da ne moraju inicijalno potrošiti veliku količinu novca za kupovinu i održavanje vlastitih resursa. Dakle korisnik ne mora samostalno kupovati fizičke uređaje (računala, servere) i održavati ih, nego jednostavno zatraži od davatelja usluga određenu količinu resursa i to plaća prema nekakvom izračunu. Nadalje, jedan od benefita cloud platformi je pružanje računalnih usluga korisnicima na jednostavan i relativno jeftin način, što korisnicima omogućuje da se usmjere na srž svojih poslovnih aktivnosti, a ne na računalnu infrastrukturu. Računalstvo u oblaku (eng. cloud computing) vuče svoje korijene iz početnih dana interneta, ali ideja cloud platformi se stvara početkom 2000-ih godina, tako je 2006. godine Amazon Web Services (AWS) pokrenuo servis pod nazivom „Elastic Computing Service (EC2)“ koji je kompanijama omogućio iznajmljivanje računalnih resursa prema njihovim potrebama. Ovo je bila prekretnica u tadašnjem načinu razmišljanja po pitanju fizičke računalne infrastrukture i njenog održavanja. Ubrzo su se i ostale poznate IT tvrtke uključile u ovu vrstu poslovanja te stvorile danas poznate cloud platforme poput Microsoft Azure-a, Google Cloud Platform i IBM Cloud-a (11).

Microsoft Azure je platforma pokrenuta 2010. godine pod nazivom Windows Azure, čija je inicijalna svrha služila za izradu i razvijanje .NET aplikacija. Ubrzo, 2014. godine naziv je promijenjen u Microsoft Azure kako bi se naglasilo da platforma podržava različite vrste operativnih sustava, programskih jezika i tehnologija, ne samo Windows proizvode. Od tada, platforma je doživjela veliki rast i ekspanziju u vidu korisnika te je danas jedna od najpoznatijih cloud platformi koja omogućuju izradu, razvijanje i upravljanje aplikacijama i servisima kroz brojne podatkovne centre u svijetu. U zadnjem desetljeću cloud platforme su postale krucijalan dio IT industrije pa ih stoga koristi enorman broj tvrtki različitih veličina, procjene su da će do 2024. godine cloud platforme vrijediti preko trilijun dolara (12). Ono što je zajedničko svim cloud platformama pa tako i Microsoft Azure platformi, je način na koji strukturiraju resurse. Dakle, cloud platforme nude širok spektar servisa, od najniže razine apstrakcije poput virtualnih mašina, sve do gotovih softvera poput Microsoft Office 365 alata. S druge strane, korisnici drugačije zahtjeve, netko želi samostalno konfigurirati infrastrukturu na kojoj će

podignuti nekakve servise i aplikacije, a imamo i korisnike koji žele gotov proizvod i ne zanima ih računalna infrastruktura. Slijedom navedenog, razvijena je podjela (13) na 3 modela:

- **Infrastruktura kao servis** (eng. *Infrastructure as a Service - IaaS*) – model koji nudi računalne resurse poput virtualnih mašina, pohrane i mreže u najjednostavnijem obliku. Dakle korisnici imaju mogućnost sami konfigurirati operacijski sustav, različite vrste softvera i aplikacija. Ovaj model rada nudi najveću razinu kontrole s obzirom na to da korisnici sami upravljaju infrastrukturom kao i aplikacijama na njoj.
- **Platforma kao servis** (eng. *Platform as a Service - PaaS*) – model koji pruža gotove platforme za izradu, pokretanje i upravljanje aplikacijama. Dakle korisnici imaju potrebnu infrastrukturu koja uključuje hardver i softver odnosno operacijski sustav, baze podataka, alate za razvijanje aplikacija i slično. Ovaj model rada idealan je za korisnike koji se žele fokusirati na pokretanje i razvijanje aplikacija, a ne na infrastrukturu koja se nalazi „ispod haube".
- **Softver kao servis** (eng. *Software as a Service - SaaS*) – model koji pruža gotove komade software-a kojima se pristupa putem interneta. Dakle korisnici imaju pristup različitim vrstama softvera i aplikacija te ih samo prilagođavaju vlastitim potrebama. Spomenuti model rada je idealan za korisnike kojima su potrebna gotova softverska rješenja poput email servisa ili alata za produktivnost.



Slika 2. Prikaz cloud servisa kroz 3 različita biznis modela (IaaS, PaaS, SaaS)

Rezimirano, *IaaS* pruža najosnovnije računalne elemente kao infrastrukturu koju je potrebno konfigurirati, *PaaS* nudi gotovu platformu koja se koristi za pokretanje i razvijanje aplikacija i *SaaS* nudi gotova softverska rješenja kojima se pristupa putem interneta. U praktičnom dijelu rada, koristili smo se uglavnom na servise koje spadaju u *PaaS* model rada, s obzirom na to da smo se fokusirali na same aplikacije i servise, odnosno njihovu funkcionalnost. S obzirom na sličnosti između cloud platformi, glavni razlog zašto smo se odlučili koristiti baš Microsoft Azure Platformu je to što Azure platforma nudi servis pod nazivom „Digital Twin“ koji se po svojim karakteristikama uklapa i funkcionalnošću uklapa u istoimenu tematiku. Kao što sam spomenuo u uvodnom dijelu rada, cilj ovog rada je implementirati koncept digitalnog dvojnika uz pomoć Microsoft Azure platforme, stoga ćemo se u ostatku ovog poglavlja upoznati sa svim Azure servisima i resursima koje smo koristili u svrhu ostvarivanja tog cilja.

5.1. Azure Digital Twin

Azure Digital Twin je platforma koja omogućuje izradu digitalnih dvojnika temeljenih na digitalnim modelima stvarnih objekata, sustava i okoliša, a sve to u svrhu poboljšanja proizvoda, optimizacije performansi, smanjivanja troškova i poboljšanja korisničkog iskustva. Riječ je o servisu koji se unutar Azure platforme nudi kao PaaS, a nudi sve one značajke koje su neophodne za funkcioniranje takvog sustava, poput povezivanja s IoT platformama i uređajima, mogućnost nadzora i analize, te pohranu podataka u drugim servisima definiranim unutar Azure platforme. Najjednostavniji opis ovog servisa bi bila grafička baza podataka, dakle možemo definirati entitete, njihova svojstva i međusobnih ih povezati. Naravno, Digital Twin servis pruža više od navedenih mogućnosti, a kako bih shvatili ulogu i mogućnosti ovog servisa, upoznat ćemo se s nekoliko njegovih značajki (14), poput:

- **Definiranje modela i izrada digitalnih instanci** – Digital Twin servis nam omogućuje izradu instanci ili entiteta temeljenih na modelima pomoću kojih opisujemo/definiramo stvarne objekte. Kako bih definirali/opisali modele koristimo specijalizirani jezik zvan Digital Twin Definition Language (DTDL) koji je vrlo sličan JavaScript Object Notation (JSON) formatu. DTDL ima jasno definiranu strukturu koja nam omogućuje definiranje modela, njegovih svojstava, komponenti, naredbi, telemetrijskih podataka i veza s drugim entitetima. Uz definiranje vlastitih modela možemo se i poslužiti postojećim modelima (eng. *ontologies*) te ih dodatno proširiti/prilagoditi prema našim

potrebama. Nakon što smo definirali model, možemo kreirati entitete ili instance tog modela, naprimjer, ako smo definirali model za prostoriju, možemo napraviti instance te prostorije poput spavaće sobe, dnevnog boravka, kuhinje i po potrebi napraviti vezu između njih. Nadalje, Azure Digital Twins nudi brojne načine pomoću kojih korisnici mogu izrađivati entitete i međusobno ih povezivati, a oni uključuju korisničko sučelje „Azure Digital Twins Explorer“ i različite verzije softverskih sučelja (eng. *Software Development Kit - SDK*) podržanih od strane nekoliko programskih jezika poput Pythona, C#, Jave, JavaScript-a i C++.

- **Stvaranje konteksta i povezivanje podataka s Digital Twin servisom** – nakon što smo izradili digitalnu repliku stvarnog objekta potrebno je istu povezati sa stvarnim podatcima, tako da digitalni dvojni uvijek ima pristup najnovijim podatcima (eng. *real-time data*). Jedan od načina kako to možemo napraviti je korištenjem Azure servisa pod nazivom IoT Hub, tako da stvarne senzore i aktuatore povežemo s virtualnim IoT uređajima. Povezivanjem stvarnih uređaja s digitalnim dvojnikom, dali smo kontekst istima, pa tako možemo znati u kojoj prostoriji se određeni uređaj nalazi, koji dodatni uređaji se nalaze u toj istoj prostoriji, ili pak dobaviti sve uređaje tog tipa unutar cijele kuće ili zgrade. Uz IoT Hub, postoje i drugi načini za povezivanja uređaja i ažuriranje digitalnih dvojnika u stvarnom vremenu, ti načini mogu uključivati REST API sučelje ili druge Azure servise poput logičkih aplikacija ili Function App-a o kojem ćemo govoriti u nastavku. Važno je spomenuti kako Azure Digital Twins pruža brojne načine interakcije s istim, kako na ulazu tako i na izlazu što pozitivno utječe na mogućnost povezivanja s ostalim resursima i servisima.

- **Izvršavanje upita** – Azure Digital Twin podržava posebno strukturirane upite (eng. *query*) naredbe temeljem kojih možemo dobiti informacije i uvid o njegovom trenutnom stanju, svojstvima, vezama, modelima i slično. Ovakve naredbe su izrazito važne s obzirom na to da se mogu kombinirati i tako koristiti u svrhe analize i upravljanja samim digitalnim dvojnikom.

- **Vizualizacija** – jedan od dodatnih benefita Azure Digital Twin servisa je povezivanje digitalnih dvojnika s 3D okruženjem. Nakon definiranja modela i izrade grafa sastavljenog od entiteta i njihovih veza, možemo ih povezati s nekakvim 3D objektima uz pomoć korisničkog sučelja nazvanog „3D Scenes Studio“. Dakle ovo sučelje olakšava vizualizaciju i identifikaciju entiteta i njihovih odnosa, a dodatno omogućava

implementaciju biznis logike. Važno je napomenuti kako je ovo probna (engl. *preview*) funkcionalnost Azure Digital Twins servisa, stoga smo se odlučili implementirati vlastitu verziju korisničkog sučelja, a s kojim ćemo se upoznati u nastavku rada.

- **Integracija s ostalim Azure servisima** – kao što sam napomenuo Azure Digital Twins podržava integraciju s ostalim Azure servisima, posebice u svrhu spremanja povijesti podataka ili prosljeđivanja događaja drugim servisima poput korisničkog sučelja. Stoga postoje relativno jednostavni načini kako možemo „hvatati" sve događaje proizašle iz Azure Digital Twins servisa i pohranjivati ih uz pomoć Azure Data Explorer ili pak prosljeđivati drugim servisima uz pomoć Event Hub servisa.

Iz navedenih značajki i mogućnosti, lako se može zaključiti kako se radi o zanimljivom servisu koji se izvanredno uklapa u cijelu priču, stoga smo se i odlučili koristiti Microsoft Azure platformu, umjesto da samostalno kreiramo servis koji nudi navedene mogućnosti.

5.2. IoT Hub

IoT Hub je servis koji omogućuje razvoj, upravljanje i komunikaciju između IoT uređaja i istoimenih aplikacija, dakle ovaj servis omogućuje povezivanje IoT uređaja s cloud servisima, odnosno prikupljanje podataka s IoT uređaja i upravljanje s istima. IoT Hub podržava nekoliko načina za komunikaciju, koje uključuju telemetriju između uređaja i oblaka (eng. *device-to-cloud telemetry*), učitavanje datoteka s uređaja na oblak i metodu zahtjev-odgovor (eng. *request-replay*) pomoću koje možemo kontrolirati uređaje. Nadalje, IoT Hub pruža usluge praćenja i nadzora uređaja, tako da možemo dodavati i spajati nove uređaje, brisati i uklanjati postojeće ili pak nadzirati potencijalne greške koje se mogu dogoditi. Kao i većina Azure servisa, IoT Hub je skalabilan servis koji podržava spajanje preko milijun IoT uređaja i konzumaciju poruka/događaja koji proizlaze iz navedenih uređaja. Pogodan je i za spajanje s ostalim Azure servisima poput Event Hub ili Event Grid servisa, a ujedno podržava komunikacijske protokole između uređaja i cloud-a poput MQTT, AMQP i HTTPS protokola. Uz prethodno spomenute značajke, IoT Hub pruža sljedeće mogućnosti (15):

- Upravljanje identitetom i pristupom, unutar ovog servisa je implementiran Azure Active Directory koji omogućuje upravljanje identitetima i pristupom unutar Azure platforme što dodatno povećava sigurnost IoT uređaja i aplikacija.

- Mogućnost grupiranja uređaja što olakšava upravljanje istima.
- Podržava mogućnost Edge computing-a, riječ je o usluzi koja može obraditi, formatirati i analizirati podatke prije nego se pošalju u cloud.
- IoT Hub podržava razne IoT uređaje, protokole i platforme poput Arduina, ESP32 i Raspberry Pi računala, ujedno podržava alate za razvoj softvera (SDK) u brojnim programskim jezicima poput C#, C++, Python-a, JavaScript-a i Jave.

Naplata ovog servisa se vrši samo za korištene kapacitete i resurse, a ista ovisi o količini poslanih podataka, broju IoT uređaja i dodatnim značajkama poput sigurnosti i pouzdanosti. Korisnici imaju opciju biranja viših razina usluge koji imaju veće performanse, ali se ujedno i više naplaćuju. Dodatna mogućnost naplate je samo za korištenu količinu podataka (eng. *pay-as-you-go*) koja osigurava dodatnu fleksibilnost u slučaju promjenjivog opterećenja u vidu zahtjeva.

5.3. Function App

Azure Function App je posebna vrsta servisa koju Azure nudi kao PaaS, riječ je o takozvanim serverless rješenjima, dakle korisnik se ne treba brinuti o infrastrukturi i skaliranju resursa, nego samo o kodu koji je potreban da bi aplikacija radila. Microsoft Azure je navedeni servis uveo 2016. godine, a posebna zanimljivost je naplaćivanje istog, za razliku od ostalih servisa gdje se resursi naplaćuju cijelo vrijeme (ako se ne koriste, u stanju mirovanja), Azure funkcije se naplaćuju samo po potrošnji resursa tijekom izvršavanja, što može dovesti do uštede novca. Prilikom korištenja Azure funkcija programer se treba fokusirati samo na kod i biznis logiku aplikacije, dok se Azure brine o svemu ostalom pa čak i skaliranju resursa, tako da u slučaju naglog povećanja broja zahtjeva prema određenoj aplikaciji nema potrebne za nekakvom „ručnom“ intervencijom i povećanjem resursa.

Važno je naglasiti da je riječ o manjim i jednostavnim funkcijama (aplikacijama) koje su pokretane temeljem nekog događaja (eng. *event-driven*), a mogu biti pokrenute pomoću određenih okidača (eng. *trigger*) poput HTTP zahtjeva, vremenskog okidača, promjenama nekog drugog Azure resursa poput Event Hub-a i slično. Ovaj princip rada zapravo omogućuje spomenuti način naplate resursa s obzirom na to da se resursi troše samo u slučaju nekog događaja odnosno okidača koji zapravo dovodi do izvršavanja logike (16). Dakle, sve što korisnik treba napraviti jest napisat nekakvu logiku pomoću programskog jezika, definirati o

kojem se programskom jeziku radi te definirati vanjske pakete ako oni postoje, a sama logika će se izvršiti na temelju sljedećih okidača (17):

- HTTP okidač – funkcija se izvrši kao posljedica HTTP zahtjeva.
- Vremenski okidač – funkcija se izvršava svako određeni vremenski interval, svaki dan, sat, minutu ili sekundu.
- Blob (Binary large objects) okidač – funkcija se izvršava kada dodamo neku novi dokument (eng. *blob*) u Azure Storage Account.
- Queue okidač – funkcija se izvršava kada se nova poruka doda u Azure Storage Account Queue.
- Event Hub okidač – funkcija se izvršava kada se novi događaj (*eng*. event) pošalje u Event Hub tijek događaja (eng. *data stream*).
- Cosmos DB okidač – funkcija se izvršava kada se nova vrijednost ili dokument spremi ili ažurira u bazu podataka pod nazivom Azure Cosmos DB

Azure Function App servis je poprilično jednostavan za korištenje s obzirom na to da podržava veliki broj programskih jezika poput C#, Jave, JavaScripta, Pythona i sličnih jezika, isto tako poprilično ga je jednostavno integrirati s drugim Azure servisima što može biti izuzetno korisno za stvaranje toka podataka između različitih servisa i aplikacija.

5.4. Container App i Container Instance

Azure Container Instance (ACI) je jedan od servisa pružen od strane Microsoft Azure koji omogućuje pokretanje kontejnera bez upravljanja infrastrukturom ili virtualnim mašinama. Korištenjem ACI-a vrlo je lako implementirati i upravljati aplikacijama koje se mogu „kontejnerizirati“ bez da moramo voditi brigu o infrastrukturi, operativnim sustavima i ostalim zahtjevima. Postoji mogućnost pokretanja jednog ili više kontejnera ovisno o potrebnim resursima s obzirom na to da je određen maksimalan broj računalnih resursa za pojedinačan ACI servis. Navedeni servis je izuzetno pogodan za implementaciju mikroservisne arhitekture s obzirom na to da možemo pokrenuti više kontejnera unutar jednog ACI servisa, gdje kontejneri mogu komunicirati putem lokalne mreže (eng. *localhost*) ili pak pokrenuti više ACI servisa za različite mikroservise. Dodatna pogodnost je kompatibilnost s Docker tehnologijom, stoga možemo koristiti Docker slike koje su dostupne na njihovoj službenoj stranici ili pak koristiti privatni repozitorij nazvan Azure Container Resgitry (ACR) gdje možemo stavljati

prilagođene Docker slike. Također, važno je napomenuti da ovaj servis omogućuje implementaciju standardnih konfiguracijskih parametra vezanih uz kontejnerizaciju poput prosljeđivanja ulaza (eng. *port-forwarding*), varijabli okruženja (eng. *environment variables*) i osiguravanja pohrane (eng. *volumes*). Naplata se vrši prema količini korištenih resursa, dakle prilikom izrade svakog ACI-a biramo količinu resursa za kontejnere unutar iste (18).

Azure Container App je složeniji servis od ACI zbog toga što pruža mogućnost podizanja više kontejnera odjednom kao i orkestraciju odnosno upravljanje istima. Neke od dodatnih mogućnosti koje Container App pruža u odnosu na ACI uključuju skalabilnost i balansiranje prometa (eng. *load-balancing*). Zapravo je Container App zamišljen za korištenje kompleksnih aplikacija s više kontejnera dok je ACI namijenjen za jednostavnije aplikacije, testiranje i razvoj. Iako Azure Container App omogućuje prosljeđivanje prometa i ulaza, uvijek koristi portove 8000 i 443, to možete biti problem za integraciju s drugim servisima, stoga smatram da je ovaj servis pogodniji za podizanje web aplikacija (19).

Dakle, riječ je o servisima koji omogućuju pokretanje aplikacija uz pomoć kontejnera, s tim da je ACI namijenjen za jednostavnije i aplikacije koje koriste individualne kontejnera dok se Container App koristi za složenije aplikacije koje zahtijevaju skaliranje, balansiranje prometa i aplikacije s više kontejnera. Važno je naglasiti da se obadva nude kao PaaS.

5.5. Event Hub

Event Hub servis omogućuje visoko skalabilnu i distribuiranu obradu velikog broja događaja u stvarnom vremenu, a koristan je u scenarijima gdje se koriste IoT uređaji, za zapisivanje logova ili pohranu podataka u bazu. Servis je izrazito važan za međusobnu integraciju različitih Azure usluga, zbog svoje sposobnosti za konzumaciju velikog broja događaja. Na ovaj način, moguće je konfigurirati neki servis da usmjeri sve svoje događaje prema Event Hub-u, a zatim stvoriti Function App ili drugi servis koji će se pretplatiti na događaje obrađene kroz Event Hub, i manipulirati istima na predefiniran način. Event Hub nudi značajke poput particioniranja događaja, koje pretplatnicima omogućuju "slušanje" događaja od određenog trenutka u povijesti. Važno je istaknuti i mogućnost autentikacije i autorizacije korisnika. Platforma Event Hub temelji se na standardnom protokolu AMQP 1.0 i omogućuje integraciju kroz različite programske jezike poput .NET, Jave, Pythona i JavaScripta. Također, Event Hub je kompatibilan s Apache Kafka platformom, što korisnicima omogućuje korištenje postojećih

Kafka klijenata i alata za slanje i primanje poruka putem Event Hub-a (20). Naplata se vrši prema količini obrađenih podataka i dodatnih mogućnosti koje želimo koristiti s navedenim servisom. Generalno možemo reći da Event Hub pruža fleksibilnu i skalabilnu platformu za obradu i analizu velikog broja događaja u stvarnom vremenu, koja podržava različite integracijske scenarije i omogućuje korisnicima da koriste alate i tehnologije s kojima su već upoznati.

5.6. Data Explorer

Azure Data Explorer (Kusto) je brzi i skalabilni servis za analizu velikog volumena strukturiranih i nestrukturiranih podataka u stvarnom vremenu. Omogućuje korisnicima da pretražuju i analiziraju velike količine podataka iz različitih izvora, uključujući IoT senzore, aplikacijske logove, operativne podatke i druge izvore podataka. Da bih pretraživali, filtrirali, agregirali i analizirali podatke koji su pohranjeni u Kusto bazi podataka, koristimo jezik za upite KQL (Kusto Query Language). Riječ je o jeziku jednostavnom za učenje a podržava i kombinaciju kompleksnih upita koji podržavaju operacije poput grupiranja, filtriranja, spajanja i sličnih naredbi. Data Explorer servis je dizajniran tako da može obraditi veliku količinu podataka i upita u stvarnom vremenu i da podržava mogućnost skaliranja. Kako bi korisnici mogli pristupati podatcima i raditi upite na bazu dizajnirano je interaktivno sučelje koje omogućuje vizualizaciju podataka . Nadalje ovaj servis omogućuje integraciju s ostalim Azure servisima poput Azure Stream Analytics, Power Bi i **Event Hub-a** što korisnicima pruža dodatnu mogućnost povezivanja i analize podataka iz različitih izvora. Važno je spomenuti kako je također riječ o PaaS platformi što korisnicima omogućuje da se usmjere na analizu i obradu podataka, a ne na upravljanje i održavanje infrastrukturom. Naplata se vrši prema količini podataka koje korisnici pohranjuju i pretražuju, uz nekoliko cjenovnih planova koji su definirani prema potrebama korisnika (21).

5.7. Storage Account

Storage Account je usluga unutar Microsoft Azure platforme se najviše koristi za pohranu podataka, dakle koristi se za upravljanje podatcima i pohranu istih. Ovaj resurs omogućuje pohranu različitih vrsta podataka što uključuje tekstualne i binarne datoteke, datoteke za sigurnosne kopije, može se koristiti i za trajnu pohranu virtualnih mašina, Docker kontejnera i sličnih virtualizacijskih opcija. Podatci se unutar Storage Account-a mogu pohraniti u različitim vrstama spremnika koji uključuju:

- Blobs (eng. *Containers*) - namijenjeni su pohrani velikih količina binarnih podataka kao što su slike, videozapisi, zvukovi ili virtualni strojevi. Blobs se mogu koristiti za statički sadržaj koji se često preuzima ili za dinamički sadržaj koji se dinamički generira i sprema.
- Datoteke (eng. *File shares*) - služe za pohranu podataka sličnih onima koji se nalaze u lokalnim datotečnim sustavima. Ova vrsta spremnika podataka omogućuje organiziranje datoteka u mape i pristupanje njima kao što je to uobičajeno kod lokalnih datotečnih sustava.
- Tablice (eng. *Tables*) - omogućuju pohranu strukturiranih podataka kao što su tablice podataka s nekoliko redaka i stupaca. Tablice se mogu koristiti za brzo spremanje i dohvaćanje podataka koji se često mijenjaju.
- Redovi (eng. *Queues*) - namijenjeni su pohrani niza poruka koje su nastale kao produkt obrade događaja (Event Processing). Koriste se za razmjenu poruka između aplikacija i omogućuju brzu i jednostavnu razmjenu poruka između korisnika.

Kao i svaki prethodno opisan servis, Storage Account nudi zanimljive značajke poput visoke dostupnosti i pouzdanosti što proizlazi iz automatskog repliciranja podataka u više regija, čime se zapravo čuvaju podatci u slučaju kvara na jednoj od Azure regija. Uz spomenute značajke bitno je naglasiti kako ova usluga uključuje brojne sigurnosne značajke poput autentikacije, autorizacije, enkripcije i zaštite od DDOS (eng. *Distributed Denial-Of-Service*) napada. Naplata ove usluge ovisi o više parametara, pojednostavljeno ovisi o vrsti korištenog spremnika, količini podataka, zahtjevima za pristup podatcima, prometu podatcima i razini skladištenja podataka (eng. *tier*). Unutar Storage Account-a postoje dvije razine skladištenja podataka:

- **Hot** – koristi se za podatke koji se često koriste i kojima se često pristupa. Ova razina skladištenja podrazumijeva višu cijenu pohrane u odnosu na Cool razinu, ali nudi brži pristup podacima i nižu latenciju. Ova razina skladištenja je idealna za podatke koji se moraju brzo dohvatiti i obrađivati, poput aplikacijskih datoteka i baza podataka.
- **Cool** – koristi se za podatke kojima se rijetko pristupa ali ih je bitno zadržati zbog arhiviranja ili zakonskih propisa. Ova razina skladištenja nudi niže cijene pohrane u odnosu na Hot razinu, ali pristup podacima može biti sporiji i s višom latencijom. Cool tier je idealan za pohranu podataka koji se često ne koriste, ali su važni za zadržavanje, poput sigurnosnih kopija i arhivskih datoteka.

Korisnicima se uvijek ostavlja mogućnost prebacivanja podataka između spomenutih razina pohrane ovisno o potrebama korisnika u određenom trenutku. Ova opcija dodatno omogućuje korisnicima da optimiziraju troškove pohrane podataka, smanjujući cijene pohrane za podatke koji se rijetko koriste i povećavajući brzinu pristupa za podatke koji su često u uporabi (22). Možemo zaključiti kako je riječ o jako bitnom servisu unutar Microsoft Azure platforme koja je fleksibilna i skalabilna što joj omogućuje da se koristi za različite primjene poput pohrane za web i mobilne aplikacije, analizu velikih podataka (eng. *Big Data Anaylsis*), sigurnosno kopiranje, arhiviranje i slične uloge.

6. Unreal Engine

Unreal Engine je jedan od najpoznatijih alata za razvoj video igara i aplikacija u stvarnom vremenu. Objavljen je 1998. godine kao alat za razvoj igara samo na računalima, ali se s vremenom razvio i na druge platforme poput igraćih konzola i mobilnih uređaja, trenutno ga održava tvrtka Epic Games. Glavna značajka ovog alata je stvaranje 3D prostora i objekata, a dodatno omogućuje realistične vizualne efekte poput sjena, refleksije, dinamičkog osvjetljenja kao i slojevite teksture što omogućuje stvaranje složenih materijala za virtualne objekte. Jedna od zanimljivih mogućnosti ovog alata je stvaranje događaja, odnosno programiranje istih, pa tako možemo stvoriti događaje poput kolizija, interakcije između objekata, animacije i slično. Nadalje, Unreal Engine posjeduje sustav za ponavljanje događaja pa tako možemo napraviti kompletan scenarij, sastavljen od velikog broja većih i manjih događaja, koji se isprepliću, mijenjaju ovisno o zadanim uvjetima i slično. Uz kompleksne mogućnosti stvaranja 3D objekata i virtualnog svijeta, izrade različitih scenarija i događaja, Unreal Engine pruža mogućnost programiranja uz pomoć C++, što omogućuje i implementaciju dodatnih alata i protokola poput slanja MQTT poruka. S obzirom na to da smo se odlučili primijeniti koncept digitalnih dvojnika na proces proizvodnje u tvornici, u nedostatku fizičke tvornice, iskoristili smo ovaj alat kako bi smo kreirali virtualnu tvornicu sastavljenu od robotskih ruku, proizvodnih traka, reciklažnih vrata i ostali komponenti (23). Osim stvaranja 3D objekata i izgleda tvornice, isprogramirali smo različita ponašanja i scenarije unutar tvornice i implementirali senzore koji šalju vrijednost svojih očitanja na jasno definirane teme unutar MQTT brokera (24). Na ovaj način smo relativno uspješno stvorili scenarij prave tvornice, odnosno proizvodne linije koja nam temeljem brojnih senzora može u stvarnom vremenu slati podatke o stanju u tvornici i brojnim proizvodnim parametrima. Zapravo smo stvorili digitalnog dvojnika temeljenog na virtualnoj tvornici koja je dizajnirana i programirana u alatu za razvoj video igara.

7. Programski okvir Angular

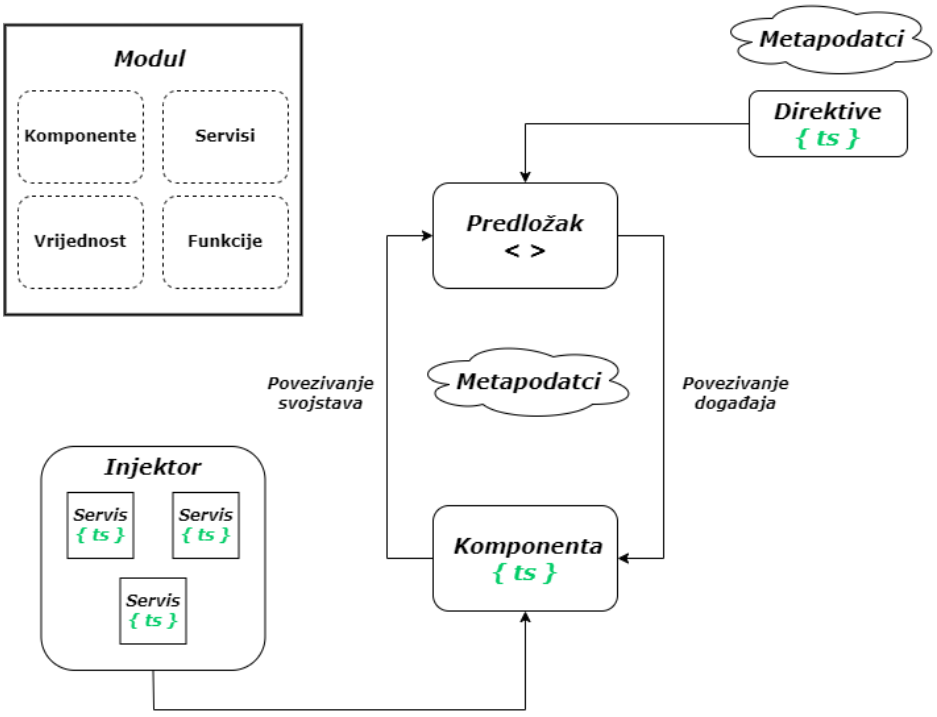
Angular je moderan programski okvir (eng. *framework*) za razvoj web aplikacija, riječ je o okviru otvorenog tipa (eng. *open-source*) koji je početno razvijen unutar kompanije Google, a trenutno ga održava zajednica web programera na čelu sa zaposlenicima spomenute tvrtke. Projekt je inicijalno razvijen 2010. godine pod nazivom AngularJS, ali je zbog nedostataka u performansama redizajniran i izgrađen pod nazivom Angular 2. Neke od promjena u odnosu na inicijalnu verziju uključivale su modernu arhitekturu, poboljšanje performansi i podršku za mobilne uređaje. Uz sve navedeno, uvedeno je korištenje novog programskog jezika pod nazivom TypeScript, koji je napravljen kao „ekspanzija“ JavaScripta i nudi mogućnosti poput statičkog definiranja tipova za svaku varijablu i izraz korišten unutar aplikacije, što smanjuje mogućnost pogreški i optimizira performanse. Verzija Angular 2 je predstavljena 2016. godine, a od tada se konstantno razvija, zadnja verzija Angular 16 je puštena na korištenje u svibnju 2023. godine, u ovom projektu koristili smo verziju Angular 13, puštenu na korištenje u studenom 2021. godine (25).

Glavna značajka ovog framework-a su njegove komponente, modularne samostalne jedinice pomoću kojih se dizajnira izgled i ponašanje korisničkog sučelja (eng. *user interface*). Svaka komponenta se sastoji od predloška (eng. *template*) koji definira izgled i strukturu komponente i klase (eng. *class*) zadužene za logiku i podatke prikazane na komponenti. Komponente su dizajnirane tako da budu ponovno iskoristive, tako da ako imamo neku komponentu poput padajućeg izbornika (eng. *dropdown*) možemo ga ponovno iskoristiti tako da mu proslijedimo potrebne podatke. Struktura aplikacije je uobičajeno definirana tako da imamo glavnu komponentu (eng. *root*) koja sadrži sve ostale komponente koje međusobno mogu komunicirati (26). U ovoj strukturi imamo roditeljske komponente (eng. *parent*) koje mogu prosljeđivati (eng. input) podatke u svoje dječije komponente (eng. *child*), na sličan način dječije komponente mogu vraćati (eng. *output*) podatke u roditeljske komponente. Generalno, komponente su zamišljene kao nezavisne jedinice koje mogu biti samostalne, ponovno upotrebljive, sposobne međusobno komunicirati, jednostavne za održavanje i pogodne za skaliranje. Dvije jako važne stvar za funkcioniranje Angular aplikacija su direktive i servisi koji dodatno olakšavaju programerima razvijanje održivih i skalabilnih aplikacija. Angular servisi omogućuju dijeljenje koda i logike između Angular komponenti, najčešće je riječ o čestim funkcionalnostima poput pristupa podataka, zapisivanja logova ili pak autentikaciji.

Servisi su najčešće implementirane unutar neke klase kao skup metoda ili svojstva koje mogu biti pozivane ili pristupane iz drugih komponenti, što smanjuje repetitivnost koda. S druge strane, Angular direktive se koriste u svrhu implementacije ponašanja i modifikacije izgleda HTML elemenata u samoj aplikaciji, a sve to radimo u svrhu dizajniranja ponovno iskoristivih komponenti, manipulacije DOM-om ili pak u svrhu osluškivanja promjena u samoj aplikaciji (eng. *event listeners*). Unutar Angular okvira postoje dvije vrste direktiva, koje uključuju:

- Strukturalne direktive – koriste se za modifikaciju DOM strukture tako da da dodajem ili brišemo elemente temeljeno na definiranim uvjetima ili pak prolazeći kroz niz stavki, primjer ovih direktiva su **ngFor** and **ngIf**.
- Atributne direktive – koriste se za modifikaciju izgleda i ponašanja postojećih elemenata, primjer ovih direktiva su **ngClass** i **ngStyle**.

Kako bih ujedinili spomenute funkcionalnosti, unutar Angular framework-a postoje moduli. Koristimo ih kako bi organizirali i grupirali povezane komponente, direktive i servise u jedinstvenu cjelinu koju možemo koristiti unutar jedne ili više aplikacije. Kako bih definirali module koristimo dekorator „@NgModule“ pomoću kojeg označavamo komponente, direktive i servise koje određeni modul uključuje (27).Također možemo definirati vanjske ovisnosti (eng. *dependency*) ili biblioteke neophodne za funkcioniranje nekog modula, što u konačnici omogućava enkapsulaciju i izolaciju različitih dijelova aplikacije (Slika 3).



Slika 3. Temeljni elementi Angular aplikacije

Angular je izrazito snažan i fleksibilan okvir za izradu skalabilnih web aplikacija izgrađenih na temelju TypeScript-a koji omogućuje izradu modularnog i ponovno iskoristivog koda. Jedan od glavnih prednosti Angular okvira je velika i aktivna zajednica korisnika koja pruža potporu, daje prijedloge, održava sam okvir i pruža programerima sigurnost kako će isti ostati aktualan i kompatibilan s najnovijim standardima. Navedeni okvir je izvanredan izbor za programere koji žele izgraditi kompleksne i skalabilne aplikacije usmjerene na modularan i ponovno upotrebljiv kod. Iz spomenutih razloga odlučili smo koristiti ovaj framework za izradu web aplikacije pomoć koje ćemo nadzirati, analizirati i upravljati digitalnim dvojnicima.

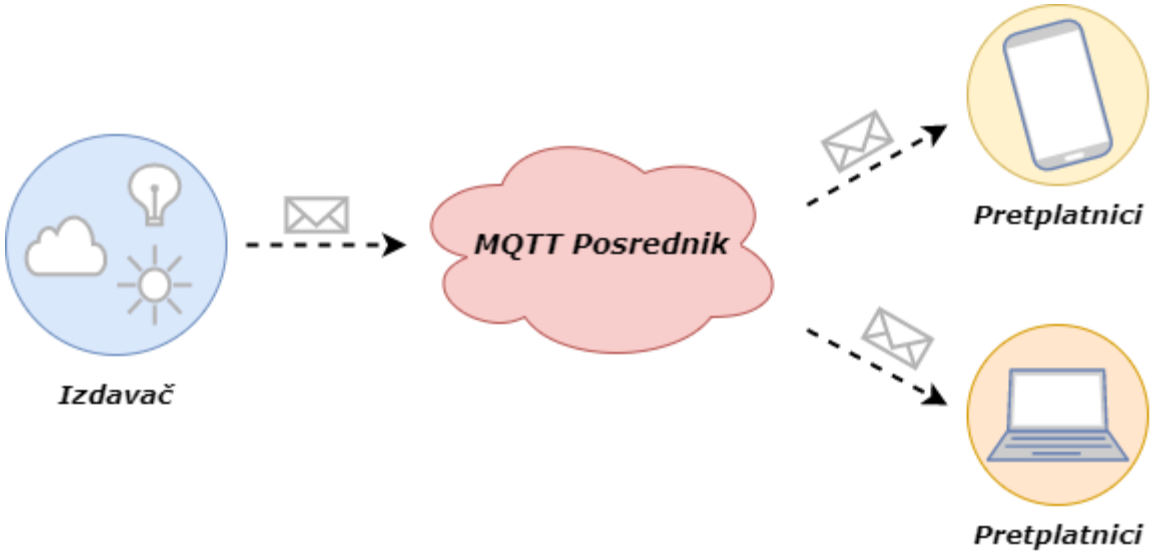
8. MQTT

MQTT (Message Queuing Telemetry Transport) je protokol koji je prvotno namijenjen za komunikaciju između IoT uređaja, međutim danas ima dosta širu primjenu. Riječ je o protokolu koji je izrazito jednostavan i lagan za implementaciju, a pruža učinkovitu i pouzdanu razmjenu poruka između IoT uređaja i aplikacija. MQTT je razvijen 1999. godine od strane poznate svjetske kompanije IBM, a inicijalna svrha je bila upravljanje i nadzor industrijskim procesima. Ubrzo je postao javno dostupan i počeo se primjenjivati u različite svrhe te u konačnici postao jednim od najpopularnijih protokola za komunikaciju u IoT svijetu gdje je bitna jednostavnost, niska latencija i pouzdanost.

Ovaj protokol se temelji na publish-subscribe arhitekturi (Slika 4) koja je sastavljena od 3 komponente:

- Izdavač (eng. *publisher*) – šalje poruke na definiranu temu (eng. *topic*)
- Posrednik (eng. *broker*) – prima poruke poslane na definiranu temu i prosljeđuje navedene poruke svim pretplatnicima koji imaju pretplatu na određenu temu.
- Pretplatnik (eng. *subscriber*) – prima poruke koje su poslane na definiranu temu

Tema koristimo kao naziv temeljem kojih identificiramo poruke, a strukturirana je u više razina, od kojih je svaka razina odvojena znakom kose crte „/“. Na primjer ova tema „demo/vibration1/“ predstavlja nekakav senzor za vibraciju temeljem kojeg on šalje očitane vrijednost, a da bih pretplatnik mogao vidjeti očitanja navedenog senzora, mora ostvariti pretplatu na tu istu temu. Važno je naglasiti kako pretplatnici mogu koristiti univerzalnu temu „/#“ kako bih čitali sve poruke poslane na bilo koju temu unutar određenog posrednika. Naravno, ovo predstavlja sigurnosni problem, ali MQTT pruža mogućnost autentikacije i autorizacije izdavača i posrednika kao i slanje komunikacije kroz sigurnosne protokole poput TLS-a (Transport Layer Security). Valja napomenuti kako MQTT pruža mogućnost zadržavanja poruka na brokeru, dakle broker se može konfigurirati tako da pretplatnici mogu primiti poruku samo ako su u trenutku slanja pretplaćeni na određenu temu, što može predstavljati problem u slučaju da se pretplatnik izgubi vezu s posrednikom. Stoga je omogućeno zadržavanje poruka (eng. *retention*) u posredniku, kako bih navedene poruke pretplatnici mogli pročitati nakon ponovne uspostave veze s posrednikom (28).



Slika 4. Komunikacijski proces unutar MQTT protokola

Danas postoje brojni MQTT davatelji usluga, od kojih ćemo posebno istaknuti **Mosquitto** i HiveMQ. Za potrebe ovog rada smo koristili Mosquitto zbog toga što je jednostavan i fleksibilan, a ova verzija MQTT-a je kompatibilna s Linux i Windows platformama. Jedna od zanimljivih značajki koju nude spomenuti davatelji usluga je integracija MQTT protokola s WebSocket-om koji omogućuje dvosmjernu komunikaciju između preglednika i poslužitelja. Standardan način ostvarivanja komunikacije putem MQTT je temeljen na TCP/IP protokolima, ali je takve protokole nezgodno koristiti u web pregledniku. Stoga su razvijene biblioteke (eng. *library*) za JavuScript poput Paho i MQTT.js koje omogućuju MQTT komunikaciju temeljenu na WebSocket-u, a sve to u stvarnom vremenu bez potrebe za dodatnim slojevima aplikacije ili nekakvim softverskim rješenjima (29).

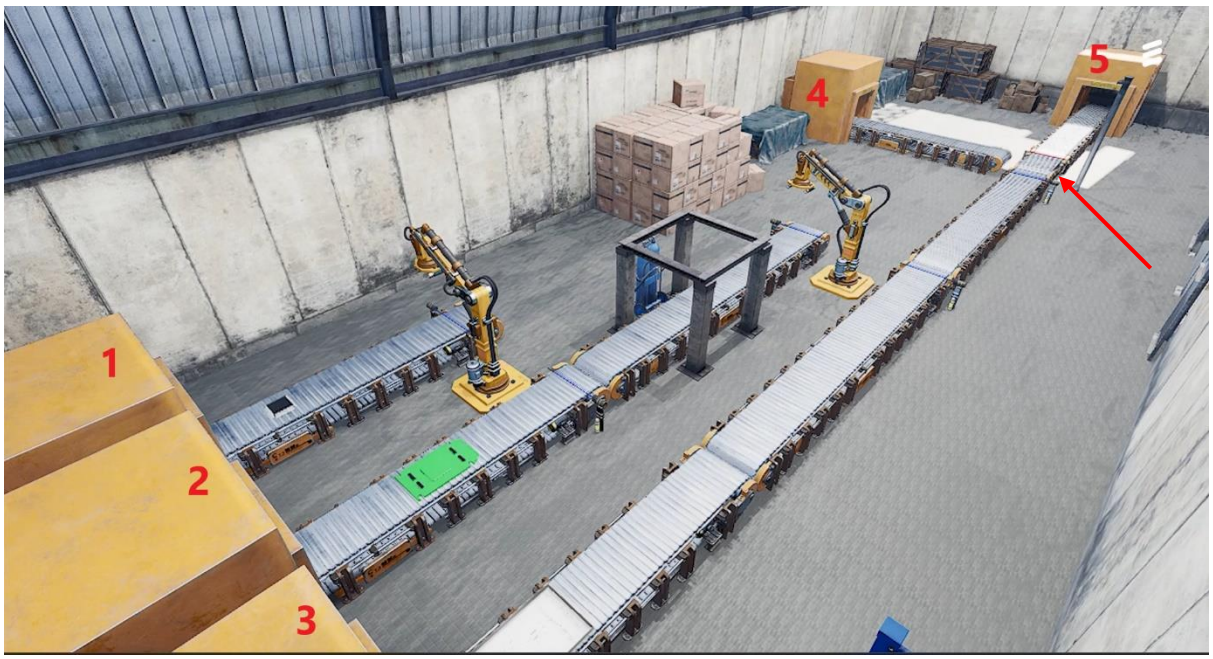
9. Unreal Engine i izrada digitalnog dvojnika

Analiza je jedan od prvih koraka u implementaciji koncepta digitalnog dvojnika u odnosu na neki sustav. Potrebno je detaljno proučiti određeni sustav kako bismo identificirali sve komponente, senzore i relevantne podatke za izradu digitalnog dvojnika. S obzirom na to da smo odlučili implementirati digitalnog dvojnika temeljenog na tvornici za proizvodnju matičnih ploča, neophodno je upoznati se s navedenom materijom, odnosno proizvodnim procesom. Kao što sam spomenuo u prethodnim dijelovima rada, u nedostatku prave tvornice, odlučili smo napraviti virtualnu tvornicu uz pomoć Unreal Engine-a, alata za izradu video igrica. Navedeni potez može biti nejasan i dvosmislen s obzirom na to da želimo napraviti digitalnog dvojnika na temelju nečeg što je već virtualno. Međutim, da bih se proveo proces integracije digitalnog dvojnika na stvarnom sustavu, morate imati pristup istom, upoznati se detaljno sa svim njegovim značajkama i informacijama koje mogu biti sigurnosno osjetljive za samog vlasnika, odnosno kompaniju. Nadalje, inicijalna ulaganja u infrastrukturu (IoT uređaje, umreženost, potrebne prilagodbe) potrebnu da bih se navedeni proces izvršio mogu biti izrazito skupa. Stoga, ako uzmemo u obzir činjenicu da se koncept digitalnog dvojnika već primjenjuje u nekoliko proizvodnih tvrtki i sustava, nema potrebe istraživati mogućnost implementacije digitalnog dvojnika na stvarnom sustavu. S obzirom na to da je cilj ovog rada integracija koncepta digitalnog dvojnika uz pomoć gotovih rješenja dizajniranih od strane Microsoft Azure Cloud platforme i s obzirom na izazove koje predstavlja implementacija na stvarnom sustav, jasno je zašto smo se odlučili ovu primjenu napraviti na virtualnom sustavu, odnosno virtualnoj tvornici.

9.1. Unreal Engine virtualna tvornica

Nakon što smo analizirali i prikupili sve potrebne informacije vezane uz tvornicu i proces proizvodnje matičnih ploča, možemo se usmjeriti na dizajniranje same tvornice. S obzirom na to da je standardni proces proizvodnje poprilično kompleksan i uključuje niz složenih radnji, isti smo pojednostavili tako da smo se fokusirali na finalni proces sastavljanja i kontrole gotovih proizvoda. Unreal Engine je izrazio snažan alat korišten od strane velikog broja korisnika i tvrtki, stoga posjeduje brojne 3D objekte i gotove komponente koje smo iskoristili u svrhu izrade virtualne tvornice. Spomenute objekte smo upotrijebili kako bih napravili

dostojan prikaz tvornice, koji se sastoji od proizvodnih vrata, robotskih ruku, proizvodnih traka, hladnjaka i segmenta za kontrolu kvalitete. Na slici (Slika 5) možete vidjeti kako izgleda naša virtualna tvornica, uz spomenute objekte, dodali smo i neke dodatne objekte (zidove, krov, kutije) kako bih ista izgledala što realističnije.



Slika 5. Prikaz virtualne tvornice za proizvodnju matičnih ploča, isječak iz Unreal Engine simulacije

Početak proizvodnje polazi od proizvodnih vrata označenim brojem 1, kroz ova vrata izlaze čipovi, koji putuju proizvodnom trakom do robotske ruke. Kroz vrata broj 2, izlaze pločice na koje je potrebno montirati čipove, a proces instalacije odrađuje prva robotska ruka. Nakon što je proces instalacije završen, matične ploče putuju do hladnjaka uz pomoć pokretne trake broj 2. Paralelno s tim, iz vrata broj 3 izlaze kutije za pakiranje, robotska ruka broj 2 preuzima matične ploče nakon hlađenja, te ih postavlja na kutije koje se nalaze na pokretnoj traci broj 3. Gotove matične ploče dolaze do kontrolnog segmenta (označen crvenom strelicom), gdje kamera (Slika 6) provjerava njihovu kvalitetu, ako matične ploče zadovoljavaju uvjete, bit će proslijeđene na vrata broj 5, a u suprotnom šalju se na recikliranje na vrata broj 4 . Tijekom instalacije može doći do oštećenja matičnih ploča uslijed visokih temperatura, što utječe na kvalitetu istih. Stoga je matične ploče potrebno dovoljno ohladiti uz pomoć hladnjaka ili na način produžimo vrijeme instalacije čipa na samu ploču. Što je proizvodnja matičnih ploča veća, manje vremena provode na pokretnim trakama, stoga je veća šansa da će njihova kvaliteta biti manja.

Svaki od navedenih objekata posjeduje virtualne IoT uređaje, posebice senzore kako bi nam javljali temperaturu uređaja, temperaturu matičnih ploča, potrošnju energije, brzinu proizvodnje, položaj robotskih ruku i kvalitetu samih čipova. Svaki od navedenih senzora šalje očitane vrijednosti prema definiranom vremenskom intervalu, a šalje ih uz pomoć MQTT servisa koji je implementiran unutar Unreal Engine-a.

Slika 6. Segment tvornice zadužen za kontrolu kvalitete matičnih ploča

Osim što smo uz pomoć Unreal Engine alata kreirali cijelu tvornicu, scenarij i omogućili slanje podataka od strane virtualnih senzora, uspjeli smo isprogramirati različite obrasce ponašanja pomoću kojih možemo upravljati proizvodnim procesom. Uz to što Unreal pruža mogućnost slanja MQTT poruka, isto tako ima mogućnost primanja poruka, odnosno pretplate na određene teme, tako možemo izvršiti neki obrazac ponašanja kad primimo određenu poruku. Na ovaj način možemo upravljati brzinom proces proizvodnje, potrošnjom energije, temperaturom, zapravo smo postigli svu onu kontrolu koju zapravo imaju radnici ili nekakav nadzornik u stvarnom proizvodnom procesu. Naprimjer, brzina proizvodnje matičnih ploča je izračunata na temelju određene matematičko-logičke formule, uz pomoć MQTT servisa, možemo poslati poruku i prilagoditi brzinu proizvodnje prema željenoj vrijednosti.

Nakon što smo kreirali tvornicu, definirali način na koji radi, implementirali senzore i aktuatore koji nam daju uvid u tvornicu kao i mogućnost upravljanja istom, možemo se usmjeriti na modeliranje i izradu digitalnog dvojnika.

9.2. Kompozitne metrike i intentovi

Kompozitna metrika (eng. *composite metric*) je termin proizašao iz engleskog jezika, a definira se kao metrika (mjera) čija se vrijednost definirana uz pomoć matematičko-logičke formule. Parametri koji ulaze u izračun mogu uključivati vrijednosti elementarnih metrika (senzorni podatci) ili vrijednost neke druge kompozitne metrike. U našem slučaju, ovaj termin smo iskoristili u svrhu definiranja vrijednosti koja se računa na temelju jedne ili više senzorskih vrijednosti u kombinaciji s nekom drugom kompozitnom metrikom, a sam izračun se prilagođava prema okolnostima (eng. *use-case*) u kojima se koristi. Naprimjer, odaberemo segment tvornice, unutar kojeg se nalazi više uređaja, a svaki od navedenih uređaja troši određenu količinu energije te ima senzor pomoću kojeg mjeri navedenu potrošnju. Kako bih jednostavnije upravljali potrošnjom energije, definirat ćemo kompozitnu metriku nad spomenutim segmentom tvornice, a njena vrijednost će biti izračunata na temelju očitanih vrijednosti iz spomenutih senzora. S obzirom na to da digitalni blizanci pružaju mogućnost učenja iz podataka, jednostavno mogu izračunati koji dodatni čimbenici imaju utjecaj na potrošnju energije i u kojem omjeru. Temeljem ovih saznanja, digitalni blizanac zna na koje čimbenike i senzore mora utjecati kako bih u mogao utjecati na vrijednost kompozitne metrike. Složeniji primjer kompozitne metrike bi uključivao izračun učinkovitosti tvornice, koja može ovisiti o potrošnji energije, količine izrađenih proizvoda i kvaliteti istih. Svaki od tih parametar može biti elementarna vrijednost, poput količine obrađenih proizvoda (senzor koji mjeri protok proizvoda), ali i vrijednost neke druge kompozitne metrike (kompozitna metrika za potrošnju energije). Važno je naglasiti da sve vrijednosti koje ulaze u izračun određene kompozitne metrike moraju biti mjerljive, jer ne možemo upravljati nečim ako ne možemo izmjeriti njegovu vrijednost.

Drugi pojam koji smo definirali u našem primjeru digitalnih blizanaca se zove namjera (eng. *intent*), a napravljen je u svrhu kako bih korisnici mogli upravljati kompozitnim metrikama, odnosno definirati vrijednost na koju žele postaviti kompozitnu metriku. U prvom koraku, korisnik definira kompozitnu metriku za izračun potrošnje energije, a uz pomoć intenta definira da ta vrijednost ne smije prijeći iznos veći od 30,000 kWh mjesečno. S obzirom na to da digitalni blizanac poznaje formulu za izračun kompozitne metrike kao i ostale čimbenike koji utječu na istu, jednostavno može prilagoditi sve faktore kako bih postigao vrijednost definiranu intentom. Naprimjer, slanjem signala na različite aktuatore unutar tvornice, digitalni blizanac može utjecati na neke čimbenike, poput smanjivanja/povećanja temperature otvaranjem

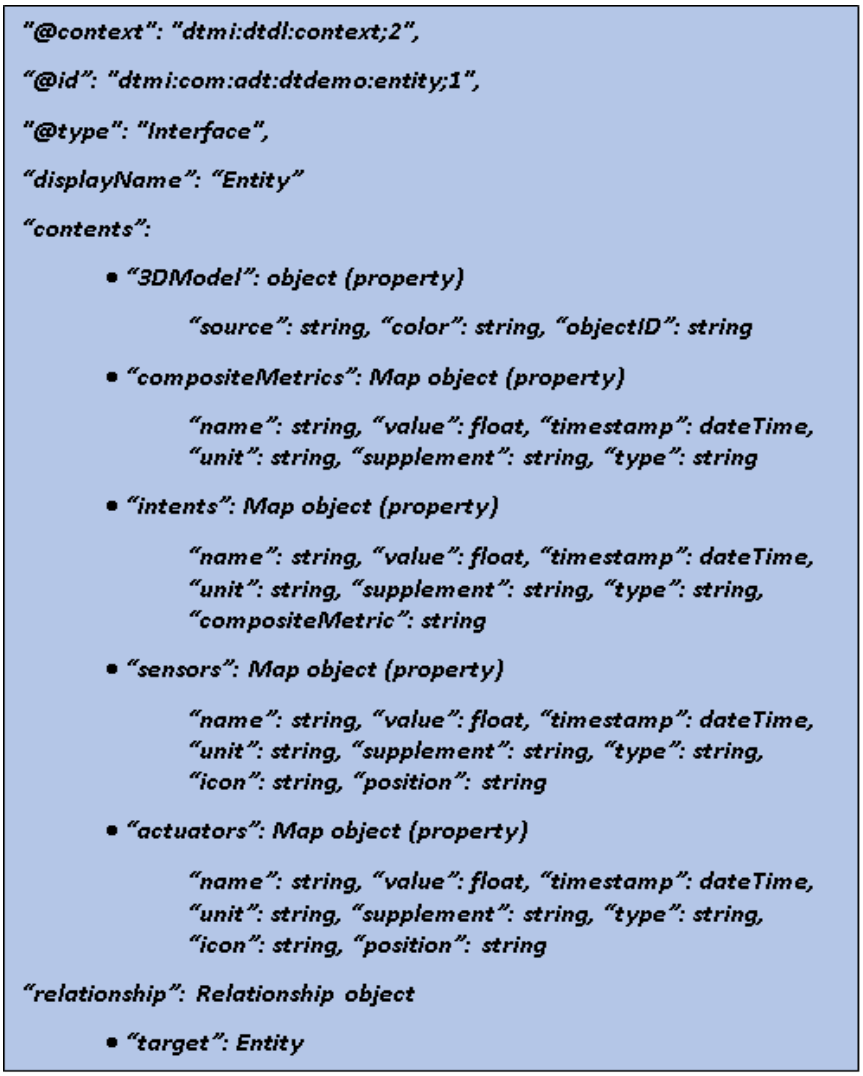
prozora ili prilagođavanje postavki nekog uređaja kako bih se smanjila potrošnja energije ili povećala u svrhu boljih performansi. Temeljem odluke poslane prema aktuatorima, stanje unutar tvornice se mijenja, senzori bilježe nove vrijednost, a krajnja vrijednost kompozitne metrike se postepeno mijenja ovisno o novo očitanim vrijednostima. Ovaj ciklus se ponavlja sve dok vrijednost kompozitne metrike ne bude kompatibilna s onom vrijednosti koju je korisnik definirano intentom. Isto tako, korisnik može napraviti i drugi intent gdje definira da želi povećati proizvodnju tvornice za 20%, ali da se pritom ne poveća mjesečnu potrošnju energije za više od 15%. Temeljem svih metoda i koncepata, digitalni blizanac poznaje na koje vrijednosti točno treba utjecati i u kojem omjeru kako bih se postigao željeni rezultat, a da pritom ne gubi vrijeme i novac na manualno podešavanje proizvodnog procesa. Naravno, nije moguće povećati proizvodnju, bez utroška energije, stoga će kvaliteta proizvoda sigurno biti manja, ali na ovaj način korisnik ima mogućnost „pametne“ optimizacije tvornice prema potrebama u određenom trenutku.

S obzirom na dugotrajnu i kompleksnu proceduru koju je potrebno slijediti kako bih se implementirali navedeni koncepti, odlučili smo pojednostaviti ovaj proces tako da unaprijed definiramo što sve ulazi u izračun metrike (uključujući dodatne čimbenike) kao i način na koji će digitalni blizanac utjecati na njenu vrijednost. Premirani nedostatak ovog poteza je to što digitalni blizanac neće naučiti koji dodatni čimbenici i u kojem omjeru utječu na vrijednost metrike, a same odluke koje donese kako bih utjecao na vrijednost kompozitne metrike neće biti temeljene na učenju iz podataka, nego na unaprijed definiranim (eng. *hard-coded*) procedurama i izračunima. S druge strane, ovaj način implementacije je dovoljno snažan kako bih se demonstrirao značaj i mogućnosti koje digitalni blizanci mogu ponuditi kroz kompozitne metrike ili neke druge značajke koje primjenom umjetne inteligencije mogu optimizirati upravljanje nekim sustavom.

9.3. Integracija digitalnog dvojnika

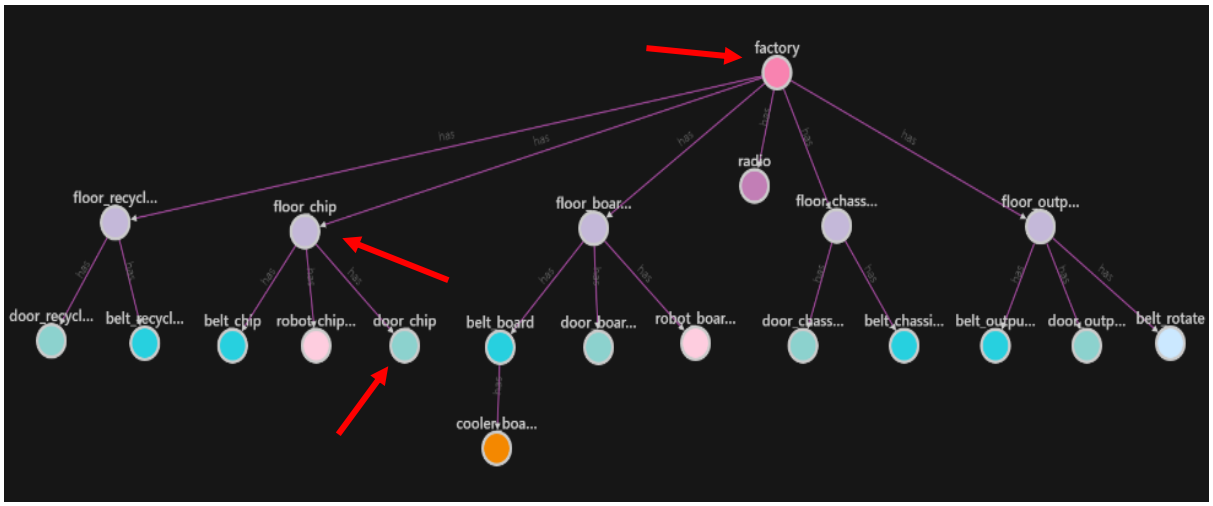
Kako bih započeli s procesom izrade digitalnog dvojnika, potrebno je prvo napraviti modele, a na temelju njih izrađujemo instance koje imaju definirane parametre, nude mogućnost povezivanja odnosno stvaranja veza s drugim instancama. Odlučili smo kako svaki 3D objekt unutar Unreal Engine-a treba imati zasebnu digitalnu instancu, ujedno smo implementirali i digitalne instance koji nemaju vizualne 3D objekte unutar tvornice. Uloga ovih „nevidljivih

instanci“ je grupiranje istih prema njihovoj poziciji ili funkciji u tvornicu, a ujedno pruža mogućnost jedinstvenog nadzora i upravljanja nad „povezanim instancama“. Za modeliranje smo koristili DTDL format, koji nudi različite tipove podataka, mogućnost ugnježđivanja, povezivanja i nasljeđivanja, ali postoje i nekakva ograničenja i pravila koja trebamo poštivati prilikom modeliranja. Pri izradi modela, definirali smo glavni model (Slika 7.) kojeg smo nazvali Entitet (eng. *entity*), a koristimo ga kao temeljni model kojeg svi ostali modeli nasljeđuju. Dakle unutar njega su definirana svojstva poput ID-a, imena za prikaz, podatak vezanih za 3D objekte, lista kompozitnih metrika (eng. *composite Metrics*), lista intentova (eng. *intents*), liste senzora, liste aktuatora i mogućnost povezivanja (eng. *relationship*) s drugim instancama koje su nastale iz istog modela ili ga nasljeđuju. Na slici (Slika 7) možemo vidjeti kako izgleda struktura glavnog Entitet modela.



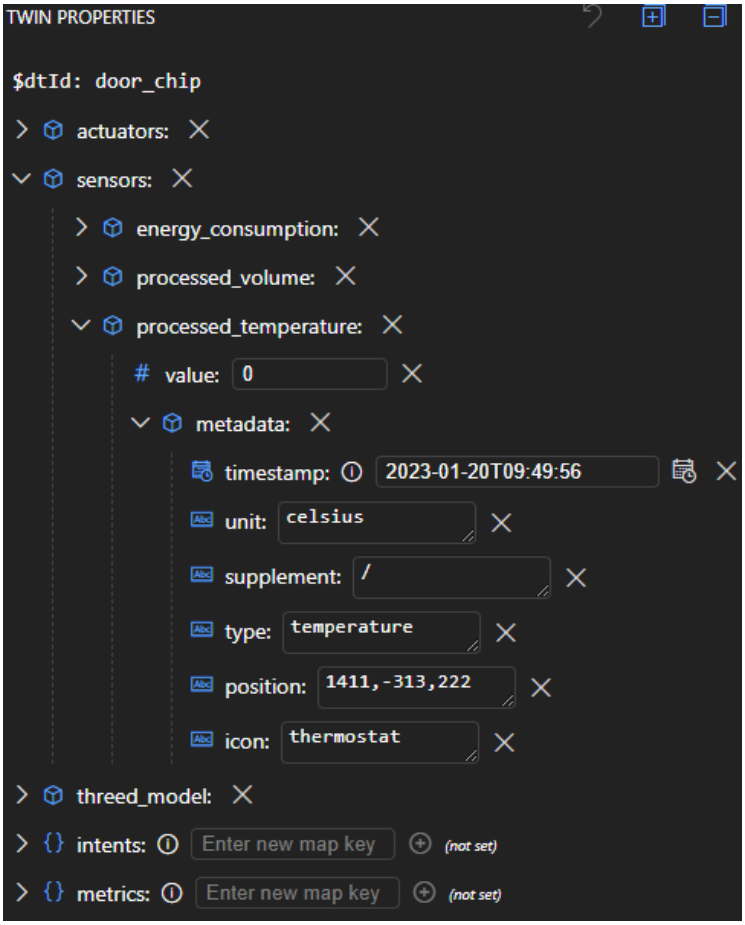
Slika 7. Struktura "Entitet" modela, definiranog korištenjem DTDL jezika. Korišteni su tekstualni tipovi podataka (*string*), numerički (*float*) i vremenski (*dateTime*) format.

Nakon definiranja modela, možemo napraviti zasebne instance i dodati sva potrebna svojstva vezana uz 3D objekte i parametre (kompozitne metrike, intentove, senzore i aktuatore). Unutar Azure Digital Twin resursa postoji alat nazvan „Azure Digital Twins Explorer“, koji korisnicima olakšava kreiranje modela, instanci i povezivanje istih. Ovaj alat pruža mogućnost učitavanja (eng. *upload*) dokumenta koji sadrži model definiran uz pomoć DTDL formata. Nadalje, alat nam omogućuje kreiranje instanci, popunjavanje svih potrebnih parametara te vizualno povezivanje instanci, odnosno kreiranje strukturu temeljena na vezama između njih. U našem slučaju smo napravili strukturu (Slika 8), gdje je glavna instanca tvornica (*factory*), zatim imamo 5 dječjih instanci *(floor_chip, floor_board, floor_recycle, floor_chassis, floor_output*) koje služe za grupiranje i radio toranj (*radio*) koji nije vidljiv na slikama zbog dimenzija. Svaka od instanci za grupiranje ima nekoliko svojih dječjih instanci koje zapravo predstavljaju 3D objekte unutar Unreal Engine tvornice.



Slika 8. Struktura digitalnog dvojnika virtualne tvornice, isječak sučelja iz aplikacije "Azure Digital Twins Explorer"

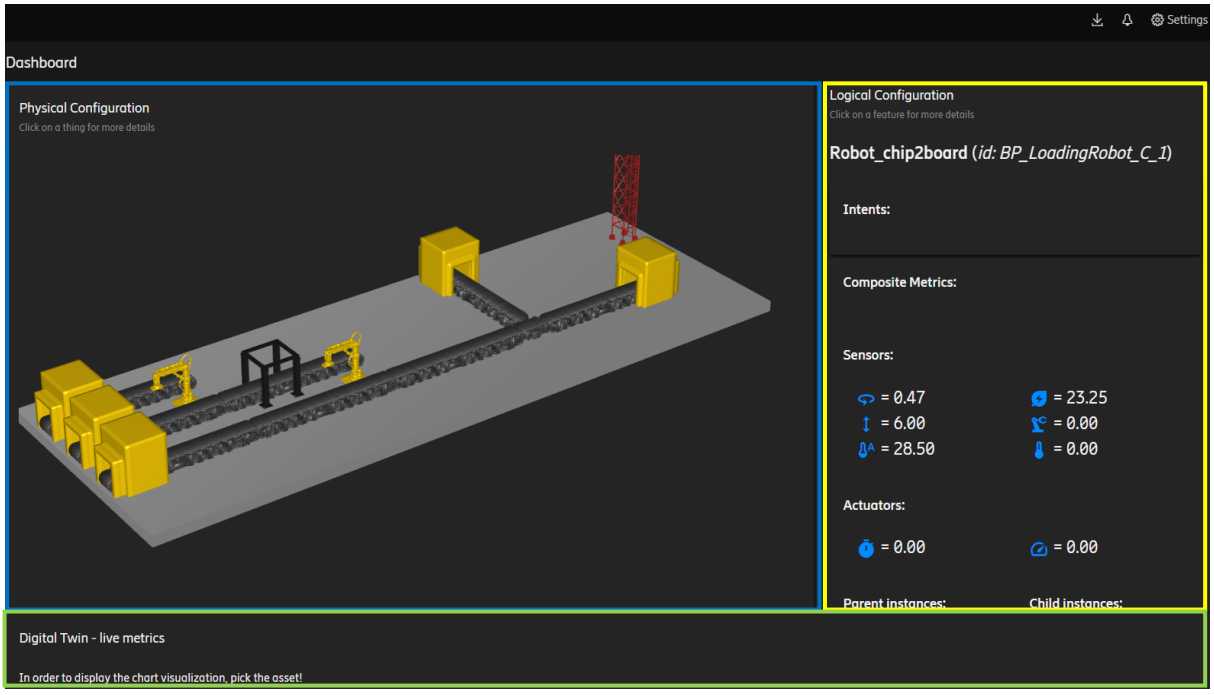
Ova struktura nam je omogućila stvaranje hijerarhije i davanje konteksta, a to znači da možemo vidjeti kako se temperaturni senzor (Slika 9) „*processed_temperature*“ nalazi unutar „*door_chip*“ instance, koja je dio „*floor_chip*“ instance, koja u konačnici vodi do „*factory*“ instance. Isto tako, na „*floor_chip*“ instanci možemo definirati kompozitnu metriku za potrošnju energije, čija će vrijednost ovisiti o vrijednostima senzora za potrošnju energije na njenim dječjim instancama („*belt_chip*“, „*robot_chip2board*“ i „*door_chip*“). Nadalje, na spomenutoj instanci možemo dodati i intent za upravljanje dodanom metrikom, što će nam indirektno omogućiti upravljanje potrošnjom energije. Ovo je primjer apstrakcije, gdje mi kao korisnici izdamo nekakvu želju ili zadatak, a digitalni dvojni samostalno konfigurira tvornicu kako bi se navedeni zadatak i ispunio.



Slika 9. Struktura instance "door_chip" prikazane u "Azure Digital Twins Explorer" aplikaciji

10. Angular UI

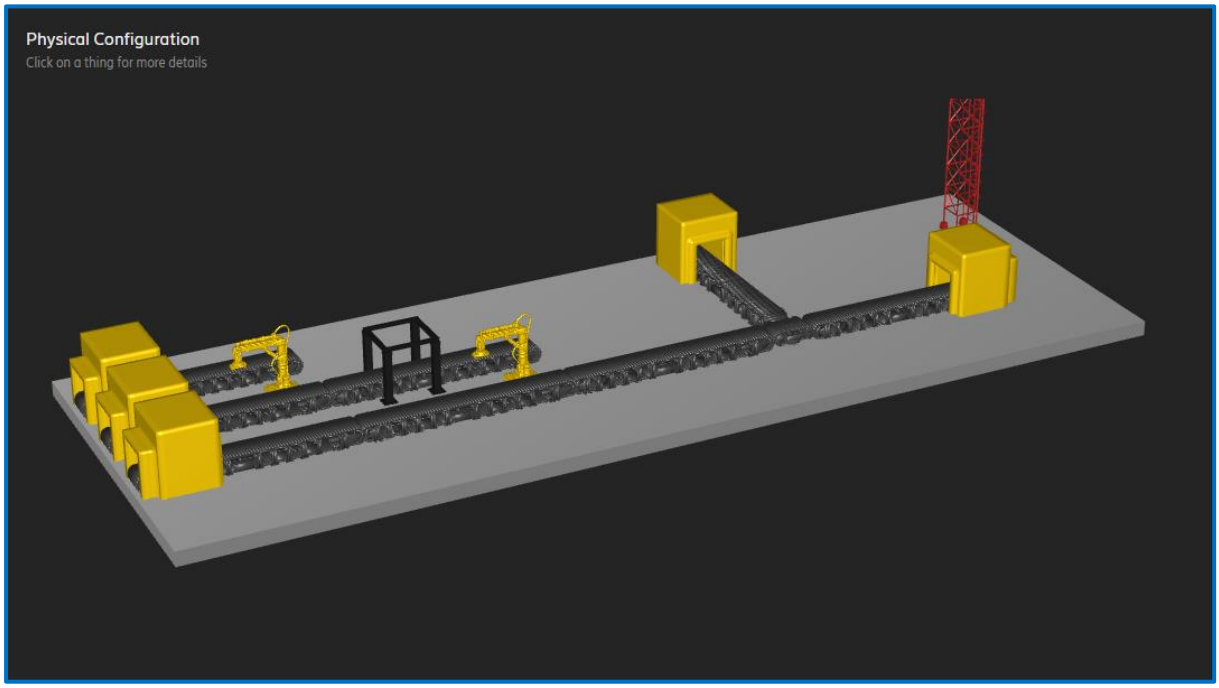
Angular UI je web aplikacija razvijena korištenjem istoimenog okvira, a koristi se u svrhu upravljanja i nadzora nad digitalnim dvojnikom. Prilikom izrade ove aplikacije, koristili smo razne gotove pakete i biblioteke kako bismo olakšali implementaciju određenih značajki. Tijekom razvoja, samo sučelje je evoluiralo i mijenjalo uloge. Prvobitno je bilo koncipirano kao alat za uređivanje digitalnih dvojnika, no kasnije smo ga prilagodili za nadzor, te na kraju smo stvorili hibridno sučelje koje omogućuje i nadzor i upravljanje digitalnim dvojnicima. Korisničko sučelje se sastoji od nekoliko Angular komponenti, pri čemu ćemo istaknuti tri glavne komponente (Slika 10), te jednu pomoćnu koja se koristi za konfiguraciju sučelja i promjenu postavki. Kako bismo olakšali instalaciju i korištenje ove aplikacije, odlučili smo je integrirati u Docker tehnologiju. To smo postigli stvaranjem Docker slike i pokretanjem aplikacije putem Azure Container App resursa.



Slika 10. Prikaz korisničkog sučelja Angular aplikacije dizajnirane u svrhu upravljanja i nadzora nad digitalnim dvojnikom

10.1. Fizička komponenta

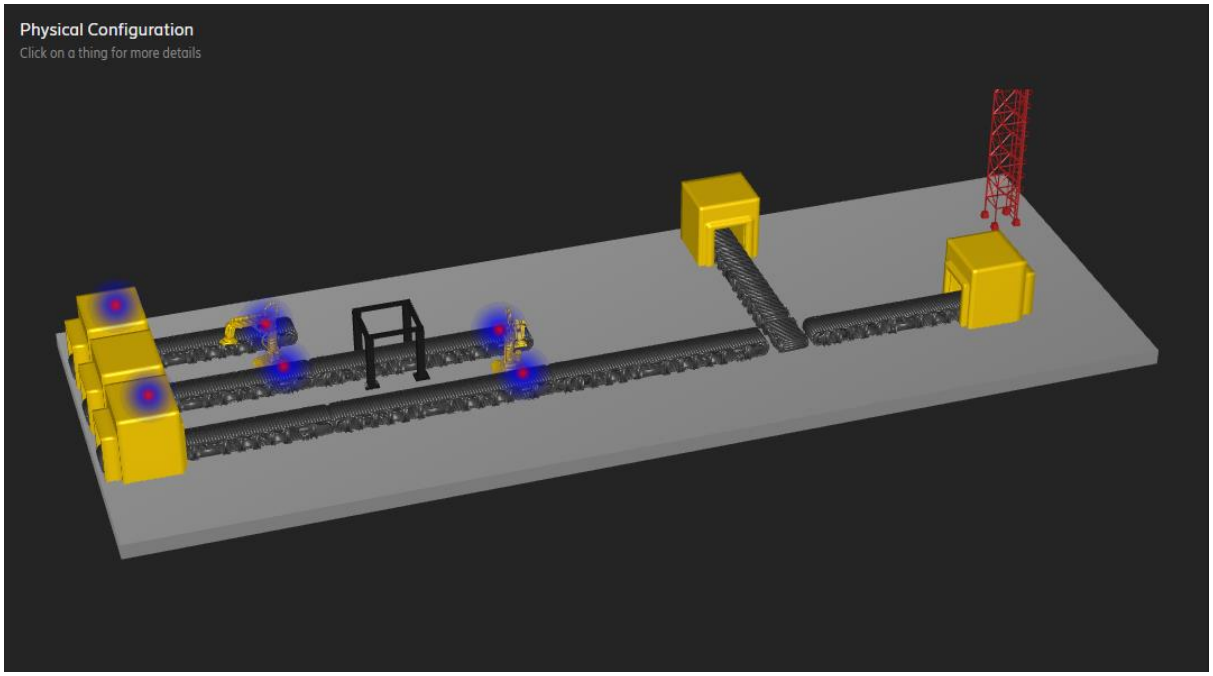
Centralna komponenta je ona na kojoj možemo vidjeti virtualni prikaz našeg digitalnog dvojnika nakon što smo ga učitali uz pomoć konfiguracijskih opcija. Naziv ove komponente je fizička konfiguracija (eng. *Physical configuration*) zbog toga što korisniku pomaže pri vizualizaciji digitalnih instanci, s obzirom na to da većina digitalnih instanci ima 3D objekt koji ih predstavlja. Ova komponenta je implementirana koristeći vanjsku biblioteku po imenu three.js, a omogućuje korisnicima kreiranje 3D svijeta i dodavanje različitih elemenata poput scena, 3D objekata, kamera, svjetla i drugih parametara. Na našem primjeru možete vidjeti 3D simulaciju (Slika 11) tvornice iz Unreal Engine-a, dakle imamo vrlo slične objekte poput proizvodnih vrata, traka, robotskih ruku, hladnjaka i radio tornja. Primijetit ćemo kako 3D simulacija rotira oko svoje osi, možemo kursorom prijeći preko 3D objekata, kako bih vidjeli njihovo ime, objekt na kojem se trenutno nalazi kursor će biti obojen u plavu boju. U trenutku kada se miš, odnosno kursor nađe iznad simulacije, rotacija će se zaustaviti. Korisnik ima opciju približavanja (eng. *zoom-in*) i udaljavanja (eng. *zoom-out*) simulacije kao i okretanja iste u željenom smjeru.



Slika 11. Isječak iz Angular aplikacije, prikaz Fizičke konfiguracije

U ovoj 3D simulaciji postoje dvije značajke koje postaju vidljive kada započnu pristizati senzorski podaci iz tvornice. Moći ćemo pratiti pokrete robotskih ruku tijekom prijenosa matičnih ploča te rotaciju trake za 90 stupnjeva, ovisno o tome je li matična ploča zadovoljila

potrebne standarde kvalitete. Druga značajka se odnosi na toplinski otisak (eng. *heatmap*) cijele simulacije koja se temelji na svim temperaturnim senzorima u tvornici. Na temelju očitanih vrijednosti sa senzora i njihove lokacije, možemo prikazati toplinski otisak. U tom prikazu, niže temperature će biti predstavljene plavom bojom, dok će više temperature biti prikazane crvenom bojom (Slika 12). Jedna od važnijih funkcija ove 3D simulacije jest da korisnik odabere jedan od željenih objekata tako da mišem klikne na objekt ispod kursora, što će rezultirati učitavanjem dodatnih informacija o objektu u komponenti koja se naziva logički panel (eng. *Logical panel*). S obzirom na to da ova radnja u sebi sadrži poziv prema HTTP funkciji koja učitava podatke iz Digital Twin resursa, može postojati manje vremensko kašnjenje (eng. *delay*) prije nego što se podatci učitaju.

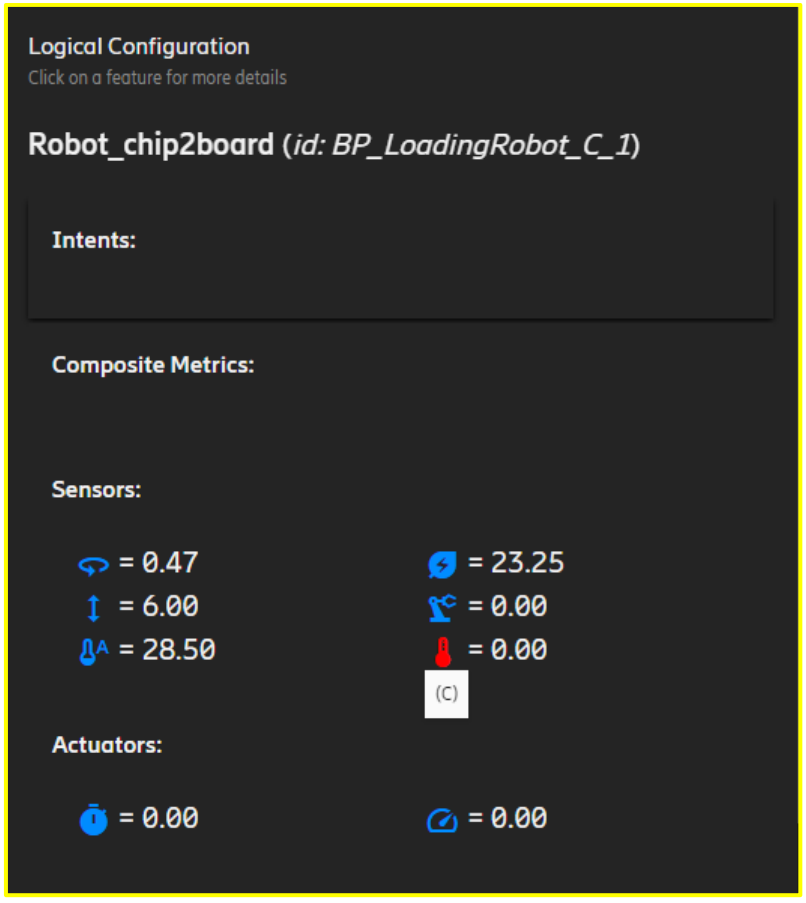


Slika 12. Isječak iz Angular aplikacije, izgled Fizičke konfiguracije u trenutku kad sučelje prima podatke iz tvornice, na slici je vidljiv i temperaturni otisak

10.2. Logička komponenta

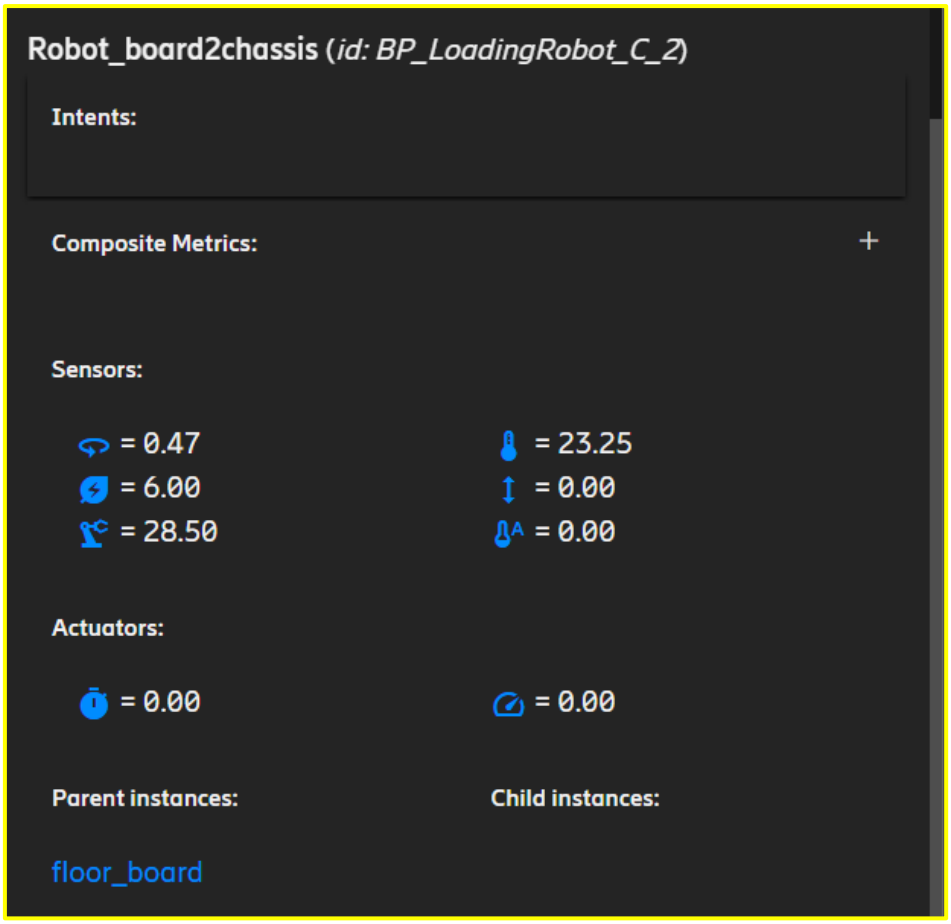
Logički panel je komponenta koja nam pruža detaljizirane informacije o nekoj instanci, dakle nakon što odaberemo neku instancu u 3D simulaciji, u ovoj komponenti možemo vidjeti podrobnije informacije o njoj (Slika 13). Dakle, na vrhu komponente se uvijek nalazi naziv odabrane instance, u zagradama se nalazi naziv 3D objekta koji je predstavlja, zatim po redu imamo listu intentova, ispod njih se nalazi lista kompozitnih metrika pa lista senzora i

naposljetku je lista aktuatora. Svaka instanca ne mora imati sve navedene parametre, dakle neka instanca može imati samo senzore, ili samo kompozitne metrike, s tim da intent parametri ovise o kompozitnim metrikama. S obzirom na to da intentovi služe u svrhu upravljanja kompozitnim metrikama, intent ne može postojati bez da na toj instanci ne postoji kompozitna metrika kojom on upravlja. Pored svakog od navedenih parametara možemo pronaći zadnju zabilježenu vrijednost, a ako postavimo kursor povrh svakog parametra ispisat će se mjerna jedinica koju taj parametar koristi.



Slika 13. Isječak iz Angular aplikacije, prikaz Logičke konfiguracije popunjene s parametrima povezanim uz robotsku ruku „*robot_chip2board*“

Ispod liste aktuatora, možemo pronaći veze odabrane instance s ostalim instancama, pa tako s lijeve strane možemo vidjeti instancu koja se smatra roditeljskom instancom (eng. *parent instance*), a s desne strane vidimo instance koje se smatraju dječjim instancama (eng. *child instance*) s obzirom na odabranu instancu. Na svaku od povezanih instanci možemo i kliknuti, a ta radnja će rezultirati odabirom navedene instance, pa će tako logički panel poprimiti sve vrijednosti i informacije od novo odabrane instance. Opisana radnja, uz odabir instance kroz 3D simulaciju su jedina dva načina na koje korisnik može odabrati instancu i promatrati njene parametre, odnosno njihove vrijednosti.

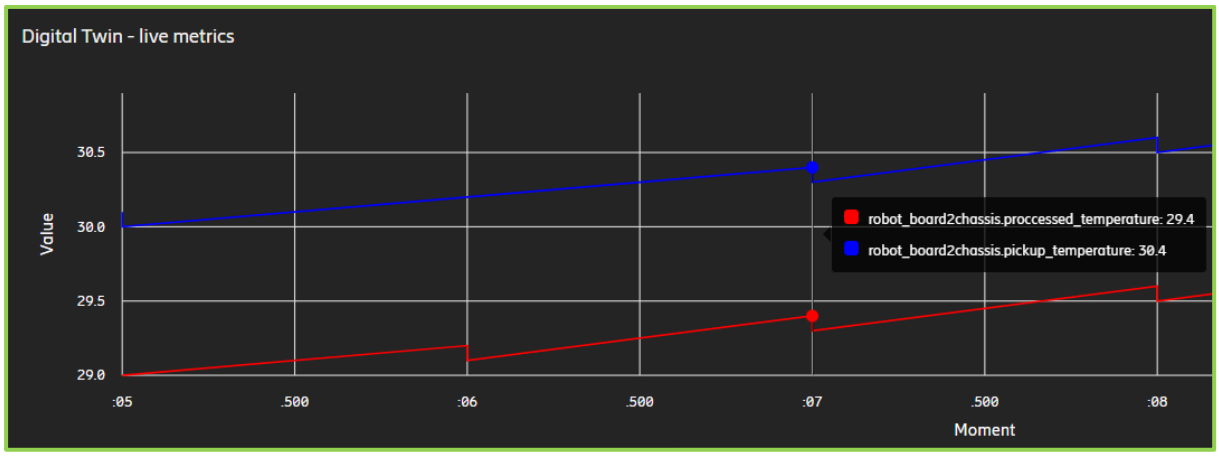


Slika 14. Isječak iz Angular aplikacije, prikaz Logičke konfiguracije s vidljivim povezanim instancama, odnosno roditeljskom instancom „*floor_board*“

Nadalje, kroz ovu komponentu korisnik ima mogućnost dodavanja novih značajki, odnosno kompozitnih metrika i intentova, a da bih korisnik mogao provesti navedene radnje mora kliknuti na malu ikonicu „+“ koja se nalazi pokraj naslova od kompozitnih metrika (**Slika *14***). Isto tako korisnik ima mogućnost ažuriranja i brisanja svih parametara relevantnih za odabranu instancu, a da bih to proveo dovoljno je dva puta kliknuti na određenu instancu što će rezultirati otvaranjem HTML forme za unos podataka. Kao što sam već spomenuo, dodavanje senzora i aktuatora se može provesti samo kroz Fizičku komponentu, s obzirom na to da su nam potrebne koordinate na kojima će se isti nalaziti. Nakon što se pokrene proizvodnja u tvornici, senzorni podatci i ostali parametri će biti direktno ažurirani u ovoj komponenti, tako da će sve promjene biti vidljive korisniku.

10.3. Komponenta za grafički prikaz podataka

Unatoč činjenici što ovaj sustav pruža mogućnost pohrane podataka u bazu, odlučili smo implementirati kratku povijest o podatcima unutar same aplikacije. Jedan od najboljih načina za vizualizaciju podatak je graf, stoga imamo i treću komponentu koju nazivamo „Podatci uživo“ (eng. *live metrics*). Ova komponenta unutar sebe sadrži linijski graf koji prikazuje zadnjih 20 vrijednosti nekog parametra, važno je naglasiti kako se vrijednosti starije od 10 minuta automatski brišu. Kako bih vidjeli povijesne vrijednosti nekog parametra, prvo moramo odabrati instancu na kojoj se parametar nalazi i onda u logičkom panelu kliknuti na ime, odnosno ikonicu parametra. Graf pruža mogućnost prikazivanja više parametara istog tipa, ako odaberemo jedan temperaturni senzor, na grafu će biti prikazani svi temperaturni senzori koji pripadaju odabranoj instanci. Onaj senzor kojeg smo odabrali, odnosno na kojeg smo kliknuli će biti obojen crvenom linijom na grafu, a ostali će biti prikazani u različitim nijansama plave boje (Slika 15).

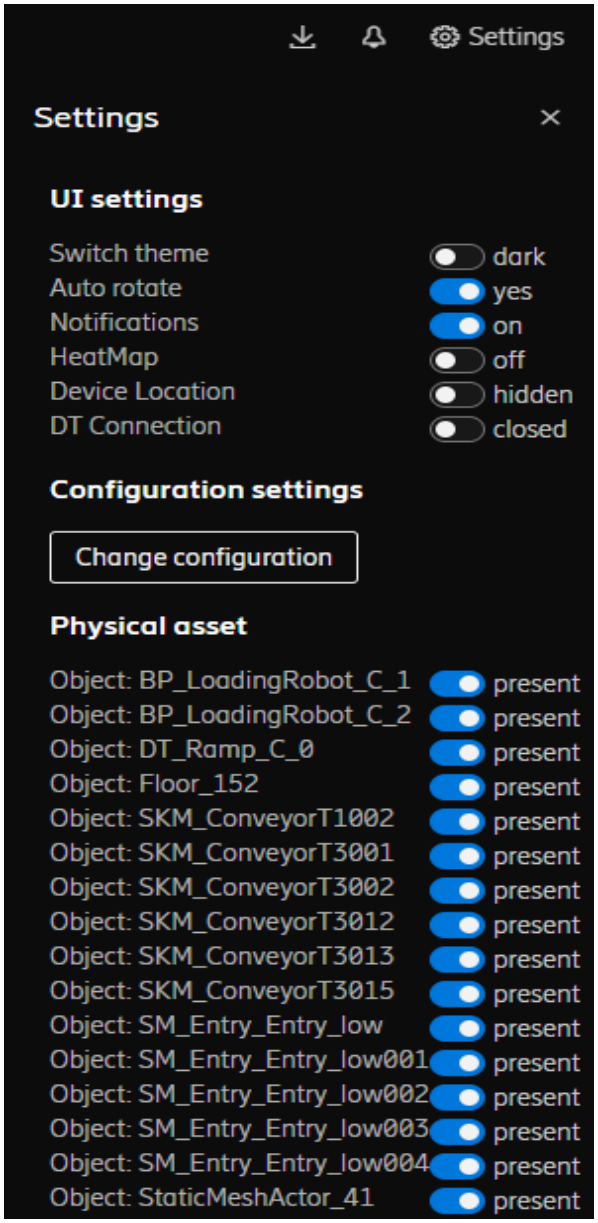


Slika 15. Isječak iz Angular aplikacije, izgled komponente za grafički prikaz podatka uživo

10.4. Konfiguracijska komponenta

Pomoćna komponenta koja je spomenuta na početku ovog paragrafa se odnosni na komponentu za promjenu postavki, odnosno konfiguracijskih parametara. Kako bih pristupili ovoj komponenti, potrebno je kliknuti na opciju za postavke (eng. *settings*) koja se nalazi u gornjem desnom kutu sučelja. Ova komponenta (Slika 16) se sastoji od tri dijela, prvi se odnosi na promjene korisničkog sučelja, pa tako imamo sklopke za promjene teme iz svijetle u tamnu,

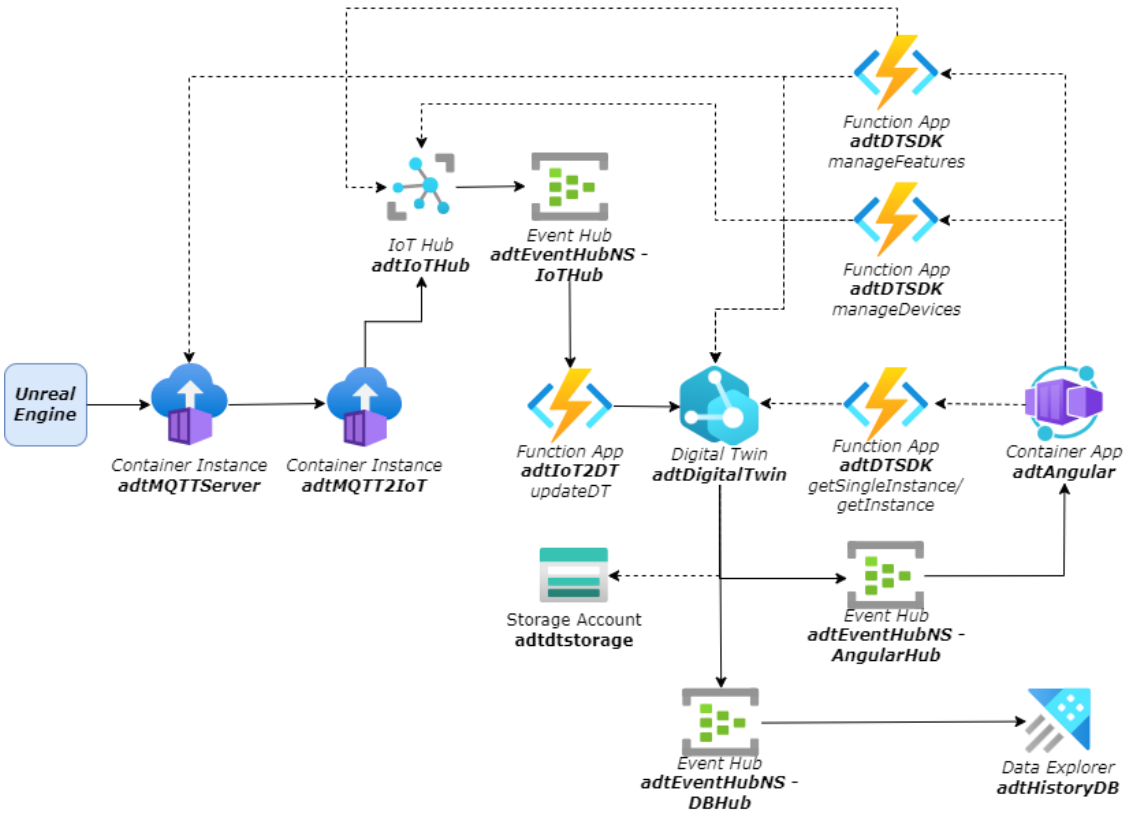
zatim opciju zaustavljanja rotacije 3D simulacije, sklopku za uključivanje i isključivanje notifikacija, sklopka za uključivanje toplinskog otiska i sklopka za uspostavljanje veze s ostatkom sustava, odnosno povezivanje s Event Hub servisom koji prosljeđuje sve događaje proizašle iz Digital Twin servisa. Drugi dio se odnosi na promjenu konfiguracije, tako da korisnik može zamjeni postojeću strukturu digitalnih dvojnika s nekom drugom strukturom. Treći dio je namijenjen za fizičke komponente, dakle tu se nalazi lista svih 3D objekata koji se nalaze na simulaciji, pa tako svaki objekt možemo učiti vidljivim odnosno nevidljivim, što pomaže pri vizualizaciji objekata. Ova komponenta služi i za generiranje obavijesti, koje nastaju kao posljedica upravljanja nad parametrima digitalnih dvojnika, stoga će svaka radnja te prirode rezultirati jednostavnom notifikacijom.



Slika 16. Isječak iz Angular aplikacije, prikaz komponente za konfiguraciju postavki

11. Struktura servisa i komunikacija

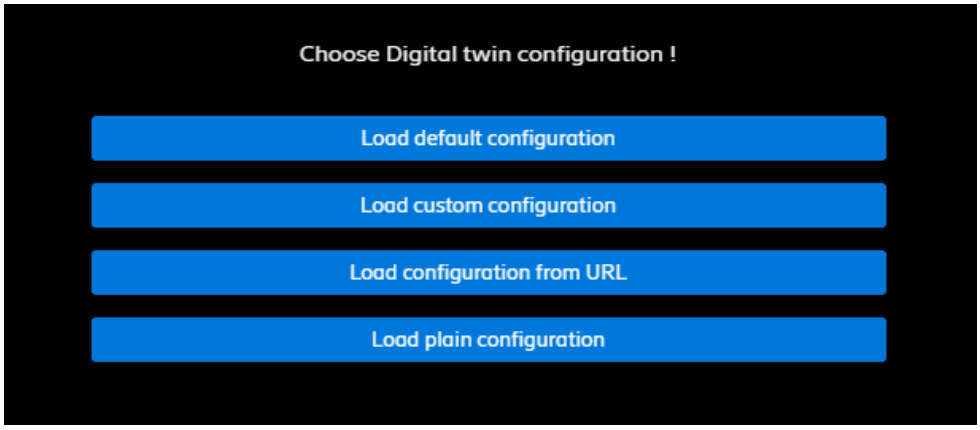
Do sada smo se upoznali s virtualnom tvornicom i načinom na koji ista funkcionira, zatim smo definirali modele potrebne za stvaranje digitalnog dvojnika na spomenutoj tvornici te smo se upoznali s terminologijom i konceptima na kojima digitalni dvojnici počivaju. Također, prikazali smo korisničko sučelje, komponente od kojih je sastavljeno te načini i metode koje korisnici mogu iskoristiti u svrhu nadzora i upravljanja nad digitalnim dvojnikom. U ovom koraku ćemo se upoznati s procesom integracije digitalnih dvojnika unutar Microsoft Azure platformu, posebno ćemo obratiti pozornost na korištene Azure servise kao i načine na koji spomenuti servisi komuniciraju. S obzirom na to da je upravljanje i nadzor nad cijelim sustavom prilagođeno korisničkom sučelju, perspektivu ćemo postaviti iz tog smjera. U teorijskom dijelu rada, imali smo priliku upoznati se s Azure platformom i nekoliko različitih servisa kao što su Function App, Container Instance, Digital Twin, IoT Hub, Event Hub i ostali. Svi navedeni servisi su bili potrebni kako bih uspješno završili proces integracije i zapravo stvorili funkcionalnog digitalnog dvojnika. Na priloženoj slici (Slika 17) možete vidjeti sve korištene servise i na koji način su međusobno povezani.



Slika 17. Struktura cjelokupnog sustava prikazana kroz Microsoft Azure servise.

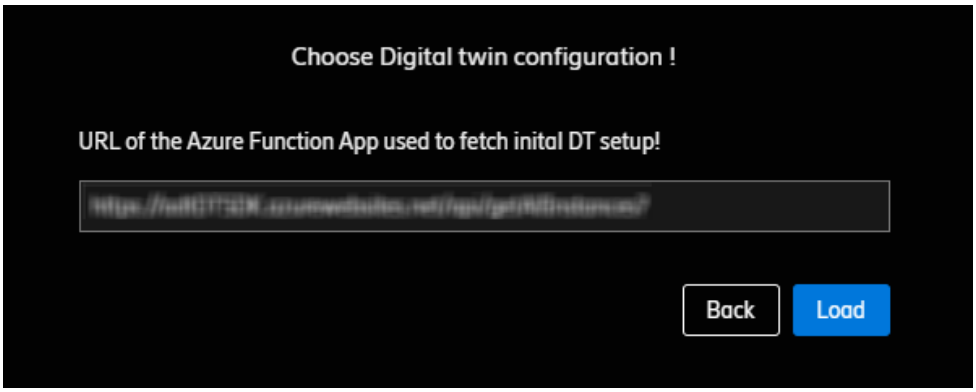
11.1. Učitavanje konfiguracije i podataka

Korisničko sučelje je jedan od najjednostavnijih načina na koji korisnici mogu ući u interakciju sa digitalnim dvojnikom. Kada korisnik pristupi Angular aplikaciji, prvo što vidi je izbornik (Slika 18) s četiri opcije za učitavanje konfiguracije. Uz pomoć navedenih opcija definiramo na koji način želimo učitati konfiguraciju potrebnu za upravljanje i nadzor nad digitalnim dvojnikom. Preporučeno je **učitavanje konfiguracije putem URL-a** (eng. *Load configuration from URL*), ali postoje mogućnosti učitavanja konfiguracije uz pomoć konfiguracijskog dokumenta (eng. *Load Custom Configuration*), kao i mogućnost učitavanja testne (eng. *Load default configuration*) ili prazne konfiguracije (eng. *Load plain configuration*).



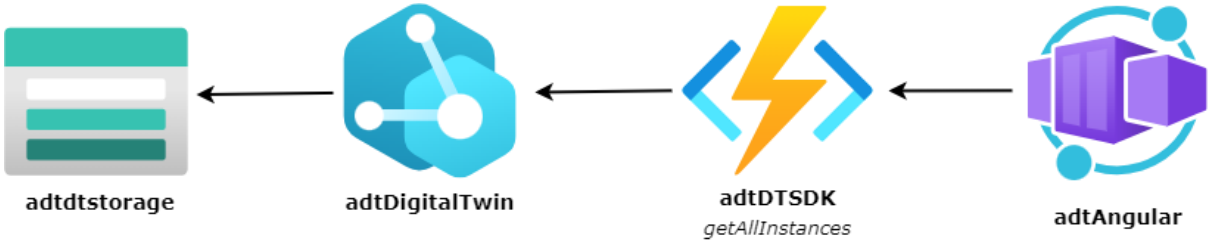
Slika 18. Isječak iz Angular aplikacije, prikaz izbornika sa opcijama za učitavanje konfiguracije

Nakon odabira učitavanja konfiguracije putem URL-a, korisnicima imaju mogućnost unosa URL-a čija je svrha dobavljanje svih podatke i konfiguracijske parametre potrebnih kako bih se učitalo korisničko sučelje. Bitno je naglasiti kako se unutar samog tekstualnog okvira (Slika 19) nalazi URL od funkcije (*getAllInstances*) koju smo prethodno dizajnirali kako bih obavljala spomenutu ulogu.



Slika 19. Isječak iz Angular aplikacije, unos URL-a korištenog u svrhu učitavanja konfiguracije

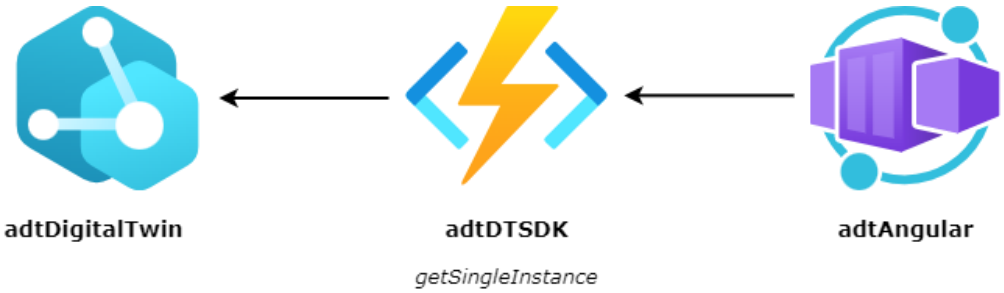
Spomenuta funkcija je podignuta unutar Function App-a (***adtDTSDK***), a njena uloga je dobavljanje podataka od svih digitalnih instanci pohranjenih unutar Digital Twin resursa (***adtDigitalTwin***). Podatci se uglavnom odnose na nazive digitalnih instanci i parametre vezane uz 3D objekte svake instance, poput boje i URL-a od svakog 3D objekta pohranjenog na Azure Storage Account resursu (***adtdtstorage***). Važno je naglasiti kako je ova funkcija konfigurirana tako da se logika sadržana u njoj izvrši putem HTTP zahtjeva (eng. *HTTP triggered*), tako da kada god neko pristupi URL-u od navedene funkcije, ona će napraviti zahtjev prema Digital Twin resursu (***adtDigitalTwin***), dobaviti potrebne podatke, formatirat ih i vratiti ih kao odgovor u JSON formatu. U praksi, ova funkcija (Slika 20) se poziva samo jednom, prilikom učitavanja konfiguracije u Angular aplikaciju, a njena glavna svrha je dobavljanje potrebnih informacija kako bih se generirali 3D objekti unutar sučelja, što olakšava vizualizaciju krajnjim korisnicima.



Slika 20. Prikaz Azure servisa korištenih za učitavanje konfiguracije uz pomoć URL-a

Nakon odabira konfiguracije, možemo vidjeti korisničko sučelje sastavljeno od 3 glavne komponente (Slika 10), a posebnu pozornost zauzima fizička komponenta, unutar koje se nalazi 3D simulacija (Slika 11 i Slika 12). Ova komponenta se odnosi na fizičku konfiguraciju, odnosno na 3D objekte koje smo učitali u prethodnom koraku, dakle tu vidimo virtualni izgled našeg digitalnog dvojnika, odnosno proizvodne linije. Jedna od krucijalnih funkcionalnosti 3D simulacije jest da korisnik može odabrati jedan od objekata jednostavnim klikom nad istim, što će rezultirati učitavanjem dodatnih informacija o objektu u drugoj komponenti koja se naziva logička komponenta.

U pozadini navedenog učitavanja se krije HTTP poziv prema funkciji (*getSingleInstance*) koja je također podignuta u sklopu Function App-a (***adtDTSDK).*** Uloga ove funkcije (Slika 21) je da na temelju poslane identifikacije (eng. *ID*) dobavi detaljnije informacije o željenoj instanci, a informacije uključuju podatke o svim parametrima vezanim za instancu, odnosno uređajima (eng. *devices*), značajkama (eng. *features*) kao i vezama između ostalih digitalnih instanci.

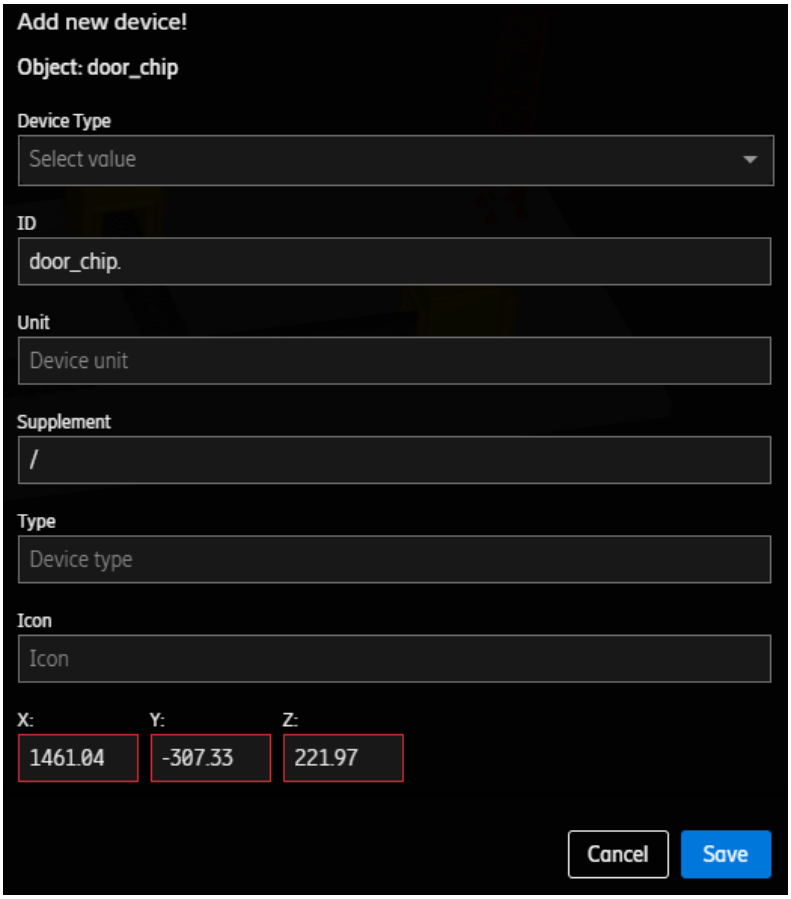


Slika 21. Prikaz Azure servisa korištenih za učitavanje pojedinačnih instanci.

Temeljem spomenutih podataka, na vrhu Logičke komponente (Slika 13) će se generirati naziv instance (ID) kao i ime 3D objekta koji je predstavlja, ispod toga se nalazi lista intentova, lista kompozitnih metrika, te liste senzora i aktuatora. Pored svake od spomenutih značajki i uređaja će se nalazi posljednja zabilježena vrijednost, ako ista postoji.

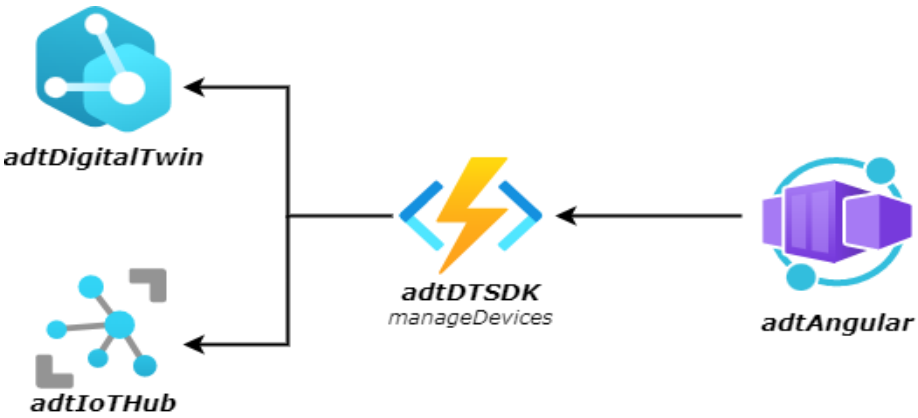
11.2. Upravljanje uređajima

Unutar korisničkog sučelja korisnicima je omogućeno upravljanje uređajima, a to smo postigli implementacijom funkcionalnosti poput dodavanja novih uređaja, ažuriranja pojedinih informacija o postojećim uređajima kao i mogućnost njihovog brisanja. Za svaku od ovih radnji, dodali smo HTML modal komponentu, riječ je o HTML formi koja se pojavljuje u obliku skočnog prozora (eng. *pop-up*), a sadrži polja za unos podataka. Kako bih korisnik dodao novi uređaj, potrebno je uz pomoć kursora doći do 3D simulacije i brzo kliknuti dva puta na onu instancu na koju želi dodati novi uređaj. Nakon navedene radnje, korisniku će biti prikazan skočni prozor (Slika 22) s imenom instance na koju želi dodati uređaj i nekoliko polja za unos koje mora popuniti kako bih dodao novi uređaj. Sva polja za unos su obvezna, pa tako korisnik ima mogućnost izbora želi li dodati senzor ili aktuator, zatim definira ID, odnosno ime tog uređaja, mjernu jedinicu, tip uređaja, dodatnu informacije o uređaju, ikonicu koja će biti prikazana u korisničkom sučelju i 3D koordinate gdje je korisnik odlučio dodati novi uređaj (popunjavaju se automatski).



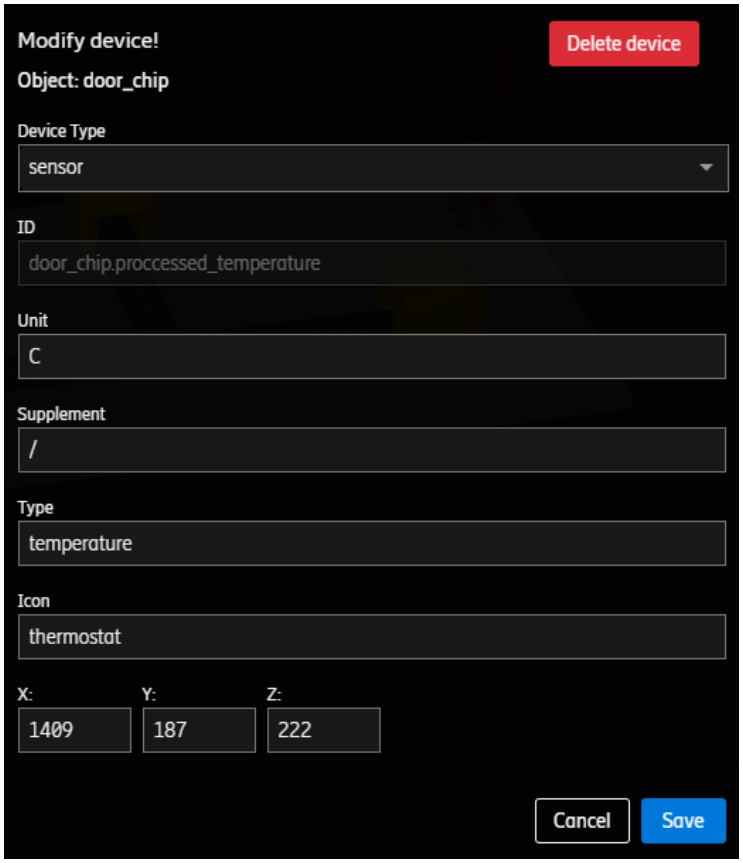
Slika 22. Isječak iz Angular aplikacije, prikaz forme za dodavanje novog uređaja

Ako su sva polja za unos ispravno popunjena, botun za pohranu (eng. *save*) promjena će postati dostupan te će se dodati novi uređaj na odabranu instancu. U pozadini ove radnje se nalazi HTTP zahtjev (Slika 23) prema Function App (***adtDTSDK***) funkciji (*manageDevices*) koja temeljem unesenih parametara dodaje novi uređaj unutar Digital Twin (***adtDigitalTwin***) i IoT Hub (***adtIoTHub***) resursa. Dakle, funkcija () prvo radi zahtjev prema Digital Twin resursu, a unutar zahtjeva se nalaze parametri potrebni koje je korisnik definirao u modal komponenti. Ako je navedena radnja uspješna, funkcija radi drugi zahtjev koji je usmjeren prema IoT Hub resursu, a svrha je kreiranja virtualnog uređaja koji ima ID jednak onome definiranom unutar Digital Twin resursa. U slučaju pozitivnog ishoda spomenutih radnji, funkcija vraća odgovor i statusni kod 200, što će omogućiti prikaz botuna za osvježavanje korisničkog sučelja, odnosno logičkog panela.



Slika 23. Prikaz Azure servisa korištenih za upravljanje uređajima

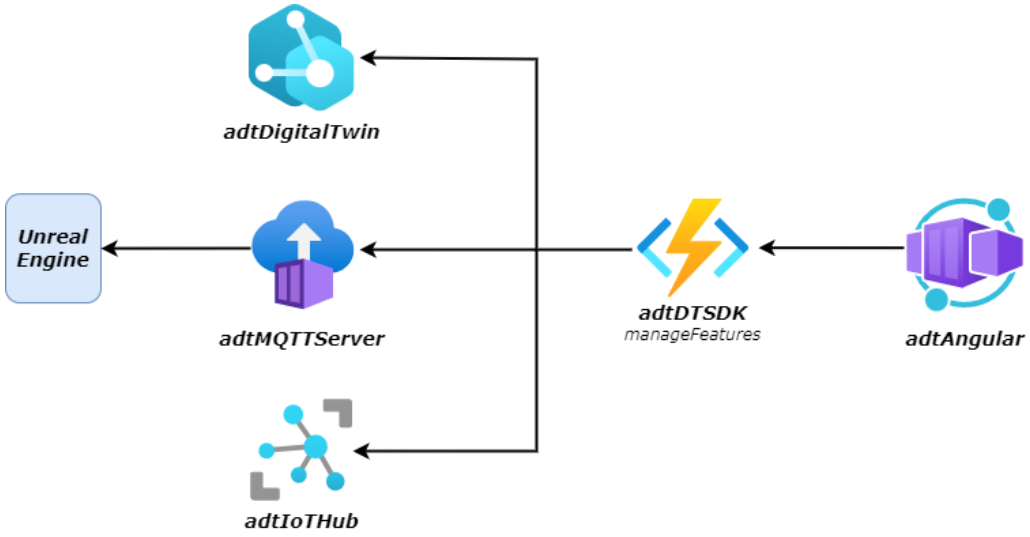
Kako bih korisnici ažurirali ili pak uklonili postojeće uređaje, potrebno je dva puta kliknuti na određeni uređaj u logičkom panelu, što će rezultirati otvaranjem skočnog prozora (Slika 24) s postojećim podatcima. Korisnik može mijenjati sve podatke osim ID-a uređaja, a promjene se spremaju samo u Digital Twin resurs. Kako bih korisnik izbrisao određeni uređaj, potrebno je u gornjem desnom kutu kliknuti na crveni botun za brisanje (eng. delete), u slučaju brisanja i ažuriranja se koristi isti Function App (***adtDTSDK)*** i funkcija (*manageDevices*) kao i u slučaju dodavanja novih uređaja.



Slika 24. Isječak iz Angular aplikacije, prikaz forme za ažuriranje/brisanje postojećeg uređaja

11.3. Upravljanje značajkama

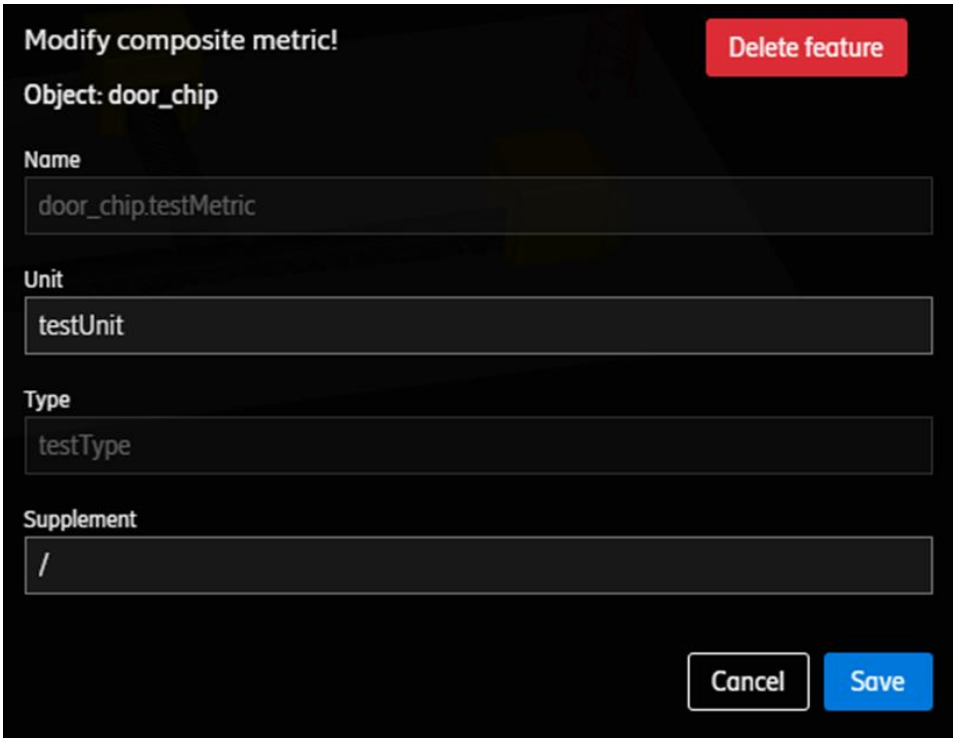
Značajke su bitna odlika digitalnih dvojnika, s obzirom na to da pomoću njih možemo upravljati digitalnim dvojnicima na višoj razini apstrakcije. Korisnicima je omogućeno upravljanje istima kao i slučaju uređaja, s tim da komponente koje sadrže formu, izgledaju drugačije nego kad uređaja. Kako bih korisnici dodali novu kompozitnu metriku, moraju kliknuti na malenu ikonicu „+“ koja se nalazi pokraj naslova „Composite Metrics“ (Slika 14). Navedena radnja će rezultati otvaranjem modal komponente s poljima za unos, navedena polja su vrlo slična kao i za uređaje osim što nema polja za unos ikonice i koordinata. Ako su svi unosi ispravni, korisnik može kliknuti na gumb za spremanje promjena i dodavanje kompozitne metrike. Unatoč činjenici što kompozitne metrike nisu uređaji, dodajemo ih u IoT Hub servis kao uređaje zbog postojeće logike u funkciji (***adtIoT2DT*** **–** *updateDT, Slika*) zaduženoj za ažuriranje Digital Twin resursa. Navedena logika, odnosno kod omogućuje ažuriranja bilo kojeg parametra vezanog uz neku digitalnu instancu, stoga bi bilo redundantno raditi dodatnu funkciju koja će obavljati istu radnju. Na ovaj način smo izbjegli redundanciju, a IoT Hub servis nema problema s dodatnim brojem uređaja i telemetrije koju je potrebno učitati.



Slika 25. Prikaz Azure servisa korištenih za upravljanje kompozitnim metrikama.

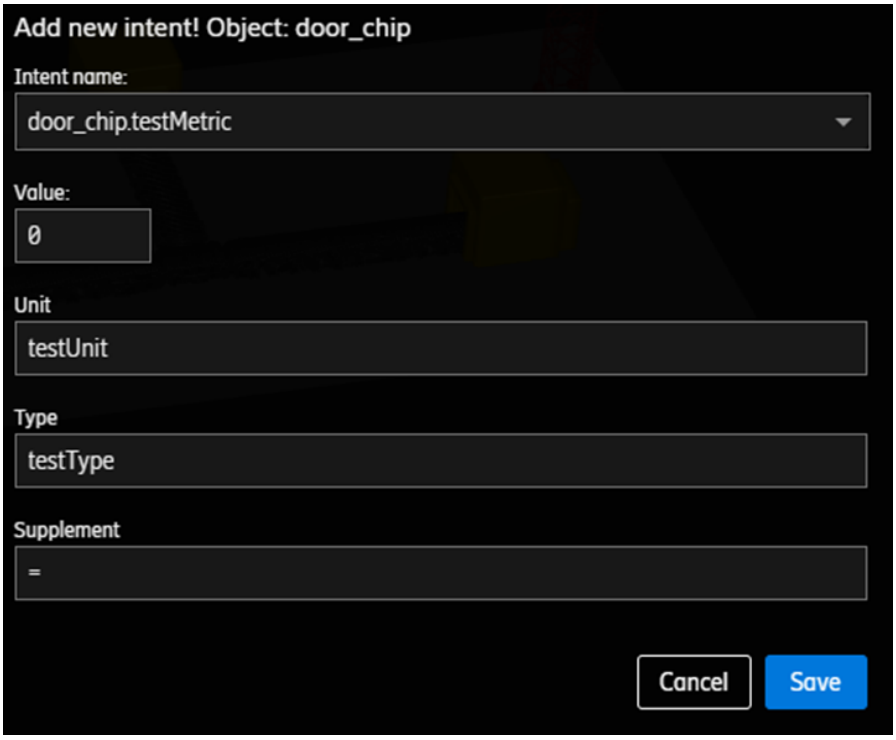
Tok podataka (Slika 25) za upravljanje kompozitnim metrika je vrlo sličan kao kod upravljanja uređajima, koristi se ista Function App (***adtDTSDK***) i funkcija (*manageDevices*) Jedina razlika je to što se uz pohranjivanje unutar Digital Twin (***adtDigitalTwin***) i IoT Hub resursa (***adtIoTHub***), dodatno šalje poruka na određenu temu unutar MQTT servera (***adtMQTTServer***), a ista služi kao okidač unutar Unreal Engine simulacije. Ažuriranje i

brisanje se događa na isti način kao i kod upravljanja uređajima, dakle da bih mogli izvesti navedene radnje, moramo dva puta kliknuti na određenu kompozitnu metriku. Klikom na odabranu metriku otvorit će se prikaz postojećih parametara (**Slika *26***) odabrane kompozitne metrike. Važno je napomenuti da ID parametar nije moguće mijenjati, a ako izvršimo promjene na ostalim parametrima metrike, one će se odraziti samo na Digital Twin (***adtDigitalTwin***) resursu, dok će brisanje rezultirati uklanjanjem kompozitne metrike iz Digital Twin i IoT Hub resursa (***adtIoTHub***).



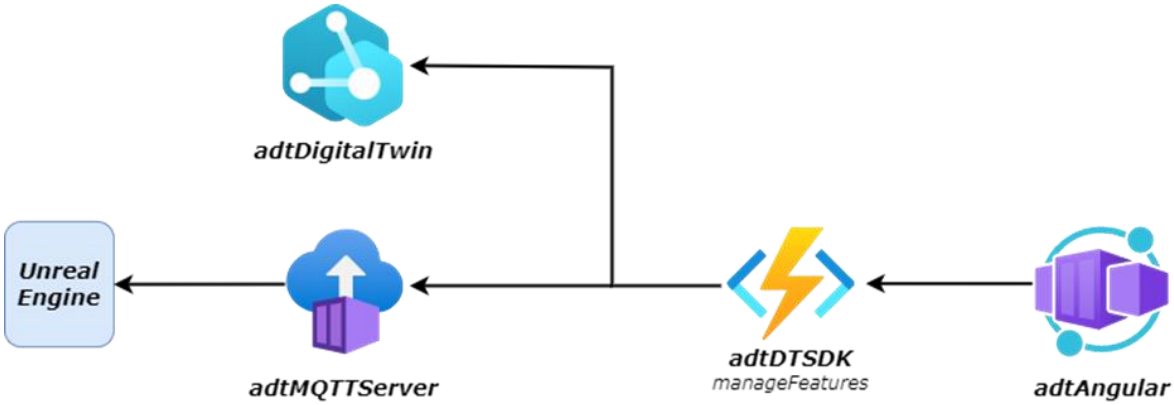
Slika 26. Isječak iz Angular aplikacije, prikaz forme za ažuriranje/brisanje kompozitne metrike

Dodavanje intentova se provodi na sličan način kao i dodavanje kompozitnih metrika, ali postoji uvjet koji definira postojanje barem jedne kompozitne metrika prije dodavanja intenta na odabranoj instanci. Dakle, korisnik mora kliknuti na „+“ ikonicu pokraj naslova „Intents“, što će otvoriti skočni prozor (Slika 27) za dodavanje intentova, polja za unos se sastoje od odabira kompozitne metrike kojom želimo upravljati, mjerne jedinice, tip intenta, željene numeričke vrijednost na koju želimo postaviti kompozitnu metriku i operatora (>, <, =).



Slika 27. Isječak iz Angular aplikacije, prikaz forme za dodavanje intenta

Ako su svi unosi ispravni, promjene će biti pohranjene samo u Digital Twin resursu (***adtDigitalTwin***), a istovremeno se šalje poruka (Slika 28) na posebnu temu unutar MQTT servera (***adtMQTTServer***) kako bi Unreal Engine mogao prilagoditi vrijednost kompozitne metrike prema vrijednosti koju je korisnik definirao prilikom unosa. Ažuriranje intenta može uključivati promjenu brojčane vrijednosti i operatora, te će svaka takva promjena biti poslana na Digital Twin resurs (***adtDigitalTwin***) kao i na MQTT server (***adtMQTTServer***), dok se brisanje odvija samo nad Digital Twin resursom (***adtDigitalTwin***).



Slika 28. Prikaz Azure servisa korištenih za upravljanje intentovima

11.4. Ažuriranje podataka u sustavu

Temelj na kojem počivaju digitalni dvojnici su senzorni podatci koji stižu u stvarnom vremenu, stoga je bilo potrebno napraviti tok podataka koji će omogućiti komunikaciju između tvornice u Unreal Engine-u i digitalnog dvojnika, odnosno korisničkog sučelja. Da bih to postigli, koristili smo postojeće servise unutar Azure platforme, ali i servise koje smo samostalno dizajnirali i programirali kako bih optimizirali performanse i nadoknadili nedostatke Azure servisa. Na slici (Slika 29) možemo vidjeti servise koji čine spomenuti tok podataka. Kao što smo spomenuli više puta, senzori unutar tvornice (***Unreal Engine***) šalju očitanja na jasno definirane teme unutar MQTT servera (***adtMQTTServer***). Struktura teme je izrazito važna, s obzirom na to da pomoću nje možemo povezati vrijednost senzora/parametra sa dijelom tvornice, odnosno digitalnom instancom na kojoj se isti nalazi. Struktura teme izgleda ovako:

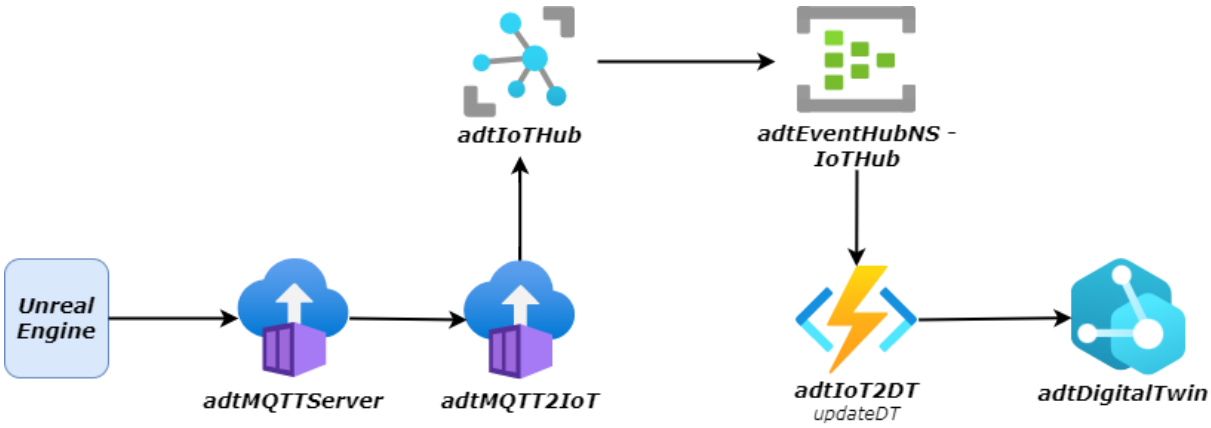
```
adt/<vrstaUređajaZnačajke>/<IDDigitalneInstance>/<IDUređajaZnačajke>
```

Prvi parametar „*adt*“ služi za identifikaciju teme, dakle nalazi se na početku svake teme vezane uz ovaj sustav, drugi parametar „*vrstaUređajaZnačajke*“ definira o kojoj vrsti uređaja (senzor ili aktuator) ili značajke (kompozitna metrika) se radi. Treći parametar „*IDDigitalneInstance*“ definira ID instance na kojoj se spomenuti uređaj ili značajka nalazi, a posljednji parametar „*IDUređajaZnačajke*“ definira ID samog uređaja ili značajke o kojem se radi. ID definiran u posljednjem parametru je jednaka nazivu svojstva (eng. *property*) digitalne instance čiju ćemo vrijednost ažurirati temeljem numeričke vrijednosti poslane na definiranu temu. Na primjerima ispod možete vidjeti kako to izgleda u praksi:

`adt/sensors/door_chip/energy_consumption`	`1.56`
`adt/actuators/door_chip/production_delay`	`0.23`
`adt/metrics/floor_chip/energy_consumption`	`4.28`

Spomenuo sam kako je potrebno kreirati nove servise kako bih se uspostavila i optimizirala komunikaciju između krajnjih točaka u ovom komunikacijskom kanalu (Slika 29). Primjer takvog servisa je Python aplikacija (***adtMQTT2IoT***) pokrenuta unutar Container Instance resursa. Ovaj servis ima dvije uloge, prva je pretplata na MQTT servis, a druga je ažuriranje IoT uređaja definiranih unutar IoT Hub servisa (***adtIoTHub***). Poseban značaj ove aplikacije je definiran u načinu na rada s više niti (eng. *multi-thread*), jedna nit je zadužena za pretplatu na

MQTT server, slušanje svih poruka poslanih na „adt“ temu, formatiranje istih prema prethodno definiranoj strukturi teme i pospremanje validnih tema i vrijednosti u red (eng. *queue*) za ažuriranje. Druga nit čita poruke pohranjene u red i po redu ih šalje na prethodno kreirane virtualne uređaje unutar IoT Hub servisa (***adtIoTHub***).



Slika 29. Prikaz Azure servisa korištenih u svrhu ažuriranja vrijednosti unutar Digital Twin resursa

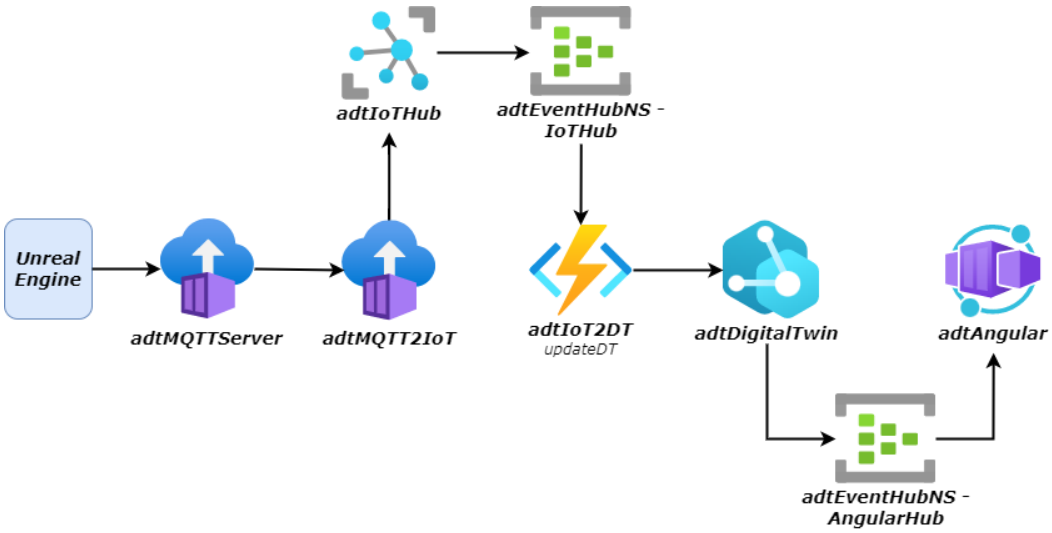
Zatim imamo IoT Hub (***adtIoTHub***), riječ je o servisu koji omogućuje konzumaciju telemetrije i vrijednosti poslane od strane uređaja i senzora. Kako bih mogao konzumirati vrijednosti s više stotina uređaja istovremeno, IoT Hub nudi mogućnost kreiranja virtualnih uređaja koje možemo zamisliti kao virtualnu repliku stvarnih uređaja. Navedeni virtualni uređaji osiguravaju sigurnosnu komunikaciju između fizičkih IoT uređaja i IoT Hub servisa, a ujedno nude i dodatne informacije o stanju uređaja poput povezanosti s ovim servisom. IoT Hub servis nudi nekoliko različitih opcija za prosljeđivanje događaja vezanih uz navedeni servis, a događaji mogu uključivati dodavanje i brisanje uređaja, spajanje i od spajanje kao i **telemetriju** istih. Posljednja vrsta događaja je izuzetna zanimljiva za naš slučaj, stoga kad god neki IoT uređaj primi novu vrijednost ona će biti proslijeđena na sljedeću točku unutar ovog toka podataka a to je Event Hub (***adtEventHubNS - IoTHub***).

Event Hub je servis koji omogućuje konzumaciju velike količine događaja, a ujedno nudi i opcije slanja događaja i pretplate na iste. U našem scenariju, ovaj Azure servis smo iskoristili kako bih međusobno povezali servise u slučaju kad je to moguće. Dakle kad bilo koji senzor unutar Unreal Engine tvornice pošalje vrijednost, ona će biti uhvaćena od strane Python multi-thread aplikaciji, proslijeđena na odgovarajući virtualni uređaj unutar IoT Hub servisa, a sam servis je konfiguriran tako da sve događaje pa tako i telemetriju prosljeđuje na Event Hub (***adtEventHubNS - IoTHub***).

Nakon Event Hub servisa, dolazi Function App (***adtIoT2DT***) i unutar njega funkciju (*updateDT*) koja je konfigurirana tako da se izvrši kad god Event Hub primi nekakav događaj (eng. *Event Hub trigger*). Pojednostavljeno, navedena funkcija je pretplaćena na Event Hub i kad god neka vrijednost bude poslana, ova funkcija će izvršiti svoju zadaću. U našem scenariju, funkcija je konfigurirana da na temelju događaja proizašlog iz IoT Hub servisa, ažurira vrijednosti u Digital Twin resursu. Dakle događaji proizašli iz IoT Hub imaju jasno definiranu strukturu koja uključuje vrstu događaja, ime uređaja koji je proizveo događaj i vrijednost događaja ako je riječ o telemetriji. Temeljem ovih parametara, funkcija (*updateDT*) se spaja na Digital Twin resurs (***adtDigitalTwin***) i pokušava ažurirati vrijednost određenog parametra unutar Digital Twin instance ako je to moguće.

Centralni resurs u ovom toku podataka je digitalni dvojni (***adtDigitalTwin***) koji zapravo daje kontekst uređajima, odnosno cijeloj priči. IoT uređaji su izuzetno bitni kako bih nam pružili neke specifične informacije o nekom sustavu, ali navedena informacija postaje znatno korisnija kada znamo gdje se spomenuti uređaj točno nalazi, što mjeri, čim može upravljati, što ovisi o njegovim očitanjima, a sve ove informacije definiramo uz pomoć konteksta, odnosno digitalnog dvojnika. Sve navedene informacije o digitalnim instancama i njihovim parametrima smo definirali uz pomoć modela, a temeljem njih smo napravili digitalne instance koje smo i međusobno povezali. Ovaj procesom smo stvorili kontekst i omogućili ažuriranje svih vrijednost proizašlih iz IoT uređaja unutar tvornice.

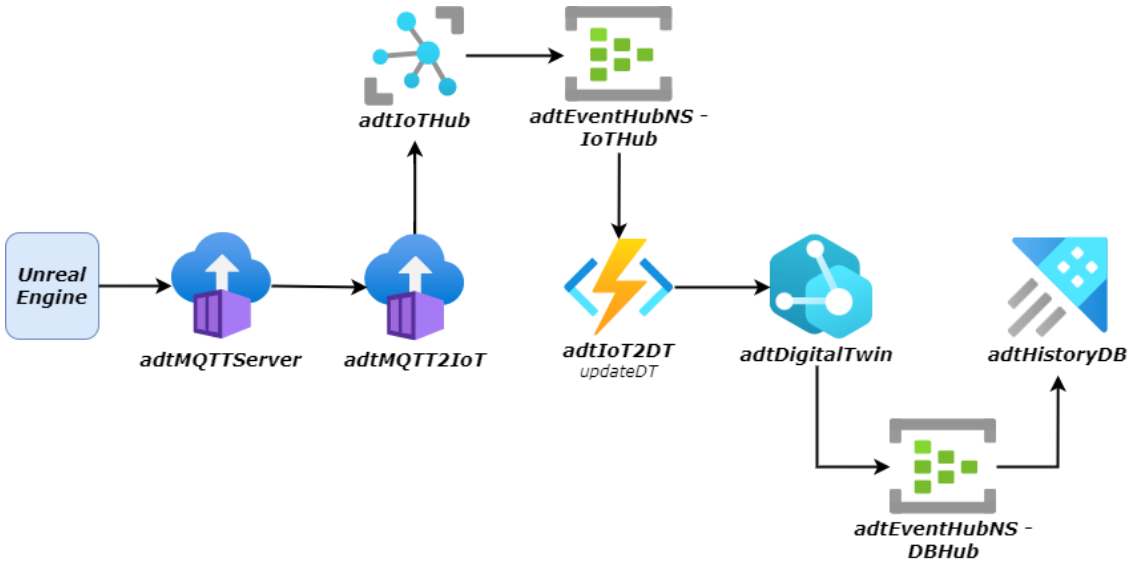
Digital Twin resurs, baš kao i IoT Hub pruža mogućnost prosljeđivanja događaja na Event Hub servis (***adtEventHubNS – AngularHub***), a navedeni događaji mogu uključivati kreiranje, brisanje i **ažuriranje instanci**, stvaranje veza između instanci i hvatanje telemetrije. Ažuriranje instanci, odnosno značajki i uređaja unutar Digital Twin instance je jedan od važnijih događaja unutar samog Digital Twin resursa, jer je to zapravo i najčešća radnja unutar ovog scenarija. Ažuriranje se može odnositi na učitavanja nove vrijednost za neki uređaj i značajku, ali i na promjene nekih drugih parametara poput mjerne jedinice, ikonice za sučelje ili pak promjene tipa senzora. Svi navedeni događaji se prosljeđuju na Event Hub servis (***adtEventHubNS - AngularHub***) što nam pruža mogućnost ažuriranja sučelja (Slika 30), odnosno pružanja najsvježijih informacija korisniku. Unutar Angular aplikacije (***adtAngular***) postoji modul zadužen za spajanje na Event Hub servis, a istim možemo upravljati pomoću sklopke „DT Connection“ unutar komponente za postavke (Slika 16). Nakon što se uspostavi konekcija s Event Hub servisom i podatci krenu pristizati, isti će biti formatirani i poslužit će za ažuriranje podataka o instancama koje smo dobavili u prethodnim koracima.



Slika 30. Prikaz Azure servisa korištenih u svrhu ažuriranja vrijednosti unutar Angular aplikacije

11.5. Pohrana podataka u bazu

Kako bih digitalni dvojni mogao učiti iz podataka i donositi odluke, ključno je navedene podatke sačuvati, a kako bih to postigli koristimo Azure servis pod nazivom Data Explorer Kusto. Riječ je o bazi podataka koju jednostavno možemo povezati s Digital Twin servisom putem Azure Portala. Proces prikupljanja podataka (Slika 31) iz Digital Twin servisa se odvija preko Event Hub-a, koji bilježi sve događaje i podatke te ih pohranjuje u bazu podataka. Ovaj tok podataka sličan je prethodnom poglavlju, ali koristimo različita zadnja dva resursa.

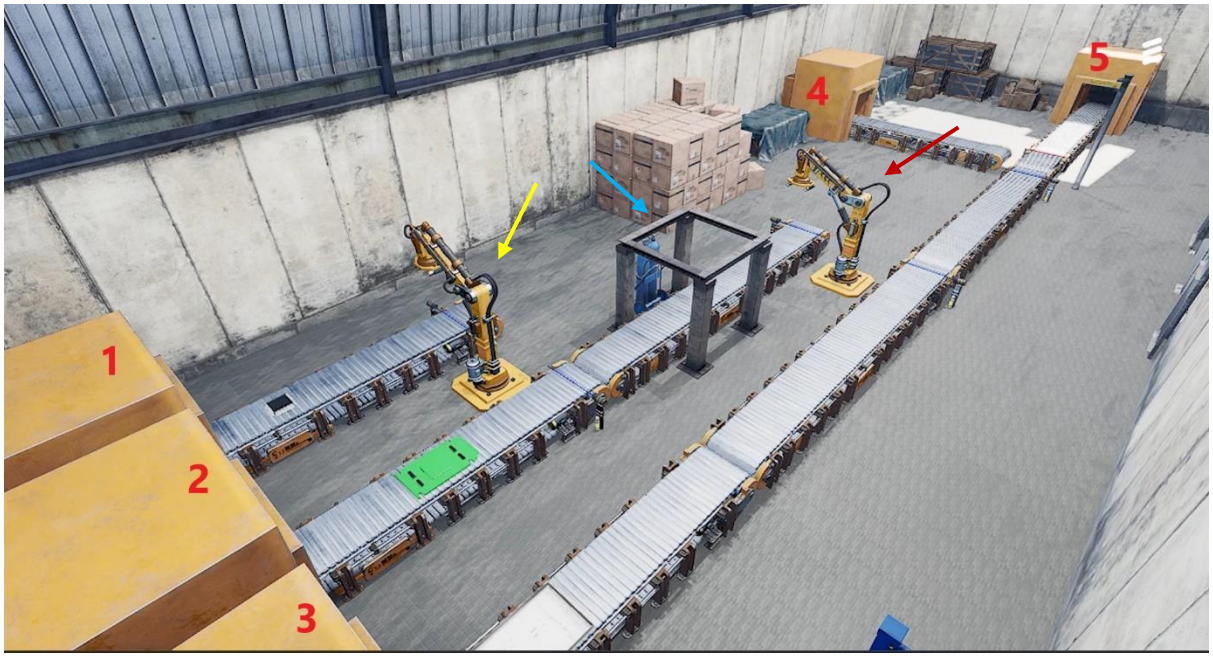


Slika 31. Prikaz Azure servisa korištenih za pohranu događaja iz Digital Twin resursa u bazu podataka

Na početku imamo uređaje u tvornice (***Unreal Engine***), koji šalju očitane vrijednosti na MQTT server (***adtMQTTServer***). Zatim Python multi-thread aplikacija (***adtMQTT2IoT***) hvata poruke s MQTT servera i prosljeđuje ih na virtualne IoT uređaje definirane unutar IoT Hub servisa (***adtIoTHub***). Nadalje, IoT Hub prosljeđuje sve događaje koji proizlaze iz virtualnih uređaja na Event Hub servis (***adtEventHubNS - IoTHub***) na kojeg je pretplaćen Azure Function App (***adtIoT2DT***), odnosno funkcija (*updateDT*) koja ima ulogu ažuriranja instanci u Digital Twin servisu (***adtDigitalTwin***) na temelju primljenih događaja. Glavna promjena se nalazi nakon Digital Twin servisa, koji sada sve događaje šalje na Event Hub (***adtEventHubNS - DBHub***) pomoću kojeg stvaramo poveznicu na bazu podataka (***adtHistoryDB***). Kreiranjem ove poveznice, stvorit ćemo tablicu unutar baze podataka koja će spremati sve događaje vezano uz Digital Twin resurs poput vremenskog trenutka događaja, imena servisa, imena instance, vrijednosti značajke ili uređaja, promjena vezanih uz poveznice između instanci i ostalih parametara.

12. Predefinirani scenariji u digitalnom dvojniku

Kako bih pokazali primjenu digitalnog dvojnika na primjeru tvornice za proizvodnju matičnih ploča, implementirali smo nekoliko različitih scenarija koji pokazuju interakciju između korisnika i digitalnog dvojnika, posredstvom korisničkog sučelja. Kao što sam objasnio u poglavlju Kompozitne metrike i intentovi, prilikom implementacije digitalnih dvojnika nismo koristili implementirali strojno učenje ni metode umjetne inteligencije za donošenje odluka i optimizaciju sustava, nego smo ručno definirali obrasce ponašanja prema unaprijed dogovorenim scenarijima. U sljedećim primjerima ću ukratko objasniti nekoliko primjera kako smo definirali spomenute obrasce ponašanja, odnosno scenarije.



Slika 32. Prikaz Unreal Engine tvornice sa označenim instancama/objektima

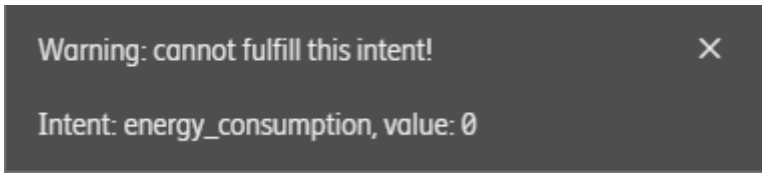
Stanje u tvornici je konfigurirano tako da svaki proizveden čip ne odgovara standardima kvalitete, što je indirektna posljedica previsoke temperature nastale prilikom instalacije na matičnu ploču. Kroz ove scenarije želimo popraviti stanje u tvornici, odnosno kvalitetu finalnih proizvoda. Dakle učitati ćemo korisničko sučelje koristeći učitavanje konfiguracije uz pomoć URL-a, zatim ćemo u postavkama stvoriti vezu s Event Hub servisom koji je zadužen za ažuriranje korisničkog sučelja. Nakon toga možemo vidjeti kako se stanje u Fizičkoj konfiguraciji mijenja, robotske ruke se okreću, kao i pokretna traka za sortiranje, a ako uključimo temperaturni otisak unutar tvornice, vidjeti ćemo da se područje oko senzora zaduženih za mjerenje temperature instalacije crveni. Nadalje, korisnik može odabrati bilo koji

objekt ili instancu unutar tvornice, tako da klikne na nju unutar 3D simulacije ili da je odabere uz pomoć poveznih instanci. Naš konačni cilj u ovom scenariju jest povećanje kvalitete finalnih ploča, odnosno da matične ploče ne budu oštećene nakon procesa proizvodnje, a to možemo postići na nekoliko načina:

- **Povećanje vremena instalacije –** s obzirom na to da je porast temperature najveći prilikom instalacije čipova na matičnu ploču, moramo utjecati na prvu robotsku ruku nazvanu „*robot_chip2board*“, označenu žutom strelicom na slici (Slika 32). Nakon što odaberemo taj objekt u 3D simulaciji, vidjeti ćemo kako ima definirane senzore i aktuatore, posebno su nam zanimljivi temperaturni senzori „*pickup_temperature*“ i „*dropdown_temperature*“ te aktuator za podešavanje vremena instalacije nazvan „*installation_time*“. Kako bih smanjili porast temperature, želimo utjecati na vrijeme instalacije, a to ćemo napraviti dodavanjem kompozitne metrike, koju ćemo nazvati „*dropdown_temperature*“. Nakon što smo uspješno kreirali metriku, poruka će biti poslana na MQTT server, na koji je ujedno pretplaćena simulacija tvornice unutar Unreal Engine-u, a posljedica ove radnje će biti izračunavanje vrijednosti novo dodane kompozitne metrike prema unaprijed definiranoj formuli. S obzirom da smo unaprijed definirali formulu za izračun ove kompozitne metrike, ujedno smo definirali da vrijeme instalacije utječe na temperaturu, što je duže vrijeme instalacije, temperatura će biti niža. Sljedeći korak je dodavanje intenta, gdje definiramo da vrijednost kompozitne metrike „*dropdown_temperature*“ ne smije biti veća od 35 °C. Ova radnja će također poslati poruku na MQTT server, Unreal Engine simulacija će tu poruku primit i prema poslanoj vrijednosti usklađivati vrijeme instalacije.
- **Hlađenje instaliranih čipova –** nakon što smo smanjili temperaturu čipova instaliranih na matičnu ploču, vidimo kako je finalna temperatura matičnih ploča i dalje visoka. Druga robotska ruka nazvana „*robot_board2chassis*“ označena crvenom strelicom na slici (Slika 32), prilikom finalne instalacije podigne temperaturu matične ploče previsoko, što ošteti samu ploču. Kao u prethodnom koraku, možemo produžiti vrijeme instalacije i na taj način smanjiti temperaturu proizvoda, ali ovim postupkom ćemo znatno produžiti cijeli proces proizvodnje. Kako bih to izbjegli, odabiremo hladnjak nazvan „*cooler_board*“, označen plavom strelicom na slici, te na njega dodajemo dva senzora. Prvi senzor „*enter_temperature*“ dodajemo na ulaz u hladnjak, drugi senzor „*leave_temperature*“ dodajemo na izlaz iz hladnjaka, svaka matična ploča mora proći kroz navedeni hladnjak prije finalne instalacije. Nakon što smo dodali

senzore na hladnjak, definiramo kompozitnu metriku nazvanu „*leave_temperature*“, ponovno se šalje poruka na MQTT, a Unreal Engine simulacije prema unaprijed definiranoj formuli računa vrijednost iste. Nakon toga dodajemo intent koji upravlja navedenom metrikom, i definiramo da vrijednost te metrike ne smije biti veća od 28 °C. U pozadini ove radnje se šalje MQTT poruka, Unreal Engine je prima te šalje naredbu aktuatoru da podesi temperaturu hladnjaka na definiranu vrijednost. Nakon ove radnje vidimo da je temperatura senzora uredna i zadovoljava standarde kvalitete

- **Podešavanje potrošnje energije –** nakon što smo popravili kvalitetu matičnih ploča, vidimo da nam se povećala potrošnja energije unutar tvornice. Svaki od fizičkih objekata unutar tvornice ima senzor za mjerenje potrošnje energije, a „roditeljske“ instance (*floor_chip, floor_board, floor_recycle, floor_chassis, floor_output i factory*) imaju kompozitne metrike za mjerenje istih. Vrijednost ovih kompozitnih metrika se računa prema vrijednostima očitanih od strane senzora koji se nalaze na niže rangiranim instancama u odnosu na njih. Umjesto da ručno konfiguriramo vrijednost kompozitne metrike za potrošnju energije na svakoj od „roditeljskih“ instanci, odabiremo glavnu instancu u tvornici, nazvanu „*factory*“, koju možemo odabrati tako da kliknemo na sivu podlogu u simulaciju, odnosno pod. Unutar Logičke konfiguracije, vidjeti ćemo vrijednost kompozitne metrike „*energy_consumption*“, kako bih utjecali na nju, dodajemo intent te definiramo da vrijednost ne smije prijeći 500 kWh dnevno. Ova radnja će kao i sve prethodno, rezultirati slanjem poruke na MQTT server, a Unreal Engine će prilagoditi vrijednost svih ostalih kompozitnih metrika kako bih postigao vrijednost definiranu intentom. Korisnik može pomoću intenta staviti da vrijednost bude jednaka 0, ali će dobiti notifikaciju (Slika 33) kako potrošnja energije ne može biti jednaka 0, s obzirom na to da se odvija proizvodnja.



Slika 33. Isječak iz Angular aplikacije, prikaz notifikacije s upozorenjem.

Postoji još nekoliko različitih scenarija za upravljanje brzinom proizvodnje, ali na prethodno definiranim primjerima možemo vidjeti naše viđenje interakcije između korisnika, digitalnog dvojnika i same tvornice.

13. Rezultati i rasprava

U ovom diplomskom radu smo detaljno opisali proces implementacije digitalnog dvojnika koji se temelji na proizvodnom procesu matičnih ploča. Svaki korak implementacije smo zasebno objasnili, te smo uveli koncepte kao što su kompozitne metrika i intentovi koji omogućuju korisniku upravljanje stvarnim sustavom na višoj razini apstrakcije. Također smo dizajnirali korisničko sučelje koje omogućuje korisniku upravljanje i nadzor digitalnog blizanca, te smo definirali scenarije koji prikazuju na koji način digitalni dvojnik može optimizirati postojeći sustav, donositi odluke i provoditi analize. Međutim, ostaje pitanje koliko je sve to primjenjivo u stvarnom svijetu i na kojim sustavima je moguće provesti implementaciju digitalnih dvojnika korištenjem infrastrukture u oblaku, odnosno Microsoft Azure platforme.

Čitanjem ovog rada, mnogi će pomisliti da su glavni nedostatci to što nismo koristili stvarnu tvornicu, kao i činjenica da digitalni dvojnik ne koristi modele temeljene na strojnom učenju, nego predefinirane scenarije. Spomenute činjenice svakako imaju određenu težinu i iziskuju dodatne napore, međutim način na koji tvornica unutar Unreal Engine-a komunicira s ostatkom sustava je vrlo sličan načinu na koji bi komunikacija u stvarnom svijetu izgledala. Isto tako implementacija modela temeljenih na strojnom učenju sigurno iziskuje određene napore i implementacijske izazove, ali s obzirom na to da se modeli temeljeni na umjetnoj inteligenciji i strojnom učenju koriste u drugim kompleksnim sustavima poput autonomne vožnje, smatram da je iste moguće implementirati.

Zapravo, postavljaju se dva glavna pitanja:

- Predstavlja li problem fizička udaljenost između stvarnog sustava/tvornice i digitalnog dvojnika pokrenutog na udaljenom Azure serveru ?
- Jesu li postojeća rješenja unutar Microsoft Azure platforme dovoljno kvalitetna i efikasna za implementaciju digitalnog dvojnika u stvarnom životu?

Kada se radi o udaljenosti između krajnjih točaka, smatram da to ovisi o specifičnom sustavu na kojem je potrebno implementirati digitalnog dvojnika. U situacijama gdje je vrijeme reakcije iznimno važno, udaljenost može predstavljati izazov zbog vremenskog kašnjenja koje nastaje prilikom slanja zahtjeva/poruka putem interneta. Međutim, ako je brzina reakcije manje kritična za određeni sustav, smatram da udaljenost ne igra značajnu ulogu.

U konkretnom slučaju kojeg smo obradili, simulacija tvornice je bila pokrenuta na lokalnom računalu, dok su korisničko sučelje Angular i resurs Digital Twin bili smješteni na Azure serverima u Norveškoj. Svaka akcija unutar sučelja koja je imala utjecaj na simulaciju tvornice bila je vidljiva gotovo odmah, bez značajnog kašnjenja. Dakle, dodavanje novog senzora ili emitiranje nove vrijednosti bi se odražavalo gotovo trenutačno.

Što se tiče postojećih rješenja i usluga unutar Microsoft Azure platforme, neki od njih su izuzetno kvalitetni i omogućuju obradu velikog broja događaja i podataka. Međutim, javljaju se problemi kada imamo usko povezane servise u kojima svaki servis obrađuje podatke i prosljeđuje ih sljedećem servisu. To često rezultira uskim grlima (eng. bottlenecks) i zastojima u protoku podataka jer Microsoft Azure ne osigurava da će svaki servis koji se pokreće unutar iste Azure regije biti smješten na istom fizičkom serveru kao i ostali servisi. Drugi problem je vezan uz Azure Digital Twins resurs. Iako je ovaj servis vrlo zanimljiv te pruža mnoge značajke i mogućnosti vezane uz digitalne dvojnike, ima nekoliko nedostataka. Glavni nedostatak je ograničenje maksimalnog broja ažuriranja u sekundi. Ovaj servis dopušta **najviše 10 ažuriranja u sekundi**, što je izuzetno malo, posebno kada se radi o velikom sustavu s mnogo IoT uređaja koji šalju desetke vrijednosti u sekundi. Iako je moguće ažurirati 1000 parametara unutar jednog ažuriranja, samo je moguće provesti 10 takvih ažuriranja u sekundi. Isto tako, u ovakvom sustavu ne možemo čekati da se akumulira više podataka prije nego ih ažuriramo, već je svaki podatak potrebno ažurirati čim pristigne. Drugi nedostatak ovog servisa je vezan uz nedovoljno snažan jezik za upite (eng. *query language*). Iako je moguće kreirati komponente pomoću DTDL-a, nije moguće dobiti sve digitalne instance koje koriste određenu komponentu. Slijedom navedenog, možemo zaključiti kako je Azure Digital je zaista zanimljiv servis koji nudi brojne značajke, ali još nije spreman za primjenu u stvarnom svijetu, uključuje određena ograničenja koja nije moguće zanemariti.

Implementacija digitalnih dvojnika korištenjem postojećih rješenja unutar Azure platforme ovisi o krajnjem sustavu na temelju kojeg želimo napraviti digitalnog dvojnika. Unatoč tome što neki servis posjeduju značajna ograničenja, iste možemo zamijeniti drugi softverskim rješenjima koji su efikasniji i pružaju dodatne mogućnosti i značajke. Isto tako, potrebno je uzet u obzir i novčane izdatke nastale kao posljedica korištenja Azure servisa te ih usporediti s potencijalnim troškovima lokalne infrastrukture, međutim to je svakako pitanje za drugi rad, odnosno istraživanje.

14. Zaključak

Umjetna inteligencija je postala sveprisutna u našem životu i nudi brojne mogućnosti za ubrzanje procesa koji su dosad zahtijevali znatno više vremena. Digitalni dvojnik je jedan od tehnoloških koncepata koji korištenjem iste može značajno unaprijediti industrijske procese, povećati zaradu, optimizirati proizvodnju te omogućiti provođenje simulacija i predviđanje problema prije nego što oni nastanu.

Na početku rada, postavljen je cilj implementacije digitalnog dvojnika temeljenog na tvornici za proizvodnju matičnih ploča, korištenjem dostupnih rješenja unutar Microsoft Azure platforme. Glavna ideja bila je iskoristiti postojeća rješenja i usluge unutar Microsoft Azure platforme, te ih nadograditi s dodatnim softverskim rješenjima, kao i konceptima poput kompozitnih metrika i intentova. Kroz ovaj proces smo uspješno prikupili potrebne informacije o stvarnom sustavu, definirali modele i izradili digitalnog dvojnika, istog povezali s podatcima iz stvarne tvornice, izradili korisničko sučelje te proveli testiranja i simulacije kako bih pokazali mogućnosti i značajke izrađenog digitalnog dvojnika. Produkt ovog procesa je sustav koji predstavlja dokaz inovativnog tehnološkog koncepta, poznatog kao Proof of Concept (PoC). U okviru PoC-a identificirali smo karakteristike i potencijalni komercijalni rezultat primjene digitalnog dvojnika uz korištenje postojećih rješenja unutar Microsoft Azure platforme. Sama platforma pruža razne usluge koje omogućuju obradu i analizu velike količine podataka u stvarnom vremenu, zajedno s dodatnim softverskim rješenjima koja olakšavaju implementaciju određenih značajki. Ipak, ključni resurs ovog sustava, Azure Digital Twins, ima ograničenja vezana uz kapacitet obrade podataka, što predstavlja izazov pri implementaciji na stvarnom sustavu s velikim brojem IoT uređaja.

Zaključno, Microsoft Azure platforma zaista pruža široku paletu servisa koji su korisni i primjenjivi u kontekstu digitalnih dvojnika. Međutim, važno je imati na umu da neki od resursa imaju značajna ograničenja koja utječu na samu implementaciju. To ne znači da znači da sama platforma nije prikladna za implementaciju digitalnih dvojnika, već da je potrebno zamijeniti određene resurse drugim softverskim rješenjima koji pružaju znatno bolje performanse. Također, prilikom procesa implementacije, ključno je procijeniti krajnji sustav na temelju kojeg razvijamo digitalnog dvojnika i odabrati optimalnu infrastrukturu za tu svrhu.

15. Literatura

1. IBM**.** "What is a Digital Twin?". *IBM Topics.* (službena internetska stranica) 2021. (Pristupljeno: 1. Svibanj 2023.). Dostupno na https://www.ibm.com/topics/what-is-a-digital-twin .

2.Jeong, Deuk-Young and Baek, Myung-Sun and Lim, Tae-Beom and Kim, Yong-Woon and Kim, Se-Han and Lee, Yong-Tae and Jung, Woo-Sug and Lee, In-Bok. Digital Twin: Technology Evolution Stages and Implementation Layers With Technology Elements 2022, IEEE Access, Vol. 10, pp. 52609-52610.

3. GE.com, General Electric Company**.** GE Digital. "Digital Twin.". (službena internetska stranica) (Pristupljeno: 1. Svibanj 2023.). Dosutpno na https://www.ge.com/digital/applications/digital-twin.

4. . Erol, Tolga and Mendi, Arif and Dogan, Dilara**.** The Digital Twin Revolution in Healthcare. Istanbul : s.n., 2020. Conference: 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT).

5. VentureBeat**.** "How Singapore created the first country-scale digital twin.". *VentureBeat.* (službena internetska stranica) 2021. (Pristupljeno: 04. Svibanj 2023.). Dostupno na https://venturebeat.com/business/how-singapore-created-the-first-country-scale-digital-twin/.

6. Siemens Ecosystem, Siemens AG. Siemens, "Simulation and Digital Twin.". (službena internetska stranica) (Pristupljeno:: 2. Svibanj 2023.). Dostupno na https://ecosystem.siemens.com/cct/simulation-and-digital-twin/overview.

7. Celik, Mohsen Attaran and Bilge Gokhan. Digital Twin: Benefits, use cases, challenges, and opportunities. 2023, Decision Analytics Journal.

8. Microsoft Azure. IoT Solutions - Internet of Things Customer Stories. (službena internetska stranica) (Pristupljeno: 3. Svibanj 2023.). Dostupno na https://azure.microsoft.com/en-us/solutions/iot/#iot-approach.

9. Ziqi Huang, Yang Shen, Jiayi Li, Marcel Fey, Christian Brecher. A Survey on AI-Driven Digital Twins in Industry 4.0: Smart Manufacturing and Advanced Robotics. s.l. : Sensors, 2021, Sensors, pp. 5-6.

10. E. Alpaydin**.** Introduction to Machine Learning (2nd Edition). Cambridge : MIT Press, 2010, pp. 1-4.

11. TechTarget. TechTarget, "The History of Cloud Computing Explained.". (službena internetska stranica) (Pristupljeno: 3. Svibanj 2023.). Dostupno na https://www.techtarget.com/whatis/feature/The-history-of-cloud-computing-explained.

12. Inc, Veritis Group**.** Veritis, "COVID-19 as a Catalyst for Cloud: Global Cloud Spending to Surpass USD 1 Trillion.". (službena internetska stranica) (Pristupljeno: 3. Svibanj 2023.). Dostupno na https://www.veritis.com/news/covid-19-as-a-catalyst-for-cloud-global-cloud-spending-to-surpass-usd-1-trillion/.

13. Chen, JD. "Azure Fundamental: IaaS, PaaS, SaaS.". Medium. (službena internetska stranica) 2021. (Pristupljeno: 5. Svibanj 2023.). Dostupno na https://medium.com/chenjd-xyz/azure-fundamental-iaas-paas-saas-973e0c406de7.

14. Microsoft**.** Azure Digital Twins - Overview. Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 7. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/digital-twins/overview.

15. "IoT Concepts and IoT Hub". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 7. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/iot-hub/iot-concepts-and-iot-hub.

16. "Azure Functions Scenarios - Python.". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 7. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/azure-functions/functions-scenarios?pivots=programming-language-python.

17. "Azure Functions triggers and bindings - Azure Functions.". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 7. Svibanj 2023.) Dostupno na https://learn.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings?tabs=csharp.

18. "Container Instances - Overview.". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 8. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/container-instances/container-instances-overview.

19. "Container Apps - Overview". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 8. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/container-apps/overview.

20. "Event Hubs - About.". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 8. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-about.

21."Azure Data Explorer - Overview.". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 9. Svibanj 20223.) Dostupno na https://learn.microsoft.com/en-us/azure/data-explorer/data-explorer-overview,.

22."Storage Accounts - Overview.". Microsoft Azure Documentation. (službena internetska stranica) 2021. (Pristupljeno: 9. Svibanj 2023.). Dostupno na https://learn.microsoft.com/en-us/azure/storage/common/storage-account-overview.

23. BairesDev**.** "What Is Unreal Engine?". BairesDev Blog. (službena internetska stranica) 2023. (Pristupljeno:11. Svibanj 2023.) Dostupno na https://www.bairesdev.com/blog/what-is-unreal-engine/.

24. Games, Epic. "MQTT Client.". Unreal Engine Marketplace. (službena internetska stranica) 2021. (Pristupljeno: 11. Svibanj 2023.). Dostupno na www.unrealengine.com/marketplace/en-US/product/mqtt-client..

25. Medium. "The History of Angular.". Medium, The Startup Lab. (službena internetska stranica) 2021. (Pristupljeno: 15. Svibanj 2023.). Dostupno na https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7.

26. Ltd, Simplilearn Solutions Pvt. "What is Angular?". Simplilearn. (službena internetska stranica) 2023. (Pristupljeno: 15. Svibanj 2023.). Dostupno na https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular.

27. Google LLC. "Architecture Components.". Angular.io. (službena internetska stranica) 2021. (Pristupljeno:16. Svibanj 2023.). Dostupno na https://angular.io/guide/architecture-components.

28. TechTarget. "MQTT (MQ Telemetry Transport).". IoT Agenda. (službena internetska stranica) 2021. (Pristupljeno: 18. Svibanj 2023.) Dostupno na https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport.

29. Foundation, Eclipse. "Mosquitto - An Open Source MQTT Broker.". Mosquitto. (službena internetska stranica) 2021. (Pristupljeno: 18. Svibanj 2023.). Dostupno na https://mosquitto.org/.

30. Software, Eiffel. "Elementary and Composite Metrics.". Eiffel Documentation. (službena internetska stranica) 2021. (Pristupljeno: 29. Svibanj 2023.). Dostupno na https://www.eiffel.org/doc/eiffelstudio/Elementary_and_composite_metrics#:~:text=Definition%20%2D%2D%20Composite%20metric%3A%20A,or%20previously%20defined%20composite%20metrics).

16. Popis slika

17. Sažetak

Implementacija koncepta digitalnog dvojnika korištenjem Microsoft Azure platforme

U ovom diplomskom radu smo obradili tematiku vezanu uz digitalne dvojnike te njihovu implementaciju korištenjem Microsoft Azure platforme. Rad je organiziran u dva dijela, prvi dio je fokusiran na teoriju, odnosno tehnologije i koncepte koje smo primijenili u drugom, odnosno praktičnom dijelu. U praktičnom dijelu istraživanja, proveli smo proces implementacije digitalnog dvojnika na primjeru tvornice za proizvodnju matičnih ploča. Ova procedura je izvedena korištenjem postojećih rješenja u Azure platformi, uz dodatna softverska rješenja i implementaciju koncepata relevantnih za umjetnu inteligenciju, odnosno strojno učenje. Rezultat ovog procesa je sustav koji predstavlja dokaz inovativnog tehnološkog koncepta, unutar kojeg smo identificirali karakteristike i potencijalni komercijalni rezultat primjene digitalnog dvojnika uz korištenje postojećih rješenja unutar Microsoft Azure platforme. Microsoft Azure platforma pruža brojne usluge i servise primjenjiva za digitalne dvojnike, koji nude različit spektar mogućnosti i značajke. Međutim, središnji servis Azure Digital Twins ima ograničenja vezana uz kapacitet konzumacije podataka, što može predstavljati izazov prilikom primjene na kompleksnom stvarnom sustavu s velikim brojem IoT uređaja. Unatoč navedenom, Azure platforma se može iskoristiti za implementaciju digitalnih dvojnika, ali je potrebno zamijeniti određene servise s drugim softverskim rješenjima koji pružaju bolje performanse. U konačnici mogućnost implementacije digitalnih dvojnika pomoću infrastrukture u oblaku ovisi isključivo o krajnjem sustavu na temelju kojeg želimo izgraditi digitalni dvojnika.

Ključne riječi: Digitalni dvojnik, Microsoft Azure platforma, IoT, Angular, MQTT, Unreal Engine

18. Abstract

Implementation of Digital Twin concept using Microsoft Azure platform

In this master's thesis, we have addressed the topic of digital twins and their implementation using the Microsoft Azure platform. The thesis is organized into two parts, with the first part focusing on the theory, including the technologies and concepts that were applied in the second part, which is the practical section. In the practical part of the research, we carried out the process of implementing a digital twin using a case study of a motherboard manufacturing factory. This procedure was performed using existing solutions within the Azure platform, along with additional software solutions and the implementation of concepts relevant to artificial intelligence, specifically machine learning. The result of this process is a system that serves as proof of an innovative technological concept, where we identified the characteristics and potential commercial outcomes of applying a digital twin using existing solutions within the Microsoft Azure platform. Microsoft Azure platform offers numerous services and features applicable to digital twins, providing a wide range of possibilities. However, the central service, Azure Digital Twins, has limitations related to data consumption capacity, which can pose challenges when applied to a complex real-world system with numerous IoT devices. Despite these limitations, the Azure platform can be utilized for implementing digital twins, but it requires substituting certain services with other software solutions that offer better performance. Ultimately, the feasibility of implementing digital twins using cloud infrastructure depends solely on the specific underlying system on which we intend to build the digital twin.

Keywords: Digital Twin, Microsoft Azure platform, IoT, Angular, MQTT, Unreal Engine

SVEUČILIŠTE U SPLITU

Sveučilišni odjel za forenzične znanosti

Izjava o akademskoj čestitosti

Ja, Domagoj Bazina, izjavljujem da je moj diplomski rad pod naslovom „Implementacija koncepta digitalnog dvojnika korištenjem Microsoft Azure platforme“ rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na izvore i radove navedene u bilješkama i popisu literature. Nijedan dio ovoga rada nije napisan na nedopušten način, odnosno nije prepisan bez citiranja i ne krši ičija autorska prava.

Izjavljujem da nijedan dio ovoga rada nije iskorišten u ijednom drugom radu pri bilo kojoj drugoj visokoškolskoj, znanstvenoj, obrazovnoj ili inoj ustanovi.

Sadržaj mojega rada u potpunosti odgovara sadržaju obranjenoga i nakon obrane uređenoga rada.

Split, 26. lipnja 2023. godine

Potpis studenta/studentice: