BUDAPESTI MŰSZAKI ÉS GAZDASÁGTUDOMÁNYI EGYETEM
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Okosautós integrációt és számítógépes vizualizációt megvalósító Big Data komponensek bekapcsolása a SensorHUB keretrendszerbe

Készítette:

Sik Dávid
mérnök informatikus
hallgató

Konzulensek:

Dr. Ekler Péter
egyetemi adjunktus

Dr. Csorba Kristóf
egyetemi docens

Tudományos Diákköri Konferencia
Budapest, 2016

TARTALOMJEGYZÉK

ÖSSZEFOGLALÓ

Napjainkban, az Internet of Things (IoT) világában napról-napra egyre több eszköz, műszer és szenzor kapcsolódik rá a világhálóra. Ezek az eszközök azonban mind-mind eltérő mintavételezéssel, különböző protokollokon, illetve más-más backend kiszolgálókkal állnak kapcsolatban. Ezen eszközök egységes kezelését, fogadását és feldolgozását tűzte ki céljául a SensorHUB keretrendszer.

A keretrendszer lényegét a megfelelő struktúra, és az egyes komponensek összekapcsolódása adja. Ezen komponensek: szenzorok, adatok gyűjtése, helyi szolgáltatások, megjelenítés és adatátvitel; felhő alapú háttérrendszer, Big Data platform menedzsment funkciókkal; szakterület specifikus szoftver komponensek; alkalmazások, szolgáltatások, illetve üzleti intelligencia jelentések és elemzések. A kliens alkalmazások backendje vagy maguk a szenzorok közvetlenül csatlakozhatnak a SensorHUB megfelelő interfészeihez, amin keresztül már könnyen megoldható az adatátvitel, majd az adatok SensorHUB oldali feldolgozása, elmentése és transzformálása.

A Pro Progressio Alapítvány Innovációs Díját 2015-ben elnyerő keretrendszer már több felhasználási területen hasznosításra került, így létezik mezőgazdasági területhez tartozó felhasználás, egészségügyi adatkezelő megvalósítás, okosváros-koncepcióhoz kapcsolódó implementáció (natív szenzorhálózatok adatainak tárolása és feldolgozása), valamint számos további szakterület bekapcsolása is folyamatban van.

Kutatásaim során két új területtel foglalkoztam, melyek feldolgozására a SensorHUB keretrendszert választottam. Az első terület a gépjárművektől érkező adatok kezelése és feldolgozása, ahol a fő cél az adatok hosszú távú tárolása mellett egyfajta összehasonlítás és elemzés is. A dolgozatban az OBD-II interfészről és a CAN bus-ról érkező adatok összehasonlítására vállalkozom a SensorHUB keretrendszerhez kapcsolódó általam fejlesztett ObdCanCompare okostelefonos kliens alkalmazást használva. Mérésekkel megvizsgálom a technológiák jellemzőit és újszerű alkalmazási területeket ismertetek, melyek a keretrendszer segítségével megvalósíthatók.

A második terület a gépi látáshoz kapcsolódik, ahol a cél a vizuális jellegű adatok betöltési lehetőségének biztosítása a SensorHUB keretrendszerbe. A dolgozat keretein belül ismertetem, hogy milyen transzformáció-láncok használhatók a vizuális adatok átalakítására, melyek lehetővé teszik az adatok algoritmikus feldolgozását immár a SensorHUB keretrendszeren belül, egy új belső, általam fejlesztett microservice komponensként.

ABSTRACT

Nowadays, day by day more and more devices, sensors and gadgets are getting connected to the Internet in the world of the Internet of Things. These devices are connecting to different backend services with different sampling rate and different protocols. The aim of the SensorHUB framework is to cooperate with these devices systematically.

The core of the framework is the appropriate structure and the connection of the components. These components are: sensors, data collection, local processing, client side visualization and data transmission; cloud based backend with Big Data analysis and management; domain specific software components; applications, services, business intelligence reports, dashboards. The backend service of the clients or the sensors are directly connecting to the corresponding interface of the SensorHUB, through which the data transmission and the SensorHUB side processing, storing and transformation can be handled easily.

The framework which won the Innovation Award of Pro Progressio Foundation in 2015 has been used in several domain area, like agricultural utilization, health data management implementation or smart-city application (storing and processing native sensor data). Connecting of other domain areas is an open research and design.

During my research I am dealing with two new domains. For their processing I have chosen the SensorHUB framework. The first new domain is processing, storing and comparing data exracted from vehicles. In this dissertation I compare the data from OBD-II and CAN bus interfaces using my newly implemented smartphone application called ObdCanCompare, measure the domain specific parameters, introduce new applications based on the data and framework.

The second domain is the computer vision, where the aim is to load the visual data into the SensorHUB. In my dissertation I introduce the transformation-chains to convert the visual data, which helps their algorithmic procession using the SensorHUB framework as a new microservice component developed be me.

1. BEVEZETÉS

Napjainkban az okos eszközök és a hozzájuk kapcsolódó szolgáltatások egyre jelentősebb szereppel bírnak a legtöbb iparág számára.

Az elmúlt évtizedekben nagyban nőtt az internetre kapcsolt eszközök száma, vagyis kiépült az Internet of Things (IoT). Ez főként az okostelefonok és a hordozható „kütyük" fejlődésének, növekedésének és elterjedésének köszönhető. Néhány tényadat a jelenlegi mobilvilággal kapcsolatban [1]:

- 4,5 milliárd telefon van a világban, ennek mintegy a fele okostelefon.
- A mobil-előfizetők száma közel 7 milliárd.
- Az eszközök kétharmadán van kamera, amelyet a felhasználók 72%-a használ is.
- A mobiltulajdonosok 91%-a 7×24 órában a telefonja közelében van.
- Minden harmadik ember olvas híreket a telefonján.
- Több mint egymilliárd ember tölt le alkalmazásokat, játszik, használ közösségi hálót.

A tendencia tartja magát, sőt a következő évekre további növekedés várható, ahogy újabb megoldási lehetőségek, technológiák kerülnek előtérbe.

Év	Eszközök száma
1984	1000
1992	1 000 000
2008	1 000 000 000
2013	10 000 000 000
2020~	50 000 000 000

1. táblázat: Az internetre kapcsolt eszközök száma jelenleg és a jövőben [1]

Ha egy okostelefonra nemcsak telefonként tekintünk, hanem szenzorok halmazaként, akkor könnyen beláthatjuk, hogy ezen adatok felhasználásával hihetetlen mennyiségű információ nyerhető ki.

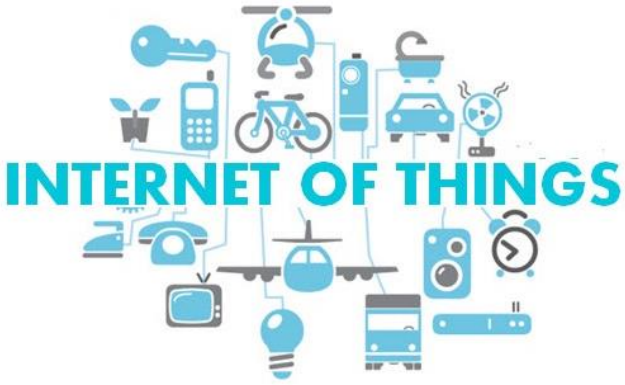
Gondoljuk csak a földrajzi koordinátákra (mind GPS, mind a bázisállomások alapján), a mikrofonra, a kamerára, a gyorsulásmérőre, a fényérzékelőre, pulzusszámmérőre stb. És ezek csak a legalapvetőbb, szinte minden okostelefonban és okosórában elérhető mérőegységek.

2017-ben egy IP-alapú szenzor átlagos ára nagyságrendileg 500 Ft lesz, 2020-ban pedig várhatóan 30 milliárd IP-szenzor lesz működésben. [1]

Ezek adathalmazait az interneten keresztül összegyűjtve egy intelligens környezet, egy világméretű kép alakulhat ki az okoseszközöket használók tulajdonságairól, szokásairól. Az adatok segítségével akár a jövőbeli tevékenységeiket, szándékaikat is meg lehet jósolni.

Az IP alapú szenzorok működéséhez a nevéből következően szükség van IP címekre is. 2011. február 3-án az internet központi adminisztrációjáért felelős IANA (Internet Assigned Numbers Authority) kiosztotta az utolsó blokkokat a közel 4 milliárd IPv4-es címek közül, ezzel kifogyasztva az összes globális szabad tartalékot. [2] Ez nem volt váratlan esemény, már az 1980-as években is elkezdtek kialakítani további technológiákat a korlátos tartomány kiküszöbölésére, így 1981-ben a címosztályokat használó hálózatkezelést, 1983-ban az osztály nélküli forgalomirányítást és a hálózati címfordítást, 1999-ban pedig megszületett az új szabvány, az IPv6. [3]

Összehasonlításképpen az IPv4-es címek 32 bitesek, így az elméleti maximálisan kioszthatóak címek száma $4*10^9$ darab, az IPv6-os címek 128 bitesek, így az elméleti maximálisan kioszthatóak címek száma $3*10^{38}$ darab cím. (Az elméleti szó fontos, mert mindkét esetben nem számítanak a privát tartományok és egyéb különleges címek.)

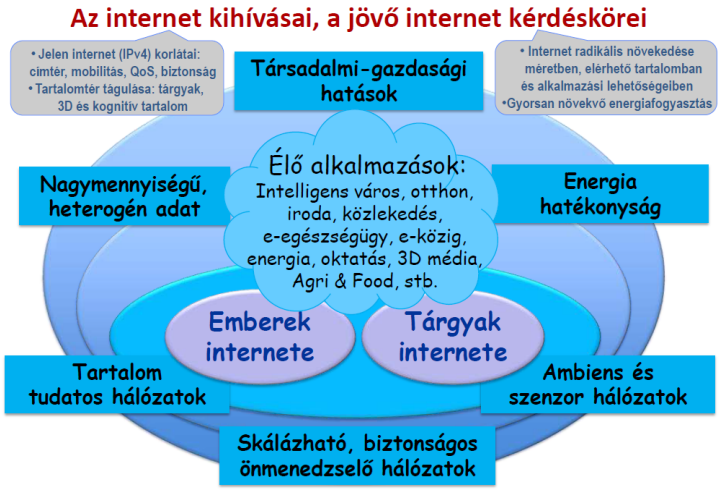


1. ábra: Az Internet of Things világa [4]

Ha összevetjük az emberiség létszámát ($7,4*10^9$ fő) [5] a kioszthatóak IP-címekkel látható, hogyha mindenkinek egyetlen internetre csatlakozó eszköze lenne, akkor sem lenne elég az IPv4-es címtartomány. (Persze itt is éltem mindenféle tényezők elhanyagolásával: egyidejűség, belső hálózat, fejlődő országok, stb.) Ha pedig vesszük napjaink eszközeit melyek az internetre csatlakoznak, így kiindulásképpen először a hálózati kártya az asztali számítógépünkben, aztán a notebookunkban, a tabletünkben, a telefonunkban, a másik telefonunkban, a nyomtatónkban, az IP-kameránkban, a riasztónkban, a hűtőnkben, a mindenféle szenzorunkban, a további háztartási eszközeinkben, járműveinkben, akkor a szükséges IP címek mennyisége robbanásszerűen meg tud növekedni.

Az IPv6-os címtartományra történő átállás sem zökkenőmentes folyamat, az internetszolgáltatók próbálták húzni, halasztani az átálláshoz szükséges infrastruktúra kiépítéseket és átszervezéseket, ameddig csak lehetett. 2008-ban indultak meg az első tesztelések, azóta egyre több és több helyen vált elérhetővé az IPv6-os címzés és a hozzákapcsolódó további szolgáltatások. [6]

Az IPv6 globális elterjedésével pedig lehetővé vált, hogy az eddigi „emberek internete” mellé a fogalmaink közé, valamint az internet világába becsatlakozzon a „tárgyak internete” is, mely stabil alapot nyújthat egy globális internet alapú társadalomhoz és a ráépülő élő, intelligens alkalmazásokhoz (város, közlekedés, otthon, e-közigazgatás, energiahasznosítás, oktatás, 3D média stb. mind-mind az okos- előtaggal). [7]



2. ábra: Az internet kihívásai, a jövő internet kérdéskörei [7]

Azonban amikor ennyi eszköz csatlakozik a világhálóra, nem egyszerű feladat a különböző adatfolyamok és kommunikáció megvalósítása, egységes kezelése. Főleg, hogy már régebb ideje ad-hoc módon kezdtek el különböző eszközöket, különböző protokollokkal, különböző módokon csatlakoztatni a világhálóra. Másik feladat pedig a hatalmas mennyiségű adat tárolása, és későbbi hasznosítása. Ezen adatokkal kapcsolatos feladatokra került bevezetésre a Big Data fogalom és a kapcsolódó elméleti koncepciók. Amíg az adatok feldolgozása egy szabvány vagy szervezet fennhatósága alatt van, addig az adott terület adatai, protokolljai egységesnek gondolhatóak, azonban ha már például két különböző szervezet infrastruktúrája között kell adatátvitelt megvalósítani, felmerülhetnek problémák.

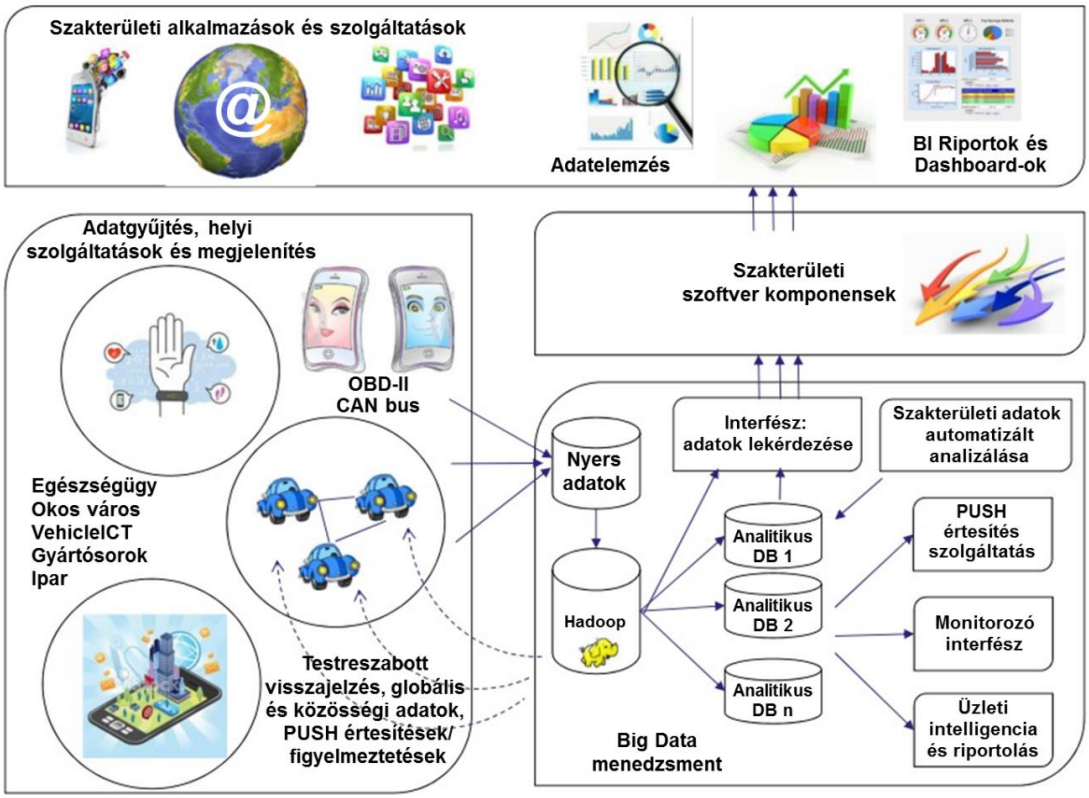
Ennek a megvalósítására egyelőre nem nagyon várható globális protokoll, azonban a feladatok hasonló kezelése miatt lehetőség van egy egységes felületre, ahol a hasonló feladatok logikáját csak egyszer kell implementálni és utána az adott szakterületre való leképezés és finomhangolás már könnyen, apró módosításokkal megvalósítható.

A következő fejezetben bemutatom a SensorHUB koncepciót és keretrendszert, annak különböző verzióit, a keretrendszer elemeit, valamint eddigi felhasználásokat. A harmadik fejezetben az általam a SensorHUB rendszerhez hozzátett komponenseket mutatom be. A negyedik fejezetben kitekintek a SensorHUB rendszerben rejlő jövőbeli lehetőségekre, célterületekre. Végül TDK dolgozatomat összefoglalással, köszönetnyilvánítással, táblázat-, ábra-, és irodalomjegyzékkel, valamint függelékkel zárom.

2. A SensorHUB koncepció és keretrendszer

A közös megvalósítás gondolatára kezdett körvonalazódni a SensorHUB koncepció és keretrendszer a Budapesti Műszaki és Gazdaságtudományi Egyetem Automatizálási és Alkalmazott Informatikai Tanszékén 2013-ban. Először egyszerű tanszéki projektnek indult, aztán az elmúlt évek során egyre több hallgató, doktorandusz kapcsolódott be a fejlesztésbe. Mindenki a saját legjobb területén dolgozott, ezzel segítve a projekt erősítését és kiemelését. Számos SensorHUB-os kötődésű szakdolgozat, diplomaterv született az elmúlt években.

Az előző fejezettel összhangban a SensorHUB képességeinek is számos alkalmazási területei lehetnek, például az egészségügy, okos városok, járműipar, gyártósorok stb. Ezek tipikusan olyan területek, ahol nagy mennyiségű szenzoradatok feldolgozása, tárolása, elemzése és felhasználása a feladat; tudjuk mi történik, és eszerint be is tudunk avatkozni.



3. ábra: A SensorHUB koncepció [8]

A SensorHUB koncepció elemei [8]:

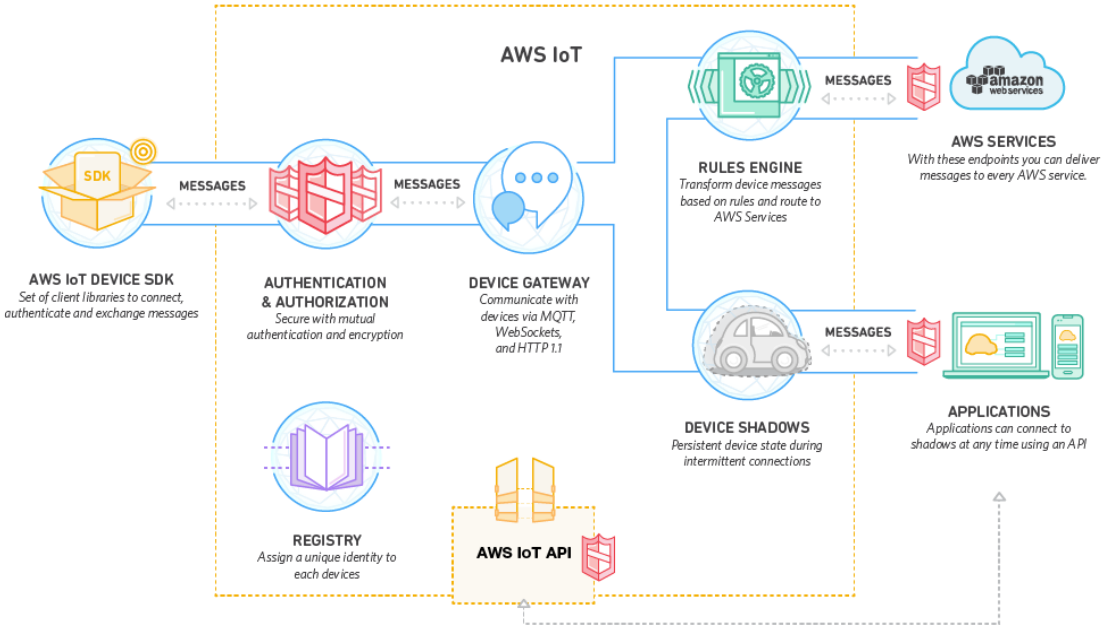
- Szenzorok, adatok gyűjtése, helyi szolgáltatások, megjelenítés és adatátvitel.
- Felhő alapú háttérrendszer, Big Data menedzsment funkciókkal.
- Szakterület specifikus szoftver komponensek.
- Alkalmazások, szolgáltatások, üzleti intelligencia jelentések és elemzések.

A SensorHUB koncepciónak előnye, hogy a gyakran előforduló, adatokkal kapcsolatos műveleteket csak egyszer kell implementálni és ezek után megfelelő dokumentációkkal támogatva bárki tud olyan alkalmazást készíteni, ami felhasználva a SensorHUB keretrendszert, képes lesz adatai megjelenítésre, feltöltésére, elemzésére. [9]

2015-ben a SensorHUB nyerte a Pro Progressio Alapítvány Innovációs Díját. [10]

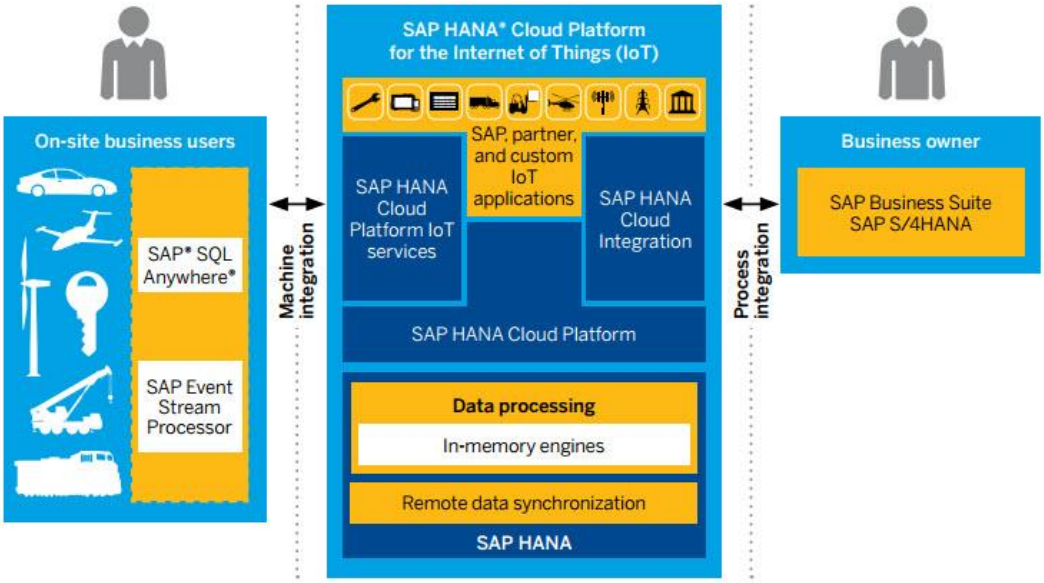
Továbbá a SensorHUB koncepció létjogosultságát mutatja, hogy számos nagy cég és szolgáltató szervezet készített hasonló PaaS (Platform as a Service; platform, mint szolgáltatás) jellegű koncepciókat, keretrendszereket.

Az egyik ilyen az AWS IoT platformja (Amazon Web Services), mely kihasználja az Amazon cég rendelkezésére álló szerverfarmot, valamint a további számítási kapacitást igénylő szolgáltatásait (Amazon S3, Amazon EC2, Amazon Machine Learning, Amazon DynamoDB stb.), így képes könnyen és biztonságosan összekötni eszközöket a felhő alapú szolgáltatásukkal: „kapcsolatot teremt milliárdnyi eszköz között billiónyi üzenettel”. [11] Az AWS IoT elérhető Amazon-os fizetős szolgáltatásként is bárki számára.



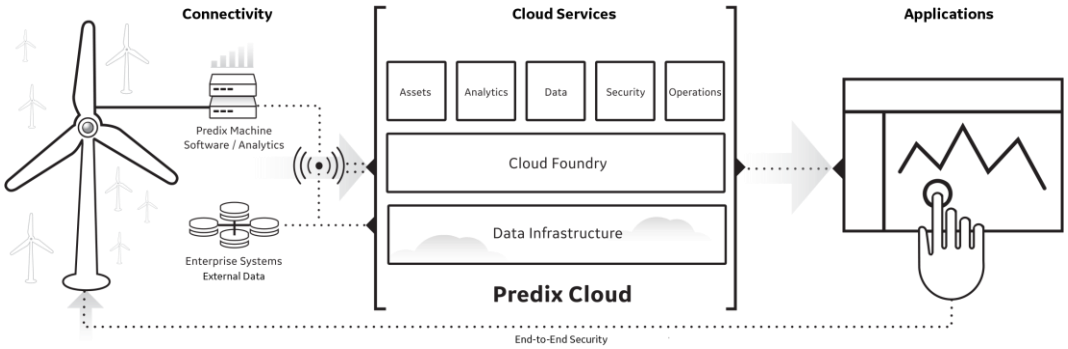
4. ábra: Az Amazon Web Services IoT platform felépítése [11]

Egy másik nagyvállalati megoldás az SAP HANA Cloud Platformja, mely szintén számos szolgáltatást támogat (távoli eszköz menedzsment, üzenetkezelés, alkalmazások használata stb.). A felépítésben kicsit más szemléletű az SAP szervezeti jellege miatt, így megkülönböztetik az felhasználó oldali üzleti résztvevőket (eszköz alapú integráció) és a folyamat integrációs üzleti résztvevőket. [12]



5. ábra: Az SAP HANA Cloud Platform felépítése [12]

Egy harmadik implementáció a General Electric Predix rendszere, ahol még nagyobb szerepet kap az IIoT, vagyis az Industrial Internet of Things, mely különböző ipari környezetben támogatja az eszköz adatok internet alapú feldolgozását. [13]



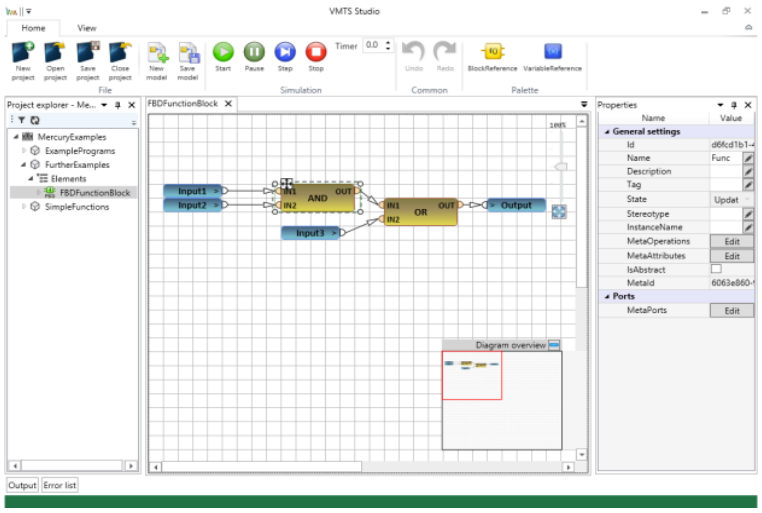
6. ábra: A GE Predix felépítése [13]

A SensorHUB keretrendszer felépítésének bemutatása után láthatóvá válik, hogy a mi keretrendszerünk is számos szolgáltatást biztosít a nagyvállalati és ipari megvalósítások közül. Ezen rendszerekkel bár versenytársak nem lehetünk, de kisvállalati környezetben egy egyedi igényekre igazítható, és jobban skálázható rendszert tudunk szolgáltatni.

Egy továbbgondolása a SensorHUB koncepciónak, amivel a nagyvállalati irányba is lehet nyitni, valamint egy új rálátást nyit az IoT alapú világra, az nem más, mint a modellvezérelt paradigma bevezetése.

Ennek a koncepciónak a szintén tanszéki fejlesztésű *Visual Modeling and Transformation System* (VMTS) az alapja, mely egy gráf alapú, szakterület specifikus, (meta)modellező és modellfeldolgozó keretrendszer. [14]

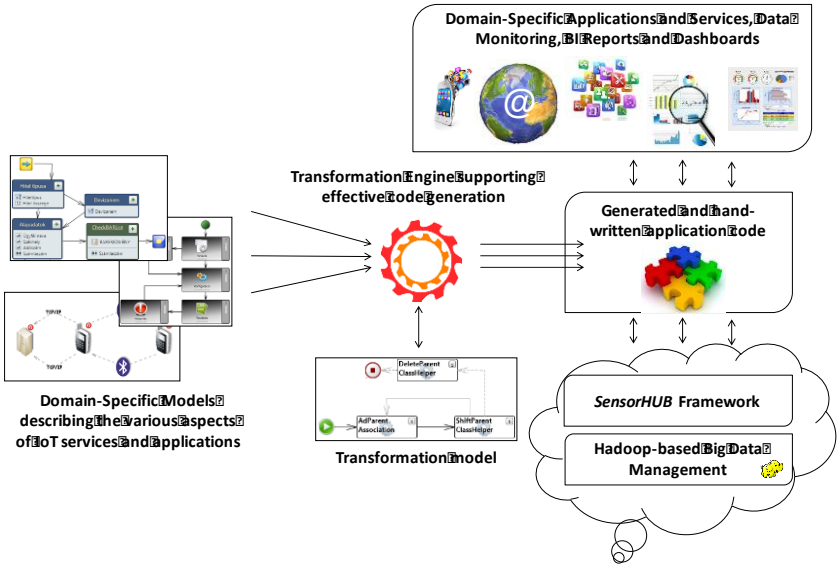
A keretrendszer tartalmaz egy grafikus felületet, ahol lehet definiálni, testreszabni és alkalmazni a különböző szakterületi nyelveket.



7. ábra: A VMTS Studio felülete [14]

A koncepció felépítése [15]:

1. IoT (szenzorok, eszközök, kapcsolódások és szolgáltatások)
2. Több szakterületes IoT (SensorHUB)
3. Modellvezérelt több szakterületes IoT (VMTS + SensorHUB)



8. ábra: A modellvezérelt több szakterületes IoT koncepciója [15]

A SensorHUB keretrendszer a SensorHUB koncepció realizálása. Az implementálás során cél volt a koncepcióban definiáltak megvalósítása és a koncepció minél több szakterületre történő kiterjeszthetősége.

A keretrendszer alapja a nagy mennyiségű adat kezelésére képes Big Data fájlrendszer, tudni kell ebbe betölteni és ebből kiolvasni az adatokat, minél több interfészt támogatva.

Napjainkra egyre több, egyre összetettebb Big Data fájlrendszer disztribúciók és kiegészítések léteznek vagy vannak fejlesztés alatt, így széles tárházból lehet válogatni ezek közül.

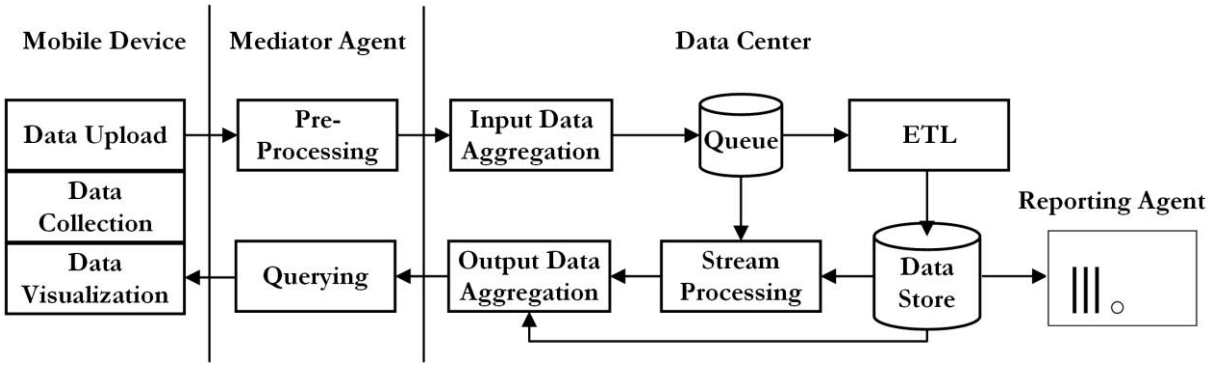
A cél ugyanaz, minél több, hasznos funkciót és technikát egyetlen egyszer jól megírni úgy, hogy ezek minél több alkalmazásba beintegrálhatóak legyenek a későbbiekben.

2.1. SensorHUB 1.0

Az utólag SensorHUB 1.0 névre keresztelt változata a keretrendszernek, még egy egyszerűbb, szűkebb infrastruktúrát tartalmaz.

Ebben a változatban a fő irányvonal az okostelefonos alkalmazások integrációja volt a keretrendszerbe és a kommunikáció megvalósítása a szerveroldali komponenssel.

A SensorHUB 1.0 fő eleme egy Android osztálykönyvtár, melyben a kommunikációval, megjelenítéssel és adatkonverzióval kapcsolatos metódusok és hálózati hívások kerültek megvalósításra. Ezek a funkciói az osztálykönyvtárnak egy Android alkalmazás projektjébe történt beimportálást követően elérhetővé válnak a kódból és hasznosíthatóak a hívások. [16]



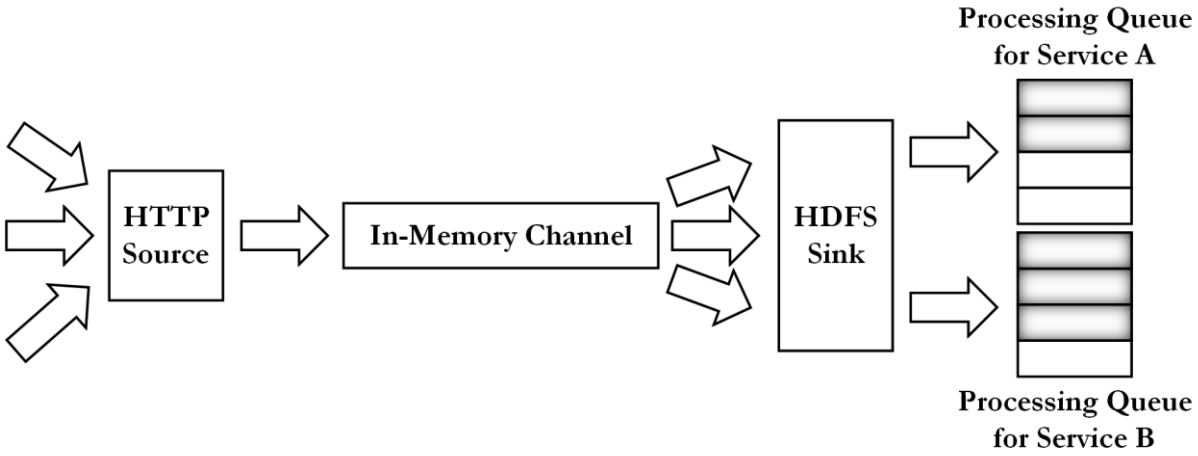
9. ábra: A SensorHUB 1.0 architektúra [16]

A mobil alkalmazás és a Hadoop-os Big Data fájlrendszer közti kapcsolatot egy Node.js szerveroldali komponens teremti meg, mely a Hadoop felé adat elő-feldolgozást, az okostelefonos kliens felé pedig egy puffert és adatlekérési felületet biztosít.

Miután mind a Hadoop-os Big Data rendszer, mind a Node.js szerveroldali komponens a SensorHUB 2.0-ban is felhasználásra került, ezért ezek részletes bemutatása a következő fejezetben olvasható.

A Hadoop rétegben, akár az online felületen keresztül, akár valamilyen riportkészítő alkalmazás segítségével kinyerhetőek a számunkra érdekes adatok. Ezt ETL (Extract-Transform-Load) transzformációkkal tehetjük meg, melyben konvertálhatjuk, szűrhetjük, tisztíthatjuk az adatainkat, hogy a megjelenítés vagy további felhasználás ellentmondás-

mentes legyen. A transzformáció végén elkészült adatot a Node.js szerver segítségével visszaküldhetjük az okostelefonra, vagy elérhetővé válhat szintén riportként.



10. ábra: HTTP kérések továbbítása és elkülönítése [16]

A Node.js szerver tudja, hogy mely alkalmazások kapcsolódnak a SensorHUB-hoz, így a beérkező adatokat megfelelően továbbítani tudja a Hadoop interfész felé, ahol megtörténik az adatok feldolgozása, eltárolása. Itt kap fontos szerepet a Node.js komponens REST (*REpresentational State Transfer*) alapú [17] hálózati interfésze melyen keresztül zajlik a kommunikáció, leginkább JSON (*JavaScript Object Notation*) vagy XML (*Extensible Markup Language*) alapú formátumban. A REST interfészben, HTTP (*HyperText Transfer Protocol*) kérések felhasználásával történik az adatátvitel, vagyis GET (adat lekérés a szervertől), illetve POST (adatküldés a szervernek) metódusokkal. [18]

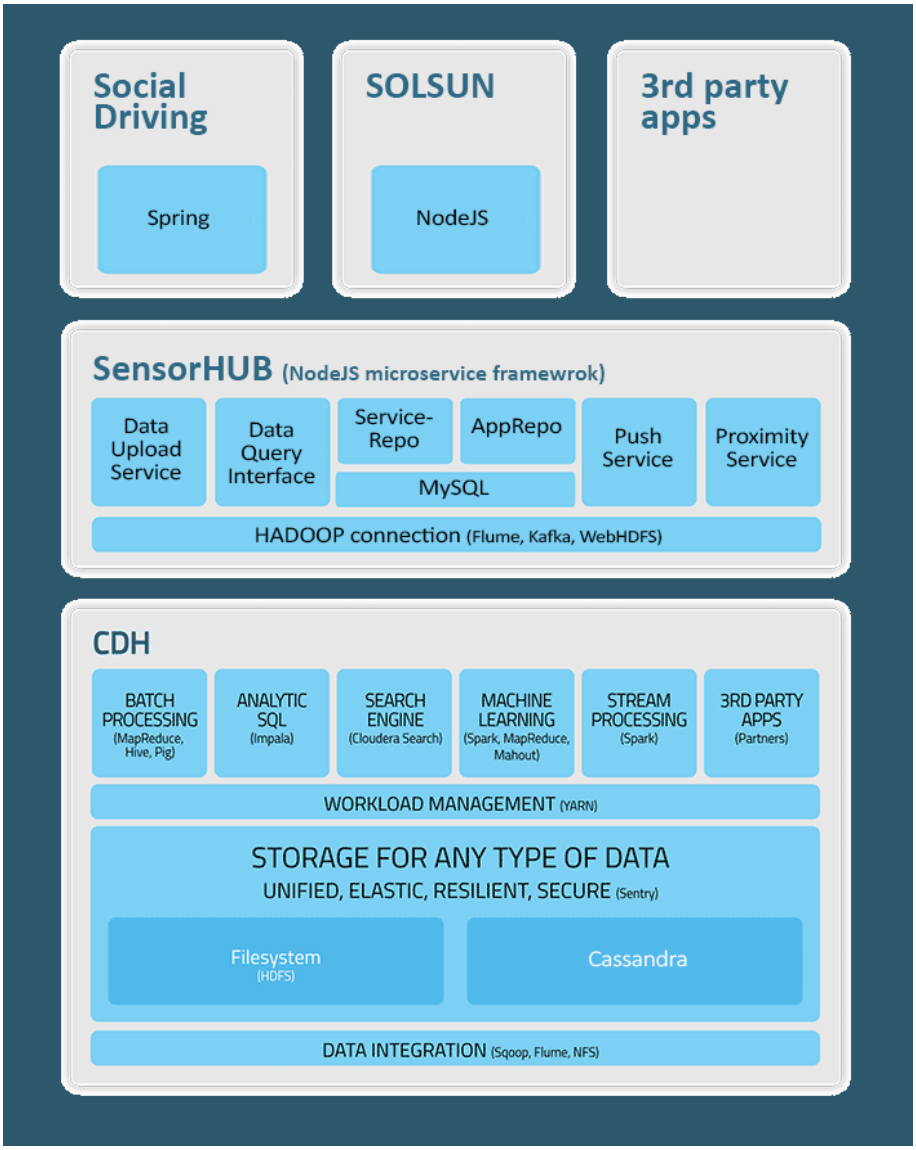
A Node.js szerver további feladata egy egyszerű adminisztrációs webes felületet biztosítani, ahol tájékozódni lehet a kapcsolódó alkalmazások és a Hadoop fájlrendszer állapotáról, metrikákról, lehetőség van figyelni az erőforrás kihasználtságokat. További opció van az alkalmazások adatáramlásának letiltására vagy engedélyezésére.

A riportkészítő alkalmazás pedig bármelyik Hadoop klaszterhez kapcsolódni képes megvalósítás lehet, mi erre a célra a Pentaho üzleti intelligencia rendszert választottuk.

2.2. SensorHUB 2.0

A SensorHUB 2.0 az előző verzióhoz képest egy teljesen újragondolt változatot jelent a megvalósítás során. Csökkent a hangsúly az eddigi főleg Cloudera Hadoop Big Data alapú megvalósításon, és nagyobb hangsúlyt kapott a szerveroldali komponens, ugyanis bevezetésre került egy egyedi tervezésű Node.js alapú microservice réteg, mely összekapcsolja a kliens oldali alkalmazásokat a Hadoop klaszterrel, és számos további adminisztrációs, menedzsment, terheléselosztó (replikációs) és kommunikációs feladatot is ellát. [19]

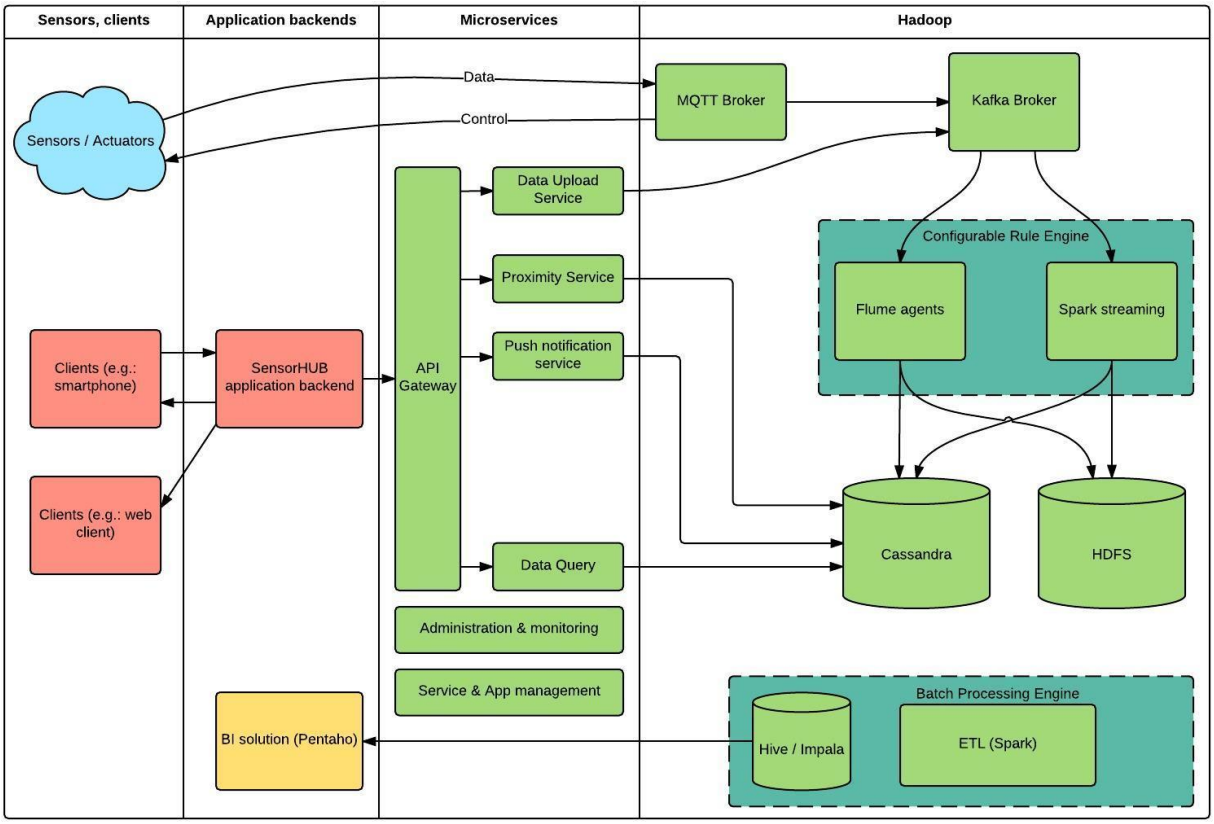
Az új verzióban jóval nagyobb szerepet kapott az időközben kurrensé vált Big Data implementációk kihasználása, az egyedi programcsomagok integrálása az adatfolyamba.



11. ábra: A SensorHUB 2.0 technológiai felépítése [19]

A technológiai felépítésből látható, hogy a SensorHUB 2.0 továbbra is három rétegből áll:

- CDH: ami a Cloudera Hadoop disztribúcióját és a hozzákapcsolódó további Big Data implementációkat takarja.
- Node.js alapú microservice réteg: ami a SensorHUB lelkét adja, itt találhatóak a gyakran használt szolgáltatások implementációi, menedzsment funkciók, adatkommunikációs interfészek.
- Végül a SensorHUB keretrendszerhez kapcsolódó és az erőforrásokat felhasználó alkalmazások.



12. ábra: A SensorHUB 2.0 architektúra [19]

A SensorHUB 2.0 architektúrájából látható, hogy itt már az okostelefonos klienseken kívül webes kliensek, illetve „natív” szenzor komponensek is kapcsolódhatnak a rendszerhez.

Az architektúra ábra első két oszlopa, ami az alkalmazás függő rész, vagyis maguk a kliensek, és a hozzájuk kapcsolódó szerveroldali komponensek (a szenzorok backend nélkül kapcsolódnak a SensorHUB-hoz). Az ábra utolsó két oszlopa pedig a SensorHUB rész, vagyis a microservice réteg és a Hadoop réteg.

A szenzorok tehát közvetlenül egy erre felkészített MQTT (*Message Queuing Telemetry Transport*) alapú publish-subscribe interfészhez kapcsolódnak a SensorHUB Hadoop felületén. Ez egy egyszerű, hat metódussal rendelkező hálózati protokoll, mely minimális feszültség és minimális hálózati erőforrást vesz igénybe a kommunikációhoz. [20]

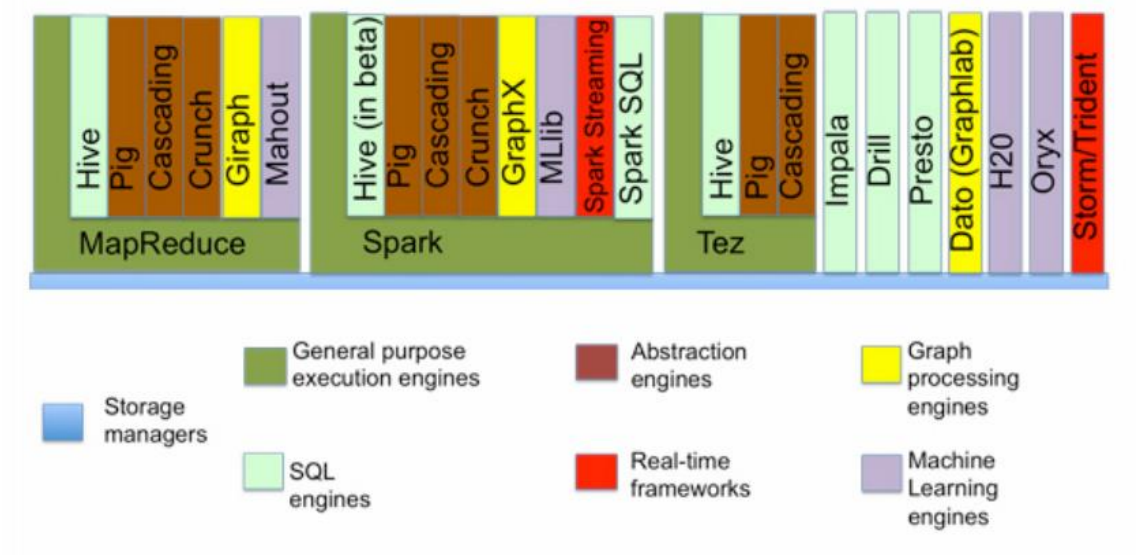
A Batch Processing Engine pedig SensorHUB 2.0-hoz szintén kapcsolható riportkészítő komponsensek számára készíti elő az adatkötegeket, Apache Spark alapú ETL transzformációkkal. Az így elkészült adathalmazból pedig az Apache Hiva vagy Cloudera Impala SQL-szerű lekérdezőnyelvek segítségével kérheti le a Pentaho rendszer azokat a részhalmazokat, melyhez a riportnak, vagy a felhasználóknak szüksége van. [21]

A többi komponens bemutatása a következő fejezetekben kerül sorra.

2.2.1. CDH réteg

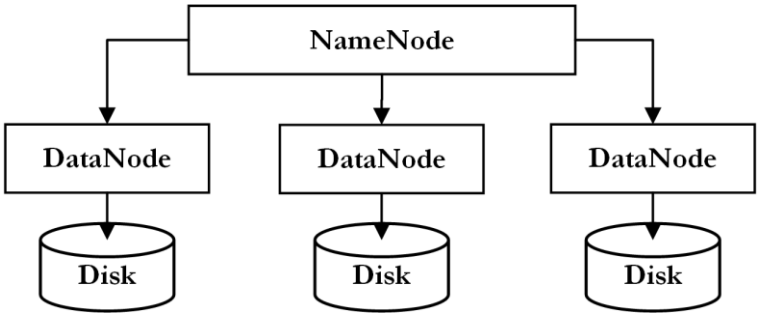
A CDH rövidítés a *Cloudera Distribution Including Apache Hadoop* kifejezést takarja, vagyis a Cloudera cég egyik disztribúciója mely tartalmazza a Hadoop rendszert, ezáltal integrálgatóvá válnak további Apache Hadoop kiegészítések. [21]

Mivel napjainkra nagyon elterjedté vált már a Hadoop rendszer és annak disztribúciói, így a következőkben ezt az ökoszisztémát mutatom be röviden.



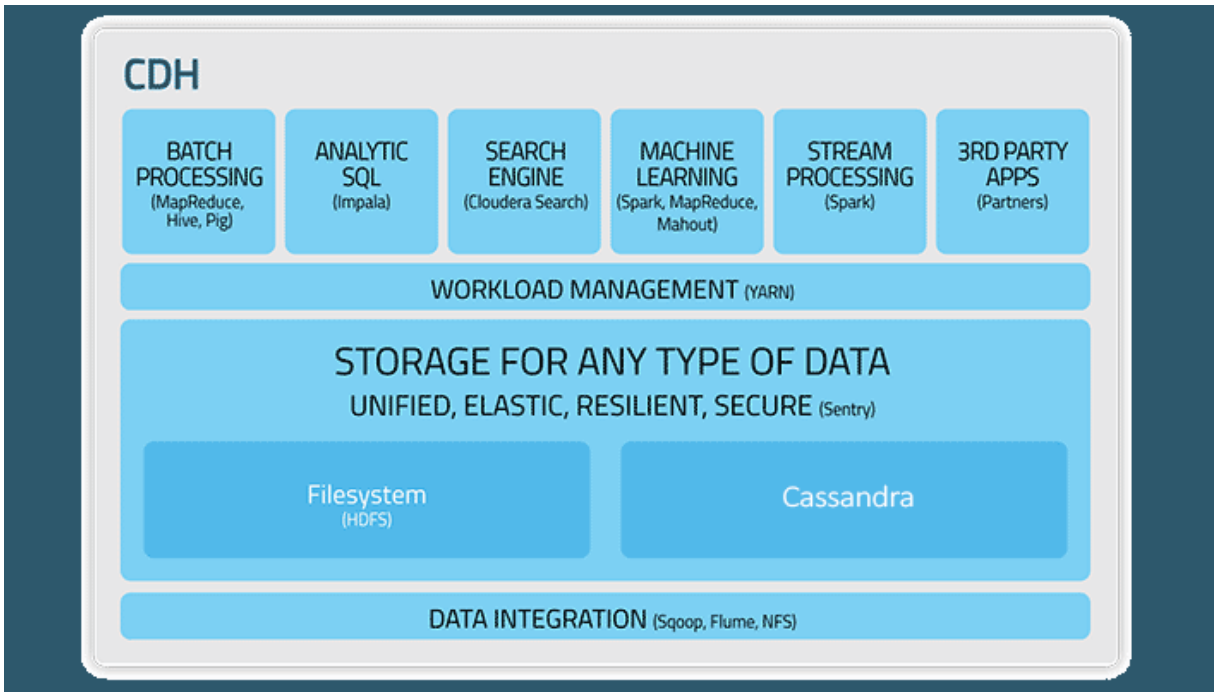
13. ábra: A Hadoop ökoszisztéma [22]

A SensorHUB Hadoop rétege, valamint a Hadoop ökoszisztémája eléggé hasonlít egymásra. Legalul helyezkedik el az adatok tárolásával és integrációjával kapcsolatos réteg, mely egy egységes, rugalmas, ellenálló és biztonságos adattárolást tesz lehetővé, a HDFS, vagyis a *Hadoop Distributed File System*. A HDFS egy elosztott fájlrendszer, a Hadoop szisztéma egyik alapja. Lényege, hogy a nagy mennyiségű adatokat, elosztva, több számítógépen tároljuk. Ezt elrejti előlünk a fájlrendszer, és mi, vagy a kapcsolódó szolgáltatások együtt, egy helyen látjuk az adatokat. Ennek megvalósítását segíti a *NameNode*, ami ismeri az egyes gépeken lévő *DataNode*-okat, amiken keresztül elérhetőek az adatok. A *NameNode* tudja, hogy mely része, melyik gépen található meg a fájloknak, követi az egyes gépek életciklusát, tárolja a metaadatokat. [22]



14. ábra: A HDFS architektúra [16]

A HDFS-re épül a YARN, vagyis a *Yet Another Resource Negotiator*, az erőforrás menedzser. A Hadoop 1.0 verziót követően a YARN látja el a HDFS feletti operációs rendszer-szerű feladatokat, a MapReduce paradigmát támogatva. Előnye, hogy számos további paradigmát támogat, gyorsabb adatelérésre képes, és ez tette lehetővé, hogy a HDFS-re további hasznos komponensek váljanak elkészíthetővé. Az erőforrás-kezelőhöz kapcsolódhat például a Spark, Tez, illetve az alap MapReduce is. De vannak olyan komponensek is, melyek YARN nélkül, közvetlenül kapcsolódnak a HDFS-hez. [22]

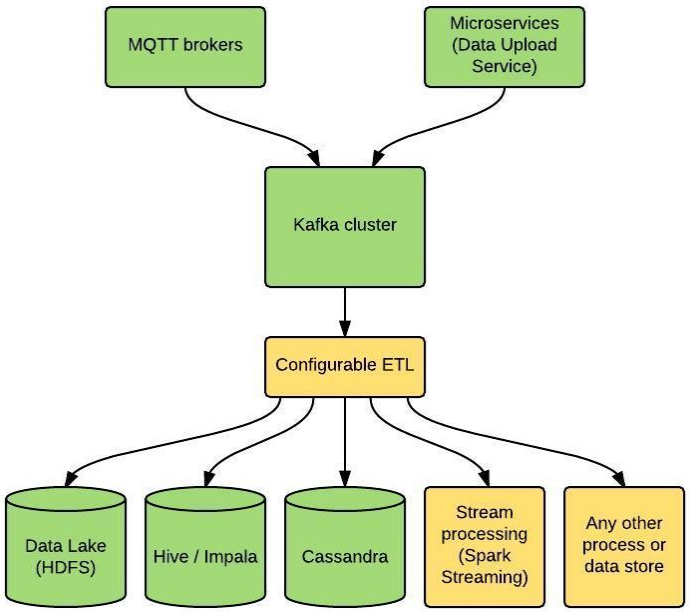


15. ábra: SensorHUB CDH (Hadoop) réteg [19]

A SensorHUB-ban használt komponensek [21][22]:

- MapReduce: A Google által definiált programozási modell, mely az elosztott adatok hatékony feldolgozását, és abból eredmények elkészítését teszi lehetővé. Két lépésből áll: map és reduce. A map során kulcs-értékpárokból, kulcs-értékpár listát készít, a reduce során pedig a listában azonos kulcshoz tartozó párok egy listába kerülnek.
- Hive: A Facebook által készített SQL alapú lekérdező nyelv a Hadoop rendszerben, segítségével az SQL-ből ismert kulcsszavak felhasználásával nyerhetünk ki adatokat.
- Pig: SQL-re hasonlító szkriptnyelv, MapReduce job-ok létrehozására, nevét onnan kapta, hogy szinte bármit leírhatunk benne, mert bármit megeszik.
- Impala: A Hive-hoz hasonló, de annál gyorsabb, a Cloudera által készített C++ alapú lekérdező nyelv.

- Cloudera Search: Bármilyen lekérdező nyelv ismerete nélkül keresési lehetőséget biztosít a fájlokban.
- Spark: Memória alapú MapReduce keretrendszer, gyorsabb, valósidejű, iteratív számításokhoz ideális. Különleges adategységeken (RDD) dolgozik, két metódusa van transformation és action. Míg az első RDD-vel tér vissza, addig a második valamilyen eredménnyel. Csak action hívás során történik meg a transformation-ök kiértékelése, de cache-elhető, így csökkenthető a kiértékelési idő.
- Mahout: Statisztikai és gépi tanulás algoritmusokat tartalmazó kiterjesztés.
- Flume: Gondoskodik az integrációról és az microservice rétegből érkező adatok HDFS-be vagy más rendszerbe (NFS: Network File System vagy Cassandra NoSQL adatbázisba) történő mentéséről.
- Sqoop: Relációs adatbázisba történő mentésről gondoskodik.
- Sentry: Valós idejű jogosultságkezelés, hibakövetés és loggolás.



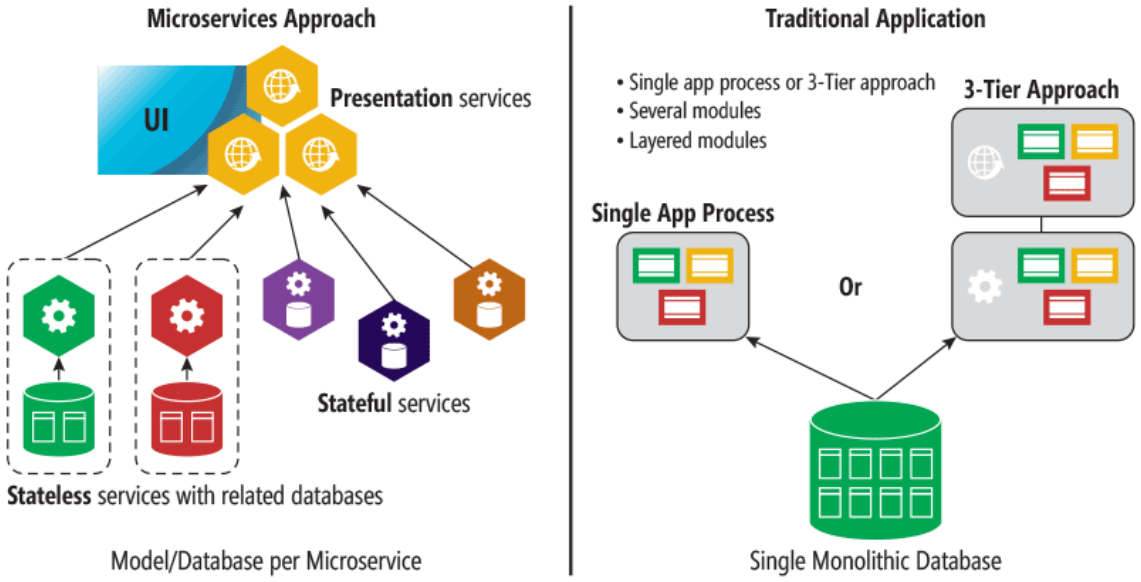
16. ábra: Kafka üzenetsor [19]

- Kafka: Big Data alapú elosztott üzenetsort valósít meg. Ez a Hadoop réteg központi adat beérkeztető pontja, amin minden adat keresztülhalad. A Kafka komponensben tetszőleges ideig tárolhatóak (sorba állítás és pufferelés funkciók) a különböző források felől érkező adatok, majd a megfelelő szolgáltatásokkal innen menthetőek le, vagy küldhetőek tovább valamilyen feldolgozásra akár ETL-lel, akár valamilyen lekérdező nyelv segítségével.

2.2.2. Node.js microservice réteg

Következik a réteg, amitől SensorHUB a SensorHUB, vagyis a microservice réteg bemutatása a technológiai felépítésből.

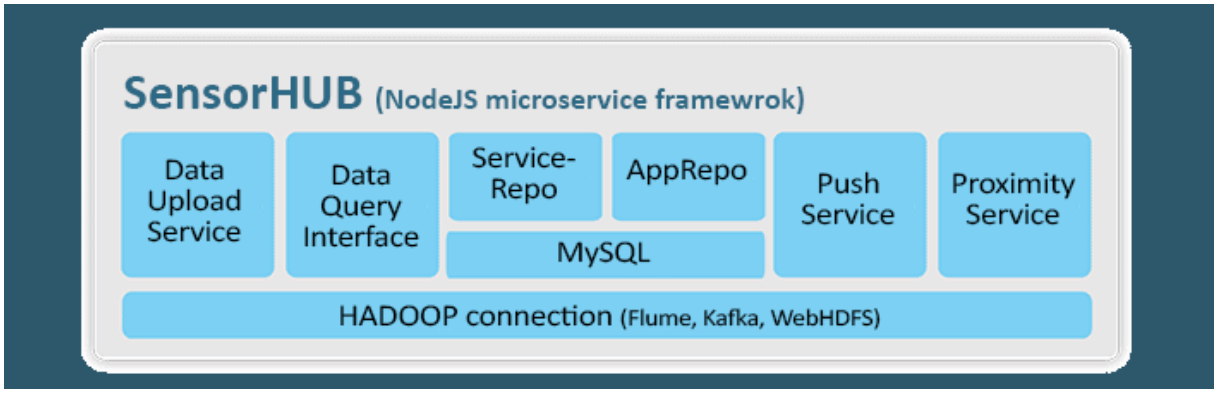
A microservice réteg teremti meg a kapcsolatot a Hadoop réteg és a felhasználói alkalmazások között. Ebben a rétegben kerültek megvalósításra a közös, gyakran használt funkciók, melyek beépíthetőek az alkalmazásba, valamint ez a réteg működik közre az adatáramlásban a Hadoop és az alkalmazások között.



17. ábra: A microservice és a monolitikus architektúra összehasonlítása [24]

A microservice rétegben található komponensek mind-mind önálló, autonóm szerver komponensek, dedikált feladattal futnak, és dedikált célt látnak el, egymással és a kapcsolódó kliensekkel REST alapú hálózati hívásokkal tudnak kommunikálni. [23][24]

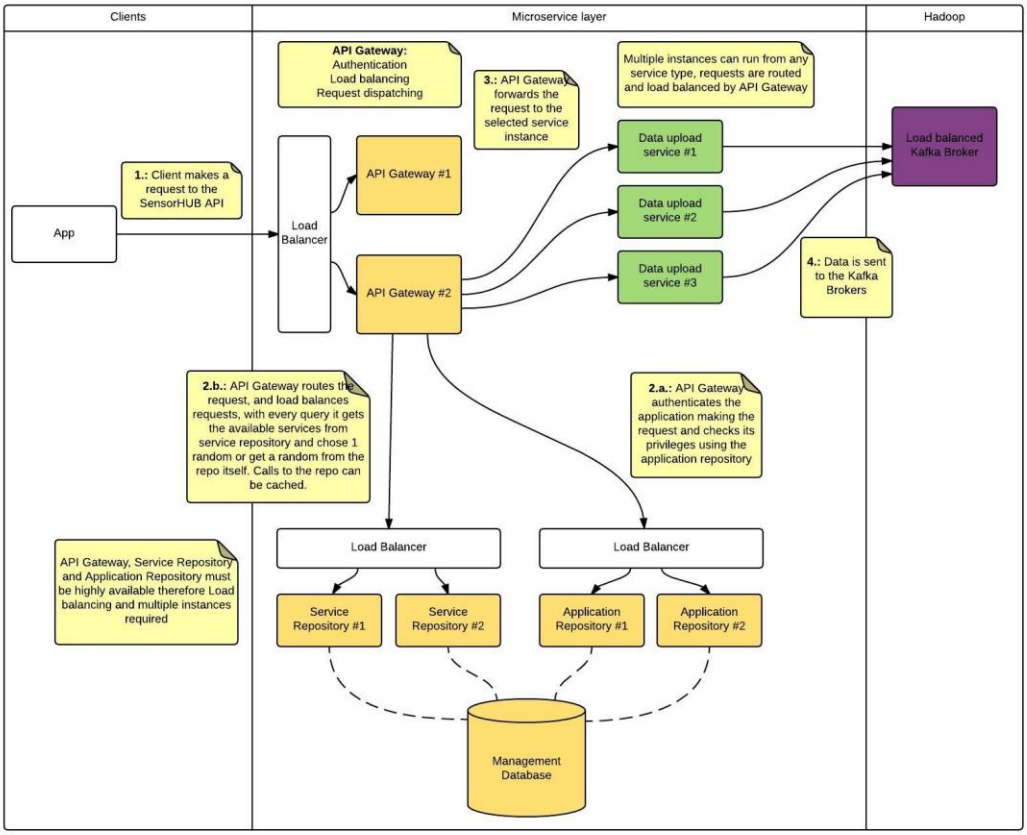
A komponensek lazán csatoltsága miatt könnyű egy újabb komponens elkészíteni, és elérhetővé tenni a rendszerben, ezáltal, a szükséges komponensek építőkockaként tehetőek össze az alkalmazásunkhoz.



18. ábra: SensorHUB microservice réteg [19]

A SensorHUB-ban használt komponensek [19]:

- API Gateway: A SensorHUB-ot használó alkalmazások belépési pontja a keretrendszerhez. Validálja a beérkező kéréseket és továbbítja a megfelelő komponenshez. Terheléselosztóval rendelkezik, replikált szolgáltatás.
- Data Upload Service: Adatfeltöltő szolgáltatás HTTP protokollon keresztül REST használatával. Továbbítja az adatokat a Kafka komponenshez a Hadoop rétegben.
- Data Query Interface: Adatok lekérését valósítja meg a Hadoop rétegből.
- MySQL (Management Database): A SensorHUB belső működését segítő adatbázis. A keretrendszer itt tárolja el a beregisztrált szolgáltatásokat és alkalmazásokat, menedzseli azok példány adatait, frissíti az kihasználtsági adatokat.
- Administration Service: A keretrendszer szolgáltatásaihoz és alkalmazásaihoz kapcsolódó webes adminisztrációs felület.
- Monitor Service: Az egyes szolgáltatásokkal kapcsolatos metrikák mérését, statisztikák számítását támogató szolgáltatás.
- Service Repository: A keretrendszer belső szolgáltatása. Nyilvántartja a futó mikro-szolgáltatások példányait, fogadja az API Gateway-en keresztül érkező kéréseket, és visszaadja az egyes szolgáltatások elérhetőségeit. Terheléselosztóval rendelkezik, replikált szolgáltatás.
- Application Repository: A keretrendszer belső szolgáltatása. Nyilvántartja a keretrendszert használó alkalmazásokat. Végpontot biztosít az alkalmazások beregisztrálásához, autentikálásához és a lekérdezésekhez. Terheléselosztóval rendelkezik, replikált szolgáltatás.
- Push Notification Service: A *Google Cloud Messaging* használatával Andoid értesítések küldését valósítja meg. [25]
- Geo Service (Proximity): A szolgáltatás segítségével a geofencing-et valósíthatjuk meg, vagyis körbekerítünk a térképen különböző területeket, és a szolgáltatás meg tudja mondani az éppen aktuális koordinátánk alapján, hogy benne vagyunk-e és ha igen, akkor melyik területben vagyunk benne.
- CV4SensorHUB: Gépi látás projektek menedzselését segítő szolgáltatás. Különböző transzformáció-láncok definiálhatók a vizuális adatok átalakítására és feldolgozására, melyek futtathatók, illetve kliens oldalról változtathatók.



19. ábra: Kérésfeldolgozás a SensorHUB keretrendszerben [19]

A kliens alkalmazás kérést indít a SensorHUB API felé (1). Az API Gateway-ek terheléselosztójának az IP címe a belépési pont. A terheléselosztó továbbítja a kérést az egyik API Gateway-nek. Az API Gateway az Application Repository terheléselosztóján keresztül eljut valamelyik konkrét Repository példányhoz, ami a Management Database adatbázist használva autentikálja az alkalmazást és ellenőrzi a jogosultságait (2.a).

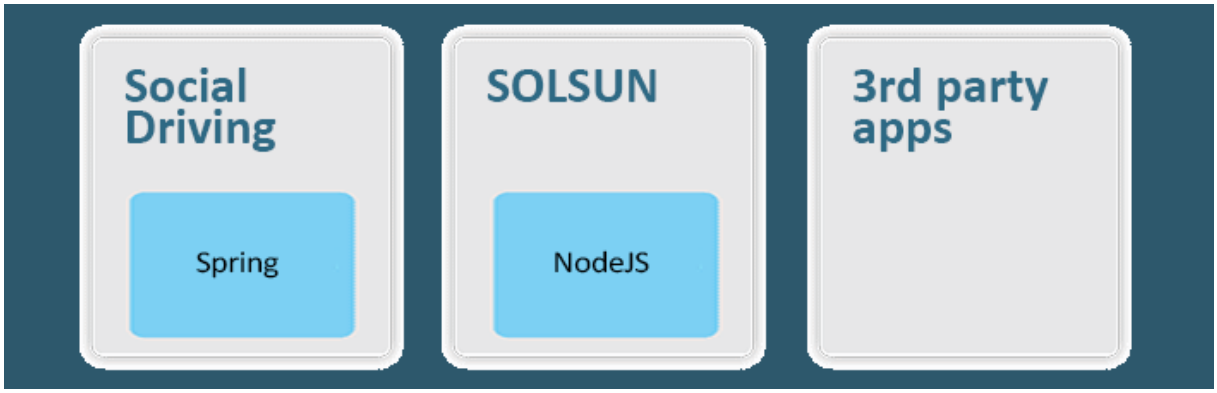
Ennek sikeressége után az API Gateway a Sevice Repository terheléselosztóján keresztül eljut valamelyik konkrét Repository-hoz, és megkeresi a kérésnek megfelelő szolgáltatást az adatbázisban (2.b).

Ezek után az API Gateway továbbítja a kérést az így kinyert szolgáltatás címére (3). Innen pedig a kérés tartalmán és a szolgáltatás típusán múlik a folytatás, például adatfeltöltés esetén a kéréssel érkező adat továbbításra kerül a Hadoop rétegbe (4).

A SensorHUB-ban minden kérés HTTP protokollon keresztül REST-es kérésekkel történik. Minden szolgáltatás külön IP címeken, illetve eltérő portokon fut, a Repository-k ezeket a címeket tárolják, és ezen információ kinyerésével fog a kérés a megfelelő szolgáltatáshoz eljutni. Tárolásra kerül továbbá az adott szolgáltatás állapota is, ez 10 másodpercenkénti állapotjelzéssel (heartbeat) történik a Service Repository felé. 30 másodperc kimaradás után a szolgáltatás törlődik az aktív komponensek listájáról. [19]

2.3. Eddigi felhasználások

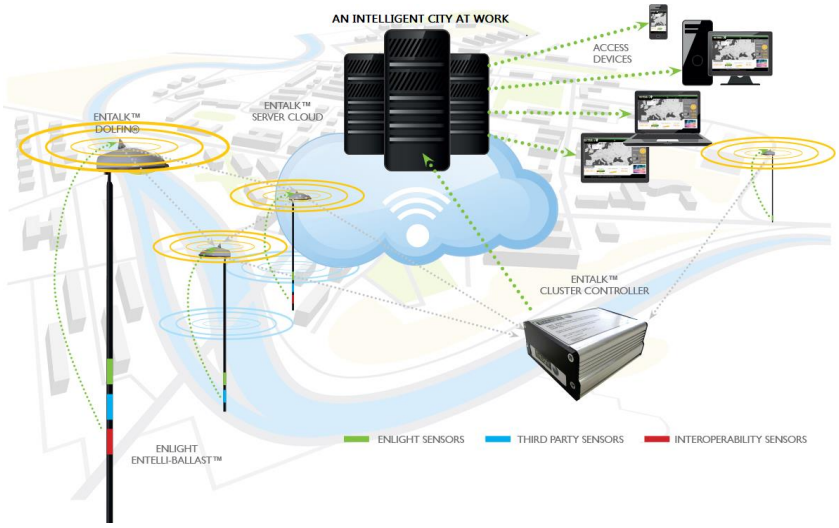
A SensorHUB rendszer már számos tanszéki, valamint további külsős projektek megvalósítása során felhasználásra került.



20. ábra: Eddigi SensorHUB felhasználások [19]

SOLSUN: Sustainable Outdoor Lighting and Sensory Urban Network

A 2015 tavaszán indult, a Climate-KIC európai innovációs partnerség által finanszírozott két éves SOLSUN projekt konzorciumát az angol EnLight cég vezeti, további tagok Budapest Főváros Önkormányzatán túl a British Telecom, a PANNON Pro Innovációs Kft., és a Budapesti Műszaki és Gazdaságtudományi Egyetem. [26][27]



21. ábra: EnLight okosváros koncepció [26]

A projekt célja megmutatni, hogyan lehet költséghatékonyan és fenntartható módon intelligens város infrastruktúrát kialakítani a városi közvilágítási rendszer hasznosításával. Az energiafogyasztás csökkentésének és a levegőszennyezés mérséklésének segítése szerepelnek a projekt elsődleges céljai között. [28]

A rendszer jelenleg Loddon kisvárosában van tesztelés alatt Angliában. A szenzorok adatokat gyűjtenek a levegőszennyezésről, a zajártalomról, a forgalomról és további klíma

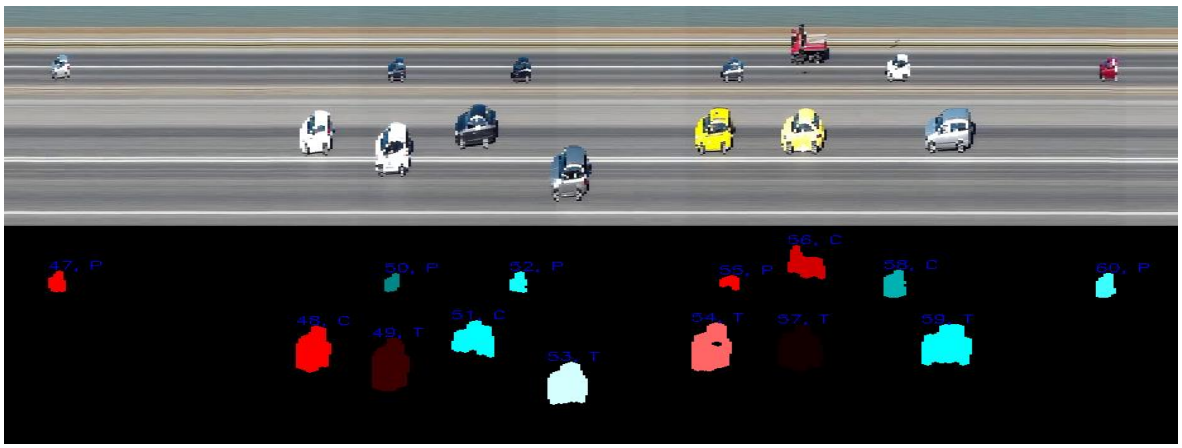
paraméterekről. Az összegyűjtött információkból levont következtetések segítik a városi forgalom szervezését, az energiahatékonyságot, valamint a városokban az üvegházhatású gázok kibocsátásának csökkentését. Vagyis fenntarthatóan üzemeltetni egy várost, a kényelem, ésszerűség, energiahatékonyság figyelembe vételével.

22. ábra: EnLight eszközök [26]

A megoldás révén, a gyűjtött szenzoradatokon alapulva, további szolgáltatások és alkalmazások telepíthetők és üzemeltethetők.

Lehetőség van okos útvonaltervezésre: leggyorsabb, legtisztább, legkivilágítottabb, legbiztonságosabb módok közül lehet választani, mind-mind az aktuális és a historikus szenzoradatokra alapozva.

Az egyes lámpákba beépített kamera segítségével forgalomszámlálás is megvalósul. Ez egy valósidejű automatizált számlálás, ahol kiemelt szerepe van az elhaladó entitások megkülönböztetésének: személygépjármű, busz, tehergépjármű, kerékpár, motorkerékpár, gyalogos. Ebből a célból vizualizációs és képfelismerő algoritmusok kerültek implementálásra.

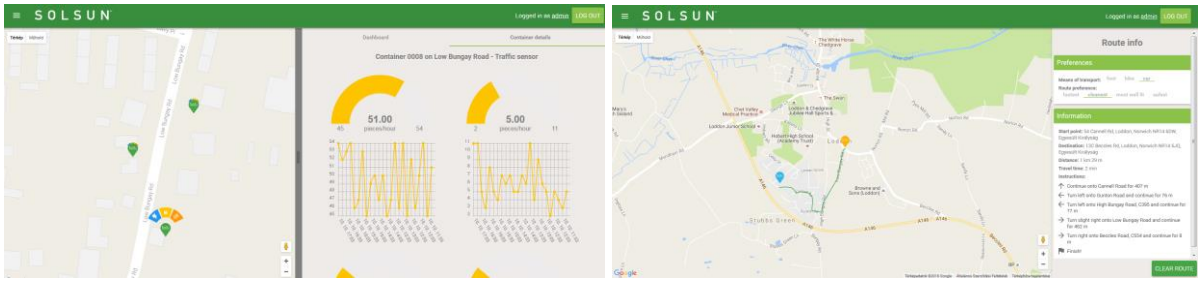


23. ábra: Forgalomszámláló kamera vetített képe

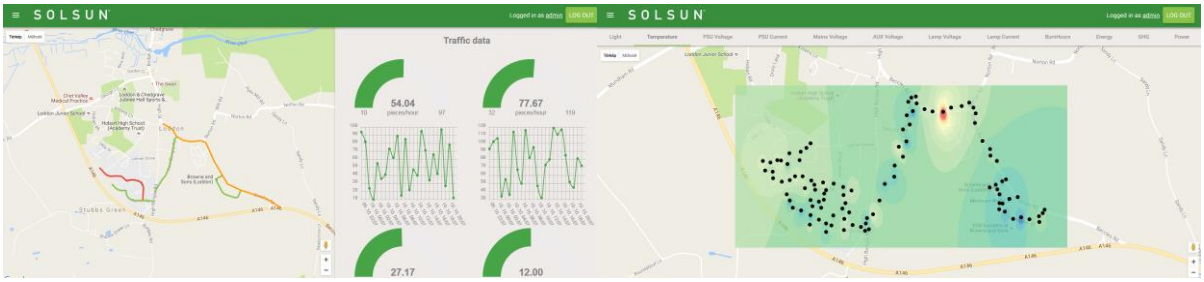
Minden lámpatest szenzor esetén elérhetőek a paraméterek diagramjai, melyek alapján további következtetések vonhatóak le és riportok készíthetőek. Ilyen paraméterek: feszültség, áram, teljesítmény, lámpatest állapota, forgalom, CO_2 érték, zaj, fényerő. Meghibásodás esetén is jelez a szenzor, ekkor a webes felületen hozzárendelhetőek a karbantartók, akik értesítést kapnak és elhárítják a felmerült hibát.

A SensorHUB *ProximityService*-ének segítségével itt is lehet területeket megjelölni, például áramszünet esetén, és a területre lépés esetén értesítést küldeni a felhasználóknak.

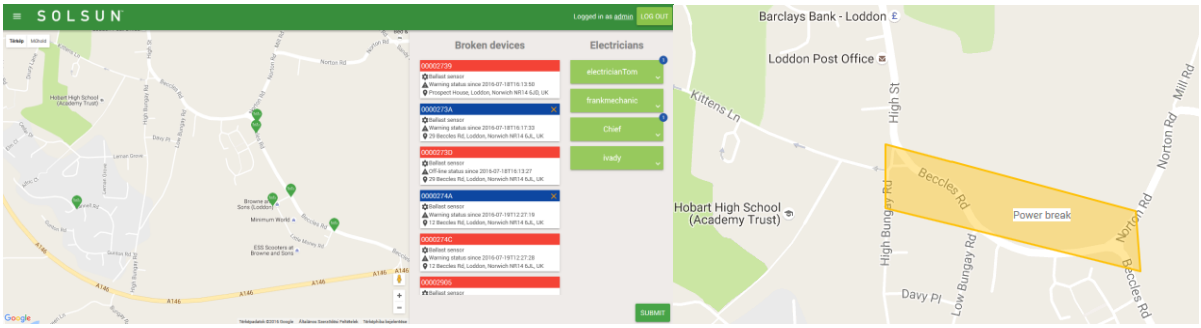
Budapest Főváros Önkormányzata 2017 tavaszára tervezi a budapesti tesztelés megkezdését.



24. ábra: SOLSUN webes felülete (paraméterek és útvonaltervező)



25. ábra: SOLSUN webes felülete (forgalomszámláló és hőtérkép)



26. ábra: SOLSUN webes felülete (hibabejelentő és geofencing)

A webes klienshez hasonlóan, bár korlátozottabb funkcionalitással az okostelefonos alkalmazásban is megtekinthetőek a szenzorok adatai, tervezhető útvonal, stb.

27. ábra: SOLSUN okostelefonos kliens képernyői

ERTI: Erdészeti Tudományos Intézet erdő-szenzor projektje

A projekt célja az erdőkben elhelyezett szenzorokból hőmérséklet, páratartalom, földnedvesség, szélerősség, csapadék, napsütéses órák száma, stb. paraméterek kinyerése, és valós idejű webes megjelenítése, valamint statisztikák készítése.

28. ábra: ERTI erdészeti felhasználás

CLAAS: Kombájn flottakövetés

A mezőgazdasági szakterületen is terjeszkednek az automatizálás és az IoT megoldások. Erre a szakterületre készül demo projekt egy kombájn flottakövető rendszer webes és mobil

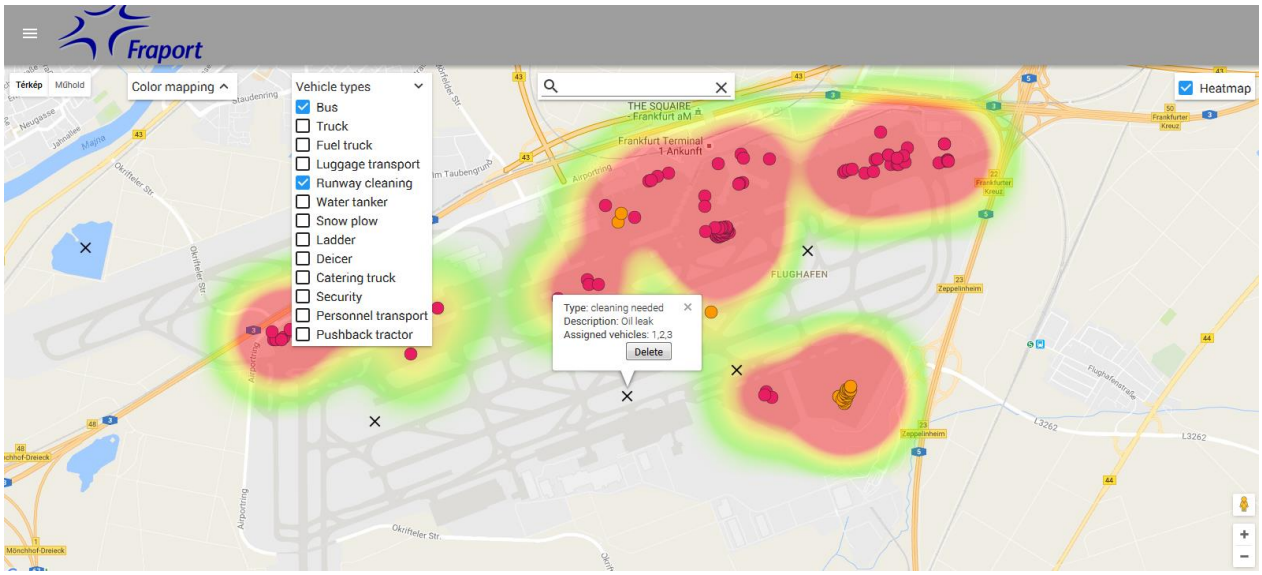
klienssel. A webes felületen Google Maps-es integrációval lehetőség van vetési statisztikák vagy szervízelési előrejelzések megtekintésére, sebességkorlátozások beállítására és még számos egyéb funkcióra. Az okostelefonos alkalmazással pedig értesülhetünk valamilyen változásról, vagy ha olyan területre léptünk be, ahol érvényes, egyedi sebességkorlátozás van érvényben.

29. ábra: CLAAS mezőgazdasági alkalmazás

Fraport: Frankfurti Repülőtér kiszolgáló egységek flottakövetése

Egy hasonló flottakövető demo a Frankfurti repülőtér kiszolgáló egységeinek flottakövetése. A Google Maps térképen lehetőség van a busz, teherautók, üzemanyagtöltők, poggyászszállítók, takarító járművek, víztöltők, catering járművek transzportbuszok stb. szűrésére, megjelenítésére. Valamint hőtérkép formájában is láthatóak az egységek.

További funkció problémák jelentése és kezelése (takarítás, baleset stb.) a térképen bejelölve és például takarítók hozzárendelése, automatizálása, valamint különféle előrejelzések megjelenítése.



30. ábra: Fraport repülőtéri felhasználás

3. A SensorHUB kiterjesztése

A SensorHUB kiterjesztése fejezetbe kerültek azok a felhasználások és alkalmazások, melyek fejlesztésébe munkámmal bekapcsolódtam, valamint részt veszek a kidolgozásukban, integrálásukban.

Elsőként a Social Driving alkalmazást mutatom be, amiben legelőször működtem közre, és elengedhetetlen az ismerete a ráépülő ObdCanCompare alkalmazás feladatának és céljának megértéséhez.

Végül egy teljesen új terület kerül ismertetésre, mely a gépi látás illetve adatfeldolgozás és a SensorHUB összekapcsolásával foglalkozik, a CV4SensorHUB projekt.

3.1. A Social Driving alkalmazás bemutatása

Egy rendkívül dinamikusan fejlődő szakterület a járműipar és az informatika integrációja, amelynek részeként az úgynevezett okosautó-alapú megoldások egy különösen aktív kutatási témakört jelent.

Ha kicsit absztraktabban gondolunk egy autóra, vagy bármilyen járműre, akkor könnyen beláthatjuk, hogy az is egy szenzorhalmaz. Annyiban tér el egy okostelefontól, hogy az adatai nem töltődnek fel az internetre, de ugyanúgy minden olvasható a műszerfalon, a kijelzőkön, mérhető az egyes rendszerek feszültsége, árama, teljesítménye.

A SensorHUB keretrendszer részeként készült VehicleICT alrendszer ezt a kapcsolatot teremti meg, vagyis az autónk mostantól okosautó lesz (még néhány további eszköz segítségével), és így az abból nyert adatok összemérhetők lesznek és bekerülnek a többi okoseszköz által gyűjtött adathalmaz mellé. [8]

Ilyen kapcsolódási lehetőségek például az OBD-II (*On-board Diagnostics II*) és a CAN bus (*Controller Area Network bus*) interfészek, amelyek segítségével a járművekből kinyerhetőek különféle információk, például sebesség, fordulatszám, valamint különböző hőmérséklet és nyomásértékek is. Ezeket felhasználva pedig elfogadhatóan közelíthető számos hasznos információ, pl. a jármű aktuális fogyasztása, illetve CO_2-kibocsájtása. [29]

Bár a CAN eszközökkel sokkal több és pontosabb adat nyerhető ki, sajnos az átlagfelhasználók számára a beszerelés nehézkes, az eszközök jármű specifikusak és drágák. Ezért mi az OBD port használata mellett döntöttünk, mely elérhető minden 2005 után gyártott gépjárműben. Az OBD-II port egy egyszerű és olcsó Bluetooth-os adapterrel tud

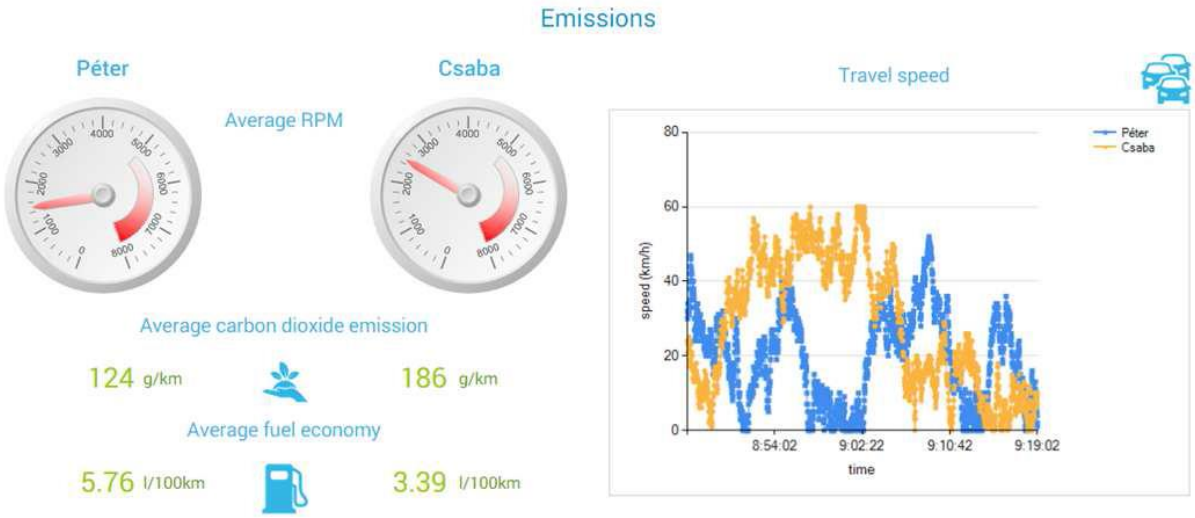
kommunikálni és adatfolyamot küldeni, melyet egy okostelefon segítségével már könnyen továbbítani lehet a szerveroldal irányába.

31. ábra: Bluetooth-os OBD adapter

A kinyert és a származtatott adatokra alapozva, majd kiegészítve további funkciókkal különböző okosautó-alapú megoldások készíthetők a jármű használója, de akár közösségek vagy ipari résztvevők (flották, biztosítók, szervizek, stb.) számára.

A Social Driving alkalmazással a járműből kinyert adatok segítségével össze lehet kapcsolni a rendszer felhasználóit, így összehasonlítható lesz vezetési stílusuk, és ha ugyanolyan autót vezetnek, akkor akár meg lehet nézni, milyen eltérések mutatkoznak a fogyasztásban a különböző vezetési stílusok eredményeként. [30]

Az adatok felhasználása számos üzleti lehetőséget is rejt. Például jutalmazni lehet azokat, akiknek autója adott idő alatt a legkevesebb káros anyagot bocsájtja ki; ők a töltőállomásokon akár kedvezményt is kaphatnának tankoláskor. Még arra is lesz lehetőségük, hogy utánanézzenek, az övékével megegyező autó mennyit fogyaszt, vagy milyen menettulajdonságokkal rendelkezik.



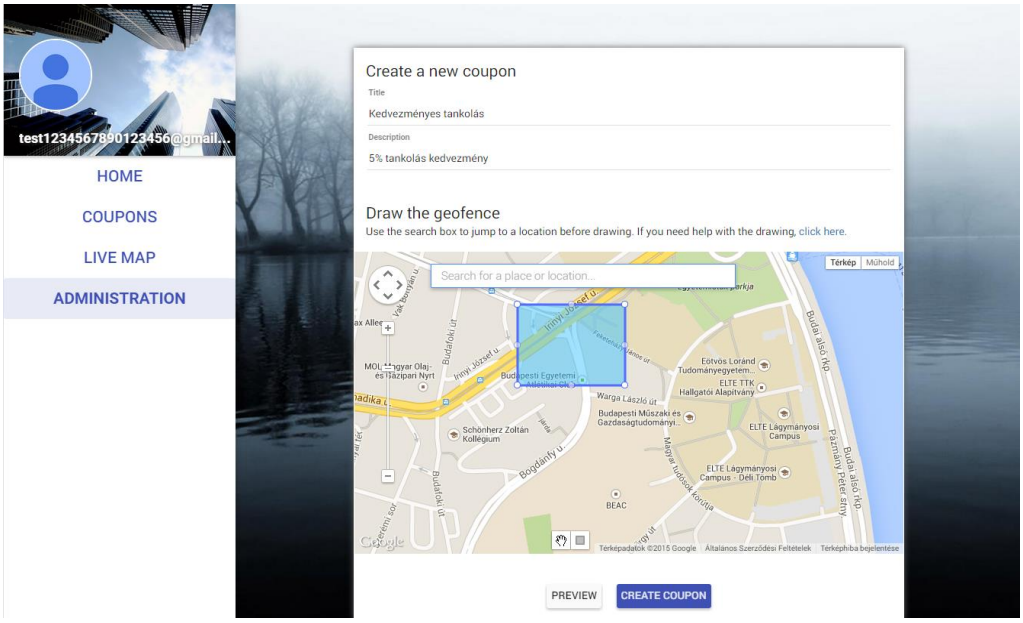
32. ábra: Összehasonlító webes riport a Social Driving adatai alapján

A hibakódok kiolvasásával és valós idejű értesítéssel, a szervizhálózatok remekül használhatnák ezeket az információkat. Jelentős hiba esetén azonnal értesíteni lehet majd az autó vezetőjét, sőt akár online motordiagnosztikát is elvégezhetnek, hiszen rendelkeznek a szükséges adatokkal, így már felkészültén fogadhatnák az autóst a szervizben, és akár személyre szabott ajánlattal is megkereshetik a tulajdonosokat. Ha pedig összekapcsoljuk ezt a rendszert az okos városával, akkor valós időben jelezhető az autósnak, ha már fölösleges a gázba taposnia, hiszen a kereszteződési lámpa mindjárt pirosra fog váltani. [31]

Lehetőség van a fontosabb paraméterek lebegőablakba történő kiszervezésére, így az alkalmazás a háttérben működik majd, nem zavarhatja vezetés közben a sofőrt például a navigálásban, de pár fontos adat mégis látható marad.

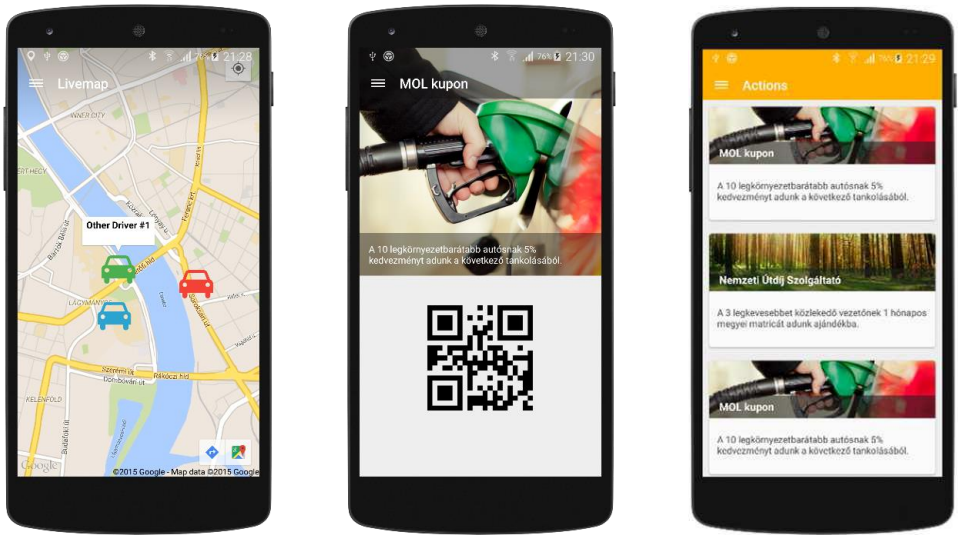
A kinyert adatok rögtön feltöltésre kerülnek a Hadoop klaszterbe, ahol a streaming szolgáltatás segítségével, az új adatok felhasználása mellett azonnal visszaküldi a telefonra az éppen aktuális összesítéseket, statisztikai számításokat, és így valós időben meg tudnak jelenni a historikus adatok is az alkalmazás képernyőjén. [32] Fontos funkció, hogy az adatok megtekintéséhez és összehasonlításához, nincs feltétlenül szükség az OBD-s adapterhez, bár ekkor csak a többiek eredménye lesz látható.

A Social Driving alkalmazás webes felületén a szerződött partnerek kuponokat hozhatnak létre, mely kuponokat valamilyen érvényességi területhez rendelhetik. Itt is a SensorHUB-os *ProximityService* kerül hasznosításra. Ha az autós bekerül a kupon érvényességi területére, akkor a telefonjára kap egy értesítést, és számára felhasználhatóvá válik a kupon által nyújtott kedvezmény.



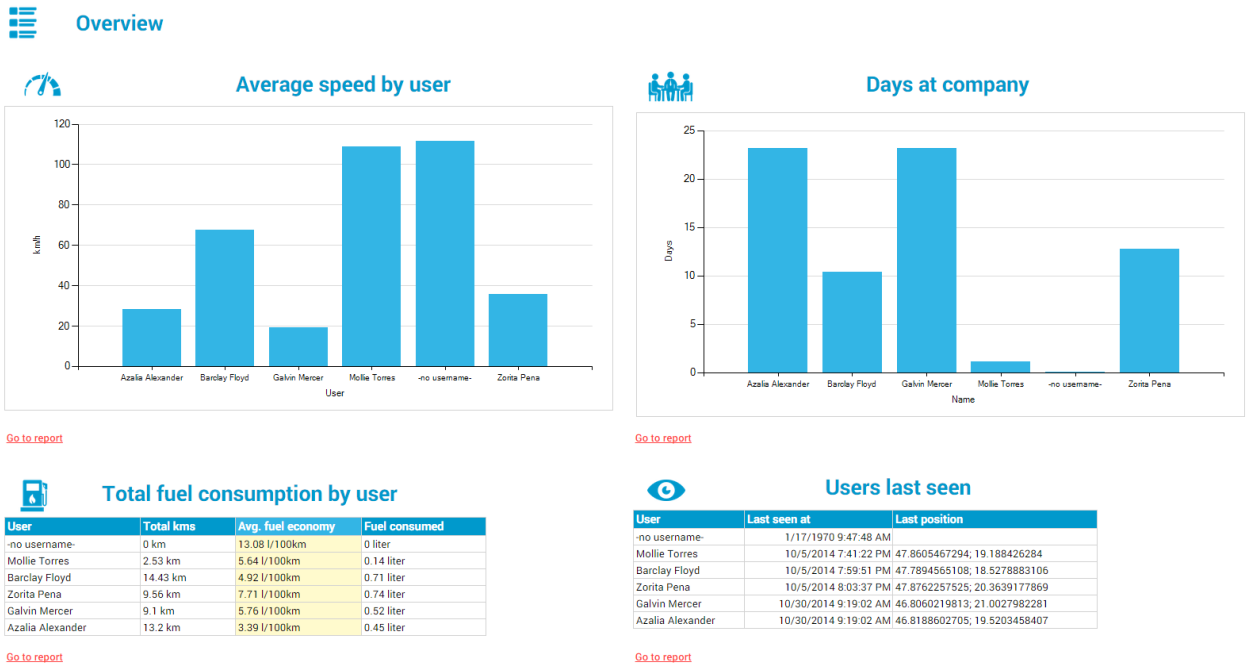
33. ábra: Új kupon létrehozása

Például valamilyen benzinkutas vállalat által létrehozott kupon, a QR-kód beolvasása után valamekkora kedvezményt tud érvényesíteni az üzemanyag árában.



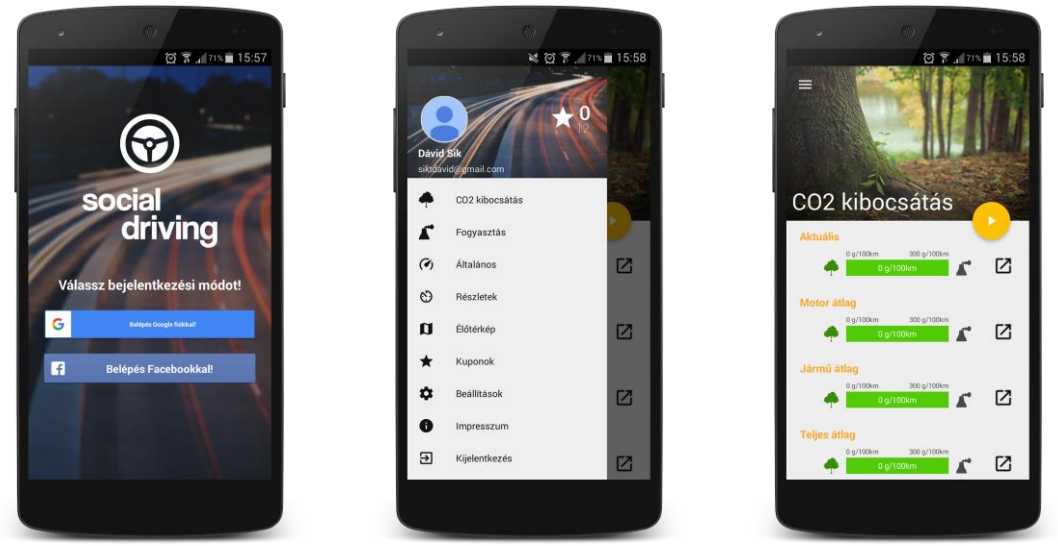
34. ábra: Terület megközelítése élőtérképen és a kupon megjelenése

A Social Driving felhasználók a webes felületen is nyomon követhetik az értékeiket, de ez az okostelefonos alkalmazásban jobban ki van dolgozva.



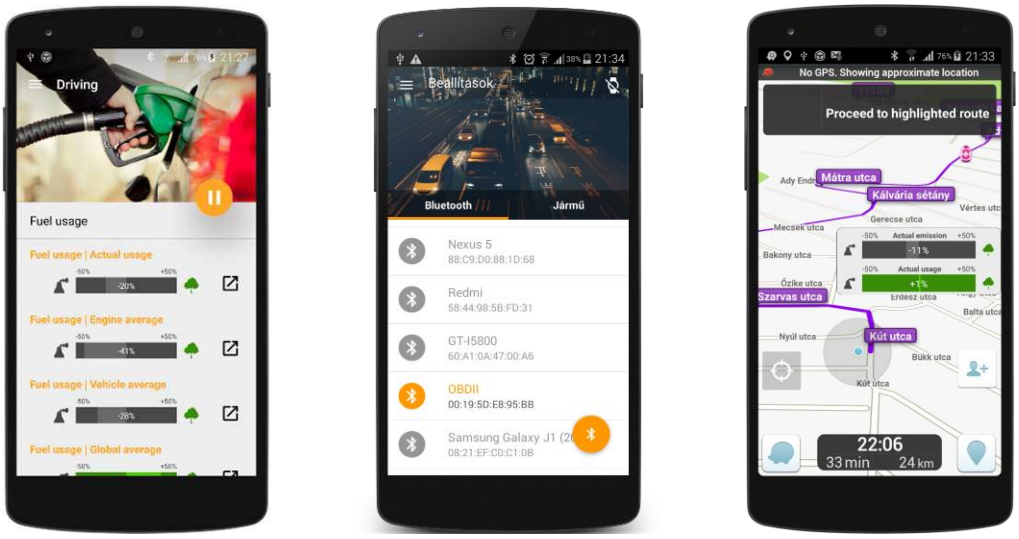
35. ábra: Paraméterek a webes felületen

Itt összehasonlíthatjuk értékeinket anonim módon a többi felhasználóval, illetve céges környezetben csoportok hozhatóak létre, melyek paramétereit szintén diagramok segítségével lehet összesíteni.



36. ábra: LoginActivity, NavigationDrawer, EmissionFragment képernyőképek

A Social Driving alkalmazást elindítva a *LoginActivity* fogad bennünket, ahol a Facebook-os vagy Google+-os bejelentkezési lehetőségek közül lehet választani. A sikeres bejelentkezés után a képernyő bal oldaláról a *NavigationDrawer*-t előhívva láthatjuk a különböző opciókat: CO_2 kibocsátás, Fogyasztás, Általános, Részletek, Élőtérkép, Kuponok, Beállítások, Impresszum és Kijelentkezés. Az első felület a CO_2 kibocsátás (*EmissionFragment*), négy diagrammal: Aktuális, Motor átlag, Jármű átlag és Teljes átlag.



37. ábra: FuelFragment, BluetoothFragment, Lebegőablakos nézet képernyőképek

A Fogyasztás (*FuelFragment*) nézeten is hasonló módon vannak csoportosítva a diagramok. A Beállítások nézeten lehet kiválasztani az OBD-s Bluetooth eszközt, valamint felvenni a járműveket, és beállítani a jármű tulajdonságait. A CO_2 kibocsátás és Fogyasztás nézeteken a diagram mellett látható gombbal, az aktuális mutató kitehető lebegőablakba, ami

más alkalmazás használata esetén is a képernyő legtetején látszódik, kevés helyet foglalva (például navigáció közben).

38. ábra: Social Driving Okosóra támogatás

A Social Driving támogatja az okosórákat is, mellyel összekapcsolva, az okosóra képernyőjén is megjeleníthetőek paraméterek, illetve az értesítések.

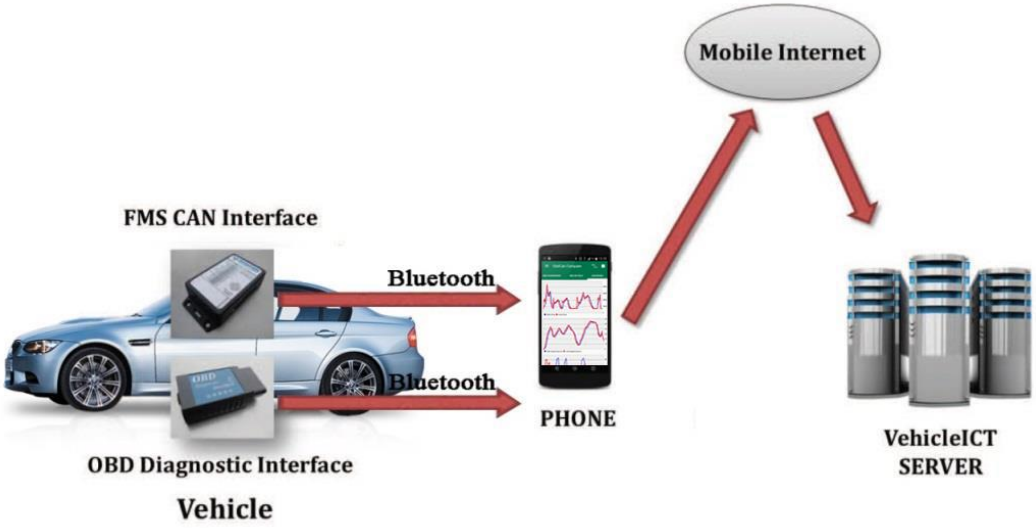
3.2. Az ObdCanCompare alkalmazás bemutatása

Az ObdCanCompare alkalmazás elkészítését abból a célból kaptam, hogy láthatóvá váljon, hogy a Social Driving alkalmazás számára a számottevő paraméterek esetén elegendő az OBD interfész használata, nem tud annyival többet a CAN bus, hogy érdemes lenne komolyabban foglalkozni az átállással.

Az összehasonlítás során az OBD-II interfész az előző fejezetben ismertetett adapterrel, CAN bus pedig az Inventure Kft. által fejlesztett adapterrel Bluetooth-on vagy USB-n keresztül tud kommunikálni, adatfolyamot küldeni. Ezeket egy okostelefon segítségével már könnyen továbbítani lehet a szerveroldal irányába. [33]

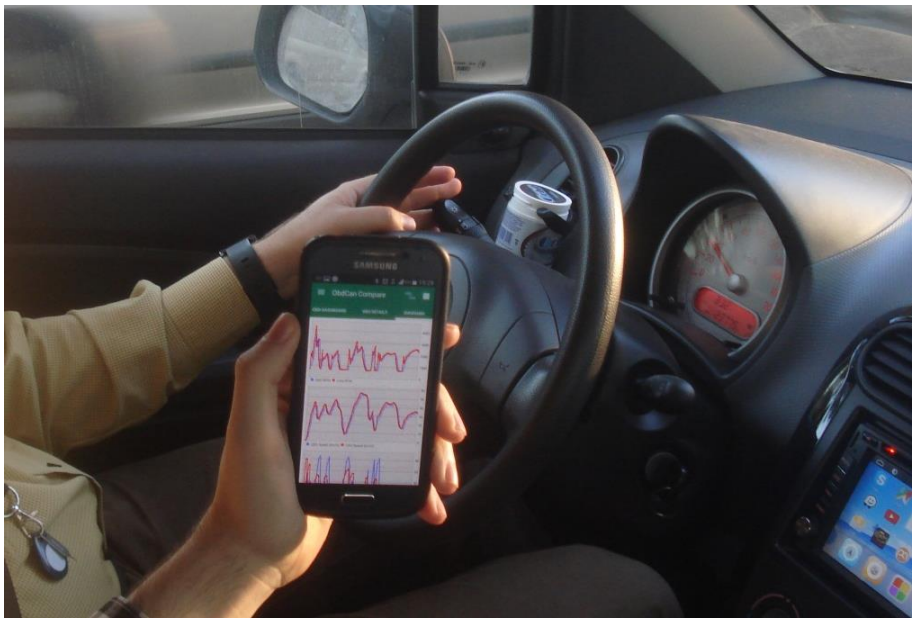
39. ábra: FMS Gateway CAN bus adapter

Összefoglalva, tehát az OBD-II és CAN bus adatok kinyerésére, SensorHUB felé történő továbbítására és az okostelefonon történő megjelenítésére és összehasonlítására készült el az ObdCanCompare Android alkalmazás.



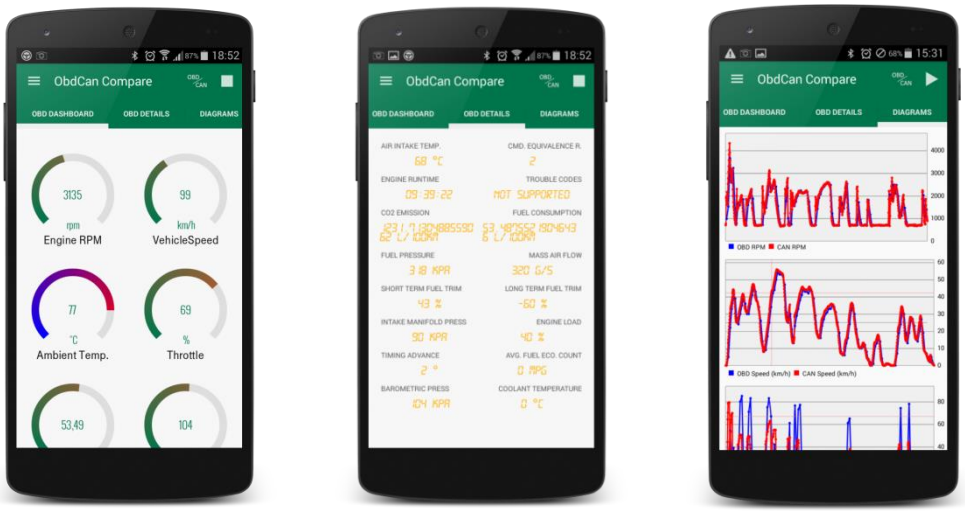
40. ábra: Adatgyűjtés az ObdCanCompare alkalmazással

A párhuzamos Bluetooth adatfolyamok fogadásához szükség volt néhány apró változtatásra a keretrendszerben, ezután következett az alkalmazás implementálása és tesztelése.



41. ábra: Az ObdCanCompare tesztelése

A SensorHUB koncepciót kihasználva és az összegyűjtött adatok összesítése céljából egy riportkészítő szolgáltatás is hozzákapcsolásra került az architektúrához. Az adatok integritásához szükség lehet az adatok tisztítására, szűrésére és paraméterezésére, mely elvégezhető a webes Big Data felhasználói felületről vagy alkalmazástól függően akár a riportkészítő alkalmazásból is.



42. ábra: Az ObdCanCompare képernyői (Dashboard, Details, Charts)

Az adatelemzést és vizualizációt a Java alapú Pentaho üzleti intelligencia rendszer segítségével végeztem el. A Pentaho egy nyílt forráskódú, Java alapú, Big Data rendszerekkel összeköthető programcsomag. Létezik Community és Enterprise változata is. [34] Az adathalmaz feldolgozásához és elemzéséhez a Pentaho Data Integration és Pentaho Report Designer alkalmazásokat használtam.

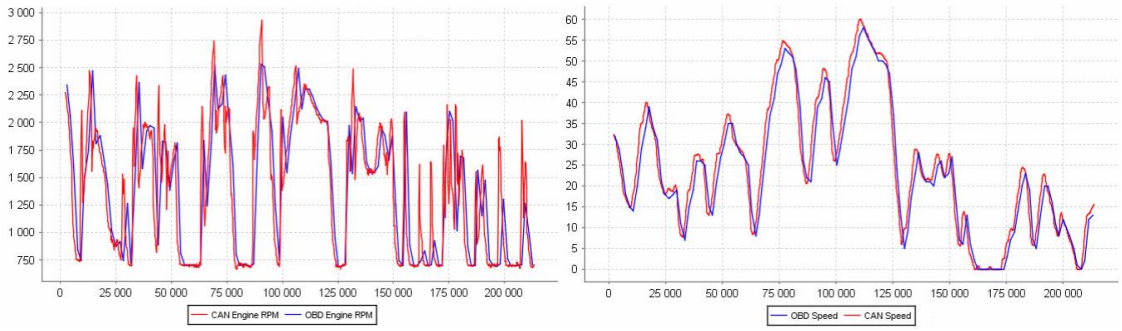
A Data Integration különböző forrásokból származó adatokat tud egységesen kezelni. ETL (*Extract-Transform-Load*) transzformációkat állíthatunk össze benne a Spoon eszköz grafikus felületén. Ez a rész az adatfolyam manipulálásáról szól. Másik lehetőség job-ok összerakása, ami a magas szintű vezérlést takarja, például e-mail küldése, ha elkészült a transzformáció, vagy fájl átvitel. [35]

43. ábra: Pentaho Data Integration adatfolyam

Az általam összeállított adatfolyam kezdete egy HiveQL szkript [36], ami táblakészítő lekérdezéseket futtat a szerveroldali Hadoop klaszteren. Ez a JSON formátumú fájlokból készít táblákat. Majd az így elkészült táblákból lekérdezi a szükséges adatokat, végül ebből Excel táblát készít. A folyamat paraméterezhető: megadható, hogy melyik mérési időtartam kerüljön bele a táblákba.

Az így elkészült táblát a Report Designer-ben lehet felhasználni különböző táblázatok, grafikonok, diagramok és további megjelenítők adatforrásaként.

Itt is különböző lekérdezéseket lehet összeállítani, ez történhet különböző lekérdező nyelvekkel. Az, hogy milyen lekérdező nyelvet használhatunk, attól függ, hogy milyen adatbázishoz kapcsolódunk (Hadoop Hive, Hive 2, Impala, MSSQL, MySQL, Oracle stb.), ezek mind SQL-szerű lekérdezéseket támogatnak, kisebb-nagyobb eltérésekkel. A Data Integration-ben Hadoop Hive 2-t, a Report Designer-ben pedig Impala-t használtam a kapcsolódás és a lekérdezések eszközeként. [35]

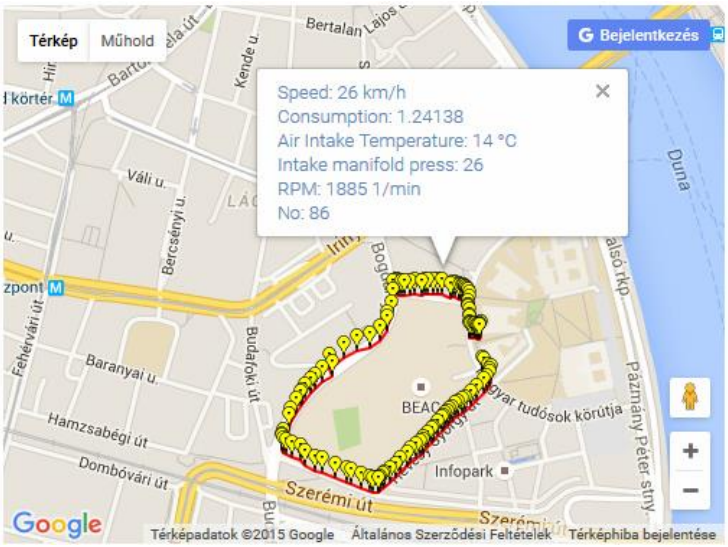


44. ábra: Fordulatszám és sebesség összehasonlítása a Pentaho reportban

Végül a lekérdezés eredményének oszlopait kell összerendelni a megjelenítő elem adatsoraival. Továbbá a riportok itt is paraméterezhetőek, és programkódokat is hozzájuk lehet rendelni.

A Report Designer kimenetei különböző formátumúak lehetnek: PDF, HTML, Excel, RTF, Text, CSV. Ezek a struktúrák feltölthetőek az üzleti intelligencia szerverre is, ami generálni tudja a megfelelő formátumú riportot.

A Pentaho csomag része még a Business Analytics Platform, ami egy Apache Tomcat Web Server. Erre publikálhatóak és ezen futtathatóak a riportok, de a Data Integration és Report Designer nélküle is használható.



45. ábra: Interaktív térkép megjelenítés a Pentaho reportban

A térképes megjelenítés nincs beépítve a Report Designer-be, ezt egy a riportba belekódolt JavaScript kódrészlet hozza létre, mely a Google Maps API adatformátumát rendeli össze a lekérdezés eredményének soraival.

Ezeknél a lekérdezéseket szükség volt arra, hogy minden adathoz szerepeljen egy diszkriminátor oszlop, aminek a segítségével elkülöníthetőek az OBD-s és CAN-es adatok. Ezt a megfelelő SQL utasításokkal könnyen létre lehetett hozni.

A két megjelenítési forma tehát konzisztens. Az ObdCanCompare-es az azonnali megjelenítéshez hasznos, míg a Pentaho-s pedig a későbbi elemzésekhez szolgál célt.

A Hadoop klaszter online HUE felülete is számos adatvizualizációs lehetőséget biztosít, de csak az egyszerűbbek közül. [37] Az adatfolyam összeállítására itt az Oozie eszközt lehet használni, amiben folyamatábraként rakhatjuk össze az ETL folyamatot. Ezt lehet a szerveren időzíteni is, például, hogy milyen időközönként fusson le. [38]

46. ábra: HUE Map a sebesség kijelzésével

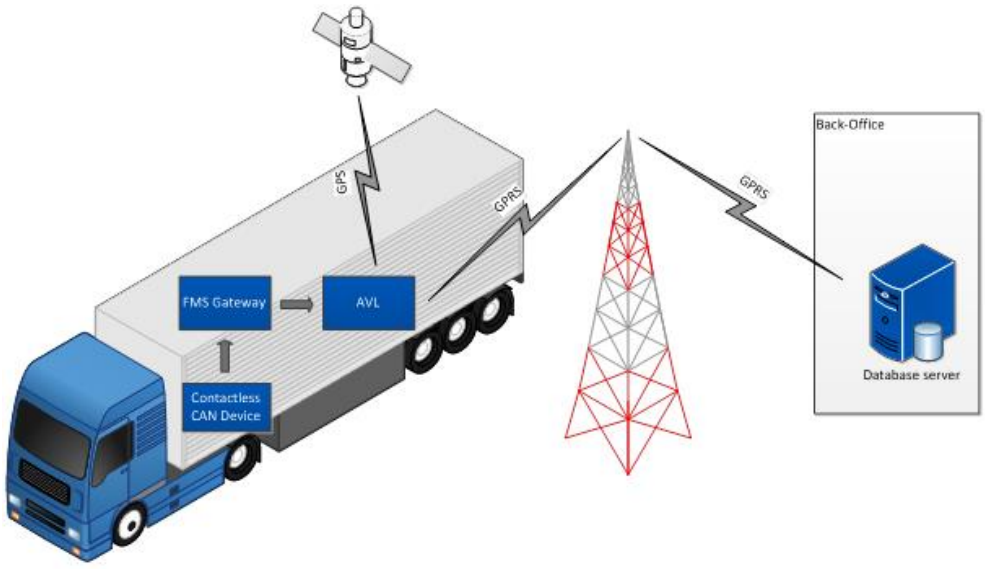
A térképes megjelenítést az online felületen sokkal egyszerűbb összerakni, nem kell kódot írni, csak ki kell választani a földrajzi hosszúságnak és szélességnek megfelelő oszlopokat. Ehhez hozzárendelhető, hogy a térképen rákattintva mit írjon ki az adott ponthoz, de a koordinátákhoz csak egy paramétert lehet kiíratni. Végül az eredmény megjelenik egy OpenStreetMap alapú térképen.

Oszlop-, vonal- és tortadiagramokat is össze lehet rakni a felület segítségével, azonban itt az a korlátozás, hogy csak egy adatsor jeleníthető meg valamely másiknak a függvényében, tehát összehasonlításra nem lehet használni.

Bizonyos adatkorlátig még az Excel is kiválóan használható adatvizualizációra. A példában most kevés adatról van szó, így felhasználhatjuk a transzformáció során létrehozott Excel táblát.

Az elkészült riportok és diagramok alapján látható, hogy az OBD-II mérés és a CAN bus mérés eredményei számos jellemzőben szinte megegyeznek egymással. Ilyen például a fordulatszám és a sebesség értéke. Itt csak egy apró késleltetés látható az OBD-s adatokban a CAN bus adataihoz képest, amit könnyen lehet korrigálni. Ugyanakkor más értékek már nagyobb eltéréseket mutattak, főleg a származtatott (aggregált) paraméterek. Itt az eltérés oka, hogy különböző összetett formulák léteznek például a fogyasztás vagy a CO_2-kibocsájtás meghatározására, amikből eltérő eredmények születhetnek az OBD-s és CAN bus-os mérések összehasonlításában. Az is eltérést jelenthet, hogy bár egyes paraméterek ugyanarra az adatra vonatkoznak, a mért értékek azokban mást jelentenek a két interfész esetén.

Ezek alapján elmondható, hogy az alapvető okosautó-alapú megoldásokhoz elegendő az OBD-II port használata, például a Social Driving alkalmazás esetében, de fontos, hogy a származtatott értékek formulái mindenütt egységesen kerüljenek felhasználásra. Amennyiben pedig professzionális megoldásra van szükségünk, például flottakövetésre, és további összetettebb adatok kinyerésére, akkor a CAN bus-alapú megoldás a jó választás.

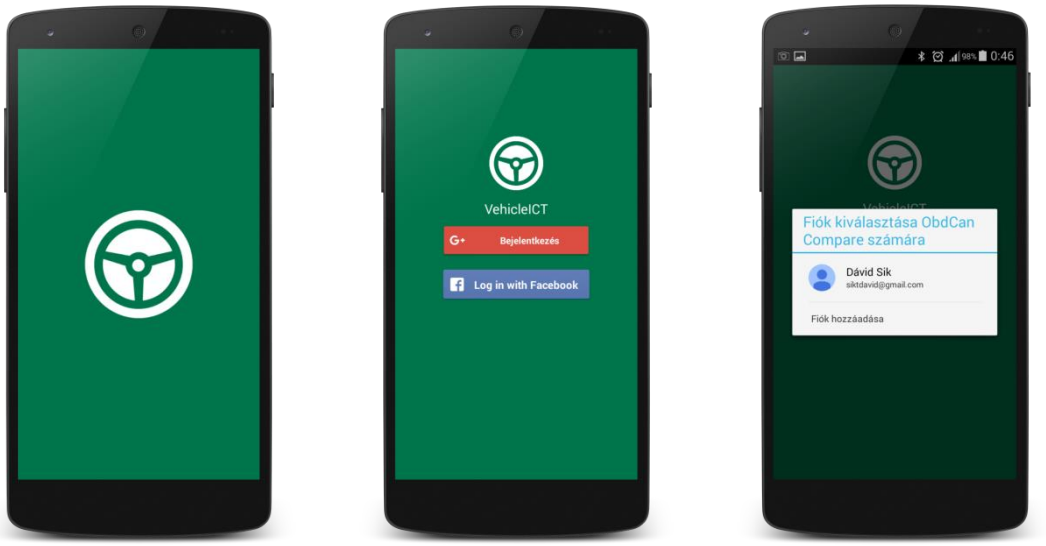


47. ábra: Online flottakezelő rendszer [33]

A jövőben az ObdCanCompare alkalmazás és a hozzátartozó riportok pedig tovább bővíthetőek, amennyiben új paraméterek kijelzésére és összehasonlítására lesz szükség. Csak előtte persze az új adatok beolvasásának logikáját implementálni kell a SensorHUB keretrendszerben is. További feladat még az adatok átmeneti tárolása a telefonon.

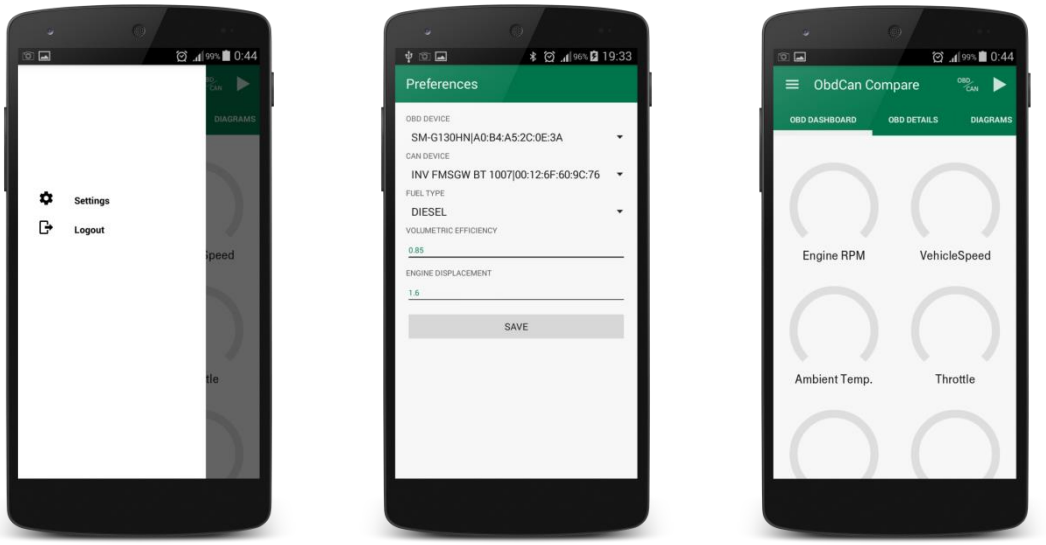
Már tervben is van a fogyasztás és a gyorsulás megvalósítása az Inventure Kft. részéről, amit először nekik is el kell készíteni az FMS Gateway eszközükbe egy-egy új üzenetként, OBD-II részről a fogyasztás már kész, a gyorsulás pedig a sebesség deriválása helyett megoldható az okostelefon beépített gyorsulásmérőjének a segítségével.

Az alkalmazás Android okostelefonra telepítve, Bluetooth bekapcsolása mellett működik. Ha a telefon internetre is tud csatlakozni, akkor az adatok a szerver oldalra is feltöltődnek.



48. ábra: SplashActivity, LoginActivity képernyőképek

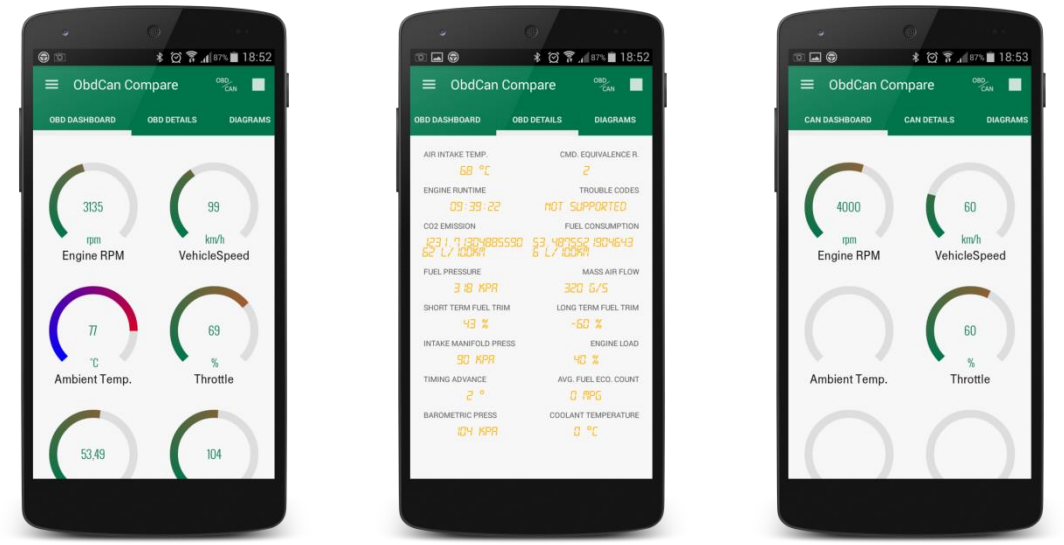
Az alkalmazás ikonját kiválasztva, először a *SplashActivity* üdvözlőképernyője jön be. Kis idő után pedig a *LoginActivity* következik, ahol a Facebook-os vagy Google+-os bejelentkezési lehetőségek közül lehet választani.



49. ábra: NavigationDrawer, PreferencesActivity, MainActivity képernyőképek

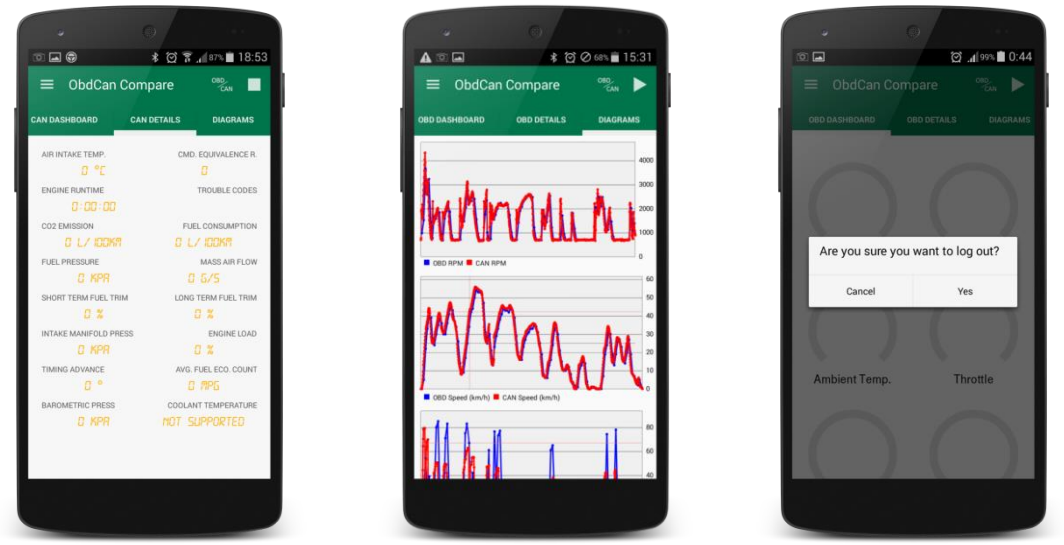
A *MainActivity*-ben a bal oldalról a *NavigationDrawer*-t előhívva a Settings opciót kiválasztva jutunk el a *PreferencesActivity*-re, ahol a mérések tulajdonságait állíthatjuk be (a

Bluetooth engedélyezése után): a két Bluetooth-os eszközt, az üzemanyag típusát, a motor térfogatát és hatásfokát. A *Save* gombra, vagy a vissza gombra nyomva visszajutunk a *MainActivity*-re.



50. ábra: ObdDashFragment, ObdDataFragment, CanDashFragment képernyőképek

A mérést a jobb felső sarokban lévő *Play* gombbal indíthatjuk el. Ekkor mind a Dashboard, mind a Details, mind a Diagrams füleken olvashatóak az adatok. Az OBD és CAN mérés adatainak megjelenítés között az *OBD/CAN* gombbal lehet váltani, ekkor a fülek felirata is megváltozik.



51. ábra: CanDataFragment, CompareFragment, LogoutDialogFragment képernyőképek

A mérést a *Stop* gombbal lehet beállítani. A kijelentkezéshez *NavigationDrawer*-t előhívva a *Logout* lehetőséget kell kiválasztani és a *Yes*-re nyomni.

A mérés elindítása után (ha volt internet csatlakozás) már elérhetőek az adatok a szerver oldalon, szűrhetőek, tisztíthatóak az adatok és készíthetőek a kimutatások.

3.3. A CV4SensorHUB mikroszolgáltatás bemutatása

A CV4SensorHUB téma célja a SensorHUB számára egy képfeldolgozási kiegészítő keretrendszer és felhasználói felület kialakítása.

A CV4SensorHUB keretrendszer az alábbi fő részekből áll [39]:

- Közös, portábilis, JSON alapú adatreprezentáció képfeldolgozási módszerekhez
- .NET WPF (és később párhuzamosan HTML5) alatt fejlesztett felhasználói felület
- Kliens vagy szerver oldalon is futtatható képfeldolgozási műveletek.

A projekt célja annak lehetővé tétele, hogy képfeldolgozási algoritmusok egy raszteres képen és címkézett polygonokon alapuló adatreprezentációval szerver és kliens oldalon is futtathatók legyenek. Amennyiben például a szerver oldali tömeges feldolgozásban van egy lépés, amiben felhasználói segítséget (vagy csak ellenőrzést és jóváhagyást) kell kérni, a kérést átküldve kliens oldalra a felhasználó megnézheti az eredményeket, esetleg belemódosíthat, további segítséget adhat a félautomata módszereknek, majd visszaküldheti az adatmodellt a szerver oldalra a további feldolgozáshoz. Így nem csak teljesen automata megoldásokat lehet futtatni, hanem olyanokat is, ahol a felhasználótól is szükség van segítségre. Ez azért fontos, mert sok alkalmazás vagy nagyon nehéz és így a teljesen automatikus feldolgozás túl sokat hibázik, vagy nem engedhetőek meg a hibák, így a végső szó mindig a felhasználóé, viszont a rendszer mindent megtesz azért, hogy a felhasználónak minél kevesebb munkája legyen. Ezeket a funkciókat valósítja meg a SensorHUB CV4SensorHUB mikroszolgáltatása.

A témakörben jelenleg 3 alkalmazási irány fejlesztése indult el, melyek közösen használják majd a közös cv4s keretrendszer egyre bővülő szolgáltatásait [39]:

- GrainAutLine: a korábban GrainAutLine néven függetlenül futó, márvány vékonycsiszolatok elemzését szolgáló rendszer új generációja.
- ChemoTracker: fehérvértestek mozgását követő alkalmazás immunológiai kutatások támogatására.

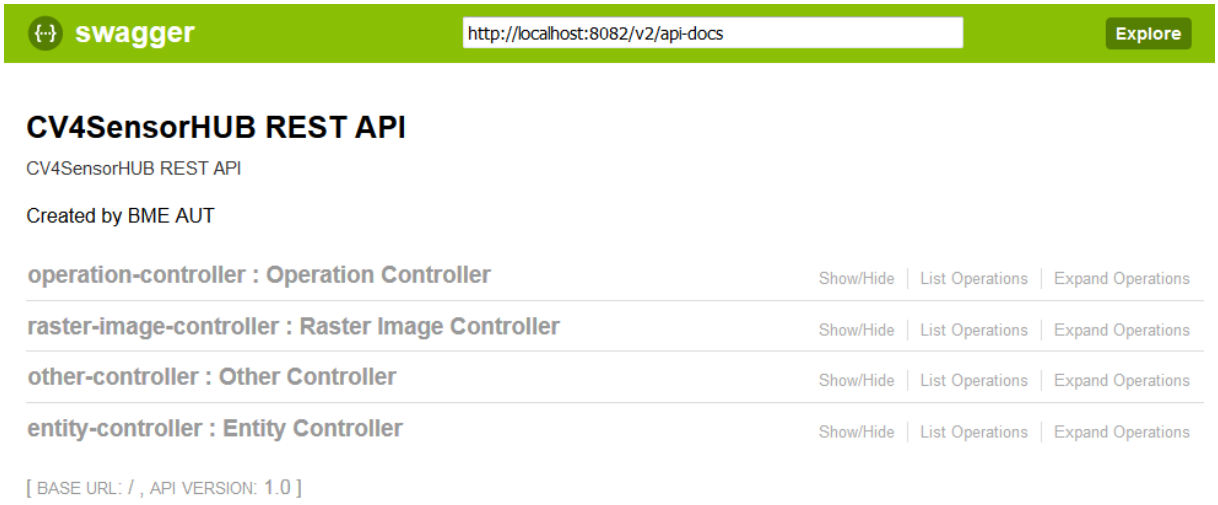
- Játékok: a cv4s keretrendszer bizonyos játékok elkészítésére is alkalmas. Ezek elkészítése során várhatóan számos olyan funkcióval bővül a keretrendszer, amit a többi alkalmazás is fel tud majd használni.

Az irányok fő célja a SensorHUB információ forrásainak kiterjesztése vizuális adatforrások felé (például állókép, mozgókép).

Első körben ez úgy néz ki, hogy a SensorHUB-nak (időben változó) polygonokat kell tudnia tárolnia és összehasonlítania. Ezeken a polygonokon lehetnek címkék, valamint lehetnek georeferáltak is, vagyis pl. minden pont GPS koordinátája is ismert.

A SensorHUB CV4SensorHUB-os microservice-ét én készítettem el Spring MVC szerveroldali keretrendszer [40] és a JSON dokumentum tárolás alapú MongoDB [41] perzisztens NoSQL adatbázis segítségével. [42]

A SpringMVC-s szerver oldalon különböző REST alapú interfészeket definiáltam (GET, POST és PUT HTTP method-okkal[17]), ezekkel hívhatóak meg a különböző szolgáltatások. Az adatok mentése és kiolvasása a MongoDB-ből történik. Mind a Tomcat-en futó szerver [43], mind az adatbázis folyamatok jól követhetőek a szerver debug felületen, az adatbázisban bekövetkező változások pedig a MongoDB Compass vizualizálós programmal.



52. ábra: CV4SensorHUB REST API Swagger felülete

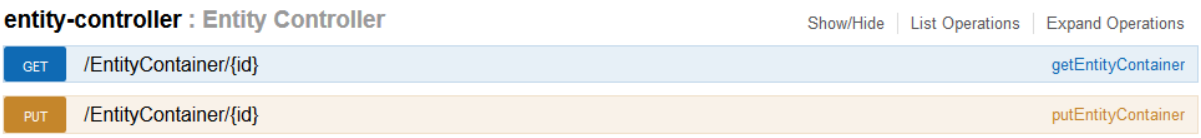
A szerver URL böngészőbe történő beírása után a fejlesztők számára egyre szélesebb körben ismert Swagger felület fogad [44], melynek segítségével nagyon könnyen ki lehet próbálni mindenféle REST-es hívást. Tájékoztat a szükséges Query paraméterekről, valamint a Body-ba összeállítandó JSON modellt is megadja, és interaktív módon is lehetőség van az adatok összeállítására.

A CV4SensorHUB szerver Swagger leírója mindig dinamikusan generálódik az üzleti logikát tartalmazó Spring MVC szerver Java kódja alapján. Ezáltal a szerveren történt változtatások rögtön, automatikusan megjelennek a Swagger kezelőfelületén is.

Az EntityContainer modell felépítése:

```
EntityContainer {
   entities (Array[Entity])}
Entity {
   __type (string),
   aux (Array[Aux]),
   points (Array[Point]),
   t (integer),
   tags (Array[Tag])}
Aux {
   Key (string),
   Value (Value)}
Point {
   _x (number),
   _y (number)}
Tag {
   Key (string),
   Value (integer)}
Value {
   __type (string),
   t (integer),
   values (Array[integer])}
```

Egy EntityContainer-re konkrét JSON példa a függelékben látható.

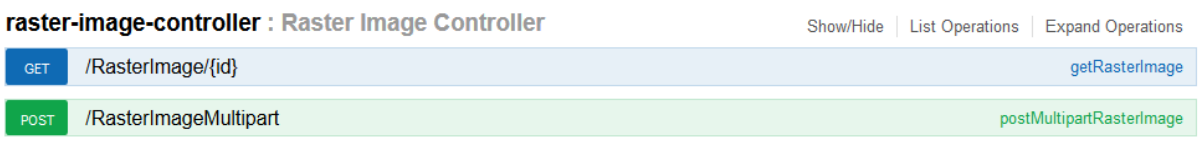


53. ábra: EntityContainer metódusok

Az entity-container csoportban van lehetőség az EntityContainer adatbázisba mentésére (frissítésére), illetve adatbázisból történő kiolvasására.

A `GET /EntityContainer/{id}` visszaadja az adott id-jű EntityContainer-t JSON formátumban az adatbázisból.

A `PUT /EntityContainer/{id}` létrehozza vagy felülírja az adott id-jú EntityContainer-t az adatbázisban.



54. ábra: RasterImage metódusok

A raster-image-controller csoportban van lehetőség képek adatbázisban mentésére (frissítésére), illetve adatbázisból történő kiolvasására. Ezen a képek a poligonok és entitások hátttereként szolgálhat, és ezeken is ki lehet jelölni különféle területeket.

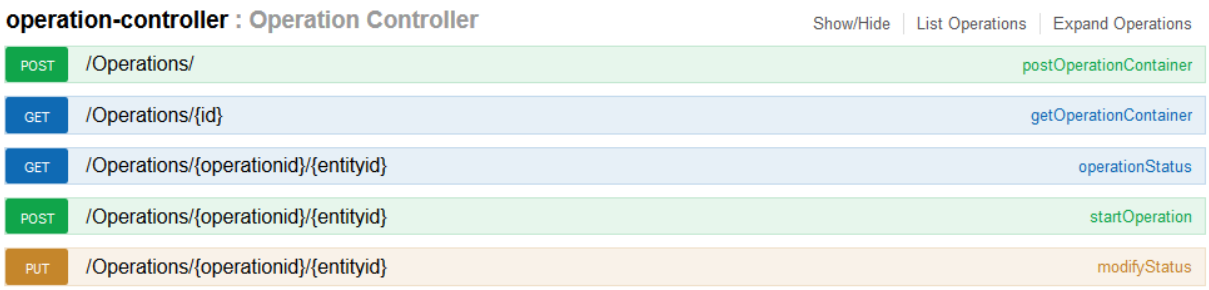
A `GET /RasterImage/{id}` visszaadja az adott id-jú képet az adatbázisból.

A `POST /RasterImageMultipart` metódussal van lehetőség a webes felületen keresztül kitallózni a fájlrendszerből a feltöltendő képet és feltölteni. Sikeres feltöltés után a metódus visszaadja a fájl generált id-ját.

Készült továbbá egy `POST /RasterImage` metódus, amivel a .NET-es oldalról lehet feltölteni a képet. Mivel a webes (multipart/form-data) és a .NET-es (application/octet-stream) formátum eltérő, ezért ez a webről nem érhető el.

Az OperationContainer modell felépítése:

```
OperationContainer {
   operations (Array[Operation])}
Operation {
   AddGlobalStatistics (AddGlobalStatistics),
   AddOutterSquare (AddOutterSquare),
   ContactUI (ContactUI),
   DeleteSelected (DeleteSelected),
   RandomMarker (RandomMarker),
   SelectSmall (SelectSmall),
   id (string)}
AddGlobalStatistics {}
AddOutterSquare {}
ContactUI {}
DeleteSelected {}
RandomMarker {
   randomNumber (integer)}
SelectSmall {
   size (integer)}
```



55. ábra: OperationContainer metódusok

Az operation-controller csoportban találhatóak az operation-ök létrehozásával, elindításával, állapotának lekérésével kapcsolatos metódusok.

A `POST /Operations/` metódussal lehet új OperationContainer-t előállítani. A JSON-ös body-ban megadható mely operation-ök, milyen sorrendben, hányszor szerepeljenek az OperationContainer-ben. A metódus visszatérési értéke az operation flow azonosítója.

A `GET /Operations/{id}` visszaadja az adott id-jú OpeationContainer konfigurációját.

A `GET /Operations/{operationid}/{entityid}` visszadja az adott OperationContainer adott EntityContainer-en történő futtatásának állapotát. Ez lehet: created, started, waiting, ui_modified, finished, exception.

A `POST /Operations/{operationid}/{entityid}` indítja el az adott EntityContainer-en az adott OperationContainer futását.

A `PUT /Operations/{operationid}/{entityid}` csak tesztelés céllal került a webre. A UI oldal beavatkozását lezáró metódus. Ezzel jelezzük a szerverhez, hogy végeztünk a kliens oldali módosításokkal, body-ként elérhető a módosított EntityContainer.

A jelenleg elérhető operation-ök:

- AddGlobalStatistics
  - A kapott EntityContainer-ben létrehoz egy új entitást, melyen elhelyezi az alábbi tag-eket: GlobalStatistics tag: jelzi, hogy itt vannak a statisztikák; EntityNumber tag: a hozzá tartozó érték az EntityContainer-ben lévő entitások száma.
- AddOutterSquare
  - Az EntityContainer-ben lévő entitások köré rajzolja a külső négyszögét.
- ContactUI
  - Felhasználói beavatkozást kér. Jelzi a UI számára, hogy az EntityContainer-ben felhasználói beavatkozás kell. Megvárja, amíg a UI letölti a projektet, módosítja, majd visszatölti. A visszatöltéskor fejeződik be a ContactUI operation és megy tovább a végrehajtás a következőre.
- DeleteSelected
  - Törli az EntityContainer selected tag-gel ellátott entitásait.
- RandomMarker(int: randomNumber)
  - A paraméterként adott számú entity-re rárak egy NeedsUserCheck tag-et.
- SelectSmall(int: size)

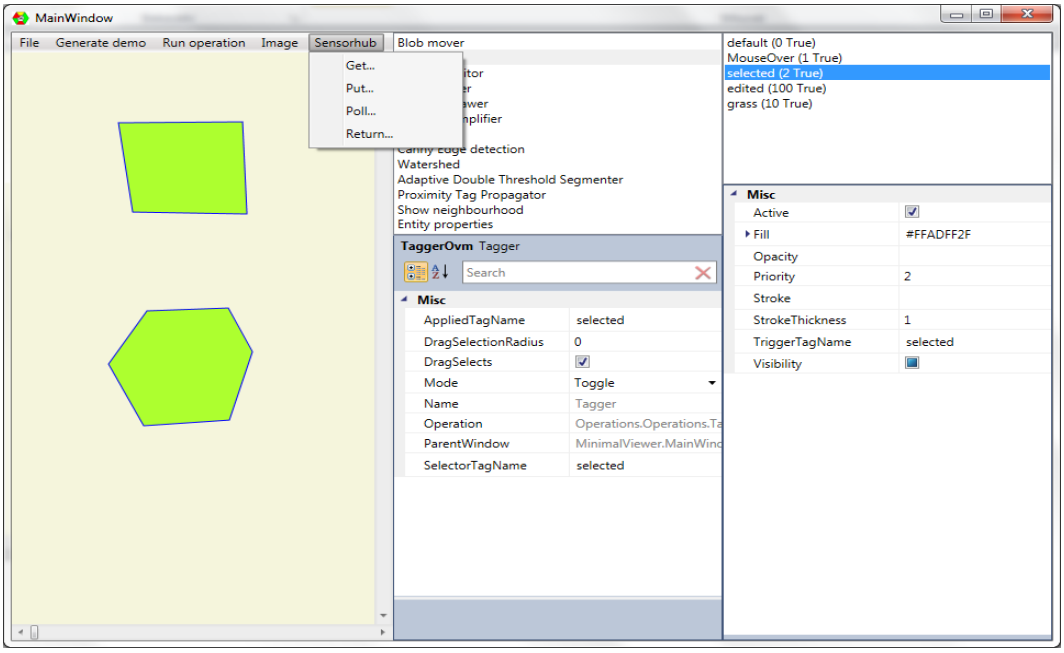
- A paraméterként átadott területnél kisebb méretű entitásain beállítja a selected tag-et.

A CV4SensorHUB projectben lévő .NET-es, WPF-es kliens programmal lehet entitásokat (poligonokat) rajzolni, módosítani, tag-gel ellátni stb.

Az általam létrehozott SensorHUB menüben négy lehetőség érhető el:

- Get… : Letölti az adott id-jú EntityContainer-t a szerverről.
- Put… : Feltölti a kiválasztott EntityContainer-t a szerverre.
- Poll…: Letölti az adott adott OperationContainer-t futtató EntityContainer-t.
- Return…: Visszatölti az adott OperationContainer-t futtató EntityContainer-t.

A Poll… és Return.. műveletekhez létrehoztam egy külön WPF-es ablakot is, ahol meg lehet adni az OperationContainer és EntityContainer azonosítóit.



56. ábra: A CV4SensorHUB kliensprogram a Sensorhub menüvel

Készítettem egy demo folyamatot, melynek segítségével lehet szemléltetni a szerver oldali operation-ök hasznosságát, valamint a felhasználói beavatkozás működését.

Először létrehoztam a szerver felületén egy új OperationContainer-t, melybe három műveletet tettem bele.

- Az első a SelectSmall, ami selected tag-et tesz a jelen paraméterezés szerint 900 pixelnél kisebb területű poligonokra.
- A második a ContactUI, vagyis a felhasználói beavatkozás lehetősége a kliens oldalon.
- A harmadik pedig a DeleteSelected, ami törli a selected tag-es poligonokat.

```
{
   "operations": [
         {
               "id": 1,
               "SelectSmall":
                     {
                           "size" : 900
                     }
         },
         {
               "id": 2,
               "ContactUI": {}
         },
         {
               "id": 3,
               "DeleteSelected": {}
         }
   ]
}
```

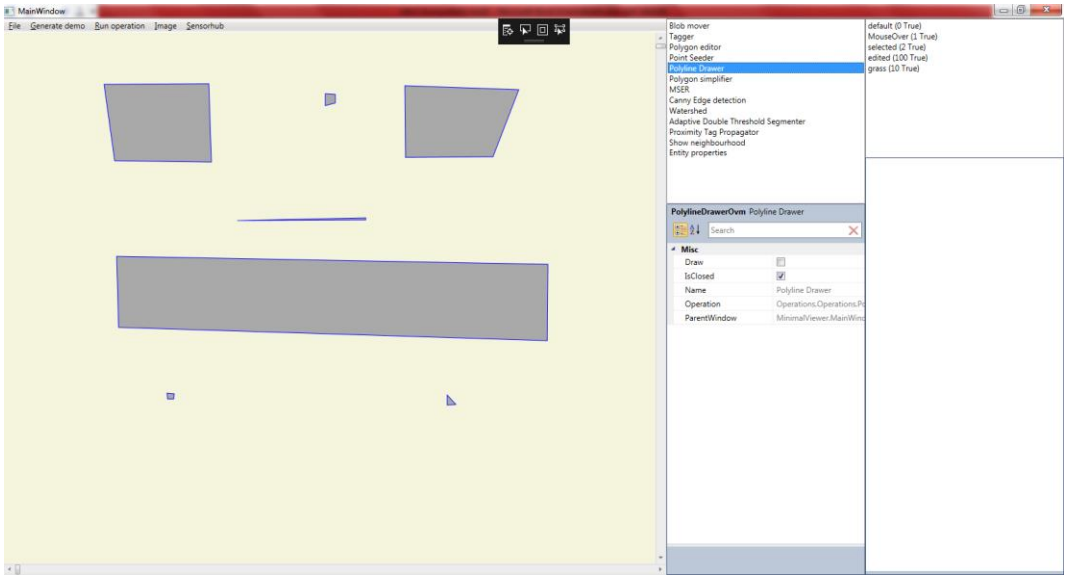
Az OperationContainert a következő URL-lel, annak body-jaként küldtem fel a szerverre:

```
POST http://localhost:8082/Operations/
```

Válaszként megkaptam az OperationContainer azonosítóját:

```
"94974d22-6f0b-419a-b91f-0acc90e8d4f5"
```

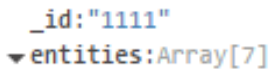
Ezután a CV4SensorHUB kliens alkalmazásban rajzoltam egy hét poligonból álló EntityContainer-t, melyben három poligon területe kisebb 900 pixelnél, és egy negyediknek is majdnem.



57. ábra: Hét poligont tartalmazó EntityContainer

Az EntityContainer-t elmentettem 1111.cv4s néven, majd a Sensorhub menü, Put… műveletével feltöltöttem a szerverre.

Ezután megnéztem a szerveroldali adatbázist megjelenítő MongoDB Compass programmal, hogy mi került bele az adatbázisba. (Egy csak szemléltető lépés, valós esetben nincs rá szükség.)



58. ábra: Hételemű EntityContainer a MongoDB Compass-ban

Most rendelkezésre áll az EntityContainer, illetve egy OperationContainer is a szerveren. Indítsuk el az 1111.cv4s EntityContainer-en a 94974d22-6f0b-419a-b91f-0acc90e8d4f5 OperationContainer-t. Ezt a következő URL meghívásával tehetjük meg:

```
POST: http://localhost:8082/Operations/94974d22-6f0b-419a-b91f-0acc90e8d4f5/1111
```

Ekkor elindul az OperationContainer futása, ezt a szerveroldal debug ablakán is követhetjük. (Egy szintén csak szemléltető lépés, valós esetben nincs rá szükség.)

```
Metadata{status='started', position=0}
Operation id: 1
1: SelectSmall 900
Operation id: 2
2: ContactUI
Metadata{status='waiting', position=2}
```

59. ábra: Szerveroldal debug ablaka ContactUI-ig

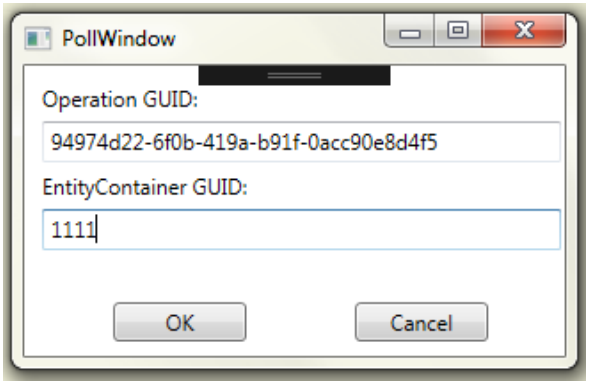
De erre REST API-n is van lehetőség a következő URL meghívásával:

```
GET: http://localhost:8082/Operations/94974d22-6f0b-419a-b91f-0acc90e8d4f5/1111
```

A válasz: "`waiting`"

Vagyis a szerveren az OperationContainer futtatása leállt, a megfelelő metaadatok beállításra kerültek. Eközben bármi mással tud foglalkozni a szerver (másik OperationContainer futtatása, adatmentés és kiolvasás, képmentés és kiolvasás, stb.), nincsen várakozó ciklusban.

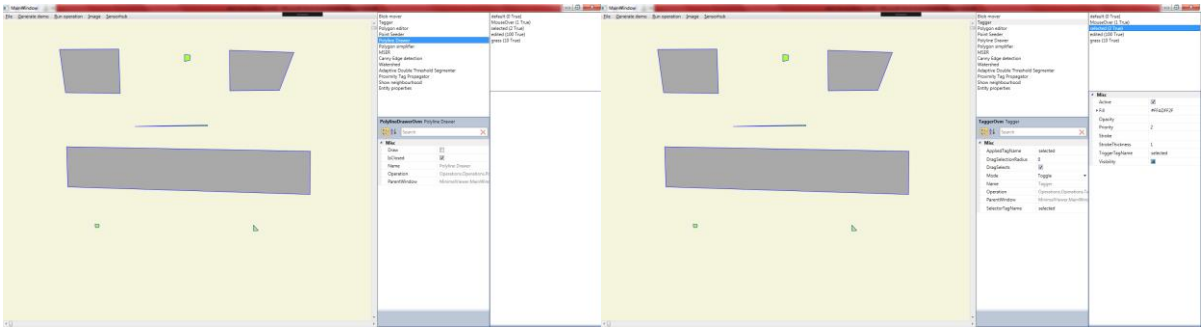
Most visszatérve a kliensbe lekérhetjük a módosított EntityContainer-t a Sensorhub menü Poll… műveletének segítségével. Ekkor meg kell adni a két azonosítót, hogy melyik EntityContainer-en futó, melyik OperationContainer hívott ContactUI-t. (Ez a felület később egyszerűsítve lesz.)



60. ábra: Azonosítók megadása a Poll művelethez

Az adatok beírása és az OK gomb megnyomása után megjelenik a 1111.cv4s, de ebben már három poligonnak zöld lett a háttérszíne, ami a selected tag-gel történt megjelölést jelenti.

Ekkor felhasználói beavatkozásként még a negyedik, a többinél jóval kisebb területű poligont is megjelöltem a selected tag-gel.



61. ábra: Felhasználói beavatkozás (negyedik poligon megjelölése)

Ezután a Sensorhub menü Return… műveletével visszatöltöttem az EntityContainer-t.

Ekkor a szerver folytatta az OperationContainer futtatását.

```
Metadata{status='started', position=0}
Operation id: 1
1: SelectSmall 900
Operation id: 2
2: ContactUI
Metadata{status='started', position=2}
Operation id: 3
3: DeleteSelected
Metadata{status='finished', position=3}
```

62. ábra: Szerveroldal debug ablaka a lefutás végén

Majd az állapotlekérő URL meghívása után, láthatjuk, hogy a futás befejeződött.

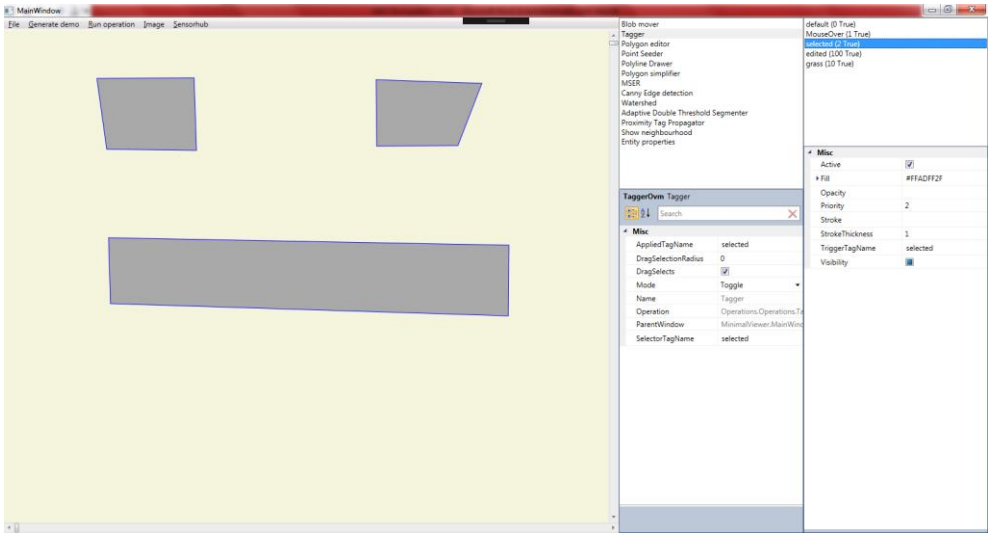
```
GET: http://localhost:8082/Operations/94974d22-6f0b-419a-b91f-0acc90e8d4f5/1111
"finished"
```

Az adatbázisban látható, hogy hét poligon helyett már csak három van az EntityContainer-ben.

```
_id:"1111"
▾entities:Array[3]
```


63. ábra: Háromelemű EntityContainer a MongoDB Compass-ban

Végül a Sensorhub menü Get… műveletével letöltve az 1111.cv4s EntityContainer-t látható, hogy a lefutás valóban sikeres volt, és a három nagy poligon maradt a fájlunkban. Az így előállt fájl tartalma a függelékben látható.



64. ábra: Eredmény az OperationContainer lefutása után

Extra lehetőségként az adatbázisból kiolvasható, hogy az egyes OperationContainer-ek mely EntityContainer-eken futnak vagy futottak, és mik a kapcsolódó állapotaik az adott pillanatban.


```
_id:"94974d22-6f0b-419a-b91f-0acc90e8d4f5"
▾operations:Array[3]
  ▾ 0:Object
    _id:"1"
  ▾ selectSmall:Object
    size:900
  ▾ 1:Object
    _id:"2"
  ▾ contactUI:Object
  ▾ 2:Object
    _id:"3"
  ▾ deleteSelected:Object
▾metadata:Object
  ▾ 0:Object
    status:"created"
    position:0
  ▾ 1111:Object
    status:"finished"
    position:3
```

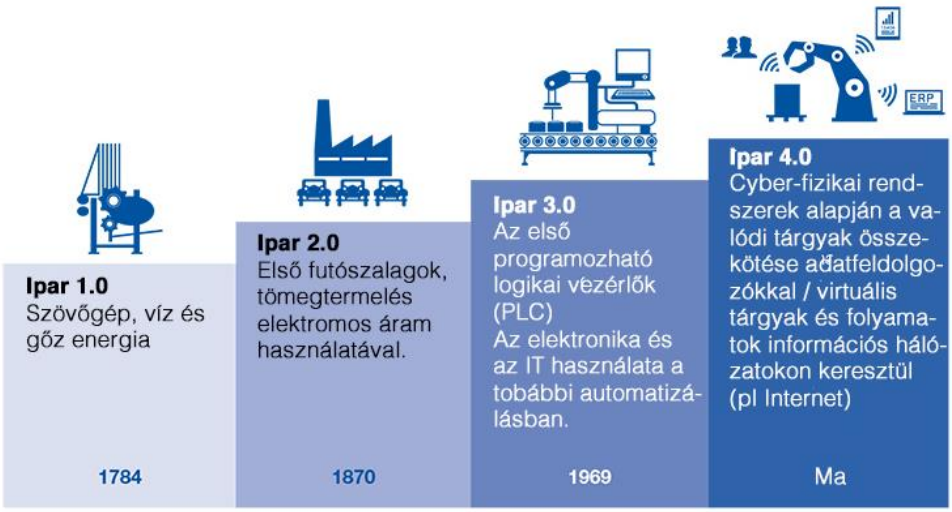

65. ábra: Az OperationContainer metaadatai

4. A SensorHUB jövőbeli lehetőségei

Az előző fejezetekben bemutatott SensorHUB alapú konkrét alkalmazásokból látható, hogy a koncepciónak és a keretrendszernek biztos jövője van, hiszen a megfelelő mikroszolgáltatások implementálása után bármely szakterületről származó komponens bekapcsolható a rendszerbe.

Természetesen a keretrendszer fő elemei folyamatos fejlesztés alatt állnak, lépést tartva a legújabb technológiákkal, így egy frissen publikált, valamilyen módon számunkra hasznos implementációkat mindenképpen beépítjük a rendszerünkbe.

Jelen állás szerint a SensorHUB életciklusát két irányba tervezzük és fejlesztjük. Az egyik a tanszéken belüli különböző más projektek, kutatások és fejlesztések alapjául szolgáló IoT keretrendszer, a másik pedig bekapcsolódás a piacra és a SensorHUB szolgáltatásainak és erőforrásainak kiajánlása az ipari, vállalati partnerek felé.

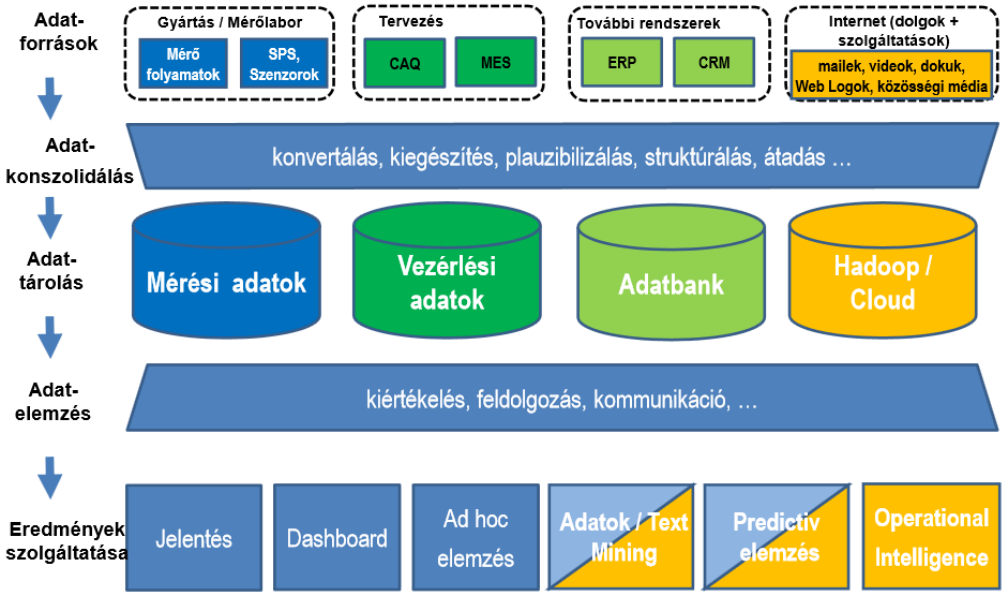


66. ábra: Ipari forradalmak [45]

A második irányba kapcsolódik be a napjainkra fogalmaink közé épülő Ipar 4.0, mely szemléletmód a negyedik ipari forradalom elérkezését takarja. [45] Az első ipari forradalom, mely egy az átfogó társadalmi, gazdasági és technológiai változást jelent, a XVIII. század második felében és a XIX. század első felében zajlott le, először Nagy-Britanniában majd Európában végül Amerikában. Alapját az 1769-ben Watt által elkészített gőzgép és a Cartwright által 1784-ben elkészített szövőgép jelentette, mely gépesítéseket ezután egyre több és több ipari felhasználásba építettek be. [46] A nagyhatású fejlődések és fejlesztések majdnem 100 év múlva elhozták a második ipari forradalmat, mely a futószalagos gyár, újabb találmányok, elektromosság bevezetésével segítették a gazdasági és társadalmi fejlődést. Megint 100 évnek kellett eltelnie, mire az elektromos ipar odáig fejlődött, a harmadik ipari

forradalomnak nevezett korszakban hogy elkészültek az első számítógépek, mikrovezérlők, 1969-ben pedig megjelent az internet, vagyis a hálózatba kapcsolt számítógépek első realizálása. Eltelt 40-50 év és eljutottunk napjainkra a negyedik ipari forradalomhoz, melynek alapját az internet, a hálózatba kapcsolt számítógépek és egyéb eszközök, valamint a mesterséges intelligencia jelenti.

Az Ipar 4.0 koncepciója az internet, IP-alapú, hálózatba kapcsolt számítógépek, eszközök és szenzorok bekapcsolása a gyártási folyamatokba, Big Data alapú adattárolással és gépi tanulási algoritmusokkal kibővítve (kiber-fizikai rendszerek és a „tárgyak internete"). Vagyis a koncepció célkitűzése az önkonfigurálhatóság, önoptimalizálás, öndiagnózis és megítélés szemlélete mellett a kompakt és intelligens termelés megvalósítása a rendelkezésre álló technológiák kihasználásával.



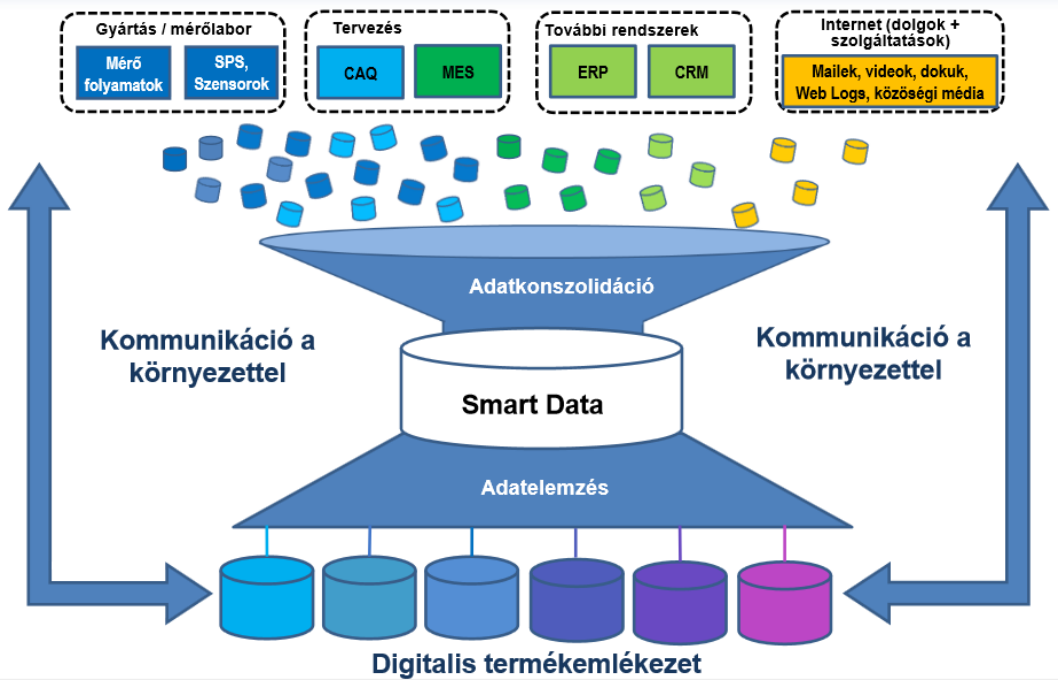
67. ábra: Hagyományos és új elemek az ipari termelésben [47]

A Big Data elveket követve az ipari és gyártási folyamatok adatforrásai felől érkező adatok először adatkonszolidáláson mennek keresztül, mely a konvertálás, kiegészítés, plauzibilizálás, struktúrálás, átadás stb. műveleteket jelentheti.

Az így elkészült tisztított, transzformált adatok kerülnek eltárolásra a megfelelő adatbázisokban.

Az eltárolt adatokból az eredmények, mérőszámok, jelentések stb. pedig a megfelelő adatelemzések lefuttatása után állnak elő, mely lehet adatszelektálás, statisztikai kiértékelés, feldolgozás, kommunikáció stb.

Az előbbi fázisok leképezhetőek a CAMERA koncepció megvalósítására, mely a Collecting (gyűjtés), Assessing (megfigyelés), Managing (irányítás), Evaluating (kiértékelés), Reporting (jelentés) és Archiving (archiválás) fázisokból áll össze. [47]



68. ábra: Industry 4.0 – Quality 4.0 [47]

A CAMERA koncepciót megfelelő szintű minőségmenedzsmenttel kiegészítve pedig előáll az Ipar 4.0 és Minőség 4.0 szimbiózisa. [47]

Ebbe a koncepcióba kapcsolódik bele a SensorHUB keretrendszer is, mely a megfelelő adminisztrációs és menedzsment komponensekkel kiegészítve meg tudja valósítani a CAMERA koncepciót, rendelkezésre tud állni az Ipar 4.0-ára átálló ipari partnerek rendszereinek alapjául, és megfelelő módon biztosítva lesz a használt szolgáltatások és adatmennyiségek monitorozása, melyek felhasználásával, valamint pénzügyi metódusok integrálásával a teljesítmény és kihasználás alapú számlázás is elérhetővé válik a felhasznált mennyiségek alapján.

Ez tehát a SensorHUB rendszer fejlesztésének következő mérföldköve, készíteni egy demonstrációs célú megvalósítást a SensorHUB Ipar 4.0 területű felhasználásával, valamilyen elképzelt gyártási folyamat vezérlésének, gépsorainak és szenzorainak IoT-sétésével, hálózatba kapcsolásával, SensorHUB-hoz illesztésével.

Ehhez hátra van az ehhez szükséges metrikákat mérő, és számlázási mikroszolgáltatások implementálása és integrálása a rendszerbe, valamint további adatbiztonságot megvalósító és megfelelő autentikációt támogató szolgáltatások beépítése.

Mindezek után a SensorHUB keretrendszer készen áll majd rá, hogy az Ipar 4.0 szereplői felől érkező kéréseknek is eleget tegyen, és biztos alapot szolgáltasson az ipari forradalom folytatásához.

5. Összefoglalás

Tudományos diákköri dolgozatomban bemutattam az Automatizálási és Alkalmazott Informatikai Tanszéken kigondolt SensorHUB koncepciót, illetve SensorHUB keretrendszert, melynek fejlesztésébe bekapcsolódhattam az elmúlt év során.

A bevezetés során rámutattam az internetre kapcsolódó eszközök számának egyre jobban gyorsuló növekedésére, melyek gerjesztették a kommunikációs protokollok megújítását: IPv4 kiegészítései, IPv6 szabványosítása és bevezetése, valamint az „emberek internete” (egy ember, egy kapcsolódó eszköz) fogalomvilága mellé behozta a „tárgyak internete” fogalmat is, mely világban, már az eszközök önállóan, autonóm módon kapcsolódnak a világhálóra, és kommunikálnak a többi eszközzel vagy emberrel. A „tárgyak internete” vagyis az IoT segítette újabb és újabb internet alapú koncepciók kigondolását és egy globális internet alapú társadalom és a ráépülő élő, intelligens alkalmazások (város, otthon stb.) megvalósítását.

Ennek az elméletnek egyik konkrétabb változata a SensorHUB koncepció, valamint implementációja a SensorHUB keretrendszer. Bemutattam a SensorHUB koncepció főbb komponenseit, valamint összehasonlítottam a piacon ma elérhető hasonló szolgáltatást nyújtó rendszerekkel, illetve kitekintettem egy akadémiai kutatásra és egy továbbfejlesztési lehetőségre, IIoT koncepcióra, a modellvezérelt több szakterületes IoT koncepcióra.

Következett a SensorHUB keretrendszer 1.0-ás, illetve 2.0-ás változatainak részletes bemutatása, a kapcsolódó fogalmak, technológiák és szolgáltatások részletes definiálásával és hivatkozásával. Szemléltetem a SensorHUB három rétegének egymásra épülését vagyis a rendszer mindentudó adatbázisaként szolgáló Hadoop réteget, a koncepció lelkét adó microservice alapú Node.js réteget, illetve bemutattam számos, már a SensorHUB-ra épülő alkalmazást és fejlesztést, ezek közül is az Európai Uniós támogatású, a közeljövőben Budapesten is bevezetésre kerülő SOLSUN projektet.

A SensorHUB kiterjesztése fejezetben bemutattam azokat a munkákat, melyek fejlesztésébe már én is bekapcsolódtam, vagy éppen teljes mértékben én oldottam meg.

Az első ilyen projekt a Social Driving okostelefonos alkalmazás, mely egyfajta közösségi vezetést valósít meg, az autó OBD interfészére csatlakozó adapter adatfolyamának a SensorHUB rendszerbe történő továbbítása segítségével. Az interfészről származó adatok megjelennek közvetlenül a telefonunk képernyőjén, illetve a szerverig eljutva, és onnan visszatöltve megkapjuk a korábbi historikus adatokkal kiátlagolt értékeket, valamint összehasonlítást a többi felhasználó értékeivel. Az alkalmazás funkcióinak és koncepciójának bemutatására képernyőképekkel kiegészített demonstrációs példát is bemutattam.

A Social Driving alkalmazás adatainak biztossága céljából készítettem el az ObdCanCompare okostelefonos alkalmazást, amiben a Social Driving-ben használt OBD interfészen, implementáltam a CAN bus interfész felől érkező adatfolyam fogadását és továbbítását a szerver felé. Az alkalmazás képernyőjén diagrammok segítségével, illetve a szerveroldalon riportkészítő szolgáltatás segítségével összehasonlíthatóvá váltak a két interfész paraméterei, és levonhattuk azt a tanulságot, hogy a Social Driving alkalmazásban használt paraméterek az OBD interfészen is megfelelő pontossággal érkeznek, vagyis hitelesen biztosítja az adatokat az alkalmazásunk. A funkciók és koncepció bemutatására ebben az esetben is mutattam képernyőképekkel kiegészített demonstrációs példát.

Végül az okosautó – okosváros szakterületet elhagyva, egy gépi látással kapcsolatos szerveroldali komponens fejlesztésével foglalkoztam az elmúlt hónapokban. Ez a CV4SensorHUB projekt, mely számos további gépi látás témájába tartozó tanszéki projekt közös komponensét jelenti, melynek segítéségével, a projektekből származó poligonhalmazokon, EntityContainer-eken és a hozzájuk kapcsolódó raszteres képeken lehet manipulációs műveletsorokat definiálni, mely a mikroszolgáltatás adatbázisában eltárolásra kerül. Ezen transzformáció-láncok, OperationContainer-ek a későbbiekben bármikor futtathatóak bármely adathalmazon. A komponens kooperálni tud a kliens oldali felhasználói felülettel, és a megfelelő művelethívás esetén be lehet avatkozni, korrigálni lehet a transzformáció lefutását. Ehhez a projekthez kapcsolódóan is mutattam képernyőképekkel kiegészített demonstrációs példát. A jövőben tervezzük az automatizált adatfeldolgozó, és perdikciós algoritmusok implementálását. Távlati célunk pedig bekapcsolódni a SOLSUN forgalomszámláló komponens adatainak feldolgozásába, valamint a Social Driving térképes adataira alapuló tanulói rendszerek integrációjának megvalósítása.

Az utolsó fejezetben áttekintést adtam az ipari forradalom jelenlegi állásáról, mely már a negyedik ipari forradalom a történelemben, és korunkhoz híven megkapta az Ipar 4.0 elnevezést. A jelenlegi állás szerint a SensorHUB koncepció és keretrendszer még néhány komponens lefejlesztése után be tud kapcsolódni, meg van a létjogosultsága és helyt tud állni ebben a világban is.

Az elmúlt 3 évben ebben az ívben zajlott a SensorHUB és a kapcsolódó projektek fejlesztése, és a fejlesztőtársak, ötletgazdák nevében is mindannyian bízunk benne, hogy a következő években is még legalább ennyi előrehaladást, kutatási eredményt tudunk mutatni az IoT világ fejlődésével párhuzamosan és biztos alapot szolgáltatunk az ipari forradalom folytatásához.

KÖSZÖNETNYILVÁNYÍTÁS

A tudományos diákköri dolgozat elkészítéséhez szeretném megköszönni konzulenseimnek Ekler Péternek és Csorba Kristófnak a támogatását és motiválását, a folyamatos konzultációkat, a szakmai segítséget, valamint a lehetőséget, hogy bekapcsolódhattam a projektek tervezésébe, kidolgozásába és fejlesztésébe.

Köszönet jár továbbá a SensorHUB fejlesztésében legjártasabb kollégáimnak, Balogh Tamásnak és Tömösvári Imrének, akik segítettek bekapcsolódni a projektbe, bemutatták a SensorHUB-ot és annak komponenseit, és mindig a rendelkezésemre álltak, ha elakadtam vagy kérdés merült fel bennem.

Köszönet Charaf Hassannak és Lengyel Lászlónak a szakirodalmakért és az általános mellettem állásért és a jó tanszéki közeg biztosításáért.

Végül köszönet illeti azokat a hallgatótársakat, kollégákat, akik közreműködtek vagy közreműködnek a SensorHUB fejlesztésében, és viszik előrébb a projektet.

A kutatás és a dolgozat az  Emberi Erőforrások Minisztériuma ÚNKP-16-2-I. kódszámú Új Nemzeti Kiválóság Programjának támogatásával jött létre.

A dolgozat elkészítését a Magyar Tudományos Akadémia Bolyai János Kutatási Ösztöndíja támogatta.

TÁBLÁZAT- ÉS ÁBRAJEGYZÉK

IRODALOMJEGYZÉK

[1] Charaf Hassan: *Az infoszféra tudást közvetítő szerepe a mai társadalomban* http://www.matud.iif.hu/2015/02/03.htm (2016. szeptember)

[2] The Number Resource Organization: Free Pool of IPv4 Address Space Depleted. https://www.nro.net/news/ipv4-free-pool-depleted (2016. október)

[3] Internet Assigned Numbers Authority: Number Resources https://www.iana.org/numbers (2016. október)

[4] Overview of „Internet of Things" http://mfieldforce.in/blog/tag/internet-of-things/ (2016. október)

[5] A Föld lakossága http://www.worldometers.info/world-population/ (2016. szeptember)

[6] IPv6 Deployment Around The World http://ipv6.com/articles/deployment/IPv6-Deployment-Status.htm (2016. október)

[7] Dr. Sallai Gyula: *Mérnöki menedzsment* tantárgyi jegyzetek https://qosip.tmit.bme.hu/foswiki/bin/view/VITMM112/ (2016. október)

[8] SensorHUB keretrendszer https://www.aut.bme.hu/Pages/Research/SensorHUB (2015. október)

[9] László Lengyel, Péter Ekler, Tamás Ujj, Tamás Balogh and Hassan Charaf: *SensorHUB: An IoT Driver Framework for Supporting Sensor Networks and Data Analysis*, International Journal of Distributed Sensor Networks, 2015. http://www.hindawi.com/journals/ijdsn/2015/454379/ (2016. október)

[10] Innovációs díjban részesült a SensorHUB https://www.vik.bme.hu/hir/874-bme-innovacios-dijban-reszesult-a-sensorhub-fejlesztes (2016. október)

[11] Amazon Web Services IoT platform https://aws.amazon.com/iot/ (2016. október)

[12] SAP HANA Cloud Platform https://hcp.sap.com/index.html (2016. október)

[13] General Electric Predix platform https://www.predix.io/ (2016. október)

[14] VMTS keretrendszer https://www.aut.bme.hu/Pages/Research/VMTS/Introduction (2016. október)

[15] László Lengyel, Péter Ekler, Gergely Mezei, Bertalan Forstner, Tamás Balogh, Imre Tömösvári, Tamás Ujj and Hassan Charaf: Model-driven Multi-Domain IoT, elfogadott könyvfejezet

[16] Tamás Balogh, Tamás István Ujj: ICT Framework for Supporting Connected Road Vehicles, BME-VIK Tudományos Diákköri Konferencia, 2014.

[17] REST https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (2016. október)

[18] HTTP methods https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html (2016. október)

[19] Tömösvári Imre: SensorHUB dokumentáció

[20] MQTT protokoll https://en.wikipedia.org/wiki/MQTT (2016. október)

[21] Cloudera CDH components http://www.cloudera.com/products/apache-hadoop/key-cdh-components.html (2016. október)

[22] Gulyás Máté, Kazi Sándor, Prekopcsák Zoltán: *Big Data elemzési eszközök nyílt forráskódú platformokon* tantárgyi jegyzetek http://dmlab.hu/subject/big-data/ (2016. október)

[23] Microservices architektúra http://microservices.io/index.html (2016. október)

[24] Microsoft Azure, Microservices architektúra https://msdn.microsoft.com/en-us/magazine/mt595752.aspx (2016. október)

[25] Google Cloud Messaging https://developers.google.com/cloud-messaging/ (2016. október)

[26] SOLSUN weboldal http://www.solsun.co.uk/index.php/solsun (2016. október)

[27] Budapest Főváros Önkormányzata http://budapest.hu/Lapok/2015/solsun.aspx (2016. október)

[28] BME-VIK SOLSUN hír https://www.vik.bme.hu/hir/857-solsun-a-fenntarthato-kulteri-vilagitas (2016. október)

[29] VehicleICT keretrendszer https://www.aut.bme.hu/Pages/Research/vehicleict (2016. október)

[30] László Lengyel, Péter Ekler, Tamás Ujj, Tamás Balogh and Hassan Charaf: *Social Driving in Connected Car Environment*, European Wireless 2015.

[31] A SensorHUB, Mérnök újság, XXII. évf. 10. szám, 2015. október

[32] Hadoop Streaming https://hadoop.apache.org/docs/r1.2.1/streaming.html (2016. október)

[33] Inventure Kft, CAN bus communication http://fmsgateway.com/glossary/can-bus-communication (2015. november)

[34] Pentaho Community http://community.pentaho.com/ (2016. október)

[35] Dudás Ákos, Ekler Péter, Tömösvári Imre: *Üzleti intelligencia* tantárgyi jegyzetek https://www.aut.bme.hu/Course/BMEVIAUMA02 (2016. október)

[36] Hive Language https://cwiki.apache.org/confluence/display/Hive/LanguageManual (2016. október)

[37] HUE – The Hadoop UI http://gethue.com/ (2016. október)

[38] Oozie Workflow Scheduler http://oozie.apache.org/ (2016. október)

[39] CV4SensorHUB kutatás https://www.aut.bme.hu/Pages/Research/cv4s (2016. október)

[40] Spring MVC http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html (2016. október)

[41] MongoDB https://www.mongodb.com (2016. október)

[42] NoSQL adatbázisok http://nosql-database.org/ (2016. október)

[43] Apache Tomcat szerver http://tomcat.apache.org/ (2016. október)

[44] Swagger UI http://swagger.io/swagger-ui/ (2016. október)

[45] Lignomat Kft. Hálózatba szervezett gyártás, avagy mi az az "Ipar 4.0" http://lignomat.hu/HiREK/Halozatba_szervezett_gyartas_avagy_mi_az_az_Ipar_40.html (2016. október)

[46] Találmányok, tudományos felfedezések/alkotások, elsők - az ipari forradalomtól napjainkig http://www.amegoldas.eoldal.hu/cikkek/talalmanyok---az-ipari-forradalomtol-napjainkig.html (2016. október)

[47] Q-DAS GmbH & Co. KG. Big Data – Industry 4.0 – Quality http://docplayer.hu/3319879-Big-data-industry-4-0-quality.html (2016. október)

FÜGGELÉK

A 1111.c4s fájl tartalma a szerveroldali transzformáció futtatása után (3 poligonnal):


```
{
   "entities":[
      {
         "__type":"Polygon:Core.Model",
         "aux":[

         ],
         "t":636114382757905032,
         "tags":[
            {
               "Key":"id",
               "Value":1
            },
            {
               "Key":"default",
               "Value":0
            }
         ],
         "points":[
            {
               "_x":151,
               "_y":78.039999999999992
            },
            {
               "_x":309,
               "_y":77.039999999999992
            },
            {
               "_x":313,
               "_y":195.04
            },
            {
               "_x":167,
               "_y":193.04
            }
         ]
      },
      {
         "__type":"Polygon:Core.Model",
         "aux":[

         ],
         "t":636114388051987836,
         "tags":[
            {
               "Key":"id",
               "Value":5
            },
            {
               "Key":"default",
               "Value":0
            }
         ],
```

```
            "points":[
               {
                  "_x":605,
                  "_y":80.039999999999992
               },
               {
                  "_x":606,
                  "_y":188.04
               },
               {
                  "_x":738,
                  "_y":187.04
               },
               {
                  "_x":777,
                  "_y":86.039999999999992
               }
            ]
         },
         {
            "__type":"Polygon:Core.Model",
            "aux":[

            ],
            "t":636114388217687314,
            "tags":[
               {
                  "Key":"id",
                  "Value":9
               },
               {
                  "Key":"default",
                  "Value":0
               }
            ],
            "points":[
               {
                  "_x":170,
                  "_y":337.04
               },
               {
                  "_x":821,
                  "_y":349.04
               },
               {
                  "_x":820,
                  "_y":464.04
               },
               {
                  "_x":173,
                  "_y":444.04
               }
            ]
         }
      ]
   }
```