Kot skupina smo se ves čas koordinirali in usklajevali naloge tako, da smo lahko izdelali full stack. Pri predmetu RAIN smo se osredotočili na praskanje podatkov iz interneta ter prikaz na čelnem delu, v katerega smo vključili tudi podatke senzorjev, katere smo implementirali v okviru predmeta NPO. Implementacija je vsebovala GPS, žiroskop in pospeškometer. Preden smo začeli, smo okvirno že vedeli katere tehnologije bomo uporabili predvsem zaradi predhodnih izkušenj. Odločili smo se za naslednje:

- Angular
- Express
- MongoDB
- Heroku
- YOLO

Raziskali smo tudi njihove alternative, vendar smo ugotovili, da je bila naša prvotna izbira idealna za to, kar smo hoteli doseči.

Alternative (Angular)

VUE

Ena izmed dobrih alternativ Angularju je **Vue.js**, ki je odprtokoden framework za grajenje uporabniških vmesnikov in Single page aplikacij.

Podatki:

- **Avtor**: Evan You
- **Izdan**: Februar 2014
- **Jezik**: JavaScript

V primerjavi z Angularju je predvsem manjši in bolj fleksibilen ter se ga je lažje naučiti.

Vue vsebuje izjemno prilagodljivo arhitekturo ki se osredotoča na dinamično procesiranje in kompozicij komponent, podobno kot pri Angularju. Za podporo poti, upravljanja s stanjem in ostalih naprednih zadev, ima Vue več podpornih knjižnic in paketov, eden izmed najbolj popularnih je trenutno Nuxt.js. Vue.js omogoča razširitev HTML z direktivi. Ti so lahko vgrajeni, ali pa jih dodamo kar sami.

Komponente

Vue razširja osnovne HTML elemente z uporabo komponent. Ti so poljubni elemetni, katerim Vue prevajalnik doda obnašanje (behaviour). V Vue je komponenta v bistvu instanca z že v naprej določenimi možnostmi.

Predloge

Tako kot Angular, Vue uporablja povezavo med DOM in podatki Vue. Vse predloge so validni HTML, katere lahko prebavi vsak HTML parser. Vue predloge spremeni v virtualne DOM funkcije. Nato

virtualni DOM objekt omogoči upodabljanje komponent v spominu še preden se te prikažejo na brskalniku. Kombinirano z odzivnostjo, Vue lahko izračuna minimalno številno komponent katere je potrebno ponovno upodobiti ter tudi minimalno število DOM manipulacij.

Odzivnost

Vue odzivni sistem uporablja čisti JavaScript. Vsaka komponenta vsebuje seznam svojih odzivnih odvistnosti med upodabljanjem, zaradi tega sistem ve natančno kdaj ponovno upodobiti in s tem tudi katere komponente.

Poti

Tako kot Angular, Vue ponuja vmesnik, ki se spreminja glede na podan URL, ne glede na to kako je bil ta spremenjen (email link, osvežitev…). Uporaba čelnega dela »routerja« omogoča tranzicijo ob dogodkih na brskalniku (kliki na gumbe, povezave). Vue ne vsebuje hashiranega čelnega routinga.

Ko smo primerjali Vue in Angular, nam razlika ni bila tako gromozanska, saj sta bolj kot ne skoraj ista JS ogrodja, Vue je le malce manjši in okretnejši. Za Angular smo se odločili, ker nam je bližje in smo z njim že imeli opravka. Prav tako je bolj kompleksen in vsebuje mnogo več knjižnic, katere je moč uporabiti pri samem projektu. Seznanjeni smo tudi z Angular material, knjižnico, ki vsebuje UI elemente za izdelavo izgleda spletne aplikacije.

REACT

React ali ReactJS je JavaScript knjižnica za grajenje uporabniških vmesnikov. Trenutno ga vdržuje Facebook in skupina neodvisnih razvijalcev in podjetij.

React lahko uporabimo kot osnovo za SPA in mobilne aplikacije. Kakorkoli, React je namenjen le upodabljanju podatkov DOM-u, zato ustvarjanje React aplikacij potrebuje uporabo dodatnih knjižnic za vdrževanje stanja in poti. Primera teh sta **Redux** in **React Router.**

Podatki:

- Avtor: Jordan Walke
- Razvijalci: Facebook in skupnost
- Izdan: Maj 2013
- Jezik: JavaScript

Osnovna uporaba

```
<div id="myReactApp"></div>

<script type="text/babel">
  function Greeter(props) {
    return <h1>{props.greeting}</h1>
  }
  var App = <Greeter greeting="Hello World!" />;
  ReactDOM.render(App, document.getElementById('myReactApp'));
</script>
```

Videti je, da je poudarek na upodabljanju elementov DOM.

Komponente

Kot pri Vue in Angular, React prav tako uporablja komponente ali entitete. Te lahko dodamo poljubnemu HTML objektu. Ko upodabljamo komponento, ta poleg pošlje še podatke ki so znani kot »props«.

Funkcionalne komponente
Vračajo JSX (JavaScriptXML).

Razredne komponente
Znane tudi kot izjavne komponente. Njihovo stanje lahko obdrži vrednosti vseskozi komponente in jih lahko pošiljamo otroškim komponentam preko »propsov«.

Virtualni DOM (Document Object Model)
React v pomnilniku ustvari cache podatkovne strukture, izračuna razlike in nato upodobi prikaz na brskalniku. Ta proces je spravo ali reconciliation. To omogoči programerju, da programira v smislu kot da je stran osvežena ob vsaki spremembi, medtem ko React osveži le tiste komponente, katere se dejansko spremenijo. To omogoči gromozanski skok v zmogljivosti. Zaradi tega ni potrebno ponovno preračunavati CSS stila, postavitve strani in upodabljati celotne strani.

Življenski cikel
Pri angularju poznamo več metod, ki predstavljajo cikel komponente (NgOnInit, NgOnDestroy..), React pa ima malce drugačne. Med temi so shouldComponentUpdate, componentDidMount, componentWillUnmount… Ena izmed najpomembnejših pa je render.

AURELIA
Aurelia je eden novejših ogrodij za izdelavo spletnih aplikacij (izdan je bil leta 2015 in je trenutno v prvi verziji). Lahko bi tudi rekli, da je naslednja generacija le teh. Aurelia ponuja svež in vznemerljiv pristop razvoju čelnega dela aplikacij, ki je močnejši alternativam. Sama sintaksa je zelo podobna tisti, ki jo ima Angular.

Priročnost > konfiguracija
Cilj Aurelie je, da prioritizira priročnost tako, da ima manjši vpliv na razvoj. Ta način zmanjša število odločitev, ki jih morajo razvijalci narediti, brez da izgubijo fleksibilnost.

Moduli
Tako kot Angular je tudi Aurelia skupek JS modulov. Grajenje aplikacij je prav tako podobno Angularju; kompozicija preprostih komponent s standardnim JS jezikom in TypeScript razredi skupno z HTML predlogami. Moduli omogočajo slednje: Binding, Dependency Injection, Metadata, Routing ter Templating skoraj identično Angularju. Te module je moč uporabiti individualno v vsakem JS projektu, vključno z Node.js.

Standardi odprtega spleta
Aurelia uporablja DOM standard, ki je neodvisen od platforme in jezika. Prav tako uporablja lastni HTML parser in JS razširitve ter ob tem ohranja DOM API za optimizacijo delovanja.

Aurelia temelji na W3C, kar bo aplikacijam, ki bodo temeljile na Aurelii, omogočalo neoviran razvoj.

Binding ali povezovanje
Aurelia z lahkoto podpira enosmerno in dvosmerno povezovanje, medtem ko je to malce težje v primeru Angularja in lahko privede do zmede med razvijalci. Pri enosmernem povezovanju podatki tečejo v eno smer, iz objektov v UI, pri dvosmernem pa v obe smeri, se pravi izmenjava med UI in objekti, kar omogoča sinhronizacijo.

Aurelia v osnovi uporablja enosmerno povezavo.

Popularnost

V primerjavi je Angular v večih primerih bolj popularen kot Aurelia, verjetno zato, ker še ne obstaja veliko uspešnih aplikacij, ki bi temeljije na Aurelii.

Jezikovna podpora

Angular zahteva, da razvijalci pišejo strogo JS v TypeScriptu. Aurelia razvijalci lahko prav tako uporabljajo TypeScript vendar lahko uporabijo tudi ES ali EcmaScript, kar razvijalcem omogoči večjo fleksibilnost.

Routing

Oba ogrodja se soočata z routing konfiguracijo na podoben način, vendar je definicija poti malce različna. V Aurelii, razvijalci definirajo starševske poti na enem mestu, otroke pa v posameznih komponentah. Ta pristop popolnoma enkapsulira komponente ter skrije njihovo notranjo kompleksnost. Za primerjavo, v Angularju je potrebno definirati poti modula v centralizirani konfiguracijski datoteki, vključno s potmi, komponentami in otroki poti. To naredi definicijo poti kompleksnejšo v Angularju

Zakaj Angular in ne njegove alternative?

Aurelia je eden novejših ogrodij in ima podobno osnovo kot Angular, vendar odpravi nekaj kompleksnosti le tega. Vue je prav tako podoben Angularju, vendar v osnovi ne vsebuje toliko možnosti kot Angular. React JS je dandanes eden najhitrejših ogrodij kar se tiče spletnih aplikacij, vendar ne ustreza naši ciljni aplikaciji, predvsem zaradi omejenosti na UI. Zdelo se nam je, da bi naši aplikaciji bolj ustrezal Angular, predvsem zaradi predznanja o njemu in zaradi njegove zmogljivosti (preprosta implementacija Google maps, storitve, izgled).

Alternative (Express)

Pregled ogrodij za NodeJs

Da lahko na zaledju pri delu z omrežjem oz. izdelovanju funkcij spletnega strežnika v NodeJs pišemo krajšo, urejeno kodo ter pohitrimo razvoj naše aplikacije, je dobro uporabiti eno od ogrodij ustvarjenih prav v ta namen. Med funkcije, ki jih potrebujemo za razvoj in jih ta programska oprema navadno zagotavlja, spadajo olajšano delo s HTTP (vzpostavitev povezave, obdelava zahtev ter

odgovorov), usmerjanje (procesiranje URL), podpora na vmesnemu nivoju (middleware) in številne druge.

So ogrodja obveznost ali izbira?

Aplikacijo je mogoče razviti brez uporabe ogrodja, vendar moramo vedeti, da Node neposredno ne vsebuje podpore za spletni razvoj, predlog za dinamično ustvarjanje odgovorov itd.. V primeru, da se odločimo za razvoj brez ogrodja, moramo vso podporo za to načrtovati sami.

Ob primerjavi kode preprostih strežnikov z enakimi funkcijami, prvega s uporabo Expressa in drugega brez, so očitno vidne razlike v sami sintaksi oz. kompleksnosti le te. Z rastjo naše aplikacije v kode razsežnosti več sto ali tisoč vrstic, bi bilo zelo težko slediti poteku programa. Z uporabo ogrodja v kombinaciji s MVC načrtovalskim vzorcem, torej razbijanjem aplikacije na več manjših, preglednih delov, si lahko močno olajšamo razvoj (še posebej kadar na projektu dela več razvijalcev, kot to tudi poteka v našem primeru). Vseeno je izvajanje programa, napisanega v »nativnem« nodejs okolju, prilagojenega reševanju našega specifičnega problema, pogosto občutno hitrejše, kot če vmes stoji še neko ogrodje.

Izbira ogrodja

Z porastom popularnosti okolja NodeJs za izdelovanje programskih rešitev na zalednem delu se je na trgu pojavilo in se še pojavlja veliko število različnih ogrodij. Med bolj priljubljene štejemo Express, Hapi, Meteor, Sails, Koa, Feathers, Adonis, LoopBack in še. Ko se odločamo za uporabo specifičnega ogrodja, moramo pomisliti na tako tehnične, kot tudi poslovne potrebe naše aplikacije. Upoštevati moramo faktorje, kot so hitrosti izvajanja in učnega procesa razvijalcev, struktura izvorne kode, fleksibilnost, razširljivost (knjižnice/moduli), razvojna skupnost, dostopna dokumentacija,…

- ### Express

Ta trenutek najbolj priljubljeno spletno aplikacijsko ogrodje je zastonjska programska oprema, ki je bila izdana pod MIT licenco leta 2010. Med največje prednosti ogrodja zagotovo spada prav njegova razširjenost. Najbolj uporabljano ogrodje seveda uporablja velika množica ljudi, to posledično zagotavlja več dokumentacije, rednega vzdrževanja, podpore in sprotne odprave napak. Ob razvoju hitreje naletimo na ljudi s težavami podobnimi našim in s tem tudi njihovih rešitev. Največkrat ga najdemo v kombinaciji s MongoDB in AngularJs, kot sestavni del MEAN sklada pri full stack razvoju. Razvijalci lahko razvijajo tako zaledni, kot tudi čelni del, ker pa je učni proces uporabe Expressa relativno kratek (dokler razumemo javascript, ter osnovne značilnosti protokolov aplikacijske plasti) se iz zalednega dela lažje znajdejo tudi razvijalci iz drugih razvojnih področij (čelnega dela). Preprost strežnik lahko postavimo v le nekaj minutah.

Zahteve lahko speljemo na različne poti (usmerjanje), odvisno od metode http (get, post, put, delete, all) ali po potrebi, glede na parametre v URL-ju. Zahtevo predstavlja objekt req (http request), odgovor pa objekt res (http response).

Ko govorimo o middlewareu mislimo na sloje procesiranja zahteve (obdelovanje na vmesnem nivoju) oz. skozi katere korake mora zahteva potovati, od prejetja, pa do poslanega odgovora. Za uporabo v Expressu imamo na voljo veliko število raznih modulov (javascript knjižnic) različnih funkcionalnosti, od dela s piškotki in sejami (cookie-parser, session), avtentikacijo in avtorizacijo uporabnika (express-stormpath), prejemanje zahtev iz različnih domen (cors), varnostjo (helmet), delo z zahtevami, ki v telesu ne vsebujejo le besedila (multer), itd.. Torej vmesna programska oprema lahko vrši kakršno koli programsko kodo (večinoma spreminja objekte zahteve ali odgovora). Na samem koncu se cikel request-response ali zaključi (send(),end(),render(),..), ali pa se kliče naslednja vmesna programska oprema v verigi (kot je določena v usmerniku) s klicem funkcije next(). Standardne module koristimo

s namestitvijo (npm) in vključitvijo, nekateri, kot so npr. static, json, urlencode, pa so v express že vgrajeni.

Ni nujno, da so naši odgovori vedno preprosti json, namreč odgovarjamo lahko tudi s stranmi dinamične vsebine. Na strežniški napravi hranimo šablone spletnih strani, katere nato ob zagonu predprocesorja napolnimo s podatki, pridobljenimi iz podatkovne baze (view engines/template engines). Med bolj poznane spadajo hbs, ejs, pug, ...

- Hapi (http API)

Med Hapi in Express je več podobnosti, kot razlik. Je nekoliko novejše ogrodje, napram starejšemu, ustaljenemu, skoraj standariziranemu expressu.Express uporablja minimalističen (lightweight) pristop, je tanek sloj na vrhu osnovne funkcijonalosti NodeJs in se predvsem zanaša na middleware. Pri Hapiju pojma middleware ne poznamo, večina funkcij oz. programov, ki jih pri obdelovanju zahtev in odgovorov potrebujemo, je integriranih in zato navadno dodatna vljučevanja niso potrebna. Če potrebujemo funkcijonalosti, ki jih hapi v osnovi ne nudi, pa vključimo plugine ali razširitve (ang. pluggins, extensions).

Plugini (moduli) so drug drugega neodvisni, zato ni nameščanja dodatnih, za delovanje potrebnih programov (dependencies), vse je namreč zapakirano v en paket. Nekateri izmed mnogih so: joi , boom, glue, qs, vision,.. Sproti jih dodajamo s klicem funkcije set().

Razširitve oz. razširitvene točke (Extension Points) so členi verige funkcij, ki tvorijo življensko cikel zahteve na našem strežniku. Privzeto obstaja 7 točk, po vrsti glede na izvajanje razvrščene so: onRequest, OnPreAuth, onCredentials, onPostAuth, onPreHandler, onPostHandler, onPreResponse. Funkcionalnosti razširjamo s dodajanjem svoje kode v te točke oz. funkcije (podobno, kot pri razvoju aplikacj za android). To storimo s klicem funkcije ext(), kjer kot parameter podamo ime razširitvene točke.

Usmerjanje poteka s klicem funkcije route nad objektov server (glavni objekt, okoli katerega gradimo aplikacijo, kot app pri expressu), nato navedemo vse dodatne specifikacije: metodo, pot, katera funkcija naj se izvede. Sintaksa je bolj »izrazita«, zaradi tega tudi daljša.

Ob primerjavi hitrosti izvajanja express in hapi strežnikov ugotovimo, da hapi deluje nekoliko počasneje.

- Koa

Za razvojem ogrodja Koa stojijo razvijalci Expressa. Razlog, zakaj koa ni le nova verzija Expressa leži v tem, da so razlike med ogrodjema preprosto prevelike. Neko Aplikacijo, napisano s ogrodjem Express, bi bilo potrebno, za prenos v Koa ogrodje, napisati praktično na novo, zato so to programsko opremo raje izdali kot knjižnico zase.

Razlika je že v samem cilju ogrodij: cilj Expressa je nadgradnja oz. prilagoditev obstoječih Node funkcionalnosti, na voljo imamo dodatne metode za manipulacijo nad req in res objektoma in za usmerjanje. Cilj Koe je med tem zamenjati Node funkcionalnosti, Koa namreč privzeta req in res objekta ne nadgradi, pač pa ju nadomesti z njuno abstrakcijo, objektom ctx (kontekst) in njegovima lastnostima ctx.request in ctx.response. Tradicionalne funkcije vmesnega nivoja niso podprte (funkcije s argumenti tipa res,req,next), ob razvoju se moramo tudi izogniti uporabi Node lastnosti, kot so statusCode, write, writeHead, end in drugih.

Koa je izredno minimalistično ogrodje, sestavlja ga manj kot tisoč vrstic kode. Ne vključuje podpore za usmerjanje, za prenos datotek itd.. Pri izdelovanju potrebnih programov za upravljanje z zahtevami in odgovori nam daje prosto pot, zato je ogrodje bolj primerno za že izkušene razvijalce, ki

Node nekoliko bolje poznajo (četudi razvoj aplikacij poteka bistveno drugače, kot z Expressom ali brez ogrodja).

Uporaba ogrodja Koa se priporoča, kadar izdelujemo aplikacijo, ki ne temelji na uporabi brskalnika (zaradi neobstoja pripravljene podpore za usmerjanje) in kadar je hitro izvajanje ključnega pomena, saj Koa omogoča nekoliko hitrejše delovanje.

- Sails

MVC ogrodje SailsJs je dodatna plast, zgrajena nad ExpressJs, zato se, če si za ogrodje izberemo Sails, priporoča predhodnje znanje uporabe Expressa. Na njegovo uporabo bi se nahitreje navadili razvijalci, ki imajo izkušnje s Ruby on Rails, katerega MVC zasnovo (poleg dodane podpore za porebe modernih podatkovno usmerjenih APIjev) poskuša emulirati.

Kjer Express za ORM, torej interakcijo s PB MongoDB uporablja Mongoose, Sails uporablja Waterline, katerega velika prednost je fleksibilnost: koncept adapterjev. S uporabo teh lahko uporabljamo širok nabor različnih PB hkrati, ali brez težav preklapljamo med njimi (MySql, PostgreSql, MongoDB, Redis, …). Pri tem kodo, ki se tiče uporabe PB pišemo samo enkrat, z nekaj manjšimi popravki.

Sails se za uporabo priporoča, kadar razvijamo realnočasovne aplikacije, obdelujemo in hranimo velike količine podatkov (big data).

Zakaj Express in ne njegove alternative?

Express je od vseh naštetih ogrodij najstarejši in najbolj uveljavljen. Kot najbolj prepoznavno, skoraj standarizirano ogrodje ima trenutno tudi največ uporabnikov, njegove osnove smo spoznali tudi pri predmetu Razvoj Aplikacij Za Internet, kjer smo ga uporabili pri dveh vajah. Na spletu ne manjka člankov, nasvetov, opisov ter razne druge dokumentacije kar se tiče uporabe tega ogrodja. Seveda obstaja verjetnost, da ga katera od alternativ v prihodosti nadomesti, kot »go to« NodeJs ogrodje, toda večina teh se sklicuje na Express: ali ga nadgrajuje, ali pa poskuša odpraviti njegove napake, dobro ga je poznati v vsakem primeru.

Viri

Primerjava express, hapi in koa:

https://medium.com/@theomalaper.cognez/express-vs-koa-and-hapi-a2c65f949b78

Express template engines: https://github.com/expressjs/express/wiki#template-engines

Hapi plugins: https://hapi.dev/plugins/#universe

Opis uporabe funkcionalnosti Expressa:

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction

Primerjava Hapi in Express ogrodij: https://www.konstantinfo.com/blog/hapi-vs-express/

Alternative (MongoDB)

MySQL

Odprtokodna baza, ki deluje na osnovi sorodnikov (angl. relation). Je v zelo znanih "stackih" kot sta LAMP in XAMP, uporablja pa jo nekaj zelo znanih strani, npr. Facebook, Twitter, Youtube.

Podatki:

- **Avtor**: MySQL AB
- **Izdan**: Maj 1995
- **Napisan v**: C, C++

V primerjavi z MongoDB je fleksibilnejši in se uporablja pri bol zahtevnih aplikacijah.

Njegova osnova je sorodstvo med entitetami. Želimo jo uporabljati takrat ko so naši podatki v večini povezani, to pomeni, da ko se en spremeni, neposredno ne vpliva na otroka, čeprav ima otrok tudi drugačno vrednost.

Lasnosti

Obstaja odprtokodni MySQL Community server in lastniški Enterprise Server. V osnovi sta si enaka, razlika je pri vtičnikih, ki jih lahko naknadno naložimo Enterprise različici. Deluje na različnih operaciskih sistemih (Linux, Windows), uporablja sprožilce (npr. kaj se zgodi ob ustvarjanju vrstice), spremenljive poglede, to pomeni, da imajo različni uporabniki različne poglede in ne vpliva na poglede drugih uporabnikov, Unicode podpora.

Omejitve

Če uporabljamo druge motorje za shranjevanje (angl. storage engine), kot privzeti InnoDB, lahko pride do težav pri različnih funkcionalnostih, na to lahko naletimo že pri tujih ključih.

Backup

"mysqldump" je orodje, ki ga dobimo zraven obeh različic MySQL. Omogoča nam "backup" iz vseh načinov shranjevanja, ki ga MySQL podpira. MySQL entreprise različica ima poleg tega tudi svoj pripomoček za backup, ki ponuja naprednejše funkcionalnosti.

Oblak

MySQL lahko teče na oblaku, kot so Microsoft Azure, Amazon EC2, Oracle Cloud Infrastructure. Velikokrat oblačne storitve ponujajo MySQL kot storitev, to pomeni, da mi ne rabimo baze konfigurirat in vzdrževat, ampak jo uporabljamo zgolj za podatke.

PostgreSQL

Odprokodni relakcijski podatkovni model. Na začetku se je imenoval POSTGRES, ampak se je kasneje preimenoval v PostgreSQL, zato ker podpira SQL. V veliko pogledih je podoben kot MySQL.

Podatki:

- **Razvijalci**: PostgreSQL Global Development Group
- **Izdan**: Julij 1996
- **Napisan** v: C

Zmogljivost

Je široko uporaben in se uporablja v velikih sistemih, kjer sta pomembna hitrost branja in pisanja. Podpira veliko optimizacij, ki so dostopne le v komercialnih rešitvah, na primer skupno branje brez zaklepanja. Uporaben je predvsem v sistemih, ki izvršujejo kompleksne poizvedbe, zato se uporablja predvsem, kot analiza podatkov.

Varnost

Uporablja vloge (angl. roles), ki jih lahko dodeljujemo uporabniku. Ima podporo SSL za povezavo med strežnik-odjemalec, ki je kriptirana.

Podpora skupnosti

Ima zelo močno in aktivno skupnost, ki konsantno izboljšujejo in dodajajo lasnosti.

Sočasna podpora

Uporablja MVCC model, ki omogoča zelo veliko stopnjo sočasnosti. To pomeni, da lahko več uporabnikov hkrati dostopa do istih podatkov.

NoSQL

NoSQL pomeni, da ne podpira SQL lasnosti, oz. hkrati podpira tudi druge lasnosti, ki jih SQL ne. PostgreSQL podpira JSON zapis in druge NoSQL lasnosti, kot so XML podpora. Porpira tudi "indeksanje" JSON podatkov za hitrejpi dostop. Zaradi tega je bolj podoben MongoDB, kot je MySQL, saj je MongoDB NoSQL baza, MySQL ni.

RethinkDB

Je odprtokodna datotečna baza, razvita od podjetja z enakim imenom. Baza shranjuje JSON dokumente z dinamičnimi shemami. Prva različica je izšla julija 2019, napisan je v C++, Python, Java, JavaScript in Bash.

ReQL

RethinkDB uporablja ReQL poizvedbeni jezik. Podpira povezava entitet (table join), grupiranje in funkcije. Obstajajo tudi neuradne različice.

Primerjava z drugimi datotečnimi bazami

Glavna lasnost, ki se razlikuje od drugih je, da ima realno časovno posodablanje. To pomeni, da pri izdelavi zelo popularne aplikacije, ne bo prišlo do upočasnitve ali zaustavitve. Je zelo preprosta za nalaganje in učenje.

Popularnost

V zadnjih letih je vedno bolj popularna. Leta 2016 je bila na 46. mestu po popularnosti podatkovnih baz.

Zakaj MongoDB in ne njegove alternative?

MongoDB se nam je zdej najbolj primeren, saj potrebujemo bazo, ki je hitra, in hitro dostopna preko našega backenda. Če bi uporabili MySQL bi v vsakem kontrolerju morali pisati poizvedbe, pri MongoDB, pa to naredimo s predefiniranimi funkcijami, zato porabimo manj časa. Prednost je tudi v tem, da ne potrebujemo kompleksnejših relacij med našimi podatki in zato MongoDB najbolj zadostuje. Je tudi edina datotečna baza, za katero imamo predznanje.

Gostovanje aplikacije (Heroku)

Za gostovanje nodejs in angular smo se odločili za Heroku. Je oblačna storitev namenjena gostovanju aplikacij v različnih programskih jezikih. Je brezplačna, vendar je lahko tudi plačljiv, če izberemo kakšne plačljive vtičnike.

Samo storitev smo spoznali že v srednji šoli, saj smo tukaj gostovali naše projekte in to je eden od razlogov zakaj smo ga izbrali. Drug večji razlog je GUI in enostavnost vzpostavitve same aplikacije, čeprav smo naleteli na majhne težave.

Razdelitev dela

Delo smo si razdelili enakopravno in vseskozi komunicirali preko raznih kanalov. Prav tako smo si taske oz. cilje zastavili na JIRI in jih sproti izpolnjevali.

Domen Osojnik: Raziskava alternativ Angularju in priprava poročila, vzpostavitev Angular okolja, priprava storitev za pridobivanje podatkov iz zaledja, implementacija Google maps, prikaz podatkov praskanja, prikaz podatkov senzorjev, izdelava statističnega modela podatkov senzorjev, implementacija pospeškometra, raziskava Darknet okolja, priprava dataseta (označevanje slik), treniranje uteži, testiranje uteži (detekcija našega naučenega modela)

Alen Poklič: Raziskava alternativ Expressu, implementacija žiroskopa, kamere in uporabniškega vmesnika mobilne aplikacije, komunikacija med mobilno aplikacijo ter zalednim delom (shranjevanje senzorskih podatkov v združeno celoto), praskanje podatkov ter prenos in shranjevanje le teh, raziskava obstoječih tehnologij/implementacij algoritmov/virov za strojno učenje (yolo3+darknet+labelimg), označevanje svoje tretjine slik.

Gal Glogovšek: Raziskal alternative MongoDB, implementacija GPS, izdelava modelov zaledja, implementirati scrapanje ob samem zagonu zaledja in preprečevanje napak, komunikacija med spletno aplikacijo in zalednim delom (posiljanje vozenj-senzorski podatki, slike in scrape podatkov), označevanje slik za izgradnjo modela, raziskava in preizkus tehnologij pri strojnem učenju (tensorflow, darkflow), dodajanje zalednega dela v Heroku, da deluje brez napak, dodajanje spletne aplikacije v Heroku, da deluje brez napak.

Težave pri delu

Največ težav se je pojavilo pri predmetu ORV. Imeli smo ogromno težav pri implementaciji darkneta na windows okolju, na MAC OS okolju pa je trening vzel ogromno časa (1 ura za 2 iteraciji, potrebovali bi jih čez 10.000 za uspešen model). Zato smo se odločili poiskati alternative ali pa uporabiti kar že izdelane uteži. Z osnovnim modelom nam je uspela prepoznava stop znaka, lastne uteži pa niso delovale, prav tako uteži prometnih znakov najdene na spletu. Poizkusili smo ogromno stvari, vendar smo pri vseh naleteli na težave (nekompatibilne knjižnice z python 3.7, novejše knjižnice tensorflowa in kerasa ne podpirajo starejše kode, trening ponekod nemogoč na AMD grafični kartici (zahteva je bil CUDA, ki je omogočen le na NVIDIA karticah)).

```
C:\Python38\lib\site-packages\keras\backend\tensorflow_backend.py:3117: SyntaxWarr
  elif training is 0 or training is False:
WARNING:tensorflow:From C:\Python38\lib\site-packages\tensorflow\python\compat\v2_
Instructions for updating:
non-resource variables are not supported in the long term
2020-06-01 00:52:52.431692: I tensorflow/core/platform/cpu_feature_guard.cc:143] Y
2020-06-01 00:52:52.442817: I tensorflow/compiler/xla/service/service.cc:168] XLA
2020-06-01 00:52:52.446291: I tensorflow/compiler/xla/service/service.cc:176]   St
2020-06-01 00:52:52.450943: W tensorflow/stream_executor/platform/default/dso_load
2020-06-01 00:52:52.455048: E tensorflow/stream_executor/cuda/cuda_driver.cc:313]
2020-06-01 00:52:52.461594: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:
2020-06-01 00:52:52.464589: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:
Create YOLOv3 model with 9 anchors and 5 classes.
C:\Python38\lib\site-packages\keras\engine\saving.py:1136: UserWarning: Skipping l
  warnings.warn('Skipping loading of weights for '
C:\Python38\lib\site-packages\keras\engine\saving.py:1136: UserWarning: Skipping l
  warnings.warn('Skipping loading of weights for '
C:\Python38\lib\site-packages\keras\engine\saving.py:1136: UserWarning: Skipping l
  warnings.warn('Skipping loading of weights for '
C:\Python38\lib\site-packages\keras\engine\saving.py:1136: UserWarning: Skipping l
  warnings.warn('Skipping loading of weights for '
C:\Python38\lib\site-packages\keras\engine\saving.py:1136: UserWarning: Skipping l
  warnings.warn('Skipping loading of weights for '
C:\Python38\lib\site-packages\keras\engine\saving.py:1136: UserWarning: Skipping l
  warnings.warn('Skipping loading of weights for '
Load weights C:\Users\DOMEN\Desktop\TrainYourOwnYOLO\2_Training\src\keras_yolo3\yo
Freeze the first 249 layers of total 252 layers.
Traceback (most recent call last):
  File "C:\Python38\lib\site-packages\keras\callbacks.py", line 745, in __init__
    from tensorflow.contrib.tensorboard.plugins import projector
ModuleNotFoundError: No module named 'tensorflow.contrib'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "Train_YOLO.py", line 165, in <module>
    logging = TensorBoard(log_dir=log_dir_time)
  File "C:\Python38\lib\site-packages\keras\callbacks.py", line 747, in __init__
    raise ImportError('You need the TensorFlow module installed to '
ImportError: You need the TensorFlow module installed to use TensorBoard.
```

Slika : Problem tensorflow

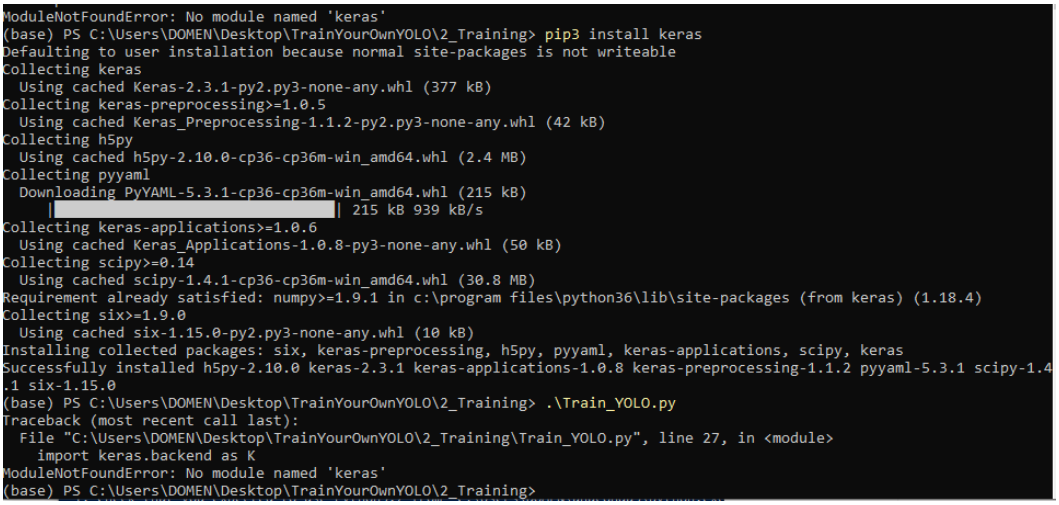
```
    name = prefix + '_' + str(K.get_uid(prefix))
  File "C:\Python38\lib\site-packages\keras\backend\tensorflow_backend.py", li
    graph = tf.get_default_graph()
AttributeError: module 'tensorflow' has no attribute 'get_default_graph'
```

Slika : Težava modula tensorflow

Nekatere težave nam je uspelo razrešiti (spreminjanje tensorflow in keras knjižnic). Preprosta rešitev bi bila namestitev starejših različic, vendar niso bile podprte na novejšem Pythonu:

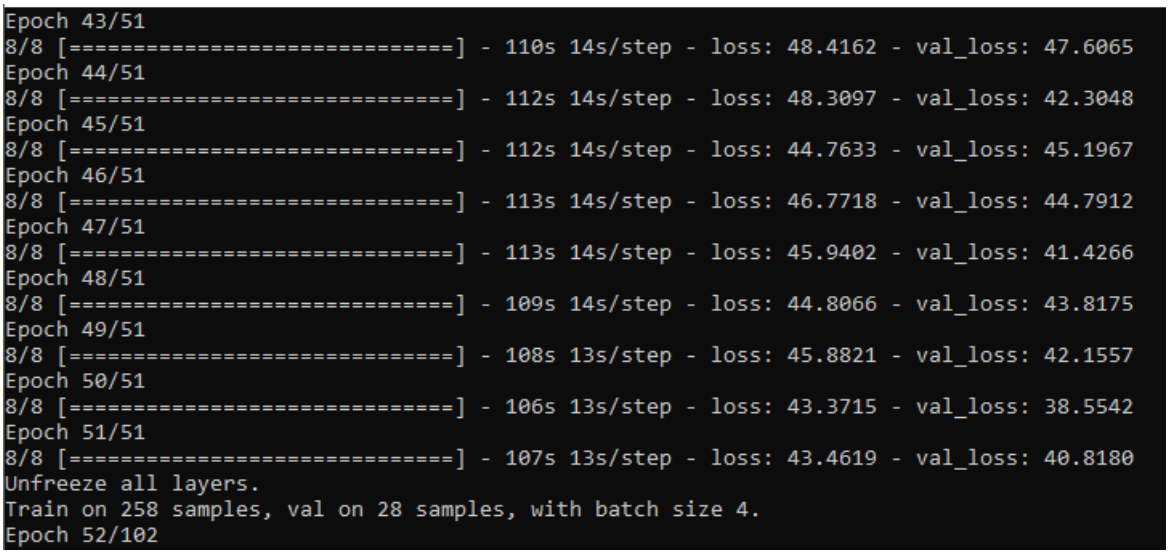
```
WARNING: Skipping tensorflow as it is not installed.
PS C:\Users\DOMEN\Desktop\TrainYourOwnYOLO\2_Training> pip install tensorflow==1.14.0
ERROR: Could not find a version that satisfies the requirement tensorflow==1.14.0 (from versions: 2.2.0rc1, 2.2.0rc2, 2.2.0rc3, 2.2.0rc4, 2.2.0)
ERROR: No matching distribution found for tensorflow==1.14.0
PS C:\Users\DOMEN\Desktop\TrainYourOwnYOLO\2_Training> pip install tensorflow==2.20
ERROR: Could not find a version that satisfies the requirement tensorflow==2.20 (from versions: 2.2.0rc1, 2.2.0rc2, 2.2.0rc3, 2.2.0rc4, 2.2.0)
ERROR: No matching distribution found for tensorflow==2.20
PS C:\Users\DOMEN\Desktop\TrainYourOwnYOLO\2_Training> pip install tensorflow==2.2.0
Collecting tensorflow==2.2.0
```

Slika : Stare verzije



Slika : Neprepoznavnost keras

Ustvarili smo še virtualno okolje na Anacondi s starejšo različico pythona ter namestili vse potrebne module. Na koncu je le uspelo in smo pričeli s treniranjem našega YOLO modela.



Slika : Treniranje YOLO