Slajd 1

Dobrý deň, vážená komisia, vítam vás na mojej prezentácii diplomovej práce. Mojou témou je vývoj súčastí webovej aplikácie na sémantickú anotáciu datasetov a mojím vedúcim je pán inžinier Miroslav Rác.

Slajd 2

Dobre takže na začiatok poviem predstavenie čo je to sémantika čo je to sémantická analýza a prečo je to potrebné a dôležité. Sémantika je odbor lingvistiky a logiky, ktorý sa zaoberá významom slov a viet. Sémantická anotácia je proces pridávania významu údajom prostredníctvom ich označovania relevantnými informáciami. Je to dôležité najmä v oblasti business intelligence, kde sa denne generuje obrovské množstvo údajov. A sémantická anotácia robí z týchto dát užitočne informácie.

Slajd 3

Úlohou mojej práce je implementovať webové prostredie ktoré zjednodušuje proces vytvárania sémantickej anotácie. Toto webové prostredie dovoľuje manažovanie a následne zobrazenie anotácii. Dovoľuje pridávať datasety a tieto datasety anotovať pomocou už existujúcich anotácii. Dôležité je aby aplikácia pracovala efektívne s dátami a pomáhala pri evaluácií AI modelov. Výslednú aplikáciu je potrebné nasadiť na cloud.

Slajd 4

Po analýze všetkých dostupných technológii a požiadaviek na aplikáciu boli vybraté nasledujúce technológie na frontend vue.js. Tailwind ako CSS framework. Primevue ako knižnica komponentov. Na backend nest.js. Ako databázu použijem grafovú databázu neo4j a ako relačnú databázu použijem PostgreSQL, a cloud provider pomocou ktorého túto aplikáciu nasadím do cloudu bude GCP google cloud platform s využitím firebase služieb.

Slajd 5

Pred implementáciou samotnej aplikácie som si vytvoril flow diagramy ktoré mi opisujú základnú funkčnosť mojej aplikácie. Taktiež som si vytvoril wireframy pre každú obrazovku mojej aplikácie avšak neukážem tu všetky wireframy aby som ušetril čas ukážem len jeden ukážkový konkrétne môžeme vidieť stránku určenú na anotáciu datasetu. Tieto wireframy sú oproti finálnemu produktu mierne zmenené. Tato zmena vychádza z toho, že počas vývoja sa menili požiadavky na zobrazenie.

Slajd 6

Na obrázku môžeme vidieť môj terajší model databázy v neo4j a integráciu so službami od firebase. Mame tu node pre dataset ktorý je pomocou vzťahu has column napojený na node column ktorý je napojený pomocou vzťahu annotated with k anotácii tato anotácia môže súvisieť z ďalšou anotáciu to je naznačene pomocou vzťahu related to. Node dataset si uchováva informáciu o tom aký používateľ ho vytvoril pomocou user uid a každý user ma svoju rolu uloženú vo firestore databáze. Údaje o datasete sú uložene aj v neo4j aj v firebase. Vidíme že vo firebase mame tabuľku users ktorá sa stará o autentifikáciu ktorá je naviazaná na firestore kde sú uložene role používateľov a informácie o datasete. Csv dataset je uploadutý do firebase storage a naviazaný pomocou download linku na firestore dataset.  
Taktiež tu môžeme vidieť postgresql databázový model kde v tabuľke dataset mame naviazaný firebase\_dataset\_id na dataset vo firebase a firebase\_user\_uid je naviazané na používateľa vo firebase. Taktiež môžeme vidieť, že dataset\_column je naviazane na na id datasetu. Vidíme že dataset\_column je naviazaný na anotáciu pomocou spájajúcej tabuľky column\_annotation a že anotácia má vzťah sama v sebe na parent annotation.

Slajd 7

Vidíme že používame viaceré databázy a keby do budúcna chceme pridať ďalšiu tak aby sme nemuseli meniť veľa kódu použili sme repository design pattern. Tu môžeme vidieť využitie repository design pattern vytvorili sme interface annotation service v ktorom sú definované všetky funkcie. Tento interface implementujú aj annotaionservicepostgresql aj annotations serviceneo4j controller pritom nevie ktorú databázu používa v module sa pomocou use factory vytvorí konkrétna inštancia ktorá je vybratá pomocou env premennej.

Slajd 8

Pre vizualizáciu boli použité dve knižnice prvou je neovis.js slúži na vizualizáciu dát pre neo4j a pripája sa priamo na neo4j databázu je nadstavba vis-network. Pre vizualizáciu postgresql databázy je použitý vis-network.

Slajd 9

Taktiež boli vytvorené seedre pre firebase neo4j a postgresql s mock dátami pre ukázanie funkcionality mojej stránky. Tieto datasety sú uložené a plne oanotované a zaoberajú sa tematikou e-comercu, reklamami a marketingom.

Slajd 10

Na tomto slajde môžeme vidieť diagram cloudovej infraštruktúry mojej aplikácie hore môžeme vidieť klienta ktorí sa pripojí na moju url a pomocou dns záznamu sa pripojí na moju front end aplikáciu ktorá je hosťovaná pomocou firebase hosting. Z tohto hosting sa posielajú requesty na back end čo sú v mojom prípade cloud functions ktoré sa vedia automaticky škálovať podľa trafiku avšak treba im nastaviť min a max. Tieto cloud funkcie pristupujú aj do databázy čo je v mojom prípade buď neo4j aura alebo postgre sql.

Slajd 11

Tu môžeme vidieť CI/CD diagram pipeline. Ktorý v mojom prípade funguje nasledovne. Keď používateľ urobí nejaké zmeny v kóde a pushne ich do github repozitára tak pomocou github action tento kód buildnem a deploynem to znamená deploynem novú verziu na firebase hosting a aj na cloud functions. Postup ako ma github actions tento deploy vykonať sa nachádza v yaml súbore kde sú konkrétne popísané kroky. Deploy na firebase hosting sa stane pomocou príkazu firebase deploy. A na cloud functions pomocou firebase deploy –only functions.

Slajd 12

Na tomto slajde môžeme vidieť vy listovane všetky moje dostupné cloud functions môžeme tu vidieť viacero prístupov mám tu jednu veľkú monolitickú aplikáciu s názvom api kde je všetka funkcionalita. Ale mam aj vytvorené mikroservici to znamená tuto monolitickú aplikáciu som rozkúskoval podľa funkcionality pre lepší výkon do mikroservic.

Slajd 13

Dobre teraz sa poďme pozrieť na momentálny vizuálny stav mojej aplikácie pomocou screenshotov stránok neukážem všetky len tie najhlavnejšie. Na tomto screene môžeme vidieť stránku hľadania datasetu. Na ľavo je menu pre navigáciu ktoré listuje všetky dostupné podstránky stránky. Vpravo mame tabuľku s momentálne dostupnými nahratými datasetmi v ktorej sa da vyhľadávať a sortovat podľa každého stĺpca . Taktiež máme aj tlačidla pre upload csv datasetu a možnosť uploadnúť dataset plus jeho anotácie. Taktiež mame možnosť dataset zmazať.

Slajd 14

Tu môžeme vidieť stránku seleknutneho datasetu keďže táto stránka je dlhá rozdelil som ju na dva screenshoty. Hore mame sticky header s tlačidlami jedno ukladá zmeny ktoré sme vykonali druhé otvára dialog pre manažment anotácii. Vieme pomocou multiselectu si vybrať ktoré stĺpce chceme mat zobrazene v tabuľke. Mame tu tabuľku ktorá ukazuje hodnoty v datasete vieme v tejto tabuľke aj vyhľadávať a sortovat ju. Máme dve tlačidla Vieme tento dataset exportnúť ako csv alebo exportnúť jeho anotácie v json formáte. Taktiež vieme zmeniť popis tohoto datasetu zmeniť jeho meno.

Slajd 15

Na tomto slajde môžeme vidieť druhu časť stránky selectnuteho datasetu. Tu môžeme vidieť už samostatne stĺpce porozdeľovane do kariet užívateľ môže pridať stlpcu popis. Vidí akými anotáciami už bol oanotovaný a po kliknutí na tuto anotáciu je presmerovaný na listovanie anotácii. Pomocou selectu môže vybrať anotáciu. Táto anotácii sa mu zobrazí v stromovej štruktúre vľavo aby videl vzťahy medzi anotáciami a v pravo pomocou knižnice neovis alebo vis-network v grafovej forme s ktorou môže manipulovať. Pri oboch zobrazenia a prejdený na konkrétnu hodnotu mu vyskočí popup z bližšími informáciami o konkrétnej anotácii. Taktiež je možné kliknúť na tlačidlo suggest annotation ktoré navrhne anotáciu ktorá by sa tu mohla hodiť pomocou analýzy všetkých datasetov ktoré boli doteraz nahraté a oanotované.

Slajd 16

Na tomto slajde môžeme vidieť stránku so zoznamom anotáciu používateľ má dostupnú tabuľku so všetkými anotáciami a vie si zobraziť konkrétnu anotáciu ktorá ho zaujíma pomocou selectu a zobrazí sa mu grafová reprezentácia ako aj stromová štruktúra. Po kliknutí na tlačidlo management of annotations sa mu otvorí dialog s pridaním anotácie.

Slajd 17

Na tomto slajde môžeme vidieť stránku pridania anotácie používateľ pridá jej meno skratku popis a zvoli či je k nejakej už existujúcej anotácii príbuzná tento krok je optional. Následne ju pridá pomocou tlačidla add annotation. Anotáciu môže zmazať tým že ju vyberie a pomocou tlačidla delete annotation ju zmaže. Alebo môže vybrať parent a childa a tieto anotácie spojiť pomocou tlačidla connect annotation.

Slajd 18

Taktiež prebehol user testing tu môžeme vidieť dva scenáre ktoré som si pripravil prvý kontroloval manažment anotácii to znamenalo vytvoriť dve anotácie spojiť ich a následne zmazať. Mal som 10 testerov čo boli moji spolužiaci trvalo to 169 sekúnd v priemere medián bol 179 a standard deviation bol 49. Druhy scenár bol manažment datasetov to znamená užívateľ mal pridať dataset zmeniť mu meno popis a oanotovat 2 stĺpce 2 anotáciami a pridať im popis a následne tento dataset zmazať. To priemerne trvalo 188 sekúnd medián bol 192 standard deviation bol 40. Pri user testingu sa našli nejaké bugy tie boli fixnuté a spätná väzba od testerov bola zaznamenaná a niektorá aj použitá.

Slajd 19

Na grafe vidíme endpointy a čas koľko trvalo spracovať požiadavku. Zanalyzoval som nalepíš región na nasadenie mojej aplikácie vidíme že medzi medzi regiónmi v Európe nie je až taký veľký rozdiel avšak najlepší sa ukázal región Europe-west-3. Dôvodom je to že je nám najbližšie.

Slajd 20

Taktiež som zanalyzoval monolit vs microservice vidíme že microservice sú trochu rýchlejšie ako monolit. Dôvodom toho je že sme jednu veľkú aplikáciu rozkúskovali do viacerých malých čo zlepšilo výkon.

Slajd 21

Z analyzoval som ako pridanie pamäte cloud funkcie zvýši výkon môžeme vidieť že čim viac pamäte som dal tým sa výkon zlepšil avšak okolo 1GB už toto pridanie na výkone bolo zanedbateľne. Dôvodom zvýšenia výkonu je že cloud funkcia mala dostupnú väčšiu pamäť a gcp taktiež spolu s pamäťou škáluje aj výkon cpu ktorí ma tato cloud funkcia k dispozícii.

Slajd 22

Ďalej som porovnal výkon neo4j versus postgresql. Porovnal som výkon pre 100 1000 10000 100000 anotácii. Sústredil som sa hlavne na operácie ktoré pracovali s anotáciami. Vidíme že postgresql je lepšia voľba pretože je rýchlejšia vo všetkých endpoinotch. Zmena nastáva až pri 100000 anotáciách kde sa neo4j stave rýchlejšou voľbou pre endpointy getExport getannotations. Avšak pre importTesting je stále pomalšia. Dôvodom prečo sú tieto endpointy rýchlejšie je pretože neo4j efektívnejšie zvláda zložíte vzťahy a veľké objemy údajov a dôvodom že import je pomalší je pretože neo4j ma väčšiu réžiu na ukladanie dát ako posgresql. To znamená pre bežné použitie je postgresql lepšou voľbou.

Slajd 23

Takže vieme že pre bežné použitie je lepší postgresql. Pozrel som sa aj na cenu nasadenia hore v prvej tabuľke môžeme vidieť najlacnejšiu možnosť ako nasadiť neo4j a postgresql pričom ak nám ide o cenu postgresql vieme nasadiť veľmi lacno. V druhej tabuľke môžeme vidieť približne rovnaké výkony opäť vidíme že postgresql je na nasadenie tou lacnejšou variantnou.

Slajd 24

Takže rekapitulácia toho čo som spravil. Bol vytvorený databázový model wireframy flow diagram. Bola vyvrtená webová aplikácia pre manažment datasetov na sémantickú anotáciu. Bola vytvorená ci/cd pipeline a aplikácia bola nasadená na cloud. Prebehol user testing. Prebehla analýza najlepšieho regiónu pre nasadenie. Prebehlo porovnanie monolitu vs microservice. Prebehla analýza výkonu na základe pamäte. V poslednom rade boli porovnane databázy neo4j a postgresql.