UVOD

Što je igrifikacij?. Što je cilj ovog rada? Što je napravljeno/napisano u ovom radu?

Pregled rada i područja

Igrifikacija nastave

Generativna umjetna inteligencija

Motivacija

Postojeć i slična riješenja

Pregled zahtjeva

Funkcionalni

nefunkcionlani

# Arhitektura sustava i korištene tehnologije

Arhitektura sustava se kao i većina današnjih servisa bazirana je na klijent-poslužitelj arhitekturi (eng. Client-server architecture). Preko klijenta se odvija interakcija s korisnikom te se inicijaliziraju zahtjevi, server ih obrađuje i dohvaća po potrebi podatke iz baze te ih prosljeđuje natrag klijentu koji ih prikazuje korisniku.

Za komunikaciju između komponenata korišten je REST API (eng. REpresentational State Transfer Application Programming Interface). API je mehanizam koji omogućuje aplikaciji ili usluzi pristup resursu unutar druge aplikacije, usluge ili baze podataka, dok se REST odnosi na stil i skup arhitektonskih ograničenja. Zahtjevi se šalju i primaju u JSON formatu, a koristi se protokol HTTP (Hypertext Transfer Protocol) i njegove metode poput GET, POST, PUT i DELETE za standardne funkcije čitanja, stvaranja, ažuriranja i brisanja zapisa (eng. CRUD – create, read, update, delete).

REST se sastoji od nekoliko principa dizajna:

* Bez stanja (eng. Statelessness), što znači da se na serverskoj strani ne pohranjuju nikakve sesije i stanja. Zbog toga svaki zahtjev treba sadržavati sve informacije potrebne kako bi se obradio
* Predmemoriranje (eng. Cacheability), radi poboljšanja performansi, odgovori servera trebaju biti označeni kao predmemorirajući ili nepredmemorirajući
* Jedinstveno sučelje, potrebno je osigurati da se informacije prenose u standardiziranom format
* Razdvajanje klijenta i servera, gdje oboje rade nezavisno jedan o drugome
* Slojeviti sustav, u komunikaciji može postojati niz različitih posrednika

Odabrani tehnološki stog (eng. Tech stack) korišten za izradu aplikacije je tako zvani PERN stog. Naziv dolazi od prvih slova njegove glavne 4 komponente: PostgreSQL, Express, React and Node.js. Na slici je vizualna reprezentacija klijent poslužitelj arhitekture i protok podataka u PERN stog. Više o samim komponentama i njegovim korištenju slijedi u idućim pod poglavljima o klijentu, poslužitelju i bazi.

A diagram of a company

AI-generated content may be incorrect.

<https://medium.com/nerd-for-tech/client-server-architecture-explained-with-examples-diagrams-and-real-world-applications-407e9e04e2d1>

<https://www.geeksforgeeks.org/rest-api-introduction/>

<https://www.ibm.com/think/topics/rest-apis#:~:text=A%20REST%20API%20is%20an,to%20connect%20distributed%20hypermedia%20systems>.

<https://www.redhat.com/en/topics/api/what-is-a-rest-api>

klijent

poslužitelj

integracija AI

Jedan od ciljeva ovog diplomskog rada bila je integracija generativne umjetne inteligencije u aplikaciju. Iskorištena je kao alat za generiranje pitanja što uvelike ubrzava i olakšava učiteljima i profesorima proces kreiranja kvizova. U ovom podpoglavlju objasniti ćemo kako smo pristupili modelima umjetne inteligencije i integrirali ih u arhitekturu sustava, a u idućem poglavlju o opisu sustava donosimo konkretne primjena u aplikaciji.

Korištena je platforma Together.ai. Radi se o platformi za distribuirano izvođenje velikih jezičnih modela, omogućuje pristup open-source modelima putem jednostavnog API-ja, bez potrebe za lokalnim izvođenjem modela na vlastitoj infrastrukturi. Podržava treniranje, fine-tuning i pokretanje sučelja na modelima umjetne inteligencije i to brzo, pouzdano i cjenovno isplativo. Glavni razlozi zbog kojeg smo odlučili istražiti i iskoristiti ovu platformu je jednostavnost i brza integracija, podrška za velik broj modela te sigurnost i privatnost putem end-to-end enkripcije i sigurnosnih protokola.

Za korištenje platforme potrebno je napraviti korisnički račun, pri čemu se dobiva API ključ za povezivanje i početni kredit od 1 USD što je bilo i više nego dovoljno za potrebe izrade i testiranja aplikacije.

const {Together} = require('together-ai')

const client = new Together({ apiKey: process.env.TOGETHER\_API\_KEY });

const systemPrompt = `Tvoj posao je generirati kviz pitanja za učenike ${grade\_level}. razreda ${school\_level} škole. Slijedi ova pravila:

        - Generiraj ${count} jedinstvenih pitanja o "${topic.name}"

        - Koristi jednostavan i jasan Hrvatski jezik

        - Pruži 4 moguća odgovora po pitanju

        - Samo jedan točan odgovor po pitanju

        - Ne ponavljaj već postavljena pitanja, ali ih iskoristi za inspiraciju: ${existingQuestions.rows.map(q => q.question).join(', ') || 'nema'}

        - Formatiraj odgovor kao:

        Pitanje: [pitanje]

        Odgovori: [odgovor1], [odgovor2], [odgovor3], [odgovor4]

        Točan odgovor: [broj 1-4]

        - Bez dodatnog teksta ili objašnjenja

        - Ne koristi oznake za formatiranje

        `;

const userPrompt = `Opis teme: ${topic.description || 'Bez dodatnog opisa'}`;

const response = await client.chat.completions.create({

            model: 'meta-llama/Llama-3-70b-chat-hf',

            messages: [

                { role: 'system', content: systemPrompt },

                { role: 'user', content: userPrompt }

            ],

            temperature: 0.7,

            max\_tokens: 1000

        });

Odsječak koda 1 prikazuje kako se povezuje s platformom Together.ai te koje smo instrukcije dali modelu. systemPropmt je uputa modelu kako treba odgovarati odnosno što treba generirati, daje mu se instrukcija kako je on generator kviza, a parametri poput razine obrazovanja (osnovna škola, srednja škola, fakultet…) i razreda preuzimaju se iz baze podataka, a unosi ih profesor pri kreiranju teme kviza, isto kao i opcionalni userPropt što su zapravo dodatne uputa za generiranje koje može zadati profesor. Iz baze se dohvaćaju i već postojeća pitanja na tu temu kako nebi bilo ponavljanja. Model također dobiva upute kako točno formatirati odgovor, što nam je jako važno za daljnju obradu, slanje na klijentsku stranu i prikaz korisniku. Možemo primijetiti i parametar temperature koji je postavljen na 0.7, u kontekstu generativnih jezičnih modela taj parametar kontrolira razinu varijabilnosti odgovora koje model daje. Vrijednost 0.7 predstavlja ravnotežu između preciznosti i raznolikosti.

Together.ai na raspolaganju ima preko 200 open-source modela. U sklopu ovog rada testirali smo nekoliko modela na platformi kako bismo usporedili njihovu učinkovitost za zadani slučaj upotrebe, a to je generiranje pitanja za kviz.

TO DO: malo opisati modele koje smo odabrali za testiranje

Testiranje je provedeno na temi 'Povijest starog Egipta' za 1. razred srednje škole, a modeli su evaluirani prema kriterijima brzine generiranja (vrijeme od zahtjeva do odgovora u sekundama), cijene generiranja (u USD za 1 milion tokena) i kvalitete i točnosti generiranih pitanja. Svaki model testiran je dvaputa: s dodatnim opisom teme i bez dodatnog opisa. S obzirom da je instrukcija modelu da generira po 5 pitanja, kvaliteta generiranih pitanja se ocjenjuje na skupu od 10 pitanja, a kriteriji ocjenjivanja su prikazani u tablici 1.

Rezultati za odabrane modele prikazani su u tablici 2.

|  |  |  |
| --- | --- | --- |
| ocijena | Jezična ispravnost i jasnoća | Relevantnost |
| 5 | Sva pitanja su jezično ispravna i potpuno jasna. Profesor nema potrebe za doradama; pitanja su odmah spremna za korištenje | Sva pitanja i ponuđeni odgovori potpuno odgovaraju zadanoj temi i razini obrazovanja. Profesor nema potrebe za doradama |
| 4 | Manji broj gramatičkih grešaka, ali rečenice su i dalje jasne i razumljive. Profesor treba ispraviti jednu do dvije riječi ili sitne gramatičke pogreške, ali pitanja su u osnovi upotrebljiva bez većih izmjena. | Većina pitanja i odgovora je relevantna, uz manji broj iznimki. Profesor treba prilagoditi ili zamijeniti jedan do dva odgovora ili pitanja koja odstupaju od teme ili razine. |
| 3 | Veći broj gramatičkih grešaka koje blago narušavaju jasnoću i razumijevanje pitanja. Profesor treba ispraviti nekoliko riječi i možda preformulirati jedno do dva pitanja ili odgovora kako bi bila potpuno razumljiva. | Oko polovice pitanja/odgovora je relevantno, ostalo odstupa od teme ili razine. Profesor treba preformulirati ili zamijeniti nekoliko pitanja i odgovora kako bi svi bili u skladu s temom i razinom. |
| 2 | Značajne jezične greške, pitanja su teško razumljiva. Profesor mora prepraviti većinu pitanja; ispravljati rečenice, zamijeniti ili preformulirati više odgovora, kako bi pitanja bila upotrebljiva. | Većina pitanja nije relevantna za zadanu temu ili razinu. Profesor mora napraviti veće izmjene – većinu pitanja i odgovora treba zamijeniti ili prilagoditi. |
| 1 | Model nije uspio generirati upotrebljiva pitanja. Profesor mora urediti I prepraviti gotovo svako ili sva pitanja. | Gotovo nijedno pitanje nije relevantno. Pitanja nisu upotrebljiva. Profesor mora samostalno osmisliti nova pitanja. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model dodati proizvođača! | Brzina (s) ($ za 1 milion tokena) | Cijena | Jezična ispravnost i jasnoća | Relevantnost |
| Meta Llama 3.3 70B Instruct Turbo | 2.45 | 0.88 | 5 | 5 |
| Meta Llama 3.3 70B Instruct Turbo Free | 4.92 | 0 | 5 | 4 |
| Qwen QwQ-32B | 7.07 | 1.20 | 2 | 4 |
| Mistral (7B) Instruct v0.3 | 3.39 | 0.20 | 3 | 3 |
| Gemma-2 Instruct (27B) | 3.95 | 0.80 | 4 | 5 |
| Refuel LLM V2 Small (not small nije radio) | 0.78 | 0.20 | 1 | 1 |
| Marin 8B Instruct | 1.02 | 0.18 | 1 | 1 |
| Meta Llama 4 Maverick Instruct | 17.02 | 0.27 / 0.85 | 5 | 5 |
| R1 1776 (limit) | 9.67 | 3.00 / 7.00 | 5 | 4 |

Najbolje rezultate pokazali su modeli Meta Llama 3.3 70B instruct turno i također Metin Llama 4 Maverick Instruct, s time da je prvom potrbeo značajnije manje vremena za generiranje. Modeli poput R1 1776, Mistral Instruct i Gemma-2 instruct također su se pokazali solidnima, te bi čak posljednji mogli izdvojiti kao najboljeg u pogledu relevantnosti sadržaja. S druge strane modeli poput Refuel i Marin nisu pružili zadovoljavajuće rezultate, sto se možda može opravdati pošto vjerojatno nisu trenirani za hrvatski jezik te bi testovi na engleskom možda pokazali drugačije rezultate.

TO DO: opet prekontrolirati rezultate i naosati zaključak potpoglavlja

https://www.byteplus.com/en/topic/416107?title=is-together-ai-reliable-a-comprehensive-analysis-of-ai-collaboration-platform

baza

Podatke smo spremali u relacijsku bazu podatak PosgreSQL s alatom pgAmin, a korišten je jezik SQL.

Na slici 1 je prikazan relacijski model baze podataka izvezen iz paAdmin-a, a u nastavku su detaljnije opisani relacije unutar baze podataka.

Korištenje aplikacije / opis sustava

Prijava i registracija

Početna stranica

3 modaliteta igre

-

-

-

AI u igrama

Administracijsko sučelje:

Testiranje aplikacije

Rezultati testiranja ?

Diskusija i analiza ?

Zaključak