dev-club-prezentacija

Od teorije do produkcije

Automatizacija studentske zajednice "Dev Club" pomoću Discord bota

Lovro Posarić, Tara Nikolić, Dijana Plantak Vukovac
Fakultet organizacije i informatike, Sveučilište u Zagrebu

Kontekst i izazovi

Povezivanje "Edukacije 4.0" i aktivnog učenja kroz digitalne prostore

Motivacija i "Dev Club"

- **edukacija 4.0:** tradicionalno obrazovanje često kaska za napretkom IT industrije. Interdisciplinarnost i digitalne kompetencije su izrazito važne.
- **"Dev Club":** osnovan kao studentski inkubator znanja posvećen razmjeni znanja u programiranju, Game Devu i sigurnosti.
- **aktivnosti:** vršnjačko mentorstvo, Game Jam natjecanja i praktične radionice izvan standardnog kurikuluma.
- **problem:** fragmentirana komunikacija na fakultetu naglasila je potrebu za centraliziranom, neformalnom platformom.

Zašto baš discord?

Ograničenja alternativa

Slack: besplatna verzija ograničava povijest poruka na 90 dana. Time se nepovratno gube vrijedne tehničke rasprave i resursi, a plaćeni modeli blokiraju širenje kluba.

MS Teams: alat s dubokom integracijom u Office 365 okruženje, no stvara previše formalno okruženje koje ubija studentsku spontanost i intrinzičnu motivaciju.

Discord kao "treće mjesto"

Dostupnost: besplatna platforma s neograničenom poviješću. Većina studenata već posjeduje račun, čime je tehnička i psihološka barijera za ulazak svedena na minimum.

Arhitektura: Trajno otvoreni glasovni kanali potiču spontanu interakciju, savršeno oponašajući sociološki koncept neutralnog i sigurnog "trećeg mjesta".

Priča iz prakse:
Brucoš Marko

Problem fragmentacije

Zamislimo Marka, studenta prve godine. Na predavanjima upija teoriju, ali kada prvi put pokuša samostalno napraviti aplikaciju, javljaju se greške koje ne razumije. Osjeća se izolirano, uči napamet, i gubi motivaciju.

Ulazak u zajednicu

Marko dobiva pozivnicu za Discord server "Dev Cluba". Bot ga odmah pozdravlja, dodjeljuje mu ulogu "Intern" i usmjerava prema bazi resursa i forumu.

Marko više nije sam u svom procesu učenja.

Vršnjačko mentorstvo na djelu

- tijekom svog prvog Game Jam natjecanja, Marko zapinje na klasičnom "Double/triple jump" problemu u kodu (lik mu beskonačno skače).
- umjesto da odustane, otvara novu temu na Dev-forumu. U manje od 10 minuta, stariji kolega uskače u privremeni glasovni kanal.
- zajedno debugiraju kod. Marko ne dobiva samo gotovo rješenje, već ga se kroz *scaffolding* vodi do samostalnog zaključka.
- za postavljanje kvalitetnog pitanja i interakciju, bot Marku dodjeljuje XP bodove. On napreduje iz Interna u **"Trainee"**-a.

Ključni scenariji upotrebe

Mentoriranje

Poticanje javnog vršnjačkog učenja. Umjesto da svatko uči izolirano, studenti grade sustav zajedničkog učenja gdje i mentor utvrđuje svoje znanje.

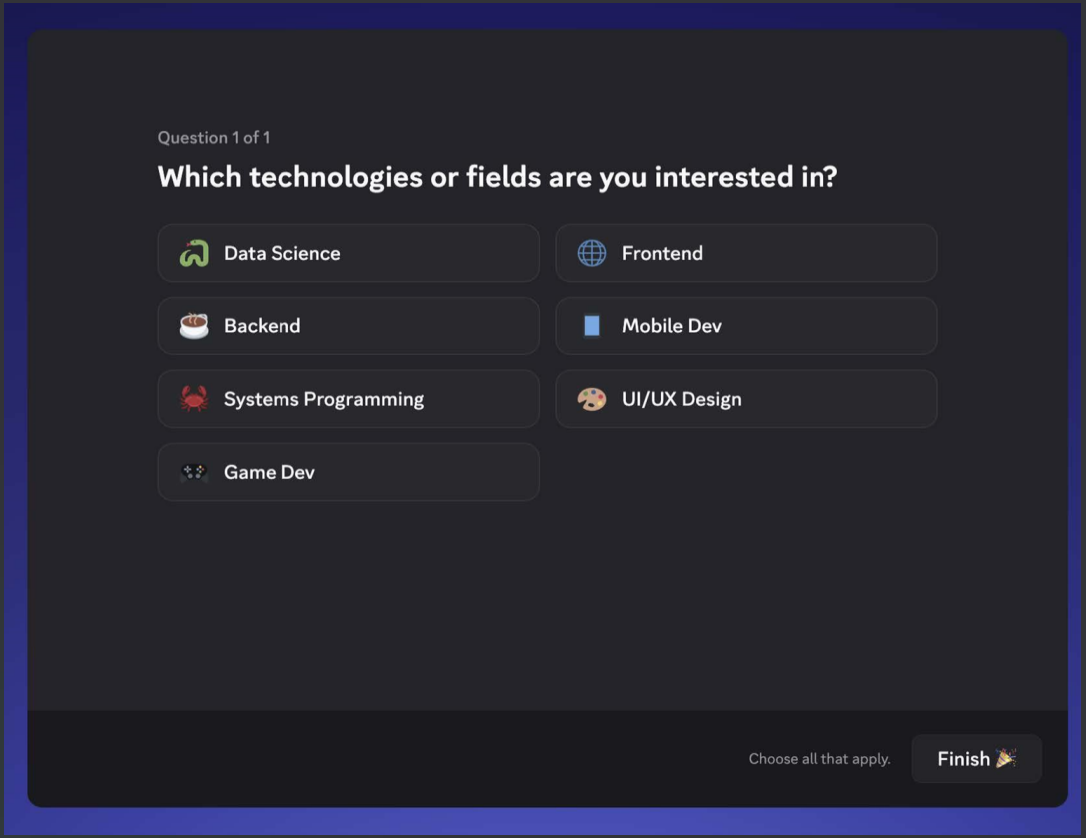
Baza znanja

Korištenje forum kanala za strukturirane tehničke rasprave o kodu, pretvarajući prolazna pitanja u trajnu i lako pretraživu bazu programerskih rješenja.

Koordinacija

Automatizirani sustav najava za radionice, sastanke i natjecanja. Kognitivno rasterećenje za organizatore uz borbu protiv "digitalnog šuma".

Personalizirani *onboarding*



Identitet i verifikacija

Prvi susret korisnika sa serverom dizajniran je kao vođeni proces. Korisnik bira područja interesa (Backend, Data Science, UI/UX...). Bot mu trenutno dodjeljuje adekvatne uloge.

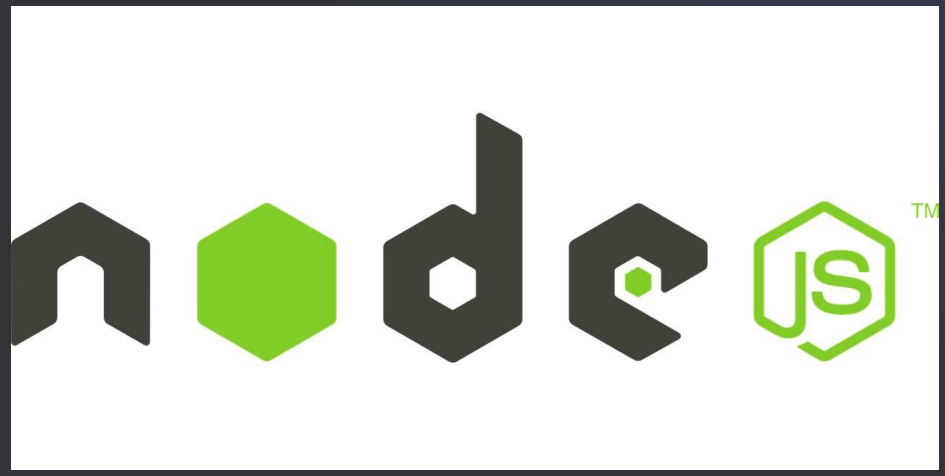
Usmjeravanje korisnika

Nakon odabira, bot stvara automatiziranu *to-do* listu: pregledaj bazu znanja, traži pomoć, provjeri nadolazeće događaje. Ovi koraci znatno smanjuju vrijeme potrebno da se dođe od pasivnog promatrača do aktivnog sudionika.

Tehnološki okvir bota

Node.js i Discord.js

Arhitektura vođena događajima (eng. *event-driven*) u **Node.js**-u idealna je za efikasan asinkroni rad bota. **Discord.js** pouzdano upravlja pomoću Gateway WebSocket i REST API-ja.



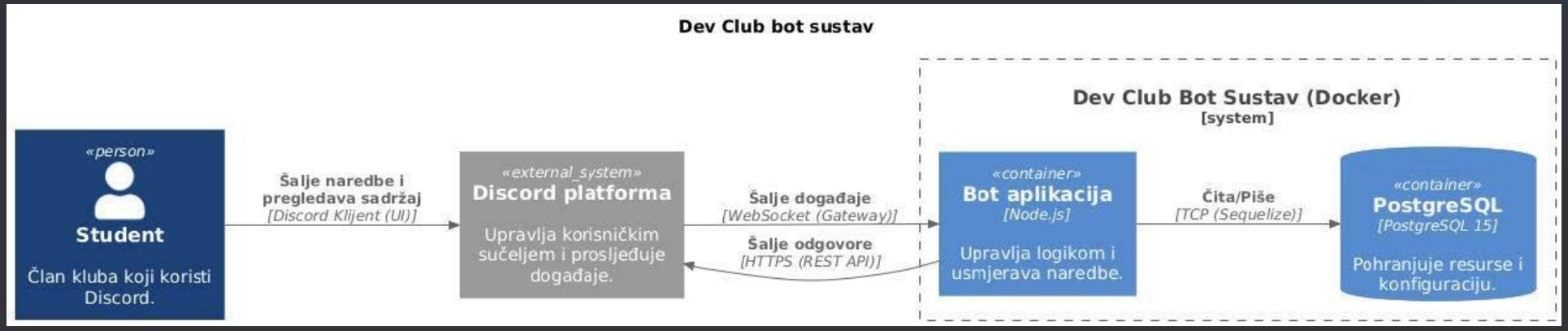
PostgreSQL i Sequelize

PostgreSQL je odabran zbog svog MVCC modela, koji osigurava konkurentno čitanje i pisanje podataka.
Sequelize ORM štiti od SQL injekcija i sakriva pozadinske detalje same baze.

Docker kontejnerizacija

Cjelokupno rješenje pakirano je u izolirani Docker kontejner čime se eliminiraju problemi s verzijama i omogućuje deploy na bilo koju infrastrukturu.

Arhitektura sustava



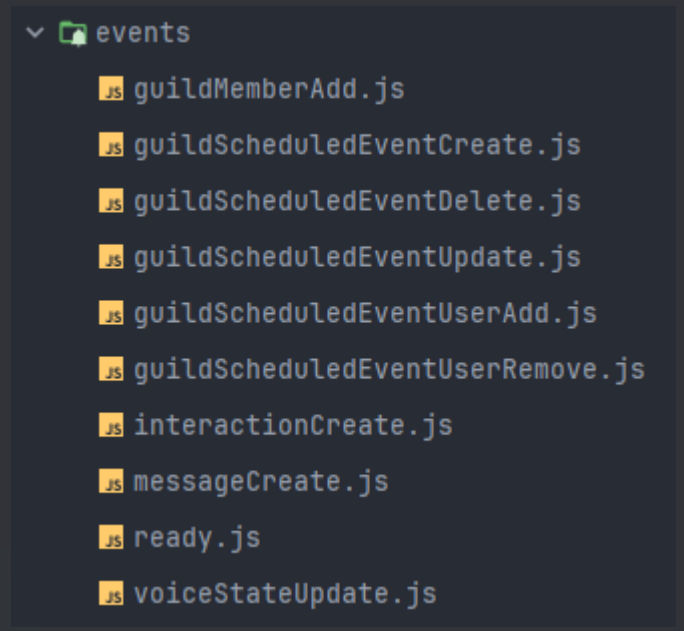
Arhitektura sustava

- **bez "god objecta":** potpuna eliminacija masivnih i neodrživih switch/if struktura. Logika svake komande je u zasebnom modulu.

```
client.on(Events.InteractionCreate,  listener: async interaction : ...  => {
    try {
        if (interaction.isButton()) {
            const { customId } = interaction;

            if (customId === 'create_mplus_listing') {
                await mplusHandler.handleCreateListing(interaction);
            } else if (customId.startsWith('mplus_join_')) {
                await mplusHandler.handleMplusJoin(interaction);
            } else if (customId.startsWith('mplus_leave_')) {
                await mplusHandler.handleMplusLeave(interaction);
            } else if (customId.startsWith('mplus_close_')) {
                await mplusHandler.handleMplusClose(interaction);
            } else if (customId === 'roster_find_player') {
                await buttonHandlers.handleRosterFindPlayer(interaction);
            } else if (customId === 'roster_log_invite') {
                await buttonHandlers.handleRosterLogInvite(interaction);
            }
        } else if (interaction.isStringSelectMenu()) {
            const { customId, values } = interaction;
            const selectedValue = values[0];

            if (customId === 'mplus_dungeon_select') {
                await mplusHandler.handleDungeonSelect(interaction);
            } else if (customId === 'mplus_setup_class') {
                await mplusHandler.handleMplusSetupClass(interaction);
            } else if (customId === 'mplus_setup_spec') {
                await mplusHandler.handleMplusSetupSpec(interaction);
            } else if (customId.startsWith('mplus_join_class_')) {
                await mplusHandler.handleMplusClassSelect(interaction);
            }
```

Arhitektura sustava

- **dinamički događaji:** bot skenira foldere repozitorija i automatski registrira *event listenere*, što značajno olakšava odvajanje odgovornosti i razvoj u timu.




```
module.exports = {
    name: Events.GuildMemberAdd,
    once: false,

    async execute(member) : Promise<void> {
        await saveNewUserToDb(member);

        await sendWelcomeMessage(member);
    },
};
```

Arhitektura sustava

- **slash commands (/reminder, /mentor find, ...):** oslanjanje na Discordov *Application Commands* model, a ne na parsiranje običnog teksta

```
module.exports = {
    data: new SlashCommandBuilder()
        .setName('reminder')
        .setDescription('Reminds you at a specific time.')
        .addStringOption( input: option : SlashCommandStringOption =>
            option.setName('when')
                .setDescription('When to remind you (e.g., "in 10 minutes", "tomorrow at 5pm")')
                .setRequired(true))
            .addStringOption( input: option : SlashCommandStringOption =>
                option.setName('message')
                    .setDescription('The reminder message')
                    .setRequired(true)),

    async execute(interaction) : Promise<any | undefined> {
        try {
            const whenInput = interaction.options.getString('when');
            const messageInput = interaction.options.getString('message');

            const parsedDate = parseDateInput(whenInput);
```

Arhitektura sustava

- **hibridni uzorci:** implementiran hibrid između uzoraka *Registry, Command* i *Strategy* za modularno usmjeravanje korisničkih interakcija (gumbi, izbornici, modalni okviri, autocomplete naredbe).

```
module.exports = {
    menus: [
        { matches: isExact( target: 'confirm_topic_selection'), execute: mentorHandlers.handleConfirmSearch },
        { matches: isExact( target: 'handle_unassign_selection'), execute: mentorHandlers.handleUnassignConfirm }
    ],
    buttons: [
        { matches: startsWith( prefix: 'mentor_action_'), execute: mentorHandlers.handleMentorActionMenu }
    ],
    modals: [
        { matches: startsWith( prefix: 'modal_search_mentor'), execute: mentorHandlers.handleModalSearch },
    ]
};
```

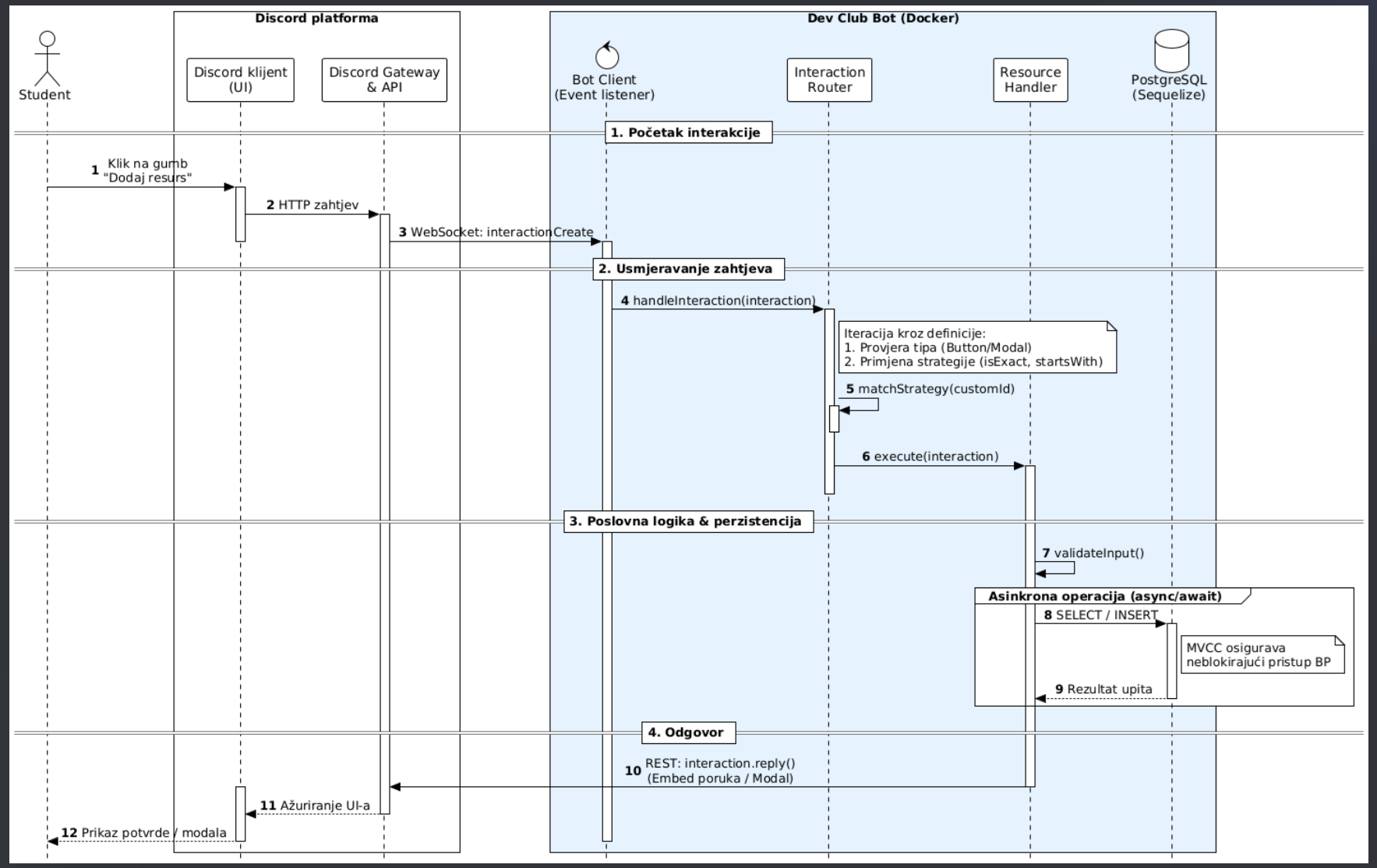
Arhitektura sustava

- **hibridni uzorci**

```
async function routeInteraction(interaction, strategyList, context, explicitId : null  = null)
    const idToMatch : never  = explicitId || interaction.customId;

    if (!idToMatch) return false;

    const strategy = strategyList.find(s => s.matches(idToMatch));

    if (strategy) {
        await strategy.execute(interaction, context);
        return true;
    }
    return false;
}
```

```
module.exports = {
    name: Events.InteractionCreate,

    async execute(interaction) : Promise<void>  {
        const context : {client: any, config: ...  = {
            client: interaction.client,
            config: config
        };

        try {
            if (interaction.isChatInputCommand()) {
                const command = interaction.client.commands.get(interaction.commandName);
                if (command) await command.execute(interaction);

                return;
            }

            if (interaction.isAutocomplete()) {
                const focusedOption = interaction.options.getFocused(true);
                const optionName = focusedOption.name;
                await routeInteraction(interaction, strategies.AUTOCOMPLETE, context, optionName);

                return;
            }

            let handled : boolean  = false;

            if (interaction.isButton()) {
                handled = await routeInteraction(interaction, strategies.BUTTONS, context);
            } else if (interaction.isStringSelectMenu()) {
                handled = await routeInteraction(interaction, strategies.MENUS, context);
            } else if (interaction.isModalSubmit()) {
                handled = await routeInteraction(interaction, strategies.MODALS, context);
            }
```

Primjer interakcije s *botom*



Dobrobiti asinkrone obrade u node.js

Sinkrono ograničenje

Tradicionalni *multi-threaded* sustavi alociraju dretvu za svaki korisnički zahtjev. Budući da bot većinu vremena provodi u *idle* stanju (čekajući API odgovore ili I/O baze), ovaj pristup nepotrebno troši serverske resurse i loše se skalira.

Event loop u praksi

Node.js koristi *single-threaded event loop*. Kada korisnik zatraži dodavanje resursa, Node.js ne blokira proces već delegira operaciju bazi. Istovremeno, bot bez zastoja nastavlja obrađivati poruke desetaka drugih studenata na serveru.

Sustav igrifikacije

Psihologija motivacije

Sustav se temelji na "Teoriji samoodređenja". Nagrađivanjem studenata XP bodovima zadovoljava se potreba za natjecanjem, a javne titule pokazuju tko su stručnjaci u pojedinim tehnologijama.

Optimizacija baze

Bot ne zapisuje svaki stečeni bod odmah u bazu jer bi to narušilo performanse sustava. Umjesto toga, koristi privremenu pohranu u radnoj memoriji te periodično izvršava masovne *bulkCreate* transakcije.

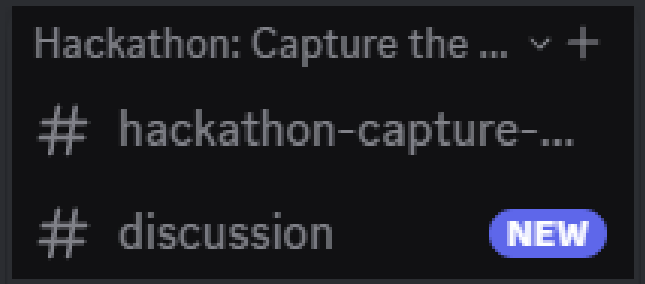
Ljestvica napredovanja

XP bodovi	Profesionalna titula	Kulturološka titula
0	Intern	Code Quality: Hello World
50	Trainee	Syntax Error
150	Junior Dev	Script Kiddie
300	Dev	Spaghetti Coder
500	Mid-Level Dev	Clean Coder
1000	Senior Dev	Refactorer
2000	Lead Dev	Optimizer
3500	Architect	System Architect
5000	Principal Eng	10x Engineer
7500	CTO	Code Whisperer
10000	Co-Founder	Binary God

Životni ciklus događaja

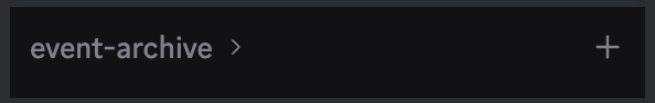
1. Najava

Student ili nastavnik organizira događaj kroz platformu. Bot automatski stvara kategoriju, kanale te obavještava zajednicu pomoću dinamički iscrtanih Embed objava.



2. Diskusija

Sudionici koriste tekstualni kanal za pripremu prije radionice i razmjenu koda tijekom samog događaja. Bot prati broj prijavljenih i ažurira najavu.



3. Digitalna arhiva

Nakon završetka događaja, kanali se ne brišu. Bot mijenja prava u *read-only* i premješta ih u mapu Arhiva čuvajući znanje za buduće generacije studenata.

Sustav podrške i sigurnost

Izolirani ticketing

Prijave problema ne događaju se u javnosti. Gumb "Create a ticket" instruira bota da asinkrono kreira zaključani kanal vidljiv samo korisniku i moderatorskom timu, čime je osigurana potpuna privatnost.

Dinamički banneri & Zod

Kako bi sučelje zahtjeva bilo pregledno, bot koristi **Canvas API** za generiranje slika statusa "u letu".
Sve što studenti upišu u forme strogo se validira pomoću **Zod** shema, prakticirajući pravilo *"Never trust user input"*.

Automatizacija pozadinskih procesa

Dinamički kanali

Privremeni glasovni kanali kreiraju se ulaskom u glavni "Voice hub". *Garbage collection* automatizirano briše kanale kada u njima ne ostane niti jedan korisnik.

Periodički poslovi (cron)

Pomoću **node-cron** biblioteke bot svakog ponedjeljka samostalno dohvaća plan iz baze i ispisuje strukturirani pregled tjedna u kanal s najavama.

REST API integracija

Express.js servis funkcionira paralelno na istom serveru. On stvara **single source of truth**, sinkronizirajući stanje bota i leaderboard izravno s web stranicom kluba.

Agilni razvoj i Kolbov ciklus

1. Konkretno iskustvo

Studenti uočavaju problem u radu bota, npr. pad baze pri pokušaju zapisivanja tisuća XP bodova odjednom.

2. Refleksija

Zajednička rasprava studenata i nastavnika na Discordu oko uzroka uskog grla analize zapisa iz Docker logova.

3. Konceptualizacija

Primjena teorije s predavanja; diskusija o upotrebi *cache* memorije i ACID transakcija kao inženjerskog rješenja.

4. Eksperimentiranje

Samostalni razvoj asinkronog "bulk" spremanja. Kod prolazi Code Review te se u rad pušta stabilnija verzija bota.

Iskustveno učenje

> Razvoj ovog bota dokazao je kako se teorijski koncepti s predavanja mogu pretvoriti u produkcijska rješenja bez stresa od formalnog ocjenjivanja.

Implementacija apstraktnih pojmova poput ACID transakcija, Node.js Event loopa i Docker kontejnera kroz rad na zajedničkom "open-source" alatu.

Budućnost sustava

- iako trenutno stabilan kao modularni monolit, rast broja studenata mogao bi opravdati migraciju na arhitekturu mikroservisa.

- izdvajanje modula za igrifikaciju uz integraciju Redis *cachiranja* osigurao bi glatko horizontalno skaliranje i u godinama koje dolaze.

dev-club-prezentacija

Pitanja?

Hvala na pažnji!

lposaric@foi.hr | tnikolic@foi.hr | dplantak@foi.hr