

ProgInz TG08.3 Home2
Iнтерно	3
Dev Guidelines	4
Git management	5
JIRA management	7
Backend	12
Deployment	13
Backend Guide	14
Modeli	15
Konfiguracija entiteta	16
DbContext	17
Migracije	18
Services	19
Controllers	20
API Keys	21
Roles management	22
Frontend	23
Općenito	24
Login	25
Službena Dokumentacija	26
Home	27
1. Opis projektnog zadatka	28
2. Analiza zahtjeva	31
4. Specifikacija zahtjeva sustava	35
5. Arhitektura i dizajn sustava	48
6. Implementacija i korisničko sučelje	58
7. Zaključak i budući rad	60
8. Popis literature	61
9. Prikaz aktivnosti grupe	62
Meetings	65
Meeting 29.10.2024	66
Meeting 28.10.2024	67
Meeting 25.10.2024	72
Meeting 23.10.2024	73
Meeting 22.10.2024	75
Meeting 18.10.2024	77
Meeting 13.10.2024	78
Meeting 10.10.2024	79
BE - Meeting 30.11.2024	80
BE - Meeting 9.12.2024	81
A. Dnevnik promjena dokumentacije	82
Untitled whiteboard (2)	95

ProgInz TG08.3 Home

Site location: [Home Page - Duck!](#)

- Filip Belina - Project Lead
- Jakov Lovaković - Backend Team Lead
- Jan Lalić - Backend Engineer
- Leo Marušić - DevOps/Database Engineer
- Mislav Marinović - Lead design
- Jan Badel - Frontend Engineer
- Martin Šainčević - Frontend engineer

- Frontend - Bootstrap + jQuery
- Backend - ASP .NET
- Database - SQLite
- Project Management - JIRA + Confluence



Interno

Dev Guidelines

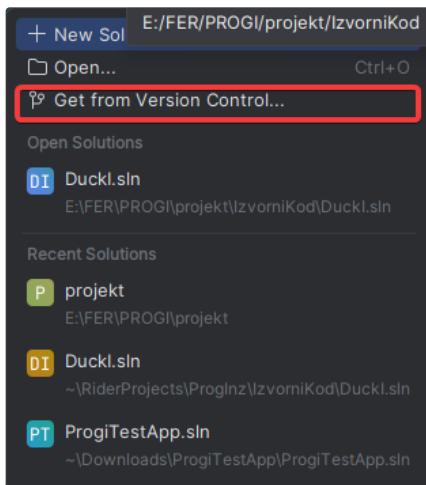
Git management

Ovaj page će opisati procese kojima koristimo github, ovaj prikaz govori o tome u sklopu Ridera, ali ideje su iste bez obzira na koji git management tool bude odabran.

1. Kloniranje repozitorija

Ako repozitorij još nije kloniran na lokalno računalo:

- Otvori Rider i odabereti **Get from Version Control** s početnog zaslona.

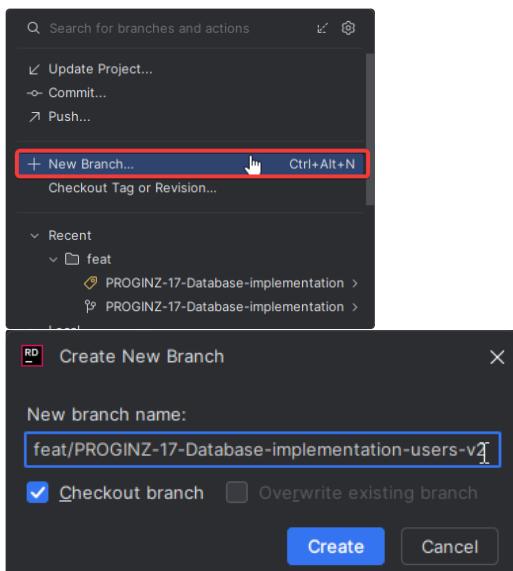


- Upiši URL GitHub repozitorija, odabereti lokalnu mapu za kloniranje i klikni na **Clone**.

2. Kreiranje nove grane iz *main* grane

Da bi kreirao novu granu:

- Provjeri da si na glavnoj (*main*) grani tako da u donjem desnom kutu Rider prozora vidiš naziv trenutne grane.
- Desnim klikom na trenutnu granu ili klikom na izbornik **Git** u alatnoj traci odabereti **New Branch....**



- Unesi naziv nove grane, npr. `feat/TicketKey-TicketName`, i klikni **Create**. Rider će te automatski prebaciti na novu granu.
- Naziv grane uvijek mora biti jedan od 4:
`feat/TicketKey-TicketName`,
`bug/TicketKey-TicketName`,

```
test/TicketKey-TicketName ,  
type/TicketName
```

3. Dodavanje i commitanje promjena

Nakon što napraviš promjene u projektu:

- Otvori alatnu traku Git-a s **View > Tool Windows > Git** ili klikni na **Commit** ikonu u donjem dijelu zaslona.
- Označi promjenjene datoteke koje želiš dodati u commit ili označi sve promjene.
- Upiši opis promjena u polje za poruku.
- Klikni na **Commit** ili **Commit and Push** (ako želiš odmah poslati promjene na GitHub).

4. Slanje promjena na GitHub (push)

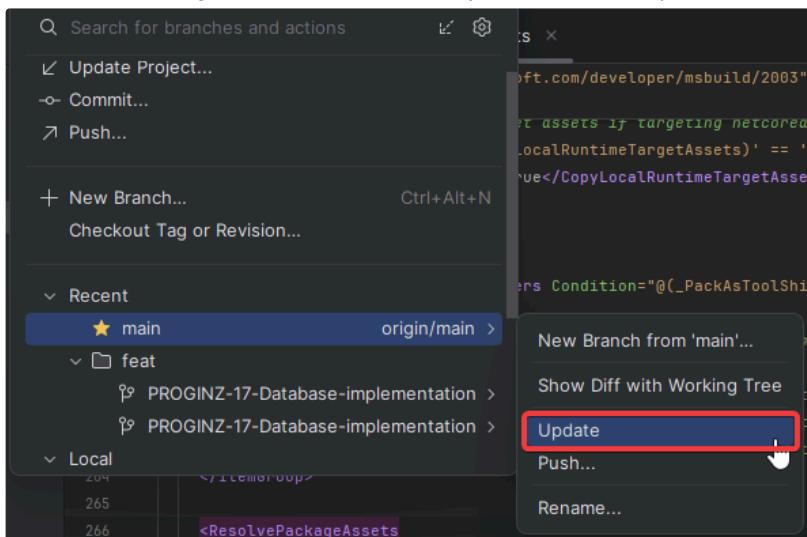
Ako si samo napravio commit i nisi još poslao promjene:

- Otvori **Git** alatnu traku.
- Klikni **Push** ili desni klik na granu i odaberis **Push**. Time ćeš poslati promjene na GitHub repozitorij.

5. Povlačenje najnovijih promjena s GitHub-a (pull)

Ako želiš povući najnovije promjene s *main* grane na GitHubu:

- Prebac se na *main* granu kroz Git izbornik u donjem desnom kutu ili pomoću **Git** alatne trake.



- Klikni na **Fetch**, **Pull** ili **Update** kako bi povukao najnovije promjene u svoj lokalni repozitorij.

6. Spajanje promjena s *main* granom

Kad završiš rad na *nova-grana* i želiš je spojiti s *main*:

- Otvori Github te na projektu napravi novi pull request baziran na grani koju želiš spojiti u glavni projekt.
- Nakon pull requesta potrebno je čekati odobrenje 2 druga developer, ili dobiti explicitno dopuštenje za mrganje od projekt leada
- Kada su svi uvjeti ispunjeni (dovoljan broj aproveova, bez ne razriješenih komentara, bez merge conflikata) dozvoljeno je dovršiti pritiskom na merge.

7. Nakon završetka na Jiri zatvoriti ticket

Potrebno je otici na ticket i prebaciti njegov state na closed

JIRA management

Stranica za quick reference kako koristiti Jiru.

Main Dashboard

Dodavanje Main Dashboard-a u quick access

Jira workflow

Otvaranje ticketa

Povezivanje sa Confluence-om

Main Dashboard

Kreiran je main dashboard koji odmah prikazuje informacije bitne za korisnika

Main Dashboard

Issues in progress		
T	Key	Summary
<input checked="" type="checkbox"/>	PROGINZ-2	Setting up the deployment environment
1-1 of 1		
3 minutes ago		

All Tickets		
T	Key	Status
<input checked="" type="checkbox"/>	PROGINZ-19	OPEN Oauth2 Implementation
<input checked="" type="checkbox"/>	PROGINZ-18	OPEN Clean up template code
<input checked="" type="checkbox"/>	PROGINZ-17	CLOSED Database implementation - users
<input checked="" type="checkbox"/>	PROGINZ-16	OPEN Login screen - fe
<input checked="" type="checkbox"/>	PROGINZ-14	CLOSED Projekiranje baze podataka
<input checked="" type="checkbox"/>	PROGINZ-13	CLOSED Poslati verziju 2 dokumentacije profesoru
<input checked="" type="checkbox"/>	PROGINZ-12	CLOSED Dokumentacija part 2
<input checked="" type="checkbox"/>	PROGINZ-11	OPEN Učenje - Design
<input checked="" type="checkbox"/>	PROGINZ-10	OPEN PROGINZ-8 / Jan
<input checked="" type="checkbox"/>	PROGINZ-9	OPEN PROGINZ-8 / Martin
1-10 of 16		
3 minutes ago		

Participating or Awaiting input		
T	Key	Summary
<input checked="" type="checkbox"/>	PROGINZ-19	Oauth2 Implementation
<input checked="" type="checkbox"/>	PROGINZ-2	Setting up the deployment environment
1-2 of 2		
3 minutes ago		

Activity Streams		
Activity Stream		
Yesterday		
	Ellip.Belina	changed the status to Closed on PROGINZ-8 - Učenje - FrontEnd with a resolution of 'Done'
	Ellip.Belina	Yesterday Comment Watch
Today		
	Ellip.Belina	changed the status to Closed on PROGINZ-12 - Database implementation - users with a resolution of 'Done'
	Ellip.Belina	4 days ago Comment Watch
	Leo Marušić	updated the Description of PROGINZ-19 - Oauth2 Implementation
		OAuth2 Open Authorization 2.0 je industrijski standardni protokol koji omogućuje web aplikacijama da pristupe resursima na drugim web aplikacijama u ime korisnika, bez potrebe za dajeњем korisničkih vjerodajnica (npr. korisničko ime i lozinka). Read more
		<input checked="" type="checkbox"/> 4 days ago Comment
		Leo Marušić created PROGINZ-19 - Oauth2 Implementation

Dashboard je podijeljen na 4 dijela i prikazuje:

- Issues in progress - ticketi koji su u statusu "in progress" - odnosi se na sve tickete
- All tickets - lista svih ticketa neovisno o statusu
- Participating or Awaiting input - ticketi u kojima je trenutni korisnik u "assingnee" ili "participants" polju, te ticketi koji zahtjevaju pažnju (vidi workflow)
- Activity stream - lista aktivnosti na Jiri i Confluence-u

Dodavanje Main Dashboard-a u quick access

Klikom na Dashboards → View all dashboards

Odvjet će se na stranicu svih dashboard-a, gdje se nalazi Main Dashboard.

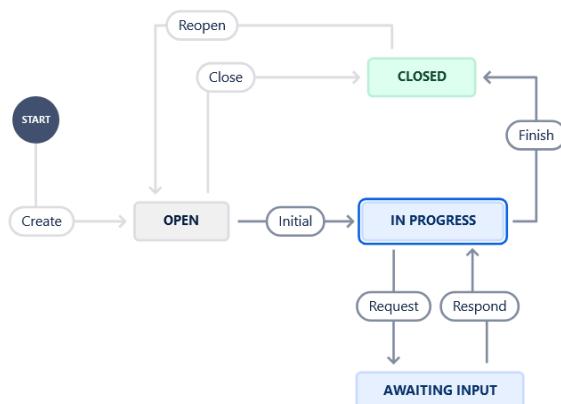
Kliknuti na zvijezdicu za lakši pristup dashboard-u.

Main Dashboard

Te će onda Dashboards tab izgledati ovako:

Jira workflow

Workflow ticketa izgleda ovako:



Kada se kreira ticket, započne odmah u "open" statusu, zatim kada neko preuzme ticket, ta osoba stavlja ticket u "in progress". Ako tijekom rada, ticket zahtjeva dodatan input od nekoga trećeg, stavlja se u "awaiting input" status (koji onda svi vide u main dashboardu). Kada ta osoba odgovori ili se napravi posao koji se zahtjeva za nastavak ticketa, stavlja se nazad u "in progress" i nastavlja se sa radom. Na kraju, ticket se stavlja u "closed" status.

Otvaranje ticketa

Klikom na Create gumb, otvara se prozor za izradu ticketa.

The screenshot shows the Jira 'Create' ticket form. At the top, there's a navigation bar with links: Your work, Projects (selected), Filters, Dashboards, Teams, Plans, Apps, and a prominent blue 'Create' button. Below the navigation, a list of requirements is provided:

- Project: ProgInz (automatski se dodjeljuje)
- Issue type: najbolje ostaviti kao "Task"
- Summary: ime ticketa - treba popuniti
- Description: detalji ticketa, može ostati prazno u početku

The main form area is titled 'Create' and includes the following fields:

- Project***: A dropdown menu showing 'ProgInz (PROGINZ)' selected.
- Issue type***: A dropdown menu showing 'Task' selected.
- Status**: A dropdown menu showing 'Open' selected.
- Summary***: A text input field containing the placeholder 'Test'.
- Description**: A rich text editor with a toolbar featuring icons for bold, italic, underline, etc. It also includes a note: 'Words not enough? Type : to add emoji.' followed by a smiling emoji.

- Assignee: dodjeliti sebi ili drugoj osobi za koju je taj ticket
- Participants: dodati ljude koji rade na ticketu
- Labels: može se dodati label po potrebi
- Parent: ignorirati
- Team: Određena su 2 tima, frontend i backend, može se njih ubaciti u to polje
- Sprint: **OBAVEZNO** postaviti na "PROGI Sprint"

Assignee

Leo Marušić

[Assign to me](#)**Participants**

Add people that participate in this task.

Labels

Select label

**Parent**

Select parent



Your issue type hierarchy determines the issues you can select here.

Team

Choose a team

Associates a team to an issue. You can use this field to search and filter issues by team.

Sprint

PROGI Sprint



Jira Software sprint field

- Story point estimate: procjena vremena u danima rada (1 dan = 8h)
- Fix versions: ignore
- Reporter: ostaviti default ili staviti na nekog drugog (npr. ako je team lead rekao da se napravi ticket)
- Attachment: upload za bilo kakav file vezan uz ticket
- Linked issued: postaviti ovisno o relaciji sa ostalim ticketima
- Flagged: ignore

Story point estimate

Measurement of complexity and/or size of a requirement.

Fix versions

None

**Reporter***

Leo Marušić

Attachment Drop files to attach or [Browse](#)**Linked Issues**

blocks



Select Issue

**Flagged** Impediment

Allows to flag issues with impediments.

 Create another

Povezivanje sa Confluence-om

Ispod "Description" polja u ticketu, nalazi se "Confluence content" sekcija za povezivanje sa confluence-om.

The screenshot shows the 'Confluence content' section within a ticket view. At the top left is a 'Deployment' card with a blue icon, labeled 'Updated last week LM'. To its right is a '+' button. Below this is a search bar containing the text 'home'. To the left of the search bar is a 'Filter by space' dropdown menu. To the right is a 'Create in Confluence' dropdown menu. A 'Matching pages' section is visible, showing a single result: 'Proglnz TG08.3 Home' with a blue icon, followed by the text 'Proglnz TG08.3 • Recently viewed'.

Backend

Deployment

All the information about the deployment of the web app.

Ubuntu server deployed on a Azure B1 virtual machine, with whole 1 CPU core and 1 Gib of RAM.

It runs an ASP .NET application as a *systemd* service which is interfaced with a *nginx* reverse proxy server.

Backend Guide

Svrha ovog priručnika je pojašnjenje backend koncepata koji se koriste u Duckl projektu.

Priručnik je pisan za [ASP.NET Core](#), 26. 10. 2024.

Backend priručnik je podijeljen na više važnih cijelina:

- [Modeli](#)
- [Konfiguracija entiteta](#)
- [DbContext](#)
- [Migracije](#)
- [Services](#)
- [Controllers](#)

Modeli

Unutar „Data” class project-a možemo pronaći direktorij „Modeli”. U taj ćemo direktorij spremati klase koje će modelirati potrebne tablice u samoj bazi, te koje će se koristiti u nekim drugim dijelovima backenda, kao npr. „[Services](#)”. Uz same modele tablice, u ovom folderu nalazi se i klasa koja nasljeđuje „[DbContext](#)”.

Primjer klase `User`:

```
1 public class User
2 {
3     [Key]
4     [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
5     public long IdUser { get; set; }
6
7     public Guid Guid { get; set; }
8
9     [Required]
10    [StringLength(1000)]
11    public string Password { get; set; }
12
13    [Required]
14    [StringLength(100)]
15    public string Mail { get; set; }
16
17    // navigational properties
18    public Student? Student { get; set; }
19    public Educator? Educator { get; set; }
20    public Administrator? Administrator { get; set; }
21    public Reviewer? Reviewer { get; set; }
22 }
```

Klasa `User` definira tablicu `Users` unutar baze. Ona sadrži više propertyja koji se prevode u stupce tablice: `IdUser`, `Guid`, `Password`, `Mail`. Svaki od ovih propertyja bit će kasnije preslikan u odgovarajući stupac tablice. Tip podataka koji će se spremati u određenom stupcu određuje tip propertyja. Npr., `long` se preslikava u bigint.

Propertyji `Student`, `Educator`, `Administrator` i `Reviewer` su navigacijski propertyji. Oni se ne preslikavaju u stupce tablice. Navigacijski propertyji služe za olakšavanja rada s podacima unutar [ASP.NET Core](#)-a. `?` na kraju svakog tipa navigacijskog propertyja označava da vrijednost tog propertyja može biti null.

Stvari unutar uglatih zagrada direktno iznad propertyja označavaju njegovo ograničenje ili svojstvo. Npr. `[Key]` označava primarni ključ, `[StringLength(1000)]` označava ograničenje na 1000 znakova.

Konfiguracija entiteta

Direktorij „Configuration” može biti pronađen unutar „Data” class project-a. Ovdje će se nalaziti sve klase koje će „manualno” puniti bazu podacima. Inače će korisnik puniti bazu podacima pozivajući odgovarajuće rute.

Primjer klase `UserConfiguration`:

```
1 public class UserConfiguration : IEntityConfiguration<User>
2 {
3     public void Configure(EntityTypeBuilder<User> builder)
4     {
5         builder.HasData(
6             new User { IdUser = 1, Guid = Guid.NewGuid(), Password = "password1", Mail = "user1@example.com"
7             },
8             new User { IdUser = 2, Guid = Guid.NewGuid(), Password = "password2", Mail = "user2@example.com"
9             },
10            new User { IdUser = 3, Guid = Guid.NewGuid(), Password = "password3", Mail = "user3@example.com"
11            },
12            new User { IdUser = 4, Guid = Guid.NewGuid(), Password = "password4", Mail = "user4@example.com"
13            },
14            new User { IdUser = 5, Guid = Guid.NewGuid(), Password = "password5", Mail = "user5@example.com" }
15        );
16    }
17 }
```

Prvo što ovdje možemo primjetiti da klasa `UserConfiguration` nasljeđuje (implementira) sučelje `IEntityTypeConfiguration<User>`, parametrizirano s `User`. To je nešto što svaka klasa koja implementira nekakvu konfiguraciju opisanu u gornjem paragrafu mora imati. Klasa se parametrizira onim modelom entiteta u čiju tablicu ubacujemo podatke.

Klasa implementira metodu u `public void Configure(EntityTypeBuilder<User> builder)` u kojoj definiramo instance entiteta kao npr. `new User { IdUser = 1, Guid = Guid.NewGuid(), Password = "password1", Mail = "user1@example.com" }`.

DbContext

`DbContext` je jedna od najvažnijih klasa u cijelome backend dijelu projekta. `DbContext` koristi se za više stvari, primarno za povezivanje baze i za dohvatanje podataka ili uređivanje podataka iz baze.

Ova klasa implementira se pomoću klase `DataContext` koja direktno nasljeđuje `DbContext`. Klasa `DataContext` može se pronaći u direktoriju „Models”, te je jedina klasa u tom direktoriju koja ne pripada entitetima.

Klasa `DataContext`:

```
1 public class DataContext : DbContext
2 {
3     public DataContext(DbContextOptions options) : base(options) { }
4
5     public DbSet<User>? Users { get; set; }
6     public DbSet<Student>? Students { get; set; }
7     public DbSet<Educator>? Educators { get; set; }
8     public DbSet<Administrator>? Administrators { get; set; }
9     public DbSet<Reviewer>? Reviewers { get; set; }
10
11    protected override void OnModelCreating(ModelBuilder modelBuilder)
12    {
13        modelBuilder.ApplyConfiguration(new StudentConfiguration());
14        modelBuilder.ApplyConfiguration(new UserConfiguration());
15        modelBuilder.ApplyConfiguration(new EducatorConfiguration());
16        modelBuilder.ApplyConfiguration(new AdministratorConfiguration());
17        modelBuilder.ApplyConfiguration(new ReviewerConfiguration());
18
19        base.OnModelCreating(modelBuilder);
20    }
21 }
```

Prvo imamo konstruktor `public DataContext(DbContextOptions options) : base(options) { }` koji call base konstruktor klase `DbContext`.

Nakon toga imamo `DbSet<T>` propertyje:

```
1 public DbSet<User>? Users { get; set; }
2 public DbSet<Student>? Students { get; set; }
3 public DbSet<Educator>? Educators { get; set; }
4 public DbSet<Administrator>? Administrators { get; set; }
5 public DbSet<Reviewer>? Reviewers { get; set; }
```

Za svaku tablicu koju kreiramo preko klasa entiteta u direktoriju „Models” moramo imati jedan `DbSet<T>` oblika iz gornjeg isječka.

Metoda:

```
1 protected override void OnModelCreating(ModelBuilder modelBuilder)
```

Služi za primjenjivanje konfiguracije koje smo spominjali u [konfiguraciji entiteta](#).

Migracije

Migracije služe kako bi primjenili sve važne konfiguracije kao npr. brisanje stupaca, dodavanje stupaca, dodavanje tablica i sl. na bazu podataka.

One se nalaze unutar „Data” class project-a.

Migracije se kreiraju pri svakoj manualnoj promjeni baze podataka, kao npr. mijenjanje strukture baze ili tablica, te seedanje podataka.

Migracije se kreiraju sljedećom naredbom u terminalu projekta:

```
Add-Migration Initial -Project Data -StartupProject DuckI.Server
```

Gdje -Project predstavlja projekt u kojem se nalazi migracija, dok StartupProject predstavlja projekt u kojem se nalazi `program.cs` u kojem je napravljena konekcija na bazu.

Ako se žele primjeniti migracije, u terminalu se pokreće `Update-Database`.

Services

Servisi su usluge koje implementiramo za različite entitete. Servisi se nalaze u „Data“ project-u.

Servisi započinju pisanjem interface-a. Unutar interface-a se zapišu definicije svih metoda koje servis implementira. Nakon toga se kreira klasa za odabrani servis, te se implementira napisani interface. Jedan interface može imati više metoda.

Klasa `UserService` :

```
1  using Data.Models;
2  using Microsoft.EntityFrameworkCore;
3
4  namespace Data.Services;
5
6  public interface IUserService
7  {
8      Task<List<User>> GetUserRolesAsync();
9  }
10
11 public class UserService : IUserService
12 {
13     private readonly DataContext _context;
14
15     public UserService(DataContext context)
16     {
17         _context = context;
18     }
19
20     public async Task<List<User>> GetUserRolesAsync()
21     {
22         // implementacija funkcije, ugl. fetchanje podataka iz baze
23     }
24 }
25 }
```

Dodatno, servis se mora registrirati u `program.cs` :

```
1 builder.Services.AddScoped<IUserService, UserService>();
```

Controllers

Kontroleri služe za implementaciju ruta. Unutar njih ćemo pozivati metode koje implementiramo unutar servisa, te ćemo slati http response.

Mi ćemo kontrolere odvajati slično kao i [servise](#), po entitetima, ali uz par iznimaka po potrebi.

Klasa `UserController`:

```
1  using Data.Dto;
2  using Data.Helpers;
3  using Data.Services;
4  using Microsoft.AspNetCore.Mvc;
5
6  namespace DuckI.Server.Controllers;
7
8  [ApiController]
9  [Route("api/users")]
10 public class UserController : ControllerBase
11 {
12     private readonly IUserService _userService;
13
14     public UserController(IUserService userService)
15     {
16         _userService = userService;
17     }
18
19     //[HttpGet("[action]")]
20     [HttpGet("displayroles")]
21     public async Task<IActionResult> DisplayRoles()
22     {
23         var users = // pozovi funkciju servisa
24         var userRoles = // koristi dto
25         return Ok(userRoles); // vrati http response
26     }
27 }
```

Na početku se mogu primjetiti dva dekoratora, `[ApiController]` i `[Route("api/users")]`. Oni označavaju kontroler i default rutu za kontroler. Svaka metoda također mora biti dekorirana, kao npr. `[HttpGet("displayroles")]`. „displayroles“ će ići na kraj `api/users/` rute, ako se želi pozvati funkcija `DisplayRoles`. Dekoratori određuju rutu koju se koristi za poziv svake od funkcija.

Svaki kontroler sadrži člana koji je referenca na objekt prikladnog servisa.

API Keys

Google Authorized redirect URLs:

<https://localhost:44385/signin-google>

<https://www.ducki.space/signin-google>

<https://ducki.space/signin-google>

Adding ClientID and Secret:

```
1 dotnet user-secrets set "Authentication:Google:Client" "544574227265-  
98vul8lksa7eugpu3sl3m7r39kcv7r86.apps.googleusercontent.com"
```

```
1 dotnet user-secrets set "Authentication:Google:ClientSecret" "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"
```

Microsoft account:

```
1 dotnet user-secrets set "Authentication:Microsoft:Client" "e233485c-0b18-4595-a33b-42ba9c833aa0"
```

```
1 dotnet user-secrets set "Authentication:Microsoft:ClientSecret" "DKR8Q-WMuHs0W4ms124h2vtTdtfcqkey-emAsaT4"
```

Microsoft Authorized redirect URLs:

<https://localhost:44385/signin-microsoft>

<https://www.ducki.space/signin-microsoft>

<https://ducki.space/signin-microsoft>

Gemini API Key

```
dotnet user-secrets set "Key:Gemini" "AIzaSyCi9RBpz8JGa1KHUSzqSiG-0Cn50Mngd08"
```

Roles management

U našem projektu korisnici mogu birati roleove. Svaki korisnik je pri prijavi default user, ali kasnije može odabrati role koji će bolje opisati njegove potrebe.

Imamo dvije stranice za role management.

Prva je `/Home/Roles`. Ova ruta služi za odabir roleova prijavljenog korisnika, te se roleovi biraju pomoću forma. Toj ruti mogu pristupiti SVI osim Admina, pošto adminine određujemo mi kao organizacija, te oni ne mogu imati dodatne uloge. `/Home/Roles` je upravljan s

1 `HomeController.cs`

Druga ruta je `/Miscellaneous/BrowseRoleApplications`. Ovoj ruti mogu pristupiti SAMO Admini. Ova ruta služi adminima za approveanje/rejectanje user applicationa za roleove. `/Miscellaneous/BrowseRoleApplications` je upravljan s

1 `MiscellaneousController.cs`

Frontend

Općenito

Naš projekt koristi [ASP.NET](#) Razor Pages kao temelj za frontend dio aplikacije, omogućujući strukturiranu i modernu arhitekturu web stranica. Koristimo Bootstrap kao CSS framework za responzivni i estetski ugodan izgled stranice, olakšavajući rad na frontendu s unaprijed definiranim stilovima i komponentama.

Tehnologije:

Razor Pages: Stranični model programiranja unutar ASP.NET-a koji omogućava organizaciju logike i sadržaja pojedinačnih stranica bez korištenja klasične MVC arhitekture.

Bootstrap: CSS framework za responsive dizajn, koji osigurava da naša aplikacija izgleda jednako dobro na mobilnim uređajima, tabletima i desktop uređajima.

Struktura projekta

Pages direktorij: Sadrži .cshtml datoteke koje predstavljaju pojedinačne stranice, zajedno s pripadajućim .cshtml.cs datotekama za poslovnu logiku i povezivanje s backendom.

Index.cshtml: Početna stranica aplikacije.

Shared direktorij: Sadrži dijeljene komponente, uključujući Layout.cshtml koji definira izgled aplikacije, navigacijski izbornik i zaglavlje.

wwwroot direktorij: Sadrži sve statičke datoteke poput slika, stilskih datoteka i JavaScript datoteka.

CSS: Koristimo prilagođeni CSS (ako je potrebno) zajedno s Bootstrap CSS-om.

JS: Smještamo JavaScript datoteke (osim ako nisu uključene iz CDN-a).

_ViewImports.cshtml i _ViewStart.cshtml: Konfiguracijske datoteke za učitavanje često korištenih namespaceova i postavljanje početnih postavki.

Bootstrap integracija

Implementacija Bootstrapa omogućava nam brzo oblikovanje korisničkog sučelja bez previše prilagođavanja. Većina elemenata (forme, kartice, navigacija, gumbi) koristi Bootstrap komponente, što osigurava konzistentan izgled i jednostavniju prilagodbu.

Responzivni dizajn: Stranice su optimizirane za rad na različitim uređajima, čime poboljšavamo korisničko iskustvo bez potrebe za dodatnim stiliziranjem.

Komponente: Koristimo komponente kao što su modali, dropdown izbornici, tablice i forme kako bi aplikacija izgledala profesionalno i bila funkcionalna.

Login

Naša Login stranica koristi **ASP.NET** Razor Pages za korisnički interfejs za prijavu korisnika. Stranica omogućava prijavu putem lokalnog računa ili korištenje vanjskih autentifikacijskih usluga, uz implementaciju Bootstrap frameworka za responzivni dizajn i modernu estetiku.

Struktura stranice

Osnovni element:

`@page` i `@model` Direktive: Ove direktive označavaju da je ovo Razor stranica koja koristi `LoginModel` model za povezivanje s backend logikom.

`ViewData["Title"]`: Postavlja naslov stranice na "Log in".

Forma za prijavu:

Lokalni račun: Omogućuje korisnicima da se prijave koristeći email-a i lozinke.

Polja za unos:

Email: Korištenje `asp-for` omogućava bindanje unosa email adrese s modelom (`Input.Email`) uz validaciju.

Lozinka: Korištenje `asp-for` za unos lozinke (`Input.Password`) uz validaciju.

Zapamti me: Opcija za zapamćivanje korisnika koristeći checkboxa.

Validacija: Uključena je **ASP.NET** validacija na strani servera za email, lozinku i "Zapamti me", s prikazom grešaka u tekstualnom formatu.

Submit gumb: Gumb za prijavu označen kao `btn-primary` koji omogućava slanje forme.

Dodatne opcije za korisnike:

Zaboravljena lozinka: Link za resetiranje lozinke.

Registracija novog korisnika: Link za registraciju novog korisnika uz parametar za povratnu URL adresu.

Ponovno slanje potvrde emaila: Link za ponovno slanje email potvrde.

Vanjske usluge za prijavu:

Vanjski login: Korisnici mogu odabrat jednu od ponuđenih opcija za prijavu(Google, Microsoft)

Responzivni dizajn:

Stranica koristi Bootstrap grid klasu s dva stupca. Prvi stupac sadrži formu za prijavu, dok drugi stupac prikazuje opcije za prijavu putem vanjskih usluga.

Bootstrap-ove klase kao što su `form-floating`, `mb-3`, i `btn btn-primary` koriste se za moderni izgled forme i gumba.

Responzivnost je osigurana automatski pomoću Bootstrapa, što omogućava pravilno prikazivanje stranice na različitim uređajima (desktop, tablet, mobilni uređaji).

Službena Dokumentacija

Home

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elekrotehnike i računarstva

Duckl

Tim: <TG08.3>

Gumene Patkice

Nastavnik: Vlado Sruk

1. Opis projektnog zadatka

Funkcionalnost

Kao projektni zadatak izvodit će se aplikacija za pomoć pri učenju u obliku AI asistenta. Koristeći materijale tekstualnih oblika poput PDF, AI asistent bi kreirao brojan sadržaj s kojim bi se korisnik mogao bolje shvatiti određenu temu uz osjećaj personaliziranog podučavanja. Uz studente aplikaciju bi koristili i edukatori koji bi imali mogućnost objavljivanja svog sadržaja, resursa i materijala koje može koristiti student ili ih AI asistent može koristiti kao izvor podataka. Nad edukatorima nadgledaju i administratori čija je uloga provjeravati točnost sadržaja koji su dodali edukatori.

Studentu aplikacija služi za spremište i upravljanje svojim materijalima. Kada će student htjeti interakciju s AI asistentom, generirali bi se resursi u obliku raznih zadataka, „flashcards“-ova, te ostalih oblika učenja koje bi studentu pomogle pri savladavanju određenih tema. Kao izvore podataka bi se koristili upravo materijali koji su spremljeni na aplikaciju, tako da se svakome studentu može kreirati točno taj sadržaj koji njemu treba.

AI asistent je izведен pomoću Google Gemini 1.5 Flash modela. Model će primati dokumente (poput PDF-a), te na temelju njih stvoriti sadržaj, zadatake i ostale resurse koji bi pomogli studentu. Student sam bira dokumente iz kojih želi generirati sadržaj.

Uz studenta, aplikacijom se mogu koristiti i edukatori koji bi imali mogućnost kreiranja i dodavanja svojih materijala za koje oni smatraju da bi mogli pomoći studentu, te se student može pretplatiti na dodatne materijale što bi omogućilo AI asistentu da se služi s tim materijalima pri generiranju sadržaja.

Edukatore nadzire administrator koji se bavi pregledom sadržaja i rada edukatora, te se bavi prijavama, poput neispravnog sadržaja ili sličnog. Oni bi odobravali edukatore, resurse i bavili time da je sadržaj aplikacije i rad edukatora u skladu sa pravilima ponašanja.

Svi korisnici aplikacije se moraju prijaviti i potvrditi svoj identitet, te je u tu svrhu korištena OAuth2 autentifikacija. Responzivnost će biti ostvarena pomoću modernih alata poput React, Typescripta te Tailwind CSS-a. U tu svrhu, koristit će se principi responzivnog dizajna za dinamičku prilagodbu korisničkog sučelja različitim veličinama i razlučivostima zaslona.

Aplikacija također sadrži kalendar preko kojega studenti mogu upravljati svoj raspored sati. Mogu staviti svoj predefinirani raspored iz .csv datoteke ili izraditi svoj klikanjem na željene datume.

Studenti mogu ostaviti recenzije o materijalima, na bazi tih recenzija se odabiru najprimjereniji materijali za učenje.

1. Svrha i ciljevi:

Ovaj projekt radi na razvoju aplikacije s AI asistentom pri učenju. Ovo bi bio asistent za učenje vođen umjetnom inteligencijom koji personalizira proces učenja za učenike putem prilagođenih obrazovnih resursa poput flash kartica i zadataka temeljenih na materijalu koji će studenti učitati s dodatnim resursima edukatora.

Aplikacija će nastavnicima dati platformu za objavljivanje dodatnog materijala za učenje koji učenici mogu koristiti za dublje proučavanje tema. Kako bi osigurali kvalitet i integritet obrazovanja, administratori će moderirati aktivnosti edukatora kroz kontrolu valjanosti ponuđenog sadržaja i prijavljivanje informacija koje bi mogle biti netočne.

Aplikacija koristi OAuth2 autentifikaciju s responzivnim dizajnom kako bi pružila iskustva na uređajima na siguran, pristupačan i prilagodljiv način.

2. Moguće koristi:

Koristi su najbolje vidljive za dvije skupine ljudi:

Studenti imaju koristi od prilagođenih pomagala za učenje koja zadovoljavaju individualne potrebe u učenju, što poboljšava

razumijevanje i prisjećanje naučenoga. Sposobnost kreiranja personaliziranog sadržaja u aplikaciji pomaže premostiti te nedostatke u razumijevanju i omogućuje učeniku da se koncentriра na ona područja koja trebaju više pozornosti.

Edukatori će moći dijeliti materijale koji mogu pomoći učenicima da bolje razumiju određene koncepte. Oni mogu učinkovitije olakšati učenje prikladnim primjerima, vježbama i vodičima za učenje.

3. Povezani radovi i razlike:

Iako već postoje slična rješenja za obrazovne svrhe, međutim, nema puno proizvoda koji kombiniraju personaliziranog asistenta uz pomoć umjetne inteligencije sa stvaranjem sadržaja u stvarnom vremenu s opcijom postavljanja globalnih materijala za edukatore. Slični koncepti na tržištu: Khan Academy: Tečajevi s modulima, ali ne bilo kakvim dinamičkim, AI generiranim resursima za učenje koji se temelje na materijalima koje su učitali određeni studenti. Quizlet: Flash kartice i kvizovi, ali oni se moraju unijeti ručno; ovaj projekt automatski generira pomoćna sredstva za učenje. Coursera/EdX: Vrijednost ovih platformi leži u strukturiranim tečajevima koje edukatori stvaraju. Ne postoji interakcija uživo s kojom bi AI prilagodio sadržaj na temelju datoteka koje su poslali korisnici. MindGrasp.ai: Najsličniji proizvod, na bazi AI materijale pretvara u notes, flashcards, i tasks, no ne postoji nikakva interakcija ili sistem s edukatorom, što bi činilo ovo teškim za organizirati u razrednom okruženju.

4. Ciljane korisničke skupine:

Ciljane skupine korisnika koje ova aplikacija namjerava obuhvatiti su:

Studenti: Primarni korisnici, traže personaliziranu poduku i pomoć pri učenju na zahtjev.

Edukatori: Sekundarni korisnici koji mogu proširiti iskustvo učenja dijeljenjem kvalitetnih obrazovnih resursa.

5. Prilagodba i prilagodljivost:

Ova je aplikacija namijenjena jednostavnoj prilagodljivosti različitim obrazovnim okruženjima. Sadržaj vođen umjetnom inteligencijom, personaliziran za svakog učenika, može obuhvaćati širok raspon predmeta i metodologija podučavanja/učenja.

6. Opseg projekta:

Funkcionalnost aplikacije bit će višestruka:

Generiranje sadržaja s pomoću umjetne inteligencije: To omogućuje stvaranje resursa za učenje iz PDF-ova i drugih tekstualnih formata korištenjem modela Google Gemini 1.5 Flash.

Autentifikacija korisnika: Omogućuje funkcionalnost provjere autentičnosti korisnika na siguran način korištenjem OAuth2 za održavanje privatnosti osobnih podataka.

Responsivni dizajn: Povećava dostupnost zahvaljujući tehnologijama React, TypeScript i Tailwind CSS za korisničko sučelje odgovarajućeg izgleda na stolnom računalu i mobilnom uređaju.

Kontrola kvalitete/Pregled i odobrenje sadržaja: Pomaže administratorima u praćenju i odobravanju materijala koje su pridonijeli edukatori.

Mehanizam povratnih informacija korisnika: Mehanizam za bilježenje i integraciju studentskih recenzija kako bi se osigurala kvaliteta i relevantnost preporučenih izvora.

Interaktivni Kalendar: služi za rađenje zadataka namijenjenih za pripremu za ispit.

7. Budućnost projekta

Ušli smo u ovaj projekt s nadom izrade aplikacije koju bi i sami koristili na dnevnoj bazi. Prva stvar koju bi voljeli je koristiti aplikaciju i sami kako bi lakše polagali kolegije na FER-u, te druga pošto svatko može dobiti API key za Gemini, želja nam je ovo nakon predaje pretvoriti u open source repozitorij koji bilo tko besplatno može upogoniti i sam koristiti aplikaciju lokalno (naravno bez dodataka dodanih za ispunjavanje kriterija predmeta)

2. Analiza zahtjeva

2.1 Dionici

U okružju aplikacije, dionici će se sastojati od:

- Vlasnik sustava (naručitelj)
- Studenti
- Edukatori (predavači, instruktori, profesori, učitelji)
- Recenzenti (revieweri)
- Administratori
- Razvojni tim

Aktori, to jest dionici će imati zasebne uloge i mogućnosti, te će svaki moći raditi i obavljati iduće funkcionalnosti:

1. A-1 Student (inicijator)

Student koristi aplikaciju za organizaciju učenja, pregled materijala i dodavanje vlastitih sadržaja.

Student može:

- Uploadati vlastiti sadržaj u obliku PDF-a
 - F-002: Prijenos PDF datoteka s vlastitim sadržajem
- Generirati zadatke i "flashcards"-e iz sadržaja u aplikaciji
 - F-003: Generacija zadataka i flashcards-a pomoću AI asistenta
- Uploadati kalendar u obliku CSV datoteke
 - F-008: Prijenos kalendarja u CSV formatu
- Dodavati datume u kalendar unutar aplikacije
 - F-006: Upravljanje osobnim kalendarom u sustavu
- Davati recenzije javno dostupnim materijalima
 - F-007: Ostavljanje recenzija na javne materijale
- Brisati vlastiti sadržaj sa sustava
 - F-006: Uklanjanje vlastitih datoteka i unosa

2. A-2 Edukator (inicijator)

Edukator koristi aplikaciju za dijeljenje obrazovnih materijala i upravljanje vlastitim sadržajem.

Edukator može:

- Uploadati javno dostupne obrazovne materijale
 - F-004: Prijenos materijala koji su javno dostupni studentima i recenzentima
- Brisati vlastiti sadržaj sa sustava
 - F-009: Uklanjanje vlastitih materijala iz sustava

3. A-3 Recenzent (inicijator)

Reviewer pomaže u održavanju kvalitete sadržaja putem evaluacije javnih materijala.

Reviewer može:

- Reviewati javno dostupan sadržaj

- F-007: Ocjenjivanje i davanje povratnih informacija o javnim materijalima
- Brisati javno dostupan sadržaj sa sustava
 - F-009: Uklanjanje materijala koji ne zadovoljavaju standarde
- Brisati recenzije
 - F-011: Brisanje neprimjerenih recenzija

4. A-4 Administrator (inicijator)

Administrator upravlja edukatorima i korisnicima te osigurava pravilno funkcioniranje aplikacije.

Administrator može:

- Odobravati edukatore
 - F-010: Provjera i odobravanje novih edukatora
- Brisati recenzije
 - F-011: Brisanje neprimjerenih recenzija

5. A-5 Baza podataka (sudionik)

Baza podataka pohranjuje sve podatke o korisnicima, materijale, generirane zadatke, recenzije.

6. A-6 Gemini 1.5 Flash (sudionik)

Služi kao AI pomoću kojeg će se generirati zadatci i flashcards-i pomoću definiranih promptova.

2.2 Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvaćanja
F-001	Sustav omogućuje korisnicima registraciju putem e-mail adrese ili OAuth2 autentifikacije.	Visok	Zahtjev dionika	Korisnik može kreirati račun putem e-maila, primiti potvrdu i uspješno se prijaviti.
F-002	Sustav omogućuje studentima dodavanje vlastitih materijala (PDF, tekstualne datoteke) u osobni profil.	Visok	Zahtjev dionika	Student može uspješno prenijeti materijale i pregledati ih unutar aplikacije.
F-003	Sustav omogućuje generiranje personaliziranih zadataka i "flashcards"-a iz odabranih materijala putem AI asistenta.	Visok	Specifikacija projekta	Na temelju priloženih materijala, AI kreira prilagođeni sadržaj za korisnika.

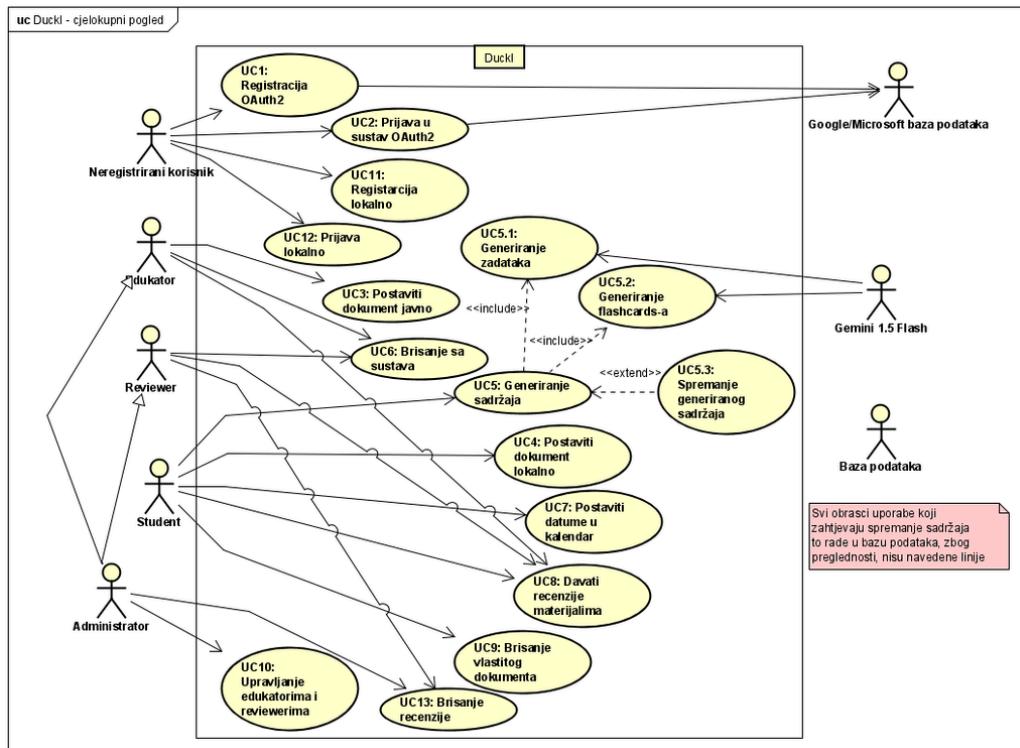
F-004	Edukatori mogu objavljivati dodatne obrazovne resurse dostupne studentima.	Srednji	Povratne informacije korisnika	Edukator može uspješno dodati resurse, a studenti ih mogu pregledati i koristiti.
F-005	Revieweri mogu pregledati i brisati sadržaj objavljen od strane edukatora.	Visok	Zahtjev dionika	Reviewer može uspješno pregledati i potvrditi/odbiti dodane resurse.
F-006	Aplikacija omogućuje studentima pregled rasporeda ispita kroz kalendar.	Srednji	Specifikacija projekta	Student može dodati termine ispita u kalendar i pregledati ih kasnije.
F-007	Sustav omogućuje korisnicima ocjenjivanje materijala.	Srednji	Specifikacija projekta	Korisnik može ostaviti recenziju na materijal, a najbolji materijali se prikazuju kao preporučeni.
F-008	Student može uploadati kalendar u obliku CSV datoteke.	Srednji	Zahtjev dionika	Student može uspješno prenijeti CSV datoteku, a sustav je prikazuje u osobnom kalendaru unutar aplikacije.
F-009	Korisnici mogu brisati vlastiti sadržaj s aplikacije	Visok	Zahtjev dionika	Korisnici mogu uspješno ukloniti svoj preneseni sadržaj iz aplikacije.
F-010	Administrator može odobravati nove edukatore na sustavu.	Visok	Zahtjev dionika	Administrator može pregledati i odobriti/odbiti zahtjeve edukatora za pridruživanje sustavu.
F-011	Reviewer i administrator nadgledaju i brišu nepremjerene i neispravne recenzije.	Nizak	Zahtjev dionika	Reviewer i administrator mogu brisati recenzije.

2.3 Ostali zahtjevi

ID zahtjeva	Opis	Prioritet
NF-1	Sustav treba omogućiti rad više korisnika u stvarnom vremenu	Visok
NF-2	Sustav treba podržavati primarno hrvatsku abecedu, te imati korisničko sučelje na engleskome jeziku.	Srednji
NF-3	Izvršavanje dijela programa koji pristupa bazi podataka ne smije trajati dulje od nekoliko sekundi.	Visok
NF-4	Sustav treba biti implementiran kao web aplikacija, te je pristup pomoću HTTPS protokola.	Visok
NF-5	Sustav mora biti intuitivan, trebalo bi ga se moći koristiti bez uputa i vodiča.	Srednji
NF-6	Sustav treba podržavati nadogradnje bez narušavanja postojećih funkcionalnosti.	Visok
NF-7	Veza s bazom podataka mora biti zaštićena, brza i otporna na vanjske pogreške.	Visok
NF-8	Prikaz i izvođenje sadržaja u aplikaciji mora biti prilagođeno za rad na manjim ekranima mobilnih uređaja.	Nizak
NF-9	Neispravno korištenje korisničkog sučelja ne smije narušiti funkcionalnost i rad sustava.	Visok

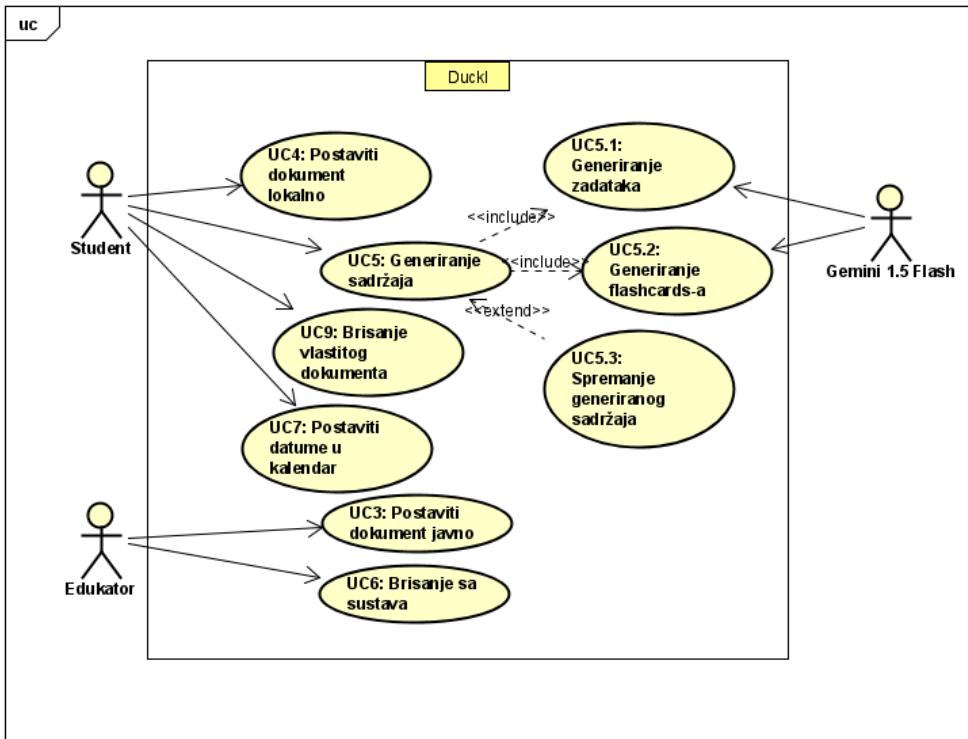
4. Specifikacija zahtjeva sustava

Cjelokupni dijagram obrazaca uporabe



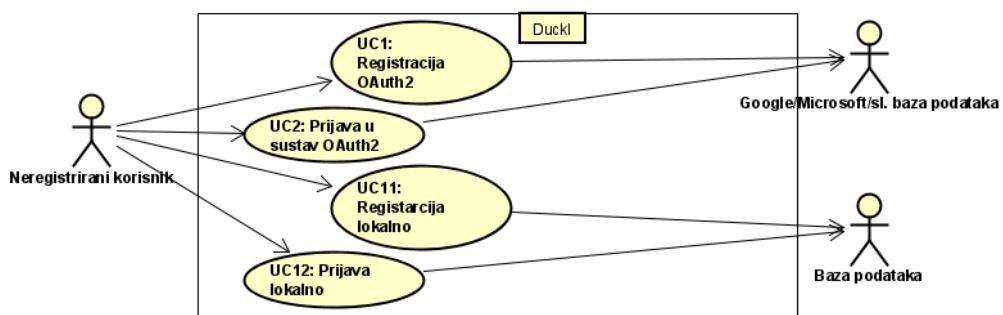
Slika 4.1: Dijagram obrasca uporabe, cjelokupni pogled

1. Visokorazinski dijagram - osnovna funkcionalnost



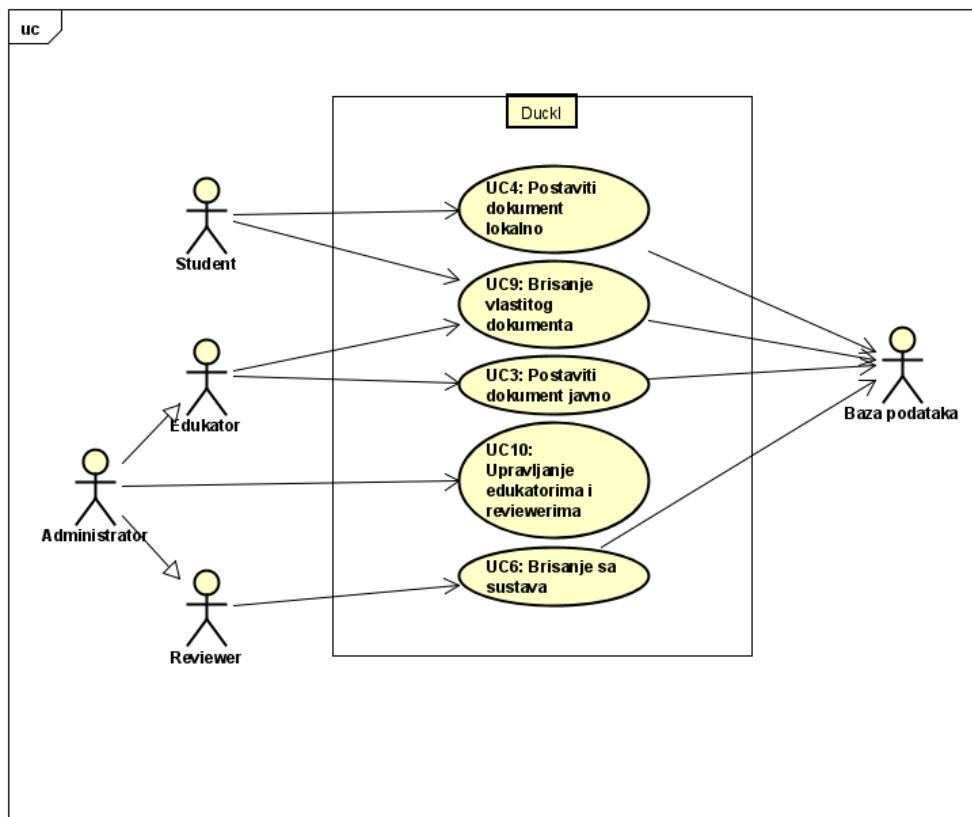
Slika 4.2: Dijagram obrasca uporabe, osnovna funkcionalnost aplikacije, te interakcija sa vanjskim alatom Gemini 1.5 Flash

2. Dijagram obrazaca prijave i registracije u sustav



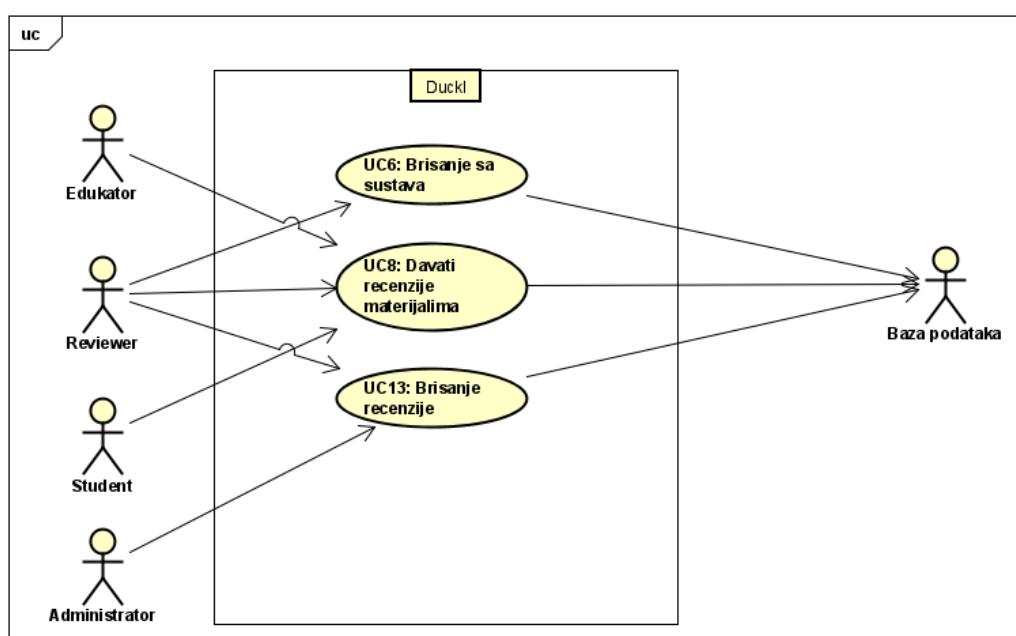
Slika 4.3: Dijagram obrasca uporabe, prijava i registarcija u sustav

3. Dijagram obrazaca uporabe za korisničke uloge



Slika 4.4: Dijagram obrasca uporabe, hijerarhija i odnos korisničkih uloga

4. Dijagram obrazaca uporabe sustava recenzije



Slika 4.5: Dijagram obrasca uporabe, sustav recenzija

Opis obrazaca uporabe

UC1 - Registracija pomoću OAuth2, neregistrirani korisnik

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Registrirati novog korisnika putem OAuth2 autentifikacije (Google, Facebook, itd.)
- **Sudionici:** OAuth2 pružatelj usluge (npr. Google)
- **Preduvjet:** Korisnik nema postojeći račun na sustavu
- **Opis osnovnog tijeka:**
 - a. Korisnik otvara stranicu za registraciju i bira opciju "Registracija putem OAuth2" (referenca na funkcionalni zahtjev F-001)
 - b. Sustav preusmjerava korisnika na stranicu pružatelja usluge (OAuth2).
 - c. Korisnik unosi podatke za autentifikaciju kod pružatelja usluge.
 - d. Pružatelj usluge vraća potvrdu sustavu i sustav registrira korisnika.
 - e. Sustav prikazuje potvrdu uspješne registracije i preusmjerava korisnika na početnu stranicu.
- **Opis mogućih odstupanja:**
 - Ako korisnik prekine postupak autentifikacije kod pružatelja usluge, sustav prikazuje poruku da registracija nije dovršena.
 - Ako pružatelj usluge odbije autentifikaciju, sustav korisniku prikazuje poruku o neuspješnoj registraciji.

UC2 - Prijava pomoću OAuth2, registrirani korisnik

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Prijaviti korisnika u sustav putem OAuth2 autentifikacije
- **Sudionici:** OAuth2 pružatelj usluge (npr. Google)
- **Preduvjet:** Korisnik ima račun na sustavu kreiran putem OAuth2
- **Opis osnovnog tijeka:**
 - a. Korisnik otvara stranicu za prijavu i odabire opciju "Prijava putem OAuth2" (F-001)
 - b. Sustav preusmjerava korisnika na stranicu pružatelja usluge.
 - c. Korisnik unosi podatke za autentifikaciju kod pružatelja usluge.
 - d. Pružatelj usluge vraća potvrdu sustavu.
 - e. Sustav prijavljuje korisnika i prikazuje mu početnu stranicu.
- **Opis mogućih odstupanja:**
 - Ako korisnik prekine postupak autentifikacije, prijava nije dovršena i korisniku se prikazuje poruka.
 - Ako pružatelj usluge odbije autentifikaciju, sustav prikazuje poruku o neuspješnoj prijavi.

UC3 - Postavljanje dokumenta javno, Edukator postavlja javno dostupan dokument

- **Glavni sudionik:** Edukator
- **Cilj:** Objaviti dokument koji će biti javno dostupan studentima
- **Sudionici:** Nema
- **Preduvjet:** Edukator je prijavljen na sustav
- **Opis osnovnog tijeka:**
 - a. Edukator pristupa opciji za dodavanje novih materijala.
 - b. Sustav traži od edukatora da odabere dokument koji želi objaviti (F-004).
 - c. Edukator odabire dokument i označava ga kao javno dostupan.
 - d. Sustav prenosi dokument i postavlja ga u javni katalog materijala.
 - e. Sustav obavještava edukatora o uspješnom prijenosu dokumenta.
- **Opis mogućih odstupanja:**

- Ako dokument nije u ispravnom formatu, sustav prikazuje poruku o pogrešci i traži ispravan format.
- Ako prijenos ne uspije zbog tehničke greške, sustav prikazuje obavijest i omogućuje edukatoru ponovni pokušaj prijenosa.

UC4 - Postavljanje dokumenta za sebe (privatno), Student postavlja dokument na sustav koji samo on može vidjeti

- **Glavni sudionik:** Student
- **Cilj:** Dodati osobni dokument dostupan samo studentu
- **Sudionici:** Nema
- **Preduvjet:** Student je prijavljen na sustav
- **Opis osnovnog tijeka:**
 - a. Student bira opciju za dodavanje novog dokumenta u svoj profil.
 - b. Sustav omogućuje studentu da odabere dokument koji želi dodati (F-002).
 - c. Student odabire dokument i označava ga kao privatni.
 - d. Sustav prenosi dokument i pohranjuje ga u studentov privatni profil.
 - e. Sustav prikazuje potvrdu o uspješnom prijenosu dokumenta.
- **Opis mogućih odstupanja:**
 - Ako dokument nije u ispravnom formatu, sustav obavještava studenta i traži novi pokušaj s ispravnim formatom.
 - Ako prijenos dokumenta ne uspije zbog greške, sustav daje studentu mogućnost ponovnog pokušaja.

UC5 - Generiranje sadržaja, student generira sadržaj pomoću vanjskog entiteta (sudionika) Gemini 1.5 Flash

- **Glavni sudionik:** Student
- **Cilj:** Generirati personalizirane zadatke i "flashcards" iz postojećih materijala
- **Sudionici:** AI asistent Gemini 1.5 Flash
- **Preduvjet:** Student je prenio materijale koje želi koristiti za generiranje sadržaja
- **Opis osnovnog tijeka:**
 - a. Student bira opciju za generiranje zadataka i "flashcards" na temelju postojećih materijala (F-003)
 - b. Sustav šalje materijale vanjskom entitetu Gemini 1.5 Flash za obradu.
 - c. Gemini 1.5 Flash generira zadatke i "flashcards" na temelju predanih materijala.
 - d. Sustav prikazuje studentu generirane zadatke i "flashcards".
 - e. Student može pregledati i spremiti generirani sadržaj u svoj profil (F-002)
- **Opis mogućih odstupanja:**
 - Ako se dogodi pogreška u komunikaciji s Gemini 1.5 Flash, sustav prikazuje poruku o pogrešci i omogućuje ponovni pokušaj.
 - Ako generirani sadržaj ne zadovoljava korisnikove potrebe, student može zatražiti ponovnu generaciju.

UC6 - Brisanje sa sustava, Reviewer može izbrisati javno dostupan dokument koji ne zadovoljava standarde

- **Glavni sudionik:** Reviewer
- **Cilj:** Izbrisati dokument koji ne zadovoljava standarde kvalitete
- **Sudionici:** Nema
- **Preduvjet:** Reviewer je prijavljen na sustav
- **Opis osnovnog tijeka:**
 - a. Reviewer pregledava javno dostupne dokumente.
 - b. Reviewer identificira dokument koji ne zadovoljava standarde (F-011).
 - c. Reviewer odabire opciju za brisanje dokumenta.
 - d. Sustav traži potvrdu brisanja.
 - e. Nakon potvrde, sustav uklanja dokument iz javnog kataloga i obavještava reviewera o uspješnom brisanju.

- **Opis mogućih odstupanja:**

- Ako reviewer greškom odabere krivi dokument za brisanje, može otkazati brisanje na ekranu s potvrdom.
- Ako brisanje ne uspije zbog tehničke pogreške, sustav prikazuje obavijest i omogućuje ponovni pokušaj.

UC7 - Postavljanje datuma u kalendar, student može postaviti datume u kalendar pomoću CSV datoteke ili ručnim unosom

- **Glavni sudionik:** Student

- **Cilj:** Postaviti datume u kalendar za ispite, zadatke i druge događaje

- **Sudionici:** Nema

- **Preduvjet:** Student je prijavljen na sustav

- **Opis osnovnog tijeka:**

- a. Student bira opciju za postavljanje datuma u kalendar.
- b. Sustav nudi studentu opciju za prijenos CSV datoteke ili ručni unos datuma (F-008)
- c. Ako student odabere CSV datoteku, sustav analizira i automatski unosi datume iz datoteke.
- d. Ako student odabere ručni unos, sustav prikazuje formular za unos datuma i događaja (F-004)
- e. Sustav sprema postavljene datume u studentov osobni kalendar.

- **Opis mogućih odstupanja:**

- Ako CSV datoteka nije u ispravnom formatu, sustav prikazuje poruku o pogrešci i traži ispravan format.
- Ako ručno uneseni datumni nisu valjani, sustav traži ponovni unos ispravnog datuma.

UC8 - Davanje recenzija materijalima, Student, Reviewer i Edukator mogu davati recenzije materijalima

- **Glavni sudionik:** Student, Reviewer ili Edukator

- **Cilj:** Ostaviti recenziju na javno dostupni materijal

- **Sudionici:** Nema

- **Preduvjet:** Korisnik je prijavljen na sustav i ima pristup javnim materijalima

- **Opis osnovnog tijeka:**

- a. Korisnik odabire materijal koji želi recenzirati.
- b. Sustav prikazuje opciju za dodavanje recenzije (F-007).
- c. Korisnik unosi recenziju (tekstualni komentar i ocjenu).
- d. Sustav pohranjuje recenziju uz materijal.
- e. Sustav prikazuje potvrdu o uspješnom unosu recenzije.

- **Opis mogućih odstupanja:**

- Ako recenzija sadrži nedopuštene riječi, sustav prikazuje obavijest i traži izmjenu.
- Ako unos recenzije ne uspije zbog tehničke pogreške, sustav omogućuje korisniku ponovni pokušaj.

UC9 - Brisanje vlastitog dokumenta, student ili edukator mogu izbrisati dokument koji su oni sami postavili

- **Glavni sudionik:** Student ili Edukator

- **Cilj:** Ukloniti vlastiti dokument sa sustava

- **Sudionici:** Nema

- **Preduvjet:** Korisnik je prijavljen na sustav i dokument je pohranjen u njegovom profilu

- **Opis osnovnog tijeka:**

- a. Korisnik pristupa svom profilu i odabire dokument koji želi izbrisati (F-009).
- b. Korisnik bira opciju za brisanje dokumenta.
- c. Sustav traži potvrdu za brisanje.

- d. Nakon potvrde, sustav uklanja dokument iz korisnikovog profila.
- e. Sustav prikazuje potvrdu o uspješnom brisanju dokumenta.

- **Opis mogućih odstupanja:**

- Ako korisnik greškom odabere krivi dokument, može otkazati brisanje na ekranu s potvrdom.
- Ako brisanje ne uspije zbog tehničke greške, sustav omogućuje korisniku ponovni pokušaj.

Evo prijedloga razrade nekoliko obrazaca uporabe prema zadanom predlošku:

UC10 - Upravljanje edukatorima, Administrator može odobriti edukatore

- **Glavni sudionik:** Administrator
- **Cilj:** Odobriti nove edukatore za pristup sustavu
- **Sudionici:** Edukatori
- **Preduvjet:** Administrator je prijavljen na sustav i ima prava pristupa za upravljanje edukatorima
- **Opis osnovnog tijeka:**
 - a. Administrator pristupa sučelju za upravljanje edukatorima (F-010)
 - b. Sustav prikazuje popis novih edukatora koji čekaju odobrenje.
 - c. Administrator pregledava informacije o edukatorima koji su podnijeli zahtjev.
 - d. Administrator odabire opciju za odobravanje ili odbijanje edukatora.
 - e. Sustav ažurira status odabranih edukatora i obavještava ih o odluci.
- **Opis mogućih odstupanja:**
 - Ako edukator nije ispunio sve potrebne podatke, sustav prikazuje administratoru upozorenje.
 - Ako sustav ne može poslati obavijest edukatoru, prikazuje se opcija za kasnije slanje.

UC11 - Registracija pomoću osobnog e-maila (lokalno), neregistrirani korisnik

- **Glavni sudionik:** Neregistrirani korisnik
- **Cilj:** Registrirati novog korisnika putem osobnog e-maila
- **Sudionici:** Nema
- **Preduvjet:** Korisnik nema postojeći račun na sustavu
- **Opis osnovnog tijeka:**
 - a. Korisnik otvara stranicu za registraciju i bira opciju za registraciju putem e-maila (F-001).
 - b. Sustav traži od korisnika unos e-mail adrese, lozinke i dodatnih podataka.
 - c. Korisnik unosi tražene podatke i potvrđuje registraciju.
 - d. Sustav šalje korisniku e-mail s poveznicom za potvrdu registracije.
 - e. Korisnik otvara poveznicu iz e-maila, a sustav potvrđuje uspješnu registraciju i omogućava prijavu.
- **Opis mogućih odstupanja:**
 - Ako korisnik unese već korištenu e-mail adresu, sustav prikazuje poruku o grešci i traži drugu adresu.
 - Ako e-mail za potvrdu nije ispravno poslan, sustav omogućuje korisniku da ponovo pošalje e-mail za potvrdu.

UC12 - Prijava pomoću osobnog e-maila (lokalno), registrirani korisnik

- **Glavni sudionik:** Registrirani korisnik
- **Cilj:** Prijaviti korisnika putem osobnog e-maila
- **Sudionici:** Nema
- **Preduvjet:** Korisnik ima račun kreiran putem e-maila
- **Opis osnovnog tijeka:**
 - a. Korisnik otvara stranicu za prijavu i unosi svoju e-mail adresu i lozinku (F-001).

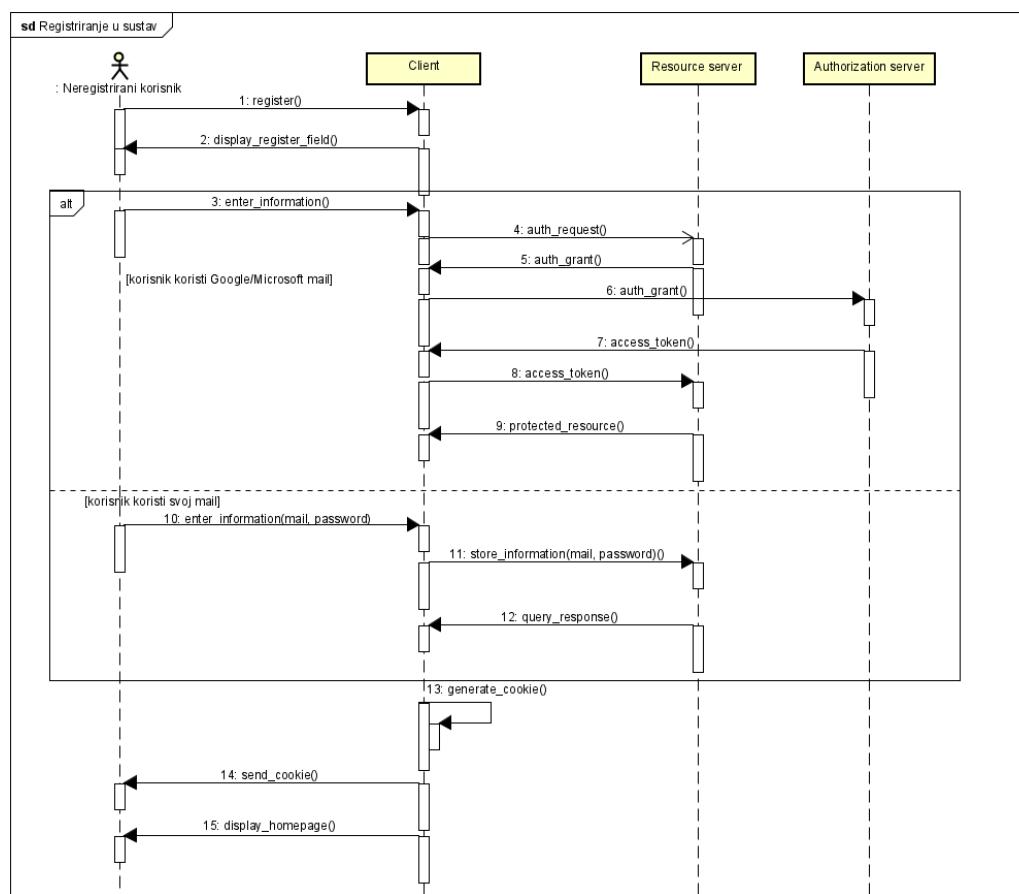
- b. Sustav provjerava unesene podatke.
- c. Ako su podaci ispravni, sustav prijavljuje korisnika i prikazuje mu početnu stranicu.
- d. Korisnik ima pristup svim opcijama aplikacije prema svojim pravima.

- **Opis mogućih odstupanja:**

- Ako korisnik uneše pogrešne podatke, sustav prikazuje poruku o neuspješnoj prijavi i omogućuje ponovni unos.
- Ako je korisnički račun privremeno zaključan zbog previše neuspjelih pokušaja, sustav prikazuje obavijest i upućuje korisnika na podršku.

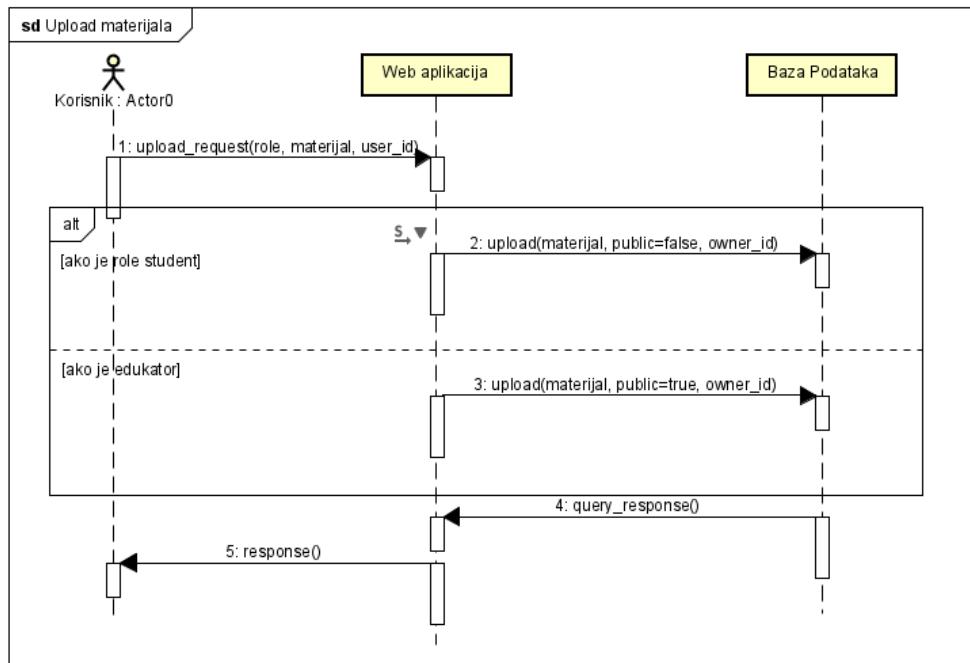
Sekvencijski dijagrami

Registriranje u sustav



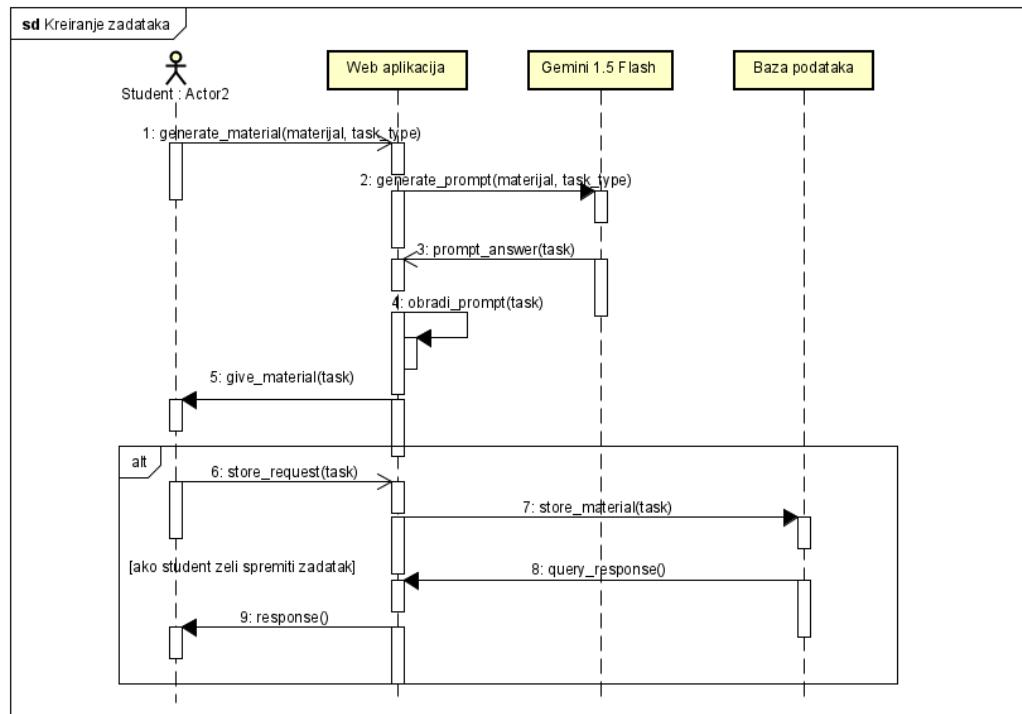
Slika 4.6: Sekvencijski dijagram, registracija u sustav

Postavljanje materijala na sustav



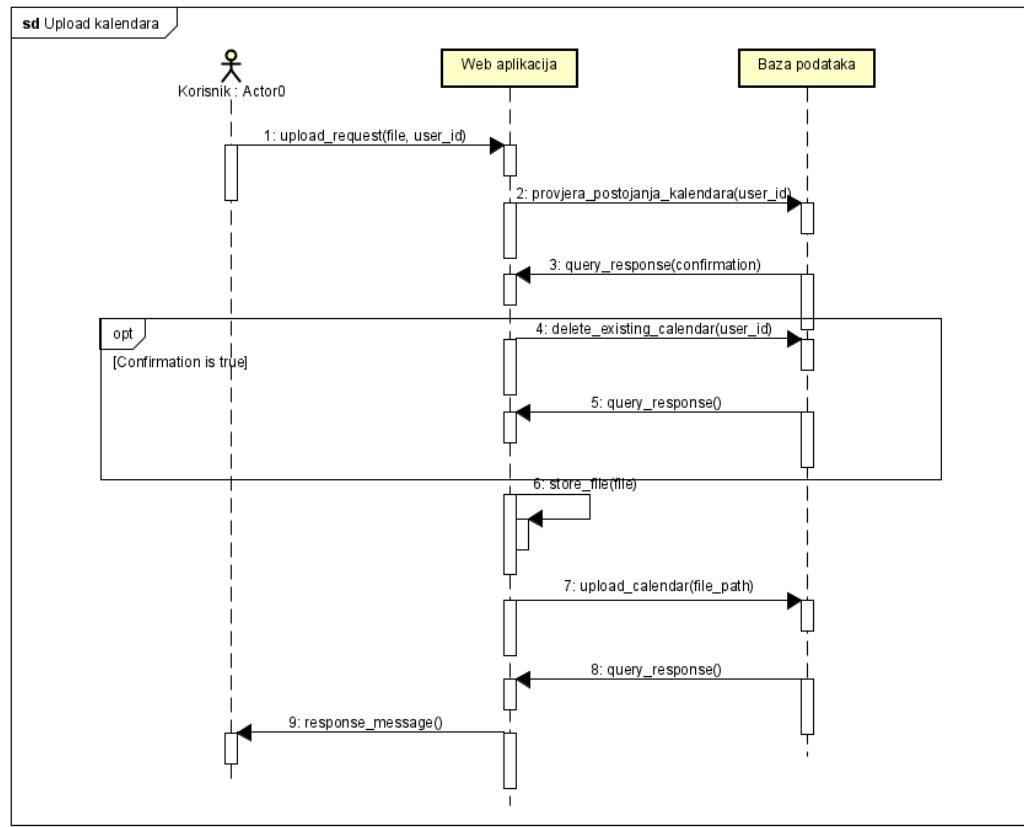
Slika 4.7: Sekvencijski dijagram, postavljanje materijala u sustav

Generiranje zadatka



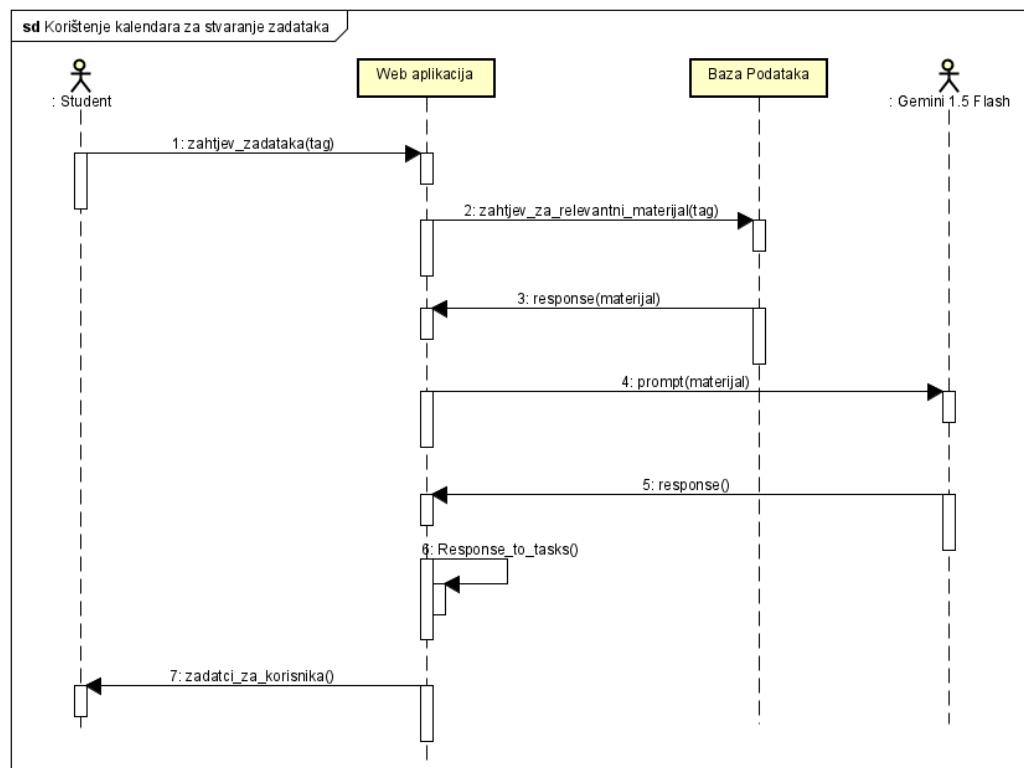
Slika 4.8: Sekvencijski dijagram, generiranje zadatka

Postavljanje kalendaru u sustav



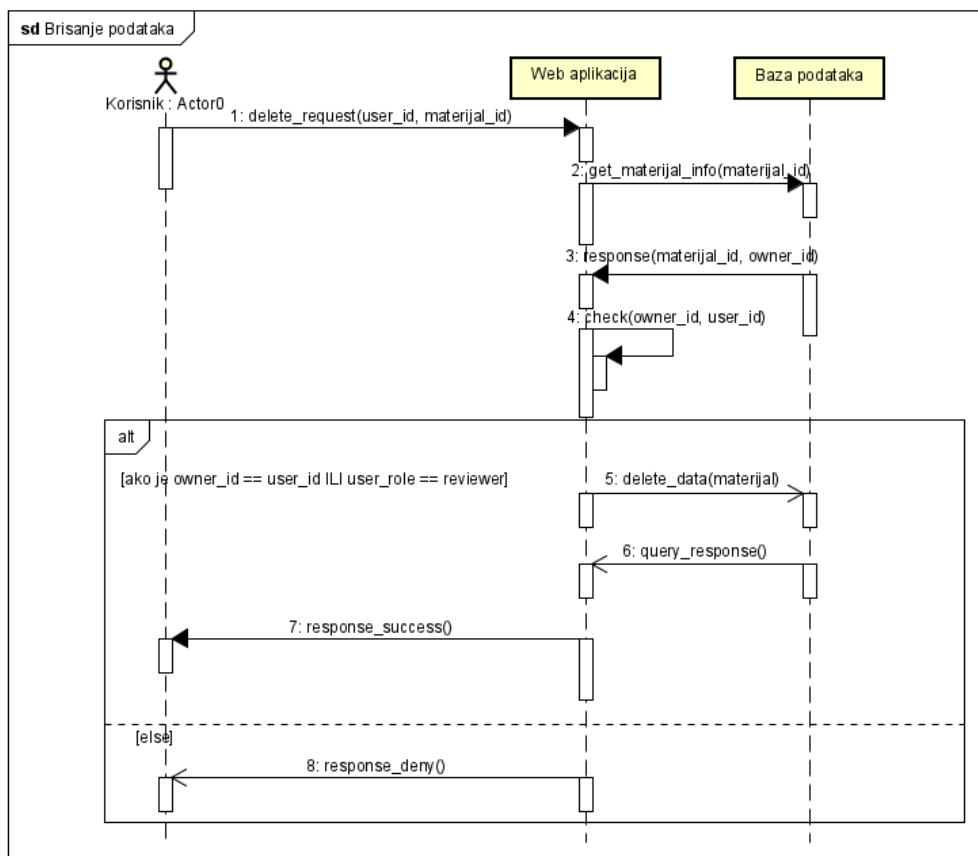
Slika 4.9: Sekvencijski dijagram, upload kalendaru

Korištenje kalendaru za generiranje zadatka



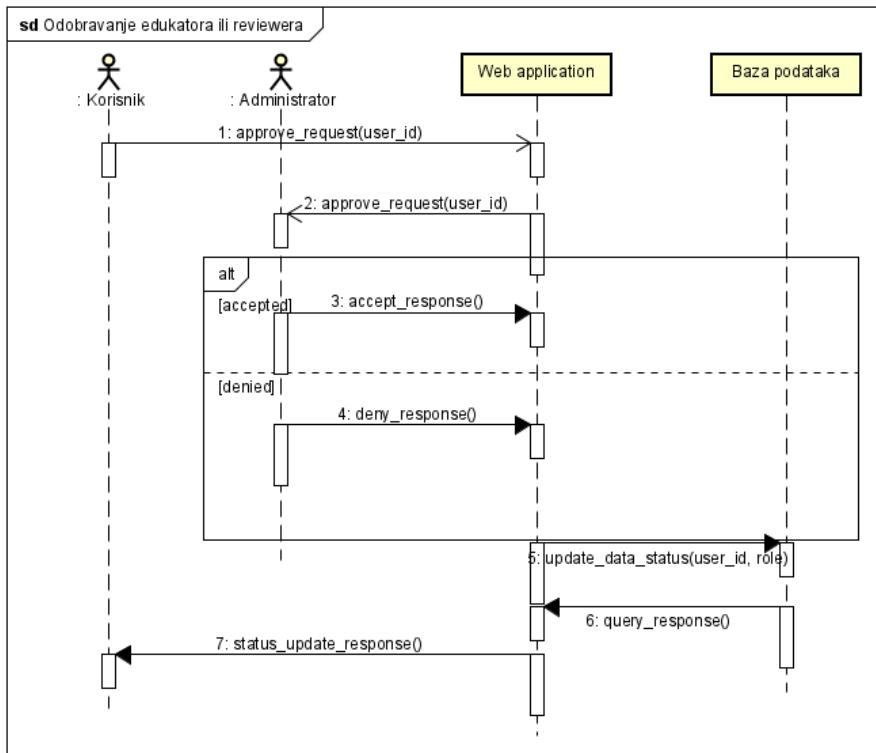
Slika 4.10: Sekvencijski dijagram, mogućnost kalendaru za učenje

Brisanje materijala



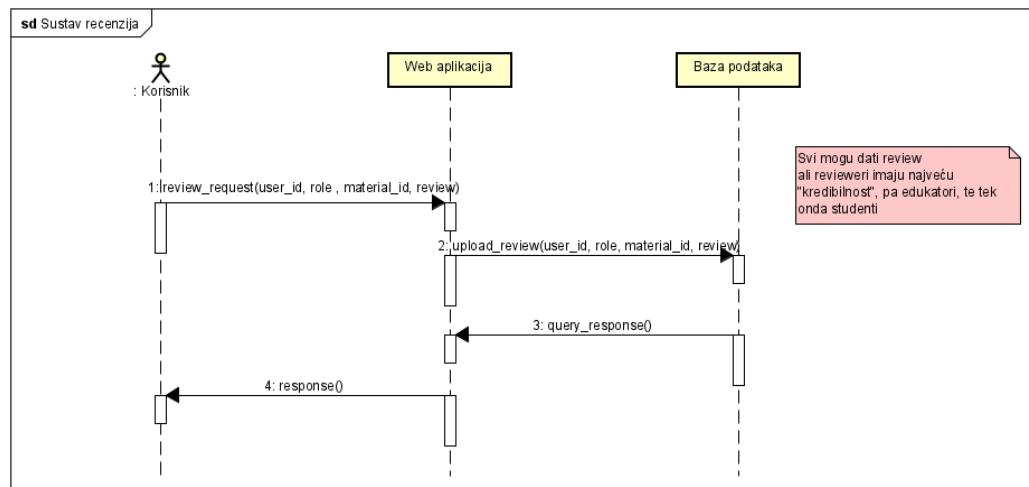
Slika 4.11: Sekvencijski dijagram, brisanje materijala

Odobravanje edukatora ili reviewera



Slika 4.12: Sekvencijski dijagram, approval edukatora

Sustav recenzija



Slika 4.13: Sekvencijski dijagram, sustav recenzija

Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

ID obrasca uporabe	Naziv	Obuhvaćeni funkcionalni zahtjevi
UC1	Registracija pomoću OAuth2, neregistrirani korisnik	F-1

UC2	Prijava pomoću OAuth2, registrirani korisnik	F-1
UC3	Postavljanje dokumenta javno, Edukator postavlja javno dostupan dokument	F-4
UC4	Postavljanje dokumenta za sebe (privatno), Student postavlja dokument na sustav koji samo on može vidjeti	F-2
UC5	Generiranje sadržaja, student generira sadržaj pomoću vanjskog entiteta (sudionika) Gemini 1.5 Flash	F-3
UC6	Brisanje sa sustava, Reviewer može izbrisati javno dostupan dokument koji ne zadovoljava standarde	F-5, F-11
UC7	Postavljanje datuma u kalendar, student može postaviti datume u kalendar pomoću CSV datoteke ili ručnim unosom	F-6, F-8
UC8	Davanje recenzija materijalima, Student, Reviewer i Edukator mogu davati recenzije materijalima	F-7
UC9	Brisanje vlastitog dokumenta, student ili edukator mogu izbrisati dokument koji su oni sami postavili	F-9
UC10	Upravljanje edukatorima, Administrator može odobriti edukatore	F-10
UC12	Prijava pomoću osobnog e-maila (lokalno), registrirani korisnik	F-1

5. Arhitektura i dizajn sustava

dio 1. revizije

Arhitektura sustava

Cilj ovog poglavlja je pružiti jasan, sažet i strukturiran pregled arhitekture Duckl programskog sustava u razvoju, uz razrađivanje ključnih komponenata i njihovih međusobnih odnosa. Ova dokumentacija treba omogućiti svim sudionicima projekta potpuno razumijevanje arhitekture sustava, načina implementacije, podjele odgovornosti te kako sustav funkcionira kao cjelina. Uključivanje odgovarajućih dijagrama pomaže u kvalitetnom dokumentiranju i vizualizaciji strukture sustava i njegovih ključnih komponenti.

Opis arhitekture

Ovdje je opisan pregled arhitekture na visokoj razini, uključujući stil arhitekture sustava.

- Stil arhitekture: Duckl programski sustav koristi monolitnu arhitekturu sustava, te koristi principe „povećaj koheziju“ (nadogradnja na podjeli pa vladaj) pošto su međusobno povezani elementi grupirani (npr. kontroleri), „smanji međuovisnost“ pošto su komponente uglavnom „single-task“, tj. fokusiraju se na jedan zadatak te se mogu samostalno testirati, „povećaj uporabu postojećeg“ pošto su dijelovi programskog koda ponovno uporabljivi, „oblikuj za ispitivanje“ pošto je kod strukturiran na način pogodan za testiranje što je ključno za programsku potporu koja koristi monolitnu arhitekturu, „oblikuj konzervativno“ pošto se rubni slučajevi prate, te se u njihovim slučajevima izbacuju iznimke. Pošto su nam resursi bili ograničeni, umjesto klijent - server arhitekture odlučili smo se za monolitnu arhitekturu kako bi olakšali implementiranje aplikaciju na neku platformu.
- Podsustavi: Podsustavi koji se koriste u backend dijelu su servisi, kontroleri, modeli, podaci. Ovisno o vrsti kontrolera, kontroleri mogu vratiti stranicu ili neke podatke s api rute. Servisi se koriste za implementaciju servisa koje koriste kontroleri. Podsustavi modeli i podaci su međusobno povezani pošto oni rješavaju bazu podataka. Unutar modela su opisane tablice koje se koriste, dok unutar podatkovnog podsustava možemo pronaći datoteke koje ne možemo spremiti u bazu organizirane po direktorijima, te ApplicationDbContext klasu koja upravlja bazom podataka.
- Preslikavanje na radnu platformu: Trenutačan deployment održan lokalno na računalu preko port-forwardinga za svrhe prezentacije početne funkcionalnosti.
- Spremista podataka: Relacijska SQLite baza podataka u kojoj se spremaju sve informacije za aplikaciju (user login info, user data, public data, other app data). Datoteke poput PDF-ova i CSV-a koji će se koristiti za prikaz korisničkih informacija, spremaju se direktno na datotečni sustav, te u bazi podataka će samo biti putanje koje pokazuju na te datoteke.
- Mrežni protokoli: HTTPS
- Globalni upravljački tok: Nakon što korisnik stisne neki gumb ili preko neke druge komponente zatraži nekakve podatke, on poziva određenu rutu preko kontrolera zaduženog za tu rutu. Prvo se provjerava ima li korisnik pristup određenoj ruti. Ako se utvrdi da ima, njegov zahtjev se proslijeđuje bazi podataka te ovisno o vrsti zahtjeva (GET, POST, ili nešto drugo), u bazi se dohvataju podaci/uređuju podaci. Nakon rada s podacima korisniku se preko kontrolera uredi stranica s izmjenama koje je zatražio.
- Sklopopskopogramske zahtjevi: Opišite potrebne tehničke zahtjeve vezane uz sklopovlje i program, uključujući podršku za specifične operativne sustave, procesore, memoriju i druge komponente.

Obrazloženje odabira arhitekture

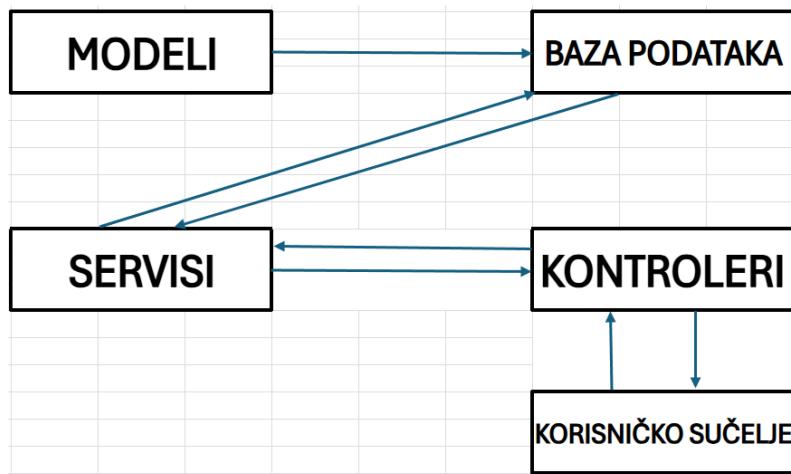
Kako je navedeno u prethodnim točkama, Duckl programski sustav koristi monolitnu arhitekturu sustava, te koristi principe „povećaj koheziju“ (nadogradnja na podjeli pa vladaj) pošto su međusobno povezani elementi grupirani (npr. kontroleri), „smanji međuovisnost“ pošto su komponente uglavnom „single-task“, tj. fokusiraju se na jedan zadatak te se mogu samostalno testirati, „povećaj uporabu postojećeg“ pošto su dijelovi programskog koda ponovno uporabljivi, „oblikuj za ispitivanje“ pošto je kod strukturiran na način pogodan za testiranje što je ključno za programsku potporu koja koristi monolitnu arhitekturu, „oblikuj konzervativno“ pošto se rubni slučajevi prate, te se u njihovim slučajevima izbacuju iznimke. Za monolitnu arhitekturu smo se odlučili radi njene jednostavnosti pošto sadrži sve što je nama

potrebno za aplikaciju (autorizaciju, prezentaciju, „biznis“ logiku, podatkovni sloj, jednostavne notifikacije). Također za monolitnu arhitekturu smo se odlučili jer programska potpora rađena u ovoj arhitekturi obično ima komponente povezane u jednu veliku aplikaciju. Time se iznimno olakšava preslikavanje na radnu platformu, što nam je predstavljalo najveće ograničenje u radu na aplikaciji. Prethodno smo razmatrali i klijent - server arhitekturu, ali zbog otežanog preslikavanja na radnu platformu odlučili smo se ipak za monolitnu arhitekturu.

Jednostavna skica monolitne arhitekture (vlastita slika):



Inicijalni dijagram visoke razine (vlastita slika):



Organizacija sustava na visokoj razini

Organizacija sustava na najvišoj razini apstrakcije:

- Baza podataka: Koristi se SQLite baza podataka. U njoj se pohranjuju kalendarji, pdf materijali i ostali važni podaci za korisnike, te se također koristi za dodjeljivanje uloga korisnika i autorizaciju.
- Datotečni sustav: Unutar Data > Files direktorija pohranjujemo podatke koje se ne mogu pohraniti u bazu podataka. U bazu podataka se pohranjuju rute na datoteku spremljenu u Data > Files, umjesto same datoteke.
- Grafičko sučelje: Aplikacija je web aplikacija koja koristi različite rute za prikazivanje stranica te dohvaćanje i manipulaciju podataka u bazi.

Organizacija aplikacije

Organizacija aplikacije na složenijoj razini, uključujući strukturu slojeva i komponenata aplikacije:

- Frontend i Backend slojevi: Frontend sloj koristi viewove za prikaz podataka. Backend koristi modele, servise i kontrolere za rad s rutama i bazom.
- MVC arhitektura: viewovi su zaduženi za prikaz korisničkog sučelja, servisi su zaduženi za logiku, dok su kontroleri zaduženi za pozivanje servisa.

Baza podataka

Za bazu podataka izabran je SQLite zbog njegove jednostavnosti. SQLite je library jezika C koji implementira mali, brzi, samostalni, visokopouzdani i potpuni SQL database engine. Cijela baza nalazi se unutar jedne `app.db` datoteke u `root` direktoriju, te se njoj može pristupati bez potrebe za serverom, što pojednostavljuje upravljanje bazom podataka i čini je visoko prenosivom na različitim platformama. SQLite se široko koristi u ugrađenim sustavima, mobilnim aplikacijama i desktop aplikacijama.

Opis tablica

`sqlite_master`

Property	Type	Description
type	text	Vrsta objekta baze podataka kao što je tablica, indeks, okidač ili prikaz.
name	text	Naziv objekta baze podataka.
tbl_name	text	Naziv tablice s kojom je povezan objekt baze podataka.
rootpage	int	Root stranica.
sql	text	SQL korišten da se napravi objekt baze podataka.

`_EFMigrationsHistory`

Property	Type	Description
ProductVersion	text	Verzija EF Core koja je korištena za primjenu migracija.
MigrationId	text	Jedinstveni identifikator za migraciju.

`_EFMigrationsHistory` primary key (`MigrationId`)

`sqlite_sequence`

Property	Type	Description
name	unknown	Tablica asocirana s AUTOINCREMENT stupcem.
seq	unknown	Zadnji redni broj korišten u stupcu AUTOINCREMENT.

`AspNetUsers`

Property	Type	Description
AccessFailedCount	integer	Broj neuspjelih pokušaja prijave. Ovo se koristi za funkciju

		zaključavanja.
ConcurrencyStamp	text	Jedinstvena vrijednost koja se mijenja kad god se korisnički profil ažurira, koristi se za optimističnu kontrolu istovremenosti, osiguravajući integritet podataka. U scenarijima u kojima postoji više pokušaja izmjene istog korisničkog zapisa istovremeno, ConcurrencyStamp osigurava da promjene nisu u sukobu.
Email	text	Adresa e-pošte korisnika.
EmailConfirmed	integer	Booleova vrijednost označava je li korisnikova adresa e-pošte potvrđena.
LockoutEnabled	integer	Booleova vrijednost označava može li se račun zaključati za korisnika.
LockoutEnd	text	Datum i vrijeme završetka zaključavanja (ako je korisnik trenutačno zaključan).
NormalizedEmail	text	Normalizirana verzija e-pošte za dosljedno postavljanje upita, obično velikim slovima. Ovo se koristi u usporedbama bez obzira na velika i mala slova.
NormalizedUserName	text	Normalizirana verzija korisničkog imena za dosljedno postavljanje upita.
PasswordHash	text	Hash verzija korisničke zaporce. AspNetCore Identity koristi siguran mehanizam za hashanje lozinki.
PhoneNumber	text	Telefonski broj korisnika.
PhoneNumberConfirmed	integer	Booleova vrijednost označava je li telefonski broj korisnika potvrđen.
SecurityStamp	text	Slučajna vrijednost označava jesu li se promijenile informacije o korisniku povezane sa sigurnošću. Na primjer, mijenja se kada korisnik promjeni lozinku, poništi lozinku ili doda vanjsku prijavu. Njegova primarna svrha je poništiti sve postojeće sesije ili kolačiće kada se promijene informacije povezane sa

		sigurnošću. Ovo je sigurnosna mjera kojom se osigurava da stare sesije više nisu valjane ako su vjerodajnice ugrožene i zatim promijenjene.
TwoFactorEnabled	integer	Booleova vrijednost označava je li dvofaktorska provjera autentičnosti omogućena za korisnika.
UserName	text	Korisničko ime korisnika. Jedinstven je i koristi se za identifikaciju.
Id	text	Primarni ključ za korisnika. Jedinstveni identifikator za svakog korisnika, obično GUID niz.

AspNetUsers primary key(Id)

AspNetUserTokens

Property	Type	Description
Value	text	Vrijednost tokena.
UserId	text	ID korisnika iz tablice AspNetUsers. Dio kompozitnog primarnog ključa.
LoginProvider	text	Ovaj stupac navodi naziv pružatelja koji je generirao token. Na primjer, to može biti interni pružatelj (poput sustava za poništavanje zaporce ili sustava za potvrdu e-pošte) ili vanjski pružatelj autentifikacije (kao što su Google, Facebook itd.) ako je token povezan s vanjskom autentifikacijom. Dio je složenog primarnog ključa.
Name	text	Ovaj stupac pohranjuje naziv tokena, kao što je token za potvrdu e-pošte, token za poništavanje lozinke ili token za dvofaktornu provjeru autentičnosti. To može biti nešto poput PasswordReset, EmailConfirmation, AccessToken, itd. Također je dio složenog primarnog ključa.

AspNetUserTokens primary key(UserId, LoginProvider, Name)

AspNetRoles

Property	Type	Description
ConcurrencyStamp	text	Jedinstveni pečat koji upravlja istodobnim uređivanjem istog zapisa uloge. Pomaže u održavanju integriteta podataka osiguravajući da istodobne operacije ne prebrišu promjene.
Name	text	Naziv uloge. Ovo je čovjeku čitljiv naziv koji se koristi u našem aplikacijskom kodu kada dodjeljujemo uloge korisnicima ili autoriziramo korisnike na temelju njihovih uloga.
NormalizedNome	text	Normalizirana verzija polja Name, obično uppercase verzija imena uloge i koristi se u usporedbama bez obzira na velika i mala slova.
Id	text	Primarni ključ za ulogu. Jedinstveni identifikator za svaku ulogu, obično GUID niz.

AspNetRoles primary key (Id)

AspNetUserRoles

Property	Type	Description
UserId	text	ID korisnika iz tablice AspNetUsers. Dio kompozitnog primarnog ključa. Odgovara stupcu Id u tablici AspNetUsers. Djeluje kao strani ključ koji se povezuje s tablicom AspNetUsers.
RoleId	text	ID uloge iz tablice AspNetRoles. Dio kompozitnog primarnog ključa. Odgovara stupcu Id u tablici AspNetRoles. Djeluje kao strani ključ koji se povezuje s tablicom AspNetRoles.

AspNetUserRoles primary key (UserId, RoleId)

AspNetUserRoles foreign key (UserId)

AspNetUserRoles foreign key (RoleId)

AspNetRoleClaims

Property	Type	Description
ClaimType	text	Ovaj stupac pohranjuje vrstu zahtjeva, kao što je dozvola, razina pristupa ili bilo koja druga vrsta koja ima smisla u kontekstu vaše aplikacije.
ClaimValue	text	Stvarna vrijednost tražbine. Na primjer, ako je vrsta zahtjeva Dozvola, vrijednost zahtjeva može biti Edit_User ili View_Reports, itd.
RoleId	text	ID uloge povezan s ovim zahtjevom. Strani ključ koji se povezuje sa stupcem Id u tablici AspNetRoles.
Id	integer	Ovo je primarni ključ tablice.

AspNetRoleClaims primary key (Id)

AspNetRoleClaims foreign key (RoleId)

AspNetUserClaims

Property	Type	Description
ClaimType	text	Vrsta zahtjeva (npr. "datum rođenja").
ClaimValue	text	Vrijednost zahtjeva (npr. "1980-01-01").
UserId	text	ID korisnika povezanog s ovim zahtjevom. Djeluje kao strani ključ koji povezuje stupac Id u tablici AspNetUsers.
Id	integer	Primarni ključ za zahtjev korisnika.

AspNetUserLogins

Property	Type	Description
ProviderDisplayName	text	Ovaj izborni stupac može pohraniti ime za prikaz davatelja usluge prijave (npr. "Google" umjesto „https://google.com“). Uglavnom se koristi za potrebe prikaza u korisničkom sučelju.
UserId	text	Lokalni korisnički ID koji je povezan s ovom prijavom. Djeluje kao strani ključ koji se povezuje sa stupcem Id u tablici AspNetUsers.

		Identificira korisnika povezanog s određenom prijavom.
LoginProvider	text	Ovaj stupac pohranjuje naziv vanjskog pružatelja autentifikacije (npr. Google, Facebook, Microsoft itd.). To je dio složenog primarnog ključa za tablicu.
ProviderKey	text	Ovaj stupac pohranjuje jedinstveni identifikator od davaljelja usluge prijave za korisnika. Na primjer, kada se korisnik prijavi pomoću Googlea, ovo polje će pohraniti jedinstveni ID koji je korisniku dodijelio Google. Također je dio složenog primarnog ključa.

AspNetUserLogins primary key (LoginProvider, ProviderKey)

AspNetUserLogins foreign key (UserId)

Calendars

Property	Type	Description
CsvPath	text	Putanja na kojoj je spremljen .csv file.
CalendarId	integer	Id kalendara.

Calendars primary key (CalendarId)

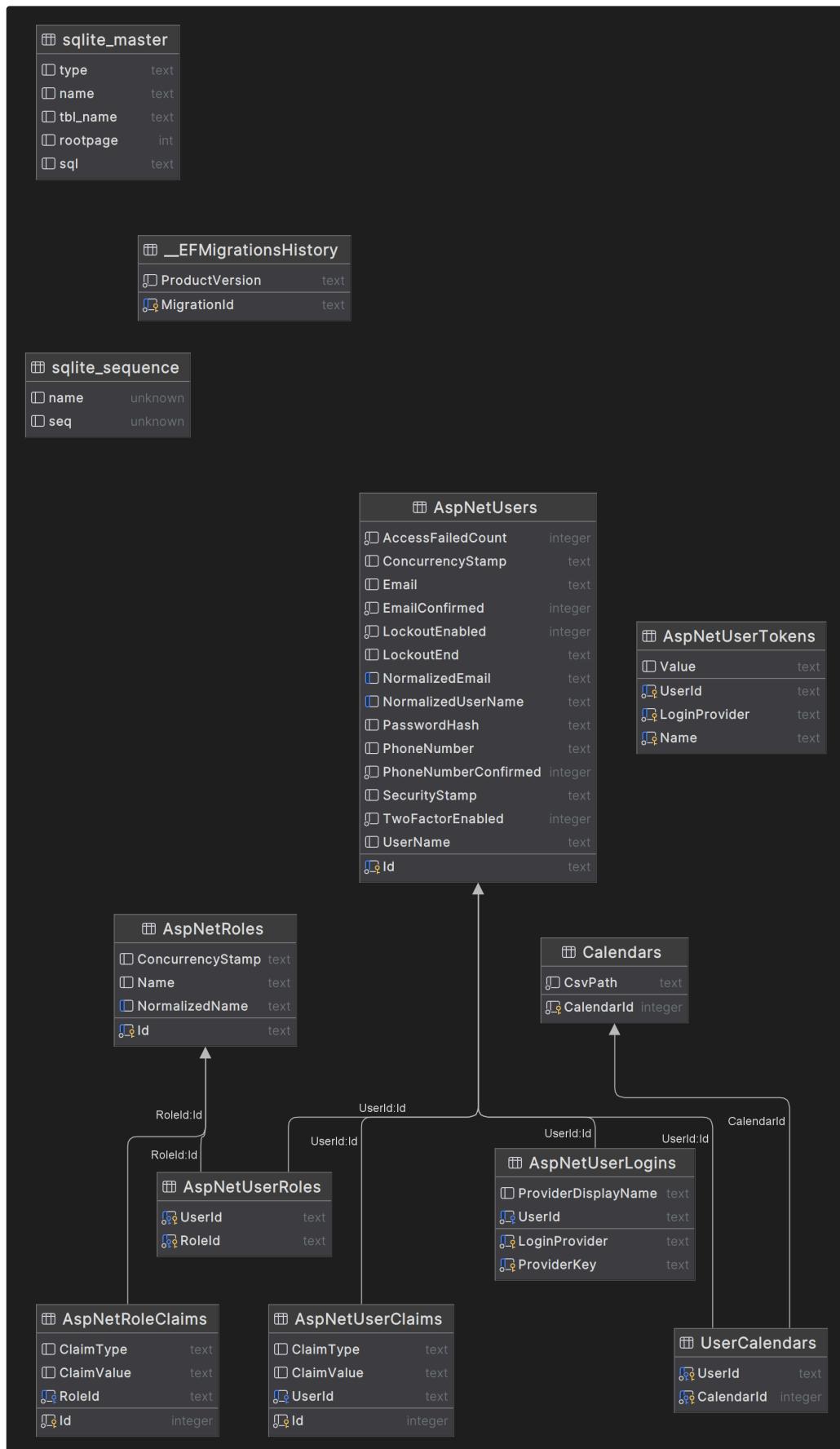
UserCalendars

Property	Type	Description
UserId	text	Id korisnika koji je foreign key na tablicu AspNetUsers.
CalendarId	integer	Id kalendara koji je foreign key na tablicu Calendars.

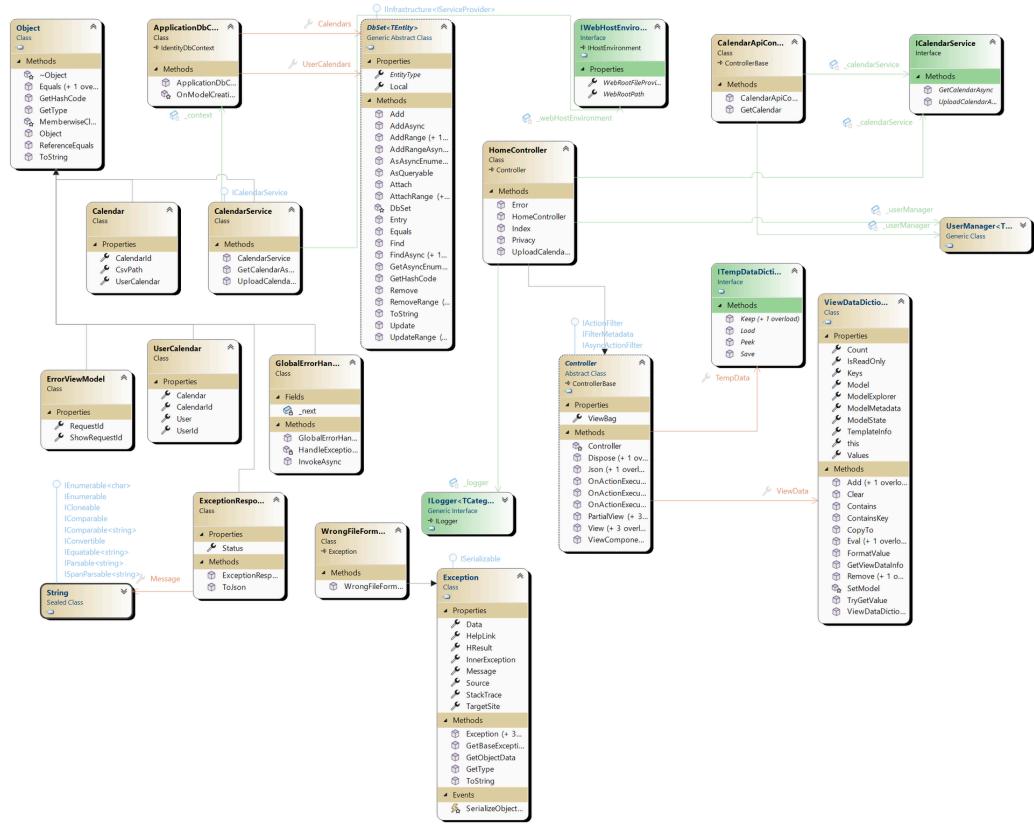
UserCalendars primary key (UserId, CalendarId)

Dijagram baze podataka

Ovdje se prikazuje dijagram baze podataka koja je realizirana za prvu kontrolnu točku (vlastita slika).



Dijagram razreda



Reference koje su se koristile u datumu pisanja ove stranice dokumentacije (13. 11. 2024):

- [T Getting Started — OAuth](#)
- [E Using OAuth and Cookies in Browser Based Apps | Best Practices | Curity](#)
- [T Introduction to Identity on ASP.NET Core](#)
- [ASP.NET MVC Pattern | .NET](#)
- [T What is monolithic architecture in software? | Definition from Tech...](#)
- [T SQLite: System Tables](#)
- [ASP.NET Core Identity Tables](#)
- [E What is EFMigrationsHistory table in EF core?](#)

Materijali s [Fakultet elektrotehnike i računarstva](#), kolegij „Programsko inženjerstvo”.

6. Implementacija i korisničko sučelje

Korištene tehnologije i alati

dio 2. revizije

Detaljno navesti sve tehnologije i alate koji su primjenjeni pri izradi dokumentacije i aplikacije. Ukratko ih opisati, te navesti njihovo značenje i mjesto primjene. Za svaki navedeni alat i tehnologiju je potrebno navesti internet poveznicu gdje se mogu preuzeti ili više saznati o njima.

Ispitivanje programskog rješenja

dio 2. revizije

U ovom poglavlju je potrebno opisati provedbu ispitivanja implementiranih funkcionalnosti na razini komponenti i na razini cijelog sustava s prikazom odabralih ispitnih slučajeva. Studenti trebaju ispitati temeljnu funkcionalnost i rubne uvjete.

Ispitivanje komponenti

Potrebno je provesti ispitivanje jedinica (engl. unit testing) nad razredima koji implementiraju temeljne funkcionalnosti. Razraditi minimalno 6 ispitnih slučajeva u kojima će se ispitati redovni slučajevi, rubni uvjeti te izazivanje pogreške (engl. exception throwing). Poželjno je stvoriti i ispitni slučaj koji koristi funkcionalnosti koje nisu implementirane. Potrebno je priložiti izvorni kod svih ispitnih slučajeva te prikaz rezultata izvođenja ispita u razvojnom okruženju (prolaz/pad ispita).

Ispitivanje sustava

Potrebno je provesti i opisati ispitivanje sustava koristeći radni okvir Selenium1. Razraditi minimalno 4 ispitna slučaja u kojima će se ispitati redovni slučajevi, rubni uvjeti te poziv funkcionalnosti koja nije implementirana/izaziva pogrešku kako bi se vidjelo na koji način sustav reagira kada nešto nije u potpunosti ostvareno. Ispitni slučaj se treba sastojati od ulaza (npr. korisničko ime i lozinka), očekivanog izlaza ili rezultata, koraka ispitivanja i dobivenog izlaza ili rezultata. Izradu ispitnih slučajeva pomoću radnog okvira Selenium moguće je provesti pomoću jednog od sljedeća dva alata:

- dodatak za preglednik Selenium IDE - snimanje korisnikovih akcija radi automatskog ponavljanja ispita
- Selenium WebDriver - podrška za pisanje ispita u jezicima Java, C#, PHP koristeći posebno programsko sučelje. Detalji o korištenju alata Selenium bit će prikazani na posebnom predavanju tijekom semestra.

Dijagram razmještaja

dio 2. revizije Potrebno je umetnuti specifikacijski dijagram razmještaja i opisati ga. Moguće je umjesto specifikacijskog dijagrama razmještaja umetnuti dijagram razmještaja instanci, pod uvjetom da taj dijagram bolje opisuje neki važniji dio sustava.

Upute za puštanje u pogon

dio 2. revizije

U ovom poglavlju potrebno je dati upute za puštanje u pogon (engl. deployment) ostvarene aplikacije. Na primjer, za web aplikacije, opisati postupak kojim se od izvornog koda dolazi do potpuno postavljene baze podataka i poslužitelja koji odgovara na upite korisnika. Za mobilnu aplikaciju, postupak kojim se aplikacija izgradi, te postavi na neku od trgovina. Za stolnu (engl. desktop) aplikaciju, postupak kojim se aplikacija instalira na računalo. Ukoliko mobilne i stolne aplikacije komuniciraju s poslužiteljem i/ili bazom podataka, opisati i postupak njihovog postavljanja. Pri izradi uputa preporučuje se naglasiti korake instalacije uporabom natuknica te koristiti što je više moguće slike ekrana (engl. screenshots) kako bi upute bile jasne i jednostavne za slijediti. Dovršenu aplikaciju potrebno je pokrenuti na javno dostupnom

poslužitelju. Studentima se preporuča korištenje neke od sljedećih besplatnih usluga: Amazon AWS, Microsoft Azure ili Heroku. Mobilne aplikacije trebaju biti objavljene na F-Droid, Google Play ili Amazon App trgovini.

7. Zaključak i budući rad

dio 2. revizije

U ovom poglavljtu potrebno je napisati osvrt na vrijeme izrade projektnog zadatka, koji su tehnički izazovi prepoznati, jesu li riješeni ili kako bi mogli biti riješeni, koja su znanja stečena pri izradi projekta, koja bi znanja bila posebno potrebna za brže i kvalitetnije ostvarenje projekta i koje bi bile perspektive za nastavak rada u projektnoj grupi. Potrebno je točno popisati funkcionalnosti koje nisu implementirane u ostvarenoj aplikaciji.

8. Popis literature

Kontinuirano osvježavanje Popisati sve reference i literaturu koja je pomogla pri ostvarivanju projekta.

1. Programsko inženjerstvo, FER ZEMRIS, <http://www.fer.hr/predmet/proinz>
2. I. Sommerville, "Software engineering", 8th ed, Addison Wesley, 2007.
3. T.C.Lethbridge, R.Langaniere, "Object-Oriented Software Engineering", 2nd ed. McGraw-Hill, 2005.
4. I. Marsic, Software engineering book", Department of Electrical and Computer Engineering, Rutgers University, <http://www.ece.rutgers.edu/~marsic/books/SE>
5. The Unified Modeling Language, [Unified Modeling Language \(UML\) description, UML diagram examples, tutorials and reference for all types of UML diagrams - use case diagrams, class, package, component, composite structure diagrams, deployments, activities, interactions, profiles, etc.](#)
6. Astah Community, [Powerful and Fast UML Diagramming Software - Astah](#)
7. [Getting Started — OAuth](#)
8. [Using OAuth and Cookies in Browser Based Apps | Best Practices | Curity](#)
9. [Introduction to Identity on ASP.NET Core](#)
10. [ASP.NET MVC Pattern | .NET](#)
11. [What is monolithic architecture in software? | Definition from Tech...](#)
12. [SQLite: System Tables](#)
13. [ASP.NET Core Identity Tables](#)
14. [What is EF.MigrationsHistory table in EF core?](#)
15. Materijali s [Fakultet elektrotehnike i računarstva](#), kolegij „Programsko inženjerstvo”.

9. Prikaz aktivnosti grupe

Dnevnik Sastajanja

Svi sastanci dodani su kao svoji page-evi radi organizacije u ovaj page na Confluencu. (Ako se ovo gleda na wikipiju sastanci su na kraju izlistani kao dodani page-evi)

Plan rada

Za ostvarivanje plana rada korišten je Jira ticketing sustav. Sve za prvu predaju bit će realizirano unutar jednog sprinta, dok će sve za drugu predaju biti unutar drugoga. U praksi trajanje sprinta je bliže 3 tjedna nego 6 no u svrhu organizacije i bez nepotrebnih komplikacija smatrali smo da je najbolje podijeliti u 2 sprinta sve. Grafovi i prikazi ovoga bit će dodani prije predaje.

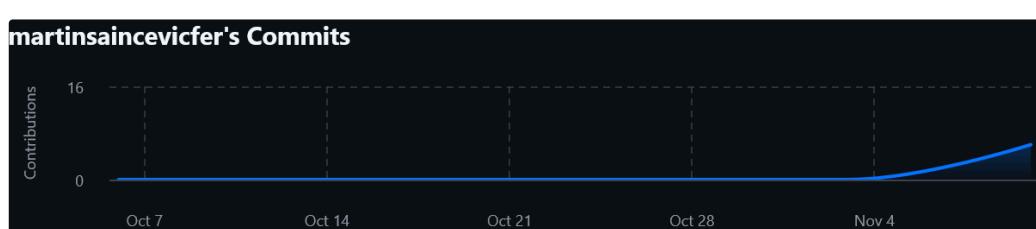
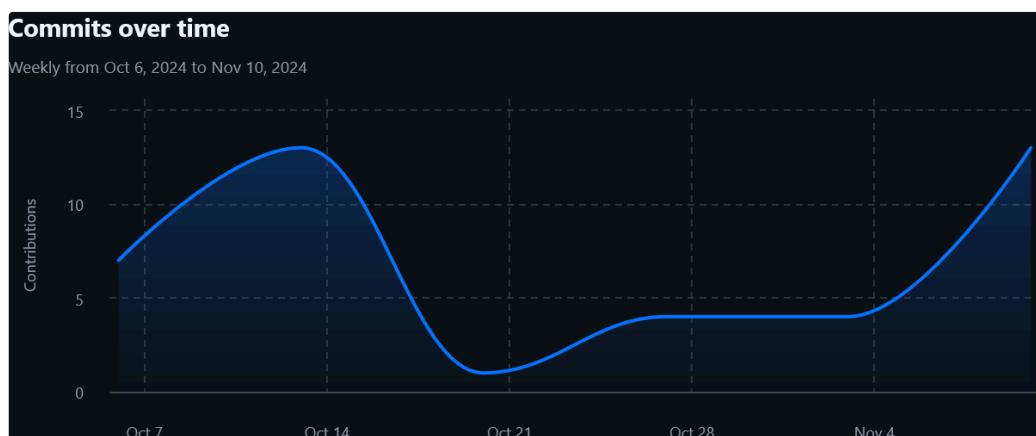
Tablica aktivnosti

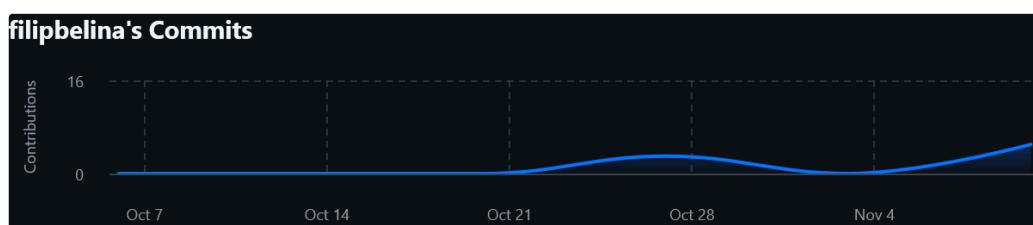
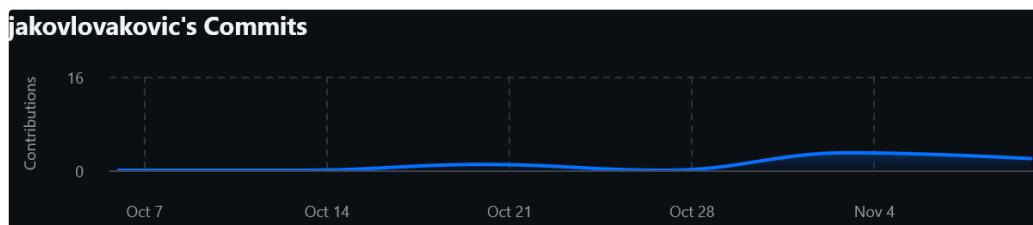
Aktivnosti su vidljive u tablici dolje prvi stupac predstavlja imena aktivnosti, dok svaki drugi broj sati koja je određena osoba napravila na toj aktivnosti.

	Belina Filip	Marušić Leo	Marinović Mislav	Lovaković Jakov	Lalić Jan	Badel Jan	Šainčević Martin
Upravljanje projektom	10	5					
Opis projektnog zadatka	2						
Funkcionalni zahtjevi	2		3				
Opis pojedinih obrazaca			1				
Dijagram obrazaca			4				
Sekvencijski dijagrami	4	4	10	4			
Opis ostalih zahtjeva			1				
Arhitektura i dizajn sustava	8						
Baza podataka		4		24			
Dijagram razreda							
Dijagram stanja							
Dijagram aktivnosti							
Dijagram komponenti					4		
Korištene tehnologije i alati	2						
Ispitivanje programskog rješenja	4						
Dijagram razmještaja							
Upute za puštanje u pogon							

Dnevnik sastajanja							
Zaključak i budući rad	0.5						
Popis literature							
Implementacija Oauth2		20		12			
Pisanje dev guidelineova	2						
Učenje tehnologija	8	10	8	16	16	16	16
Sastanci	15	15	10	15	9	5	5
Pregledavanje pull requestova	4	1		1			
Bugfixing	2	2		3			
Kalendar				8	4		12
Depoloyment		10					
Home page						16	9
Modeliranje izgleda aplikacije			4				
Exception handeler						8	
Ukupno	63.5	71	41	83	41	37	42

Dijagram pregleda promjena





Poslednji mesec rada na prvoj predaji

Redom prikazani: Lovaković, Belina, Šainčević, Marušić, Lalić, Badel, Marinović

Kjučni izazovi i rješenja

- Najveći izazovi bili su .NET + React povezati sa Oauth2, te deployanje .NET aplikacije
 - Opis izazova: Naše tehnologije činile su povezivanje Oauth2 iznimno izazovnime, te .NET nije bio lagan za deployati na Linux serveru
 - Rješenja: Projekt je dobio veliki redesign koji je predvodio Leo Marušić koji nam je olakšao implementaciju.

Meetings

Lista svih sastanaka do sada:

- Meeting 29.10.2024.
- Meeting 28.10.2024.
- Meeting 25.10.2024.
- Meeting 23.10.2024.
- Meeting 22.10.2024.
- Meeting 18.10.2024.
- Meeting 13.10.2024.
- Meeting 10.10.2024.
- BE - Meeting 30.11.2024.
- BE - Meeting 9.12.2024.

Meeting 29.10.2024.

Datum: 29.10.2024.

Vrijeme: 18:00 - 19:00

Sudionici: Mislav Marinković, Filip Belina, Jakov Lovaković, Leo Marušić, Jan Badel, Martin Šainčević, Jan Lalić

Dnevni red:

1. Generalna rasprava

Meeting 28.10.2024.

Datum: 28.10.2024.

Vrijeme: 17:00 - 22:00

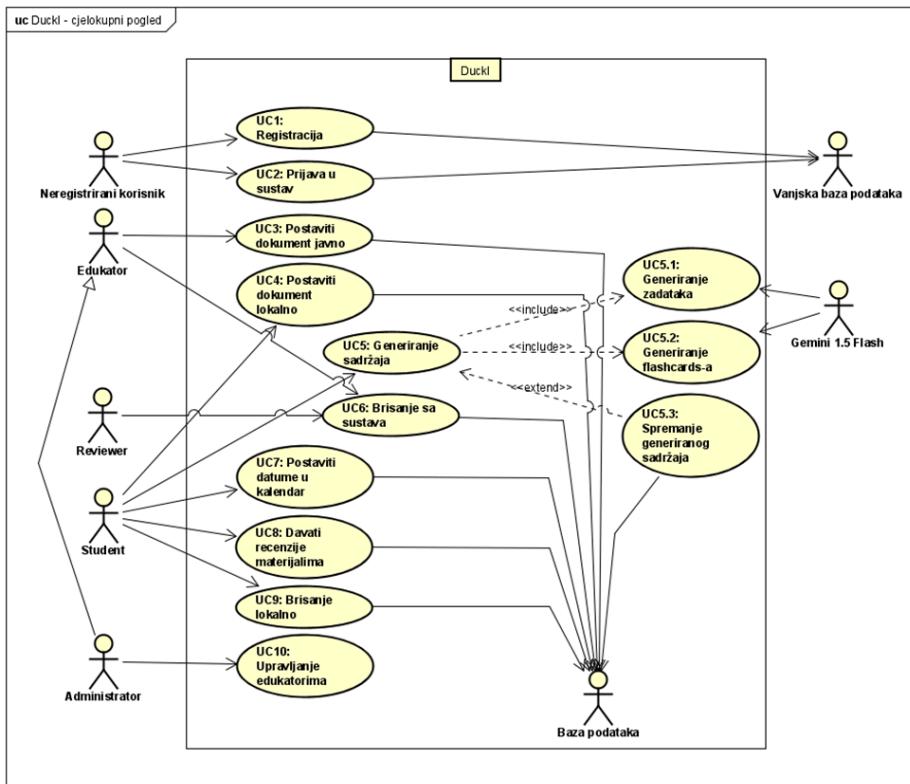
Sudionici: Mislav Marinković, Filip Belina, Jakov Lovaković, Leo Marušić

Dnevni red:

1. Izrada Use-Case dijagrama
2. Izrada sekvencijskih dijagrama

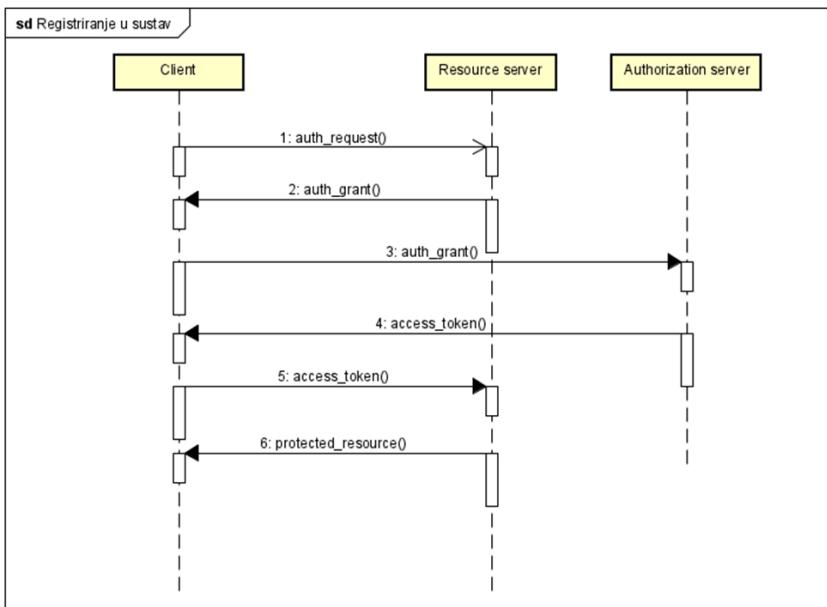
Dijagram obrazaca uporabe

(potencijalno dodati obrasce uporabe u obliku tablica, poput one na primjeru na Moodlu)

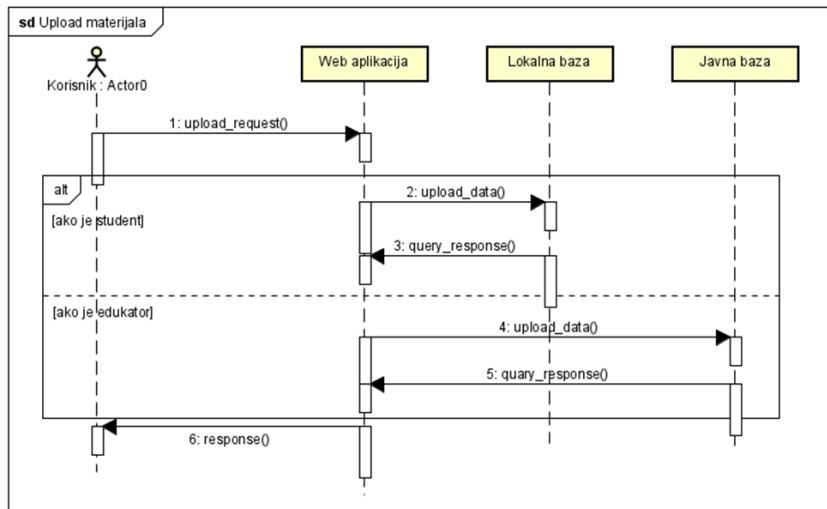


Sekvencijski dijagrami

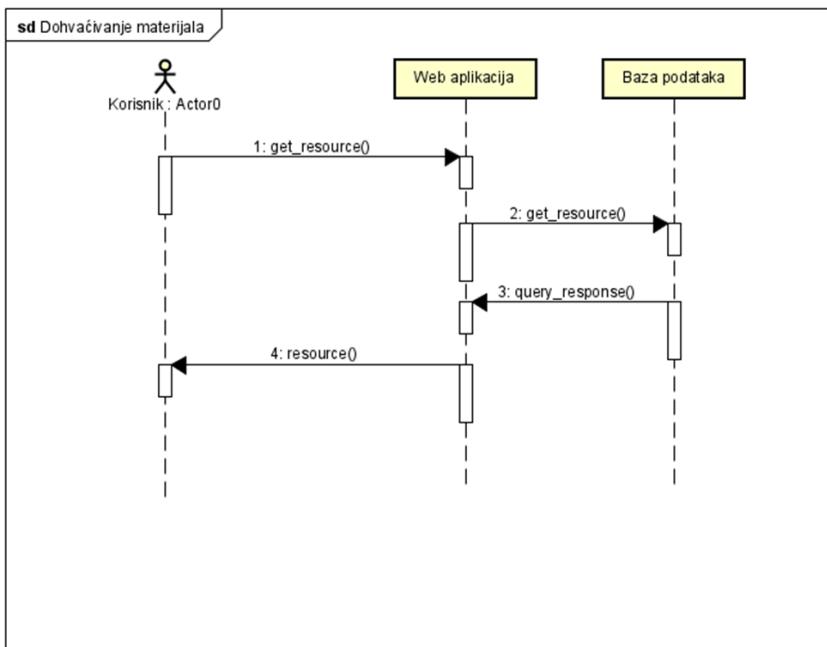
Registiranje u sustav



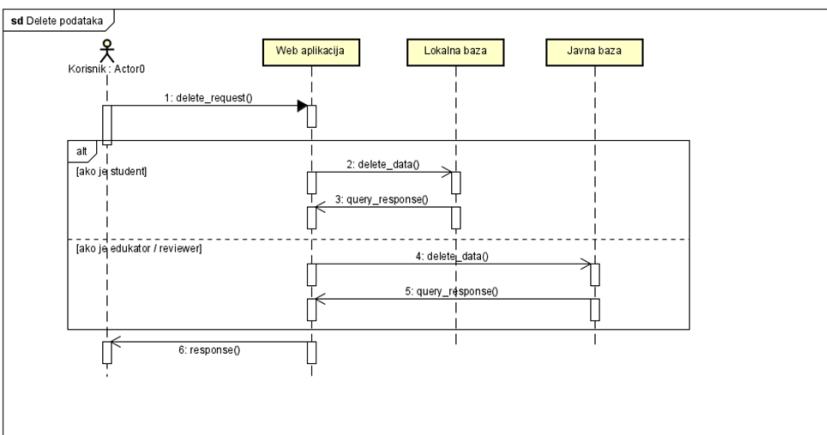
Upload materijala



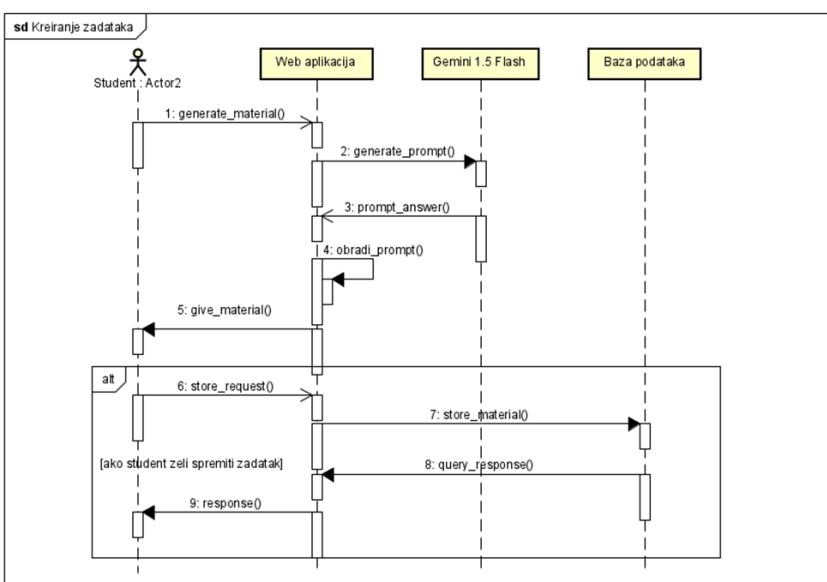
Dohvaćivanje materijala



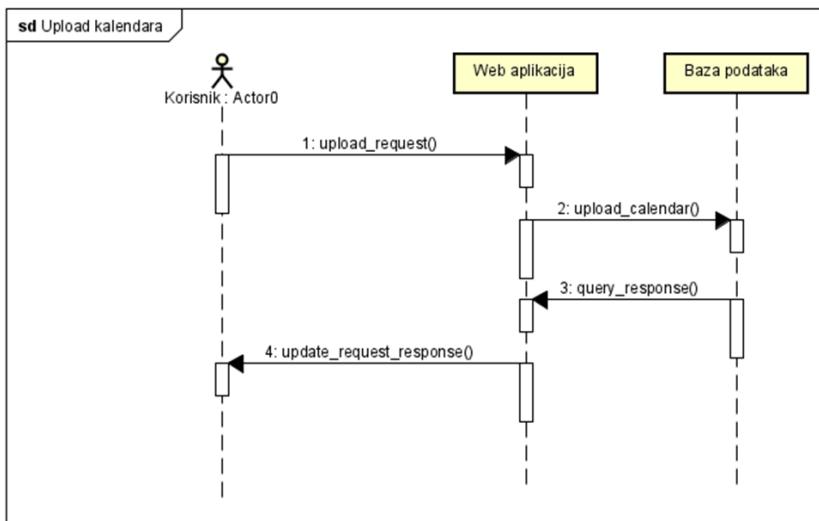
Brisanje materijala



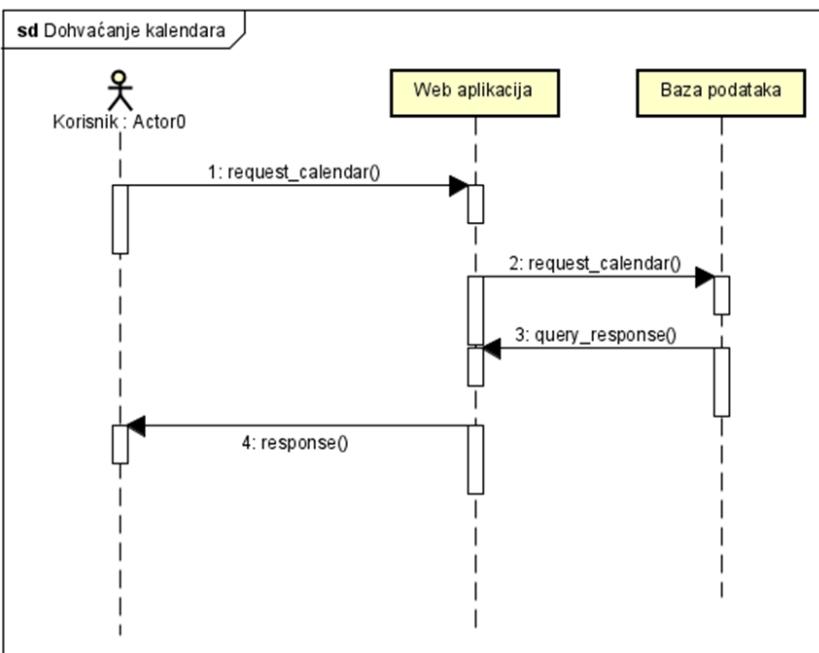
Kreiranje zadataka



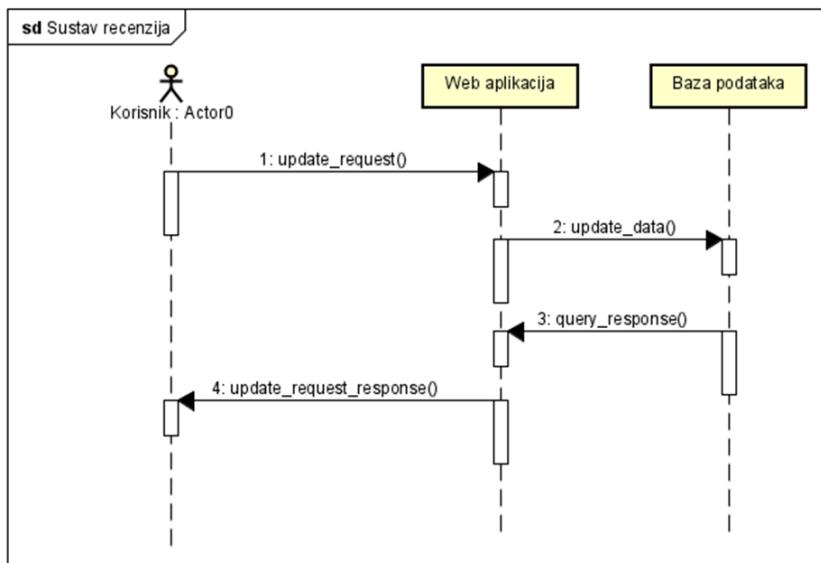
Upload kalendarja



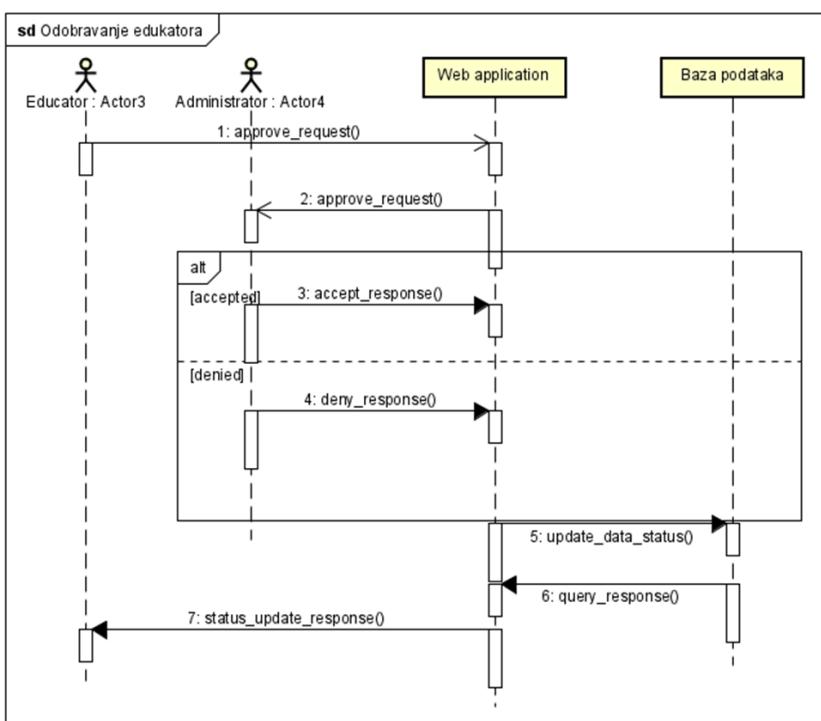
Dohvaćanje kalendarja



Sustav recenzija



Odobravanje edukatora



Meeting 25.10.2024.

Datum: 25.10.2024.

Vrijeme: 19:30 - 20:30

Sudionici: Filip Belina, Jakov Lovaković, Jan Badel, Leo Marušić, Martin Šainčević, Jan Lalić

Dnevni red:

1. Struktura web aplikacije
2. Implementacija baze podataka
3. Standardizacija procesa razvoja

1. Struktura web aplikacije

Definirana je struktura web aplikacije, uključujući organizaciju datoteka i koda. Backend tim je izradio prvu fazu implementacije baze podataka za pohranu korisničkih podataka.

2. Implementacija baze podataka

Voditelj backend tima je prezentirao strukturu datoteka i koda te objasnio kako raditi s njima.

3. Standardizacija procesa razvoja

Kako bi se osigurao efikasan i ujednačen proces razvoja, definirani su standardi za:

- **Kreiranje zadataka (ticketa):** Utvrđen je način kreiranja zadataka u JIRA-i, uključujući potrebne informacije i format opisa.
- **Kreiranje grana (brancheva):** Definisana je konvencija imenovanja grana u Git rezitoriju.
- **Pull zahtjevi (pull requests):** Utvrđene su smjernice za kreiranje pull zahtjeva, uključujući opis promjena i proces revizije koda.
- **Confluence dokumentacija:** Definiran je način dokumentiranja u Confluence-u, uključujući strukturu stranica i potrebne informacije.

Sva će dokumentacija biti prvo kreirana u Confluence-u, a zatim će biti migrirana u GitHub wiki. Ova će dokumentacija poslužiti kao osnova za izradu službene dokumentacije projekta.

Zaključak:

Sastanak je bio uspješan, a tim je napravio značajan korak u definiranju strukture web aplikacije i standardizaciji procesa razvoja.

Meeting 23.10.2024.

Datum: 23.10.2024.

Vrijeme: 19:00 - 20:00

Sudionici: Jakov Lovaković, Leo Marušić, Filip Belina, Martin Šainčević

Dnevni red:

1. Ažuriranje zahtjeva projekta
2. Redizajn sheme baze podataka
3. Demo aplikacija i testiranje

1. Ažuriranje zahtjeva projekta

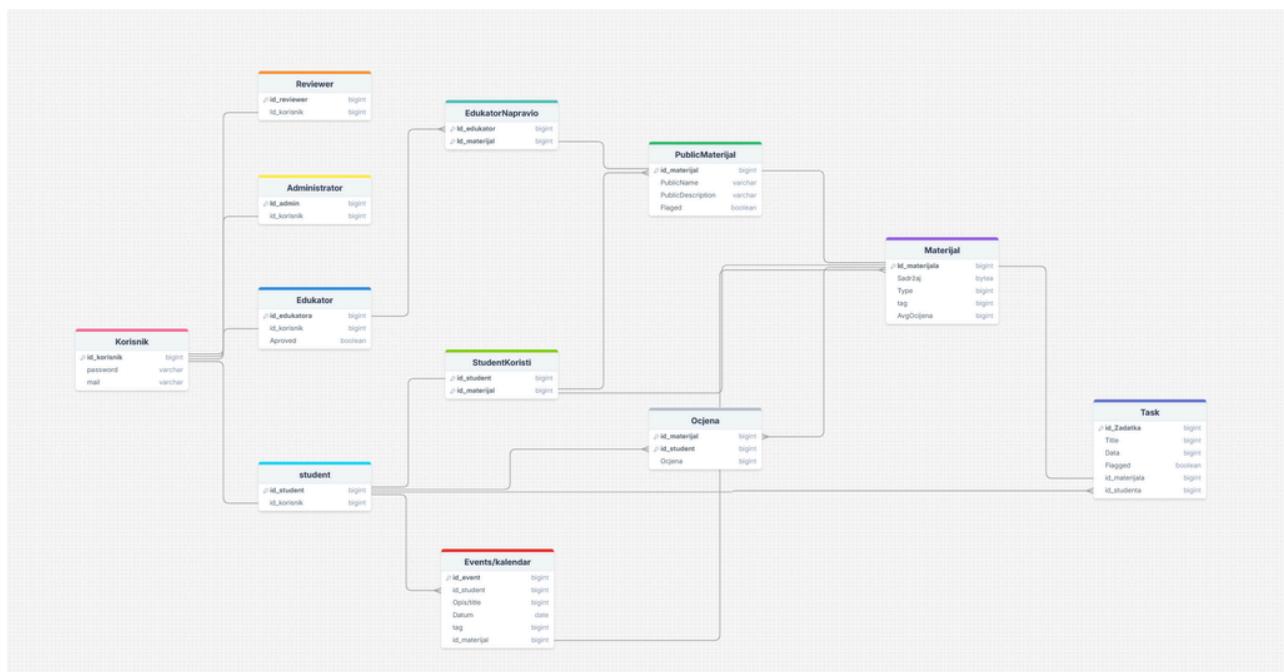
Nakon primanja dodatnih zahtjeva za projekt, neke od optionalnih funkcionalnosti sada su postale obavezne. To je utjecalo na dizajn baze podataka i zahtjevalo je redizajn.

2. Redizajn sheme baze podataka

Shema baze podataka je redizajnjirana kako bi se prilagodila novim funkcionalnostima. Ipak, neki aspekti sheme još nisu finalizirani, uključujući:

- **Tipove podataka:** Potrebno je donijeti konačnu odluku o tipovima podataka za određene atribute.
- **Polja tablica:** Neka polja u tablicama zahtijevaju dodatnu diskusiju i definiranje.
- **Relacije:** Određene relacije između entiteta trebaju biti detaljnije razrađene.

Ove će se točke razjasniti na sljedećem sastanku.



3. Demo aplikacija i testiranje

Demo aplikacija je modificirana kako bi koristila PostgreSQL umjesto Microsoft SQL baze podataka. Testiranje aplikacije je započelo na više računala prije samog deploymenta.

Zaključak:

Unatoč promjenama u zahtjevima, tim je uspješno redizajnirao shemu baze podataka i napravio napredak u razvoju demo aplikacije.

Meeting 22.10.2024.

Zapisnik sa sastanka

Datum: 22.10.2024.

Vrijeme: 17:00 - 21:00

Sudionici: Filip Belina, Jakov Lovaković, Leo Marušić, Jan Lalić

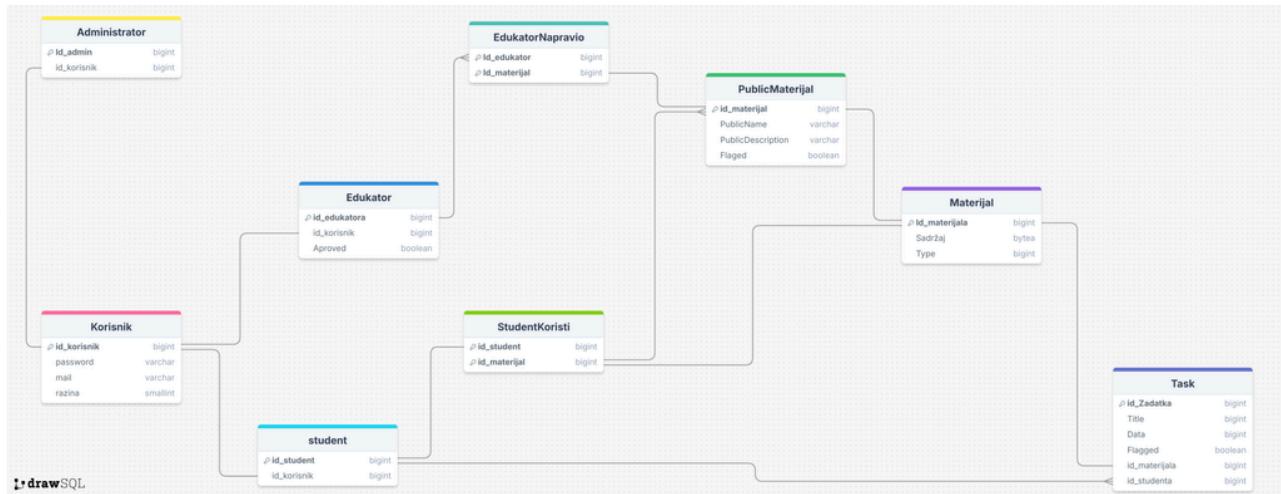
Dnevni red:

1. Dizajn sheme baze podataka
2. Proces učitavanja podataka
3. Demo aplikacija

1. Dizajn sheme baze podataka

Tim je detaljno razradio shemu baze podataka, definirajući:

- **Entitete:** Identificirani su ključni entiteti potrebni za pohranu podataka aplikacije (npr. korisnik, dokument, pitanje, flash kartica, kalendar).
- **Relacije:** Definisane su relacije između entiteta, uključujući tipove relacija (jedan-na-jedan, jedan-na-više, više-na-više).
- **Atribute:** Za svaki entitet su specificirani atributi i njihovi tipovi podataka. [TBD]



2. Proces učitavanja podataka

Razmatran je proces učitavanja podataka u bazu podataka od strane korisnika. Definisani su:

- **Formati podataka:** Utvrđeni su formati datoteka koje će korisnici moći učitavati (npr. PDF za dokumente).
- **Metode učitavanja:** Razmotrene su različite metode učitavanja podataka (npr. direktno učitavanje, učitavanje putem API-ja).
- **Validacija podataka:** Definisane su procedure za validaciju učitanih podataka kako bi se osigurala konzistentnost i integritet baze podataka.

3. Demo aplikacija

Tim je započeo s izradom demo aplikacije s ciljem:

- **Postavljanja baze podataka:** Kreiranje baze podataka na temelju definirane sheme.
- **Konfiguriranja okruženja za razvoj:** Postavljanje razvojnog okruženja i potrebnih alata.
- **Testiranja procesa učitavanja podataka:** Verifikacija funkcionalnosti učitavanja podataka u bazu.

Zaključak:

Sastanak je bio produktivan, a tim je ostvario značajan napredak u dizajnu baze podataka i definiranju procesa učitavanja podataka. Izrada demo aplikacije će omogućiti daljnje testiranje i validaciju arhitekture baze podataka.

Meeting 18.10.2024.

Datum: 18.10.2024.

Vrijeme: 20:00 - 21:00

Sudionici: Svi

Dnevni red:

1. Uvod u JIRA i Confluence

1. Uvod u JIRA i Confluence

Održana je prezentacija o korištenju JIRA i Confluence alata za upravljanje projektima i timsku suradnju.

Sudionici su upoznati s osnovnim funkcionalnostima JIRA-e, uključujući:

- **Kreiranje zadataka (issues):** Objasnjen je proces kreiranja novih zadataka, definiranje prioriteta, dodjeljivanje tipova zadataka i postavljanje rokova.
- **Dodjeljivanje i upravljanje zadacima:** Prikazano je kako dodijeliti zadatke članovima tima, pratiti njihov napredak i ažurirati status.
- **Povezivanje zadataka:** Demonstrirano je kako povezati međusobno povezane zadatke radi bolje organizacije i praćenja.

Također, prezentirane su i ključne funkcionalnosti Confluence-a:

- **Kreiranje stranica:** Objasnjeno je kako kreirati i uređivati stranice, dodavati sadržaj (tekst, slike, tablice) te ih organizirati u hijerarhijske strukture.
- **Povezivanje JIRA zadataka i Confluence stranica:** Prikazano je kako povezati JIRA zadatke s relevantnim Confluence stranicama radi bolje dokumentacije i centraliziranog pristupa informacijama.
- **Timská suradnja:** Istaknute su mogućnosti Confluence-a za timsku suradnju, kao što su dijeljenje dokumenata, komentiranje i praćenje promjena.

Zaključak:

Sudionici su stekli osnovno znanje o korištenju JIRA i Confluence alata. Ova će znanja omogućiti efikasnije upravljanje projektom, bolju organizaciju zadataka i transparentniju komunikaciju unutar tima.

Meeting 13.10.2024.

Datum: 13.10.2024.

Vrijeme: 19:00 - 20:00

Sudionici: svi

Dnevni red:

1. Definiranje teme projekta
2. Osnovna ideja aplikacije
3. Odabir tehnologija

1. Definiranje teme projekta

Na sastanku je detaljnije razrađena tema projekta i finaliziran njen opis.

2. Osnovna ideja aplikacije

Definirana je osnovna ideja aplikacije koja će koristiti AI, putem Gemini API-ja, za obradu PDF dokumenata i kreiranje personaliziranih materijala za učenje (flash kartice, pitanja i sl.). Aplikacija će uključivati kalendar za praćenje ispita i planiranje učenja, te sustav dnevnih podsjetnika ili pitanja za kontinuirano učenje.

3. Odabir tehnologija

Započeta je diskusija o odabiru tehnologija za razvoj aplikacije. Sljedeće tehnologije su predložene:

- **Frontend:** ReactJS
- **Backend:**  [ASP.NET Core | Open-source web framework for .NET](#)
- **Baza podataka:** PostgreSQL
- **Upravljanje projektom:** JIRA + Confluence

Konačna odluka o korištenju ovih tehnologija bit će donesena nakon dodatnog istraživanja i razmatranja alternativnih opcija.

Zaključak:

Na sastanku je uspješno definirana tema projekta i osnovna ideja aplikacije. Započet je proces odabira tehnologija, a konačne odluke će biti donesene na sljedećem sastanku.

Meeting 10.10.2024.

Datum: 10.10.2024.

Vrijeme: 18:00 - 19:00

Sudionici: Svi

Dnevni red:

1. Upoznavanje i podjela uloga
2. Odabir teme projekta
3. Postavljanje Github repozitorija
4. Odabir imena tima

1. Upoznavanje i podjela uloga

Članovi tima su se međusobno upoznali i predstavili. Definisane su sljedeće uloge unutar tima:

- Filip Belina - Project Lead
- Jakov Lovaković - Backend Team Lead
- Jan Lalić - Backend Engineer
- Leo Marušić - DevOps/Database Engineer
- Mislav Marinović - Lead design
- Jan Badel - Frontend Engineer
- Martin Šainčević - Frontend engineer

2. Odabir teme projekta

Tim je donio odluku o razvoju projekta na vlastitu temu. Ukratko su predstavljene ideje za potencijalne teme. Konačna odluka o temi će biti donesena na sljedećem sastanku.

3. Postavljanje Github repozitorija

Dogovoreno je postavljanje Github repozitorija za upravljanje kodom projekta. [Ime] će biti zadužen/a za kreiranje repozitorija i dodjeljivanje pristupa ostalim članovima tima.

4. Odabir imena tima

Predložena su sljedeća imena za tim: [Navedite predložena imena]. Konačan odabir imena će se provesti putem glasanja do [datum].

Zaključak:

Sastanak je uspješno održan, te su definirane osnovne smjernice za daljnji rad na projektu.

BE - Meeting 30.11.2024.

Jakov Lovaković i Jan Lalić

45 min

- diskutiranje o dodavanju novih tablica, interakciji admina s bazom, provjera postojanosti prijave na role, općenito diskutiranje implementacije roleova i podjela posla

BE - Meeting 9.12.2024.

Sudionici: Jankov Lovaković, Filip Belina i Jan Lalić

Trajanje: 1 h

Diskutiranje implementacije i planu rada na projektu vezano uz dodavanje rolea SuperStudent, uploada, reviewa i searchanje/selekcija pdfova

A. Dnevnik promjena dokumentacije

Rev.	Opis promjene/dodataka	Autori	Datum
0.1	Napravljen predložak dokumentacije i inicijalizacija repozitorija	Leo Marušić	10.10.2023
0.2	Postavljanje timskih uloga i općih ciljeva; inicijalna struktura aplikacije	Filip Belina	10.10.2024
0.3	Definiranje projekta DuckL - AI asistent za učenje; razmatranje tehnologija: ReactJS za frontend, http://ASP.NET za backend, PostgreSQL za bazu podataka	Cjeli tim	10.10.2024

0.4	Demo aplikacija: kreiran inicijalni UI i povezivanje s PostgreSQL bazom podataka	Jakov Lovaković	25.10.2024
0.5	Standardizacija procesa razvoja: uspostava pravila za ticketing u JIRA-i, kreiranje grana, pull request standardi	Filip Belina, Leo Marušić	25.10.2024
0.6	Izrada Use-Case i sekvencijskih dijagrama za funkcionalnosti poput upload materijala, registracija, dodavanje kalendar, kopirani confluence pagevi na wiki	Filip Belina, Mislav Marinović, Leo Marušić, Jakov Lovaković	28.10.2024

0.7	Project redesign, implementiran Oauth2	Leo Marušić	5.11.2024
0.8	Napravljena prva verzija kalendar-a	Jakov Lovaković	8.11.2024
0.9	Napravljen globalni error handler	Jan Lalić	10.11.2024
0.10	Napravljen izgled kalendar-a za predaju	Martin Šainčević	12.11.2024
0.11	Mali bug fix za register vezan uz registraciju novog korisnika ako je korisnik trenutno ulogiran	Jakov Lovaković	13.11.2024
0.12	Deployana aplikacija	Leo Marušić	14.11.2024
0.13	Napravljena stranica za kalendar i poboljšan njegov izgled	Martin Šainčević	10.12.2024

	<ul style="list-style-type: none"> • implementirati dizajn za sve stranice pod rutom /Identity/Account/ • implementirati dizajn za sve stranice pod rutom /Identity/Account/Manage/ • implementirati dizajn za header i footer • Home i Privacy buttons su maknuti, Home je accessible preko logtipa, a privacy preko footer-a • resend email confirmation se sada nalazi na 	Jan Badel	12.12.2024
--	---	-----------	------------

	<p>register</p> <p>screenu</p> <ul style="list-style-type: none"> • CSS se nalazi u zasebnim fileovima • custom.css je nadodani file u kojem se overrideaju neke Bootstrap klase (vezane uz boje) • buttoni veiw material (za sada beskoristan)view calendar/ch at se sada nalaze u _LoginParti al pageu 		
	<ul style="list-style-type: none"> • Added Google Gemini text prompting • Fixed calendar working on 	Leo Marušić	30.11.2024

Windows/Li
nux
systems

- Calendar directory is now being created if it does not exist
- Removed the redundant email confirmation from the user registration process
- User will now automatically login after registration
- User can now set an username (had to make a custom sign-in manager for that... wonderful)

	<ul style="list-style-type: none"> Also updated .gitignore to include user data files 		
	<ul style="list-style-type: none"> Dodana nova tablica UserRoleStatuses Na ruti /Home/Role s moguće je pristupiti testnom sučelju za dodavanje roleova (korišteno pri provjeri ispravnosti implementacija). Testno sučelje može koristiti frontend tim da vide kako se implementira backend dio ove funkcionalnosti u 	Jan Lalić, Jakov Lovaković	2. 12. 2024.

	<p>frontend, ili ga može redizajnirati i ostaviti kakav je.</p> <ul style="list-style-type: none"> • Popravljene manje stvari (dodata autorizacija na određene rute, promijenjen o ime CalendarController klase kako bi kod bio konzistentniiji). 		
	Approveanje roleova kao admin i status rolea za korisnika	Jan Lalić, Jako Lovaković	8. 12. 2024.
	<p>Izvadio sam javascript i css dijelove koda iz Index.cshtml te stavio to u dvije odvojene datoteke.</p> <p>Stvorio sam</p>	Martin Šainčević	10. 12. 2024.

novi view
Calendar.csht
ml te uz pomoć
Jakova
napravio da se
može otići na
tu stranicu kroz
nav bar.
Također sam
napravio
odvojeni
index.js koji je
skoro identičan
calendar.js a
koji će u
budućnosti
mijenjati za
ostvarivanje
malo drukčijeg
kalendara na
home stranici.
Home i
Calendar
stranice koriste
isti prijašnje
navedeni css
dokument.
Home stranicu
će dalje
mijenjati kad se
ostvare
funkcionalnosti
koje ćemo
prikazivati na

	<p>dashboardu na toj stranici.</p> <p>Nakon uploada novog csv dokumenta za kalendar korisnik je automatski preusmjeren na Home stranicu što smatram da nije problem no htio sam samo napomenuti.</p>		
	<p>Dodan SuperStudent role i način za davanje tog rolea korisniku</p>	<p>Jan Lalić, Jakov Lovaković</p>	12. 12. 2024.
	<p>Dodano uploadanje pdf-ova za edukatora i studenta.</p> <p>Ruta za upload je pdf/uploadpdf (i za edukatora i za studenta).</p> <p>Dodana lokalizacija.</p> <p>Dodano</p>	<p>Jan Lalić, Jakov Lovaković</p>	12. 12. 2024.

	<p>postavljanje tagova i njihovo pregledavanje. Ruta za tagove admin/tags</p>		
	<p>Dodano: - browseanje flagganih i private pdfova (na index page, i za superstudenta i educatora, samo sto educator ima samo svoje, on nema opcije unflagganja s te stranice pošto ni nemože flaggat)</p> <ul style="list-style-type: none"> • browseanje i flagganje public pdfova (view public material gumb, pdf/viewpub licmaterial, 	<p>Jan Ialić, Jakov Lovaković</p>	<p>14. 12. 2024.</p>

	<p>educator nemoze flaggat) unflagganje public materiala s index pagea</p> <ul style="list-style-type: none"> • otvaranje pdfova 		
	<p>Dodana je mogućnost upoticanja i downvotanja public pdfova. Dodana mogućnost removanja pdfova; student može svoje privatne pdfove maknuti, edukator svoje i reviewer može edukatorove pdfove maknuti uz opis zašto je maknut. Edukator ima reportlog u kojemu može vidjeti koji su</p>	<p>Jan Lalić, Jakov Lovaković</p>	18. 12. 2024.

	<p>mu kanuti pdfovi.</p> <ul style="list-style-type: none"> • implementirati dizajn za Roles page • implementirati dizajn za BrowseRole Application page • veći font za izbornik stranica Account Management • vidljive Role u Profile pageu 	Jan Badel	19. 12. 2024.
--	--	-----------	---------------

Untitled whiteboard (2)