

Proglaz TG08.3 Home . . . . .	2
Službena Dokumentacija . . . . .	3
Home . . . . .	4
1. Opis projektnog zadatka . . . . .	5
2. Analiza zahtjeva . . . . .	8
4. Specifikacija zahtjeva sustava . . . . .	12
5. Arhitektura i dizajn sustava . . . . .	14
6. Implementacija i korisničko sučelje . . . . .	22
7. Zaključak i budući rad . . . . .	24
8. Popis literature . . . . .	25
9. Prikaz aktivnosti grupe . . . . .	26
Meetings . . . . .	27
Meeting 29.10.2024. . . . .	28
Meeting 28.10.2024. . . . .	29
Meeting 25.10.2024. . . . .	34
Meeting 23.10.2024. . . . .	35
Meeting 22.10.2024. . . . .	37
Meeting 18.10.2024. . . . .	39
Meeting 13.10.2024. . . . .	40
Meeting 10.10.2024. . . . .	41
A. Dnevnik promjena dokumentacije . . . . .	42

# ProgInz TG08.3 Home

- Filip Belina - Prject 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 - ReactJS
- Backend - ASP .NET
- Database - PostgreSQL
- Project Management - JIRA + Confluence



Home

Programsko inženjerstvo ak.god 2024./2025

Sveučilište u Zagrebu

Fakultet elektrotehnike i računarstva

DuckI

Tim: <TG08.3>

Gumene Patkice

Nastavnik: Vlado Sruk

# 1. Opis projektnog zadatka

## Funkcionalnost

Kao projektni zadatak izvoditi 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 dodane od strane edukatora.

Studentu aplikacija služi za spremište i upravljanje svojim materijalima. Kada će student htjeti interakciju sa 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 izveden pomoću Google Gemini 1.5 Flash modela. Model će primati dokumente (poput PDF-a), te na temelju njih stvoriti sadržaj, zadatke 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 kreacije i dodavanje 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 generaciji.

Nad edukatorima bi mogao nadzirati 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 validan.

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, Typescript-a te Tailwind CSS-a. U tu svrhu, koristiti će se principi responzivnog dizajna za dinamičku prilagodbu korisničkog sučelja različitim veličinama i razlučivostima zaslona.

Aplikacija također sadrži kalendar koji studentima služi za upravljanje s vremenom.

U kalendar mogu staviti unos rasporeda ispita.

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 sa 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 učiti zajedno 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 kvalitetu 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:

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

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

Edukatori će moći dijeliti materijale koji mogu pomoći učenicima da bolje razumiju određene koncepte. Ovo bi bio sjajan put da se stigne dalje od njihovih učionica. Oni mogu učinkovitije olakšati učenje prikladnim primjerima, vježbama i vodičima za učenje.

### 3. Povezani radovi i razlike:

Doista već postoje srodna rješenja u obrazovanju međutim, nema puno proizvoda koji kombiniraju personaliziranog asistenta uz pomoć umjetne inteligencije sa stvaranjem sadržaja u stvarnom vremenu sa 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: Najbliži proizvod, na bazi AI materijale pretvara u notes, flashcards, i tasks, no ne postoji nikakva interakcija ili sistem sa edukatorom, što bi činilo ovo teškime 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 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.

Responzivni 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 rađanje zadataka namijenjenih za pripremu za ispit.

## 7. Budućnost projekta

Ušli smo u ovaj projekt sa 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 FERu, te druga pošto svatko može dobiti API key za gemeni, 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 okruženju 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-001: Prenos PDF datoteka s vlastitim sadržajem
- Generirati zadatke i "flashcards"-e iz sadržaja u aplikaciji
  - F-002: Generacija zadataka i flashcards-a pomoću AI asistenta
- Uploadati kalendar u obliku CSV datoteke
  - F-003: Prenos kalendara u CSV formatu
- Dodavati datume u kalendar unutar aplikacije
  - F-004: Upravljanje osobnim kalendarom u sustavu
- Davati recenzije javno dostupnim materijalima
  - F-005: 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-007: Prenos materijala koji su javno dostupni studentima i recenzentima
- Brisati vlastiti sadržaj sa sustava
  - F-008: Uklanjanje vlastitih materijala iz sustava

#### 3. A-3 Recenzent (sudionik)

Reviewer pomaže u održavanju kvalitete sadržaja putem evaluacije javnih materijala.

Reviewer može:

- Reviewati javno dostupan sadržaj



- F-009: Ocjenjivanje i davanje povratnih informacija o javnim materijalima
- Brisati javno dostupan sadržaj sa sustava
  - F-010: Uklanjanje materijala koji ne zadovoljavaju standarde

#### 4. A-4 Administrator (inicijator)

Administrator upravlja edukatorima i korisnicima te osigurava pravilno funkcioniranje aplikacije.

Administrator može:

- Odobravati edukatore
  - F-011: Provjera i odobravanje novih edukatora

## 2.2 Funkcionalni zahtjevi

ID zahtjeva	Opis	Prioritet	Izvor	Kriteriji prihvatanja
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	Srednji	Specifikacija projekta	Student može dodati termine

	studentima pregled rasporeda ispita kroz kalendar.			ispita u kalendar i pregledati ih kasnije.
F-007	Sustav omogućuje studentima ocjenjivanje materijala.	Srednji	Specifikacija projekta	Student 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 sa 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.

## 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

### Obrasci uporabe

#### Dijagrami obrazaca uporabe

Prikazati odnos aktora i obrazaca uporabe odgovarajućim UML dijagramom. Nije nužno nacrtati sve na jednom dijagramu. Modelirati po razinama apstrakcije i skupovima srodnih funkcionalnosti.

##### 1. Visokorazinski dijagram obrazaca uporabe cijelog sustava

| Prikazuje najvažnije funkcionalnosti sustava iz perspektive krajnjih korisnika.

| Slika sustava - osnova za razumijevanje glavnih ciljeva projekta.

##### 2. dijagram obrazaca uporabe za ključne značajke

| detaljnije dijagrame koji se odnose na specifične značajke, poput procesa prijave korisnika, upravljanja podacima, ... obrade transakcija.

| U ranim fazama projekta, važno je jasno definirati koje su osnovne funkcionalnosti sustava, omogućava preciznije planiranje razvoj i podijele funkcionalnosti na manje zadatke.

##### 3. dijagram obrazaca uporabe za korisničke uloge

| Identificira glavne korisničke aktore i njihove interakcije sa sustavom.

| definira različitih vrsta korisnika i funkcionalnosti koje su im potrebne

| Osigurava bolje razumijevanje prioritete i specifične potrebe različitih grupa korisnika.

##### 4. dijagram obrazaca uporabe za osnovne poslovne procese

| prikazuje kako sustav podržava osnovne poslovne procese organizacije.

##### 5. dijagram obrazaca uporabe za kritične sustave i integracije

| interakcije sustava s vanjskim sustavima ili integracijama

### Opis obrazaca uporabe

Funkcionalne zahtjeve razraditi u obliku obrazaca uporabe. Preporuka: Svaki obrazac je potrebno razraditi prema donjem predlošku.

Ukoliko u nekom koraku može doći do odstupanja, potrebno je to odstupanje opisati i po mogućnosti ponuditi rješenje kojim bi se tijekom obrasca vratio na osnovni tijek.

#### UC broj obrasca - ime obrasca

- Glavni sudionik:
- Cilj:
- Sudionici:
- Preduvjet:
- Opis osnovnog tijeka:

1. opis korak jedan ( \_ referenca na funkcijski zahtjev\_ *F-001*)
2. opis korak dva
3. opis korak tri
4. opis korak četiri (*F-022.1*)
5. opis korak pet

- Opis mogućih odstupanja:

1. <opis rješenja mogućeg scenarija korak 1>
2. <opis rješenja mogućeg scenarija korak 2>
3. b <opis mogućeg scenarija odstupanja u koraku 2>
4. a <opis mogućeg scenarija odstupanja u koraku 3>

## Sekvencijski dijagrami

Nacrtati sekvencijske dijagrame koji modeliraju najvažnije dijelove sustava. Ukoliko postoji nedoumica oko odabira, razjasniti s asistentom. Uz svaki dijagram napisati detaljni opis dijagrama.

| izvorne kodove dijagrama pohranjujte u svom GIT-u

## Provjera uključenosti ključnih funkcionalnosti u obrasce uporabe

| za prikazane obrasce uporabe tablično navedite koje funkcionalne zahtjeve obuhvaćaju

## 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 podijeli 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 spremati u bazu organizirane po direktorijima, te ApplicationDbContext klasu koja upravlja bazom podataka.
- Preslikavanje na radnu platformu: Trenutan deployment odrađen lokalno na računalu preko port-forwardinga za svrhe prezentacije početne funkcionalnosti.
- Spremišta 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 dohvaćaju podaci/uređuju podaci. Nakon rada s podacima korisniku se preko kontrolera uredi stranica s izmjenama koje je zatražio.
- Sklopovskoprogramski zahtjevi: Opišite potrebne tehničke zahtjeve vezane uz sklopovlje i program, uključujući podršku za specifične operativne sustave, procesore, memoriju i druge komponente.

### Obrazloženje odabira arhitekture

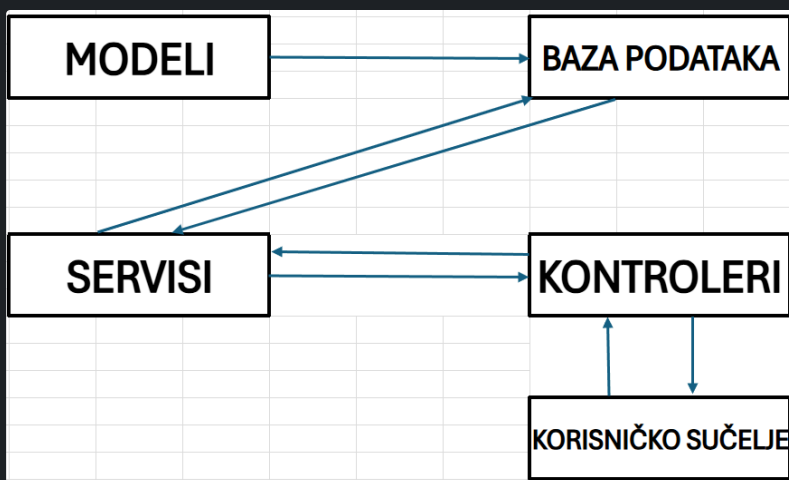
Kako je navedeno u prethodnim točkama, Duckl programski sustav koristi monolitnu arhitekturu sustava, te koristi principe „povećaj koheziju” (nadogradnja na podijeli pa vladaj) pošto su međusobno povezani elementi grupirani (npr. kontroleri), „smanji međuovisnost” pošto su komponente uglavnom „single-task”, 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 server - klijent arhitekturu, ali zbog otežanog preslikavanje 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 pohranju kalendari, 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 kroisti različite rute za prikazivanje stranica te dohvaćanje i manipulaciju podataka u bazi.

## Organizacija aplikacije

Organizacija aplikacije na složenijoj razini, uključujući strukturu slojeva i komponenta 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, visokopouzdan 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	
name	text	
tbl_name	text	
rootpage	int	
sql	text	

\_EFMigrationsHistory

Property	Type	Description
ProductVersion	text	
MigrationId	text	

\_EFMigrationsHistory primary key (MigrationId)

sqlite\_sequence

Property	Type	Description
name	unknown	
seq	unknown	

AspNetUsers

Property	Type	Description
AccessFailedCount	integer	
ConcurrencyStamp	text	
Email	text	
EmailConfirmed	integer	
LockoutEnabled	integer	
LockoutEnd	text	



NormalizedEmail	text	
NormalizedUserName	text	
PasswordHash	text	
PhoneNumber	text	
PhoneNumberConfirmed	integer	
SecurityStamp	text	
TwoFactorEnabled	integer	
UserName	text	
Id	text	

AspNetUsers primary key(Id)

AspNetUserTokens

Property	Type	Description
Value	text	
UserId	text	
LoginProvider	text	
Name	text	

AspNetUserTokens primary key(UserId, LoginProvider, Name)

AspNetRoles

Property	Type	Description
ConcurrencyStamp	text	
Name	text	
NormalizedName	text	
Id	text	

AspNetRoles primary key (Id)

AspNetUserRoles

Property	Type	Description
UserId	text	
RoleId	text	

AspNetUserRoles primary key (UserId, RoleId)

AspNetUserRoles foreign key (UserId)

AspNetUserRoles foreign key (RoleId)

AspNetRoleClaims

Property	Type	Description
ClaimType	text	
ClaimValue	text	
RoleId	text	
Id	integer	

AspNetRoleClaims primary key (Id)  
AspNetRoleClaims foreign key (RoleId)

AspNetUserClaims

Property	Type	Description
ClaimType	text	
ClaimValue	text	
UserId	text	
Id	integer	

AspNetUserLogins

Property	Type	Description
ProviderDisplayName	text	
UserId	text	
LoginProvider	text	
ProviderKey	text	

AspNetUserLogins primary key (LoginProvider, ProviderKey)  
AspNetUserLogins forieng key (UserId)

Calendars

Property	Type	Description
CsvPath	text	
CalendarId	integer	

Calendars primary key (CalendarId)

UserCalendars

Property	Type	Description
UserId	integer	

CalendarId	integer	
------------	---------	--

UserCalendars primary key (UserId, CalendarId)

## Dijagram baze podataka

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

sqlite_master	
type	text
name	text
tbl_name	text
rootpage	int
sql	text

_EFMigrationsHistory	
ProductVersion	text
MigrationId	text

sqlite_sequence	
name	unknown
seq	unknown

AspNetUsers	
AccessFailedCount	integer
ConcurrencyStamp	text
Email	text
EmailConfirmed	integer
LockoutEnabled	integer
LockoutEnd	text
NormalizedEmail	text
NormalizedUserName	text
PasswordHash	text
PhoneNumber	text
PhoneNumberConfirmed	integer
SecurityStamp	text
TwoFactorEnabled	integer
UserName	text
Id	text

AspNetUserTokens	
Value	text
UserId	text
LoginProvider	text
Name	text

AspNetRoles	
ConcurrencyStamp	text
Name	text
NormalizedName	text
Id	text

Calendars	
CsvPath	text
CalendarId	integer

RoleId:Id

UserId:Id

RoleId:Id

UserId:Id

UserId:Id

UserId:Id

CalendarId

AspNetUserRoles	
UserId	text
RoleId	text

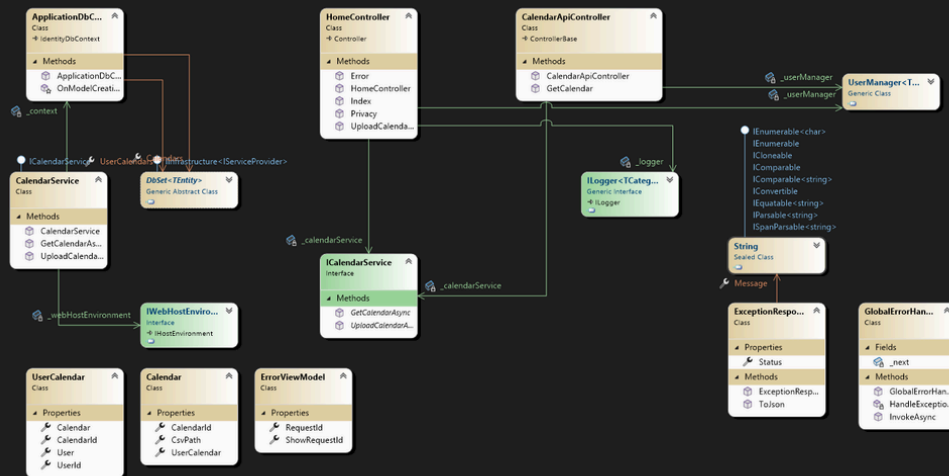
AspNetUserLogins	
ProviderDisplayName	text
UserId	text
LoginProvider	text
ProviderKey	text

AspNetRoleClaims	
ClaimType	text
ClaimValue	text
RoleId	text
Id	integer

AspNetUserClaims	
ClaimType	text
ClaimValue	text
UserId	text
Id	integer

UserCalendars	
UserId	text
CalendarId	integer

## Dijagram razreda



Reference koje su se koristile u datumu pisanja ove dokumentacije (12. 11. 2024):

[Getting Started — OAuth](#)

[Using OAuth and Cookies in Browser Based Apps | Best Practices | Curity](#)

[Introduction to Identity on ASP.NET Core](#)

[ASP.NET MVC Pattern | .NET](#)

[What is monolithic architecture in software? | Definition from Tech...](#)

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 primijenjeni 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 odabranih 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. Dovušenu aplikaciju potrebno je pokrenuti na javno dostupnom



## 7. Zaključak i budući rad



### *dio 2. revizije*

U ovom poglavlju 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

## 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 wiki-u 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 će biti izlistane kao imena Jira ticekta te uz njih nadopisano ime *glavne osobe* tj osobe kojoj je taj ticket dodijenjen. Na nekim ticketima radilo je više ljudi što je odraženo kao uloženi sati.

- bice dodano prije predaje

Ime člana tima	Broj uložениh sati
Belina Filip	61
Lovaković Jakov	83
Marušić Leo	79
Badel Jan	33
Lalić Jan	41
Marinović Mislav	41
Šainčević Martin	42

## Dijagram pregleda promjena

Prenijeti dijagram pregleda promjena nad datotekama projekta. Potrebno je na kraju projekta generirane grafove s githuba prenijeti u ovo poglavlje dokumentacije. Dijagrami za vlastiti projekt se mogu preuzeti s [GitHub - Build and ship software on a single, collaborative platfo](#)  stranice, u izborniku Repository, pritiskom na stavku Contributors.

## Ključni izazovi i rješenja

- Zaključno
- Opis izazova: Glavni izazovi tijekom projekta (npr. kašnjenje u razvoju, tehnički problemi).
- Rješenja: Način na koji su izazovi riješeni, kao i naučene lekcije koje su doprinijele napretku tima.



# 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

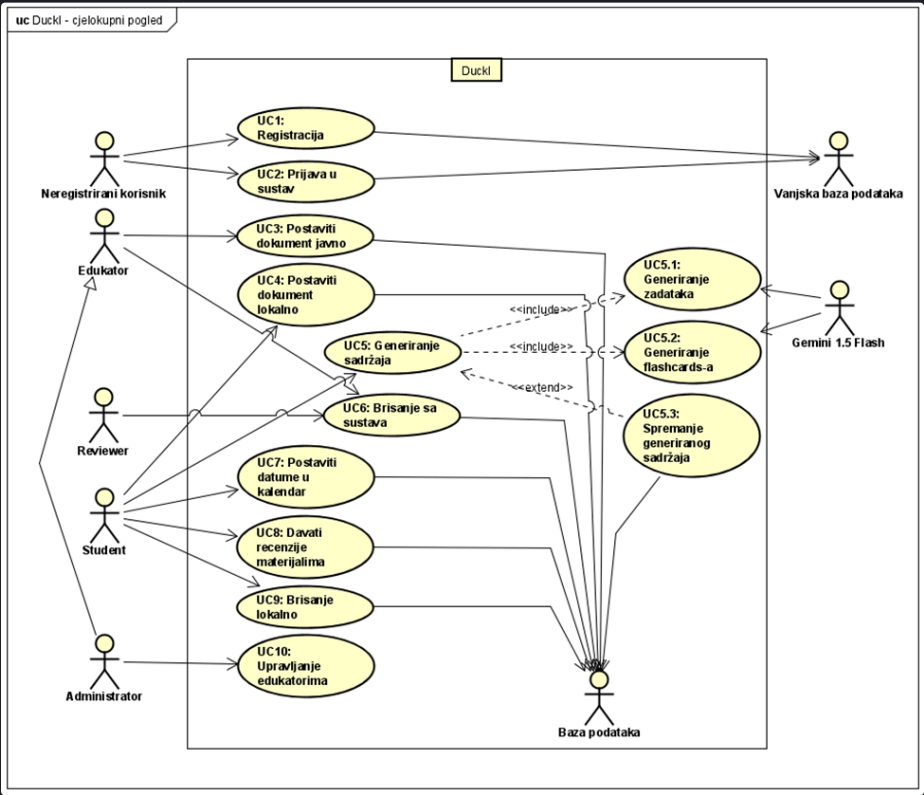
Sudionici: Mislav Marinković, Filip Belina, Jakov Lovaković, Leo Marušić

Dnevni red:

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

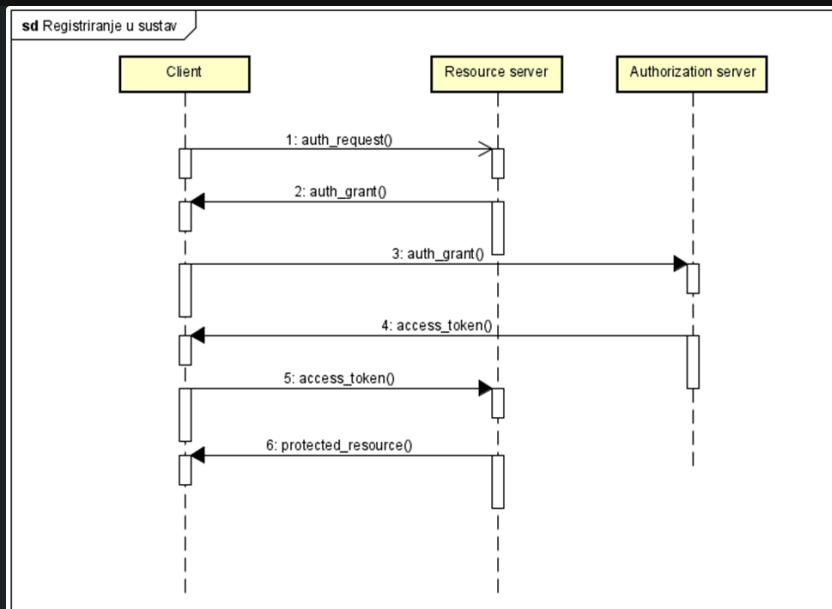
Dijagram obrazaca uporabe

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

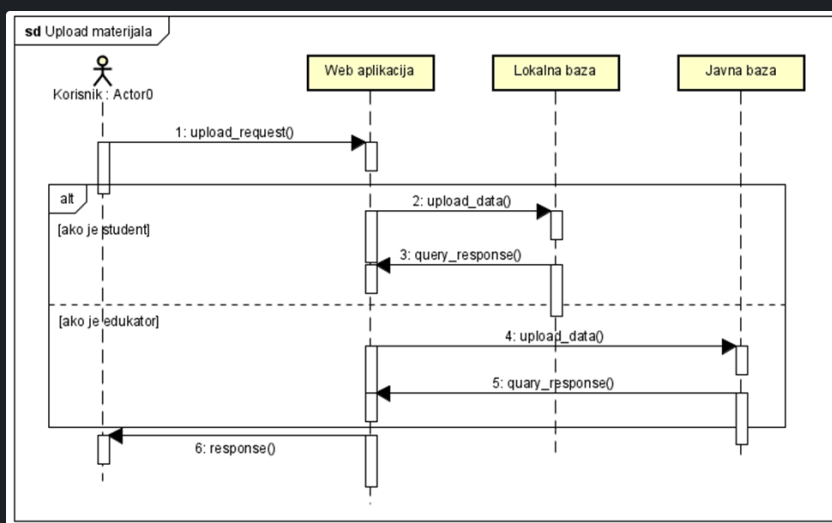


Sekvencijski dijagrami

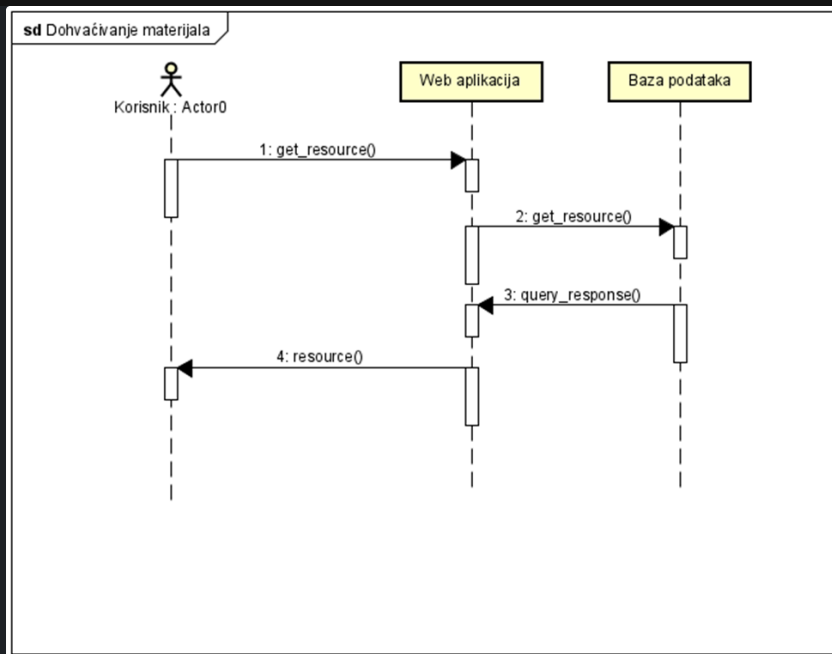
Registriranje u sustav



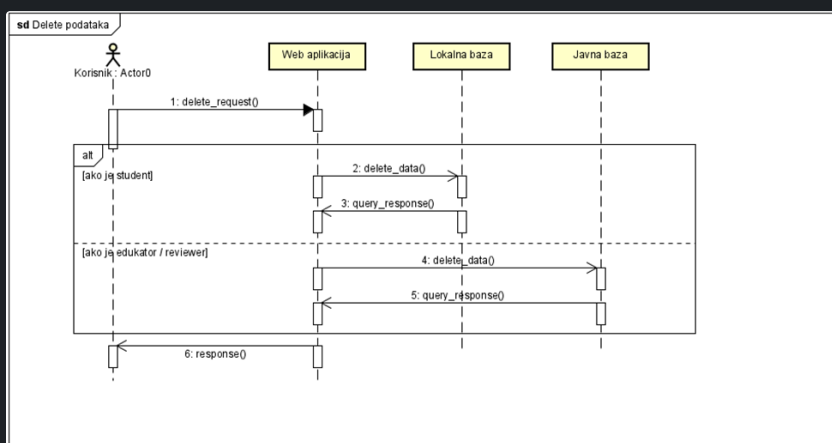
### Upload materijala



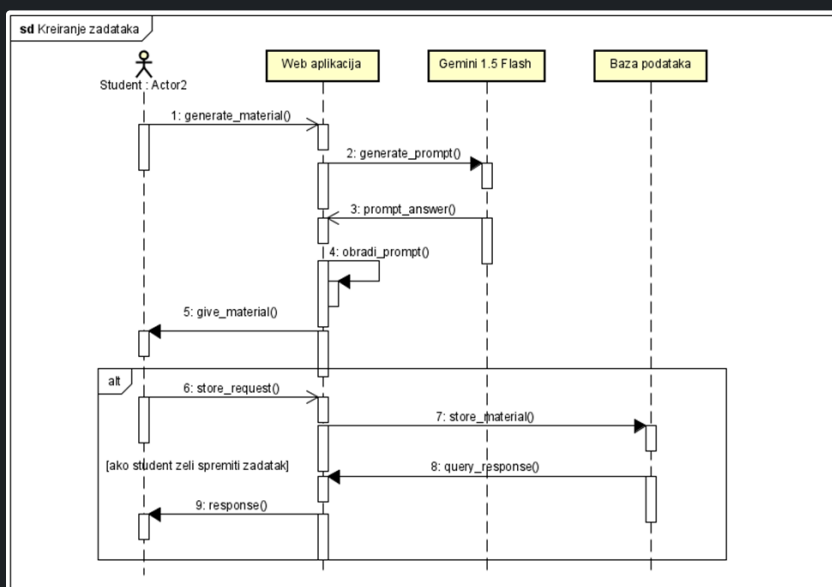
### Dohvaćivanje materijala



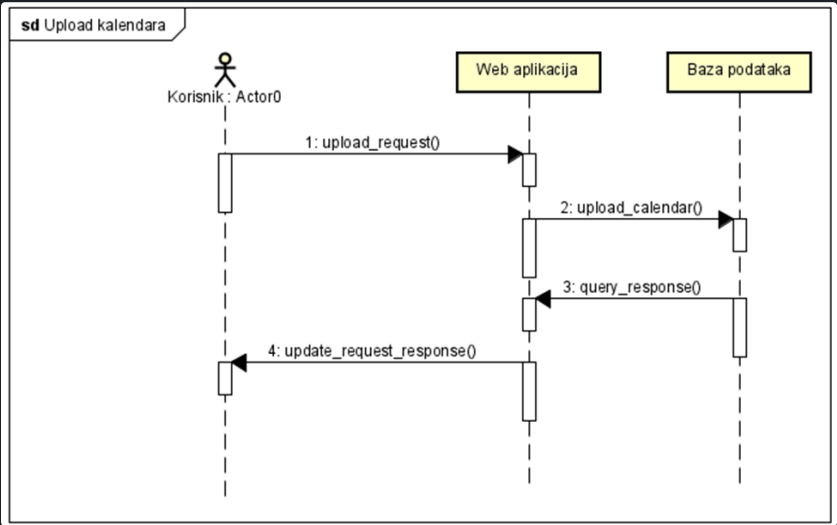
### Brisanje materijala



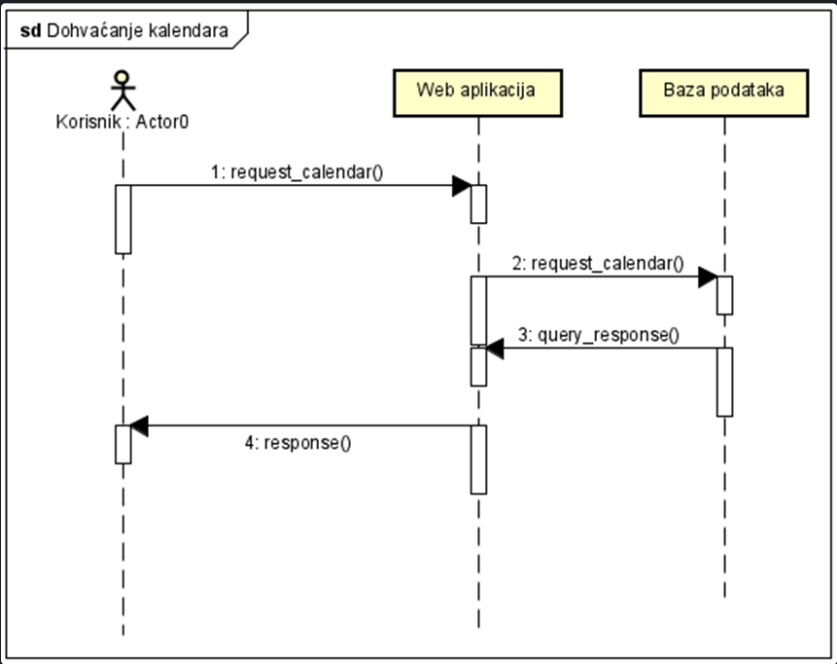
### Kreiranje zadataka



Upload kalendara

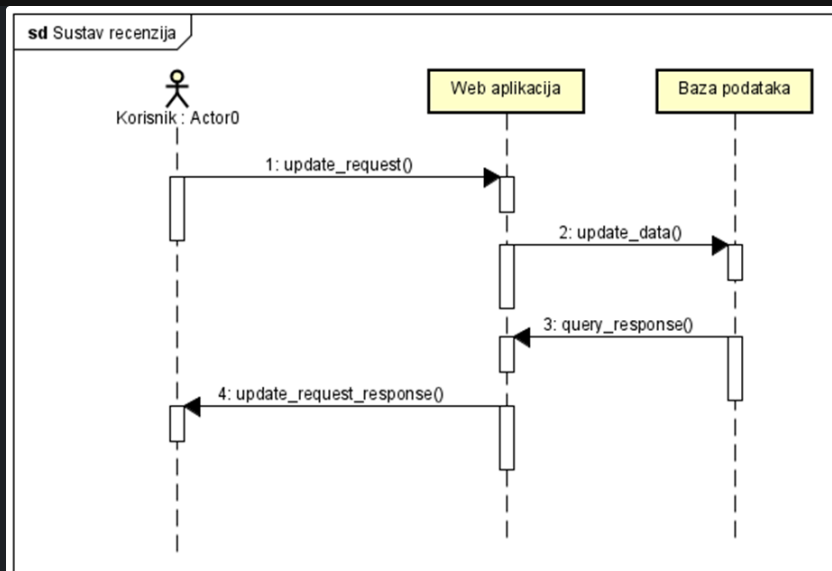


Dohvaćivanje kalendara

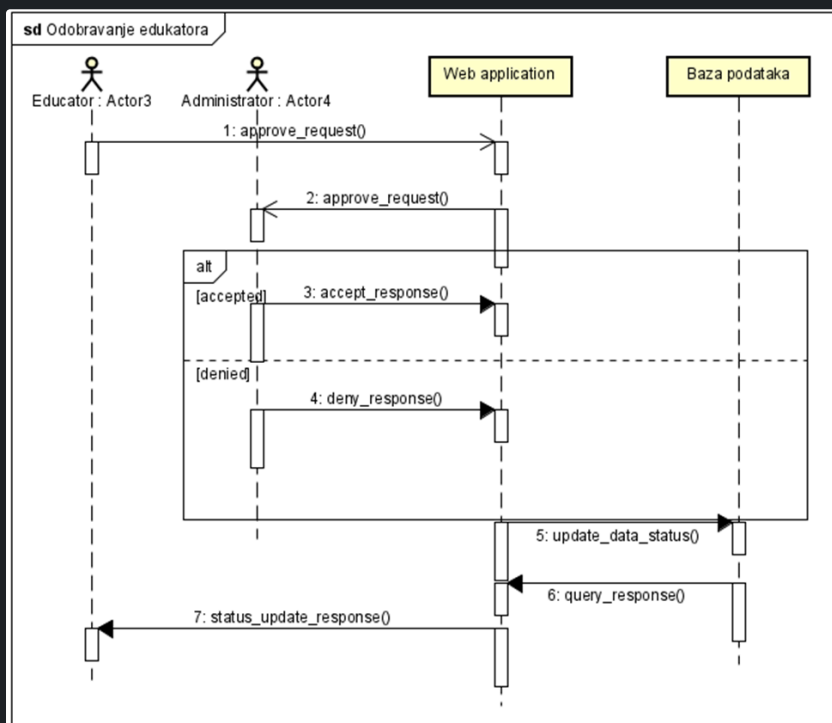


Sustav recenzija





#### Odobrovanje 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 repozitoriju.
- **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

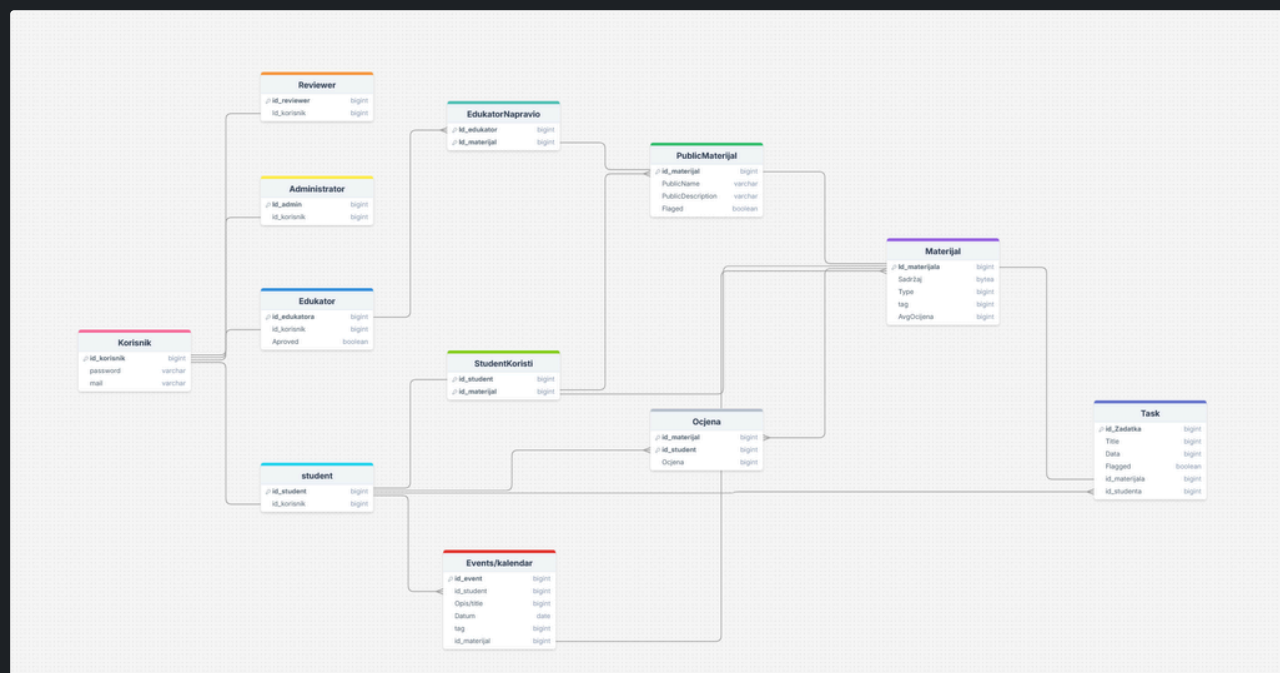
Nakon primanja dodatnih zahtjeva za projekt, neke od opcionalnih funkcionalnosti sada su postale obavezne. To je utjecalo na dizajn baze podataka i zahtijevalo je redizajn.

### 2. Redizajn sheme baze podataka

Schema baze podataka je redizajnirana 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:**



# 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

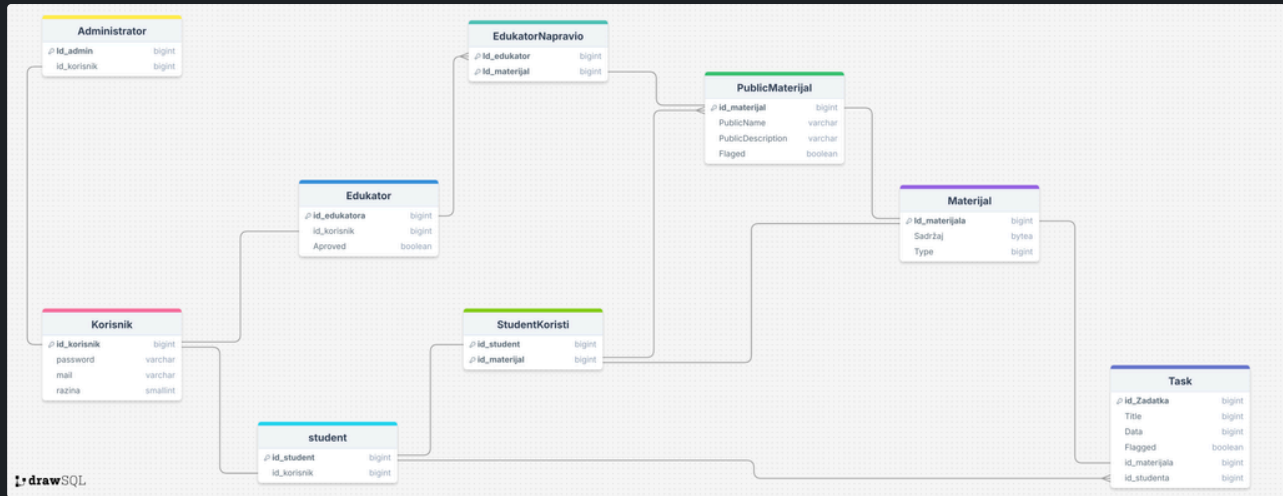
### Dnevni red:

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).
- **Attribute:** 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):** Objasnjeno 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.
- **Timska 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 - Prject 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.

