# **Hello Work**

-Portal za pronalazak posla-

Arhitekturni projekat

|  |  |  |
| --- | --- | --- |
|  |  |  |

# **Pregled izmena**

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 02.03.2024. | 1.0 | Predlog projekta | Đorđe Ivanović  Aleksa Jovanović  David Stanojević |
| 10.03.2024. | 1.0 | Revizija | Đorđe Ivanović  Aleksa Jovanović  David Stanojević |
| 17.03.2024. | 1.0 | Inicijalna verzija | Đorđe Ivanović  Aleksa Jovanović  David Stanojević |
| 31.03.2024. | 1.0 | Specifikacija zahteva | Đorđe Ivanović  Aleksa Jovanović  David Stanojević |
| 26.04.2024 | 1.0 | Arhitekturni projekat | Đorđe Ivanović  Aleksa Jovanović  David Stanojević |

# **Sadržaj**

[**Hello Work** 1](#_Toc165197137)

[**Pregled izmena** 2](#_Toc165197138)

[**Sadržaj** 3](#_Toc165197139)

[**Arhitekturni projekat** 4](#_Toc165197140)

[**1.** **Cilj dokumenta** 4](#_Toc165197141)

[**2.** **Opseg dokumenta** 4](#_Toc165197142)

[**3.** **Reference** 4](#_Toc165197143)

[**4.** **Predstavljanje arhitekture** 4](#_Toc165197144)

[**5.** **Ciljevi i ograničenja arhitekture** 4](#_Toc165197145)

[**6.** **Pogled na slučajeve korišćenja** 4](#_Toc165197146)

[**6.1.** **Dijaglami slučajeva korišćenja** 5](#_Toc165197147)

[**6.2.** **Kratak opis sličajeva korišćenja** 6](#_Toc165197148)

[6.2.9. Ažuriranje podataka o članu 7](#_Toc165197149)

[**7.** **Pogled na logičku arhitekturu sistema** 7](#_Toc165197150)

[**7.1.** **Pregled arhitekture – organizacija paketa i podsistema u slojeve** 8](#_Toc165197151)

[**8.** **Pogled na procese** 9](#_Toc165197152)

[**8.1.** **Procesi** 9](#_Toc165197153)

[**8.1.1.** **Veb čitač** 9](#_Toc165197154)

[**8.1.2.** **Veb server** 9](#_Toc165197155)

[**8.1.3.** **Apache mod\_php** 10](#_Toc165197156)

[**8.1.4.** **MySQL Server** 10](#_Toc165197157)

[**9.** **Pogled na raspoređivanje sistema** 10](#_Toc165197158)

[**9.1.** **Klijent** 10](#_Toc165197159)

[**9.2.** **Web server** 10](#_Toc165197160)

[**9.3.** **DBMS server** 10](#_Toc165197161)

[**10.** **Pogled na implementaciju sistema** 10](#_Toc165197162)

[**10.1.** **Model domena** 11](#_Toc165197163)

[**10.2.** **Šema baze podataka** 12](#_Toc165197164)

[**10.3.** **Komponenta sistema** 12](#_Toc165197165)

[**10.3.1.** **Komponente korisnickog interfejsa** 12](#_Toc165197166)

[**10.3.2.** **Komponente aplikacione logike** 13](#_Toc165197167)

[**10.3.3.** **Komponente za pristup podacima** 14](#_Toc165197168)

[**11.** **Performanse** 14](#_Toc165197169)

[**12.** **Kvalitet** 15](#_Toc165197170)

[**12.1.** **Tehnoloske inovacije** 15](#_Toc165197171)

[**12.2.** **Posvecenost kvalitetu** 15](#_Toc165197172)

# **Arhitekturni projekat**

1. **Cilj dokumenta**  
   Cilj ovog dokumenta je predstaviti osnovne ideje i zahteve za razvoj softverske platforme koja omogućava objavljivanje i pregledanje poslova od strane firmi i korisnika. Ovaj dokument služi kao osnova za dalji razvoj projekta, pružajući jasnu sliku o funkcionalnostima, tehnološkim zahtevima i očekivanim rezultatima. Takođe, cilj je uspostaviti osnovne smernice i ciljeve kako bi se olakšalo razumevanje svrhe i obima projekta svim zainteresovanim stranama, uključujući razvojni tim, klijente i potencijalne korisnike.

## **Opseg dokumenta**

Dokument opisuje temu i osnovne karakteristike projekta, motivaciju i potrebna znanja za njegovu izradu, motivaciju i osobine članove tima, komunikaciju među njima i vreme potrebno za izradu projekta.

## **Reference**

Spisak korišćene literature:

1. Hello Work – Predlog projekta, Aurora Team 2024
2. Hello Work –Vizija sistema, Aurora Team 2024
3. Hello Work – Plan realizacije projekta, Aurora Team 2024
4. Hello Work – Planirani raspored akrivnosti na projektu, Aurora Team 2024
5. Hello Work – Specifikacija zahteva, Aurora Team

## **Predstavljanje arhitekture**

Arhitektura sistema u dokumentu je prikazana kao serija pogleda na sistem: pogled na slučajeve korišćenja, pogled na logičku arhitekturu sistema, pogled na procese, pogled na razmeštaj komponenti sistema i pogled na implementaciju. Ovi pogledi su predstavljeni odgovarajućim UML dijagramima.

## **Ciljevi i ograničenja arhitekture**

Ključni zahtevi i sistemska ograničenja koja imaju značajan uticaj na izbor arhitekture i projektovanje sistema su:

* 1. HelloWork platforma će biti implementiran kao Web aplikacija zasnovana na PHP jeziku sa framework-om Laravel i MySQL bazi podataka.
  2. Klijentski deo HelloWork platforme će biti optimizovan za sledeće Web čitače: Vivaldi, Brave, Google Chrome, Mozilla Firefox, Safari, Microsoft Edge
  3. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

## **Pogled na slučajeve korišćenja**

U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [5].

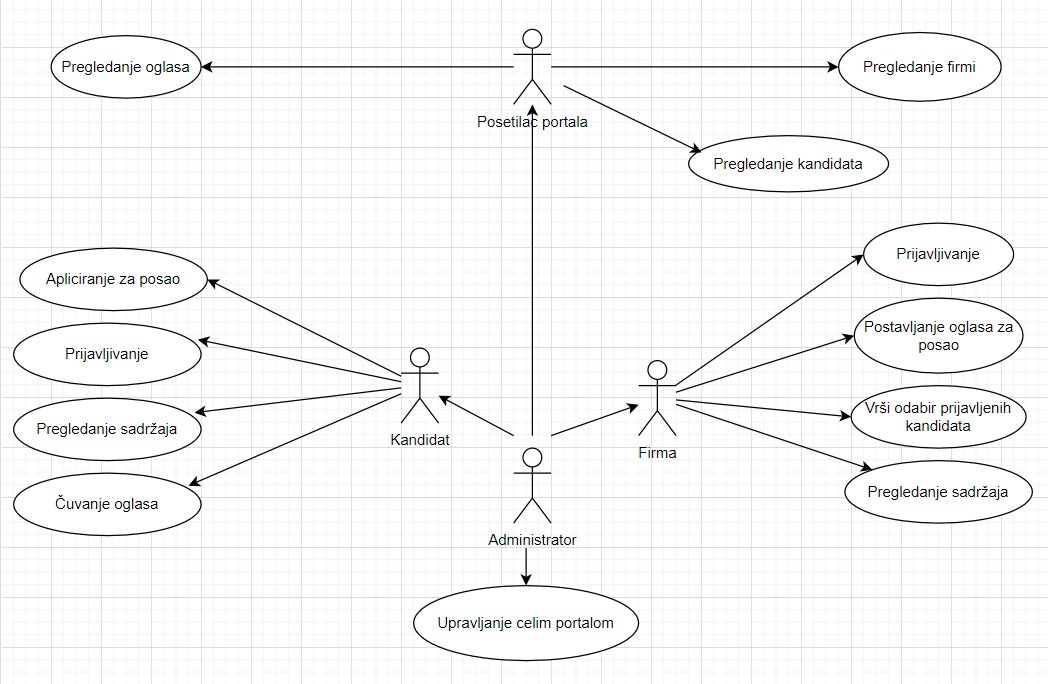
Slučajevi korišćenja HelloWork platforme su:

* + *Pregledanje oglasa*
    - Pregled osnovnih informacija o oglasu
    - Pregled osnovnih informacija o poslodavcu
  + *Pregledanje firme*
    - Pregled informacija o firmi (poslodavcu)
    - Pregled svih oglasa poslodavca
  + *Pregledanje kandidata*
    - Pregled informacija o kandidatu
    - Pregled svih apliciranja kandidata
  + Apliciranje za posao
  + Prijavljivanje
  + *Pregledanje sadržaja*
    - Pregled oglasa za posao
    - Pregled profila poslodavca
    - Filtriranje poslova
    - Pretraživanje poslova
  + Čuvanje oglasa
  + Postavljanje oglasa za posao
  + Ažuriranje informacija
  + Odibar prijavljenih kandidata
  + Upravljanje celim portalom

Ove slučajeve korišćenja mogu da iniciraju posetioci platforme, kandidat, poslodavac i administrator.

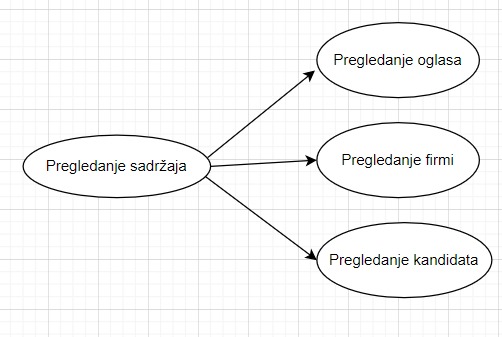
### **Dijaglami slučajeva korišćenja**

Osnovni UML dijagram koji prikazuje korisnike i slučajeve korišćenja Hello Work portala prikazan je na sledećoj slici:

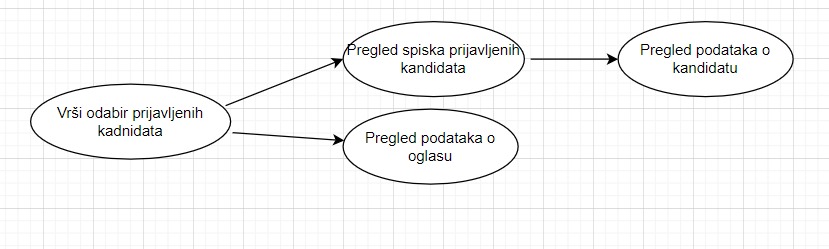


Slučaj korišćenja *pregled sadržaja*  i *vrši odabir prijavljenih kandidata* obuhvata složenije radnje koje se mogu razložiti dalje na pojedinačne slučajeve korišćenja.

Detaljni UML dijagram za slučaj korišćenja *pregled sadržaja* je prikazan na sledećoj slici:



Detaljan UML dijagram za slučaj korišćenja *odabir prijavljenih kandidata* je prikazan na sledećoj slici:



### **Kratak opis sličajeva korišćenja**

#### Pregledanje oglasa

Kratak opis:Pregledanje oglasa o poslovima koji se nalaze na portalu

Akteri koji iniciraju slučaj korišćenja:Posetilac profila, kandidat, firma, administrator

#### Pregledanje firmi

Kratak opis:Pregledanje profila firmi koji se nalaze na portalu

Akteri koji iniciraju slučaj korišćenja:Posetilac profila, kandidat, firma, administratorNema

#### Pregledanje kandidata

Kratak opis:Pregledanje profila firmi koji se nalaze na portalu

Akteri koji iniciraju slučaj korišćenja:Posetilac profila, kandidat, firma, administrator

#### Postavljenje oglasa za posao

Kratak opis:Firma tjs. poslodavac postavlja oglas za posao koji nudi i koji će moći da pregledaju svi korisnici platforme

Akteri: Firma

#### Prijavljivanje

Kratak opis:Prijavljivanje korisnika na portal u cilju pristupa specifičnim funkcijama koji zahtevaju autorizaciju.

Akteri koji iniciraju slučaj korišćenja:Kandidat, firma, administrator

#### Apliciranje za posao

Kratak opis:Korisnik pregleda oglase za koje je zainteresovan. Kada pronađe adekvatan posao može da aplicira na njega.

Akteri koji iniciraju slučaj korišćenja:Kandidat

* + 1. Čuvanje oglasa

Kratak opis:Kandidat ima mogućnost da sačuva oglas za koji misli da će mu biti kasnije p otreban. Tom sačuvanom oglasu može da pristupi iz sekcije *Sačuvani oglasi*

Akteri koji iniciraju slučaj korišćenja: Kandidat, firma

#### Upravljanje celim portalom

Kratak opis:Administrator portala može da manipuliše sadržajem celog portala, briše oglase, briše profile...

Akteri: Administrator

### Ažuriranje podataka o članu

Kratak opis:Ažuriranje podataka o sebi od strane prijavljenog člana

Akteri koji iniciraju slučaj korišćenja: Kandidat, firma, administrator

## **Pogled na logičku arhitekturu sistema**

U ovom odeljku je dat pregled logičke arhitekture sistema. Ovaj pogled sadrži opis najznačajnijih klasa, njihove organizacije u pakete i podsisteme, i organizacija podsistema u slojeve. U cilju opisivanja dinamičkih aspekata arhitekture, ovaj odeljak može da uključi opise realizacije najznačajnijih slučajeva korišćenja. Da bi se ilustrovala veza između arhitekturno značajnih klasa, podsistema, paketa ili slojeva moguće je uključiti i odgovarajuće dijagrame klasa.

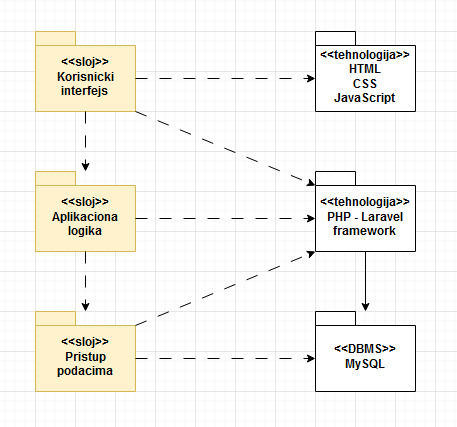
Logički pogled na HelloWork platformu obuhvata 3 glavna paketa: Korisnički interfejs, Aplikaciona logika, Pristup podacima.

Paket *Korisnički interfejs* sadrži Web stranice, PHP Laravel kodove i multimedijalni sadržaj koji realizuju grafički dizajn i forme preko kojih korisnici sistema komuniciraju sa sistemom.

Paket *Aplikaciona logika* predstavlja srednji sloj sistema koji sadrži PHP Laravel kodove zadužene za realizaciju funkcionalnosti specifičnih za domen sistema koji se razvija.

Paket *Pristup podacima* sadrži PHP Laravel kodove koje predstavljaju interfejs za pristup, dodavanje i ažuriranje podataka koji se čuvaju u bazi podataka.

### **Pregled arhitekture – organizacija paketa i podsistema u slojeve**



#### Korisnični interfejs

Sloj

Ovaj sloj realizuje korisnički interfejs portala. U njemu su sadržane sve HTML, multimedijalni sadržaji i PHP skripte koje generišu HTML stranice preko kojih korisnici komuniciraju sa sistemom.Sloj korisničkog interfejsa zavisi od sloja aplikacione logike, kao i paketa HTML i PHP.

#### Aplikaciona logika

Sloj

Sloj aplikacione logike je srednji sloj u troslojnoj arhitekturi HelloWork platforme. Sadrži Php Laravel kod skripte koje realizuju funkcionalnost karakterističnu za domen primene portala i uspostavljaju vezu između korisničkog interfejsa i sloja za pristup podacima.

Ovaj sloj zavisi od sloja za pristup podacima i PHP Laravel-a.

#### Pristup podacima

Sloj

Sloj za pristup podacima se nalazi na dnu troslojne arhitekture i sadrži PHP Laravel skripte zadužene za pribavljanje, dodavanje i ažuriranje podataka koji se čuvaju u MySQL bazi podataka.

Ovaj sloj ne zavisi od drugih slojeva, ali je zavisan od paketa PHP i MySQL baza podataka.

#### HTML

Tehnologija

Tehnologija HTML-a definiše gradivne elemente stranica koje se prikazuju u Web čitaču i koje omogućavaju prikaz formatiranih informacija i realizaciju formi za unos i ažuriranje podataka.

#### Php Laravel

Tehnologija

PHP Laravel obezbeđuje mehanizam za pisanje i izvršavanje koda na strani servera. Radai se o MVC frameworku koji obezbeđuje bolju organizaciju projekta. Koriste se i PHP skripte za generisanje HTML stranica za korisnički interfejs. Pomoću ovog framework-a se vrši pristupanje bazi podatak u cilju pribavljanja, unosa i ažuriranja podataka.

#### MySQL

DBMS

MySQL predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju PeNcIL portala.

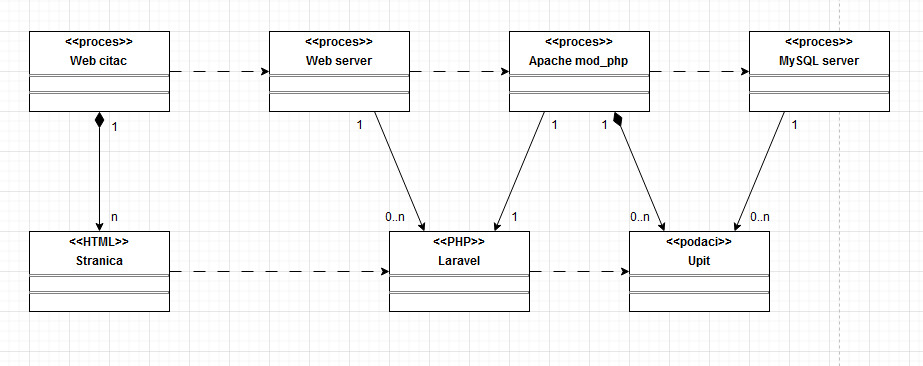
## **Pogled na procese**

U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe spada u opis procesne arhitekture. Web aplikacije zasnovane na PHP-u imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanta PHP Web aplikacije nije potrebno voditi računa o načinu rada Web servera i načinu izvršavanja skripti.

Ilustracije radi u nastavku je dat opis procesa uključenih u izvršenje HelloWork platforme kao Web aplikacije.

### **Procesi**

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju HelloWork platforme. Dijagram je opšteg tipa i može se primeniti na bilo koju Web aplikaciju zasnovanu na PHP-u i MySQL bazi podataka.



### **Veb čitač**

Web čitač je proces koji izvršava funkcionalnost aplikacije za prikaz HTML stranica dobijenih od nekog Web servera. U najopštijem slučaju Web čitač u jednom trenutku može da prikazuje samo jednu HTML stranicu. Web čitač zavisi od Web servera koji generiše i vraća odgovarajuću HTML stranicu na zahtev.

### **Veb server**

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača. Ukoliko je zahtevana stranica PHP skript, Web server inicira izvršenje Apache mod\_php procesa koji obrađuje odgovarajući skript i generiše sadržaj koji se vraća čitaču. Web server može paralelno da inicira veći broj Apache mod\_php procesa.

### **Apache mod\_php**

Apache mod\_php omogućava Apache web serveru da direktno interpretira PHP skripte, bez potrebe za zasebnim PHP-CGI procesom. Kada web server prima zahtev za PHP stranicu, mod\_php direktno učitava i izvršava PHP kod unutar Apache procesa. Ovo eliminira dodatni sloj komunikacije između web servera i PHP interpretera, što može rezultirati bržim izvršavanjem PHP skripti.

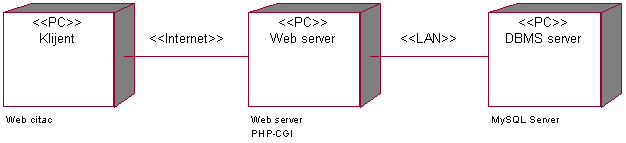
### **MySQL Server**

MySQL Server je proces koji izvršava funkcionalnost MySQL sistema za upravljanje bazama podataka. Ovaj proces može konkurentno da prihvati određen broj upita, izvrši ih nad bazom podataka i vrati rezultate procesu koji je upite postavio.

## **Pogled na raspoređivanje sistema**

Pogled na raspoređivanje sistema prikazuje različite fizičke čvorove za najopštiju konfiguraciju sistema. Fizičkim čvorovima koji predstavljaju procesore vrši se dodeljivanje identifikovanih procesa.

Na sledećoj slici dat je UML dijagram raspoređivanja HelloWork platforme.



### **Klijent**

Pristup HelloWork platforme se obavlja preko klijentskih računara na kojima se izvršava Web čitač. Za povezivanje između klijenta i Web servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

### **Web server**

Računar na kome se izvršava Web server opslužuje više klijenata koji pristupaju preko Interneta. Pored osnovnog procesa koji realizuje funkcionalnost Web servera, na ovom računaru mogu da se izvršavaju i procesi Apache mod\_php koji vrše obradu zadatih PHP skripti. U najopštioj konfiguraciji DBMS se izvršava na posebnoj mašini koja je sa Web serverom u lokalnoj mreži (LAN).

### **DBMS server**

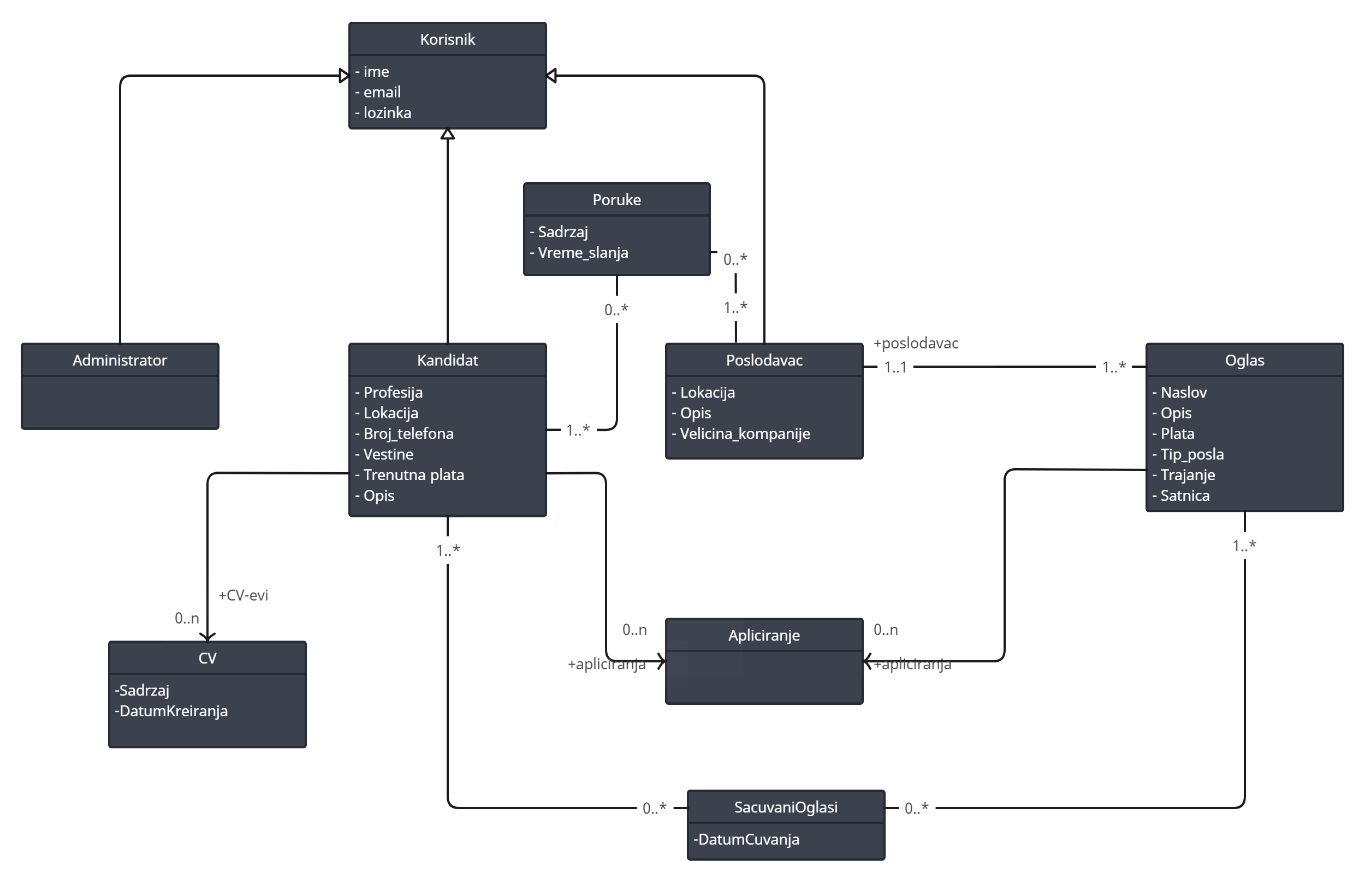
DBMS server je računar na kome se izvršava MySQL Server proces koji realizuje funkcionalnost sistema za upravljanje bazama podataka.

## **Pogled na implementaciju sistema**

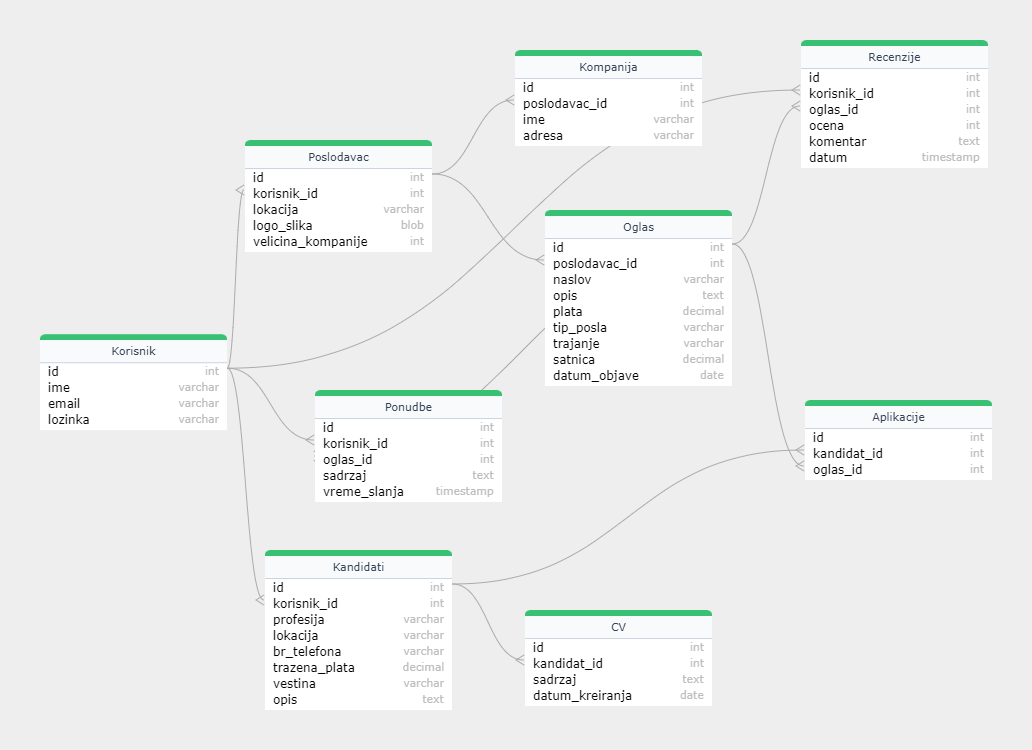
Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju HelloWork platforme ovaj odeljak sadrži model domena, šemu baze podataka i prikaz komponenti sistema razvrstanih u ranije identifikovane pakete.

### **Model domena**

Model domena za koji se HelloWork platforma projektuje je ilustrovan UML dijagramom klasa. U njemu su prikazane domenske klase, neki od njihovih atributa, kao i veze koje se mogu identifikovati između njih.Model domena predstavlja osnovu za projektovanje baze podataka, ali i identifikaciju nekih od komponenti koje će biti implementirane.



### **Šema baze podataka**

Detaljna šema baze podataka je prikazana na sledećem dijagramu

### **Komponenta sistema**

Komponente sistema HelloWork su PHP skripti čiji će pregled biti dat po arhitekturnim slojevima. Za ilustraciju će biti korišćeni UML dijagrami komponenti, ali i dijagrami klasa. U slučajevima gde je PHP skript prikazan kao klasa atributi predstavljaju ulazne podatke koji se uzimaju iz GET ili POST dela HTTP poruke, dok metodi predstavljaju funkcije definisane u okviru skripta.

### **Komponente korisnickog interfejsa**

Dizajn korisničkog interfejsa je obuhvaćen dvema komponentama:

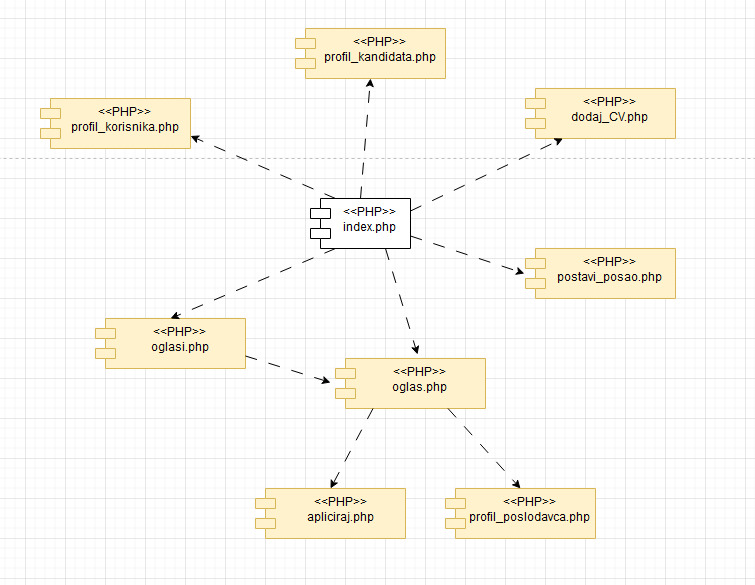


Komponenta **index.php** je implementira stranicu portala čiji sadržaj može da varira od parametra koji joj se proslede u zahtevu.

Komponenta **main.css** predstavlja opis stilova za pojedine HTML elemente koji se javljaju na različitim stranicama.

### **Komponente aplikacione logike**

Komponente koje realizuju domen problema se uključuju isključivo preko **index.php** komponente korisničkog interfejsa. Na taj način zadržavaju sva podešavanja stila definisana u ovom skriptu. Na sledećem dijagramu su prikazane komponente ovog sloja i njihove međusobne zavisnosti:



### **Komponente za pristup podacima**

U arhitekturi našeg vebsajta, ključnu ulogu igraju komponente za pristup podacima. Koristimo Laravel, koji nam omogućava da pišemo čist i izražajan kod za interakciju sa bazom podataka. Laravel koristi njegov ORM (Object-Relational Mapping) sistem poznat kao Eloquent. Eloquent nam omogućava da definisemo naše modele koji odgovaraju tabelama u bazi podataka, a zatim koristimo te modele da vršimo operacije nad podacima.

Modeli i Eloquent

Svaki model u Laravelu je direktan odraz tabele u bazi podataka, ali sa dodatnim supermoćima. Na primer, ako imamo tabelu korisnici, kreiramo Eloquent model Korisnik. Ovaj model nam služi kao vrata kroz koja pristupamo podacima. Eloquent nam omogućava da koristimo razne ugrađene funkcije za manipulaciju podacima:

* create(): Brzo dodavanje novih zapisa u bazu.
* update(): Ažuriranje postojećih zapisa sa novim informacijama.
* find(): Pronalaženje zapisa po primarnom ključu.
* findOrFail(): Slično kao find(), ali baca izuzetak ako zapis nije pronađen.
* delete(): Uklanjanje zapisa iz baze.

Pored ovih, koristimo i druge naprednije metode kao što su:

* where(): Filtriranje zapisa po određenim kriterijumima.
* firstOrCreate(): Pronalazi prvi zapis koji odgovara uslovima ili ga kreira ako ne postoji.
* updateOrCreate(): Ažurira zapis koji odgovara uslovima ili ga kreira ako ne postoji.
* chunk(): Procesuiranje velikih skupova rezultata u manjim “komadima” radi bolje efikasnosti.

Veze između modela

Eloquent takođe olakšava definisanje veza između modela. Na primer, ako Korisnik ima mnogo Poruka, jednostavno definišemo metodu poruke() unutar modela Korisnik koja vraća rezultat metode hasMany(). Ovo nam omogućava da pristupimo svim porukama korisnika kroz izraz $korisnik->poruke.

## **Performanse**

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu i vremena odziva za pristup bazi podataka specificirane u zahtevima u pogledu performansi [5]:

* Sistem će da podrži do 10000 simultanih pristupa korisnika portalu.
* Vreme potrebno za pristupanje bazi podataka u cilju izvršenje nekog upita ne sme da bude veće od 4 sekundi.

Zahtevane performanse su zadovoljene izborom tehnologija na kojima će sistem biti razvijen i definisane hardverske platforme [5].

## **Kvalitet**

Naša platforma HelloWork je dizajnirana sa fokusom na visoku dostupnost i pouzdanost, kako bismo osigurali da naši korisnici imaju neprekidan pristup uslugama koje pružamo. U skladu sa tim, postavili smo sledeće standarde:

* **Dostupnost**: HelloWork platforma će biti dostupna 24/7/365, garantujući da korisnici mogu pristupiti našim uslugama u bilo koje vreme. Težimo ka tome da vreme nedostupnosti platforme ne prelazi 5% tokom bilo kog meseca, što je znatno ispod industrijskog standarda.
* Srednje vreme između otkaza (**MTBF**): Naš cilj je da srednje vreme između dva sukcesivna otkaza bude najmanje 100 sati. Ovo postižemo kroz rigorozno testiranje, redovno održavanje i upotrebu robusnih tehnologija.

### **Tehnoloske inovacije**

* Implementirali smo napredne tehnike **load balancinga** kako bismo ravnomerno rasporedili zahteve na više servera, čime se smanjuje rizik od preopterećenja i poboljšava ukupna performansa.
* Koristimo redundantne sisteme skladištenja kako bismo osigurali da su podaci sigurni i dostupni čak i u slučaju hardverskog kvara.
* Naša infrastruktura je dizajnirana da bude skalabilna, omogućavajući nam da brzo prilagodimo resurse u skladu sa rastućim zahtevima korisnika.

### **Posvecenost kvalitetu**

* Redovno sprovodimo automatizovane testove kako bismo osigurali da sve komponente sistema funkcionišu kako treba.
* Angažujemo stručnjake za cyber security kako bismo zaštitili našu platformu od spoljnih pretnji i osigurali sigurnost korisničkih podataka.
* Naš tim za podršku je dostupan 24/7 kako bi brzo reagovao na bilo kakve tehničke probleme ili upite korisnika.
* Zaključak je da HelloWork stavlja pouzdanost i dostupnost na prvo mesto, obezbeđujući da naši korisnici imaju stalni pristup kvalitetnim uslugama koje očekuju i zaslužuju.