Fitnessivity

Aplikacija za praćenje fitnes napretka

Arhitekturni projekat

Verzija 1.0

Pregled izmena

|  |  |  |  |
| --- | --- | --- | --- |
| **Datum** | **Verzija** | **Opis** | **Autor** |
| 22.04.2023. | 1.0 | Inicijalna verzija | Miloš Miljković, 19040  Dušan Stojanović, 17450 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Sadržaj

1. Cilj dokumenta 4

2. Opseg dokumenta 4

3. Reference 4

4. Predstavljanje arhitekture 4

5. Ciljevi i ograničenja arhitekture 4

6. Pogled na slučajeve korišćenja 4

6.1 Dijagrami slučajeva korišćenja 5

6.2 Kratak opis slučajeva korišćenja 5

6.2.1 Registracija 5

6.2.2 Pregled osnovnih informacija 5

6.2.3 Kreiranje i čuvanje programa za vežbanje 5

6.2.4 Prikaz statistike i praćenje napretka 5

6.2.5 Dodavanje prijatelja 6

6.2.6 Pregled programa po tipu 6

6.2.7 Prijava programa za neprikladan sadržaj 6

6.2.8 Pregled prijavljenih programa za neprikladan sadržaj 6

6.2.9 Pregled i unos recenzija 6

6.2.10 Prijava 6

6.2.11 Ažuriranje podataka o korisniku 6

6.2.12 Brisanje programa 6

6.2.13 Brisanje prijatelja 6

6.2.14 Prikaz reklama 6

6.2.15 Komunikacija između korisnika 6

7. Pogled na logičku arhitekturu sistema 6

7.1 Pregled arhitekture – organizacija paketa i podsistema u slojeve 7

8. Pogled na procese 8

8.1 Procesi 8

8.1.1 Web Pregledač 9

8.1.2 Web server 9

8.1.3 JavaScript 9

8.1.4 FitnessivityServer 9

9. Pogled na implementaciju sistema 9

9.1 Komunikacioni protokol 9

9.2 Komponente sistema 9

9.2.1 Komponente klijentske aplikacije 9

9.2.2 Komponente serverske aplikacije 9

10. Pogled na raspoređivanje sistema 9

10.1 Klijent 9

10.2 Server 9

11. Performanse 9

12. Kvalitet 10

Arhitekturni projekat

# Cilj dokumenta

Cilj ovog dokumenta je detaljni opis arhitekture Fitnessivity sistema.

# Opseg dokumenta

Dokument se odnosi na Fitnessivity sistem koji će biti razvijen od strane SWETeam-a. Fitnessivity predstavlja klijent-server sistem za chat-ovanje. Namena sistema je jednostavna komunikacija između trenutno prijavljenih korisnika slanjem tekstualnih poruka.

# Reference

Spisak korišćene literature:

1. Fitnessivity – Predlog projekta, Brogrammers-Fitnessivity-01, V1.0, 2023, Bogrammers.
2. Fitnessivity – Planirani raspored aktivnosti na projektu, Brogrammers-Fitnessivity-01mpp, V1.0, 2023, Brogrammers.
3. Fitnessivity – Plan realizacije projekta, Brogrammers-Fitnessivity-02, V1.0, 2023, Brogrammers.
4. Fitnessivity – Vizija sistema, Brogrammers-Fitnessivity-03, V1.0, 2023, Brogrammers.
5. Fitnessivity – Specifikacija zahteva, Brogrammers-Fitnessivity-04, V1.0, 2023, Brogrammers.

# 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. Klijent-server model komunikacije koji podrazumeva postojanje centralizovanog servera preko kog se vrši razmena poruka između učesnika [4].
2. Multiplatformska podrška. Treba obezbediti mogućnost izvršenja na MS Windows i Linux operativnim sistemima [4].
3. Svi zahtevi u pogledu performansi dati u [5] moraju biti uzeti u obzir pri izboru arhitekture i razvoju sistema.

# Pogled na slučajeve korišćenja

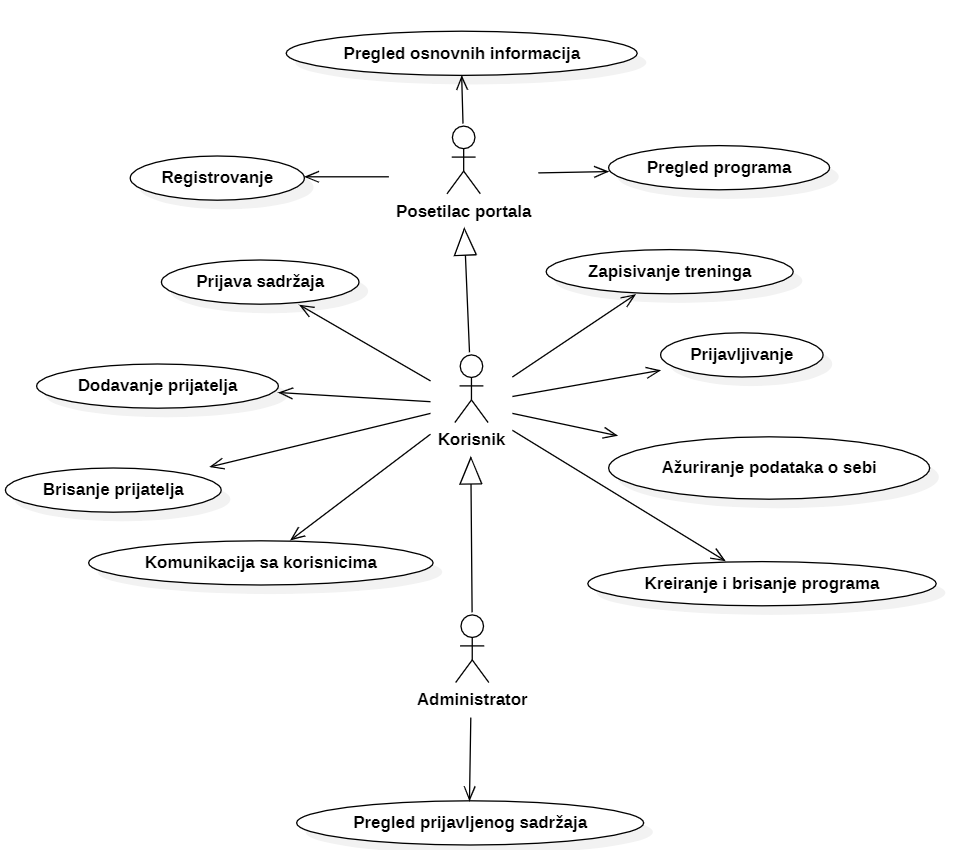
U ovom odeljku je dat pogled na slučajeve korišćenja definisane u specifikaciji zahteva [5].

Slučajevi korišćenja Fitnessivity sistema su:

* Registracija
* Pregled osnovnih informacija
* Prikaz statistike i praćenje napretka
* Dodavanje prijatelja
* Pregled programa po tipu
* Prijavljivanje programa za neprikladan sadržaj
* Pregled i unos recenzija
* Prijava
* Ažuriranje podataka o korisniku
* Brisanje programa
* Brisanje prijatelja
* Komunikacija između korisnika
* Prikazivanje reklama

Ove slučajevi korišćenja mogu da iniciraju administrator ili korisnik.

## Dijagrami slučajeva korišćenja

UML dijagram koji prikazuje korisnike i slučajeve korišćenja Fitnessivity sistema prikazan je na sledećoj slici:

## Kratak opis slučajeva korišćenja

Napomena: Sve privilegije koje ima korisnik ima i admin, tako da svugde gde je korisnik pomenut individualno i administratori imaju tu privilegiju. U slučaju da administrator ima više mogućnosti, biće pomenut zajedno sa korisnikom.

### Registracija

Kratak opis Registracija korisnika na portal Fitnessivity u cilju stvaranja novog naloga i pristupa funkcijama koje zahtevaju autorizaciju.

Akteri koji iniciraju slučaj korišćenja: Posetioci.

### Pregled osnovnih informacija

Kratak opis:Pregled osnovnih informacija o korisniku, uključujući informacije o profilu, statistike aktivnosti i ostalim podacima.

Akteri koji iniciraju slučaj korišćenja: Korisnik, Administrator.

### Kreiranje i čuvanje programa za vežbanje

Kratak opis: Kreiranje i čuvanje individualnog programa vežbanja korisnika, koji se sastoji od vežbi prilagođenih njegovim ciljevima i sposobnostima.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Prikaz statistike i praćenje napretka

Kratak opis: Korisnicima se pruža mogućnost pregleda statistike o njihovim aktivnostima i praćenja napretka u vežbanju.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Dodavanje prijatelja

Kratak opis: Korisnik želi da doda prijatelja na platformi kako bi mogli da prate i podele svoj napredak u vežbanju.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Pregled programa po tipu

Kratak opis: Korisnik želi da pregleda sve programe vežbanja koji su dostupni na platformi Fitnessivity po određenom tipu vežbanja.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Prijava programa za neprikladan sadržaj

Kratak opis: Omogućavanje korisnicima da prijave programe za vežbanje sa neprikladnim sadržajem kako bi se očuvala integritet platforme i sprečilo širenje neprimerenog sadržaja.

Akteri koji iniciraju slučaj korišćenja: Korisnik, Administrator.

### Pregled prijavljenih programa za neprikladan sadržaj

Kratak opis: Administrator ima mogućnost da pregleda sve programe za vežbanje koji su prijavljeni kao neprikladni od strane korisnika, kako bi procenio da li sadrže neprikladan sadržaj ili ne.

Akteri koji iniciraju slučaj korišćenja: Administrator.

### Pregled i unos recenzija

Kratak opis: Korisnik ima mogućnost pregleda i unos recenzija za programe za vežbanje koje je prethodno koristio.

Akteri koji iniciraju slučaj korišćenja: Administrator.

### Prijava

Kratak opis: Prijavljivanje korisnika na stranicu Fitnessivity u cilju pristupa specifičnim funkcijama koje zahtevaju autorizaciju.

Akteri koji iniciraju slučaj korišćenja: Korisnik, Administrator.

### Ažuriranje podataka o korisniku

Kratak opis: Ažuriranje podataka o sebi od strane prijavljenog korisnika.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Brisanje programa

Kratak opis: Korisnik briše sopstveni program za vežbanje.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Brisanje prijatelja

Kratak opis: Korisnik želi da obriše korisnika iz listu prijatelja.

Akteri koji iniciraju slučaj korišćenja: Korisnik.

### Prikaz reklama

Kratak opis: Sistem prikazuje reklame korisnicima na različitim mestima na portalu Fitnessivity.

Akteri koji iniciraju slučaj korišćenja: Administrator, Oglašivač.

### Komunikacija između korisnika

Kratak opis: Slanje poruka između registrovanih korisnika na platformi.

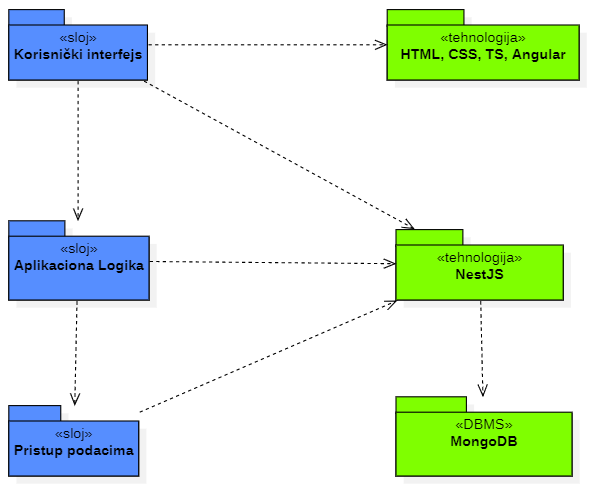
Akteri koji iniciraju slučaj korišćenja: Korisnik.

# 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.

Logička arhitektura Fitnessivity sistema obuhvata 3 glavna podsistema: Korisnički interfejs, aplikaciona logika, pristup podacima.

## Pregled arhitekture – organizacija paketa i podsistema u slojeve



**Korisnički interfejs:**

Korisnički interfejs predstavlja ono što korisnik vidi i interaktuje s aplikacijom. Ovo je frontend deo aplikacije koji se sastoji od HTML, CSS i TypeScript koda. Angular se koristi za razvoj korisničkog interfejsa, što olakšava stvaranje interaktivnih i dinamičnih web aplikacija.

**Aplikaciona logika:**

Aplikaciona logika je backend deo aplikacije koji se sastoji od nestjs koda. Ovaj deo aplikacije je odgovoran za obradu zahteva koje korisnik šalje preko korisničkog interfejsa i za komunikaciju s bazom podataka. Ovde se obavlja logika poput validacije unosa, autentifikacije korisnika, izračunavanja podataka, itd.

**Pristup podacima:**

Pristup podacima se ostvaruje pomoću MongoDB baze podataka. Ovo je deo aplikacije koji se bavi čuvanjem i dohvaćanjem podataka. Nestjs se koristi za interakciju s bazom podataka putem Mongoose ORM-a, što omogućava lako definisanje i manipulaciju sa podacima u bazi. Ovde se obavlja čitanje, upisivanje, brisanje i ažuriranje podataka u bazi.

**HTML (Hypertext Markup Language):**

Ovo je jezik koji se koristi za kreiranje struktura web stranica i definisanje elemenata na stranici kao što su naslovi, slike, paragrafi, tabele itd.

**CSS (Cascading Style Sheets):**

CSS se koristi za definisanje stilova na web stranicama, kao što su boje, fontovi, pozadine, margine, padding itd.

**TS (TypeScript):**

TypeScript je programski jezik koji se koristi za razvoj klijentske strane web aplikacija. TypeScript je nadskup JavaScript-a i obezbeđuje dodatne funkcije i sigurnost tipova koje JavaScript nema.

**Angular:**

Angular je popularni okvir za razvoj klijentske strane web aplikacija. Angular nudi mnoge funkcije kao što su modularnost, reaktivnost, enkapsulaciju komponenti i mnoge druge. Pomogoću angulara se mnogo lakše manipuliše HTML,CSS,TS tehnologijama ujedinjeno.

**NestJS:**

NestJS je framework za razvoj server-side aplikacija u TypeScriptu. Pruža modularnu arhitekturu i mnoge ugrađene funkcionalnosti za olakšavanje razvoja. Zapravo predstavlja progresivan NodeJS framework.

**MongoDB:**

MongoDB predstavlja sistem za upravljanje bazama podataka koji će se koristiti za realizaciju BoardBuddy aplikacije. MongoDB čuva podatke kao JSON (JavaScript Object Notation) dokumente sa dinamičkim šemama. Za razliku od tradicionalnih baza podataka, MongoDB koristi koncept takozvanih NoSQL baza podataka, koje ne koriste SQL za povezivanje, već koriste kolekcije i dokumente za skladištenje podataka.

# Pogled na procese

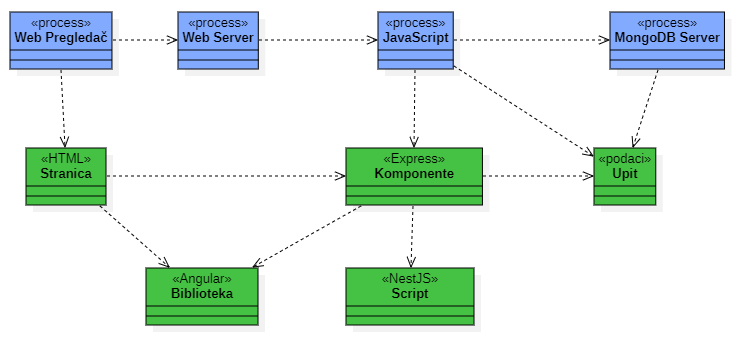
U ovom odeljku je sadržan pogled na procesnu arhitekturu sistema. Ovaj opis treba da sadrži specifikaciju različitih zadataka (procesa i niti) uključenih u rad sistema. Takođe je potrebno dati dijagrame koji pokazuju njihovu interakciju i konfiguraciju. Dodela objekata i klasa na određene zadatke takođe spada u opis procesne arhitekture.

Web platforme zasnovane na MongoDB-u, NestJS i Angularu imaju relativno jednostavan procesni model koji je u potpunosti pod kontrolom Web servera. Sa stanovišta projektanata Web platforme nije potrebno voditi računa o načinu rada Web servera i načinu izvršavanja skripti.

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

## Procesi

Na sledećem UML dijagramu klasa prikazani su procesi koji učestvuju u izvršenju Fitnessivity sistema.



### Web Pregledač

Web pregledač 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. Po uništavanju procesa Web čitač, uništava se i prikazana HTML stranica.

### Web server

Web server je proces koji izvršava funkcionalnost opsluživanja zahteva prispelih sa više Web čitača. Ukoliko je zahtevana stranica JavaScript, Web server inicira izvršenje RxJS procesa koji obrađuju pristligli upit i generiše sadržaj koji se vraća Web čitaču. Web server može paralelno da inicira veći broj JavaScript procesa.

### JavaScript

Svaki typescript se prevodi u adekvatni JavaScript proces pre izvršenja. JavaScript proces se obavlja preko NestJS za backend i Angular u klijentskoj strani. Za izvršenje JavaScript skripti NestJS proces može da zahteva usluge MongoDB servera-a. Komunikacija između JS procesa i MongoDB servera se obavlja preko prosleđivanja upita i vraćanja rezultata.

### MongoDB server

Serverska aplikacija koja poseduje glavnu nit *Main* zaduženu za podizanje GUI-a implementiranog klasom *Widget*.

# Pogled na implementaciju sistema

Pogled na implementaciju prikazuje različite aspekte bitne za implementaciju sistema. U slučaju Fitnessivity-a ovaj odeljak sadrži model domena i šemu baze podataka.

## Model domena

Model domena za koji se aplikacija Fitnessivity 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.

slika

## Šema baze podataka

Na slici ispod je prikazan UML klasni dijagram kojim je predstavljena šema NoSql baze podataka aplikacije Fitnessivity. Baza podataka je kreirana korišćenjem MongoDB Compass alata, a dijagram je kreiran korišćenjem StarUML-a.

slika

# 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 komponenti.

## Klijent

Klijentski računari su zaduženi za izvršenje Fitnessivity aplikacije. Za povezivanje između klijenta i servera koristi se Internet infrastruktura tako da nema ograničenja u pogledu lokacije klijenta.

## Server

Računar na kome se izvršava Fitnessivity server opslužuje više klijenata koji pristupaju preko Interneta.

# Performanse

Izabrana arhitektura softvera podržava zahteve u pogledu broja korisnika koji mogu simultano pristupati sistemu specificirane u zahtevima u pogledu performansi [5]:

1. Sistem će da podrži do 100 simultanih pristupa korisnika portalu.

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

# Kvalitet

Izabrana arhitektura softvera podržava zahteve u pogledu dostupnosti i srednjeg vremena između otkaza specificirane u zahtevima u pogledu pouzdanosti :

1. Fitnessivity sistem će biti dostupan 24 časa dnevno, 7 dana u nedelji. Vreme kada portal nije dostupan ne sme da pređe 30%.
2. Srednje vreme između dva sukcesivna otkaza ne sme da padne ispod 96 sati.