

**HealthCare**

**Projektovanje i implementacija modela podataka**

Julije Kostov 15680

Strahinja Laktović 15691

HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

## Pregled izmena

Datum	Verzija	Opis	Autor
21.12.2018.	1.0	Inicijalna verzija	JS

HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

## Sadržaj

1. Opis dokumenta	4
2. Šema baze podataka	4
2.1 Tabela user	5
2.2 Tabela measurements	6
2.3 Tabela examinations	6
2.4 Tabela prescriptions	7
3. Workflow data-access i business layer-a	8

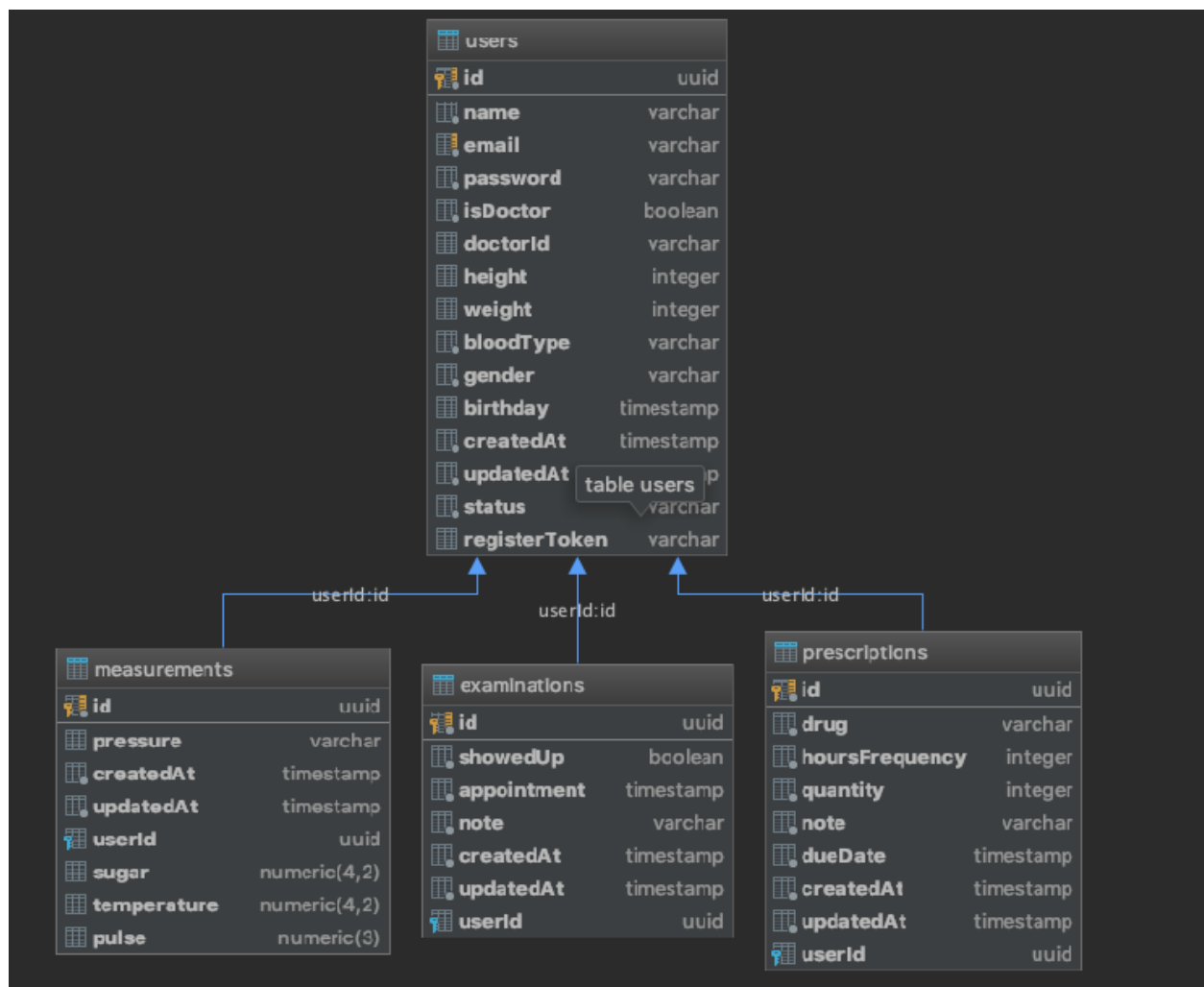
HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

## Opis dokumenta

Svrha pisanja ovog dokumenta jeste opis modela podataka i modela perzistencije koji se koristi u aplikaciji Healthcare. Za model perzistencije korišćena je PostgreSQL baza podataka. Korišćen je i objektno- relacioni mapper typeORM da bi se programeru omogućilo da lakše komunicira s bazom.

## Šema baze podataka

Aplikacija Healthcare sadrži sistem za autentifikaciju korisnika. Nakon logovanja ili registrovanja na sistem, korisnik sa ulogom doktor može kreirati i menjati postojeće preglede, recepte, kao i menjati izabranog lekara pacijenata, odnosno postavljanje sebe. Pacijenti mogu da dodaju nova merenja raznih parametara.

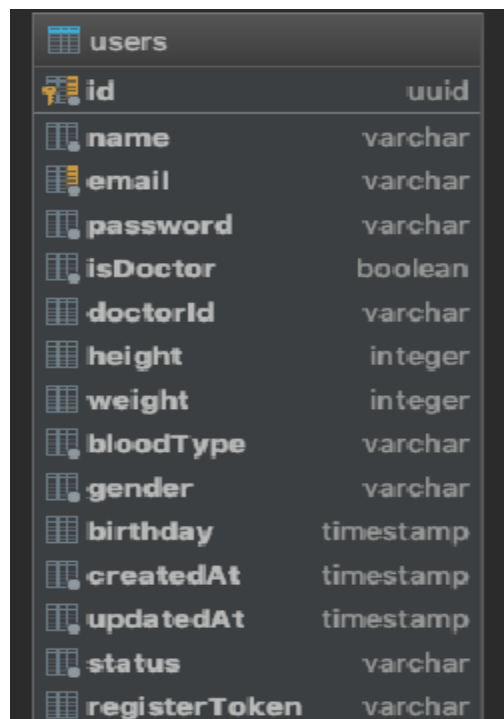


HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

## Tabela user

Za korisnika se u bazi pamti sledeće:

- id - tipa uuid,
- name - kao podatak tipa varchar,
- email - tipa varchar i mora biti unikatan,
- password - koji se pamti u enkriptovanom obliku kao varchar,
- isDoctor - tipa Boolean,
- doctorId - tipa varchar koji je nullable,
- height - tipa integer koji je nullable,
- weight - tipa integer koji je nullable,
- bloodType - tipa varchar,
- gender - tipa varchar,
- birthday - tipa timestamp koji je nullable,
- status - tipa varchar,
- registerToken - tipa varchar koji je nullable,
- createdAt - tipa timestamp,
- updatedAt - tipa timestamp,



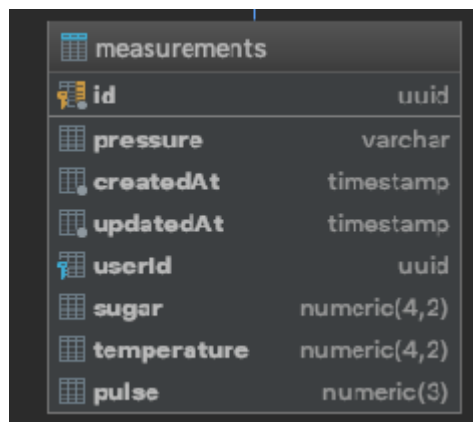
users	
id	uuid
name	varchar
email	varchar
password	varchar
isDoctor	boolean
doctorId	varchar
height	integer
weight	integer
bloodType	varchar
gender	varchar
birthday	timestamp
createdAt	timestamp
updatedAt	timestamp
status	varchar
registerToken	varchar

HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

## Tabela Measurements

Za merenja se u bazi pamti sledeće:

- id - tipa uuid,
- pressure - kao podatak tipa varchar koji je nullable,
- userId - tipa uuid koji je strani ključ ka useru.
- sugar - tipa integer koji je nullable,
- pulse - tipa integer koji je nullable,
- temperature - tipa integer koji je nullable,
- createdAt - tipa timestamp,
- updatedAt - tipa timestamp.



measurements	
id	uuid
pressure	varchar
createdAt	timestamp
updatedAt	timestamp
userId	uuid
sugar	numeric(4,2)
temperature	numeric(4,2)
pulse	numeric(3)

## Tabela Examinations

Za pregleda se u bazi pamti sledeće:

- id - tipa uuid ,
- userId - tipa uuid koji je strani ključ ka useru,
- showedUp - tipa Boolean,
- appointment - tipa timestamp,
- note - tipa varchar,
- createdAt - tipa timestamp,
- updatedAt - tipa timestamp.

HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

examinations	
id	uuid
showedUp	boolean
appointment	timestamp
note	varchar
createdAt	timestamp
updatedAt	timestamp
userId	uuid

## Tabela Prescriptions

Za recepte se u bazi pamti sledeće:

- id - tipa uuid,
- userId - tipa uuid koji je strani ključ ka useru,
- drug - tipa varchar,
- hoursFrequency - tipa integer,
- quantity - tipa integer,
- note - tipa varchar,
- dueDate - tipa timestamp,
- createdAt - tipa timestamp,
- updatedAt - tipa timestamp.

prescriptions	
id	uuid
drug	varchar
hoursFrequency	integer
quantity	integer
note	varchar
dueDate	timestamp
createdAt	timestamp
updatedAt	timestamp
userId	uuid

HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

## Workflow Data-Access i Business layer-a

U svakom modulu postoji folder dto gde se nalaze POTO (plain old typescript object) klase u kojima je dekoratorima naznačena validacija atributa, dok se preko imena kasnije mapiraju u odgovarajuće attribute entity klase.

```
import { IsNumber, IsString } from 'class-validator';

export class CreatePrescriptionDto {

    @IsString()
    readonly drug: string;

    @IsNumber()
    readonly hoursFrequency: number;

    @IsNumber()
    readonly quantity: number;

    @IsString()
    readonly note: string;

    @IsString()
    readonly dueDate: string;

    @IsString()
    readonly userId: string;
}
```

U entity folderima nalazi se entitet klasa koje predstavljaju tabele u bazi.

```
@Entity({
  name: 'prescriptions',
})
export class Prescription {
  @IsUUID('4')
  @PrimaryGeneratedColumn('uuid')
  id: string;

  @Column()
  drug: string;

  @Column()
  hoursFrequency: number;

  @Column()
  quantity: number;

  @Column()
  note: string;

  @Column()
  dueDate: Date;

  @ManyToOne(type => User, user => user.prescriptions)
  user: User;

  @CreateDateColumn({ type: 'timestamp' })
  createdAt: Date;

  @UpdateDateColumn({ type: 'timestamp' })
  updatedAt: Date;
}
```



HealthCare	Verzija: 1.0
Projektovanje i implementacija modela podataka	Datum: 21.12.2018. godine

Objekti se mogu sačuvati na tri načina:

1. Prosledjivanjem dto objekta as entity objekat gde je tip podataka isti za sve attribute obeju klasa. Ovim se data mapiranjem direktno dto čuva kao objekat entity klase.
2. Kreiranjem Partial<ImeEntiteta> klase gde se odredjeni atributi operatorom spread automatski preslikaju iz dto objekta u Partial<ImeKlase> objekat, a drugi ručno npr. a) string u dto-u kao datetime u Partial klasi ili b) string u enum.
3. Kreiranjem Entity objekta prosledjivanjem dto-a konstruktoru i mapiranjem slično kao pod 2 Validni objekti kreirani na jedan ova 3 načina se prosledjuju kao parametar repository funkciji i bivaju sačuvani.

```

async create(createPrescriptionDto: CreatePrescriptionDto): Promise<Prescription | HttpException> {
    const { userId, ... prescriptionDto } = createPrescriptionDto;

    const user: User = await this.usersRepository.findOne(userId);
    if (!user) {
        return new HttpException( response: 'User not found.', HttpStatus.BAD_REQUEST);
    }

    const prescription: Partial<Prescription> = {
        ... prescriptionDto,
        user: null,
        dueDate: new Date(prescriptionDto.dueDate),
    };
    prescription.user = user;

    return await this.prescriptionsRepository.save(prescription as Prescription);
}

```

U kontrolerima koriste se interfejsi servisa i entitet klase. Oni su deo biznis sloja. Predstavljaju osnovu aplikacije na koju ne utiču presentation i persistence framework-ovi. Implementacije servisa pripadaju data access sloju.

```

@Controller('users')
export class UsersController {
    private readonly usersService: IUsersService;

    constructor(@Inject('UsersService') usersService: IUsersService) {
        this.usersService = usersService;
    }
}

```