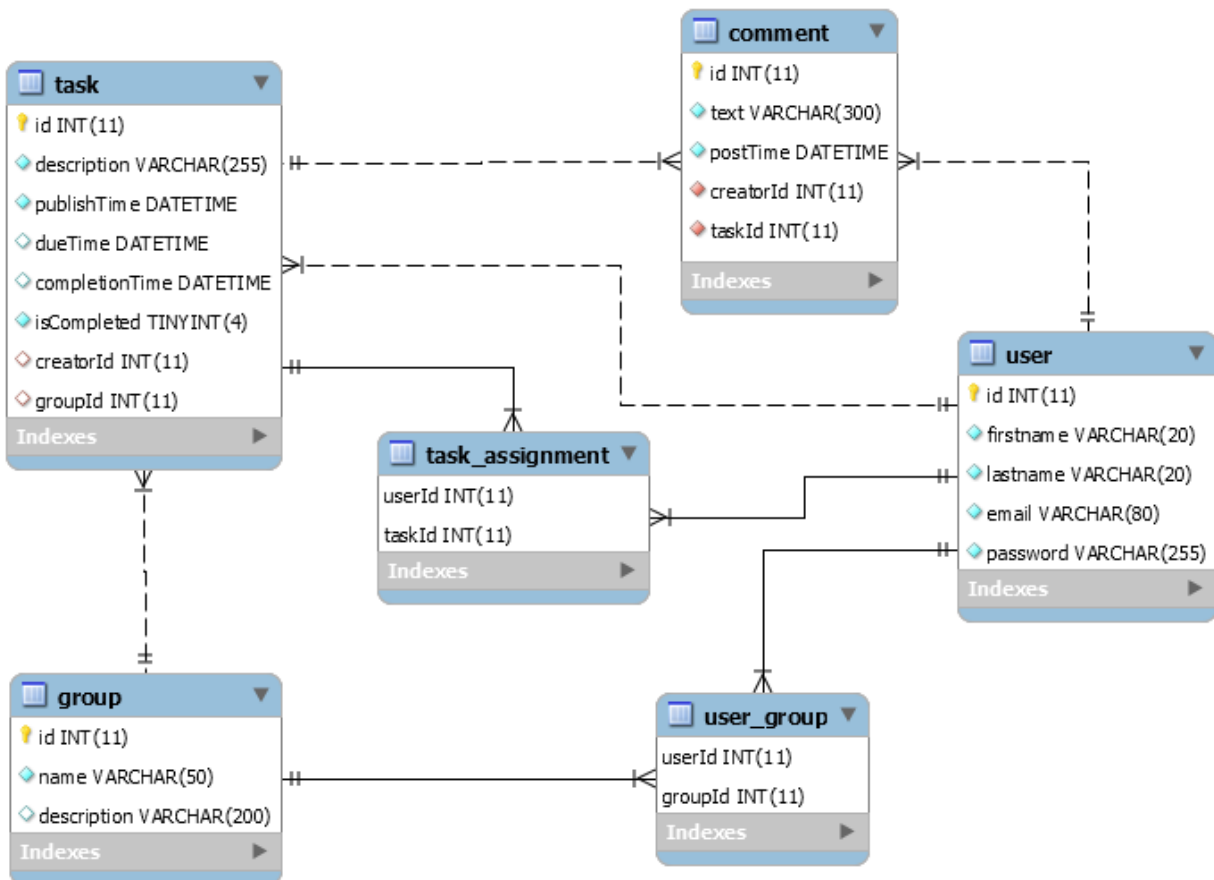


Time Cruncher

Business model I model perzistencije

Model podataka (business model):

Aplikacija Time cruncher koristi MySQL bazu za skladištenje podataka.



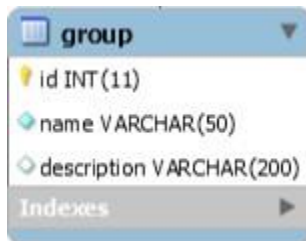
- Tabele u bazi podataka:

U tabeli *user* se pamte informacije o korisnicima aplikacije.



user	
id	INT(11)
first_name	VARCHAR(20)
last_name	VARCHAR(20)
email	VARCHAR(80)
password	VARCHAR(255)
Indexes	

U tabeli *group* se pamte informacije o grupama.



group	
id	INT(11)
name	VARCHAR(50)
description	VARCHAR(200)
Indexes	

U tabeli *task* se pamte informacije o zadacima. Kroz kolone *creatorId* i *groupId* se implementira više prema jedan (many to one) veza sa tabelama *user* i *group* respektivno.

- *creatorId* : identifikator korisnika (*user*) koji je kreirao zadatak (*task*)
- *groupId* : identifikator grupe (*group*) kojoj zadatak pripada.



task	
id	INT(11)
description	VARCHAR(255)
publishTime	DATETIME
dueTime	DATETIME
completionTime	DATETIME
isCompleted	TINYINT(4)
creatorId	INT(11)
groupId	INT(11)
Indexes	

Tabele *task_assignment* i *user_group* implementiraju više prema više (many to many) veze između entiteta.

- *task_assignment* : Veza između korisnika (*user*) i zadataka (*task*) koji su im dodeljeni.

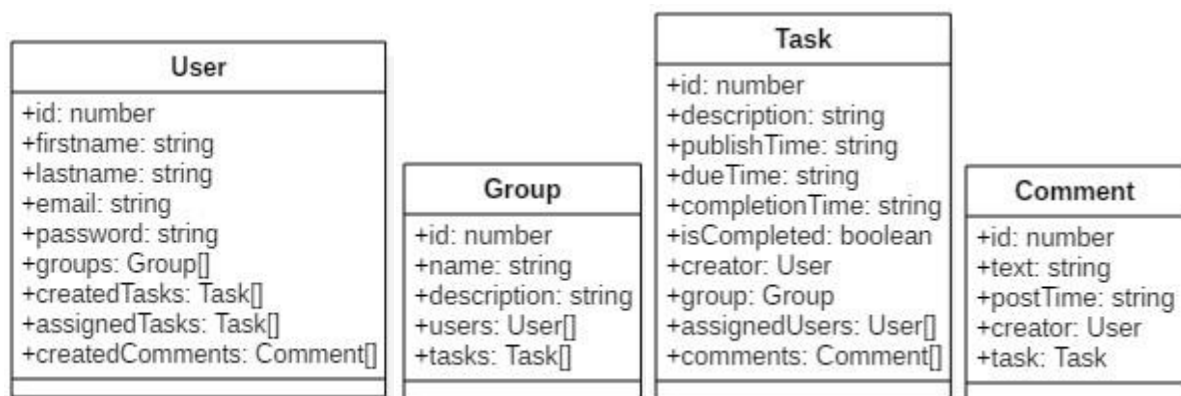
task_assignment
userId INT(11)
taskId INT(11)
Indexes

- *user_group*: Veza između korisnika (*user*) i grupa (*group*) kojima oni pripadaju.

user_group
userId INT(11)
groupId INT(11)
Indexes

Model prezistencije (Entity model):

Business model aplikacije Time Cruncher je definisan kroz 4 entiteta, predstavljenih klasama:



TypeORM framework u okviru NestJS frameworka obezbeđuje funkcionalnosti mapiranja relacionog modela u objektni model pomoću dekoratora kojima se definiše koja se kolona mapira na koji atribut klase entiteta.

```
@Entity()
export class Comment{
  @PrimaryGeneratedColumn()
  id: number;

  @Column({
    type: 'varchar',
    length: '300',
    nullable: false,
  })
  text: string;

  @CreateDateColumn({
    type: 'datetime',
  })
  postTime: string;

  @ManyToOne(type => User, user => user.createdComments, {eager: true, nullable: false, onDelete: 'CASCADE'})
  @JoinColumn()
  creator: User;

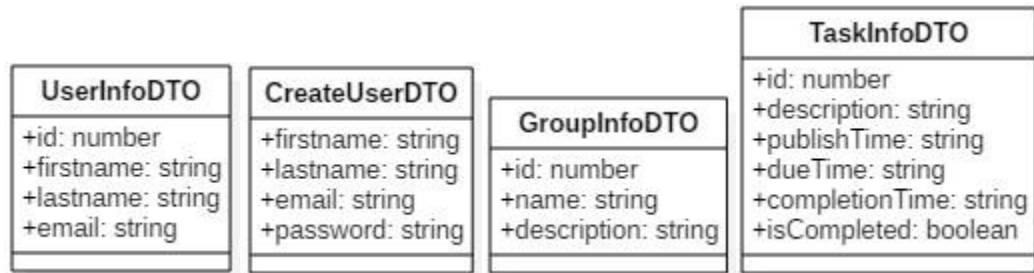
  @ManyToOne(type => Task, task => task.creator, {nullable: false, onDelete: 'CASCADE'})
  @JoinColumn()
  task: Task;
}
```

API:

- URI: “/users”

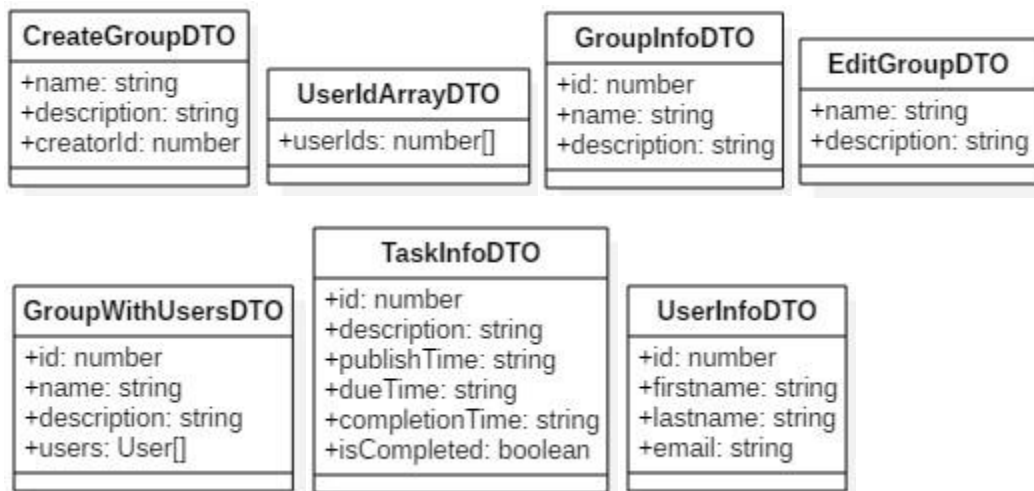
URI	Type	Properties	RequestBody	ResponseBody
-	GET	-	-	UserInfoDTO[]
-	POST	-	CreateUserDTO	UserInfoDTO
-	PUT	id	CreateUserDTO	UserInfoDTO
/:id	GET	id	-	UserInfoDTO
/:id	DELETE	id	-	-
/:id/assignedTasks	GET	id	-	TaskInfoDTO[]
/:id/createdTasks	GET	id	-	TaskInfoDTO[]
/:id/groups	GET	id	-	GroupInfoDTO[]

/:id/daily/:dateString	GET	Id, dateString	-	TaskInfoDTO[]
/:id/monthly/:dateString	GET	Id, dateString	-	TaskInfoDTO[]



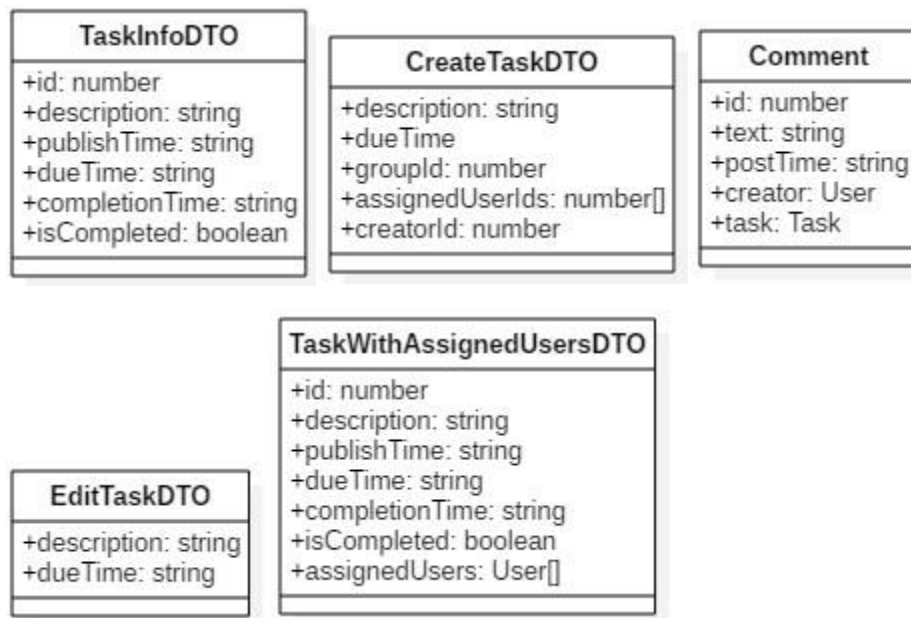
- URI: “/groups”

URI	Type	Properties	RequestBody	ResponseBodyData
-	GET	-	-	GroupInfoDTO[]
-	POST	-	CreateGroupDTO	GroupInfoDTO
/:id	GET	id	-	GroupInfoDTO
/:id	DELETE	id	-	-
/:id	PUT	id	EditGroupDTO	GroupInfoDTO
/:id/addUsers	PUT	id	UserIdArrayDTO	GroupWithUsersDTO
/:id/tasks	GET	id	-	TaskInfoDTO
/:id/users	GET	id	-	UserInfoDTO



- URI: "/tasks"

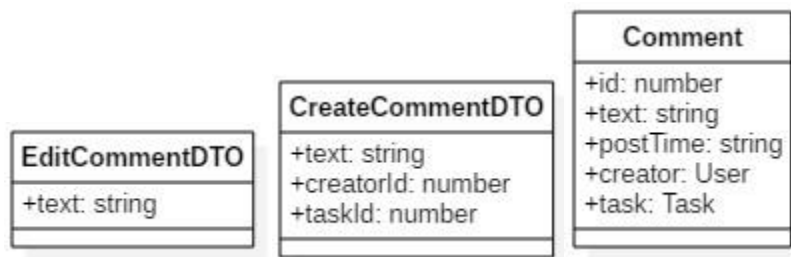
URI	Type	Properties	RequestBody	ResponseBodyData
-	GET	-	-	TaskInfoDTO[]
-	POST	-	CreateTaskDTO	TaskInfoDTO
/:id	GET	id	-	TaskInfoDTO
/:id	DELETE	id	-	-
/:id	PUT	id	EditTaskDTO	TaskInfoDTO
/:id/comments	GET	id	-	Comment[]
/:id/markCompleted	PUT	id	-	TaskInfoDTO
/:id/resetCompleted	PUT	id	-	TaskInfoDTO
/:id/assignUsers	PUT	Id	UserIdArrayDTO	TaskWithAssignedUsersDTO



- URI: "/comments"

URI	Type	Properties	RequestBody	ResponseBodyData
-	GET	-	-	Comment[]

-	POST	-	CreateCommentDTO	Comment
/:id	GET	id	-	Comment
/:id	DELETE	id	-	-
/:id	PUT	id	EditCommentDTO	Comment



Format odgovora na zahteve:

- Ukoliko je zahtev uspešan, telo odgovora će sadržati tražene podatke, ili poruku u slučaju DELETE zahteva, a u zaglavlju odgovora će se nalaziti odgovarajući statusni kod odgovora. U slučaju neuspešnog zahteva, telo odgovora će sadržati sledeći objekat:

```

{
  "message": <poruka>
}
  
```

, pri čemu će poruka u slučaju neuspešne validacije sadržati niz objekata sa atributima:

- property – naziv atributa klase gde je validacija utvrdila grešku
- value – vrednost u atributu gde je pronađena greška

, a u slučaju ostalih grešaka poruku tipa string.