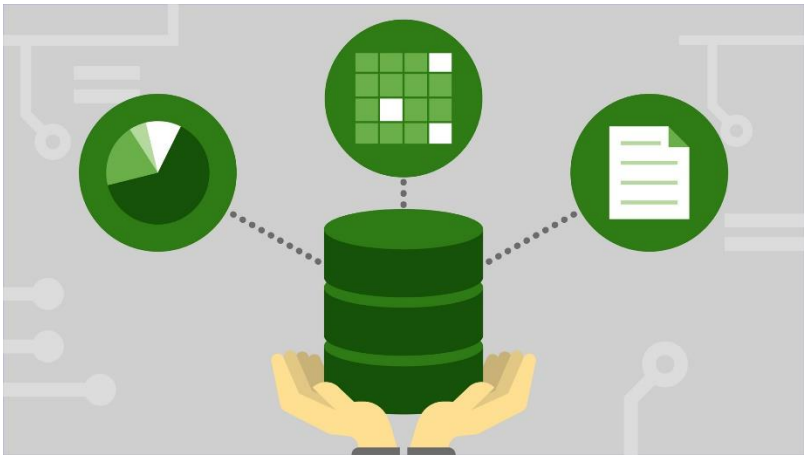
به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر



آزمایشگاه پایگاه‌داده

دستورکار شماره ۶

شماره دانشجویی

۸۱۰۱۹۵۵۲۶

آذر ۹۹

پارسا صدری سینکی

گزارش فعالیت‌های انجام شده

تمام کد های این پروژه در مخزن زیر موجود هست.

github.com/PSS1998/To-Do

بخش چهارم

در بخش چهارم ابتدا کلیت Authentication ساخته شد و سپس به jwt تغییر کرد که برای این مراحل لازم بود یوزر و پسورد به اسکیما دیتابیس یوزر اضافه شود.

دو تابع اصلی این مرحله مربوط به فایل jwt.strategy.ts است.

```
  constructor() {
    super({
      jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
      ignoreExpiration: false,
      secretOrKey: jwtConstants.secret,
    });
  }

  async validate(payload: any) {
    return { userId: payload.sub, username: payload.username };
  }
```

تابع super قسمت secret مهم می باشد که نباید در محیط production لو برود.

قسمت دوم مهم این بخش اضافه کردن @UseGuards(AuthGuard()) به ابتدا تمامی تابع ها به جز لاگین و ساخت یوزر برای این که حتما لاگین کرده باشند بود و توکن jwt را همراه درخواست بفرستند.

بخش پنجم

این قسمت اصلی پروژه بود که در آن ابتدا لازم بود entity های برنامه ToDo را درست کنیم و سپس سرویس و کنترل آن را بسازیم.

برای طراحی دیتابیس مان یک جدول task درست کردیم که یک فیلد title دارد. یک جدول item درست کردم که یک فیلد description دارد و یک جدول tag که فیلد name دارد.

از جدول user به task رابطه یک به چند وجود دارد. از جدول task به دو جدول دیگر نیز رابطه یک به چند وجود دارد.

برای هر سطر task یوزر نویسنده آن را در فیلد author نگه داشتم. برای tag و item هم به همین صورت owner آن ها را در هنگام ساخت نگه می دارم.

```
@Entity()
export default class TaskEntity extends BaseEntity {

  @PrimaryGeneratedColumn()
  id: number;
```

```
  @Column({ length: 500 })
  title: string;

  // n:1 relation with UserEntity
  @ManyToOne( type => UserEntity , user => user.id)
  author: UserEntity;

  // 1:n relation with TagEntity
  @OneToMany( type => TagEntity , tag => tag.id)
  tags: TagEntity[];

  // 1:n relation with ItemEntity
  @OneToMany( type => ItemEntity , item => item.id)
  items: ItemEntity[];

}
```

```
@Entity()
export default class ItemEntity extends BaseEntity {

  @PrimaryGeneratedColumn()
  id: number;

  @Column()
  description: string;

  // n:1 relation with TaskEntity
  @ManyToOne( type => TaskEntity , task => task.id, { onDelete: "CASCADE" }
)
  task: TaskEntity;

}
```

```
@Entity()
export default class TagEntity extends BaseEntity {

  @PrimaryGeneratedColumn()
  id: number;

  @Column({ length: 500 })
  name: string;

  // n:1 relation with TaskEntity
```

```
  @ManyToOne( type => TaskEntity , task => task.id, { onDelete: "CASCADE" }
)
  task: TaskEntity;

}
```

در مرحله بعد سرویس های ToDo را درست کردم.

که برای هر سه Task، Item و Tag هر کدام یک فانکشن POST، GET، DELETE و PUT را در فایل todo.service.ts نوشتم.

برای دستورات GET در Task فقط موارد مربوط به یوزر لاگین شده را بر می گردانم و برای Item و Tag هم موارد مربوط به Task ای که مشخص شده را فقط بر می گردانم.

نمونه کوئری Task را در خط زیر مشاهده می کنید.

```
const user: UserEntity = await UserEntity.findOne({where: {id: userID}});
```

در مرحله بعد برای کنترلر ها هم برای هر سرویس دقیقا یک کنترل معادل نوشتم.

برای ساخت تسک جدید و ثبت یوزر سازنده آن از توکن jwt فرستاده شده یوزر استفاده کردم که در خط زیر مشاهده می کنید.

```
    const jwtToken: string = request.headers.authorization;
    const jwtDecoded: any = jwt.decode(jwtToken.split(" ")[1]);
    const user: UserEntity = await UserEntity.findOne({where: {username: jw
tDecoded.username}});
```

برای GET کردن تسک ها به همین روش فقط تسک های مربوط به همان یوزر را بر میگرداند کد.

ID های GET ها را توسط متغیر در URL می فرستم به کنترلر ولی برای DELETE و UPDATE از روش :id در url دریافتی استفاده می کنم که این دو مورد را در خطوط بعدی می توانید مشاهده کنید.

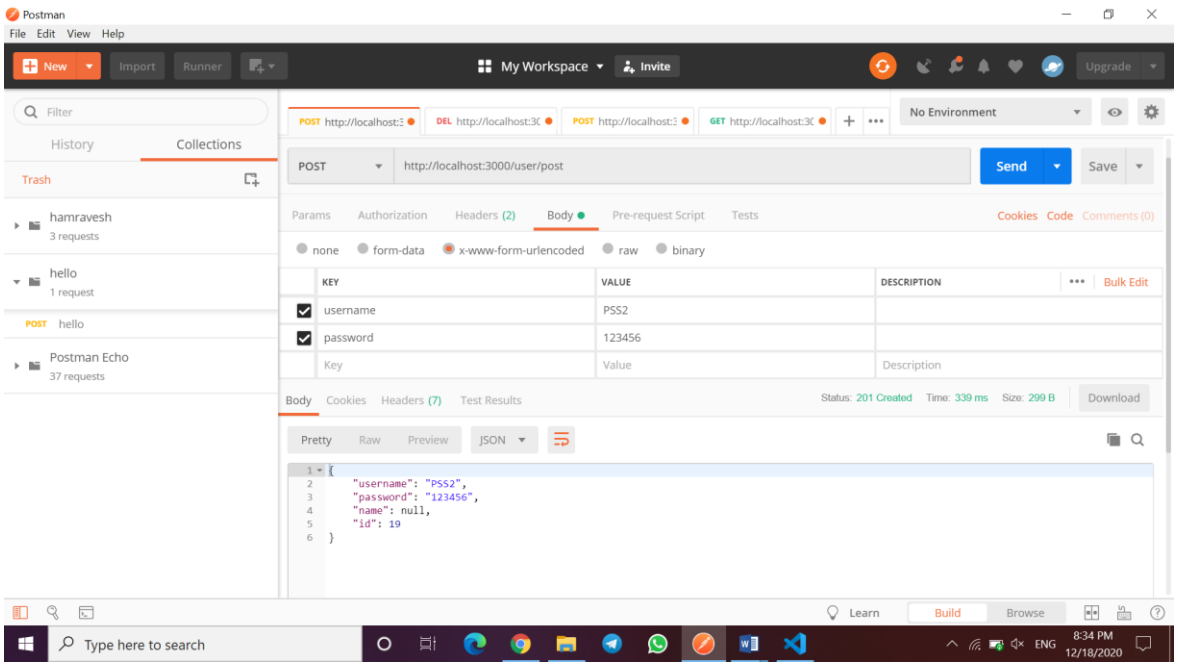
```
  @Get('tag')
  async getTag( @Body('taskID', ParseIntPipe) taskID: number) {
    return this.todoService.getAllTags(taskID);
  }
```

```
  @Delete('task/delete/:id')
  async deleteTask( @Param('id') id: string) {
    return this.todoService.deleteTask(+id);
  }
```

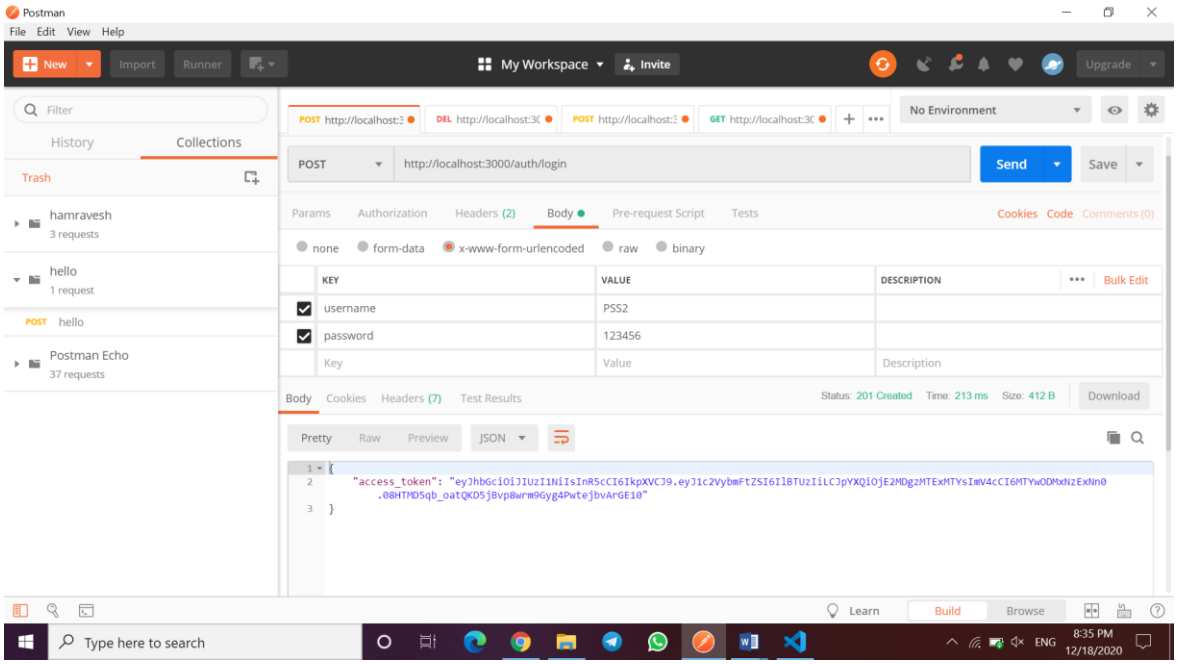
در مورد DTO ها برای هر کدام یک مورد update و create ساخته ام برای این که در صورت لزوم محتوی آن بتواند متفاوت باشد ولی در کد من هر دو مشابه هستند.

در انتها تصاویر عمکرد درست برنامه را قرار می دهم.

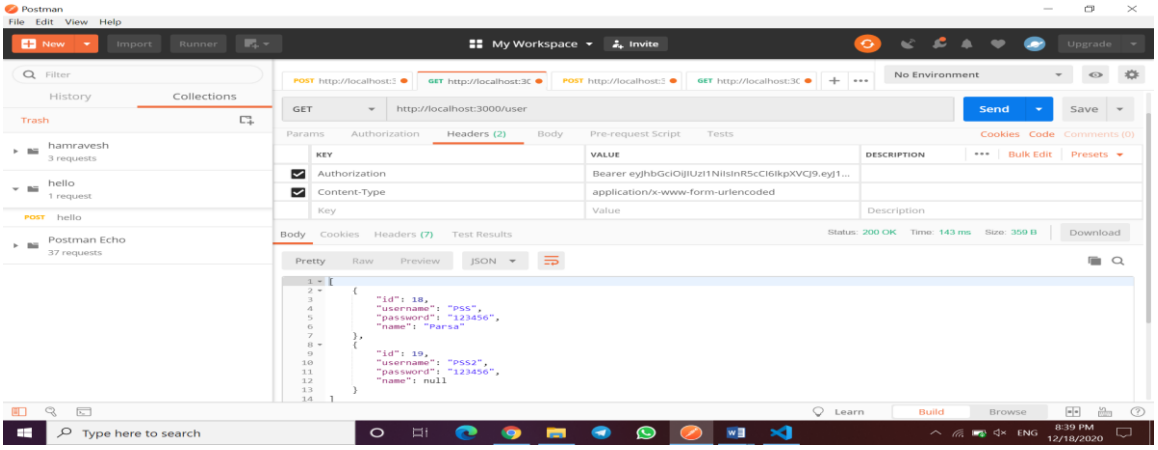
ساخت یوزر:



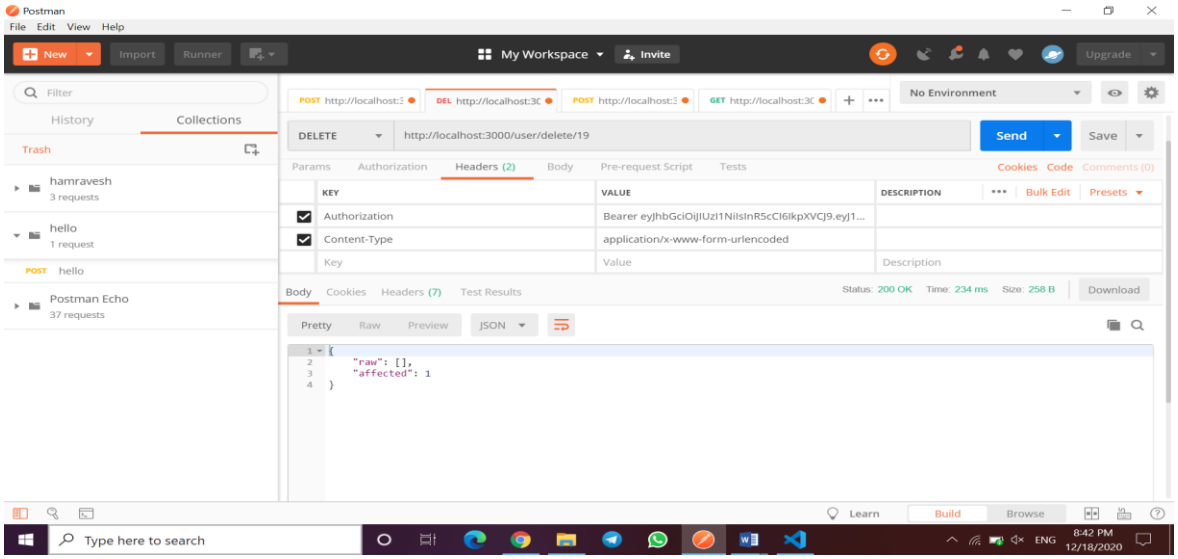
لاگین:

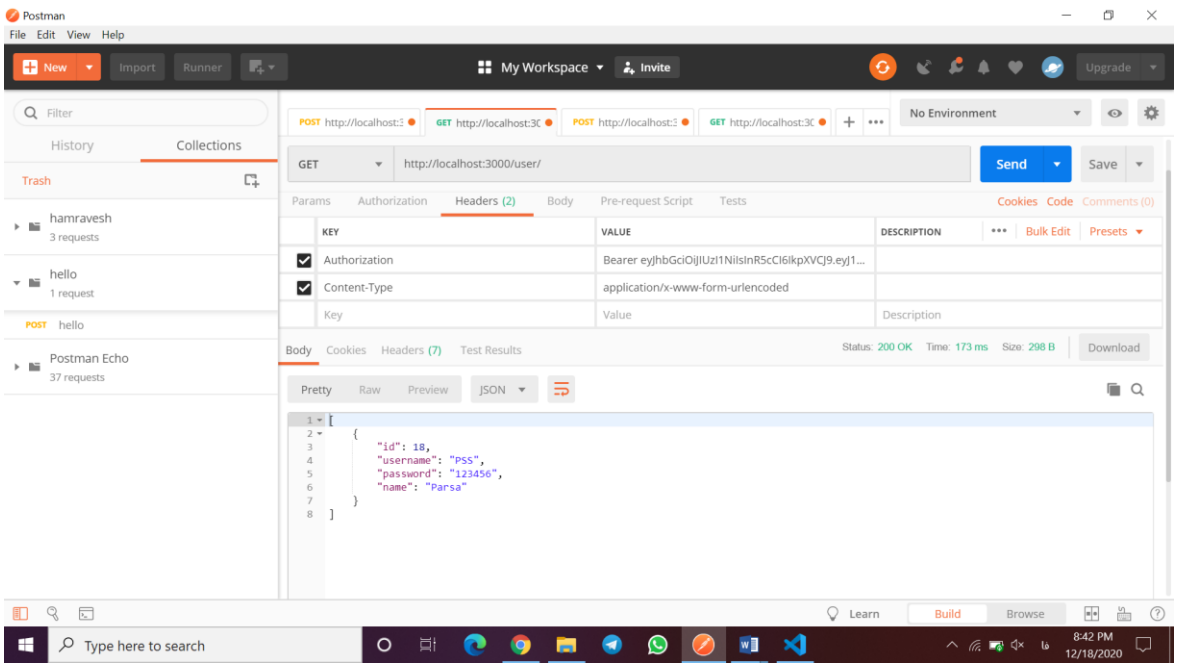


دریافت یوزر ها:

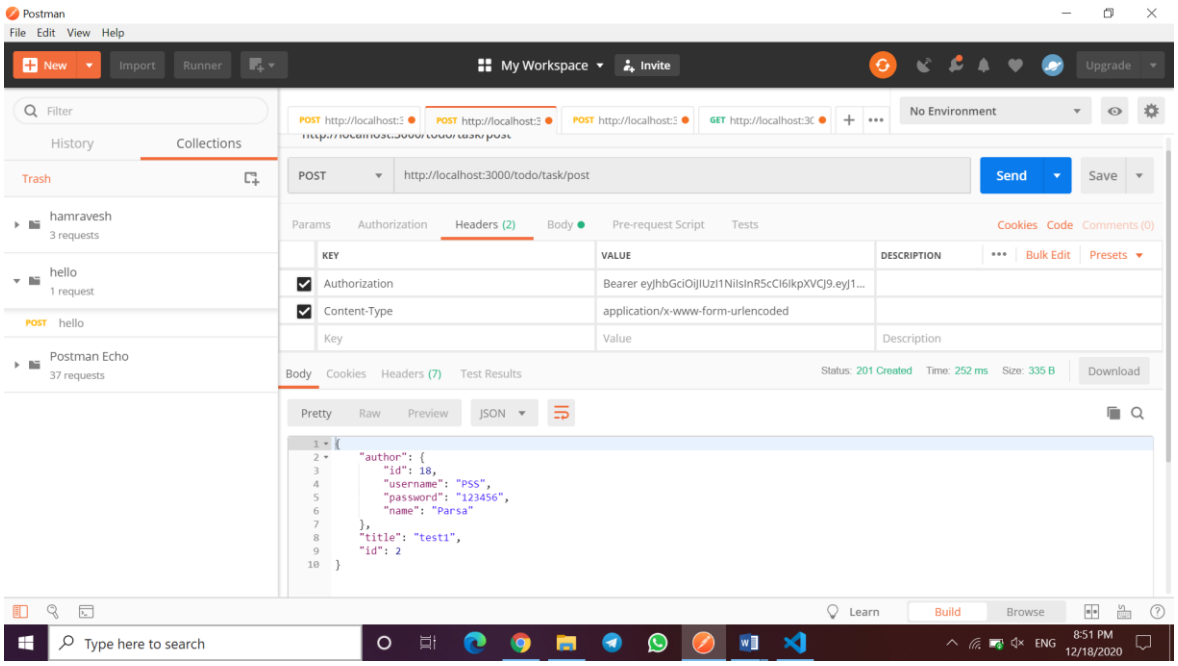

Postman
File Edit View Help
New Import Runner
My Workspace Invite
Upgrade
No Environment
GET http://localhost:3000/user
Send Save
Params Authorization Headers (2) Body Pre-request Script Tests
KEY VALUE DESCRIPTION
Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...
Content-Type application/x-www-form-urlencoded
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 143 ms Size: 359 B Download
Pretty Raw Preview JSON
[{ "id": 18, "username": "PSS", "password": "123456", "name": "Parsa" }, { "id": 19, "username": "PSS2", "password": "123456", "name": null }]
History Collections
Trash
hamravesh 3 requests
hello 1 request
POST hello
Postman Echo 37 requests
Learn Build Browse
Type here to search
ENG 8:39 PM 12/18/2020

حذف یوزر:

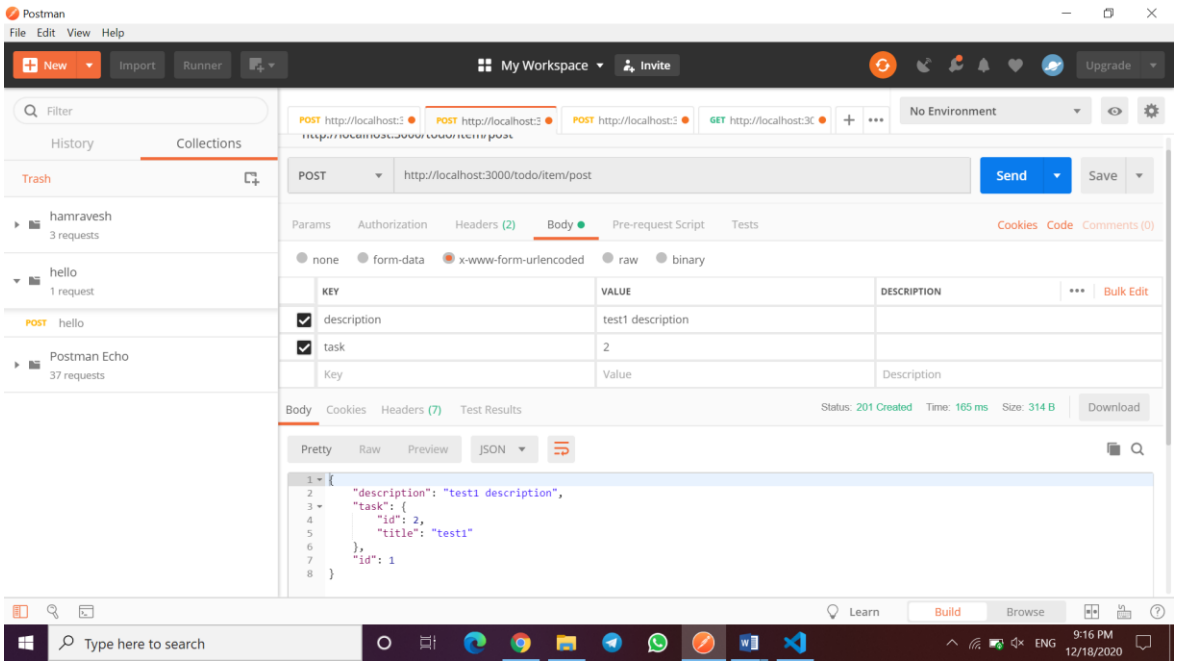

Postman
File Edit View Help
New Import Runner
My Workspace Invite
Upgrade
No Environment
DELETE http://localhost:3000/user/delete/19
Send Save
Params Authorization Headers (2) Body Pre-request Script Tests
KEY VALUE DESCRIPTION
Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...
Content-Type application/x-www-form-urlencoded
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 234 ms Size: 258 B Download
Pretty Raw Preview JSON
{ "raw": [], "affected": 1 }
History Collections
Trash
hamravesh 3 requests
hello 1 request
POST hello
Postman Echo 37 requests
Learn Build Browse
Type here to search
ENG 8:42 PM 12/18/2020


Postman
File Edit View Help
New Import Runner
My Workspace Invite
Upgrade
No Environment
GET http://localhost:3000/user/
Send Save
Params Authorization Headers (2) Body Pre-request Script Tests
KEY VALUE DESCRIPTION
Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...
Content-Type application/x-www-form-urlencoded
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 173 ms Size: 298 B Download
Pretty Raw Preview JSON
[{ "id": 18, "username": "PSS", "password": "123456", "name": "Parsa" }]
History Collections
Trash
hamravesh 3 requests
hello 1 request
POST hello
Postman Echo 37 requests
Learn Build Browse
Type here to search
8:42 PM 12/18/2020

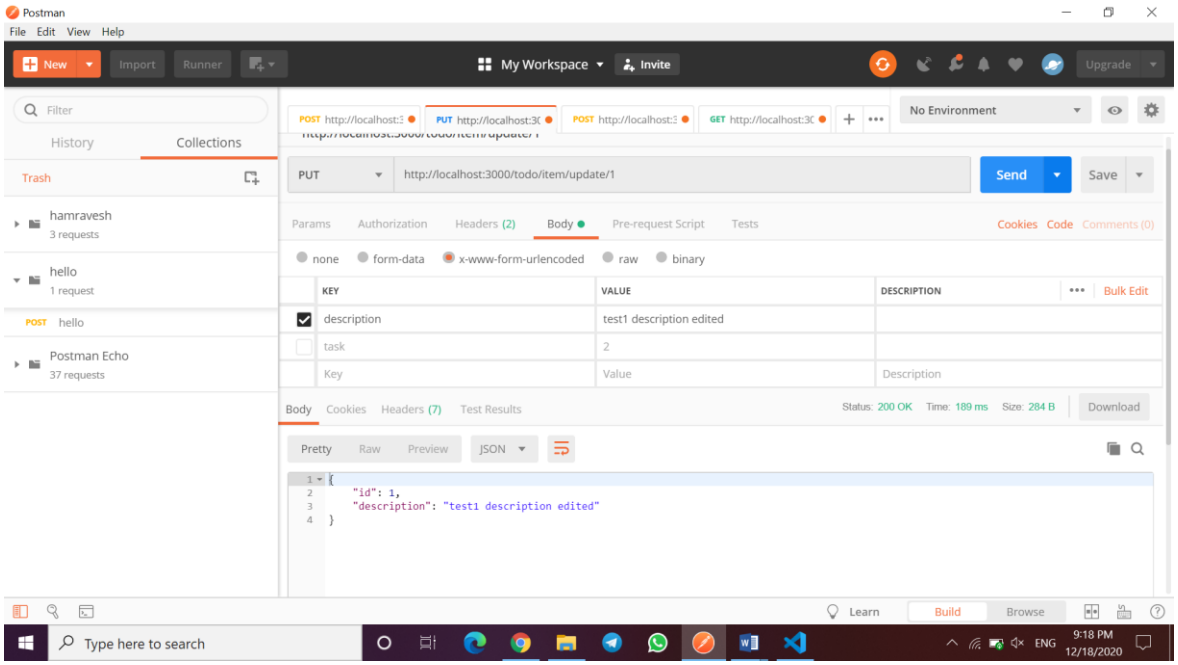
ساخت Task:



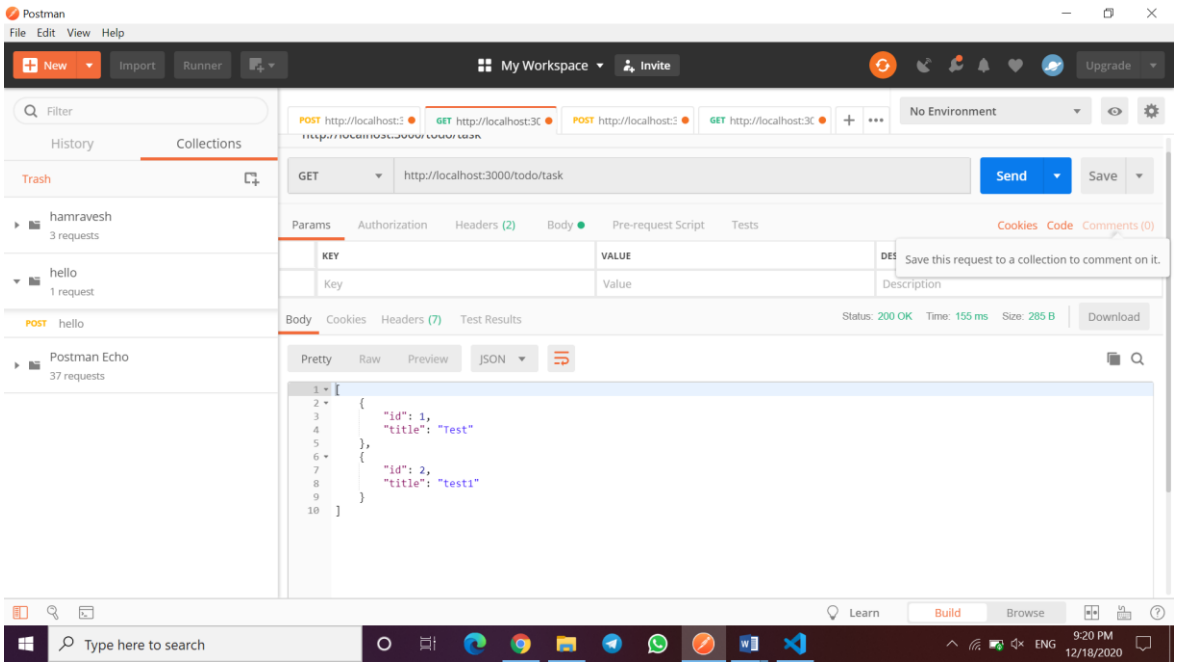
ساخت item برای تسک قبلی:



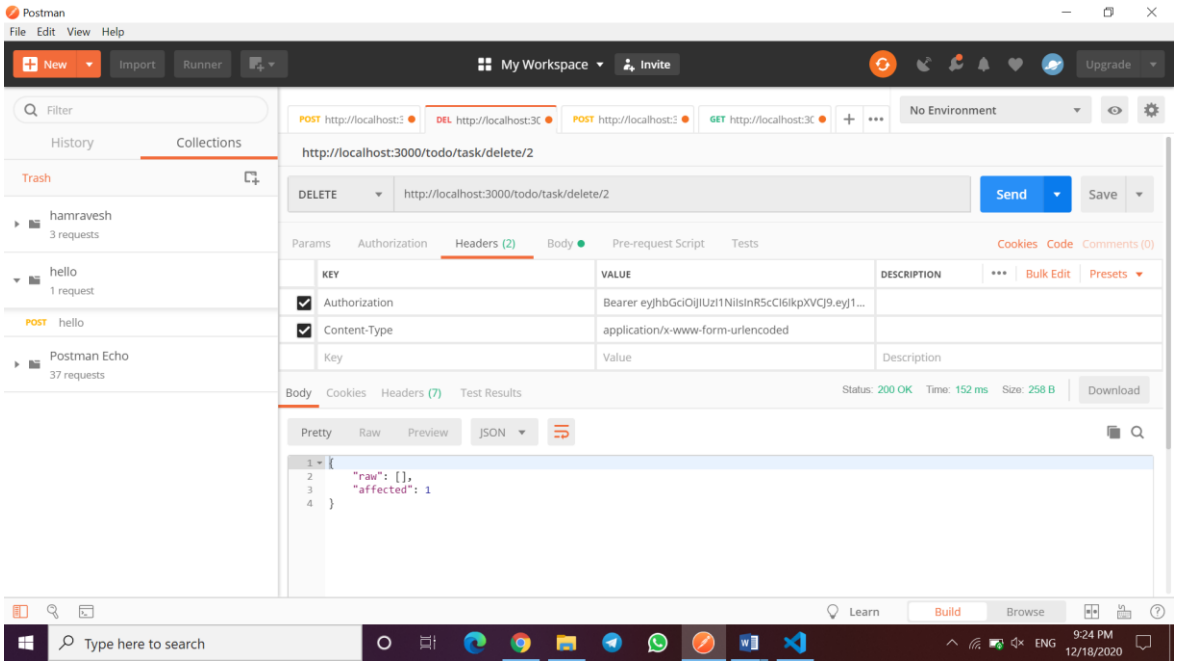
تغییر متن item قبلی:

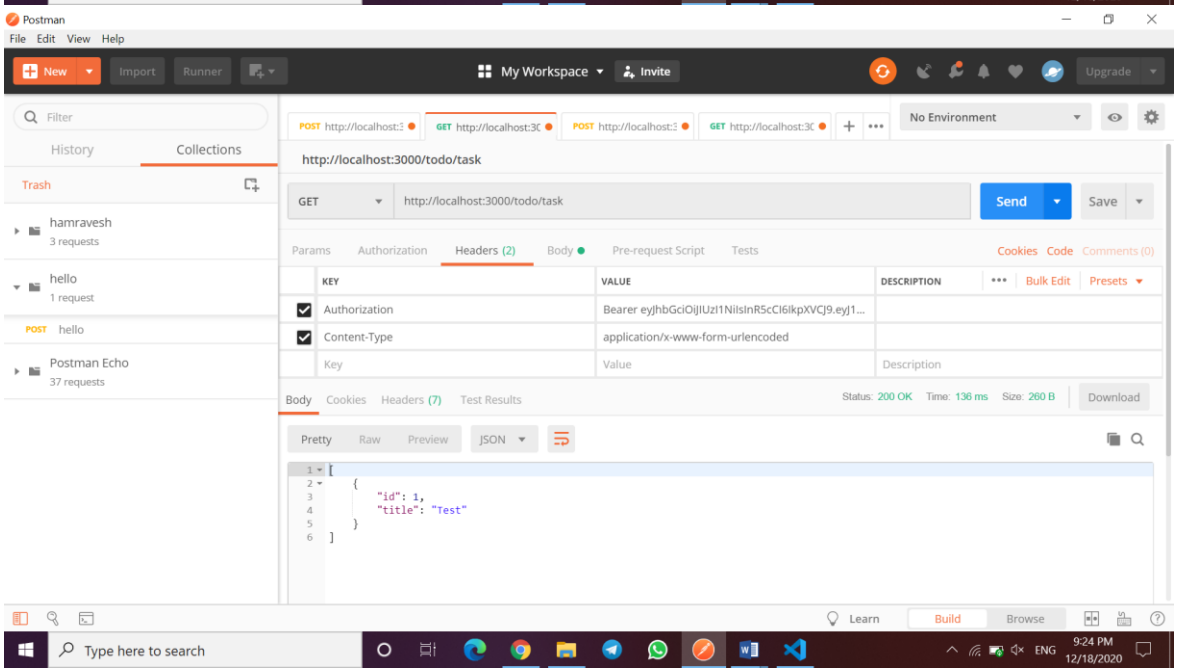


دریافت تمام Task ها:



حذف task:


Postman
File Edit View Help
New Import Runner
My Workspace Invite
Upgrade
No Environment
History Collections
Trash
hamravesh
3 requests
hello
1 request
POST hello
Postman Echo
37 requests
http://localhost:3000/todo/task/delete/2
DELETE http://localhost:3000/todo/task/delete/2
Send Save
Params Authorization Headers (2) Body Pre-request Script Tests
Cookies Code Comments (0)
KEY VALUE DESCRIPTION Bulk Edit Presets
Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...
Content-Type application/x-www-form-urlencoded
Key Value Description
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 152 ms Size: 258 B Download
Pretty Raw Preview JSON
{
"raw": [],
"affected": 1
}
Learn Build Browse
Type here to search
ENG 9:24 PM 12/18/2020


Postman
File Edit View Help
New Import Runner
My Workspace Invite
Upgrade
No Environment
History Collections
Trash
hamravesh
3 requests
hello
1 request
POST hello
Postman Echo
37 requests
http://localhost:3000/todo/task
GET http://localhost:3000/todo/task
Send Save
Params Authorization Headers (2) Body Pre-request Script Tests
Cookies Code Comments (0)
KEY VALUE DESCRIPTION Bulk Edit Presets
Authorization Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...
Content-Type application/x-www-form-urlencoded
Key Value Description
Body Cookies Headers (7) Test Results
Status: 200 OK Time: 136 ms Size: 260 B Download
Pretty Raw Preview JSON
[
{
"id": 1,
"title": "Test"
}
]
Learn Build Browse
Type here to search
ENG 9:24 PM 12/18/2020

مشکلات و توضیحات تکمیلی

دستور npm install –g @nestjs/nest-cli پیام خطا داد و به جای آن از دستور npm install –g @nestjs/cli استفاده کردم.

چک های مربوط به داده های دیتابیس مثل تعداد حرف اسم در صورت رعایت نکردن پیام خطا ندادند. و مجبور به اضافه کردن خط زیر در فایل mian.ts شدم.

```
app.useGlobalPipes(new ValidationPipe());
```