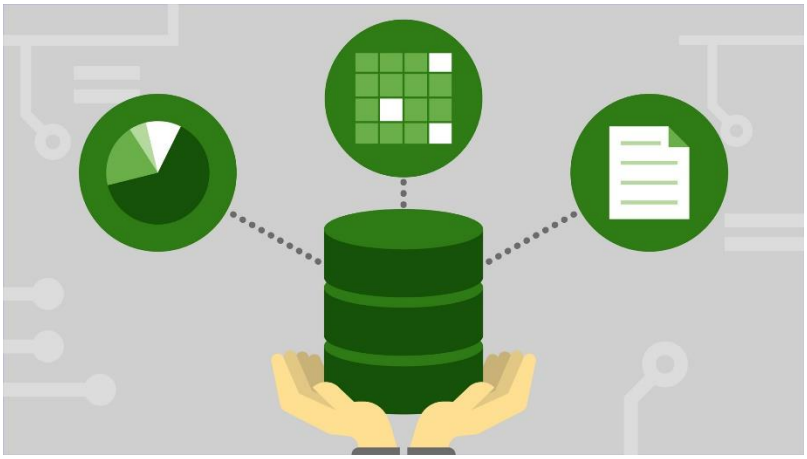
به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر



آزمایشگاه پایگاه‌داده

دستورکار شماره ۲

شماره دانشجویی

۸۱۰۱۹۶۴۹۱

طاها شعبانی

بهار ۱۴۰۰

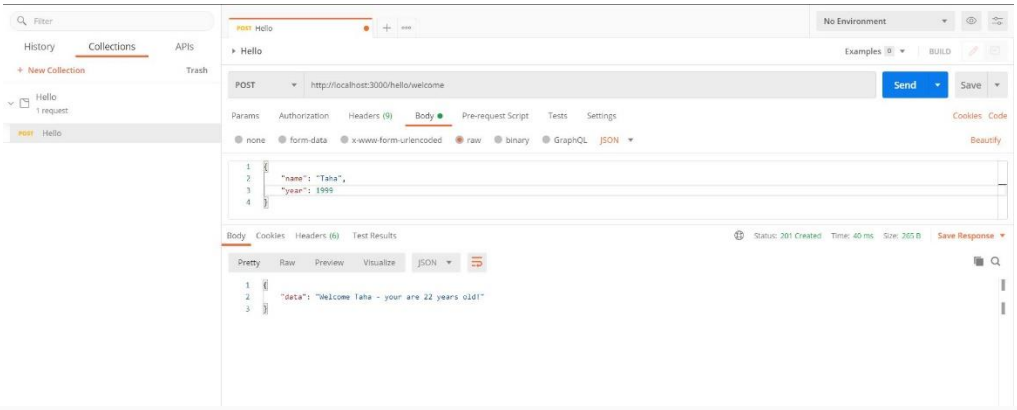
گزارش فعالیت‌های انجام شده

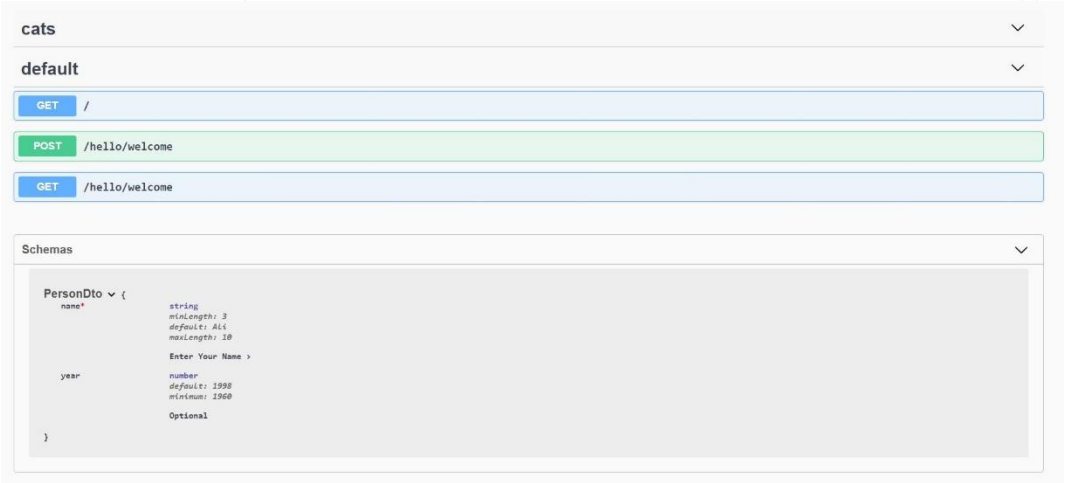
مقدمه و بخش ۱

در این قسمت و پس از طی کردن مراحل ابتدایی که در گزارش کار آمده بود و در زیر نیز عکس‌هایی بسیار مختصر از این مراحل مشاهده می‌کنیم، تا انتهای بخش ۱ این آزمایش انجام شده است:

https://github.com/tahaShm/db-lab-2

Part 1 commit SHA: `5b44f3d74bb80aaa24a22ae1f9c5f9032b0a9d37`





بخش ۲

قابلیت‌هایی که از دستور کار اول انتخاب شده اند:

دو قابلیت فریلنسر:

۱. فریلنسر می‌تواند ثبت نام کرده و اطلاعات پروفایلی و رزومه خود را ارتقا دهد.

۲. فریلنسر می‌تواند گزارش‌های مالی خود را مشاهده و درخواست دریافت وجه خود از سایت را ارسال کند.

دو قابلیت کارفرما:

۱. کارفرما می‌تواند ثبت سفارش و یا ویرایش یک سفارش را انجام دهد.

۲. کارفرما می‌توانند در صورت بروز ابهام، سوال، چالش، درگیری با فریلنسر و ... با ارسال تیکت دغدغه خود را مطرح کند.

موارد ستاره دار در پایین جدول تعریف شده اند:

API	HTTP Method	Description	Request Body	Response Body
/freelancers	POST	ثبت نام یک فریلنسر جدید	Freelancer*	200 successful/
/freelancers/{freelancerId}	GET	جستجو و دریافت اطلاعات فریلنسر با ایدی مشخص	Freelancer Id	200 Freelancer*/ 400 invalid Id/
/freelancers/{freelancerId}	PUT	ویرایش اطلاعات فریلنسر مورد نظر	Freelancer Id, Freelancer*	200 successful/ 400 invalid Id/
/freelancers/{freelancerId}	DELETE	حذف اطلاعات فریلنسر مورد نظر	Freelancer Id	200 successful/ 400 invalid Id/
/freelancers/{freelancerId}/withdraws	POST	ثبت درخواست برداشت وجه از سایت	Withdraw	200 successful/ 400 invalid Id/
/freelancers/{freelancerId}/withdraws	GET	دریافت اطلاعات تراکنش‌ها و تعاملات مالی	Freelancer Id	200 List of Withdraws*/ 400 invalid Id/
/freelancers/{freelancerId}/withdraws/{withdrawId}	PUT	ویرایش درخواست‌های مالی	Freelancer Id, Withdraw Id	200 successful/ 400 invalid Id/ 403 forbidden
/freelancers/{freelancerId}/withdraws/{withdrawId}	DELETE	حذف اطلاعات درخواست مالی	Freelancer Id, Withdraw Id	200 successful/

				400 invalid Id/ 403 forbidden
/employers/{employerId}/orders	POST	ثبت سفارش جدید از طرف کارفرما	Employer Id, Order*	200 successful/ 400 Invalid Id
/employers/{employerId}/orders/{orderId}	GET	دریافت اطلاعات سفارش مورد نظر	Employer Id, Order Id	200 Order*/ 400 invalid Id/ 403 Forbidden
/employers/{employerId}/orders/{orderId}	PUT	ویرایش اطلاعات سفارش مورد نظر	Employer Id, Order Id, Order*	200 successful/ 400 invalid Id/ 403 Forbidden
/employers/{employerId}/orders/{orderId}	DELETE	حذف اطلاعات سفارش مورد نظر	Employer Id, Order Id	200 successful/ 400 invalid Id/ 403 Forbidden
/employers/{employerId}/tickets	POST	ثبت تیکت جدید (توسط کارفرما)	Employer Id, Ticket*	200 successful/ 400 invalid Id
/employers/{employerId}/tickets/{ticketId}	GET	دریافت اطلاعات تیکت	Employer Id, Ticket Id	200 successful/ 400 invalid Id/ 403 Forbidden
/employers/{employerId}/tickets/{ticketId}	PUT	ویرایش اطلاعات تیکت مورد نظر	Employer Id, Ticket Id, Ticket*	200 successful/ 400 invalid Id/ 403 Forbidden
/employers/{employerId}/tickets/{ticketId}	DELETE	حذف تیکت مورد نظر	Employer Id, Ticket Id	200 successful/

400 invalid Id/ 403 Forbidden				

Freelancer	Withdraw	Order	Ticket

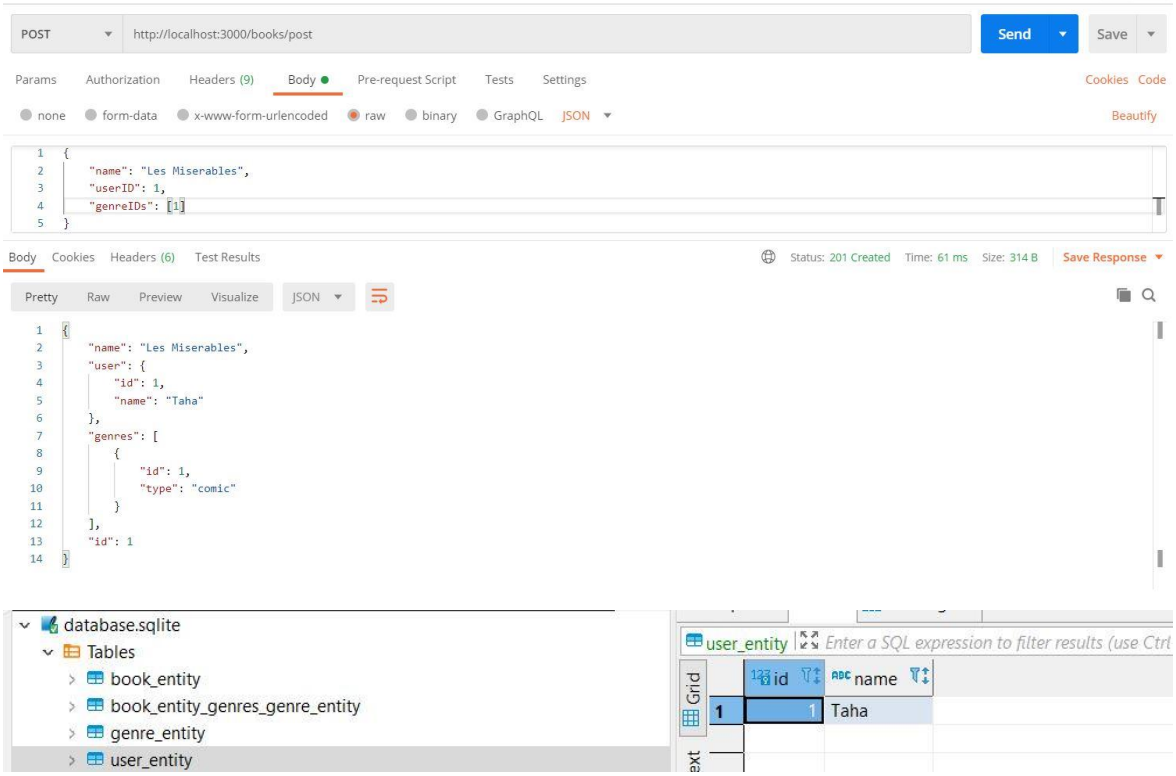
{
Name: string,
Email: string,
phoneNumber: string,
password: string,
image: string| base64,
degrees: json,
experiences: json,
skills: json,
}
	{
	withdrawTime: date,
	wNum: number,
	amount: number,
	status: string
	}
		{
		Name: string,
		descText: string,
		size: number,
		neededSkills: json,
		type: string,
		dueDate: date,
		area: string,
		minGuarantee: number,
		descFile: json
		}
			{
			creationTime: date,
			ticketNum: number,
			type: string,
			title: string,
			desc: string
			}

البته لازم به ذکر است که در بخش ۶، علاوه بر این api ها که لازمه اصلی هستند، یک سری api های دیگر نیز برای سهولت کار در نظر گرفته شده است.

Part 2 commit SHA: `a8f0ca010f2dab97cb8764c63475559167497947`

بخش ۳

در این مرحله و پس از نصب پکیج‌های typeorm و sqlite و همچنین طی کردن مراحل ذکر شده در لینک موجود در صورت پروژه و رفع مشکلات نسبتا زیاد routing ها و سایر مواردی که در لینک وجود داشت، در نهایت اضافه شدن هر سه مولفه user, genre و book با موفقیت بررسی و در DBeaver نیز با ساخت یک پروژه جدید sqlite تست شد:



Part 3 Commit SHA: `a905097df938e3e63d0e87c46eb5c66042de7b9c`

بخش ۴

در این بخش نیز پس از نصب پکیج مربوط به postgre در nest و اعمال تغییرات لازم هم در فایل ormconfig.json و همینطور افزودن قابلیت‌های حذف و آپدیت که به ترتیب با http method های DELETE و PUT انجام شد، و با استفاده از توابع delete و update کلاس BaseEntity، در نهایت قابلیت حذف و آپدیت نیز به هر سه مورد book, user و genre اضافه و تست شد. برای مثال در زیر مثال تست آپدیت برای Book را آورده ایم: (برای جلوگیری از طولانی شدن گزارش عکس پاک کردن کتاب را نمی‌گذاریم اما از طریق کامیت بخش ۴ قابل تست و دسترسی است)

Part 4 Commit SHA: `91a514912e7fb7f939452504314b7f1411a678b6`

بخش ۵: امتیازی (jwt)

بخش ۶

در این بخش پس از ساخت کلاس‌های dto و همینطور BaseEntity برای entity های لازم و برقراری relation ها مطابق با طراحی‌های آزمایش شماره ۱، در دو commit یکی برای api های freelancers و دیگری برای api های employers ماژول‌ها را تعریف کردیم.

همچنین با مشخص کردن error code ها و نحوه ورودی در swagger نیز، امکان تست آن نیزبه سادگی وجود دارد.

همچنین در یک فایل با نام temp.txt نمونه‌هایی از داده‌ها برای ساخت محتوا نیز قرار داده شده است.

پس از تعریف دقیق api ها و تست صحت همگی آن‌ها، در نهایت خروجی در دو کامیت زیر در دسترس است:

Part 6.1 (freelancers) Commit SHA: `23d51fc6f077d5fe72bb9887bd88e26ce521c7b7`

Part 6.2 (employers) Commit SHA: `28628ad0c22f3e55c5b7160ab48dada81284088f`

POST /freelancers

GET /freelancers

GET /freelancers/{id}

DELETE /freelancers/{id}

PUT /freelancers/{id}

POST /freelancers/{id}/withdraws

GET /freelancers/{id}/withdraws

GET /freelancers/{id}/withdraws/{wid}

DELETE /freelancers/{id}/withdraws/{wid}

PUT /freelancers/{id}/withdraws/{wid}

POST /employers

GET /employers

GET /employers/{id}

DELETE /employers/{id}

POST /employers/{id}/orders

GET /employers/{id}/orders

GET /employers/{id}/orders/{oid}

DELETE /employers/{id}/orders/{oid}

PUT /employers/{id}/orders/{oid}

POST /employers/{id}/tickets

GET /employers/{id}/tickets

GET /employers/{id}/tickets/{tid}

DELETE /employers/{id}/tickets/{tid}

PUT /employers/{id}/tickets/{tid}

مشکلات و توضیحات تکمیلی

یکی از منابع که در مرحله مقدمه وجود داشت دارای کد با مشکلات زیادی بود که زمانی صرف رفع ارورهای این مرحله شد. (بخش books, users,, genres)

آنچه آموختم

1. آشنایی و یادگیری کار با فریم ورک nest.js و آشنایی با ساختار module, component, service و … که شباهت زیادی به فریم ورک سمت کلایتن angular دارد.
2. نحوه تعامل کد سمت سرور با پایگاه داده و نحوه تست و ارزیابی آن به واسطه ابزارهایی نظیر swagger و postman.
3. فرایند طراحی api و design آن و پیاده سازی در ساختار کد سمت سرور با ابزار typeorm.
4. در آزمایش ۱ و ۲ یک روال منطقی از طراحی پایگاه داده تا پیاده سازی Apiهای لازم را یاد گرفتیم.