| | |
|---|---|
| دیتابیس | mysql<br>-در قسمت db.config نام کاربری و رمز عبور دیتابیس تغیر داده شود.<br>-اسکریپت دیتابیس جهت اجرا قرار داده شده است.<br>-فایل پستمن ارائه شده است. |
| فریم ورک | express |
| orm | sequelize |

با توجه به توضیحاتی که داده شده، جدول topic و جدول section دارای رابطه ی n:m هستند.

همین طور qa و topic نیز رابطه ی n:m دارند.

در شروع کار مدل ها ایجاد شدند و با دستورات خود sequelize روابط آن ها تشکیل شد.

(در قسمت: model/index.js )

درواقع دو جدول میانی به وجود آمد که رابطه ی میان section-topic و qa-topic را نشان می دهد.

---

برای تمایز روت های مربوط به ادمین و روت های یوزرهای معمولی نیازمند جدول دیگری با نام یوز بودیم؛

طبق اطلاعات داده شده هر کاربر می تواند دو رول ادمین و غیر ادمین را داشته باشد .

* برای اجتناب از پیچیدگی کار، کاربر sepideh با رمز ۱۲۳۴ به عنوان کاربر ادمین به سیستم اضافه شد.

و دو روت برای رجیستر و لاگین به روت های یوزر ایجاد کردیم.

در ادامه برای شناسایی آیدی کاربر در هر ورود و همچنین ادمین بودن و نبودن کاربر دو middleware به نام های verify token و isAdmin اضافه شدند.

بعد از این مرحله می توانیم آیدی کاربر را در هر درخواست رصد کنیم و بتوانیم سیستم view و like و dislike را مدیریت کنیم.

برای سیستم like و view باید بدانیم کاربری که مثلا برای QA خاصی درخواست داده ، آیا قبلا مشاهده کرده یا نه و آیا آن را لایک کرده است یاخیر.

برای این کار جدول واسطی بین یوزر و QA ساختیم و فیلد بولین islike را به آن اضافه کردیم.

سناریو را به این شکل است که ایجاد رکورد در این جدول برابر view برای آن یوزر است.

و در قسمتی(روتی) که اطلاعات QA درخواست می شود، رکورد جدید در این جدول ثبت می شود.

---

```
      app.get('/qa/info/:qa_id',  [auth.verifyToken],
controller.info)
```

برای مدیریت لایک نیز روتی نوشته شد که آیدی QA را گرفته و وجود رکورد برای کاربر درخواست کننده را بررسی کند. اگر شخص قبلا QA را دیده باشد پس رکوردی وجود دارد و فقط طبق اطلاعات ارسالی کاربر رکورد مربوطه به روز رسانی می‌شود.(like & dislike)

اما اگر قبلا دیده نشده باشد، رکوردی را با مقدار لایک ارسال شده، می سازد.

در واقع می توانیم اینطور در نظر بگیریم کاربری که میخواهد QA را لایک کند در واقع آن را دیده است پس ویو می خورد.

---

```
app.put('/qa/like/:qa_id/:like',  [auth.verifyToken],
controller.like)
```

برای اینکه ادمین بتواند لیست ها را سورت کند، میتوانیم req.query.order ارسال کنیم که بر اساس محتوای ارسالی برای نوع سورت تصمیم گیری شود:

```
let where;
_req.query.order == 1 ?
where = "order by t_id ASC" :
where = "order by t_id DESC"
```

البته با توجه به تستی بودن پروژه این مراحل در ساده ترین شکل ممکن انجام شد.

همچنین

برای جست و جوی QA نیز از req.query.search استفاده میکنیم که عبارت دریافتی از کاربر را با دستور لایک روی نام QA اعمال می کنیم.