باسمه تعالی

دانشکده علوم و مهندسی کامپیوتر

تمرین شماره ۲

درس مهندسی اینترنت (توسعه برنامه‌های وبی)

موعد تحویل: ۱۴۰۱/۰۲/۵

مدرس:

محمدرضا رازیان

تدریس‌یاران:

علیرضا چقامیرزایی (سرگروه تدریس‌یاران)

نیما حیدری‌نسب، امین ساوه‌دورودی، حسن مجیدزنوزی،

عباس میرقاسمی و محمد حسام نصیری

بهار ۱۴۰۱ - ۱۴۰۲

مقدمه

اکثر سامانه های تحت وب حال حاضر دارای دو بخش کد هستند که یک بخش سمت سرور و دیگری سمت کاربر می باشد. در این تمرین سعی داریم تا به صورت کاربردی از Node.js, Express.js, Mongoose, Git جهت طراحی کد سمت سرور یک سامانه انتخاب واحد تحت وب استفاده نماییم.

پیاده سازی فاز اول API سامانه انتخاب واحد

در این تمرین قرار است با ساخت و پیاده سازی فاز اول سامانه انتخاب واحد، به صورت عملی با زبان برنامه نویسی جاوا اسکریپت و محیط اجرای Node آشنا شوید. یک سامانه انتخاب واحد براساس آنچه در درس پایگاه داده آموخته اید، دارای طرحواره (Schema) های دانشجو، استاد، درس و مدیرآموزش است. ابتدا در گام اول انتظار می رود با کمک mongoose به دیتابیس mongoDB خود وصل شده و یک دیتابیس جدید ساخته و موجودیت های خود را در آن تعریف کنید.

شرح اطلاعات موجودیت ها

فیلدهای اجباری که باید توسط شما پیاده‌سازی شوند در زیر ذکر شده است. به صورت عمومی تمامی موجودیت‌ها باید شامل اطلاعات زیر باشند و اطلاعات خاص هر موجودیت در زیر آمده است:

اطلاعات جامع کاربران

- نام و نام خانوادگی
- شماره دانشجویی/استادی/ کاربری
- رمز عبور
- ایمیل
- تلفن همراه

دانشجو

- مقطع تحصیلی
- سال ورودی
- ترم ورودی
- معدل
- دانشکده

- رشته

استاد

- دانشکده
- رشته
- مرتبه

مدیر آموزشی

- دانشکده

مدیر IT

درس مصوب

- نام درس
- پیش نیاز(ها)
- هم نیاز(ها)
- واحد

درس ترمی

علاوه بر ویژگی های درس مصوب، شامل اطلاعات زیر نیز خواهد بود:

- تاریخ و زمان کلاس
- تاریخ و زمان امتحان
- مکان امتحان
- استاد درس
- ظرفیت
- ترم تحصیلی

لازم به ذکر است که اطلاعات دیگری شامل لیست‌های مورد نیاز مانند لیست دروس دانشجو، استاد و ... در فازهای بعدی باید به موجودیت ها اضافه خواهند شد. بنا به نظر خود اگر نیاز به تعریف موجودیت های دیگر مانند ترم، امتحان و ... می باشد، تعریف شود.

طراحی نقاط اتصال (Endpoints)

به زبان ساده، واسط برنامه نویسی (API) واسطی برایتان فراهم می‌کند تا از منطق آن استفاده کنید. برای استفاده از آن نیز لازم نیست که جزئیاتِ چگونه عمل کردن این منطق را بدانید. تا زمانی که در سطحِ برنامه‌نویسیِ (نه گرافیکی) چیزی هستید، می‌توانید از این تعریف برای هر زبان، پروتکل یا محیطی استفاده کنید.

اما یک API چگونه این کار را انجام می‌دهد؟ یک واسط برنامه نویسی قابلیت‌ها/توابع را در کنار هم قرار می‌دهد و شما می‌توانید با استفاده از نقاط اتصال به آنها دسترسی پیدا کنید (معمولاً URLهایی ارائه می‌شوند، که برای ارتباط با واسط برنامه نویسی (API) باید از آنها استفاده کنید). این نقاط اتصال تنها روش ارتباط با یک واسط برنامه نویسی (API) هستند. هر نقطه اتصال برای درخواست و پاسخ فرمت‌های مشخصی دارد، که می‌توانید آنها را در اسناد مربوط به واسط برنامه نویسی پیدا کنید.

Endpoint های عمومی :

- **ورود کاربران** **POST** `/login`

Endpoint های دیگر نیاز به احراز هویت خواهند داشت که بررسی شود آیا کاربر وارد سامانه شده است یا خیر. اگر عملیات ورود را انجام نداده بود با خطای مناسب به آن پاسخ داده شود.

مدیر IT

- ساخت، ویرایش، حذف، دریافت استاد و لیست اساتید

- **POST** `/admin/Professor`
- **PUT** `/admin/Professor/{ID}`
- **DELETE** `/admin/Professor/{ID}`
- **GET** `/admin/Professors`
- **GET** `/admin/Professor/{ID}`

- ساخت، ویرایش و حذف، دریافت دانشجو و لیست دانشجویان

- **POST** `/admin/student`
- **PUT** `/admin/student/{ID}`

- **DELETE** /admin/student/{ID}
- **GET** /admin/students
- **GET** /admin/student/{ID}

- ساخت، ویرایش و حذف، دریافت مدیر آموزش و لیست مدیران

- **POST** /admin/manager
- **PUT** /admin/manager/{ID}
- **DELETE** /admin/manager/{ID}
- **GET** /admin/managers
- **GET** /admin/manager/{ID}

- دیدن لیست دروس

مدیر آموزش

- ساخت، ویرایش و حذف دروس مصوب و ترمی

- **POST** /course
- **PUT** /course/{ID}
- **DELETE** /course/{ID}
- **GET** /courses
- **GET** /course/{ID}

- دیدن لیست اساتید و دانشجویان

- **GET** /students
- **GET** /student/{ID}
- **GET** /Professors
- **GET** /Professor/{ID}

دانشجو

- ویرایش اطلاعات قابل تغییر خود (در اینجا باید بررسی شود که آیدی ارسال شده مربوط به خود دانشجو باشد در غیر این صورت با خطای مناسب پاسخ داده شود)

- **PUT** /student/{ID}

- دیدن لیست دروس با امکان فیلتر بر اساس رشته

- **GET** /courses
- **GET** /course/{ID}

استاد

- ویرایش اطلاعات قابل تغییر خود (در اینجا مانند مورد قبلی مشابه اعتبارسنجی (validation) مناسب شود.)

❖ **PUT** `/Professor/{ID}`

- دیدن لیست دروس با امکان فیلتر بر اساس رشته

- **GET** `/courses`
- **GET** `/course/{ID}`

در این پروژه ساختار پروژه بر اساس معماری تمیز (Clean Architecture) پیاده‌سازی شود. و بقیه معماری در فاز های بعد نیز بر اساس همین معماری پیاده سازی شود.

- استفاده از فایل .env برای نگه داشتن اطلاعات حساس خود الزامیست.
- در طول انجام پروژه خود از Git استفاده شود و تاریخچه کامیت‌های شما بررسی خواهد شد.
- مدیریت خطاها به صورت کامل باید انجام شود و از استاندارد RFC 7231 پیروی شود. همچنین در خود سیستم نیز در هنگام خطاها از common log format استفاده باید شود.
- برای مستند سازی API ها از OpenAPI باید استفاده شود. تحویل شما منوط به موجود بودن کامل مستندات می باشد.

شیوه و مکان ارسال تمرین

لطفا تمامی فایل‌ها و پوشه‌های ساخته شده را به صورت فایل فشرده با فرمت zip در بخش مربوط به این تمرین در سایت کوئرا بارگذاری نمایید. توجه شود که پروژه باید دارای فایل gitignore مناسب باشد و فولدر node_module در فایل ارسالی قرار نداشته باشد.

موفق باشید