امتحان پایان ترم درس مهندسی نت

بهار 1401

هدف این تمرین پیاده سازی api برای مدیریت درس های دانشجویان است.

اطلاعات هر دانشجو در جدول زیر قابل مشاهده است:

Key	Type	Description	Example
id	int	student id	97243038
average	Float	average score	17.2
courses	Course[]	List of courses	*Shown later*
last_updated	Time	Latest update timestamp	23:51 05-12-2020

همچنین مشخصات موجودیت Course در جدول زیر قابل مشاهده است:

Key	Type	Description	Example
name	String	course name	internet engineering
id	int	course id	5
grade	Float	course grade	20

بخش اول - پیاده‌سازی چهار عملیات اصلی CRUD ویژه‌ی دانشجو ها:
در جدول زیر endpointهای چهار قسمت API را مشاهده می‌کنید و در ادامه برای هر کدام توضیحات بیشتری آورده شده است.

Endpoint	Description
POST /students	Create a new student
GET /students	Get all the students
{PUT /students/studentid	student id Edit an existing student by
{DELETE /students/studentid	student id Delete an existing student by

۱. عملیات ایجاد یا Create:
در این endpoint از API پس از دریافت شماره دانشجویی پول توسط سرور، اگر آن تکراری نباشد، یک دانشجوی جدید ایجاد شده و نتیجه‌ی آن برمی‌گردد. در صورت وجود خطا نیز می‌بایست ارور مناسب برگردانده شود. دقت کنید که مقدار last_updated در جواب سرور همان زمان ایجاد دانشجو است.
نمونه‌ای از درخواست کاربر:

```
// POST /wallets { "studentid": 97243038, }
```

پاسخ سرور (در صورت نبود مشکل):

```
{ "studentid": "97243038", "average": 0.0, "courses": [], "last_updated": "2022-6-21 12:32", "code": 200, "message": "student added successfully!" }
```

۲. عملیات دریافت، خواندن یا Read:

در این endpoint از API نیازی به وارد کردن اطلاعات نیست و صرفاً با درخواست زدن، اطلاعات تمامی دانشجویان موجود برگردانده می‌شود.

درخواست به سرور:

```
// GET /students
```

پاسخ سرور پس از درخواست:

```
{ "size": 5, "students": [ { "studentid": "97243038", "average": 17.2, "Courses":[ { "name": "internet engineering", "id": "5", "grade": 20,}, {...}, {...} ], "last_updated": "2022-6-21 12:32" }, {...}, {...}, {...}, {...} ], "code": 200, "message": "All students received successfully!" }
```

۳. عملیات ویرایش یا Update:

در این endpoint تنها می‌توان شماره دانشجویی مورد نظر را عوض کرد که به راحتی با درخواست زیر قابل انجام است:

```
// PUT /students/{studentid} { "studentid": "97240000", }
```

پاسخ سرور (در صورت نبود مشکل):

```
{ "studentid": "97243038", "average": 17.2, "Courses":[ {...}, {...} ], "last_updated": "2022-6-21 12:32", "code": 200, "message": "studentid changed successfully!" }
```

۴. عملیات حذف یا Delete:

در آخرین endpoint مربوط به دانشجویان با استفاده از شماره دانشجویی می‌توان آن دانشجو را حذف کرد!

```
// DELETE /students/studentid
```

پاسخ سرور (در صورت نبود مشکل):

```
{ "studentid": "97243038", "average": 17.2, "Courses":[ {...}, {...} ], "last_updated": "2022-6-21 12:32", "code": 200, "message": "student deleted successfully!" }
```

بخش دوم - پیاده‌سازی چهار عملیات اصلی CRUD ویژه‌ی دروس:
در جدول زیر endpointهای چهار قسمت API بخش دوم را مشاهده می‌کنید. دقت کنید که تمامی عملیات CRUD مربوط به دروس حتماً باید مربوط به یک دانشجوی خاص باشند.

Endpoint	Description
`POST /{studentid}/course`	Create a new course for a student
`GET /{studentid}`	Get a students's information
`PUT /{studentid}/{courseid}`	Edit an existing course by `courseid`
`DELETE /{studentid}/{courseid}`	Delete an existing course by `id`

۱. عملیات ایجاد یا Create:
در این endpoint از API پس از دریافت مشخصات درس توسط سرور، اگر آن نام یا آی‌دی آن تکراری نباشد، یک درس جدید برای دانشجوی مورد نظر ایجاد شده و نتیجه‌ی آن برمی‌گردد. در صورت وجود خطا نیز می‌بایست ارور مناسب برگردانده شود.
نمونه‌ای از درخواست کاربر:

```
// POST /{studentid}/courses { "name": "internet engineering", "id": 5, "grade": 20}
```

پاسخ سرور (در صورت نبود مشکل):

```
{ "name": "internet engineering", "id": 5, "grade": 20, "code": 200, "message": "course added successfully!" }
```

۲. عملیات دریافت، خواندن یا Read:

در این endpoint از API نیازی به وارد کردن اطلاعات نیست و صرفاً با درخواست زدن، اطلاعات آن دانشجو به همراه تمامی درس های آن برگردانده می‌شود.

درخواست به سرور:

```
// GET /{studentid}
```

پاسخ سرور پس از درخواست:

```
{ "studentid": "97243038", "average": 17.2, "Courses":[ { "name": "internet engineering", "id": "5", "grade": 20,}, {...}, {...} ], "last_updated": "2022-6-21 12:32", "code": 200, "message": "All courses received successfully!" }
```

۳. عملیات ویرایش یا Update:

در این endpoint می‌توانید اطلاعات مربوط به یک درس خاص از دانشجو را عوض کنید:

```
// PUT /{studentid}/{courseid} { "name": "internet engineering", "id": 5, "grade":5 }
```

پاسخ سرور (در صورت نبود مشکل):

```
{ "name": "internet engineering", "id": 5, "grade":5, "code": 200, "message": "grade updated successfully!" }
```

۴. عملیات حذف یا Delete:

در آخرین endpoint مربوط به درس ها با استفاده از آی‌دی یک درس در یک دانشجو می‌توان آن درس را حذف کرد دقت کنید که یک راه برای بررسی درستی این عملیات این است که اطلاعات همان دانشجو را بخوانیم.

```
// DELETE /{studentid}/{courseid}
```

پاسخ سرور (در صورت نبود مشکل):

```
{ "name": "internet engineering", "id": 5, "grade":5, "code": 200, "message": "course deleted successfully!" }
```

نکات

بسیار مهم: در صورتی که تغییراتی در `grade` درس های موجود برای یک دانشجو انجام شود (با اضافه کردن درس یا تغییر نمره `grade` یک درس یا حذف یک درس)، مقدار ویژگی `average` آن دانشجو باید آپدیت شود. به عبارت دیگر مقدار `average` یک دانشجو باید همیشه آپدیت باشد.

- فرض کنید تمام دروس تاثیر یکسانی در معدل دارند(تعداد واحد درس ها برای محاسبه معدل بی اهمیت است).
- می‌توانید فیلد های `course name ,course id ,student id` را به هر صورتی که می‌خواهید نام‌گذاری کنید. مثال‌ها صرفاً برای درک بهتر آورده شده‌اند و نیازی به داشتن کارکترهای خاص یا فاصله در فیلد ها نیست.
- آپدیت شدن درس های هر دانشجو (اضافه کردن، ویرایش ویا حذف درس)، مثل آپدیت کردن `grade` درس مقدار `last_updated` دانشجو را آپدیت می‌کند.