مدرسان: رامتین خسروی، محمدامین صادقی
طراحان: برخی از دستیاران آموزش درس

مهلت تحویل: شنبه ۱۶ دی‌ماه ۱۳۹۶ ، ساعت ۲۳:۵۵

گردابیفای نسخه‌ی وب:

هدف این فاز از پروژه، گسترش گردابیفای بر بستر شبکه است. برای این کار یک سرور توسعه داده می‌شود که از رابط برنامه‌نویسی برنامه‌ی فاز اول استفاده می‌کند و امکان دسترسی به برخی از متدهای آن را از طریق بستر شبکه فراهم می‌کند. در این فاز شما باید در پروژه‌ی خود یک وب‌سرور ایجاد کرده و Web API [1] آن را طراحی کنید تا بتوان با آن ارتباط برقرار کرد. این ارتباط باید برپایه‌ی پروتوکل HTTP [2] طراحی شود.

برای این کار شما از مدل کلاینت و سرور[3] استفاده می‌کنید که یک مدل رایانش توزیع شده[4] است. این معماری زمانی مورد استفاده قرار می‌گیرد که خدمت موردنظر باید روی چندین سیستم ارائه شود. برای رسیدگی به این موضوع، به‌جای این‌که خدمت مورد نظر روی هر یک از سیستم‌ها پیاده‌سازی و اجرا شود، می‌توان خدمت را روی یک سیستم مرکزی (سرور) پیاده‌سازی کرد و سیستم‌های دیگر (کلاینت) می‌توانند با درخواست از این سیستم خدمت مورد نظر را به کاربر خود بدهند. شما در ادامه‌ی کلاس درس به طور دقیق‌تر با مفاهیم این معماری نرم‌افزار آشنا خواهید شد.

در این فاز مرورگر وب[5] نقش کلاینت را ایفا می‌کند که از طریق پروتوکل HTTP با سرور شما ارتباط برقرار می‌کند. این پروتکل از تعدادی **متد** پشتیبانی می‌کند که هدف آن‌ها توصیف عملکرد درخواستی از سرور است. دو متد ارتباطی مهم عبارتند از GET و POST که به ترتیب برای درخواست اطلاعات و ارسال اطلاعات به آن استفاده می‌شوند. همچنین **پارامترهای پرسمان**[6] یک راه برای مشخص کردن اطلاعات ارسال شده به سرور هستند که هر یک شامل یک نام و یک مقدار هستند.

[1] Web Application Programming Interface

[2] Hyper Text Transfer Protocol

[3] Client Server Model

[4] Distributed computing

[5] Web Browser

[6] Query Parameter

درخواست‌ها[7]

نحوه‌ی تعامل با یک سرور HTTP بدین صورت است که ابتدا از سمت کلاینت یک درخواست به آن فرستاده می‌شود و سپس سرور پاسخ مناسبی را برای آن تولید می‌کند و به کلاینت ارسال می‌کند. برای این کار لازم است تا تعدادی ***نقطه‌ی دسترسی***[8] در سرور تعریف شوند تا درخواست‌ها به آن آدرس‌ها فرستاده شوند. هر نقطه‌ی دسترسی توسط یک URI[9] (آدرس) مشخص می‌شود که با یک فرمت مشخص درخواست‌ها را دریافت می‌کند و پاسخ مشخصی را به آن‌ها می‌دهد.

مثلا نقطه‌ی دسترسی مربوط به عمل ورود کاربر به سیستم دارای آدرسی همچون [http://gerdabify.com]**loign**/ است. این نقطه‌ی دسترسی درخواست‌های HTTP با متد POST را دریافت می‌کند که در آن پارامترهای پرسمان username و password وجود دارد و اگر مقادیر این پارامترها معتبر بودند، سرور در پاسخ تاییدیه‌ی ورود کاربر را برای کلاینت ارسال می‌کند.

در ادامه جزییات درخواست‌هایی که سرور شما باید به آن‌ها رسیدگی کند شرح داده می‌شوند. لازم به ذکر است که پاسخ سرور به هر یک از این درخواست‌ها باید به زبان HTML باشد مثلا درخواست GET /musics حتما باید یک صفحه شامل نام کامل موسیقی‌ها باشد، اما ظاهر جزییات دیگر این صفحه به دلخواه شما خواهد بود.

- **ورود:**

این درخواست باید هویت کاربر را با استفاده از نام‌کاربری و گذرواژه فراهم شده، احراز کند. پس از احراز هویت، سرور باید یک توکن[10] (برای سادگی می‌توانید توکن را همان username در نظر بگیرید) را در پاسخ به کلاینت برگرداند.

حالت‌های خطا:

۱) نام‌کاربری در سیستم موجود نباشد یا گذرواژه درست نباشد.

```
Request:
POST /login
Params:{
	username = gerdab
	password = 123456
}
```

در این درخواست از متد POST استفاده می‌شود که به آدرس /login ارسال می‌شود و پارامترهای پرسمان ارسال شده به سرور عبارتند از username که مقدار نام‌کاربری (برای مثال gerdab) و password که مقدار گذرواژه‌ی کاربر (برای مثال 123456) را مشخص می‌کنند.

---

7 Requests

8 Endpoint

9 Uniform Resource Identifier

10 سرور برای پاسخ به هر درخواست ابتدا باید دریابد که درخواست توسط چه کاربری داده‌شده است تا متناسب با آن پاسخ دهد. احراز هویت یک کاربر با فراهم کردن نام‌کاربری و گذرواژه توسط او انجام می‌شود. اما برای این که کاربر در هر درخواست خود نام‌کاربری و گذرواژه‌ی خود را به سرور نفرستد، سرور پس از احرازهویت کاربر یک توکن برای او ایجاد می‌کند. از این پس سرور هر درخواستی که شامل این توکن باشد را به عنوان درخواستی از طرف آن کاربر می‌شناسد. برای همین سرور باید پس از احراز هویت یک جدول از نام‌کاربری‌ها و توکن‌های ایجاد شده برای آن‌ها ذخیره کند.

- **ثبت‌نام (امتیازی):**

این درخواست باید کاربر را در سیستم با استفاده از داده‌های فراهم شده ثبت‌نام کند.
حالت‌های خطا:
۱) نام‌کاربری فراهم شده از قبل در سیستم موجود باشد.
۳) نام کاربری در درخواست موجود نباشد.
۴) نام فراهم در درخواست موجود نباشد.
۵) نام خانوادگی در درخواست موجود نباشد.
۶) نوع‌کاربر در درخواست موجود نباشد.
۷) گذرواژه در در درخواست موجود نباشد.
۸) نوع کاربر یکی از مقدارهای aritst, editor یا admin نباشد.

```
Request:
POST /signup
Params:
        username  = greedy
        firstanme = navid
        lastname  = gerdab
        userType  = artist
        password  = 123456
}
```

با توجه به امتیازی بودن بخش ثبت‌نام درصورتی که این بخش را پیاده‌سازی نمی‌کنید باید چند کاربر ثبت‌نام شده از قبل در سیستم وجود داشته باشند.

توجه: برای رسیدگی به تمام درخواست‌هایی که در ادامه آمده است نیاز است تا توکن دریافت شده در در آن‌ها قرار داده شود. اگر توکن موجود در درخواست معتبر نباشد باید یک ارور مناسب در پاسخ به کلاینت داده شود. در حالت‌های خطا هم پاسخ سرور باید به خطای
ایجاد شده و دلیل آن را به شیوه‌ی مناسب پاسخ دهد.

- **جستجو:**

این درخواست برای جستجو در میان تمام موسیقی‌های موجود در کتابخانه سرور به کار می‌رود که کلیدواژه جستجو در پارامتر keyword آن قرار دارد. پاسخ این درخواست شامل لیست نام کامل موسیقی‌هایی است که کلیدواژه‌ی جستجو در نام آن‌ها وجود دارد. این پاسخ باید در قالب یک جدول به کاربر نمایش داده شود. برای این کار از زبان HTML استفاده کنید.

```
Request:
POST /search
Params:{
        token = sessionToken
        keyword = searchKeyword
}
```

- **موسیقی‌ها:**

پاسخ به این درخواست باید یک صفحه شامل لیست نام کامل تمام موسیقی‌های موجود در کتابخانه‌ی سرور باشد. این پاسخ باید در قالب یک جدول به کاربر نمایش داده شود. برای این کار از زبان HTML استفاده کنید.

```
Request:
GET /musics
Params:{
	token = sessionToken
}
```

- **آلبوم‌ها:**

پاسخ به این درخواست باید یک صفحه شامل لیست نام کامل تمام آلبوم‌های موجود در کتابخانه‌ی سرور باشد. این پاسخ باید در قالب یک جدول به کاربر نمایش داده شود. برای این کار از زبان HTML استفاده کنید.

```
Request:
GET /albums
Params:{
	token = sessionToken
}
```

- **لیست‌پخش‌ها:**

پاسخ به این درخواست باید یک صفحه شامل لیست نام کامل تمام لیست‌پخش‌های کاربری که وارد سیستم شده است باشد. این پاسخ باید در قالب یک جدول به کاربر نمایش داده شود. برای این کار از زبان HTML استفاده کنید.

```
Request:
GET /playlists
Params:{
	token = sessionToken
}
```

این درخواست باید یک لیست‌پخش با نام مشخص شده برای کاربری که وارد سیستم شده‌است ایجاد کند.

```
Request:
POST /playlists
Params:{
	token = sessionToken
	name = playlistName
}
```

- **هنرمندان:**

پاسخ به این درخواست باید یک صفحه شامل لیست نام کامل تمام هنرمندان موجود در کتابخانه‌ی سرور باشد. این پاسخ باید در قالب یک جدول به کاربر نمایش داده شود. برای این کار از زبان HTML استفاده کنید.

```
Request:
GET /artists
Params:{
        token = sessionToken
        token = username
}
```

نکات پایانی:

- برای انجام فاز بعدی نیاز به انجام این فاز خواهید داشت.
- برای ایجاد و برپایی یک سرور HTTP باید از کتابخانه‌ی GerdabServer استفاده کنید که در اختیار شما قرار می‌گیرد.
- برای تست برنامه‌ی خود می‌توانید از مرورگر دلخواه خود یا دستور cURL در لینوکس و یا نرم‌افزارهایی همچون Postman استفاده کنید.
- با توجه به این که در هنگام توسعه و تست برنامه سرور و کلاینت هر دو روی یک کامپیوتر اجرا می‌شوند، برای دسترسی به سرور و نقطه‌ی های دسترسی باید از آدرس‌هایی مشابه آدرس زیر استفاده کنید که مقدار 127.0.0.1 [11] در مفاهیم شبکه‌ای نشانگر کامپیوتر محلی (کامپیوتری که آدرس از آن صدا زده می‌شود) است. مقدار ۵۰۰۰ نیز **پورت** اجرایی سرور است و مقدار آن را تغییر ندهید.

http://127.0.0.1:5000/login

نحوه‌ی تحویل

تمامی فایل‌های برنامه‌ی خود را در آرشیوی با نام A7-SID.zip در صفحه‌ی CECM درس بارگذاری کنید. برای مثال، اگر شماره‌ی دانشجویی شما ۸۱۰۱۱۲۳۴۵ است، نام فایل شما باید A7-810112345.zip باشد.لطفاً از روش‌های دیگر فشرده‌سازی مانند rar یا tar.gz استفاده نکنید.

- برنامه‌ی شما باید در سیستم‌عامل لینوکس و با مترجم g++ با استاندارد C++98 ترجمه و در زمان معقول برای ورودی‌های آزمون اجرا شود.
- به فرمت و نام فایل‌های خود دقت کنید.
- از صحت فرمت ورودی‌ها و خروجی‌های برنامه‌ی خود مطمئن شوید.
- هدف این تمرین یادگیری شماست. لطفاً تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق قوانین درس با آن برخورد خواهد شد.

[11] localhost IP