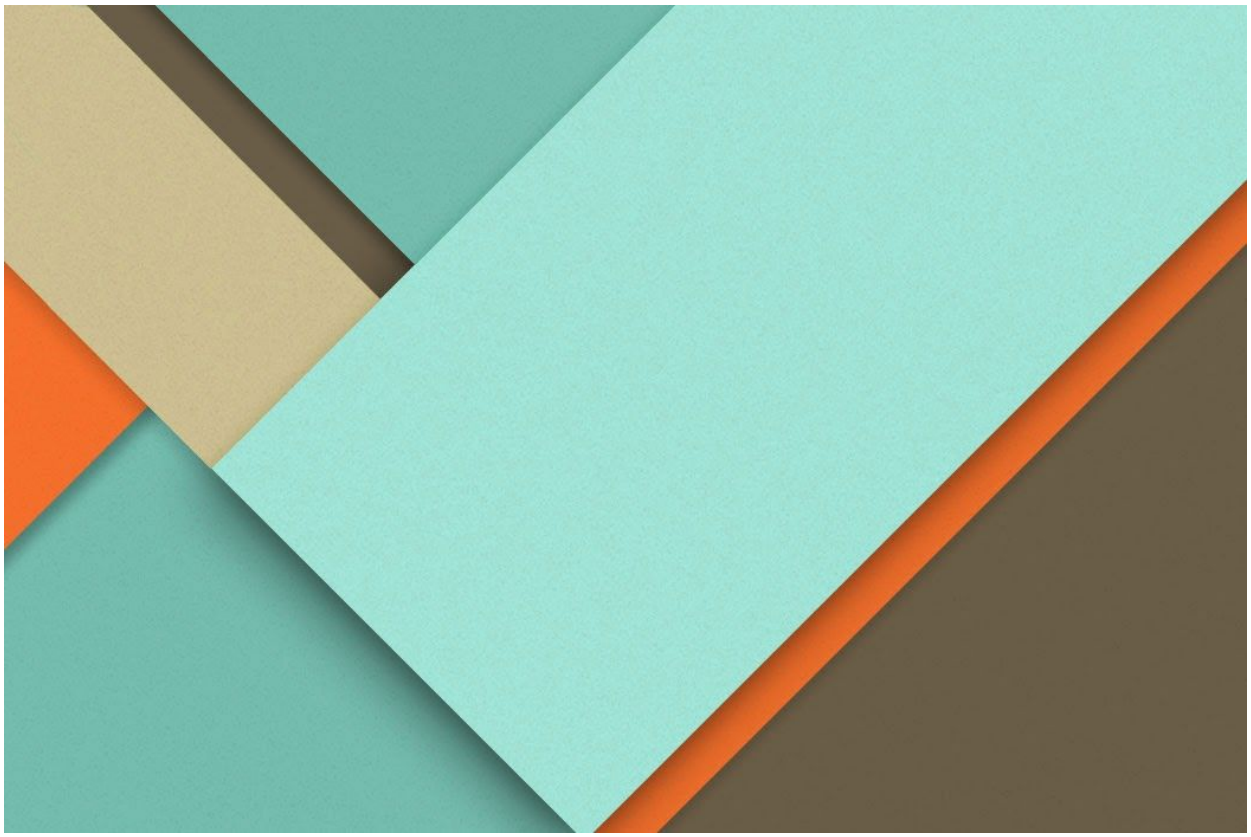

پروژه مبانی برنامه سازی

پاییز 00-99

دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف

فاز2: توییتر (سرور)

مسئولین فاز دو:

محمدهادی اثناعشری، علی حاتمی تاجیک، احسان موفق،

اشکان خادمیان، علیرضا بابازاده

فهرست

مقدمه

تا به اینجا محیط کاربری برنامه خود را ساخته‌اید و حال نوبت به بستری رسیده است که باید درخواست‌های کاربران را مدیریت کنید. در فاز دوم، شما باید سروری برای اپلیکیشن خود بسازید تا درخواست‌ها را از کاربران دریافت کند و پاسخ مناسبی به آنها بازگرداند.

در ادامه پیش‌نیازهای مختلف پروژه شامل کار سرور، پایگاه داده، توکن، فرمت ارسال اطلاعات و ماهیت‌های پروژه‌تان آورده شده. پس از آن نیز تک‌تک دستوراتی که لازم است در سرور بررسی شود توضیح داده خواهند شد. پیشنهاد می‌کنیم داک را به همین ترتیبی که نوشته شده مطالعه کنید چون هر صفحه ممکن است تکمیل‌کننده اطلاعات صفحات قبلی باشد.

خیلی مهمه که بدونید، پیاده‌سازی‌هایی که برای ماهیت‌ها، پایگاه داده و توکن گفته شده، همه و همه پیشنهادی‌اند! از شما انتظار می‌رود سرورتان پایگاه داده‌ای داشته باشد که پس از هر بار خاموش شدن سرور، اطلاعاتش پاک نشود و دستوراتی که در بخش دستورات آمده را بتواند به نحو احسن بررسی کند و سیستم اعتبارسنجی مبتنی بر توکن را پیاده سازی کند.

اینکه چگونه اینها را پیاده‌سازی کنید کاملا مطابق میل شماست و دستتان برای هرگونه شخصی سازی باز خواهد بود. توصیه‌هایی که در بخش‌های مختلف به شما ارائه شده همگی از روی تجربه است و پیاده‌سازی آنها به آن شکل، ساختار مرسوم تری داشته و کمتر با باگ مواجه می‌شود.

پیش‌نیازهای فنی (Prerequisites)

نه تنها الان، بلکه احتمالا تا آخر زندگی‌تان به عنوان یک مهندس کامپیوتر، قرار نیست همه پروژه‌هایتان اینطور باشد که از اول همه پیش‌نیازهایش را بلد باشید. خیلی اوقات پروژه‌ای را قبول می‌کنید که قبل از پذیرفتنش حتی یک کلمه از محوریت کلی آن پروژه نشنیده‌اید (اینو برای این گفتیم تا بدونید وقتی شبکه به پروژه ترم یکتون اضافه شد به چنین واقعیتی که در این رشته موجوده فکر می‌کردیم)، ولی یادتون نره که هدف اصلی هر پروژه، بهتر شدن خودتونه.

سرور چیست

منظور از سرور در فاز دو پروژه شما اپلیکیشن سرور است که روی آدرسی خاص در شبکه منتظر اتصال کلاینت‌ها می‌ماند و منظور سخت‌افزار آن نیست.

فرایندی که درون سرور اتفاق خواهد افتاد شامل چند مرحله خواهد بود :

- اتصال کلاینت‌ها به سرور

قبل از هر چیزی کاربر باید به سرور متصل شود تا بتواند درخواست‌های خود را به آن ارسال کند (همان کاری که قبل از ارسال درخواست به سرور در برنامه کلاینت خود انجام می‌دادید). سرور نیز باید منتظر بماند تا یک کلاینت به آن متصل شود و پس از آن به درخواست‌های دریافت شده پاسخ مناسب بدهد.

- دریافت درخواست

پس از اینکه یک کاربر ارتباط خود را با سرور برقرار کرد، درخواستی به آن خواهد فرستاد و سرور باید آن را دریافت کند.

- احراز هویت

سرور برای اینکه درخواست ارسال شده از سمت کاربر را پردازش کند، باید بداند که این درخواست از سمت کدام کاربر ارسال شده است. این کار از سه طریق انجام می‌شود:

- کاربر قصد ثبت‌نام دارد.
- کاربر با نام کاربری و رمز عبور قصد ورود دارد.
- کاربر با استفاده از توکن یکتایی که در هنگام ورود به او تعلق گرفته در سیستم شناسایی خواهد شد. (در این باره در ادامه خواهید خواند)

- پردازش درخواست

پس از اینکه کاربر شناسایی شد (اگر شناسایی نشده باشد، ارتباط به اتمام می‌رسد) باید کارهای مربوط به درخواست کاربر انجام شوند که بسته به نوع درخواست و پارامترهای آن باید تغییرات مناسب را در پایگاه‌داده اعمال کنید (نهراسید 😁 توضیحات مربوط به پایگاه‌داده در ادامه این مستند آمده است) و پاسخ مناسب را آماده کنید.

- ارسال پاسخ

پاسخی که در بخش قبل (پردازش درخواست) آماده شده است، باید به کاربر ارسال شود.

پایگاه داده

بیاید با این شروع کنیم که پایگاه داده یا (Data Base) چیست؟

به زبان ساده پایگاه داده بخشی از پروژه است که اطلاعات مورد نظر ما را به صورت ساختارمند ذخیره می‌کند.

فایده‌اش چیست؟

در این مورد باید باید بگیم که فرض کنید در حال اجرا کردن یک برنامه هستید حال برنامه به صورت اتفاقی crash می‌کند و بسته می‌شود، در این حین شما در حال کار با اطلاعات یک یوزر هستید و اگر برنامه کرش(crash) کند شما این اطلاعات را از دست می‌دهید، مشخصا دوست ندارید این اتفاق برایتان بیفتد به همین خاطر داده‌ها را به اصطلاح داخل پایگاه داده ذخیره می‌کنیم.

ما تو این فاز از شما می‌خواهیم که یک پایگاه داده خیلی ساده به کمک فایل‌ها (txt یا json) بسازید، دقت کنید که شما تعیین‌کننده ساختار این پایگاه هستید و هر چه بهتر این ساختار تعیین شود، برای خودتان و آیندگان بهتر خواهد بود.

کار با فایل

رسیدیم به بخش زیبای کار با فایل. در این قسمت شما باید یک سری فایل بسازید و پایگاه داده خودتان را تشکیل دهید همانطور که اشاره شد این فایل‌ها می‌توانند از هر پسوندی باشند ولی بهتر است که شما از پسوند json یا txt استفاده کنید.

اگر با این دستورات آشنایی ندارید، حتما آنها را جست‌وجو کنید (گوگل همیشه در اولویته!) و این لینک را هم حتما بخوانید.

پیشنهادی

برای اینکه شما بهتر بتوانید ساختار پایگاه داده را طراحی کنید می‌توانید به پوشه ریسورس (Resource) سروری که در فاز یک در اختیارتان گذاشته شد و آن را ایجاد می‌کرد، سر بزنید و ببینید و از چگونگی آن مطلع شوید.

فرمت پیشنهادی ارسال و دریافت اطلاعات

مقدمه

در فاز یک پیام‌های دریافتی از سرور با یک فرمت خاص بود. در این قسمت به توضیح آن فرمت خاص و نحوه‌ی کار با آن فرمت می‌پردازیم.

فرمت پاسخی که سرور فاز یک برای شما ارسال می‌کرد تنها قرارداد بین شما و آن سرور بود تا بتوانید به راحتی بخش‌های مختلف یک پاسخ را از هم جدا کنید و از آنها استفاده کنید.

ساختار این فرمت خاص

در این فرمت دارایی‌های با اهمیتی وجود دارد (این دارایی‌ها همان اطلاعات ما هستند) که طریقه مشخص کردن آنها به شکل زیر است:

key:value

دارایی‌ها در این فرمت بین دو آکولاد ({ }) قرار می‌گیرند :

{key1:value1,key2:value2}

برای مثال رشته بالا دارای دو دارایی است. دارایی‌ها به وسیله کلیدشان قابل دسترسی هستند. یعنی کلیدها شناسه‌های یکتایی هستند که از طریق آنها می‌توان به مقدار آن دارایی‌ها دسترسی داشت. یعنی بخش پراهمیت هر دارایی value آن است و از کلید آن فقط برای دسترسی به آن مقدار پراهمیت استفاده می‌شود. (مشخص است که کلیدها باید یکتا باشند تا دو دارایی باهم تداخل نداشته باشند)

کلیدها از نوع رشته هستند؛ برای مثال:

{“name”:value1}

در رشته بالا کلید "name" وجود دارد که با استفاده از آن می‌توانیم به مقدار value1 دسترسی داشته باشیم.

مقادیر (valueها) می‌توانند چند حالت داشته باشند :

- یک رشته : این رشته‌ها باید بین دو دابل‌کوتیشن‌مارک (") باشند.
- یک عدد: به صورت مستقیم در جلو کلید می‌آید.
- یک boolean: به صورت true یا false در جلو کلید می‌آید.
- یک رشته دیگر با همین فرمت

- یک آرایه: آرایه‌ها با استفاده از براکت ([ ]) نشان داده می‌شوند، اعضای آن با ویرگول از هم جدا شده‌اند و مقادیری که داخل آن است می‌تواند هر یک از valueهای نام برده (حتی یک آرایه دیگر) باشد. نوع این valueها می‌تواند با هم متفاوت باشد، اما معمولا مقادیری که داخل یک آرایه است از یک جنس است.

مثال:

{“name”:"Ehsan","age":20,"married?":false,"friends":[{“name”:"Dana","age":20}],"address":{“city”:"Tehran","street":"N.A."}}

حال تک تک کلیدهای این مثال را بررسی می‌کنیم:

کلید "name" دارای مقدار "Ehsan" است و یک رشته است. کلید "age" کلید بعدی است که دارای مقدار 20 است و یک عدد است. کلید "married?" دارای مقدار false است.
با استفاده از کلید "Friends" می‌توان به یک آرایه دسترسی پیدا کرد که خود حاوی یک رشته دیگر با همین فرمت است که اطلاعات دوستان درون آن است (داخل این آرایه می‌توانست رشته‌های دیگر و یا حتی مقادیر دیگری نیز قرار بگیرد). و در آخر کلید "address" یک رشته با همین فرمت است که نشانی را در اختیار ما قرار خواهد داد و با استفاده از کلیدهای آن ("city","street") می‌توان به اطلاعات مورد نیاز دست پیدا کرد.

نکته‌ای حائز اهمیت وجود دارد و آن این است که ترتیب قرارگیری این دارایی‌ها اهمیتی ندارد و تنها چیزی که در این فرمت اهمیت دارد، دارایی‌ها و کلیدهای آنها است که با استفاده از کلیدها بتوان به اطلاعات مورد نیاز دست پیدا کرد.

نحوه استفاده از این فرمت

برای استفاده از این فرمت باید اطلاعات مختلف را از درون رشته آن استخراج کنید. به این عمل پارس کردن (parsing) می‌گویند. مزیتی که این فرمت دارد این است که بخش‌های مختلف آن با نمادهای مختلفی از هم مجزا شده‌اند (مثلا کل دارایی‌ها درون دو آکولاد است، آرایه با براکت مشخص شده است. کلید و مقدار آن با دو نقطه کنار هم هستند و دارایی‌ها با ویرگول از هم جدا شده‌اند) و این ویژگی جداسازی بخش‌های مختلف و حتی ساختن چنین فرمتی برای اطلاعات را ساده می‌کند.

شما می‌توانید استخراج اطلاعات و ساخت چنین فرمتی را خودتان و با پردازش رشته‌ها انجام دهید (از آنجا که تمام کلیدها و مقادیر آنها را به صورت رشته استفاده می‌کنید ساختن آن ساده خواهد بود و با کمی فکر و خلاقیت می‌توانید parser خودتان را بنویسید و از آن استفاده کنید).

ماهیت‌ها

مقدمه

هر پروژه در یک نرم‌افزار ماهیت‌هایی دارد. هر کدام از این ماهیت‌ها به پیاده‌سازی بخشی از آن نرم‌افزار مربوط می‌شوند و در ویژگی‌هایی (features) که آن نرم‌افزار دارند دخیل‌اند. این ماهیت‌ها، ذخیره‌سازی اطلاعات (که در پایگاه داده با آن آشنا شدید) را سازمان‌یافته‌تر و راحت‌تر می‌کنند.

در ادامه یک ساختار پیشنهادی برای ماهیت‌های این پروژه آمده است. دقت کنید که در این پروژه خدماتی که برنامه شما ارائه می‌دهد مطلوب است و پیاده سازی آنها تنها به خود شما بستگی دارد و می‌توانید آن را هرچقدر که خواستید شخصی‌سازی کنید **(خواسته‌های پروژه را باید برآورده کنید)**.

کاربر (User)

- نام‌کاربری (username):

شناسه هر کاربر نام‌کاربری (username) آن است که باید یکتا باشد. یعنی نباید هیچ دو کاربری، نام‌کاربری یکسانی داشته باشند. نام‌کاربری چیزی است که کاربران را از هم متمایز می‌کند.

- رمز عبور (password):

کاربر، با استفاده از رمز عبور می‌تواند لاگین کند.

- بیوگرافی (bio):

اطلاعات کوتاه در مورد هر فرد که در صفحه پروفایل او نمایش داده می‌شود و توسط کاربر قابل تغییر است.

- فالوورها (followers):

لیستی (آرایه‌ای) از نام کاربری افرادی که یک فرد خاص را دنبال می‌کنند.

- فالووینگ‌ها (followings):

لیستی از نام کاربری‌هایی که یک فرد دنبال می‌کند.

- شناسه توییت‌ها (personal_tweets):

شناسه (id) توییت‌هایی که یک فرد تا به حال زده است.

توییت (Tweet)

- **شناسه (id):**

هر توییت یک شناسه یکتا دارد که آن توییت را از توییت‌های دیگر متمایز می‌کند.

- **نویسنده (author):**

برای آنکه بدانید هر توییت متعلق به کدام یوز است.

- **متن توییت (content):**

بخش اصلی یک توییت که بدون آن یک توییت بی‌معنا خواهد بود.

- **کامنت‌ها:**

لیستی از کامنت‌ها باید برای هر توییت موجود باشد. این نگه‌داری می‌تواند به صورت‌های مختلفی اتفاق بیفتد (برای مثال کامنت‌ها را به صورت یک لیست از فرستنده‌ها و پیام‌ها نگهداری کنید یا برای آن یک ماهیت جدا در نظر بگیرید).

- **تعداد لایک‌ها (likes):**

این بخش تعداد لایک‌های یک توییت را نمایش می‌دهد. واضح است که این مقدار در ابتدا باید صفر باشد.

توکن

توکن چیست؟

توکن قطعه‌ای از داده است که به تنهایی هیچ معنی و کاربردی ندارد، اما همراه با سیستم صحیح توکن‌سازی، به یک بازیگر حیاتی در امنیت برنامه شما تبدیل می‌شود. اعتبارسنجی مبتنی بر توکن با اطمینان از اینکه هر درخواست از یک کاربر با یک رمز خاص (توکن) همراه است که سرور صحت آن را تأیید می‌کند و فقط پس از آن به درخواست پاسخ می‌دهد، کار می‌کند.

توکن چطور کار می‌کند؟

توکن یک رمز است که به صورت **یکتا** برای هر کاربر در زمان ورود به سیستم تولید و برای او ارسال می‌شود. از آن پس کاربر همراه با درخواست‌های خود باید این رمز را نیز به سرور ارسال کند تا سرور با بررسی اینکه این رمز را تولید کرده است یا نه اطمینان حاصل کند که درخواست از طرف یک کاربر معتبر است.

چرا از توکن استفاده کنیم؟

دلیل اینکه از نام‌کاربری درون درخواست‌ها استفاده نمی‌کنیم این است که یک درخواست شامل نام‌کاربری می‌تواند از طرف هر کسی (شاید جناب اخلال‌گر) ارسال شده باشد. مشکلی که استفاده از نام‌کاربری و رمز عبور به جای توکن برای اطمینان از معتبر بودن کاربر دارد این است که احراز هویت هرباره برای یک نام‌کاربری و رمز عبور می‌تواند زمان بر باشد و کارایی سرور را پایین بیاورد.

به همین خاطر به هر کاربر پس از یکبار احراز هویت یک رمز خاص (توکن) داده می‌شود تا در درخواست‌های بعدی از آن به عنوان مدرکی برای اثبات اعتبار خود استفاده کند!

دریافت درخواست از کلاینت و ارسال جواب

در این بخش به توضیح درخواست‌ها و پاسخ‌های مناسب به آنها می‌پردازیم. موارد ستاره دار (*) مواردی هستند که در صورت پیاده‌سازی آنها **نمره امتیازی** به شما تعلق می‌گیرد. دقت کنید که برای بدست آوردن امتیاز آنها، ویژگی مورد نظر باید هم در سرور پیاده‌سازی شده باشد و هم در اپلیکیشن **کلاینت** قابل استفاده باشد و **پیاده‌سازی آن در سرور به تنهایی کافی نیست!**

فرم کلی درخواست و پاسخ

شما باید متناسب با درخواستی که برای شما ارسال شده پاسخ مناسبی برای کاربر ارسال کنید. این پاسخ به هر فرمتی می‌تواند باشد (حتی می‌توانید آن را رمزگذاری کنید 😉). فقط باید در نظر داشته باشید برنامه کلاینت که در فاز یک پیاده‌سازی کرده‌اید باید با این فرم همخوانی داشته باشد. یعنی شما می‌توانید به هر فرمی که خواستید درخواست‌ها را دریافت کنید و پاسخ خود را ارسال کنید اما باید اپلیکیشن کلاینت خود را مطابق با آن تغییر دهید. از این رو پیشنهاد می‌کنیم درخواست‌ها و پاسخ‌های خود را به همان فرمی که اپلیکیشن کلاینت شما با آن کار می‌کند ارسال کنید. (اگر این فرم را فراموش کرده‌اید داک فاز یک یا کدتان را مرور کنید)

پاسخ‌های مشترک

ارور اشتباه ساختار

درخواست‌هایی که به سرور می‌رسد به هر دلیلی (اشتباه از طرف درخواست دهنده، خطا در دریافت اطلاعات یا ...) به درستی به سرور نرسد و ساختار درخواست مطابق خواسته سرور شما نباشد. در این مواقع باید پیغام خطایی برای فرستنده ارسال کنید مبنی بر اینکه درخواست او به درستی به سرور نرسیده است یا اینکه فرمت آن اشتباه است. اشتباه در فرمت می‌تواند هر دلیلی داشته باشد. مثلا درون درخواست لایکی که فرستاده شده است یکی از پارامترهای آن (همان‌هایی که با یک ویرگول از هم جدا می‌شوند) مثل شناسه توییت یا توکن وجود نداشته باشد، دستوری که فرستاده شده است نام اشتباهی داشته باشد (مثلا ابتدای آن بجای like عبارت lick نوشته شده باشد) و یا حتی پارامترهای اضافه‌تر از انتظار فرستاده شده باشد!

عدم اعتبار توکن

به غیر درخواست‌های ساخت کاربر و ورود کاربر بقیه درخواست‌ها باید با یک توکن همراه باشند. اگر این توکن یک توکن معتبر نباشد درخواست نباید انجام بگیرد (در صورت عدم اعتبار توکن حتی بعضی از دستورات قابل اجرا نخواهند بود) و پیغام خطای مناسبی باید به کاربر ارسال شود.

پیغام موفقیت

برخی از درخواست‌ها (مانند لایک کردن یا کامنت گذاشتن) اگر با خطایی روبه‌رو نشوند، در پاسخ آنها تنها باید یک پیغام موفقیت ارسال شود. تصمیم اینکه کدام درخواست‌ها پاسخی از این نوع بگیرند با شماست به شرط اینکه به ویژگی‌هایی که مورد انتظار است لطمه‌ای وارد نشود. شما حتی می‌توانید برای تمام درخواست‌ها پاسخ خاص خود را داشته باشید اما باید دقت کنید برنامه کلاینت شما باید با سرور شما و پاسخ‌های آن سازگاری داشته باشد.

نمایش تغییرات سرور

پیشنهاد می‌کنیم تا تغییراتی را که اتفاق می‌افتد، در خروجی کنسول سرور خود نمایش دهید. این کار در عیب‌یابی برنامه به شما کمک خواهد کرد. برای مثال زمانی که یک کاربر برای سرور درخواستی فرستاد این مسئله را نمایش دهید و پیغام ارسالی به او را نیز نمایش دهید (مانند چیزهایی که سروری که در فاز یک به شما داده شد در کنسول چاپ می‌کرد).

دستوراتی که باید هندل (Handle) شوند

ثبت کاربر جدید

این درخواست نیازی به توکن ندارد و با گرفتن یک نام‌کاربری و یک رمز عبور یک حساب کاربری جدید باید ایجاد کند. (و هر عقل سلیمی حکم می‌کند این حساب کاربری جدید در دیتابیس ثبت شود (دو نقطه دی))

این درخواست ممکن است با خطای زیر همراه باشد :

- نام‌کاربری از قبل موجود باشد.

که اگر این خطا رخ دهد عملیاتی انجام نشده و پیغام خطایی مناسب باید به کاربر ارسال شود.

ورود کاربر به حساب خود

این درخواست نیز نیازی به توکن ندارد. مطابقت نام‌کاربری و رمز عبور باید بررسی شود در صورت تطابق آنها، پیامی مناسب به همراه یک توکن تولید شده برای این کاربر، برای او ارسال شود.

این درخواست ممکن است با خطاهای زیر همراه باشد :

- نام کاربری موجود نباشد.
- رمز عبور اشتباه باشد.
- کاربر از قبل وارد شده باشد.

که اگر این خطاها رخ دهد عملیاتی انجام نشده و پیغام خطای مناسبی باید به کاربر ارسال شود.

ارسال توییت

در این درخواست علاوه بر توکن متن توییت وجود خواهد داشت. پس از اعتبارسنجی توکن، متن ارسال شده باید به عنوان یک توییت جدید به پروفایل کاربر درخواست دهنده اضافه شود و فالورهای او نیز پس از تازه‌سازی می‌توانند این توییت را ببینند.

تازه‌سازی

با این درخواست باید بعد از شناسایی کاربر توییت‌هایی که از قبل ندیده بوده را برای او ارسال کنید. ترتیب آنها اهمیتی ندارد.

اندازه این پاسخ را تا حد معقولی بزرگ فرض کنید تا وقتی تعداد توییت‌ها زیاد شد در دفعه اول پس از ورود مشکلی برای گرفتن توییت‌ها نداشته باشید، البته انتظار نمی‌رود توییت‌های بسیار زیادی را دریافت کنید و اگر تعداد آنها معقول باشد قابل قبول است.

پس از اعتبار سنجی توکن، توییت‌های جدیدی که قبل از این دستور برای کاربر ارسال نشده بود باید برای وی ارسال شود. (خروج کاربر از سیستم را در نظر بگیرید)

لایک

درون درخواست یک شناسه توییت موجود است که باید یکی به تعداد لایک‌های آن افزوده شود. دقت کنید که پس از اینکه تعداد لایک‌ها اضافه شد این توییت باید به عنوان توییت آپدیت شده در هنگام تازه‌سازی به مخاطبان آن ارسال شود.

این درخواست ممکن است با خطاهای زیر همراه باشد :

- توییتی با این شناسه وجود نداشته باشد
- لایک تکراری* (این یک Cross Reference است D:)

که اگر این خطاها رخ دهند پیغام خطایی مناسب باید به کاربر ارسال شود.

کامنت

درون درخواست علاوه بر توکن دو چیز دیگر موجود است :

- شناسه توییتی که قرار است کامنت بر روی آن گذاشته شود.
- متن کامنت

اگر توییت مورد نظر وجود داشت، آن کامنت باید به نام کاربر ارسال کننده (که از توکن مشخص می‌شود) برای آن توییت ثبت شود.

توجه کنید که نشان دادن کامنت‌ها در برنامه کلاینت امتیازی بوده است، اما فرستادن آنها به عنوان جواب برای کلاینت یک امر اجباری است.

سرچ

در این درخواست علاوه بر توکن یک نام کاربری هم برای سرور ارسال می‌شود که باید در میان کاربران موجود روی سرور آن را جست‌وجو کنید. در صورت وجود این کاربر باید یک پیغام شامل موفقیت عملیات به همراه پروفایل کاربر مورد نظر به عنوان پاسخ ارسال شود. اگر هم این کاربر وجود نداشت پیغام خطای مناسبی باید به کاربر ارسال شود.

این پروفایلی که به کاربر ارسال می‌شود باید دارای بخش‌های زیر باشد:

- نام کاربری
- بیوگرافی
- تعداد following و follower
- تمام توییت‌های کاربر

فالو

در این درخواست علاوه بر توکن که از آن باید برای شناسایی کاربر استفاده کنید، نام‌کاربری دیگری نیز وجود دارد که همان کاربری است که باید دنبال (فالو) شود. این دستور کاربر مورد نظر را به پروفایل درخواست دهنده اضافه می‌کند. دقت کنید که از این به بعد باید توییت‌های کاربر تازه دنبال شده درون تایم لاین نمایش داده شود.

این درخواست ممکن است با خطاهای زیر همراه باشد :

- چنین نام کاربری وجود ندارد.
- کاربر پیش از این فالو شده است.

که اگر این خطاها رخ دهند، پیغام خطایی مناسب باید به کاربر ارسال شود.

آنفالو

در این درخواست علاوه بر توکن که از آن باید برای شناسایی کاربر استفاده کنید، نام‌کاربری دیگری نیز وجود دارد که همان کاربری است که باید آنفالو شود. این دستور کاربر مورد نظر را از پروفایل درخواست‌دهنده حذف می‌کند. دقت کنید که از این به بعد توییت‌های کاربر آنفالو شده نباید درون تایم‌لاین نمایش داده شود.

تغییر بیو

این درخواست علاوه بر توکن پارامتر دیگری دارد که همان متن جدید بیو است. پس از تغییر دادن این قسمت در پروفایل باید پیغام موفقیت به کاربر ارسال شود.

لاگ اوت

این درخواست تنها شامل توکن است. پس از یافتن کاربر درخواست دهنده از روی توکن، توکن وی غیرفعال شده و پیغام موفقیت برای کاربر ارسال می‌شود.

دقت کنید اگر کاربر لاگین نکرده باشد، توکن وی موجود نخواهد بود و در همان مرحله اعتبارسنجی باید پیغام خطا برای درخواست دهنده ارسال شود.

توییت پروفایل

این درخواست پروفایل درخواست دهنده را برای او ارسال می‌کند. بخش‌هایی که باید در پاسخ به این درخواست باشد مانند بخش‌های موجود در سرچ است.

تغییر رمز عبور*

این درخواست علاوه بر توکن که به وسیله آن کاربر هدف تشخیص داده می‌شود، شامل یک رشته (حداکثر شانزده کاراکتری) خواهد بود که همان رمز عبور جدید است. پس از تغییر رمز عبور پیغام موفقیت برای کاربر ارسال خواهد شد.

این درخواست ممکن است با خطای زیر همراه باشد :

- رمز عبور طولانی تر از شانزده کاراکتر باشد.

که اگر این خطا رخ دهد عملیاتی انجام نشده و پیغام خطایی مناسب باید به کاربر ارسال شود.

توجه کنید که پیاده‌سازی این دستور هم باید در سرور و هم باید در کلاینت باشد تا نمره آن را دریافت کنید!

حذف توییت *

با استفاده از این درخواست کاربر می‌تواند یکی از توییت‌های خود را حذف کند. این درخواست شامل یک توکن و یک شناسه توییت خواهد بود. این دستور، یک دستور تازه است که در فاز دوم (همین فاز) معرفی شده‌است. مانند تمام درخواست‌های دیگر شما می‌توانید هر فرمتی برای این درخواست در نظر بگیرید. پیشنهاد ما فرم زیر است:

delete <token>, <tweet_id>

این دستور توییت را از پروفایل کاربر حذف می‌کند. توجه کنید کاربرانی که فالور این کاربر هستند پس از تازه‌سازی دیگر نباید این توییت را ببینند. پس از اینکه تغییرات اتفاق افتاد، پیغام موفقیت باید برای کاربر ارسال شود.

این درخواست ممکن است با خطاهای زیر همراه باشد :

- توییت مربوط به کاربر درخواست دهنده نباشد.
- چنین توییتی وجود نداشته باشد.

که اگر این خطاها رخ دهد عملیاتی انجام نشده و پیغام خطایی مناسب باید به کاربر ارسال شود.

توجه کنید این درخواست هم باید در سرور پیاده شود و تغییرات آن هم در اپلیکیشن کلاینت شما باید اعمال شود. برای مثال باید گزینه حذف توییت در منوی برنامه شما بوجود بیاید. همینطور باید فکری برای حذف توییت از تایم‌لاین دیگر کاربران بکنید. 😉 (اگر ریتوییت(مورد بعدی) را هم پیاده‌سازی می‌کنید، باید حواستان به پاک شدن توییت ریتوییت شده (😢) نیز باشد)

ریتوییت *

این درخواست به این معنی است که کاربر می‌خواهد توییت یکی دیگر از کاربران با ذکر نام آن کاربر (اگر کپی‌رایت رعایت نشود پیگرد قانونی خواهد داشت 😎) در پروفایلش وجود داشته باشد. همانند دستور قبل هر فرمتی را می‌توانید برای این درخواست در نظر بگیرید. ما فرمت زیر را پیشنهاد می‌دهیم:

retweet <token>, <tweet_id>

اینکه یک توییت، ریتوییت شده‌ی یک توییت دیگر است نیز باید در تایم‌لاین نمایش داده شود. شما می‌توانید هر طور که دوست دارید این موضوع را به اطلاع کاربر برسانید. ما نمایش زیر را پیشنهاد می‌دهیم:

A.kousheshi (cnn) 1295
Hi This is a tweet text!!
Likes: 400k, comments: 16

که cnn صاحب اصلی توییت است. A.koosheshi کسی است که آنرا ریتوییت کرده و 1295 شناسه توییت cnn است. بهتر است زمانی چنین فرمتی برای کاربر نمایش دهد که او cnn را دنبال نمی‌کند و این توییت توسط a.koosheshi به دست او رسیده. البته اگر این اتفاق نیفتد ایرادی ندارد و قابل قبول است.

این درخواست ممکن است با خطای زیر همراه باشد :

- توییت مورد نظر وجود نداشته باشد.

که اگر این خطا رخ دهد عملیاتی انجام نشده و پیغام خطایی مناسب باید به کاربر ارسال شود.

توجه کنید زمانی امتیاز این بخش را دریافت می‌کنید که قابلیت ریتوییت هم در سرور شما و هم در اپلیکیشن کلاینت شما پیاده‌سازی شده باشد. یعنی برای این دستور شما باید کلاینت خود را تغییر دهید (علاوه بر پیاده سازی آن در سرور) تا بتوانید امتیاز آن را دریافت کنید.

همینطور توجه کنید اگر حذف توییت را پیاده سازی کرده‌اید، پس از حذف یک توییت ریتوییت‌های آن نیز دیگر وجود نخواهند داشت.

مرتب سازی بر اساس زمان *

در زمان ارسال درخواست تازه‌سازی، توییت‌هایی که توسط کاربر از قبل دیده نشده است، باید برای او فرستاده بشود. اگر این توییت‌ها را بر اساس زمان ارسال آنها مرتب کنید (آخرین توییت موجود در لیست تازه‌سازی باید جدیدترین باشد) از نمره بیشتری (امتیازی) برخوردار خواهید شد. دقت کنید برای دریافت امتیاز این بخش، در زمان تازه‌سازی توییت‌های موجود در تایم‌لاین باید از جدیدترین به قدیمی‌ترین مرتب شوند. (توییت‌ها هم در پاسخ سرور باید مرتب شده باشند و هم در تایم‌لاین)

لایک تکراری*

زمانی که کاربری اقدام به لایک کردن دوباره یک توییت می‌کند شما می‌توانید یکی از دو واکنش زیر را داشته باشید :

- پیغام خطایی مبنی بر تکراری بودن لایک او برای او ارسال کنید.
- یا اینکه لایک او را از توییت بردارید و از تعداد لایک‌های توییت کم کنید. (کاربر در آینده باز می‌تواند آن توییت را لایک کند)

(خواستان باشد که هر دو مورد بالا جزو موارد امتیازی است و اگر بخواهید می‌توانید هیچکدام از واکنش‌ها را نشان ندهید و با هر درخواست لایک، یکی به لایک‌های آن توییت اضافه کنید)

تازه‌سازی‌های بزرگ *

در بخش تازه‌سازی گفته شد نیازی نیست برای تازه‌سازی، تعداد زیاد توییت‌ها را هندل کنید و فرض شده بود که تعداد آنها معقول است. حال اگر ساز و کاری پیاده‌سازی کنید تا هر تعداد توییت دیده نشده وجود داشته باشد (برای مثال کاربر ده سال از توییتر استفاده نکرده و پس از ده سال قصد تازه‌سازی دارد) بدون مشکل بتواند دریافت شود از نمره امتیازی برخوردار خواهید شد.

توجه کنید که این ساز و کار باید هم در کلاینت و هم در سرور موجود باشد تا بتوانید نمره آن را دریافت کنید.

سرچ پیشرفته کاربر *

در جست‌وجو بین کاربران تنها کسی را که نامش کاملا با عبارت جست‌وجو شده برابر است نشان داده نشود. اگر عبارتی (مثلا "al") جست و جو شود تمام کاربرانی که این عبارت را دارند (برای مثال ali008, alireza.babaz, nahal) باید به کاربر نشان داده شوند و او بتواند از بین آنها انتخاب کند. مانند قسمت‌های قبلی پیاده سازی فرمت درخواست و پاسخ و همینطور نحوه نشان دادن نتیجه جست‌وجو با شما خواهد بود.

سرچ توییت *

این قسمت هم یک بخش جدید است که باید جایی برای آن در منوهای برنامه خود باز کنید. زمانی که کاربر عبارتی را در میان توییت‌ها جست‌وجو می‌کند، برنامه شما باید تمام توییت‌هایی که شامل هشتگی با آن عبارت هستند را به کاربر نمایش دهد (حتی توییت‌های کاربرانی که توسط درخواست‌دهنده فالو نشده‌اند).

برای مثال اگر یک کاربر عبارت CE_SUT را جست‌وجو کند باید تمام توییت‌های شامل #CE_SUT به او نمایش داده شود.

نوتیفیکیشن *

می‌خواهیم برای توییترمان نوتیفیکیشنی با سبک و سیاقی خاص طراحی کنیم.

اول از همه باید بدانید که مسئله ارسال نوتیف که در برنامه‌های خود مشاهده می‌کنید، متفاوت با چیزی است که اینجا ارائه می‌شود؛ اما به هر حال سبک و مدل اینجا هم، خالی از لطف نیست:).

روند طراحی بدین گونه است که در منوی تایم‌لاین، دو مورد باید اضافه شود:

- علاقه‌مندی‌ها(Favorites)
- اعلان‌های من(My Notifications)

علاقه‌مندی‌ها باید مجموعه‌ای از کلمات خاص به سبک قابل تبدیل به هشتگ باشد. به عبارت بهتر، شما هشتگ‌هایی را که در مورد آنان مایل به دریافت توییت هستید، در علاقه‌مندی‌ها اضافه می‌کنید. سعی کنید تعداد آنان را محدود در نظر بگیرید(حداکثر 5 عنوان).

دقت کنید که علاقه‌مندی، در کلاینت انتخاب می‌شود و آن را به سرور می‌فرستید.

حال اگر سرور توییتی از کسی دریافت کند که این هشتگ مورد علاقه شما را داشته باشد، آن را در قسمت اعلان‌های شما ذخیره می‌کند. مثلا به فرمت زیر:

A.kousheshi 1295

Hi This is a tweet text!!#Your_Favorite

در واقع این توییت در بخش اعلان کاربری که چنین علاقه‌مندی‌ای (Your_Favorite) دارد، ذخیره می‌شود و به عنوان اعلان، به کاربر منتقل می‌شود. **تعداد اعلان‌هایی که به کاربر می‌دهیم، 10 تای آخر است.** یعنی اگر کاربری دارای بیش از 10 اعلان باشد، قبلی‌ها حذف شده و صرفا 10 تای آخر به کاربر منتقل می‌شود.