# به نام خدا


# درس برنامه‌سازی پیشرفته

## فاز سوم پروژه

**دانشکده مهندسی کامپیوتر**

دانشگاه صنعتی شریف

نیم سال دوم ۰۲ ـ ۰۱

---

استاد:

**دکتر محمدامین فضلی**

مهلت ارسال:

**چک پوینت فاز سوم: ۳۱ خرداد**

**فاز سوم: ۱۰ تیر**

**ساعت ۲۳:۵۹:۵۹**

مسئول پروژه:

**آرمان بابایی**

مسئول فاز سوم:

**محمدمهدی قیدی**

طراحان فاز سوم:

**رضا صومی، پرهام رضایی، سپهر میزانیان، عماد امام‌جمعه، امیرمحمد درخشنده، حسین مسعودی، رضا حیدری، نیکان واسعی، مازیار شمسی پور**

مسئولین تنظیم مستند:

**آرمان بابایی، محمدمهدی قیدی**

# فهرست

## نکات قابل توجه

- پس از اتمام این فاز، در گیت خود یک تگ با ورژن "v3.0.0" بزنید. در روز تحویل حضوری این tag بررسی خواهد شد و کدهای پس از آن نمره‌ای نخواهد گرفت. برای اطلاعات بیش‌تر در مورد شیوه ورژن‌گذاری، می‌توانید به این لینک مراجعه کنید. البته برای این پروژه صرفا رعایت کردن همان ورژن گفته شده کافیست، اما خوب است که با منطق ورژن‌بندی هم آشنا بشوید.
- در روز تحویل حضوری مشارکت تمام اعضای تیم در پروژه بررسی خواهد شد و در صورت عدم مشارکت بعضی از اعضا، نمره‌ی ایشان برای آن فاز پروژه "صفر" لحاظ می‌گردد. مشارکت، با توجه به commit های افراد تیم در مخزن گیت‌هاب پروژه بررسی می‌شود.
- توجه کنید که به دلیل نزدیک بودن به مهلت ارسال نمرات، امکان تاخیر برای فاز سوم وجود ندارد.
- در صورت کشف تقلب از هریک از تیم‌ها، برای بار اول منفی نمرهٔ آن فاز برای آن تیم ثبت می‌شود و برای بار دوم، نمرهٔ منفی کل پروژه برای تیم لحاظ خواهد شد که معادل مردود شدن در درس است.
- مشابه فاز قبل لازم است برای دریافت تمپلیت ایشوی فاز سوم، پرونده‌ی موردنظر را از تمپلیت پروژه ذخیره کنید.

# مقدمه

همانطور که می‌دانید، در دو فاز قبلی بدنه‌ی اصلی مورد نیاز برای اجرای بازی روی یک کامپیوتر را پیاده سازی کردیم؛ یعنی بخش های مربوط به منطق و گرافیکی بازی که برای کامل بودن بازی کافی هستند. پس در این بخش می‌خواهیم چه کنیم؟ قرار است قابلیت‌هایی را به بازی‌مان اضافه کنیم تا بشود بر بستر اینترنت هم بازی را اجرا و با بقیه بازیکنان از راه دور بازی کرد.

# چک پوینت اول

## معماری Server-Client

معماری کارخواه ـ کارگزار یا Server-Client و معماری همتا به همتا Peer-to-Peer معروف ترین معماری های شبکه‌های کامپیوتری هستند. در معماری Server-Client کاربران معمولی کلاینت نامیده شده و هر کدام از آن‌ها درخواست‌هایی را برای سرور ارسال می‌کنند. سرور به منابع و اطلاعات اصلی برنامه دسترسی دارد و پردازش‌های اصلی داده‌ها در آن انجام شده و در نهایت نتیجه به شکل مناسبی به کلاینت اطلاع داده می شود. برای این فاز پروژه، توصیه می‌شود از معماری کلاینت سرور استفاده نمایید. به این شکل که سرور، اطلاعات اصلی نظیر بازی‌های در حال انجام و... را در اختیار داشته و بسته به درخواست‌هایی که برای آن ارسال می‌شود، پاسخ مناسب را برای هر کلاینت ارسال می‌کند. به بیان دیگر، بخش عمده منطق برنامه باید در سمت سرور رسیدگی شود. برای آشنایی بیشتر با این معماری و نحوه پیاده‌سازی آن، داک شبکه را مطالعه کنید.

## بازی همزمان

بازی شما تاکنون بدین صورت بود که شما از طریق یک کامپیوتر به جای افراد مختلف بازی می‌کردید و هر کدام از این افراد، تنها می‌توانستند در نوبت خودشان حرکت کنند. به این نوع بازی‌ها، بازی‌های turn-based می‌گویند.

در این فاز، جدای از اینکه (چنانچه پیشتر گفته شد) دستگاه‌های گوناگون باید بتوانند به یک بازی متصل شوند، بلکه دیگر بازی شما مانند سابق turn-based نخواهد بود. به این صورت که هرچند ممکن است شما در لایه‌های زیری هر حرکت را در همان لحظه اعمال نکنید و تاخیری میان از درخواست و انجام حرکت ایجاد شود، لازم است این تاخیر از چشم کاربر پوشیده شود. در فاز سوم، افراد مختلف باید به طور همزمان بتوانند از دستگاه‌های مختلف بازی کنند، در نتیجه در همان لحظه‌ای که شخص ۱ درحال بازی است، اجازه حرکت برای شخص ۲ نیز باید وجود داشته باشد. روش‌های گوناگونی برای این پیاده سازی وجود دارد که می‌توانید آنها را در اینترنت جست و جو کنید.

## احراز هویت

به داک احراز هویت مراجعه کنید.

# توضیح بخش‌های مختلف فاز سوم

## مپ

در فاز۲ یک گرافیک برای منوی مپ خود طراحی کردید، در این فاز شما باید به تکمیل این بخش بپردازید تا بتوانید مپ خود را با دیگران به اشتراک بگذارید.
و به طور دقیق‌تر در این بخش قرار است امکانات جدیدی به مدیریت مپ‌هایتان بیفزایید.
دو قابلیت به آنچه در فاز قبل داشتید، افزوده شده است.

- قابلیت اول: ذخیره‌سازی متعدد
قابلیت ذخیره کردن بیش از یک مپ را نیز در برنامه خود ایجاد کنید. در این فاز باید بتوانید تعداد بیشتر از یک مپ را نیز ذخیره کنید و منو یا بخش لازم برای انتخاب مپ از بین آن‌ها را نیز فراهم کنید.
نکته: انتظار می‌رود هنگام ذخیره سازی مپ تکراری نبودن نام آن چک شود.

- قابلیت دوم: اشتراک گذاری مپ
در این بخش شما می‌توانید مپ های ذخیره شده خود را با دیگر کاربران به اشتراک بگذارید. برای طراحی دو راه پیش رو دارید:

- راه ۱ : ارسال peer to peer
باید بتوانید با انتخاب یک کاربر و مپ دلخواه از بین مپ‌های ذخیره شده‌تان، آن را برای او ارسال کنید. برای این کار می‌توانید یک منو مخصوص این کار طراحی کنید یا به سلیقه خود این قابلیت را در بخشی از برنامه‌تان فراهم کنید. گیرنده باید پیام مپ‌ها را یا در منویی مخصوص یا به صورت نوتیفیکیشن دریافت کند. او باید بتواند مپ را رد کرده و یا آن را به مپ‌های ذخیره شده خود بیفزاید.
نکته ۱ : مشابه قابلیت ۱ ، انتظار می‌رود عدم تکرر نام مپ‌ها چک شود.
نکته ۲ : میتوانید از منوی مشترکی برای ارسال و دریافت مپ‌ها استفاده کنید.

- راه ۲ : لازم است که یک صفحه طراحی کنید که کاربران در آن مپ‌های خود را برای عموم به اشتراک بگذارند. شما می‌توانید از بین مپ‌های ذخیره شده خود یک مپی که تاکنون در صفحه قرار نداده اید را انتخاب کرده و قرار دهید. همچنین می‌توانید با مراجعه به صفحه و مشاهده مپ‌های سایر کاربران، مپ مدنظر خود را انتخاب و ذخیره کنید.

**نکات:**

- انتظار می‌رود کاربر نتواند مپی که یک بار ذخیره کرده است را برای بار دوم ذخیره کند.

- انتظار می رود مپ‌های ذخیره شده و مپ‌های ارسال شده خود کاربر از سایر مپ‌ها به شیوه ظاهری مجزا شده باشد.
- اگر کاربر مپی را در صفحه قرار داده و به هر دلیلی دیگر آن را ذخیره ندارد، این نباید موجب پاک شدن مپ از روی صفحه شود.
- هدف صفحه‌ای مشابه کامیونیتی مپ‌های بازی‌هایی مثل minecraft است.

## چت

در این فاز شما باید یک چت‌روم طراحی کنید. گرافیک و ظاهر چت‌روم به دلخواه شما خواهد بود (هر چه زیباتر بهتر) اما چت روم شما، باید از قابلیت‌های زیر پشتیبانی کند:

- چت‌روم شما باید ۳ بخش مختلف داشته باشد:
  - public chat
  - private chat
  - rooms
- در بخش public chat تمام بازیکنان میتوانند بدون محدودیت با یکدیگر صحبت کنند و پیام بفرستند.
- در بخش private chat هر بازیکن میتواند بازیکن دیگری را بر حسب نام او در بازی جستجو کند و با او وارد یک چت خصوصی شود. واضح است که چت این دو بازیکن باید از دید سایر بازیکن‌ها، پنهان بماند.
- در بخش rooms هر بازیکن می‌تواند تعدادی room با نام‌های مختلف ایجاد کند و در هر room بازیکنان دیگری را که می‌خواهد (با جستجو کردن نام آن‌ها) به آن room اضافه کند تا در نهایت، بازیکنان یک room بتوانند با یک‌دیگر چت کنند (مشابه عملکرد چت گروهی در اپلیکیشن‌های رایج)
- برای هر بازی که ساخته می‌شود، یک room هم متناظر آن ساخته می‌شود. تمام اعضای بازی می‌توانند در این room مشارکت کنند و سازنده‌ی آن، همان سازنده‌ی بازی است.
- هر بازیکن باید توانایی انجام قابلیت‌های زیر روی پیام‌های خودش (منظور پیام‌هایی است که ارسال‌کننده آن‌ها، خود شخص است) را داشته باشد:
  - قابلیت ادیت کردن پیام
  - قابلیت حذف پیام فقط برای خود شخص
  - قابلیت حذف پیام برای خود و طرف مقابل (در public chat و یا در یک room پیام باید برای همه بازیکن ها پاک شود)
- هر پیام باید حاوی این اطلاعات باشد:
  - نام فرستنده
  - زمان ارسال پیام در قالب hh : mm

- متن پیام
- آواتار شخص فرستنده
- علامتی برای نشان دادن اینکه پیام ارسال شده است
- علامتی برای نشان دادن اینکه پیام دیده یا seen شده است(مشابه اپلیکیشن های معروف)

- هر بازیکن باید توانایی reaction دادن به پیام های داخل چت را داشته باشد (به پیام‌های خود نیز می‌توان reaction داد)، بدین صورت که از میان چند ایموجی پیش فرض دلخواه (انتخاب تعداد و طرح ایموجی‌ها بر عهده خودتان است، فقط توجه کنید که حداقل ۳ و حداکثر ۶ ایموجی میتوانید انتخاب کنید)، باید بتوان فقط یک ایموجی برای reaction انتخاب کرد. همچنین بعد از انتخاب reaction تصویر آن باید کنار پیام نشان داده شده و باقی بماند.
- دقت کنید که تمام اطلاعات (history) مربوط به چت های مختلف، باید پس از بستن بازی و باز کردن دوباره ی آن، حفظ شود. برای این کار می‌توانید از ذخیره سازی اطلاعات مربوطه در فایل و خواندن مجدد اطلاعات از فایل، استفاده کنید.

## جدول امتیازات

به جدول امتیازاتی که در فازهای قبلی پیاده کرده‌اید باید چند ویژگی جدید اضافه کنید:

- نمایش آنلاین یا آفلاین بودن افراد
- آپدیت شدن خودکار اطلاعات جدول در صورت تغییر (مثلا اگر امتیاز یک نفر تغییر کند جدول باید در لحظه آپدیت شود- یا یک نفر که آفلاین بوده است آنلاین شود)
- قابلیت مشاهده ویدئو آخرین بازی ۱۰ نفر برتر (امتیازی)

## دوستی

در این فاز باید این قابلیت را پیاده سازی کنید که هر کاربر بتواند به کاربران دیگر درخواست دوستی بدهد و در صورت موافقت طرف مقابل، به لیست دوستان کاربر درخواست‌دهنده و کاربری که درخواست را قبول کرده است اضافه می‌شود. قابلیت ارسال درخواست دوستی باید حداقل در حالت‌های زیر وجود داشته باشد:

- سرچ کردن نام کاربری یک بازیکن، مشاهده پروفایل وی و ارسال درخواست دوستی
- هر منویی که قابلیت مشاهده پروفایل در آن وجود دارد (به طور مثال جدول امتیازات)

توجه کنید هر کاربر می‌تواند حداکثر ۱۰۰ دوست داشته باشد. لیست دوستان هر فرد در منوی پروفایل باید قابل مشاهده باشد. لیست درخواست‌های دوستی در حالت انتظار (تایید یا رد) نیز در منوی پروفایل باید وجود داشته باشد.

## لابی

برای اینکه بازیکنان بتوانند با فرد دیگری بازی کنند نیاز به یک منوی جدید به نام لابی داریم. بازیکنان برای بازی یا باید خود یک درخواست بازی ثبت کرده و منتظر رقیب بمانند یا می‌توانند یکی از گروه‌هایی که قبلا درخواست داده‌اند را انتخاب کند و به گروه بازی آن‌ها بپیوندد. کسی که درخواست بازی را ثبت می‌کند باید همان ابتدا مشخص کند که بازی ۲، ۳ یا ... نفره خواهد بود (تعداد را باید مشخص کند.) پس از ثبت درخواست یک بازی به لیست بازی‌های داخل این منو اضافه می‌شود که اطلاعات آن از قبیل ظرفیت بازی و nick-name بازیکنان آماده برای آن بازی مشخص است. از این پس بازیکنان دیگر نیز می‌توانند به این بازی اضافه شوند و هنگامی که ظرفیت بازی تکمیل شود سرور برای آن‌ها یک بازی جدید می‌سازد و بازی شروع می‌شود. همچنین کسی که بازی را ساخته است(admin) باید بتواند بازی را زودتر از اینکه ظرفیت مشخص شده کامل شود نیز شروع کند تنها با این شرط که حداقل یک نفر دیگر در گروه مربوط به بازی او باشد. نکته‌ی دیگری که باید توجه داشته باشید این است که هر بازیکنی که داخل یک گروه بازی اضافه می‌شود باید بتواند قبل از شروع بازی با زدن دکمه‌ای از گروه آن بازی خارج شود و همچنین اگر admin یک گروه بازی از آن گروه خارج شود نفر بعدی داخل گروه admin می‌شود و در صورتی که بازیکن دیگری در گروه نباشد گروه بازی بسته می‌شود و از لیست داخل منوی لابی آن بازی حذف می‌شود. به طور کلی هر بازیکنی که از یک گروه بازی انصراف می‌دهد یا اتصالش قطع می‌شود باید از گروه بازی حذف شود و اگر admin باشد نفر بعدی آن گروه admin شود. از قابلیت‌های دیگری که باید این منو داشته باشد این است که باید برای لیست گروه‌های بازی که در این منو نمایش داده می‌شود یک دکمه refresh قرار دهید که هر بار که بازیکن این دکمه را فشار می‌دهد لیست بازی‌ها رفرش شوند به صورت رندوم ۱۰ گروه بازی به بازیکن نمایش داده شود.

نکات:

- افراد هر گروه بازی بتوانند داخل چت روم مخصوص به گروه بازی با هم چت کنند.
- برای هر گروه بازی هنگام ساخته شدن یک شناسه یکتا ساخته شود و بازیکنان بتوانند بر اساس شناسه داخل این منو، بازی مطلوب خود را پیدا کنند. در واقع به کمک این شناسه بازیکنان هر گروه بازی می‌توانند داخل چت روم گلوبال از بازیکنان دیگر دعوت کنند که با آن‌ها بازی کنند. به این صورت که بازیکنان دیگر بتوانند با سرچ آن شناسه در این منو به گروه بازی اضافه شوند.
- admin هر گروه بازی بتواند گروه را private یا public کند. به این صورت که در صورتی که private باشد تنها افراد با شناسه بتوانند به این گروه وارد شوند و

همچنین اطلاعات گروه، داخل لیست گروه‌های بازی داخل لابی نمایش داده نشود و اگر از private به public وضعیت گروه تغییر داده شود از آن به بعد افراد دیگر نیز بتواند بدون شناسه و از طریق لیست گروه‌های بازی به بازی اضافه شوند. کاربرد اصلی private کردن گروه به این منظور است که admin بتواند شناسه را به دوست‌های خود داخل چت روم private بفرستد و تنها با آن‌ها بازی کند.

- برای جلوگیری از به وجود آمدن گروه‌های بازی اضافی، اگر داخل یک گروه بازی به مدت ۵ دقیقه بازیکنی وارد نشود سرور باید گروه بازی را ببندد تا از به وجود آمدن گروه‌های اضافه‌ای که قصد بازی ندارند، جلوگیری شود.

## ادامه بازی ناتمام (امتیازی)

اگر در میان یک بازی ارتباط یک بازیکن با سرور قطع شود به مدت محدودی(مثلا حداکثر دو دقیقه) فرصت دارد تا دوباره به بازی متصل شود. به این صورت که اگر قبل از دو دقیقه دوباره به سرور متصل شود، پس از لاگین به ادامه بازی می‌پردازد و در غیر این صورت از بازی حذف می‌شود و در صورت دو نفره بودن آن بازی نفر دیگر برنده اعلام می‌شود و در غیر این صورت بازی بین نفرات باقی‌مانده ادامه می‌یابد با این تفاوت که دیگر نوبتی برای نفر حذف‌شده در نظر گرفته نمی‌شود.

# پخش تلویزیون (امتیازی)

تلویزیون بخش جدیدی است که در این فاز اضافه شده است. این بخش دارای دو قسمت پخش زنده ی بازی ها و بازپخش بازی های قبلی کاربر می باشد. توضیحات این دو بخش به این صورت است:

## پخش زنده‌ی بازی‌ها

هدف این قسمت به اشتراک‌گذاری برخط بازی‌هاست و در واقع این قابلیت را به وجود می‌آورد که بتوان به صورت زنده بازی را تماشا کرد. هر کاربری که در حال بازی کردن است می‌تواند بازی خود را به صورت زنده استریم کند. یک روش پیشنهادی برای پیاده‌سازی این بخش ثبت گزارش (log) است. در این روش ابتدا حالت اولیه بازی (شامل وضعیت اولیه نقشه و بازیکن‌ها و ...) ذخیره می‌شود و سپس تمامی حرکات بعدی ذخیره می‌شوند. به عبارت دیگر پس از هر اتفاق، جزئیات آن در قالب مشخصی ذخیره می‌شود. سپس چون حالت اولیه بازی را می‌دانیم برای هر کاربری که قصد مشاهده‌ی پخش زنده را داشته باشد، یک نمونه از بازی را می سازیم و تمامی حرکات را براساس این گزارش‌ها اجرا می‌کنیم تا به حالت فعلی بازی برسیم. به همین شکل اگر به طور مداوم با استفاده از ادامه‌ی گزارش‌ها بازی را برای کاربر بسازیم، پخش زنده ادامه پیدا می کند و با تمام شدن گزارش‌ها بازی خاتمه پیدا می کند و استریم متوقف می‌شود.

## بازپخش بازی‌های قبلی

در این بخش کاربر می‌تواند بازی‌های قبلی خود را دوباره مشاهده و بررسی کند.در اینجا نیز می توانید از روش پیشنهادی ثبت گزارش ها (log) بازی‌های قبلی خود را ذخیره کنید و در هر حرکت، به حرکت‌های قبلی و بعدی در بازی دست یابید. همچنین این حرکت‌ها می‌توانند به صورت خودکار نیز پشت سر هم پخش شوند. کاربر می‌تواند سرعت پخش این حرکات را تنظیم کند.

**توجه کنید** که در هر دوی این بخش‌ها room مربوط به بازی هم با همان ترتیب زمانی تکرار می‌شود. علاوه بر این لازم است جرکت به جلو و عقب در زمان هم برای این دو قسمت در نظر گرفته شوند.

## پایگاه داده (امتیازی)

شما میتوانید برای ذخیره‌سازی اطلاعات مورد نیازی که قبلا در فایل ذخیره میکردید، از پایگاه داده‌های دیگر نیز استفاده کنید. برای اطلاعات بیشتر به داک پایگاه داده مراجعه کنید.

# Rest و Fast Communication

## Rest

در این بخش میخواهیم با مفاهیمی مثل Rest و Request و API آشنا شویم. این مفاهیم متاسفانه با وجود اهمیت بسیار بالایی که دارند در درس AP به آنها یا پرداخته نمی شود یا خیلی کم پرداخته می‌شود. به همین دلیل تصمیم گرفتیم در این بخش شما را با این مفاهیم آشنا کنیم و اگر بیشتر علاقمند بودید چند لینک در انتهای بخش گذاشته‌ایم که می‌توانید با مطالعه آن ها مطالب مفیدتری در این مورد یاد بگیرید.

بگذارید با یک مثال ساده شروع کنیم. فرض کنید شما مسئول فروش یک شرکت هستید و هر روز برای شما چندین سفارش از خریدارهای مختلف می‌آید و شما باید این سفارش‌ها را بررسی کنید و در صورت موافقت به آن‌ها جواب بدهید. شما برای این کار یک سیستم جامع طراحی می کنید، در این سیستم تعدادی جایگاه برای فروش قرار دارد، هر خریدار می‌تواند فرم خرید را پر کند و به یک جایگاه تحویل بدهد و در صورت موافقت شرکت محصول مورد نظر به او تحویل داده می‌شود. در مثالی که زدیم جایگاه ها مانند هاAPI عمل می‌کنند، شرکت شما مانند Server و مشتریان مانند Client عمل می‌کنند. همچنین هر فرمی که پر می‌شود مانند یک Request است. در نهایت کل این سیستم و قوانین رد و بدل کردن درخواست‌ها و جواب درخواست‌ها را یک Protocol می‌نامیم.

مثالی که بالا زدیم برای آشنایی شما با برخی از مفاهیم پایه ای شبکه بود حال به سراغ Rest می رویم. Rest در واقع یک پروتکل است که به شما می‌گوید که Request ها و هاResponse باید به چه فرمتی باشند و از چه اصولی پیروی کنند. در ادامه چند مورد از اصول(Rest (Principles را بیان می‌کنیم.

- Uniform interface: تمام هایAPI Request برای یک Resource باید به یک شکل باشند، بدون توجه به اینکه از کجا می‌آید.
- Client-server decoupling: در طراحی Rest باید اپلیکیشن های Client و Server کاملا از هم مستقل باشند. تنها اطلاعاتی که Client باید بداند URI منبعی است که درخواست کرده است.
- Statelessness: در طراحی Rest تمام هاAPI بدون حالت هستند به این معنی که تمام اطلاعات لازم برای پردازش یک Request باید در خود آن موجود باشد و نباید هیچ session ای در سمت سرور ذخیره شده باشد.
- Cacheability: باید تا جای ممکن Resource ها چه در سمت Client و چه در سمت Server، Cache بشوند. این کار باعث بهبود سرعت در سمت client و همچنین قابلیت Scalability در سمت Server می‌شود.

- Layered system: برای اینکه API را ساده نگه داریم و بتوانیم آن را به راحتی Scale کنیم می توانیم سیستم خود را به لایه های مختلفی بشکونیم که با هم در ارتباط هستند. و در واقع در عمل هر داده‌ای که می آید هر یک از لایه ها یک عملیاتی را روی داده انجام می دهند و آن را به لایه بعدی می فرستند. با این کار امنیت سیستم هم بیشتر می‌شود.

برای مطالعه بیشتر در مورد Rest و Rest API می توانید از این لینک استفاده کنید. در این لینک هم می توانید یک مثال از پیاده سازی Rest در جاوا ببینید.

## Fast Communication

احتمالا بعد خواندن توضیحات مربوط به Rest متوجه شده باشید که ممکن است این روش یکسری جا ها کند باشد و یا مناسب کاری که می خواهیم انجام بدهیم نباشد. برای مثال با وجود اینکه های‌REST API به منظور کاهش پیچیدگی پروتکل های قبلی مانند SOAP ساخته شده‌اند، اما با افزایش تعداد endpoint ها و resource ها پیچیده و پیچیده‌تر می‌شوند و این موضوع چالشی اساسی برای توسعه دهندگان ایجاد می‌کند. یکی دیگر از مشکلات های‌REST API سختی انجام تغییرات اساسی روی آن‌ها است. به صورتی که برای این که این هاAPI از استفاده‌های جدید پشتیبانی کنند نیاز به اعمال تغییرات گسترده روی آن‌ها هستیم که می‌تواند بسیار زمان گیر و سخت باشد. به خاطر همین مشکلات توسعه دهندگان زیادی سراغ جایگزین هایی برای REST رفته‌اند که چند مورد از آن‌ها را در ادامه معرفی می‌کنیم.

- GRPC: یکی از معروفترین روش‌هایی که به عنوان جایگزینی برای Rest وجود دارد GRPC یا همان Google Remote Procedure Calls است. GRPC هم همانند Rest مثل یک قرارداد می ماند که Server و Client با هم سر نحوه رد و بدل کردن Request و Response ها می‌بندند. یکی از تفاوت های مهم GRPC با Rest این است که در Rest ما یکسری قواعد داریم که بهتر است از آنها پیروی کنیم و هیچ اجباری در آنها نیست، ولی در GRPC از یکسری فایل با پسوند proto، برای نوشتن قوانین استفاده می کنیم که باعث می شوند رعایت آن قوانین اجباری باشند. در این لینک و این لینک می توانید توضیحات بیشتر و مقایسه بین Rest و GRPC را ببینید. همچنین می توانید از این پروژه برای راه اندازی GRPC در جاوا کمک بگیرید.
- WebSockets: یکی دیگر از روش هایی که امروزه خیلی مرسوم است WebSockets است. فرق اصلی آن با Rest این است که با این روش می توانیم یک ارتباط دو طرفه real-time داشته باشیم. به عنوان مثال در برنامه هایی که chat دارند یا برنامه هایی مثل map که نیازمند دریافت داده با صورت real-time هستند این روش خیلی

پرکاربرد است. در این روش به طور کلی شروع ارتباط بوسیله handshaking انجام می شود که در این لینک هم می توانید بیشتر درباره آن مطالعه کنید و در ادامه هم داده ها رد و بدل می شوند. برای پیاده سازی این روش هم در java می توانید از jsr-۳۶۵ استفاده کنید که در این لینک هم یک نمونه پروژه با java قرار داده شده.

- خیلی روش های دیگر هم وجود دارند مثل MQTT و EDA و ... که برای مطالعه بیشتر آنها می توانید به این لینک مراجعه کنید.