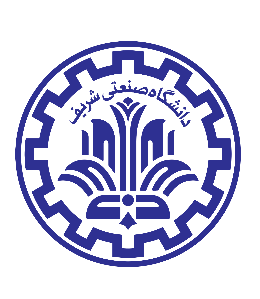
**بسمه تعالی**

**گزارش پروژه درس شبیه­سازی کامپیوتری**

استاد

**دکتر بردیا صفایی**

محمدعلی خدابنده­لو

متین داغیانی

**دانشگاه صنعتی شریف**

**تابستان 1401**

فهرست

[مقدمه و اهداف آزمایش 3](#_Toc66128949)

[تئوری آزمایش 3](#_Toc66128950)

[بخش 1 3](#_Toc66128951)

[بخش 2 3](#_Toc66128952)

[کارهای آزمایشگاهی 3](#_Toc66128953)

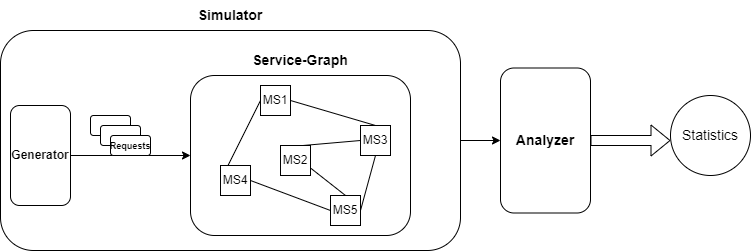
[بخش 1 3](#_Toc66128954)

[بخش 2 3](#_Toc66128955)

# ساختار کلی

به طور کلی شبیه­ساز پیاده­سازی شده از اجزای اصلی Simulator، Generator و Analyzer تشکیل شده است. زبان پیاده­سازی پایتون بوده و از کتاب­خانه به خصوص مربوط به شبیه­سازی استفاده نشده است.

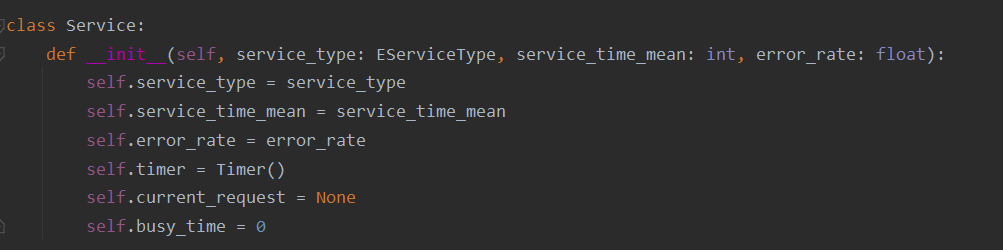
در شکل زیر نمای کلی معماری شبیه­ساز به همراه اجزای آن را ملاحظه می­کنید. در ادامه اجزای مختلف به طور خلاصه شرح داده شده اند.



# قسمت­های مختلف کد

## Service

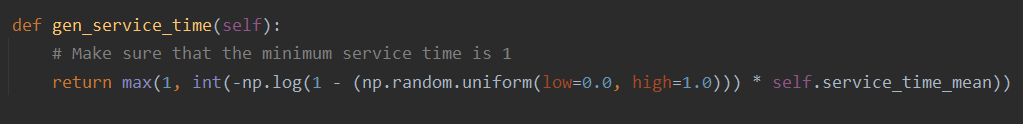
هر نمونه از این کلاس حاوی اطلاعات یکی از instanceهای سرویس­دهنده در یک ماکروسرویس به خصوص است. در تصویر زیر بخشی از کد مربوطه را ملاحظه می­کنید.



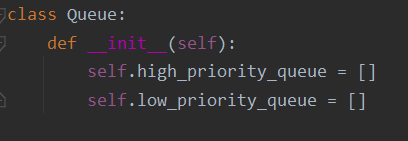
هر نمونه از این کلاس می­تواند یکی از 7 نوع *درگاه وب، درگاه موبایل، مدیریت مشتریان و ...* باشد. هم­چنین اطلاعات لازم برای خدمت­گزاری که برای عملکرد و تهیه آمارها مورد نیاز است نگهداری می­شوند. متغیر service\_time\_mean نمایان­گر میانگین زمان خدمت­گزاری است. هم­چنین به منظور کنترل زمان خدمت رسانی یک timer نگهداری می­شود که با گذشت هر واحد زمان شبیه­سازی به روز شده و مدت زمان باقی­مانده خدمت­گزاری را نشان می­دهد.

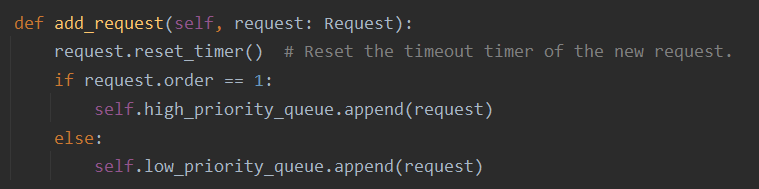
current\_request به درخواستی اشاره دارد که هم­اکنون توسط سرویس در حال خدمت­گزاری است. در صورت idle بودن سرور، برابر با None می­باشد.

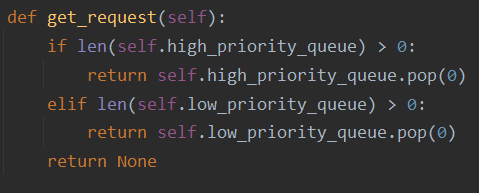
برای تولید کردن واریته توزیع نمایی با میانگین نرخ مشخص، از متد زیر استفاده شده است.

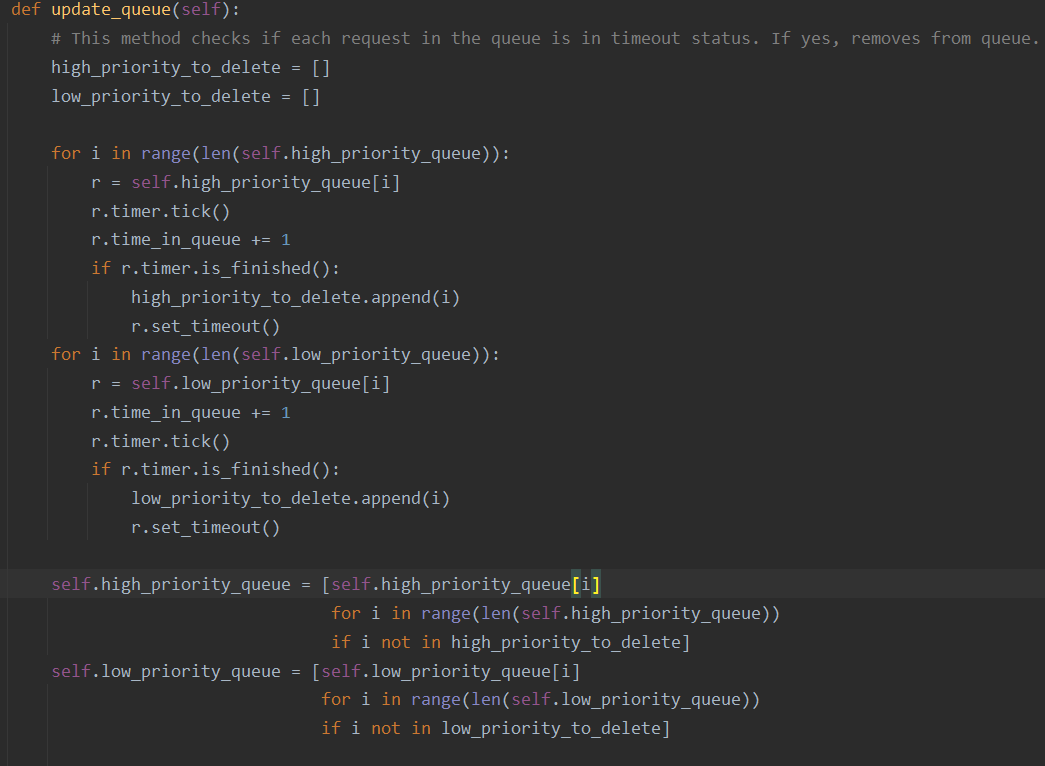


## Queue

این کلاس مسئول مدیریت هر یک از صف­های متناظر با ماکروسرویس­ها است. در شکل زیر فیلد­های اصلی را ملاحظه می­کنید.

برای مدیریت درخواست­ها با اولیت­های مختلف، یک صف برای درخواست­های با اولویت بالا و یک صف برای درخواست­ها با اولیت پایین در نظر گرفته شده است.هنگام اضافه شدن درخواست به صف، بر اساس اولویت به یکی از این دو لیست اضافه می­شود:

هم­چنین برای انتخاب و انتقال یک درخواست به یک سرور، ابتدا درخواست­ها در صف با اولویت بالا چک شده و سپس در صورت خالی بودن آن صف با اولویت پایین­تر چک می­شود. سیاست انتخاب از هر دوصف FIFO است:

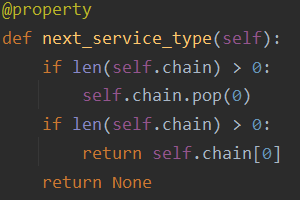
هم­چنین برای مدیریت و حذف درخواست­هایی Timeout شده، متد update\_queue در هر واحد زمانی صدا زده می­شود:

## Request

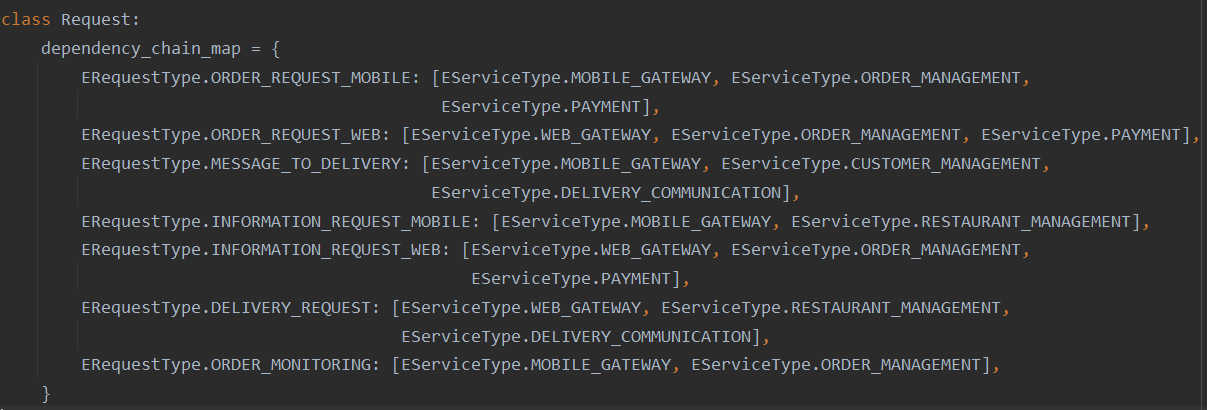
Text

Description automatically generatedهر نمونه از این کلاس نشان­دهنده یک درخواست ایجاد شده است که بر اساس نوع آن (*ثبت سفارش با موبایل، سفارش با وب، ارسال پیام به پیک و ...)* وارد سیستم­شده و در طول شبیه­سازی به ماکروسرویس­های متناظر منتقل شده و سپس از سیستم خارج می­شود. هم­چنین اطلاعات لازم برای بررسی­های تحلیلی و آماری پس از خروج از سیستم در آن نگهداری می­شود.

برای پیاده­سازی *حداکثر زمان انتظار در صف*، متغیرهای timer و waiting\_time در نظر گرفته شده­اند. هم­چنین متغیر time\_out نشان­دهنده وضعیت timeout شدن درخواست را نشان می­دهد.

پس از اتمام سرویس­دهی یکی از ماکروسرویس­ها به هر درخواست، باید نوع ماکروسرویس بعدی مشخص شود. بدین منظور لیستی از ServiceType ها در هر درخواست نگهداری می­شود(متغیر chain) که می­توان به کمک متد زیر مقصد بعدی را مشخص کرد:

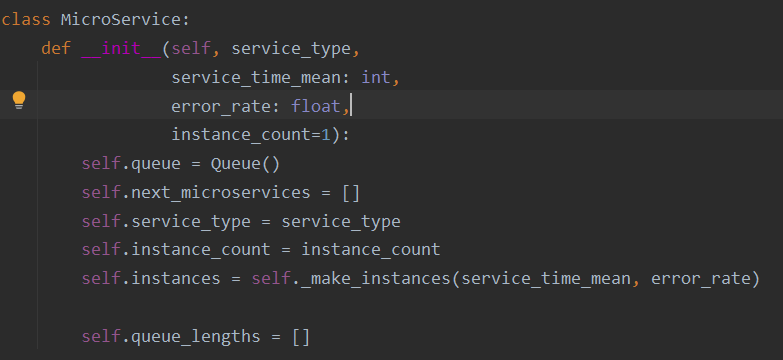
برای مدیریت انواع درخواست­ها و ماکروسرویس­های متناظر از یک دیکشنری به صورت زیر استفاده شده است.



## Micro-Service

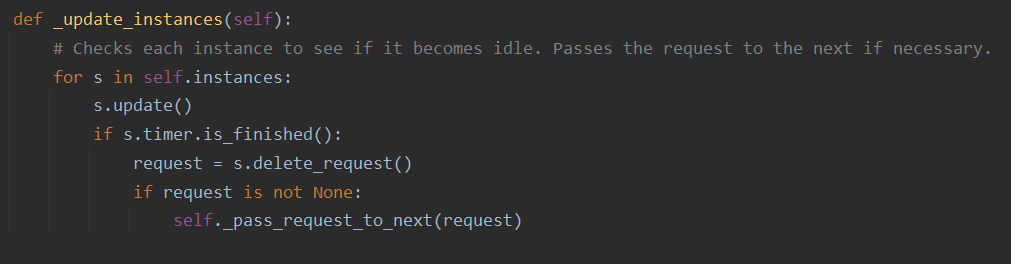
هر نمونه از این کلاس نشان­دهنده یکی از ماکروسرویس­های موجود در سیستم است و شامل موارد زیر می­شود:

1. **صف درخواست­ها**
2. **نوع ماکروسرویس**
3. **ماکرو­سرویس­هایی که درخواست از این ماکروسرویس­ می­تواند به آنها منتقل شود**
4. **تعداد و لیستی از نمونه سرویس­ها**
5. **لیستی حاوی طول صف که در طول زمان به روزرسانی شده و برای تحلیل نتایج کاربرد دارد.**



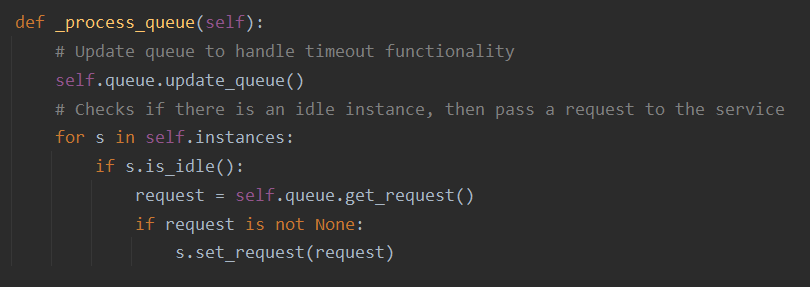
این کلاس هم­چنین دو وظیفه مهم دارد که در هر واحد زمانی به ترتیب زیر اجرا می­شوند:

1. **به روزرسانی instanceها:**

در این مرحله وضعیت هر یک از نمونه­های سرور چک می­شود. اگر زمان خدمت­راسنی یک نمونه به پایان رسیده باشد، درخواست مورد خدمت­رسانی دریافت شده، سپس بر اساس نوع آن به یکی از ماکروسرویس­های بعدی منتقل می­شود:

1. **پردازش صف درخواست­ها**

در این مرحله ابتدا صف مربوطه و درخواست­های آن به روز می­شوند، سپس اگر نمونه idleی موجود بود، یک درخواست از صف حذف شده و به سرور مربوطه نگاشت می­شود:

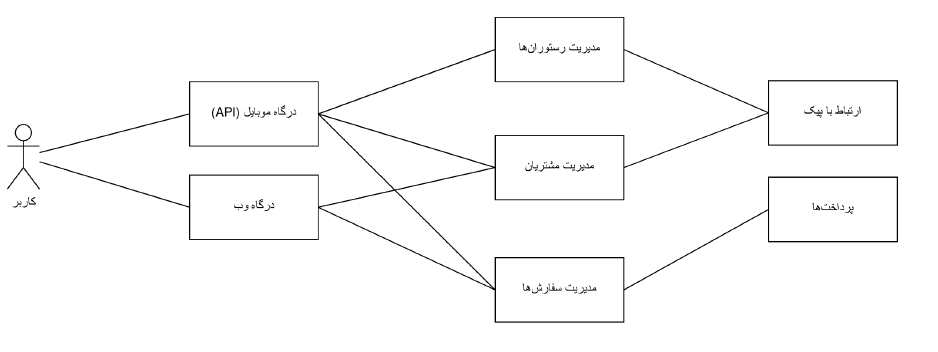


در نهایت لازم است تا متد update() در هر واحد زمانی شبیه­ساز صدا زده شود:

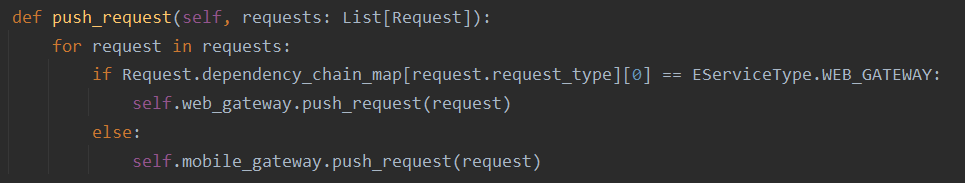
*Text

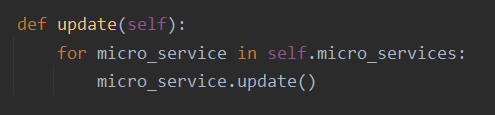
Description automatically generated*

## Service Graph

این کلاس وظیفه مقداردهی اولیه ماکروسرویس­ها و اتصال ­آنها به یکدیگر را دارند تا کل سیستم به صورت یک گراف با رئوس ماکروسرویس مطابق شکل زیر تشکیل شود.

هنگام ورود لیست درخواست­ها در هر واحد زمانی، پس از بررسی نوع آن­ها، به یکی از ماکروسرویس­های اولیه یعنی *درگاه موبایل* یا *درگاه وب*  اضافه می­شوند.



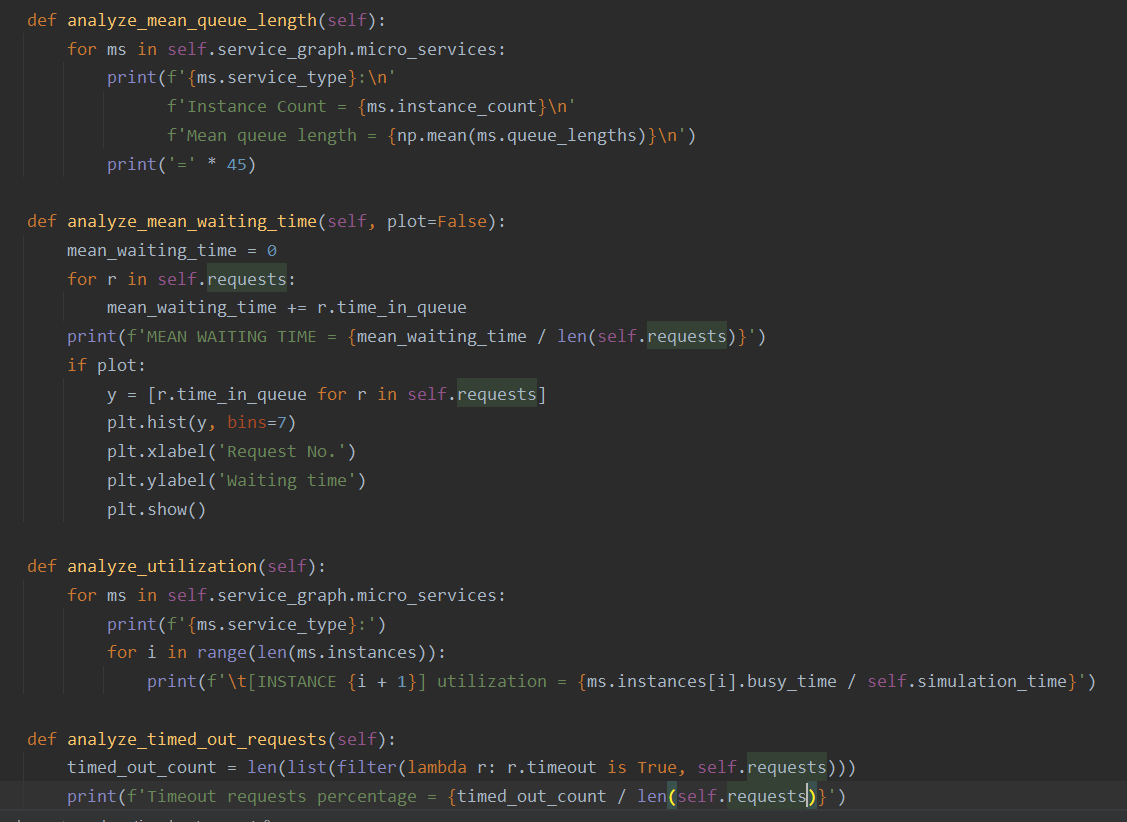
هم­چنین در هر واحد زمانی، تمام ماکروسرویس­ها به روز­رسانی می­شوند:

## Simulator

## Generator

## Analyzer

همان­طور که اشاره شد، تمام درخواست­های خروجی شبیه­ساز در یک لیست­نگهداری می­شوند. این کلاس با دریافت این لیست و هم­چنین تمام ماکروسرویس­ها، نتایج تحلیلی مورد نظر پروژه را محاسبه می­کند.



# نحوه اجرا

پس از مقداردهی متغیرهای ورودی شبیه­ساز، با اجرای فایل main.py شبیه­سازی آغاز شده و در نهایت نتایج آماری آن در کنسول نمایش داده می­شوند.

# نتایج تحلیلی

## بخش 1

## بخش 2