مستندات معماری سیستم

- **معرفی**

در این سرویس کاربر داده خود را به صورت باینری ، همراه با شناسه یکتا و شناسه کاربری خود ارسال میکند.
سرویس وظیفه دارد که یکتا بودن داده و شناسه یکتا را بررسی کند.

  - داده ورودی : داده‌ای که کاربر به سرویس ارسال میکند.
  - شناسه کاربری : شناسه‌ای که سیستم دیتا مربوط به هر کاربر را توسط آن تفکیک و ذخیره سازی میکند.
  - شناسه یکتا : شناسه یکتا فقط یک بار در کل سرویس وجود دارد و هیچ موجودیت دیگری با همان شناسه نمی‌تواند در سیستم وجود داشته باشد.

- **ارتباط کاربر با سرویس**

ارتباطات کاربر با سرویس بر بستر پروتکلHTTP می‌باشد.

- **محدودیت‌های ارتباط کاربر با سرویس**
  - هر کاربر میتواند تعداد درخواست مشخصی در هر یک دقیقه به سرویس ارسال کند ، محدودیت تعداد درخواست در یک دقیقه در سیستم قابل تغییر است.
  - هر کاربر میتوند فقط حجم مشخصی داده در ماه به سمت سرویس ارسال کند ، محدودیت حجم ارسالی در ماه در سیستم قابل تغییر است.

- **بررسی شناسه یکتا**

شناسه یکتایی که توسط کابر به سرویس ارسال میشود نباید در سرویس وجود داشته باشد و برای بررسی کردن شناسه یکتا نباید زمان و منابع زیادی مورد استفاده قرار گیرد ، به همین دلیل شناسه یکتا به فرمت UUID در نظر گرفته شده است.

UUID : UUID یا Universally Unique Identifier به صورت ساده یک شناسه یکتا و یکدست است که به‌طور جهانی شناخته می‌شود. این شناسه از یک مقدار 128 بیتی (معمولاً نمایش شده به‌صورت رشته‌ای با 32 کاراکتر هگزادسیمال) تشکیل شده است.

UUID به طور خودکار ایجاد می‌شود و هرگز تکراری نیست. برای ایجاد UUID از مقادیری مانند زمان کنونی، آدرس MAC دستگاه، و موارد دیگر استفاده می‌شود.

- **بررسی داده‌های تکراری**
  - یکی از اهداف مهم سرویس جلوگیری از پردازش داده‌های تکراری است.
  - داده‌های تکراری در سیستم داده‌هایی شناخته میشوند که قبلا توسط سیستم پردازش شده‌اند.
  - از راهکارهای تشخیص داده‌های تکراری استفاده از hash function ها میباشد ، هش فانکشن‌ها دیتای ورودی را هش میکنند و آن را به یک رشته تبدیل میکنند که با آن رشته checksum گفته میشود.
  - داده‌هایی که checksum آن‌ها مشابه همدیگر باشد داده تکراری شناخته میشوند.

- **بررسی checksum**

سرویس وظیفه دارد که در زمان محدود ، checksum هر داده‌ای با هر حجم ممکن بدست بیاورد. به همین خاطر معایب و مزایا hash functionها را بررسی میکنیم.

- SHA256
  - ❖ معایب : زمان بالای پردازی نسبت به بقیه الگوریتم‌ها.
  - ❖ مزایا : احتمال صفر در برابر ایجاد هش تکراری.
- MD5
  - ❖ معایب : احتمال پایین در ایجاد هش تکراری.
  - ❖ مزایا : زمان پایین پردازش برای انواع داده حجیم و سبک.

در سرویس از الگوریتم MD5 برای ساخت checksum داده‌های سیستم استفاده شده است.

- **پایگاه‌های داده استفاده شده در سیستم**

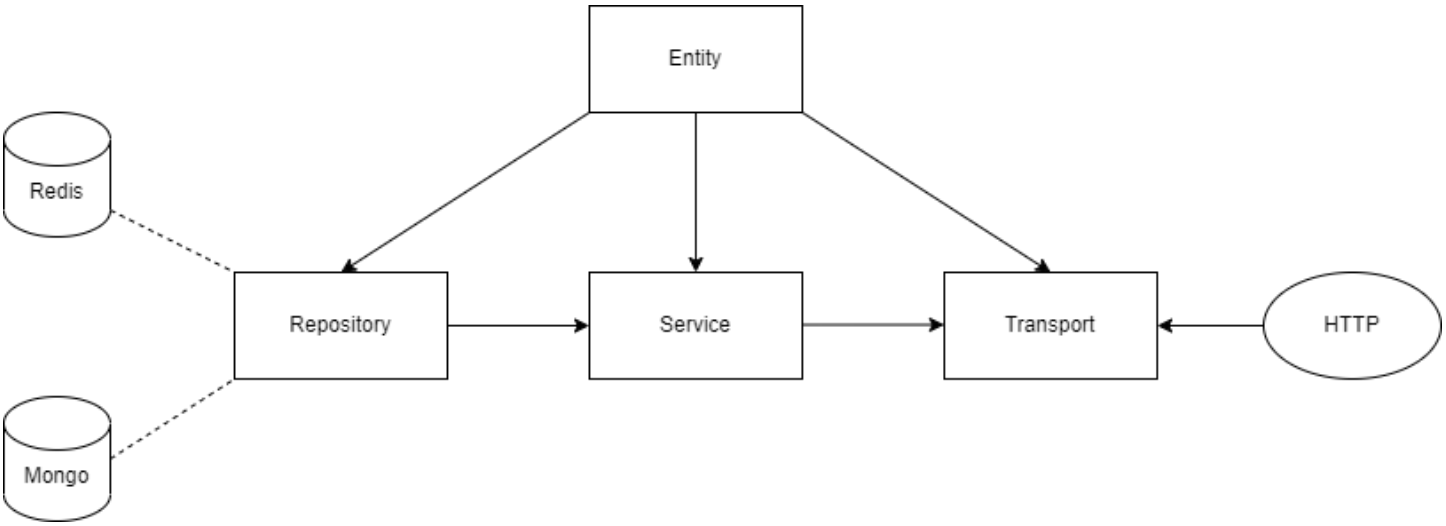
❖ Redis

با توجه به نیاز به پیاده‌سازی کوتای کاربران و نیاز به کش کردن میزان تعداد درخواست‌های کاربر در دقیقه و حجم ارسالی کاربر در ماه، از پایگاه داده Redis برای انجام این کار استفاده می‌کنیم. Redis به علت قابلیت کش کردن داده در زمان مشخص و سرعت بالا و استفاده بهینه از منابع، به‌عنوان پایگاه داده اصلی در سرویس پیاده‌سازی شده است. با استفاده از Redis می‌توانیم اطلاعات مربوط به هر کاربر مانند تعداد درخواست‌ها در دقیقه و حجم ارسالی در ماه را کش کنیم. از قابلیت کش کردن زمان‌بندی‌شده Redis برای داده‌ها استفاده می‌کنیم، تا اطلاعات به‌روز شده همیشه در دسترس باشند و از منابع سیستم به بهترین شکل استفاده شود. از طرفی، سرعت بالا و زمان پاسخ‌دهی کم Redis باعث شده که از این پایگاه داده برای بخش کوتای کاربران مورد استفاده قرار گیرد.

❖ Mongodb

برای ذخیره سازی داده‌ها و checksum آن‌ها همچنین برای برسی غیر تکراری بودن شناسه یکتا از Mongodb در سرویس استفاده شده است، همچنین از این پایگاه داده به دلیل انعطاف‌پذیری ذخیره‌سازی داده‌ها و ویژگی‌های منحصربه‌فردی که ارائه می‌دهد، به‌عنوان یک پایگاه داده قدرتمند و مناسب در سرویس انتخاب شده است. با این انتخاب ، می‌توان سرویس را به‌صورت بهتر و با توجه به نیازها و تغییرات بهبود داد.

معماری نرم افزار



- **معرفی معماری**

در معماری نرم افزار از Clean architecture استفاده شده است که نرم افزار را به چهار قسمت تقسیم میکند تغییر در هر یک از لایه‌های بالایی باعث ایجاد تغییر در لایه‌های پایین‌تر میشود.

بخش‌ها و لایه های معماری به صورت زیر است :

- Transport

این لایه وظیفه برقراری ارتباط با کاربر را دارد و به اصطلاح این لایه دروازه معماری ما میباشد؛ پیاده سازی بستر ارتباطی در این لایه بسیار ساده میباشد. این لایه وابسته به تمام لایه‌ها است.

- Service

این لایه وظیفه دارد که منطق و بیزنس لاجیک را برقرار نمایید. همچنین این لایه وابسته به Repository و Entity است.

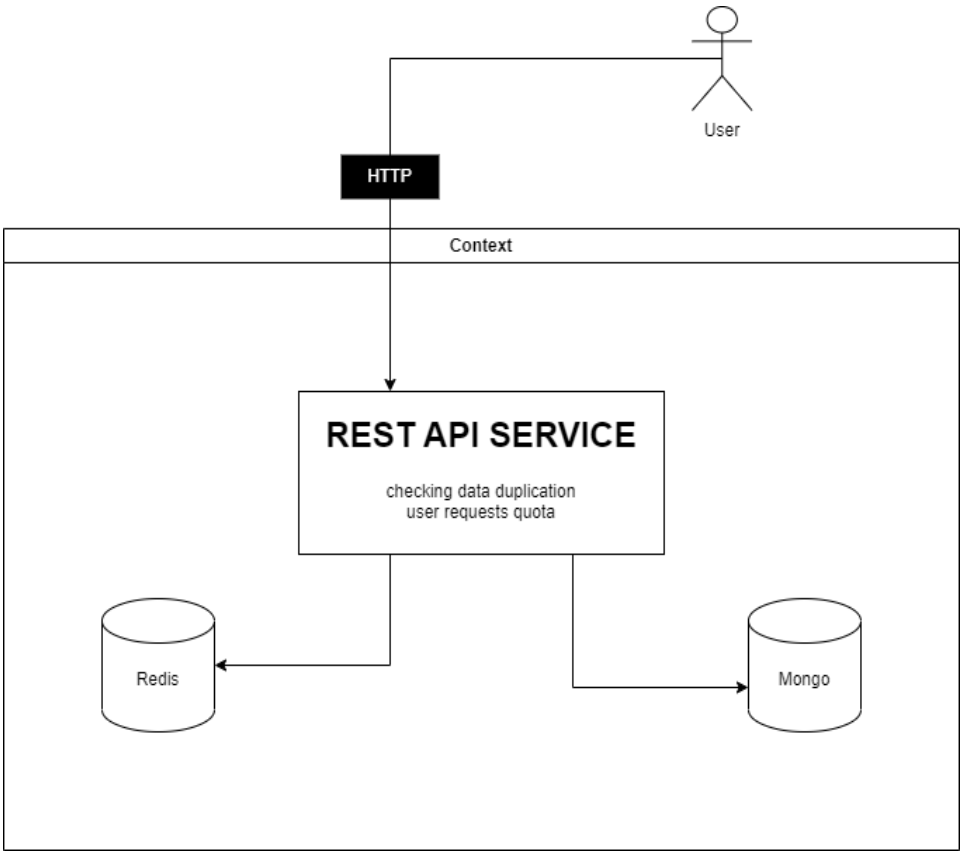
- Repository

این لایه وظیفه دارد که با سرویس‌ها یا سیستم‌های خارجی ارتباط برقرار کند. برای مثال میتوان گفت که این لایه دروازه خروجی نرم‌افزار است. همچنین این لایه وابسته به Entity است.

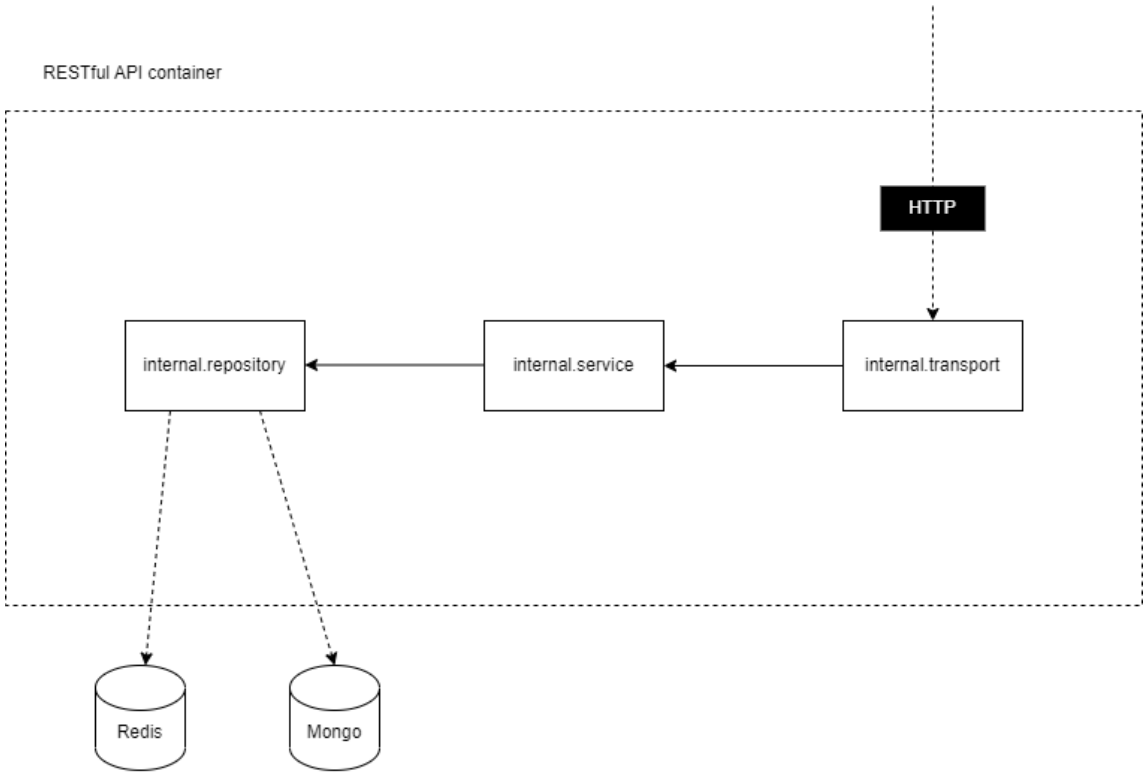
- Entity

تمام مدل‌ها و بیزنس لاجیک‌های نرم افزار در این قسمت قرار میگیرد ؛ تغییر دادن این قسمت منجر به تغییر یافتن کل سرویس خواهد شد و تمام لایه‌های نرم افزار وابسته به این قسمت میباشند.

- C4 Model
  - Container



  - Container



- Nginx

برای سرویس‌دهی به تعداد بالای کاربران نیاز به استقرار چند سرویس داریم تا به صورت موازی با یکدیگر، با کاربران ارتباط برقرار کنند. به هم این دلیل بهترین انتخاب برای پیاده سازی load balancer سرویس nginx است که به صورت رندوم بین سرویس‌های ساخته شده ترافیک را تقسیم میکند.

- Docker

برای اجرای کد از داکر استفاده شده است.

مراحل اجرای سرویس

git clone https://github.com/amookia/arvan-backend-challenge

cd deployments

docker-compose up -d

- مستندات rest-api

تمام مستندات rest-api در پوشه docs در لینک زیر قرار داده شده است.

https://github.com/amookia/arvan-backend-challenge