

طراحی کلی:

سرویس اول تصویر و آدرس ایمیل کاربر را به عنوان درخواست می‌گیرد و بعدش: MinIO برای ذخیره‌سازی تصویر استفاده می‌کند
RabbitMQ برای ارسال تصویر به سرویس دوم به کار میره
PostgreSQL برای ذخیره اطلاعات درخواست شامل ID، ایمیل و وضعیت pending استفاده می‌کند

سرویس دوم تصویر دریافتی از RabbitMQ را پردازش می‌کند:
با استفاده از HuggingFace API، یک کپشن برای تصویر تولید می‌کند
کپشن تولیدشده در پایگاه داده ذخیره شده و وضعیت درخواست به‌روزرسانی می‌شود

سرویس سوم بر اساس کپشن تولیدشده، تصویر جدیدی تولید می‌کند:
از HuggingFace API برای تولید تصویر جدید استفاده می‌شود
تصویر جدید در MinIO ذخیره و URL آن در پایگاه داده به‌روزرسانی می‌شود
url تصویر جدید به ایمیل کاربر ارسال می‌شود

برای انجام این پروژه از پلتفرم هم‌روش استفاده کردم. هم‌روش یک سرویس ابری قدرتمند است که به من کمک کرد تا سرویس‌های مختلف پروژه را به صورت کانتینریزه شده مدیریت و دیپلوی کنم. ابتدا برای هر یک از سرویس‌ها، یک داکر ایمج ایجاد کردم و سپس این ایمج‌ها را در Docker Hub بارگذاری کردم. پس از آن، از طریق هم‌روش و با استفاده از نام داکر ایمج‌ها، به راحتی سرویس‌ها را روی سرورهای ابری دیپلوی و اجرا کردم. استفاده از هم‌روش باعث شد که فرآیند استقرار و مدیریت سرویس‌ها به صورت یکپارچه و بهینه انجام شود.

سرویس اول: ثبت درخواست پردازش تصویر

این سرویس مسئول ثبت درخواست‌های پردازش تصویر است. کاربران یک تصویر آپلود می‌کنند و سرویس این درخواست را با ذخیره تصویر در MinIO، ارسال پیام به RabbitMQ و ذخیره جزئیات درخواست در پایگاه داده PostgreSQL پردازش می‌کند

ویژگی‌های مهم این سرویس:

دیتابیس PostgreSQL: اطلاعات مربوط به درخواست‌های پردازش تصویر، یعنی ایمیل کاربر، وضعیت پردازش، کپشن تصویر و URL تصویر جدید در پایگاه داده ذخیره می‌شود سرویس به صورت خودکار در صورت نیاز جدول مربوط به درخواست‌ها را ایجاد می‌کند.

ارتباط با RabbitMQ دریافت درخواست، یک پیام شامل اطلاعات تصویر به صف image_queue در RabbitMQ ارسال می‌شود تا توسط سرویس‌های دیگر پردازش شود. RabbitMQ برای مدیریت صف‌ها و ارتباط غیرهمزمان بین سرویس‌ها استفاده می‌شود.

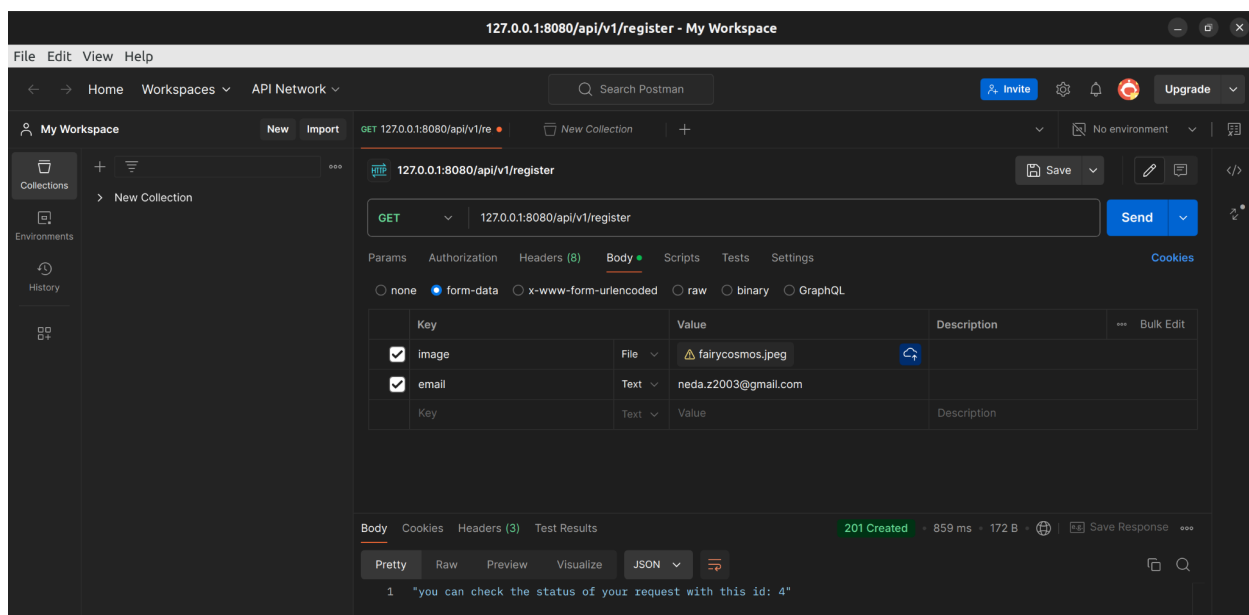
MinIO: تصویر اصلی آپلود شده توسط کاربر در سرویس MinIO ذخیره میشه. minio یک سیستم ذخیره سازی اشیاء مشابه با S3 است که برای ذخیره سازی فایل ها به کار می رود.

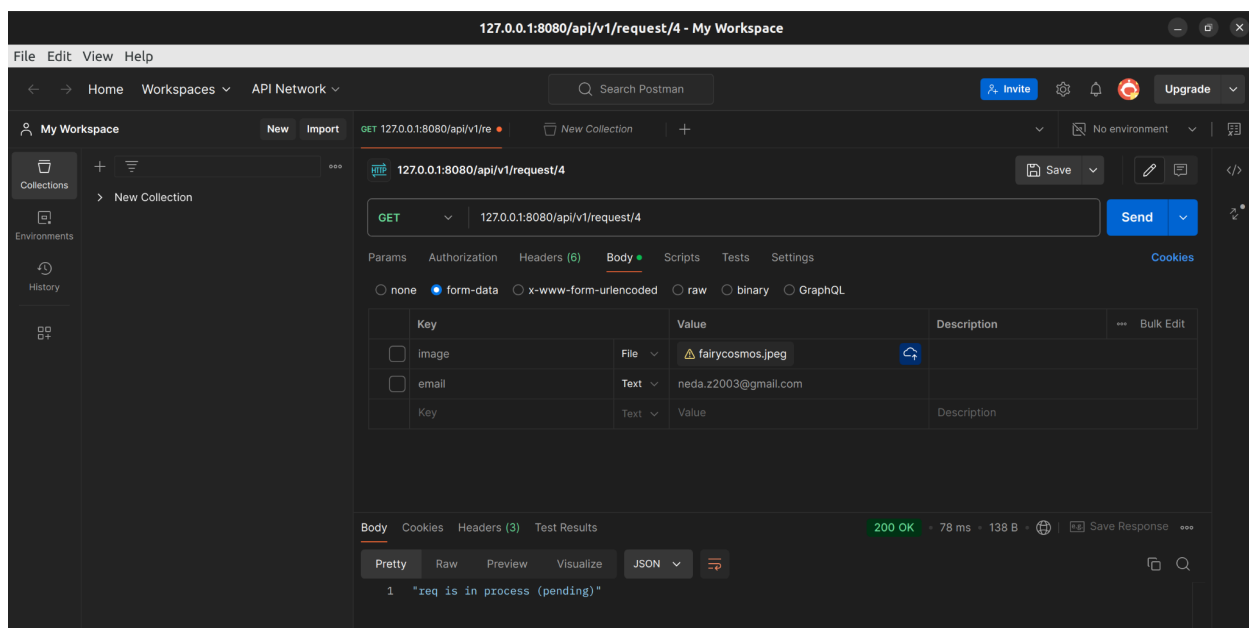
API ها: دوتا اند پوینت مهم وجود داره:

ثبت درخواست: کاربران می توانند یک تصویر را با ارسال یک درخواست POST به `api/v1/register/` ثبت کنند.
وضعیت درخواست: کاربران می توانند با استفاده از درخواست GET و ارسال شناسه درخواست به `api/v1/request/:id/`، وضعیت پردازش تصویر را بررسی کنند

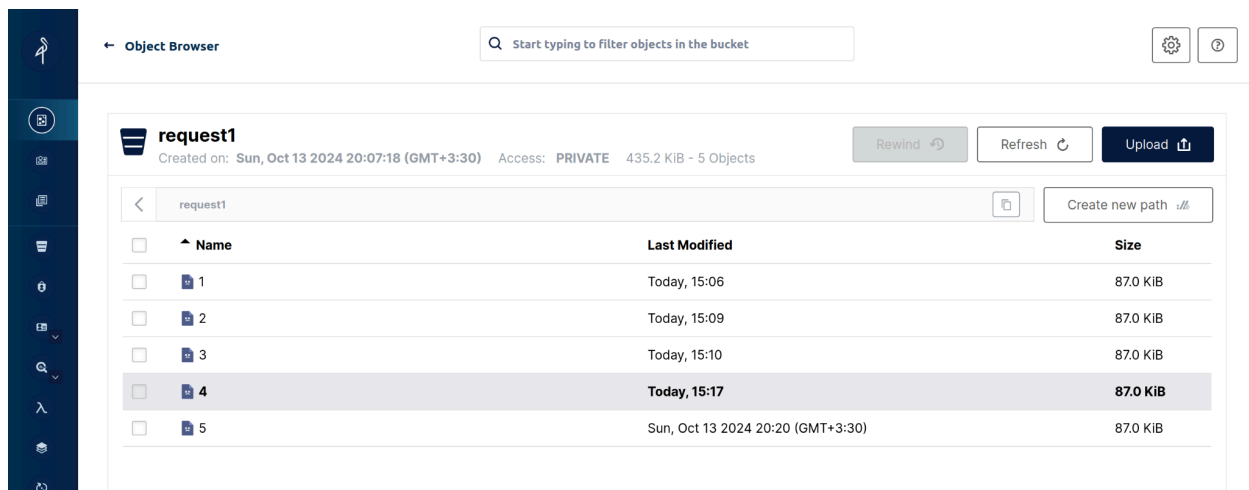
این سرویس از Echo برای مدیریت درخواست ها و پاسخ ها استفاده می کنه . اطلاعات اتصال به RabbitMQ، PostgreSQL و MinIO از طریق فایل پیکربندی YAML خوانده می شوند و هنگام راه اندازی سرویس، اتصال به این سرویس ها برقرار می شود

نتایج ریکوست ها در postman :





عکس ذخیره شده در باکت minio:



سرویس دوم: تولید کپشن برای تصویر

نمای کلی: این سرویس وظیفه دریافت تصاویر از سرویس اول، پردازش آن‌ها با استفاده از API HuggingFace و تولید کپشن (توضیح) برای تصویر را بر عهده دارد. سپس کپشن تولیدشده را در پایگاه داده PostgreSQL ذخیره می‌کند و وضعیت درخواست را به‌روزرسانی می‌کند

ویژگی‌های مهم:

دریافت پیام از RabbitMQ: سرویس دوم پیام‌هایی که از سرویس اول در صف RabbitMQ قرار داده شده‌اند را دریافت می‌کند. پیام‌ها شامل شناسه تصویر هستند که به این سرویس اجازه می‌دهد تصویر مربوطه را از MinIO بازپایی کند

پردازش تصویر و تولید کپشن: تصویر با استفاده از HuggingFace API برای تولید کپشن ارسال می‌شود. این API از مدل‌های دیپ لرنینگ برای تحلیل تصویر و تولید توضیح متنی استفاده می‌کند

ذخیره کپشن و به‌روزرسانی وضعیت: کپشن تولیدشده به همراه وضعیت ready در پایگاه داده PostgreSQL ذخیره می‌شود. وضعیت به ready تغییر می‌کند که نشان‌دهنده پایان موفقیت‌آمیز پردازش تصویر است.

ارتباط با MinIO: تصویر اصلی از سرویس minio بازیابی شده و پس از پردازش، وضعیت آن در سیستم به‌روزرسانی می‌شود.

API های HuggingFace:

این سرویس از HuggingFace API برای تولید کپشن بر اساس تصویر استفاده می‌کند. API از طریق درخواست HTTP POST با تصویر ارسال‌شده ارتباط برقرار می‌کند و متن کپشن را به عنوان پاسخ بازمی‌گرداند

همچنین پیام‌های دریافتی از RabbitMQ پردازش شده و کپشن مربوط به تصویر ایجاد می‌شود. از Echo برای مدیریت سرور و درخواست‌ها استفاده شده

نتیجه ریکوست‌ها در دیتابیس request_service بعد از ران کردن سرویس دوم:

```
nda@nda-Vivobook-ASUSLaptop-M3401QC: ~/Downloads
nda@nda-Vivobook-ASUSLaptop-M3401QC:~/Downloads$ psql -U postgres -h b7dea484-e154-4352-99fe-147b0c320e9e.hsbc.ir -p 30610
Password for user postgres:
psql (17.0 (Ubuntu 17.0-1.pgdg22.04+1), server 16.1 (Debian 16.1-1.pgdg110+1))
Type "help" for help.

postgres=# \c request_service
psql (17.0 (Ubuntu 17.0-1.pgdg22.04+1), server 16.1 (Debian 16.1-1.pgdg110+1))
You are now connected to database "request_service" as user "postgres".
request_service=# select * from requests;
 id | email | status | image_caption | new_image_url
-----+-----+-----+-----+-----
  4 | neda.z2003@gmail.com | ready | painting of a dog flying over a city with a strawberry in its mouth |
(1 row)

request_service=# fd;'
```

سرویس سوم: پردازش تصویر از کپشن

این سرویس پردازش درخواست‌های تولید تصویر بر اساس کپشن‌های ورودی از کاربران را انجام می‌دهد. پس از دریافت کپشن، سرویس با استفاده از API هاگینگ فیس، تصویر مربوطه را تولید کرده و تصویر حاصل را در minio ذخیره می‌کند. همچنین، وضعیت پردازش را در پایگاه داده PostgreSQL به‌روزرسانی کرده و به کاربران ایمیلی مبنی بر تکمیل پردازش ارسال می‌کند. ویژگی‌های مهم این سرویس:

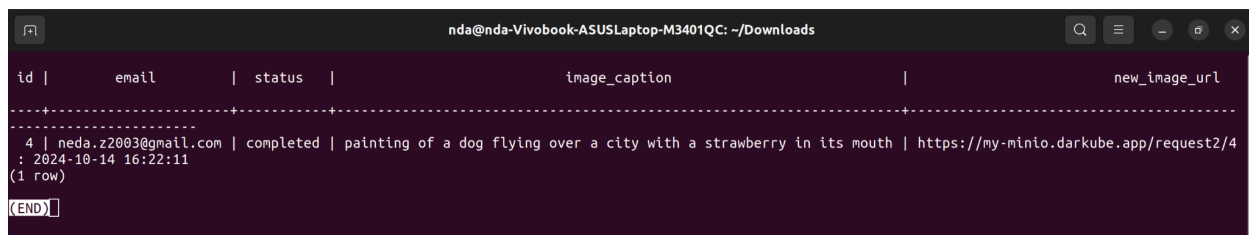
پایگاه داده PostgreSQL: اطلاعات مربوط به درخواست‌های پردازش تصویر، شامل ایمیل کاربر، وضعیت پردازش، کپشن تصویر و URL تصویر جدید در پایگاه داده ذخیره می‌شود. این سرویس به‌طور خودکار جدول مربوط به درخواست‌ها را در صورت نیاز ایجاد می‌کند

API هاگینگ فیس: برای تولید تصویر بر اساس کپشن، درخواست به API هاگینگ فیس ارسال می‌شود و تصویر حاصل به عنوان پاسخ دریافت می‌شود. این فرآیند به صورت غیرهمزمان انجام می‌شود و به کاربران این امکان را می‌دهد که به‌طور مداوم درخواست‌های خود را ثبت کنند.

ایمیل به کاربران: پس از جنریت کردن و بارگذاری تصویر، ایمیلی به کاربر ارسال می‌شود تا او را از موفقیت تولید تصویر مطلع کند.

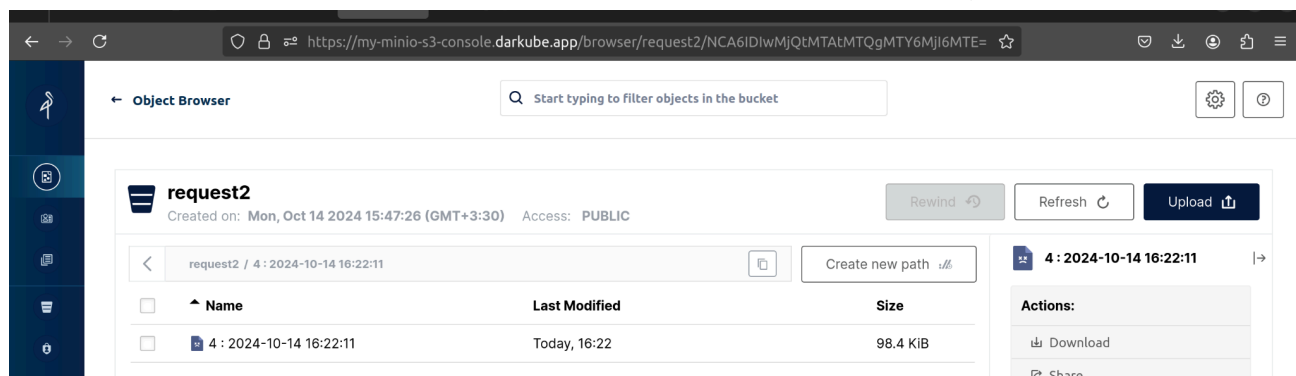
مدیریت درخواست‌ها: این سرویس از یک تایمر برای پردازش دوره‌ای درخواست‌ها استفاده می‌کند و به‌صورت خودکار درخواست‌های آماده را برای پردازش بررسی می‌کند.

نتیجه ران کردن سرویس سوم در دیتابیس:

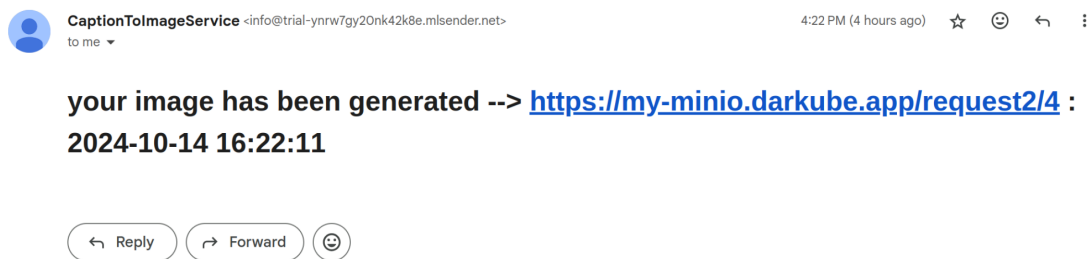


```
nda@nda-Vivobook-ASUSLaptop-M3401QC: ~/Downloads
id | email | status | image_caption | new_image_url
-----+-----+-----+-----+-----
4 | neda.z2003@gmail.com | completed | painting of a dog flying over a city with a strawberry in its mouth | https://my-minio.darkube.app/request2/4
: 2024-10-14 16:22:11
(1 row)
(END)
```

عکس جدید جنریت شده در باکت دوم بابلینک minio :

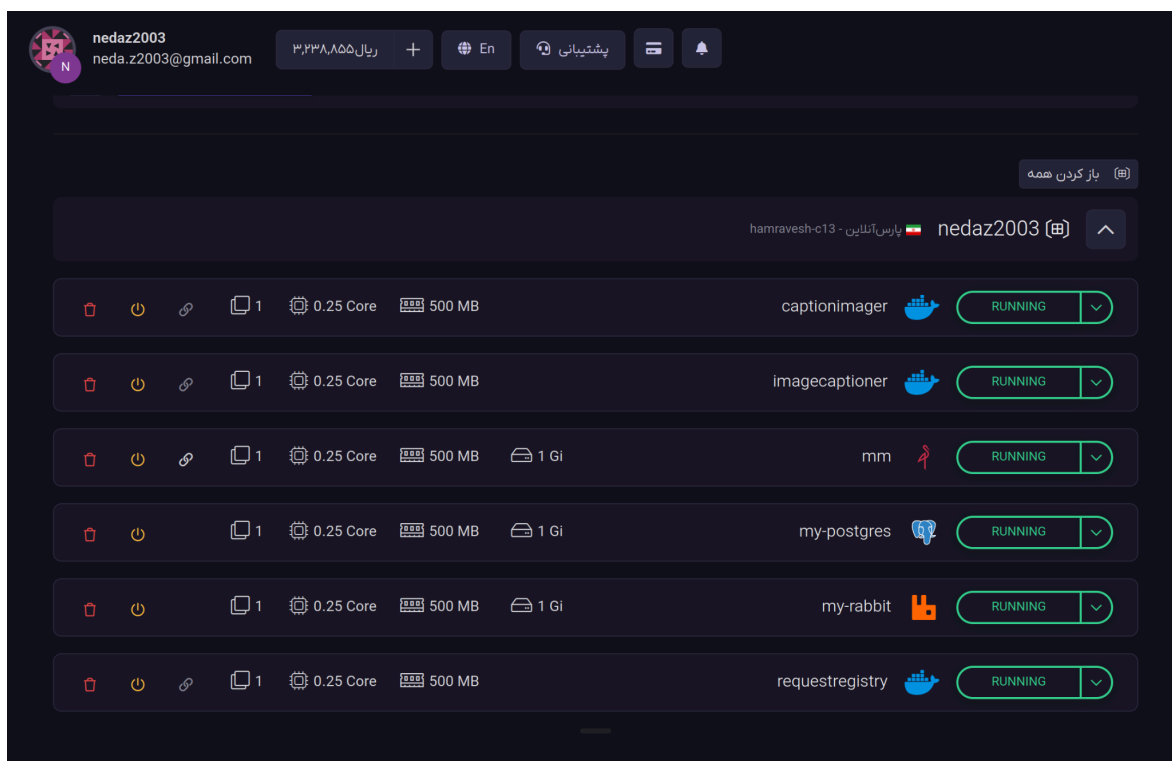


نتیجه ایمیل :



برای دیپلوی کردن سه سرویس بر روی هم روش، از روش دیپلوی با داکر ایمیج استفاده کردم. برای این کار، ابتدا برای هر سرویس یک داکر فایل نوشتم و سپس با استفاده از دستور `docker build`، داکر ایمیج‌های مربوطه را ایجاد کردم. بعد از ساخت این ایمیج‌ها، آن‌ها را در یک ریپازیتوری عمومی روی Docker Hub قرار دادم. سپس در هم روش، با استفاده از نام این داکر ایمیج‌ها، سرویس‌ها را دیپلوی کردم.

اپ های ساخته شده و استفاده شده در این پروژه در هم روش:



نتیجه نهایی:

عکس ارسالی : Image.jpg

4 : 2024-10-14 16:22:11 : عکس جنریت شده از هاگینگ فیس

