گزارش پروژه بات لیمو

۱. توضیحات کلی

این فایل شامل توضیحاتی درباره پروژه تحویل داده شده داده است:

۱. اسم باتی که ساخته شده است testbot می‌باشد؛

۲. پروژه به صورت کامل انجام شده و تمام feature های درخواست شده تحویل داده شده اند؛

۳. فایل های پروژه در لینک این لینک در گیت‌هاب قابل مشاهده می‌باشد؛

۴. دستورات بات به زبان انگلیسی هستند (مانند gitlab/) ولی پیام های بات به فارسی نوشته شده‌اند؛

۵. نحوه کار با بات به صورت درخواست شده می‌باشد و دارای دستور های زیر است.

۵-۱. help/: با زدن این دستور، توضیحات بات برای شما ارسال می‌شود.

۵-۲. gitlab/: با زدن این دستور میتوانید یکی از دو گزینه را انتخاب کنید:

- **projects**: با زدن این دستور، در صورتی که برای اولین بار باشد که از این دستور استفاده میکنید، بات از شما gitlab_token را درخواست میکند. در این مرحله باید دقت کنید که این توکن را درست وارد کنید چراکه اگر اشتباه وارد بکنید باید توکن شما از داخل دیتابیس تغییر داده شود. (به دلیل کمبود وقت برای تحویل پروژه، پروسه تصحیح توکن در بات تعبیه نشده است). پس از وارد کردن gitlab_token بات لیستی از پروژه های private را برای کاربر در لیمو میفرستد. در صورت درخواست دوباره از بات برای پروژه ها، بات دیگر از کاربر درخواست gitlab_token نمیکند و از توکنی که در دیتابیس ذخیره کرده است استفاده میکند.

- **events**: با زدن این دستور، بات برای شما یک url و یک token میفرستد که میتوانید به عنوان webhook از آن استفاده بکنید. این وب هوک در صورتی که gitlab به آن یک ریکوئست POST بزند، هِدِری با عنوان X-Gitlab-Token را شناسایی کرده و key آن هدر را برای لیمو می‌فرستد. مکانی که بات این ایونت ها را میفرستد همان thread ای خواهد بود که مکالمه با آن در آن thread شروع شده است (بات thread_root_id های تمام مکالمه های شروع شده با آن را در دیتابیس ذخیره می‌کند).

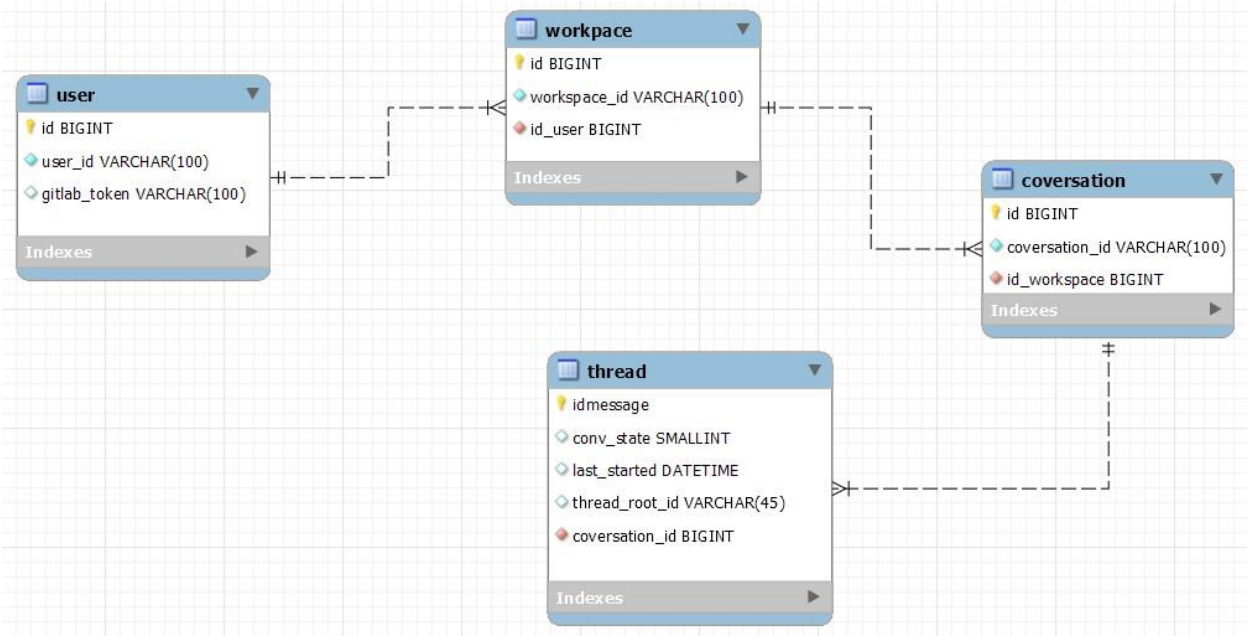
۲. یادگیری ها

برای انجام پروژه زمان زیادی صرف شده است. از جمله:

- **Docker**: دانش من در مورد داکر صفر بود و برای انجام این پروژه من تقریبا ۲.۵ روز صرف یادگیری داکر کردم که بتوانم در پروژه از آن استفاده بکنم. آشنایی با مفهوم container ها، نحوه کار docker و نحوه استفاده از-docker compose برای سهولت در لود کردن یکجای پروژه و مدیریت volume ها و port ها از مواردی بود که باید یاد میگرفتم.
- **Linux**: من دانش ابتدایی در مورد لینوکس داشتم ولی به طور کامل با دستورات آن و سیستم فایل های آن آشنا نبودم و در طی پروژه دانش خود در مورد لینوکس را ارتقا دادم. موارد بسیار زیادی بود (مانند دستورهای لازم برای کار با host ها، نصب کردن برنامه ها، نحوه ارسال httprequest با لینوکس، برنامه های لازم برای سهولت کار با ترمینال و یا اتصال به host های مختلف مانند terminus و ...) که من باید در اینترنت جست و جو میکردم تا بتوانم کارهای لازم را در لینوکس انجام بدهم. زمان زیادی برای جست و جو و یادگیری مباحث لینوکس صرف شده.
- **PostgreSQL**: من قبلا با دیتابیس MySQL کار کرده بودم ولی برای این پروژه نحوه کار با این دیتابیس را نیز یاد گرفتم. همچنین از برنامه Pg4Admin به عنوان رابط کاربری با دیتابیس استفاده کردم.
- **Django Async**: قبل از تست اولی که از بنده گرفته شده من آشنایی کاملی با نحوه برنامه نویسی async نداشتم و در یادگیری ای که در پروژه اول داشتم با این نوع برنامه نویسی آشنایی کامل پیدا کردم. در این پروژه نیز با نحوه نوشتن ویوهای جنگو به صورت async آشنا شدم. (از جمله نحوه استفاده از دو تابع sync_to_async و async_to_sync).

۳. Data Base

در این پروژه همانطور که خواسته شده بود از دیتابیس PostgreSQL استفاده شده. برای ایجاد مدل ها و جدول ها نیز از DjangoORM استفاده شده است. ساختار جداول دیتابیس به صورت شکل زیر می‌باشد:



ساختار دیتابیس باید به صورتی میبود که بات بتواند حافظه ای از تمام موارد زیر داشته باشد:

- **User ها**: هر یوزری که برای اولین بار با بات تعامل دارد در دیتابیس ذخیره میشود.
- **Workspace ها و Conversation ها**: هر یوزر ممکن است در workspace ها و یا conversation های مختلفی با بات تعامل داشته باشد. بات باید حافظه ای از این تعاملات داشته باشد.
- **Thread ها**: برای اینکه بات بتواند در صورت وجود event های مختلف در همان thread ای که مکالمه در آن شروع شده است پیام ارسال کند باید حافظه ای از thread_root_id ها داشته باشد تا بتواند در آن thread به کاربر پیام ارسال کند.
- **Webhook Token ها**: برای verify کردن ریکوئست های POST ای که از گیتلب به بات ارسال میشود بات باید حافظه ای از این webhook ها داشته باشد تا بتواند به کاربر پیام ارسال کند.
- **GitLab Token ها**: در صورت استفاده از دستور projects کاربر فقط یک بار از کاربر GitLab token او را دریافت میکند و در موارد استفاده بعدی از این دستور دیگر این توکن را از او درخواست نمیکند (بدیهی است که این پیش فرض در نظر گرفته شده است که برای هر کاربر فقط یک GitLab token میتواند وجود داشته باشد). به دلیل کمبود وقت برای تحویل پروژه فعلا سیستمی برای تغییر و یا حذف این توکن تعبیه نشده است.

۴. WebHook

برای تعبیه webhook از یک view جنگو استفاده شده. برای ارسال پیام به بات، این ویو باید به صورت async نوشته میشد. همچنین برای این ویو باید از تگ csrf_exempt استفاده میشد تا گیتلب بتواند به آن پیام بفرستد.

برای استفاده از DjangoORM در این ویو تمام توابع این ماژول (که باید به صورت sync ران شوند) در داخل یک تابع sync تعبیه شده، سپس دکوراتور sync_to_async بر روی آن تابع اعمال شده و سپس آن تابع داخل ویوی async، await شده است (چراکه استفاده از تابع sync_to_async بر روی تک تک دستورات به دلیلی که نتوانستم متوجه آن بشوم امکانپذیر نبود).

۵. سیستم مکالمه

برای تعبیه سیستم مکالمه بات با کاربر من از ایده ConversationHanlder بات های تلگرام استفاده کردم و برای هر مرحله از مکالمه یک state تعیین کردم. در صورتی که کاربر دستور و یا داده مناسب را وارد کند مکالمه به state بعدی میرود و در غیر این صورت در همان state میماند تا زمانی که کاربر داده مناسب را وارد کند. میتوان برای هر مکالمه یک exit command تعبیه کرد (مانند دستور cancel) تا کاربر هر زمان که میخواهد از مکالمه خارج شود (دستور کنسل به دلیل ضیق وقت تعبیه نشده است). برای هر مکالمه آخرین state ای که نشان دهنده خروج از مکالمه است 1- می‌باشد که در کلاس Constants ذخیره شده است.

۶. Docker And Starting The Project

برای لود کردن پروژه در سرور از داکر استفاده شده است. برای استفاده از پایتون از ایمیج با تگ python:3.8.10 استفاده شده. از آنجایی که من مدت زیادی است از ورژن 3.8 پایتون استفاده می‌کنم برای این پروژه نیز از همین ورژن استفاده کردم چرا که معمولا تغییر ورژن ارور های ناخواسته‌ای را ایجاد میکند و متاسفانه فعلا فرصت تغییر به ورژن 3.9 را نداشته‌ام.

برای استفاده از ایمیج های python و postgres مجبور شدم تا فایل‌های ایمج ها را در سیستم خودم ایجاد کرده و سپس به سرور انتقال دهم چراکه سرور امکان دانلود ایمیج ها از سایت داکر را نداشت.

با انجام دستور docker-compose up در داخل سرور، داکر، جنگو و دیتابیس PostgreSQL را لود میکند. قبل از ران کردن سرور جنگو بات ابتدا دستور python djmanage.py migrate را اجرا میکند تا تمام migration ها انجام شوند. ولی برای ران کردن بات باید خودمان از داخل container جنگو این کار را انجام دهیم.

نحوه استارت کردن پروژه

برای استارت کردن پروژه باید ابتدا دیتابیس و جنگو را با استفاده از docker-compose لود کنیم و سپس در داخل container ای که جنگو در آن ران می‌شود بات را ران کنیم که شامل اقدامات زیر می‌شود:

۱. اتصال به سرور و cd کردن به مسیر /home/alavi/code/Limoo-Test-Bot
۲. ران کردن دستور docker-compose up -d برای لود کردن دیتابیس و جنگو؛
۳. ران کردن دستور docker ps و کپی کردن نام container ای که جنگو در آن در حال ران شدن است؛
۴. ران کردن دستور docker exec -it <container_name or container_id> /bin/bash برای لود کردن ترمینال داخل container جنگو؛
۵. ران کردن دستور python botmanage.py runbot برای ران کردن بات؛

حال هم بات و هم جنگو در حال ران شدن می‌باشند.

به احتمال زیاد راه بهتری برای اتوماسیون ران کردن بات از داخل container وجود دارد و این پروسه جای بهتر شدن دارد.

۷. Django ORM

برای تعامل بات با دیتابیس از ماژول ORM جنگو استفاده شده است. در اینجا مورد بسیار مشکل آفرین ویژگی "async_unsafe" ماژول DjangoORM بود که باعث میشد هنگام استفاده از کد های DjangoORM در داخل بات از تابع sync_to_async استفاده کرده و سپس آن تابع را await کرد. با وجود تلاش های زیاد برای استفاده از این تابع برای دستورات sync ماژول DjangoORM، و به دلیل کمبود وقت برای تحویل پروژه، راه دیگری جز غیرفعال کردن ویژگی async_unsafe این ماژول پیدا نکردم. این کار با انجام کارهای لازم که در این لینک توضیح داده شده انجام شد.

یک راه حل برای استفاده از کد های DjangoORM در داخل توابع async جمع کردن تمام کد های این ماژول در یک تابع sync و سپس استفاده از دکوراتور sync_to_async می‌باشد ولی ساختار کد بات به نحوی بود که این کار به سادگی قابل انجام نبود چراکه این ساختار از قبل با تلاش زیاد ایجاد شده بود و تغییر آن به زمان زیادی نیاز داشت. (توجه کنید که call کردن این دکوراتور روی تک تک توابع مربوط به DjangoORM نتیجه بخش نبود).

به نظر می‌رسد که غیرفعال کردن ویژگی async_unsafe ماژول DjangoORM در صورت scale out شدن پروژه مشکلاتی ایجاد کند و لذا باید در صورت ادامه‌ی پروژه این مشکل حل شود.

۸. Source Control

در push کردن تغییرات سعی شده commit ها به صورت اصولی پیاده شوند. هر commit نه باید بیش از اندازه بزرگ باشد و نه بیش از حد کوچک و ناچیز. چراکه در صورت اتفاق افتادن هر کدام از این موارد، در صورت وجود افراد دیگر در پروژه مشکلات زیادی در سینک شدن افراد با کد ایجاد خواهد شد.

تمام تغییرات در یک branch اعمال شده اند و در صورت وجود افراد دیگر در پروژه این کار اصولی نیست. ولی برای راحتی کار و تسریع در انجام پروژه این کار انجام شده است.

۹. Code

- در نوشتن کد سعی شده تا کد کاملا تمیز و اصولی نوشته شود.
- اصل Don’t Repeat Yourself رعایت شده است.
- سعی شده برای قسمت هایی از کد که دارای ابهام هستند کامنت ها و توضیحات لازم نوشته شود.
- برای قسمت هایی از کد که امکان بهتر شدن داشتند TODO نوشته شده تا در صورت ادامه پروژه این کار ها انجام شوند.
- سعی شده در داخل کد هیچ موردی hard-code نشود و تمام constant ها در داخل فایل src/config/settings به صورت کلاس بندی شده نوشته شده‌اند.
- ماژول بات داخل ماژول جنگو تعبیه شده و source code آن به عنوان یکی از اپ های جنگو داخل فولدر src قرار دارند.
- به هنگام ران شدن بات، بات ابتدا تنظیمات جنگو را config میکند تا بتواند از ماژول ORM آن استفاده کند. این کار با استفاده از تابع config_django انجام شده که داخل فایل src/utilities/util میباشد.
- کد هایی که احتمال استفاده آنها در کل پروژه وجود داشت داخل فایل src/utilities/util قرار داده شده‌اند. همچنین کد هایی که دیگر احتمال استفاده از آنها داده نمی‌شد و یا کد های archive شده که ممکن است در آینده به آنها نیاز داشته باشیم داخل پوشه src/utilities قرار داده شده‌اند.
- بهتر بود که فایل gitlab_async_api نیز داخل همین فولدر قرار داده شود ولی به دلیل استفاده از این ماژول در جاهای مختلف و همچنین تسریع در تحویل پروژه این کار انجام نشد.
- داخل فولدر utilities یک فایل به نام gitlab_api وجود دارد. این ماژول در ابتدای پروژه اول ایجاد شده بود و کاربردی ندارد. همچنین باتِ نمونه با اسم sample_bot و کد های لازم برای install کردن بات در سیستم عامل ویندوز در فایلی با نام install در این فولدر قرار دارند.

کلام آخر

امیدوارم توضیحاتی که نوشته شده‌اند کامل و جامع باشند و امیدوارم توانسته باشم کاری تمیز و کامل تحویل داده باشم. از بابت تاخیر در تحویل دادن این فایل عذرخواهی می‌کنم چراکه نخواستم چیزی از قلم بیافتد.

بنده برای تحویل این پروژه ساعتی در روز نبوده که کار نکرده باشم و به هیچ وجه تمایل به تحویل دادن کار ناقص نداشتم. من علاقه زیادی به کارم دارم و در صورت همکاری با شما نیز انشاالله همین اخلاق کاری را ادامه خواهم داد.

در مجموع این پروژه تجربه جالبی برای بنده بود و از شما بابت این چالش متشکرم.

با تشکر از شما

محمد علوی

۰۹۱۴۴۷۴۹۱۶۸