УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Облачные и туманные вычисления»

Проект

Разработка Telegram бота "InterviewBot"

Студент

Кузнецов М. А.

P34131

Преподаватель

Перл О. В.

Санкт-Петербург, 2023 г.

Содержание

Сведения о приложении 2

Роли 3

UseCase диаграмма 3

Стек разработки 4

Архитектура приложения 4

Диаграмма структуры базы данных 5

Целевая нагрузка системы 6

Масштабирование 7

Сведения о приложении

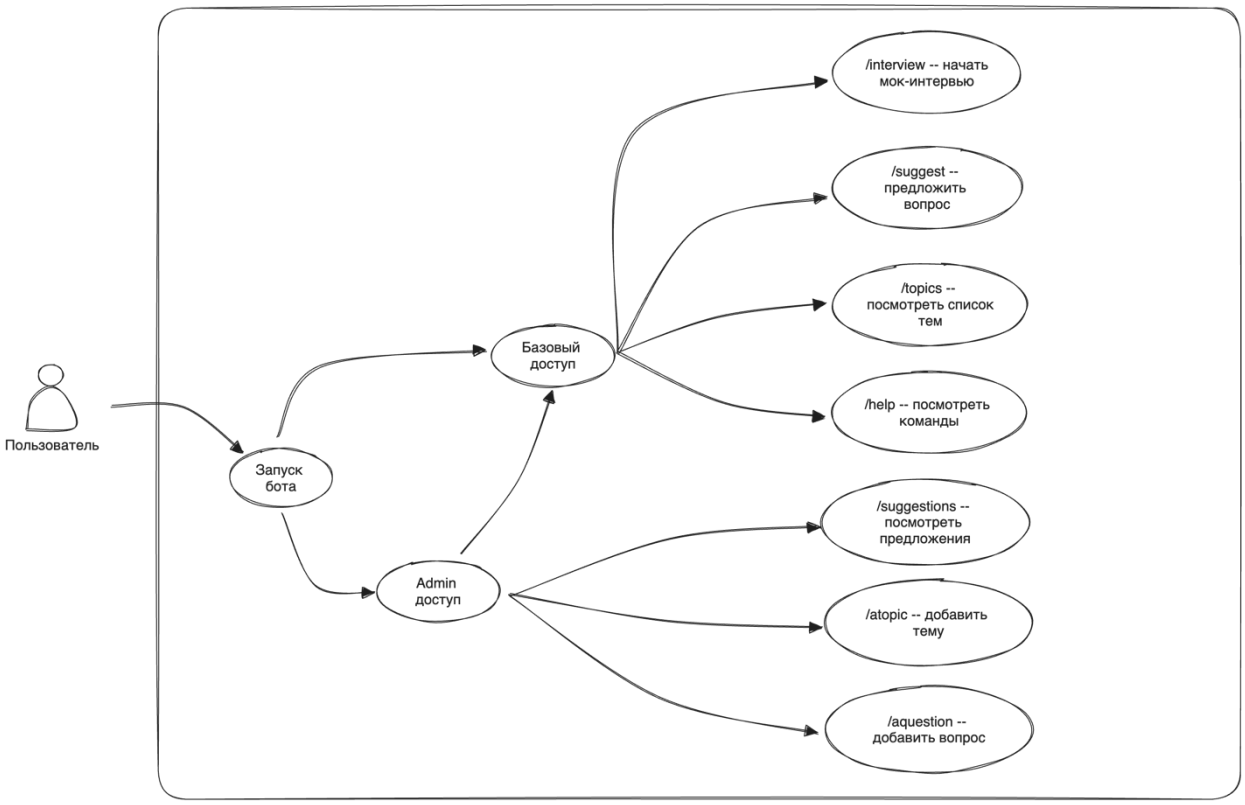
Телеграмм-бот InterviewBot, который в себе будет содержать вопросы по разным топикам и темам, относящихся к IT в целом. Например, градация вопросов по языкам, алгоритмическим задачам и т. п. Возможны также и более общие темы -- устройство Linux, System Design и т. д. Еще, как возможность, пользователи сами смогут расширять существующую базу вопросов, то есть, через интерфейс бота добавлять свои вопросы. Другими словами – все это является некой совместной расширяемой базой знаний.

Роли

На данный момент предусматривается две основные роли:

1. Гость – это любой человек, который может найти бота через поиск, перейти по ссылке, взаимодействовать с ним. Данная роль разрешает использование основных и базовых команд бота. Назначается по-умолчанию каждому новому пользователю.
2. Администратор – для получения данной роли требуется admin-key. Данный ключ генерируется автоматически при каждом запуске экземпляра бота на сервере. С помощью команды /adminkey можно авторизоваться в боте и получить дополнительный доступ к другим командам, которые позволяют управлять вопросами и темами, просматривать предложения.

UseCase диаграмма

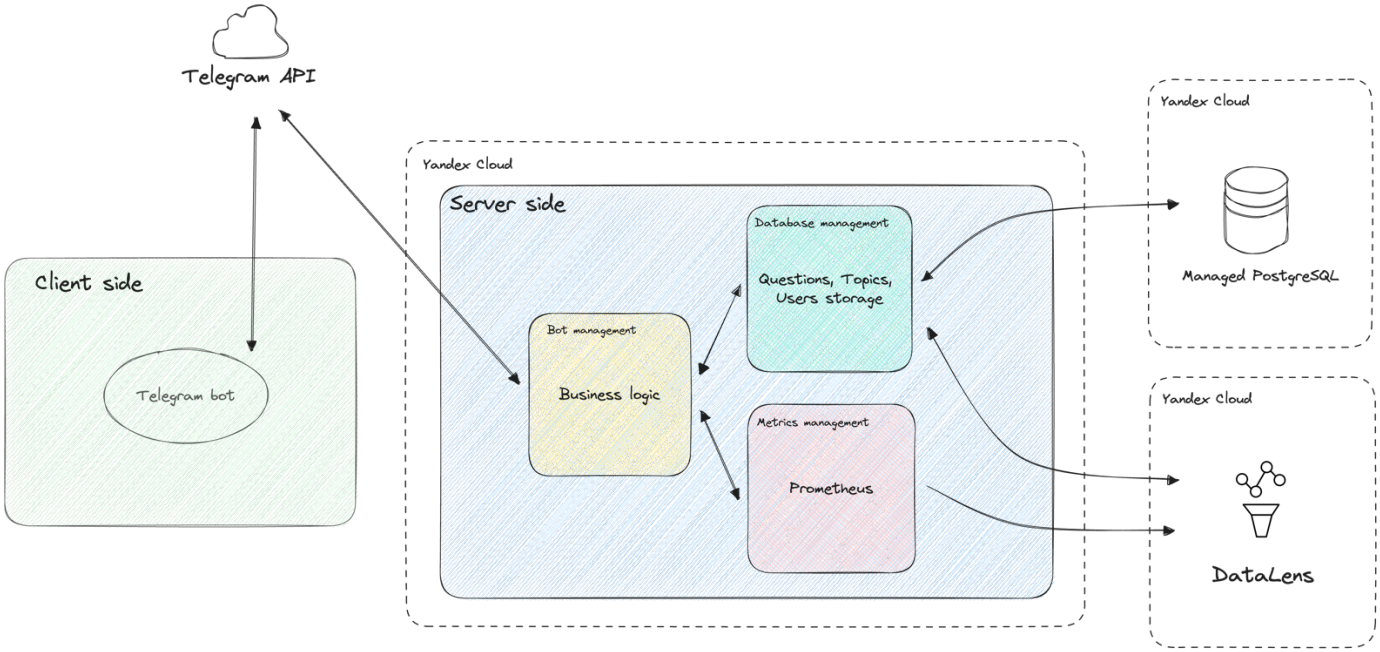


Стек разработки

При разработке используются следующие технологии и зависимости:

- Go – в качестве главного языка программирования бота.
- Telegram Bot API – библиотека для удобного взаимодействия с API Telegram.
- PostgreSQL – open-source база данных, которая предоставит весь необходимый функционал для хранения данных.
- pgx – библиотека для Go для работы с базой данных PostgreSQL.

Архитектура приложения

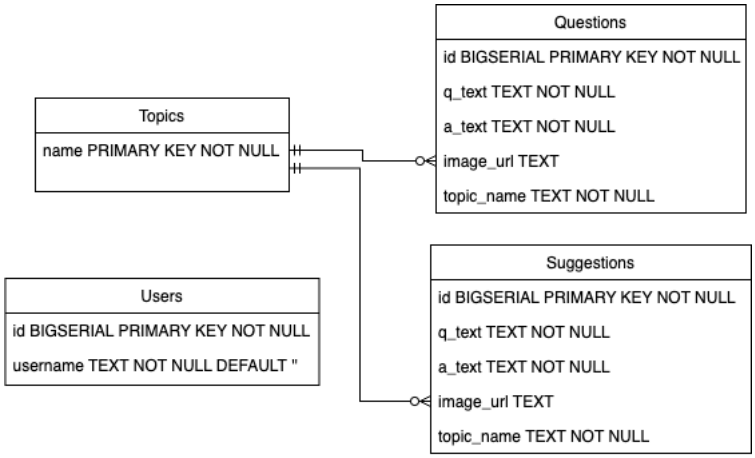


Архитектура приложения подразделяется на три слоя:

1. Клиентская часть – интегрирована через интерфейс бота Telegram. Управляется через Telegram API.
2. Серверная часть – реализация логики бота. Представляет собой монолитное решение, в котором реализована бизнес-логика, отвечающая за все необходимые операции и функционал, управление хранением данных и сбором метрик. Метрики и работа с БД также взаимодействуют с сервисами Yandex Cloud.

3. Внешние сервисы – это подключаемые сервисы Yandex Cloud, такие как Managed PostgreSQL, DataLens. А также образ VM, на котором запущено приложение. В DataLens также есть поддержка работы с PostgreSQL, что позволяет подключить последний для анализа.

Диаграмма структуры базы данных



Важное уточнение 1: зависимости таблиц Topics-Suggestions и Topics-Questions установлены напрямую в коде. Для PostgreSQL это две несвязанные таблицы, topic_name у таблиц Questions и Suggestions заполняется из кода и содержит соответствующее имя в таблице Topics. Это сделано, чтобы избежать жесткой связности сущностей.

Важное уточнение 2: каждая из таблиц содержит дополнительных два поля:

1. created_at – указывает на время создания записи.
2. updated_at – указывает на время последнего обновления записи.

Важное уточнение 3: таблица Users на данном этапе выполняет роль хранилища всех пользователей, которые взаимодействовали с ботом хотя бы раз. Эти данные могут быть полезны для анализа. В перспективе данная сущность может быть использована для поддержки авторизации.

Важное уточнение 4: на данном этапе ради тестовых целей было принято решение хранить изображения вопросов на локальном root диске. image_url указывает на путь к изображению на диске. При перспективе развития можно будет использовать подключаемое хранилище.

Topics:

- name – название топика (Java, Go, System Design, …)

Users:

- id – уникальный id пользователя
- username – имя пользователя из Telegram

Questions (содержит существующие вопросы):

- id – уникальный id вопроса
- q_text – текст вопроса
- a_text – текст ответа
- image_url – путь к изображению
- topic_name – название топика

Suggestions (содержит предлагаемые вопросы):

- id – уникальный id вопроса
- q_text – текст вопроса
- a_text – текст ответа
- image_url – путь к изображению
- topic_name – название топика

Нагрузка системы

Для расчета потенциального количества пользователей, с которыми может справиться сервис, будем использовать следующую формулу:

```
(number of CPU cores / Average Response Time in seconds) * 60 * User Click Frequency in seconds = Maximum simultaneous users
```

Number of CPU cores – 0.4 vCPU

Average Response Time in seconds – 0.5

User Click Frequency in seconds – 60 (допустим, по минуте на вопрос)

Подставляя наши данные получим 2880. Результат, само собой, не является достоверно верным, но соответствует примерным ожиданиям для выбранной конфигурации сервера.

Масштабирование

Рассмотрим потенциальные стороны:

1. Увеличение числа пользователей, и как следствие, нагрузки – мы можем использовать уже более распределенную обработку поступающих сообщений на отдельных экземплярах приложения. Другими словами, на разных машинах поднимаются дополнительные инстансы, которые помогут равномерно распределить поступающую нагрузку. Учитывая особенности работы с API Telegram, простым балансировщиком тут не обойтись, так как каждый из экземпляров будет получать в обработку одновременно одно и то же сообщение. Для решения данной проблемы можно добавить брокера сообщений, например, RabbitMQ, который будет аккумулировать в себе поступающие запросы из API Telegram и отдавать их на обработку нашим экземплярам приложения.
2. Увеличение объема данных в системе – мы можем улучшить конфигурации наших managed PaaS сервисов, сделать реплики. В случае с картинками мы будем использовать внешнее расширяемое хранилище данных.
3. Уменьшение числа пользователей – от нас никаких действий не требуется, так как у нас выбрана наименьшая конфигурация из возможных.
4. Уменьшение объема данных в системе – аналогично п. 3.