**Описание базы данных**

В качестве СУБД используется MySQL, которая размещена на сервере вместе с самим ботом.

Вся информация хранится в 5 таблицах:

* data (таблица для кэширования информации с внешних ресурсов)
* distribution (вспомогательная таблица для формирования сообщений и рассылки, подробнее о ней чуть позже)
* phrases (таблица, в которой хранится список из 100 цитат, из которого выбирается цитата дня)
* users (таблица, содержащая информацию о пользователе: ID, имя, город, время отправки сообщения, выбранные категории, уведомления)
* weather (таблица, где представлен список городов с прогнозом погоды для каждого, т.е. также происходит кэширование информации)

**Таблица data:**

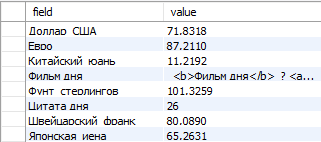
CREATE TABLE `data` (

`field` varchar(50) CHARACTER SET utf8 NOT NULL,

`value` varchar(500) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

PRIMARY KEY (`field`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci



Вся информация с внешних источников обновляется в базе данных с заданной периодичностью, а не запрашивается при необходимости отправки пользователю у внешних ресурсов, такой подход обеспечивает надежность системы, так как в базе всегда будет содержаться какая-то информация (хоть и не слишком актуальная) в случае, если внешние сервисы будут недоступны.

Для определения фильма дня используется суточный рейтинг сайта «Кинопоиск» (<https://www.kinopoisk.ru/popular/films/>). При обновлении текущего поля сразу формируется полноценный текст с указанием названия фильма, страны, жанра, рейтинга и описания. Также используется HTML форматирование текста для выделения заголовка и создания гипертекста.

Информация о курсах валют получается с официального сайта ЦБ РФ (<http://www.cbr.ru/scripts/XML_daily.asp>).

Номер цитаты дня выбирается случайным образом из всего списка цитат.

Информация таблицы data обновляется раз сутки в 00:00, так как более частого обновления данной информации не требуется.

**Таблица distribution:**

CREATE TABLE `distribution` (

`id` varchar(10) CHARACTER SET utf8 NOT NULL,

`message` varchar(1500) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`notifications` tinyint(4) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci

Данная таблица во многом является вспомогательной, но необходима из-за механизма, по которому бот отправляет сообщения. В боте учтена большая пользовательская нагрузка, поэтому, чтобы все сообщения приходили пользователям вовремя и как можно с меньшей задержкой, формирование текста сообщения для каждого пользователя (что является достаточно трудоемким процессом) начинается за 5 минут до конца часа (бот отправляет сообщения каждый час в 00 минут). После того как текст сообщения сформирован, он заносится в данную таблицу с указанием ID пользователя и информации об уведомлениях.

Затем ровно в 00 минут каждого часа бот получает все записи о рассылке из базы данных и начинает отправлять сообщения пользователям. При таком подходе максимальная задержка в рассылке сообщений составляет около 15-20 секунд.

**Таблица phrases**

CREATE TABLE `phrases` (

`id` tinyint(4) NOT NULL,

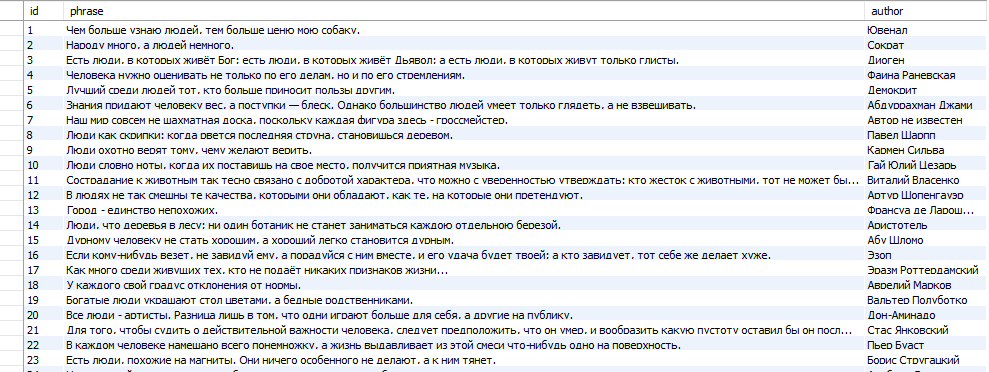
`phrase` varchar(255) NOT NULL,

`author` varchar(255) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `phrase\_UNIQUE` (`phrase`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8



В таблице хранится список цитат с указанием автора и номера цитаты.

**Таблица users**

CREATE TABLE `users` (

`id` varchar(10) NOT NULL,

`name` varchar(45) NOT NULL,

`city` varchar(45) DEFAULT NULL,

`times\_of\_day` tinyint(4) NOT NULL,

`exchange\_rates` tinyint(4) NOT NULL DEFAULT '0',

`weather\_city` tinyint(4) NOT NULL DEFAULT '0',

`quote\_of\_the\_day` tinyint(4) NOT NULL DEFAULT '0',

`movie\_of\_the\_day` tinyint(4) NOT NULL DEFAULT '0',

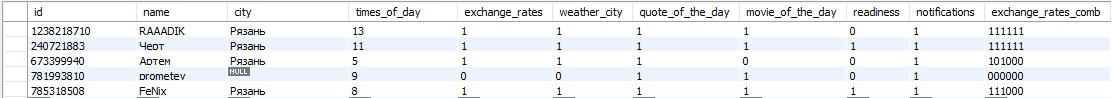
`readiness` tinyint(4) NOT NULL DEFAULT '0',

`notifications` tinyint(4) NOT NULL DEFAULT '1',

`exchange\_rates\_comb` varchar(45) NOT NULL DEFAULT '000000',

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8



В данной таблице хранится информация по всем пользователям, а также их настройкам. Так как выбранная СУБД не поддерживает логический тип, в качестве флагов (включена/не включена отправка определенной информации) используются значения 0 и 1. Выбранные пользователем валюты хранятся в виде последовательности 0 и 1.

Важным параметром является столбец readiness, который показывает, что пользователь закончил настройку, если значение установлено в 1. Перед установкой данного параметра в 1 бот проверяет, что пользователь выбрал хотя бы одну категорию интересующей его информации, иначе настройку рассылки закончить будет невозможно, так как отправлять пользователю пустое сообщение нет смысла.

Столбец notifications отвечает за отправку сообщений в беззвучном режиме (0, если беззвучно). Беззвучные сообщения являются встроенной возможностью телеграмма, которая позволяет не отвлекать пользователя. В боте пользователь самостоятельно настраивает данный параметр.

**Таблица weather**

CREATE TABLE `weather` (

`city` varchar(45) NOT NULL,

`temperature` varchar(10) NOT NULL,

`wind\_speed` varchar(10) NOT NULL,

`wind\_direction` varchar(10) NOT NULL,

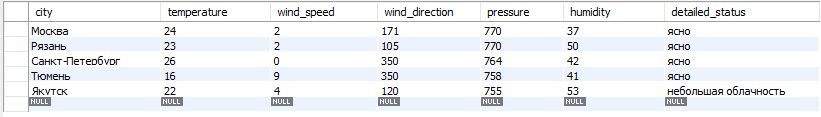
`pressure` varchar(10) NOT NULL,

`humidity` varchar(10) NOT NULL,

`detailed\_status` varchar(45) NOT NULL,

PRIMARY KEY (`city`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8



В данной таблице хранятся города и прогноз погоды для них. Для каждого города указывается: температура, скорость ветра, направление ветра (в градусах, которые при отправке бот переводит в более понятный для пользователя вид), давление, влажность и статус погоды. Когда пользователь вводит город, которого нет еще в базе данных, он добавляется в эту таблицу. Каждые час бот обновляет информацию о погоде для каждого города, используя сервис OpenWeatherMap (<https://openweathermap.org/api>).

**Бот**

Бот написан на языке Python с использованием библиотеки aiogram, которая является полностью асинхронным фреймворком для Telegram Bot API. Благодаря реализованной асинхронности, каждое действие пользователей (и каждая функция соответственно) обрабатывается в отдельном потоке, что обеспечивает высокое быстродействие и надежность, так как в случае возникновения ошибки у одного пользователя, это не приведет к тому, что «упадет» весь бот.

Код бота разделен на два файла, первый файл – основной код, предназначенный для обработки действий пользователей, второй файл – конфигурационный, он содержит в себе все тексты сообщений и общие переменные (их нецелесообразно указывать в основном файле, так как это ухудшит читаемость), настройки подключения к БД (для безопасности данных в случае передачи кода кому-то).

Для работы бота создается диспетчер, он регистрирует функции-обработчики действий пользователя (хэндлеры), дополнительно ограничивая перечень вызывающих их событий через фильтры. После получения очередного события с серверов телеграмма, диспетчер выберет нужную функцию обработки, подходящую по всем фильтрам.

При запуске бота дополнительно создаются два потока, которые являются таймерами, вызывающими необходимые функции в определенное время.



Первый из них предназначен для рассылки сообщений пользователей (принцип рассылки был описан выше). Второй необходим для обновления информации, связанной с фильмом и цитатой дня, а также курсами валют.

Помимо этого, каждый час (DELAY = 3600) бот запускает функцию обновления прогноза погоды для городов. В данном случае точное время обновления данных не важно, поэтому обновление происходит каждый час от времени запуска бота.



Таймеры работают по следующему принципу: сначала они получают текущее время в секундах, затем отнимают это время от определенного значения, тем самым получают время в секундах, которое они должны «поспать». После «пробуждения» вызывается необходимая функция, и после ее выполнения алгоритм повторяется.

Для формирования многих текстов разделов используются f-строки, которые представляют собой общий шаблон текста, в который затем подставляются значения в зависимости от пользователя.

Порядок описания функций представлен снизу-вверх.

**Функция weather\_update**

Предназначена для обновления прогноза погоды. Бот сначала получает список городов из БД, а затем в цикле обращается к функции get\_weather\_information, которая возвращает прогноз погоды для города, после чего полученная информация заносится в БД.

**Функция data\_retrieval (+ getDiscription, getFilm, req)**

Предназначена для обновления курсов валют, фильма и цитаты дня. Для обновления курсов валют бот обращается к XML файлу сайта ЦБ РФ и получает из него необходимые валюты по их индексам. Для обновления цитаты дня бот рандомно выбирает значение от 1 до 100. Для обновления фильма дня используется парсинг сайта «Кинопоиск», парсер находит необходимые поля страницы, после чего формируется готовый текст. Вся собранная информация в конце заносится в БД.

**Функция sending\_messages**

Предназначена для рассылки сообщений пользователям. Сначала бот получает из БД ID пользователей, сообщение для каждого и информацию о звуке уведомления для каждого из них. После этого в цикле происходит рассылка сообщений всем пользователям.

**Функция message\_formation**

Предназначена для формирования сообщений пользователям. В ней из БД берется информация о ID пользователей, которым нужно отправить сообщение, их конфигурация настройки бота, а также вся информация для информационных категорий. Определяется время суток. Затем в цикле бот формирует сообщение, основываясь на флагах (выбрана/не выбрана категория пользователем) и заносит полученное сообщение в БД.

**Функция telegram\_API\_error**

Предназначена для обработки ошибок, связанных с Bot API. Таким образом, если одна и та же непредвиденная ситуация будет возникать в различных хэндлерах, то её обработка будет проходить в данном хэндлере ошибок, что сокращает код и повышает его читабельность.

**Функция send\_text**

Предназначена для удаления лишних сообщений, отправленных пользователем, что позволяет исключить ненужные сообщения в диалоге с ботом.

**Функция delete\_message**

Предназначена для удаления сообщений рассылки при нажатии соответствующей кнопки.