Кэш

Кэш и его инвалидация — очень сложные темы, поэтому вкратце опишем варианты реализации кэша в чатботах, их плюсы и минусы.

Кэш — это временная память бота. Данные, которые скорее всего не поменяются в ближайшем будущем, которые вы решаете не генерировать раз за разом, а сохранить один раз и пользоваться ими. **Кэш — это временные данные**. Если ваши данные постоянны, их проще хранить в JSON-файле, а не в кэше. Меню пиццерии — не постоянные данные. Меню может меняться: менеджеры могут добавлять и убирать товары на Moltin, менять их стоимость, двигать между категориями.

Инвалидация кэша — это процесс его **обновления**. У вас должны быть строгие правила, в каких случаях кэш пора обновлять. Иначе будет не понятно, что в кэше сейчас, и будет большая путанца. Ниже несколько примеров, как его инвалидировать.

1. Обновление, если кэш лежит дольше часа — это самый первый и очевидный способ. Давайте создадим функцию `get_menu`, которая будет доставать меню из Redis, проверять дату создания, если меню лежит в Redis дольше часа — вызывается функция `create_menu` и полученное меню пишется в Redis:

```
def get_menu():
    cached_menu = database.get("menu")
    time_diff = datetime.now() - cached_menu['created_at']
    if time_diff.hours > 0:
        menu = create_menu()
        database.set("menu", menu)
    else:
        menu = cached_menu
    return menu
```

Теперь ждать придётся не всем пользователям, а только нескольким, которым посчастиливилось написать боту как раз тогда, когда кэш просрочился. **Такая схема плоха**, потому что если менеджер внесёт правки в Moltin, ему придётся **ждать целый час**, чтобы проверить изменения в боте.

2. Если кэш лежит дольше 5 минут — такой же способ, но уменьшаем время хранения кэша до 5 минут. Если менеджер меняет что-то в Moltin, он может подождать. Но от такого кэша мало толку: если боту пишут примерно раз в 5 минут, каждому пользователю придётся ждать, будто кэша и нет.

3. Генерация кэша раз в 5 минут — казалось бы, то же самое, но нет. Теперь вы пишете отдельный скрипт, который раз в 5 минут генерирует меню и кладёт его в БД:

```
def cache_menu():
    menu = create_menu()
    database.set("menu", menu)
```

А бот только достаёт меню из БД, т.е. работает **только на чтение**:

```
def get_menu():
    cached_menu = database.get("menu")
    return cached_menu
```

Теперь на сервере можно запускать бота отдельно, а постоянное создание меню и его кэширование — отдельно. Для запуска скриптов раз раз в несколько минут, часов или дней есть специальный инструмент — cron. Для помощи с `cron` также есть хороший сайт, который хочется порекомендовать: crontab.guru.

У этой схемы всё ещё есть минусы: инвалидация кэша происходит не в тот же момент, что и обновления данных в Moltin. Для этого есть ещё несколько примеров:

4. Webhook/polling Moltin. Сайт не поддерживает такой функционал, но в теории он мог бы сам сообщать вам по API, что данные поменялись и пора обновить кэш.

5. is_changed. Такой ручки у API Moltin тоже нет, но можно было бы спрашивать "А меню товаров поменялось?". Если поменялось — генерируем новое.