

# **Системы хранения данных в Почте VK WorkSpace**

**Документ для администраторов**

# Оглавление

---

Назначение документа	3
Термины и определения	3
Таблица упомянутых сервисов	3
Система хранения почтовых ящиков	4
Хранение профиля пользователя и структуры ящика	5
Tarantool xlog	7
Zepto	7
Логика хранения письма	7
Как хранятся вложения	8
Процесс чтения писем	9

# Назначение документа

---

В документе содержится информация о принципах работы системы хранения данных.

## Термины и определения

---

Скелет письма — тело письма со ссылками на вложения.

Индекс почтового ящика (main-индекс) — снимок состояния почтового ящика и журнал изменения состояний.

## Таблица упомянутых сервисов

В документе упоминаются следующие сервисы Почты:

Название	Описание
<b>Mescalito</b>	Хранилище индексов писем
<b>Tarantool xlog (xtaz)</b>	Хранилище динамично меняемых данных почтовых ящиков
<b>Zubr</b>	Прокси для запросов профилей пользователей
<b>Zepto</b>	В контексте этого документа — хранилище снимотов
<b>Deliveryd</b>	Сервис покладки писем
<b>Ameli</b>	Сервис сборки писем
<b>Arbuzapi</b>	gRPC-прокси, изолирующее MySQL базы данных
<b>Profapi</b>	gRPC API для пользовательского профиля

# Система хранения почтовых ящиков

В структуре почтового ящика есть:

- Письма.
- Папки.
- Структура папок.

Письмо состоит из темы, тела (текста письма) и вложений.

В Почте используется принцип разделения писем на несколько слоев:

- Индексы.
- Письма.
- Вложения.

Каждый слой хранится отдельно от другого.

На схеме ниже показаны связи данных и компоненты, в которых данные хранятся и обрабатываются:



Учетная запись пользователя является первичной сущностью. Почтовый ящик создается только после создания пользователя и состоит из информации о пользователе:

- ФИО.
- Телефон.
- Фото профиля.

- Ссылка на почтовый ящик.
- Другая информация.

По ссылке хранится структура почтового ящика. Данные обрабатываются сервисом Zubr.

Структура хранения начинается с индекса почтового ящика. В индексе хранится:

- Структура папок.
- Списки писем.
- Ссылки на письма.
- Метаинформация о письмах: флаги о прочтении, важность письма - хранятся в main-индексе.

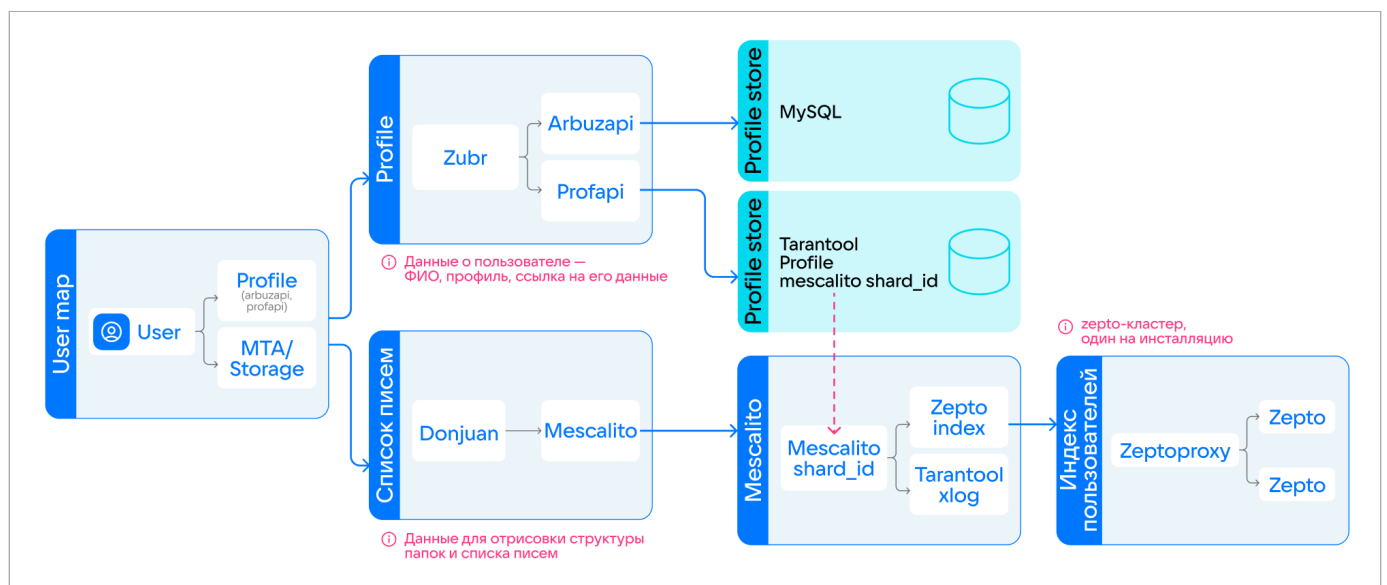
Эти данные обрабатываются сервисом Mescalito (ниже будет более подробная информация о сервисе).

Само письмо подвергается разделению для хранения. Сначала от письма отделяются все вложения. После этого остается текстовая часть письма со ссылками на вложения (скелет письма). Скелеты хранятся в Zepto.

Вложения находятся отдельно в сервисе Mailcloud — условное название хранилищ вложений, где хранятся blob-объекты. То есть из письма достаются вложения, а на их место вставляется внутренняя ссылка на вложения. Разделением писем на скелеты и вложения занимается сервис Deliveryd.

## Хранение профиля пользователя и структуры ящика

Более детальная схема хранения профиля пользователя выглядит следующим образом:



При переходе пользователя в свой почтовый ящик происходит следующее:

1. Через сервис **Zubr** отправляется запрос на **Arbuzapi** и **Profapi**.

2. С помощью них вычитываются данные о профиле из Tarantool и в ответе приходит mescalito shard\_id (он же xtazik). В MySQL хранится таблица `mPOP.xtazik`, которая кешируется в памяти сервиса Zubr. В таблице для каждого шарда хранится информация о том, какие Mescalito его обслуживают. Полученная информация возвращается в Zubr.
3. Сервис **Donjuan** отправляет запрос на получение списка писем и структуры папок в Mescalito.
4. **Mescalito** можно сравнить с СУБД, каждый пользователь это отдельная независимая БД, состоящая из 2х частей:
  - Снапшот (хранится в zepto).
  - xlog (хранится в tarantool).**xtaz-tarantool** — это точка входа в индексы, в нем хранится:
  - Массив zepto-id частей (чанков) снапшота.
  - Текущий xlog для main-индекса.
  - Текущий xlog для opt-индекса.
  - Очереди для оффлайн-утилит (чеки/компакты).
5. Mescalito запрашивает данные из **Tarantool xlog** (в Deployer называется **Xtaz**). В Tarantool хранится журнал действий пользователей. Если в xlog нет полного состояния почтового ящика, значит данные были перенесены в Zepto. В таком случае Mescalito самостоятельно обратится в Zepto, получит снапшот, наложит на него журнал действий пользователя и передаст обратно в сервис Donjuan.
6. По итогу процесса пользователь получит отображение своего профиля и список писем.

### Если ящика нет в кеше Mescalito

Mescalito сделает следующее:

1. Отправит запрос в Tarantool xlog, получит xlog и массив ID снапшотов в Zepto.
2. Скачает снапшот из Zepto.
3. Распакует снапшот.
4. Построит индексы и применит к ним xlog.
5. Обработает запрос и вернет ответ.

### Если ящик в кеше Mescalito

Mescalito сделает следующее:

1. Отправит запрос в Tarantool xlog.
2. В ответ получит изменения, о которых ещё не знает.
3. Применит изменения к текущему состоянию ящика в памяти.
4. Обработает запрос и вернет ответ.

# Tarantool xlog

Журнал xlog хранит журнал изменения состояния почтового ящика и список идентификаторов снимотов почтового ящика в Zepto. Представим, что у нас есть список писем, эти все письма разложены по папкам, у каких-то писем есть установленные флаги. Например, информация (флаги) о том, прочитано письмо или нет. Пользователь может устанавливать и снимать у писем флаги и перемещать их из папки в папку. Журнал всех этих изменений хранит **Tarantool xlog**. БД Tarantool выбрана для ускорения обработки запросов.

## Zepto

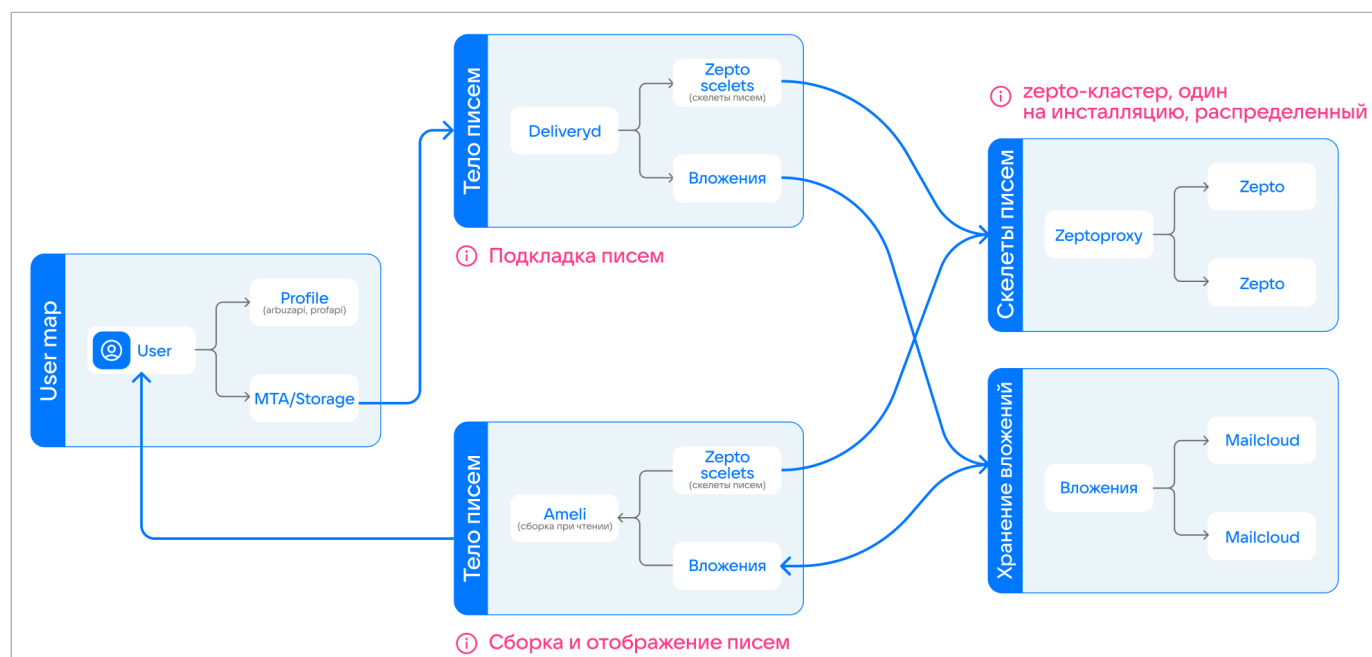
Периодически с почтовых ящиков снимается снимот и сохраняется как некое персистентное состояние на момент времени. Снимоты хранятся в Zepto, на случай если состояние почтового ящика давно не менялась, то информация уже не нужна в оперативной памяти. Данные в Tarantool накапливаются относительно последнего снимота, содержащегося в Zepto.

## Логика хранения письма

Чтобы пользователь смог прочитать письмо, Почте нужно выполнить 2 основных действия:

1. Сохранить, когда оно пришло.
2. Прочитать, когда пользователь решил его открыть.

Схема хранения и чтения писем выглядит так:



Рассмотрим метод подкладки письма:

1. Все письма приходят в сервис **Deliveryd** (delivery-агент). Он занимается покладкой письма в систему хранения данных.
2. Письма разделяются на 2 части: скелет письма (без вложений) и вложения. Deliveryd выделяет из письма вложения и сохраняет в хранилище (Mailcloud).
3. Deliveryd получает уникальный ID вложения (вложения хранятся в одном экземпляре).
4. Затем вставляет ID вложений в скелет в те места, где они были в изначальном письме.
5. Скелет с ID вложений Deliveryd сохраняет в Zepto.

## Как хранятся вложения

- Вложения хранятся в отдельном хранилище, в дедуплицированном виде. Сохранение в отдельном хранилище зависит от настроек парсера на серверах с Deliveryd (опция `cloud_min_size`).
- Если вложение занимает более 64 Кб, то оно удаляется из скелета и заменяется ID вложения.
- Если вложение занимает менее 64 Кб, то оно хранится в скелете.
- Если вложение пришло как часть тела письма (inline вложение), то парсер все равно извлечет вложение. В первую очередь определяется `content-type`, все, что не `text/*` и больше минимального размера, парсер попытается отправить в хранилище вложений.

Вложения находятся в двух системах хранения:

- **blobcloud** — временное хранилище вложений Почты.
- **stz** — постоянное хранилище вложений Почты.

Правило удаления вложений из **blobcloud**:

- Клиенты передают данные с временем хранения (TTL), по умолчанию время равно пяти часам. Максимально возможное время хранения — 35 дней.
- Если в течение этого времени клиенты не придут и не выполнят `touch` с новым временем хранения, то вложение из blobcloud удаляется.

Вложения хранятся в хранилищах в виде отдельных файлов. Имя задается по их контрольной сумме.

При расчете контрольной суммы `sha1` используются следующие данные:

- соль (`mgCloud`) + тело\_файла + размер\_файла
- Имя файла не учитывается. Если изменить только имя файла и добавить его в хранилище вложений, то файл не задублируется.

### Примеры хранения вложения

Во временном хранилище:

```
blobcloud1/mailcloud/1/A8/43/A8432866BCD7DDCA9F5804150255A8DBA6BB817B
```

В постоянном хранилище:

```
stz1/mailcloud/1/A8/43/A8432866BCD7DDCA9F5804150255A8DBA6BB817B
```

У каждого файла в хранилище есть 12-байтный заголовок, в этом заголовке находятся три числа (`uint32`):

- Тип хэша — `const 0x72ba0388`, чтобы знать, как интерпретировать данные в файле.
- Длина оригинального файла.
- Флаги. Значение 0 — если файл не сжат, значение 1, если хранится в сжатом виде.

Дальше идет либо тело оригинального файла как есть (если флаг 0), либо gzip-сжатый файл, но без gzip заголовка.

Чтобы посмотреть оригинал сжатого файла, преобразуйте его с помощью команды:

```
(printf "\x1f\x8b\x08\x00\x00\x00\x00\x67"; dd if=A8432866BCD7DDCA9F5804150255A8DBA6BB817B  
bs=1 skip=12 2>/dev/null) | gunzip > result
```

Для просмотра несжатого файла необходимо удалить первые 12 байт файла.

## Процесс чтения писем

---

Чтение письма происходит таким образом:

1. Запрос на чтение письма приходит в сервис Ameli — этот сервис занимается сборкой письма.
2. Ameli обращается в Zepto и вычитывает скелет письма.
3. Видит, что в скелете есть ссылки на вложения.
4. Идет в хранилище вложений и забирает их.
5. Вставляет вложения в скелет письма. Так формируется объект письма.
6. Письмо передается на соответствующие сервисы для передачи пользователю.

Если при запросе в Ameli не передали `zepto_id` (основной режим), то Ameli обратится в Mescalito и получит структуру частей (`parts`) письма и `zepto_id`. И из этой структуры частей получит информацию о том, что в письме есть облачные вложения.

Если в запросе передали `zepto_id` (редко используемый режим), тогда Ameli:

1. Скачает скелет.
2. Разберет его.
3. Сгенерирует структуру частей, как если бы она получила ее из Mescalito.
4. Начнет обрабатывать запрос как обычно.

 Автор: Груздев Никита

 27 апреля 2026 г.