Календарь VK WorkSpace

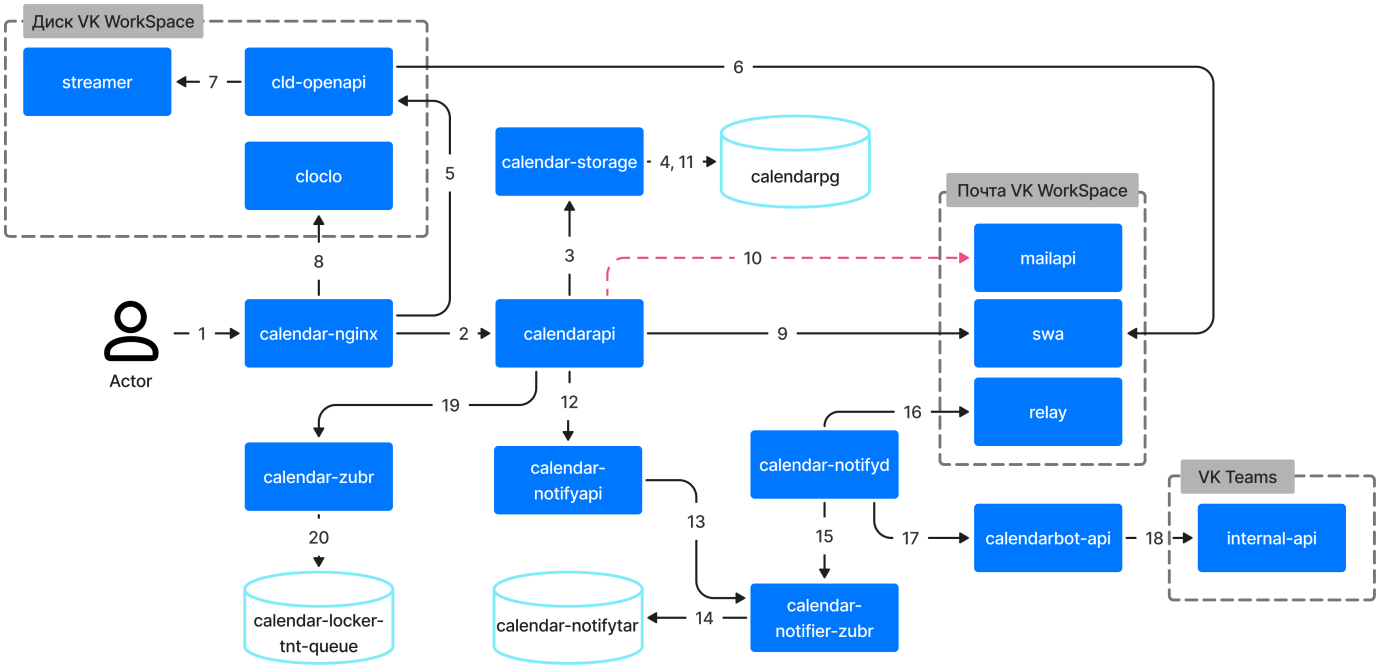
Календарь VK WorkSpace

Портал с документацией: https://biz.mail.ru/docs/on-premises/

Оглавление

Создание, редактирование, удаление события из веб-интерфейса Календаря, принятие или отклонение события

Схема взаимодействия сервисов приведена на рисунке ниже:



Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Протокол взаимодействия	Получатель
1	Actor	HTTP	calendar-nginx — сервис балансировки и проксирования запросов
2	calendar-nginx	HTTP	calendarapi — основной API для межсерверного общения
3	calendarapi	gRPC	calendar-storage — интерфейс взаимодействия с БД. Наружу предоставляет gRPC методы, внутри содержит реализацию запросов к БД и работу с шардированием
4	calendar-storage	Сетевой протокол PostgreSQL	calendarpg — главная база данных Календаря VK WorkSpace

Номер потока	Источник	Протокол взаимодействия	Получатель
5	calendar-nginx	HTTP	cld-openapi — API Диска VK WorkSpace, обслуживающий OAuth клиентов
6	cld-openapi	HTTP	swa — группа сервисов аутентификации (clauth, GoCube, UMI) для выдачи, хранениея и валидации сессионных токенов для пользователей
7	cld-openapi	HTTP	streamer — сервис для сохранения и скачивания файлов хранилища
8	calendar-ngin	HTTP	cloclo — сервис по распределению и сбору информации для получения данных из Диска VK WorkSpace
9	calendarapi	HTTP	swa
10	calendarapi	HTTP	mailapi — основной API Почты VK WorkSpace
11	calendar-storage	Сетевой протокол PostgreSQL	calendarpg
12	calendarapi	HTTP	calendar-notifyapi — API очереди уведомлений Календаря
13	calendar-notifyapi	HTTP	calendar-notifier-zubr — прокси-сервис для доступа к Tarantool. Содержит в себе логику работы с шардированием
14	calendar-notifier-zubr	Бинарный протокол Tarantool	calendar-notifytar — очередь рассылки уведомлений Календаря
15	calendar-notifyd	HTTP	calendar-notifier-zubr
16	calendar-notifyd	HTTP	relay — сервис для пересылки почтовых сообщений

Номер потока	Источник	Протокол взаимодействия	Получатель
17	calendar-notifyd	gRPC	calendarbot-api — API бота календаря Супераппа VK WorkSpace
18	calendarbot-api	HTTP	internal-api — API на стороне Супераппа VK WorkSpace
19	calendarapi	gRPC	calendar-zubr — прокси, изолирующее БД locker-tnt-queue
20	calenar-zubr	gRPC	calenar-locker-tnt-queue

Развернутое описание потоков даных:

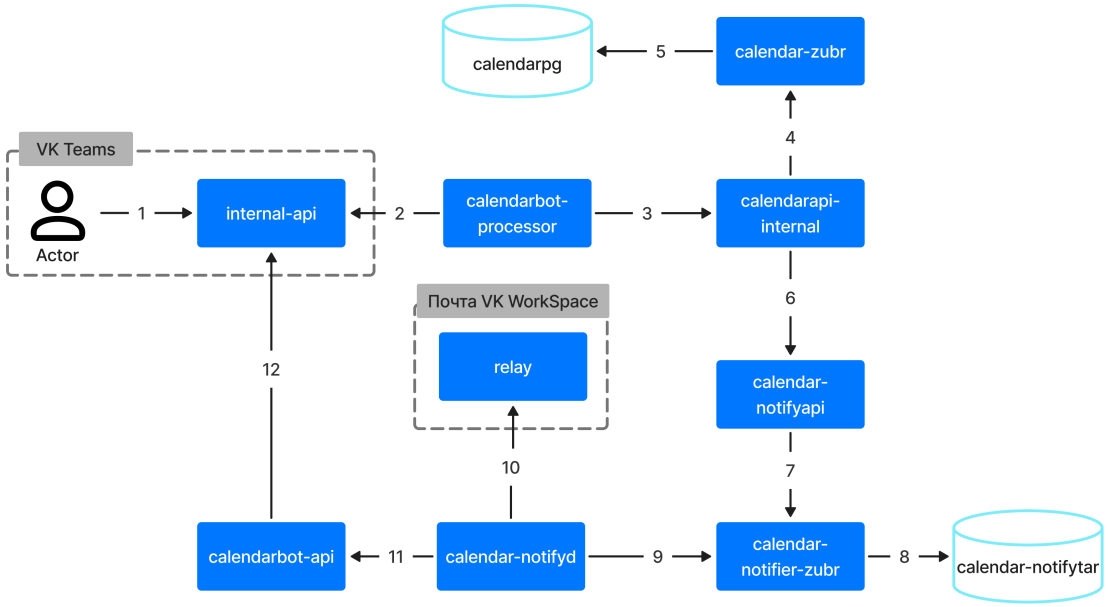
1. При добавлении участника **actor** взаимодействует с сервисом **calendar-nginx** при открытии формы события.
2. Сервис **calendar-nginx** перенаправляет запрос в **calendarapi** для получения свободного времени участника.
3. Далее сервис **calendarapi** перенаправляет запрос в **calendar-storage** для взаимодействия с БД.
4. Сервис **calendar-storage** отправляет запрос в **calendarpg** для получения данных.
5. При наличии во встрече прикрепленных файлов сервис **calendar-nginx** перенаправляет запрос в **cld-openapi**.
6. Далее сервис **cld-openapi** перенаправляет запрос для проверки авторизации в **swa**.
7. После проверки авторизации запрос будет перенаправлен в **streamer** для загрузки файлов. После загрузки в Календарь вернутся ссылки на вложения и прикрепятся в тело встречи. Файл станет доступным для чтения. Для чтения файла запрос от клиента передается сервису **cld-openapi**, в котором генерируется ссылка и возвращается в **calendar-nginx**.
8. Затем полученную ссылку **calendar-nginx** отправляет в сервис **cloclo**, после чего инициируется процесс скачивания (чтения) файла.
9. При сохранении встречи **calendar-nginx** отправляет запрос через сервис **calendarapi** в **swa** для авторизации пользователя.
10. Сервис **calendarapi** может отправить запрос с проверкой валидности почтовых адресов в **mailapi** (подключаемый функционал).
11. Далее выполняется запись созданного события в **calendarpg**.
12. Если в созданной встрече есть другие участники, сервис **calendarapi** передает задачу на рассылку уведомлений с полной информацией о событии и шаблоном уведомления в **calendar-notifyapi**.
13. Далее **calendar-notifyapi** отправляет задачу запрос в прокси-сервис **calendar-notifier-zubr**.
14. Затем **calendar-notifier-zubr** отправляет информацию в **calendar-notifytar**.

15. Далее сервис **calendar-notifyd** забирает через **calendar-notifier-zubr** очереди шаблоны встреч и преобразует их в уведомления для Почты.
16. После преобразования **calendar-notifyd** передает уведомления в сервис Почты VK WorkSpace **relay**.
17. Для Суперaппа VK WorkSpace уведомления отправляются через сервис **calendarbot-api**.
18. Далее **calendarbot-api** передает уведомления в сервис **internal-api** на стороне Суперaппа VK WorkSpace.
19. При редактировании встреч сервис **calendarapi** отправляет запрос в **calenar-zubr**.
20. Затем в **calenar-locker-tnt-queue** выполняется проверка, не редактируется ли встреча в данный момент. Если событие не заблокировано для редактирования, перезаписывается информация в **calendarpg**. По завершению обновления данных в БД пользователям будут отправлены уведомления об изменениях во встрече.

Если событие было удалено, в **calendarpg** отправляется запрос на удаление записи. После удаления записи всем участникам события будут отправлены уведомления об отмене встречи.

Принятие или отклонение приглашения в Суперaппе VK WorkSpace

Схема взаимодействия сервисов приведена на рисунке ниже:



Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Описание потока	Получатель
1	Actor	HTTP	internal-api — API на стороне Суперaппа VK WorkSpace

Номер потока	Источник	Описание потока	Получатель
2	calendarbot-processor	HTTP	internal-api
3	calendarbot-processor	gRPC	calendarapi-internal — API календаря для межсерверного общения
4	calendarapi-internal	iProto	calendar-zubr — прокси БД calendarpg
5	calendar-zubr	Сетевой протокол PostgreSQL	calendarpg — главная база данных Календаря VK WorkSpace
6	calendarapi-internal	HTTP	calendar-notifyapi — API очереди уведомлений Календаря
7	calendar-notifyapi	iProto	calendar-notifier-zubr — прокси-сервис для доступа к Tarantool. Содержит в себе логику работы с шардированием
8	calendar-notifier-zubr	Бинарный протокол Tarantool	calendar-notifytar — очередь рассылки уведомлений Календаря
9	calendar-notifyd	HTTP	calendar-notifier-zubr
10	calendar-notifyd	HTTP	relay — сервис для пересылки почтовых сообщений без авторизации
11	calendar-notifyd	gRPC	calendarbot-api — API бота календаря Суперапп VK WorkSpace
12	calendarbot-api	HTTP	internal-api

Развернутое описание потоков даных:

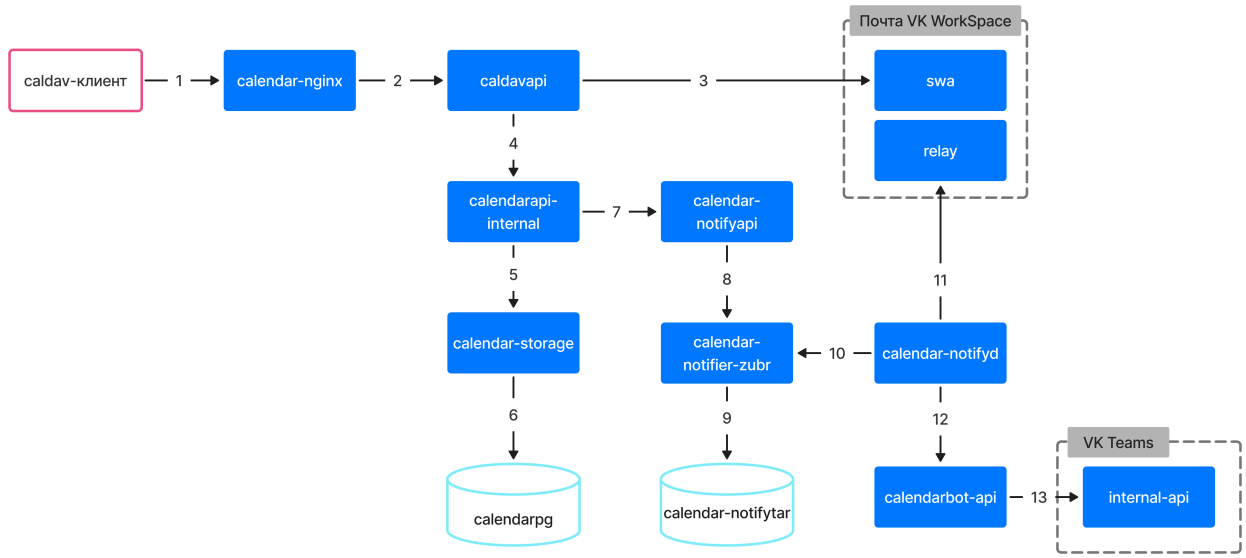
1. Пользователь в веб-интерфейсе нажимает кнопку для принятия или отклонения события, данные передаются в **internal-api**.
2. Сервис **calendarbot-processor** опрашивает сервис **internal-api** на наличие запросов.

3. При наличии нового запроса **calendarbot-processor** передает информацию в **calendarapi-internal** в виде токена, который представляет собой UID шаринга события.
4. Сервис **calendarapi-internal** передает информацию в **calendar-zubr**.
5. Далее **calendar-zubr** передает данные для записи в БД **calendarpg**.

При любом ответе пользователя автору события будут отправлены уведомления о принятом пользователем решении в Почту VK WorkSpace и Суперапп VK WorkSpace (в случае, если уведомления были включены). Дальнейшая цепочка, по которой будет отправлено уведомление автору, соответствует описанию предыдущей схемы.

Создание, редактирование, удаление события при синхронизации по CalDAV

Схема взаимодействия сервисов приведена на рисунке ниже:



Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Описание потока	Получатель
1	caldav-клиент	HTTP	calendar-nginx — сервис балансировки и проксирования запросов
2	calendar-nginx	HTTP	caldavapi — сервис, реализующий сетевой протокол CalDAV. Используется для синхронизации календаря с внешними клиентами
3	caldavapi	HTTP	swa — группа сервисов аутентификации (clauth, GoCube, UMI) для выдачи,

Номер потока	Источник	Описание потока	Получатель
			хранениея и валидации сессионных токенов для пользователей
4	caldavapi	gRPC	calendarapi-internal — API для межсерверного общения
5	calendarapi-internal	gRPC	calendar-storage — интерфейс взаимодействия с БД. Наружу предоставляет gRPC методы, внутри содержит реализацию запросов к БД и работу с шардированием
6	calendar-storage	Сетевой протокол PostgreSQL	calendarpg — главная база данных Календаря VK WorkSpace
7	calendarapi-internal	HTTP	calendar-notifyapi — API очереди уведомлений Календаря
8	calendar-notifyapi	HTTP	calendar-notifier-zubr — прокси-сервис для доступа к Tarantool. Содержит в себе логику работы с шардированием
9	calendar-notifier-zubr	Бинарный протокол Tarantool	calendar-notifytar — очередь рассылки уведомлений Календаря
10	calendar-notifyd	HTTP	calendar-notifier-zubr — прокси-сервис для доступа к Tarantool. Содержит в себе логику работы с шардированием
11	calendar-notifyd	HTTP	relay — сервис для пересылки почтовых сообщений
12	calendar-notifyd	HTTP	calendarbot-api — API бота календаря Супераппа VK WorkSpace
13	calendarbot-api	HTTP	internal-api — API на стороне Супераппа VK WorkSpace

Описание потоков даных при созданнии события:

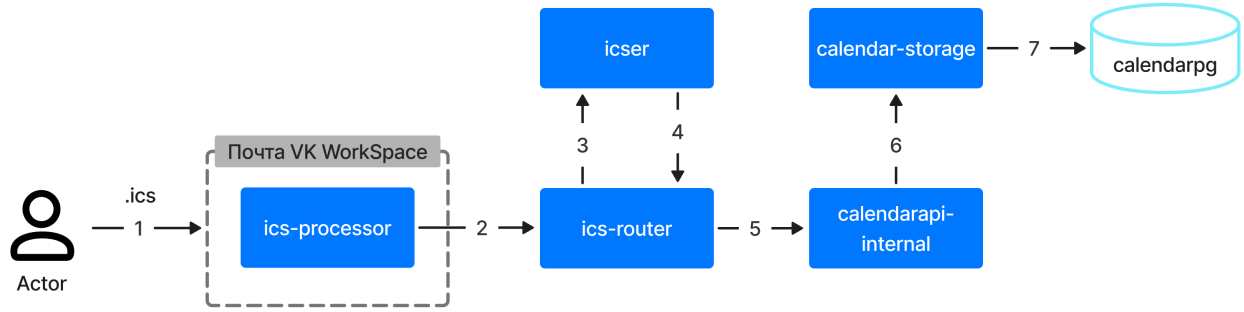
1. При создании события **caldav-клиент** отправляет запрос в **calendar-nginx**.
2. Сервис **calendar-nginx** перенаправляет запрос в **caldavapi**.
3. Далее сервис **caldavapi** перенаправляет запрос в **swa** для авторизации.
4. Затем **caldavapi** передает данные о событии в **calendarapi-internal**.
5. Сервис **calendarapi-internal** передает запрос в сервис **calendar-storage**.
6. Далее **calendar-storage** передает данные для записи в БД **calendarpg**. Затем выполняется цепочка для рассылки уведомлений, потоки с 7 по 13.

При редактировании события выполняется та же цепочка, что и при создании, только с обновлением данных уже существующей записи в **calendarpg** и рассылкой новых уведомлений.

При удалении события выполняется запрос в **calendarpg** для удаления записи и рассылаются уведомления об отмене события.

Доставка приглашения от внешнего отправителя

Схема взаимодействия сервисов приведена на рисунке ниже:



Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Протокол взаимодействия	Получатель
1	Actor	HTTP	ics-processor — обработчик очереди писем с вложениями .ics из внешних календарей
2	ics-processor	HTTP	ics-router — балансировщик нагрузки между сервисами icser
3	ics-router	gRPC	icser — сервис для разбора .ics

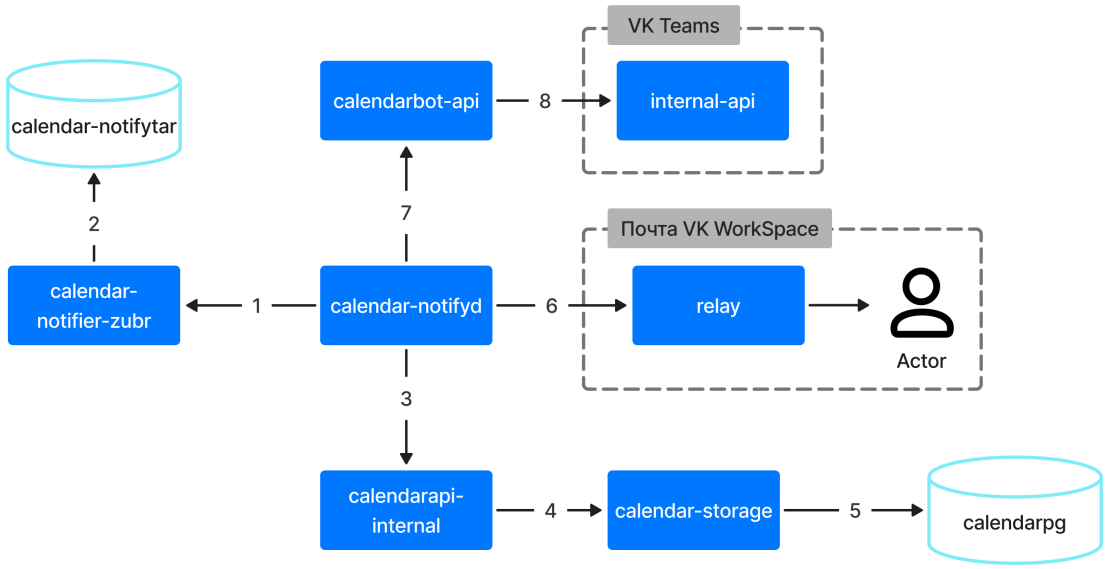
Номер потока	Источник	Протокол взаимодействия	Получатель
4	icser	gRPC	ics-router
5	ics-router	gRPC	calendarapi-internal — API для межсерверного общения
6	calendarapi-internal	gRPC	calendar-storage — интерфейс взаимодействия с БД. Наружу предоставляет gRPC методы, внутри содержит реализацию запросов к БД и работу с шардированием
7	calendar-storage	Сетевой протокол PostgreSQL	calendarpg — главная база данных Календаря VK WorkSpace

Развернутое описание потоков даных:

1. Внешний пользователь отправляет приглашение на почту.
2. Сервис **ics-processor** передает ics-файл в сервис **ics-router**.
3. Далее сервис **ics-router** перенаправляет данные в сервис **icser** для разбора.
4. Затем сервис **icser** выполняет разбор ics-файла и передает вложения и метаинформацию обратно в **ics-router**.
5. Далее **ics-router** перенаправляет запрос в **calendarapi-internal**.
6. Сервис **calendarapi-internal** формирует и отправляет запрос для записи в БД через сервис **calendar-storage**.
7. Далее **calendar-storage** передает полученные данный для записи в **calendarpg**. Событие будет отображено в интерфейсе Календаря.

Отправка напоминаний

Схема взаимодействия сервисов приведена на рисунке ниже:



Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Описание потока	Получатель
1	calendar-notifyd	Бинарный протокол Tarantool	calendar-notifier-zubr — прокси-сервис для доступа к Tarantool. Содержит в себе логику работы с шардированием
2	calendar-notifier-zubr	iProto	calendar-notifytar — очередь рассылки уведомлений Календаря
3	calendar-notifyd	HTTP	calendarapi-internal — API календаря для межсерверного общения
4	calendarapi-internal	gRPC	calendar-storage — прокси БД calendarpg
5	calendar-storage	Сетевой протокол PostgreSQL	calendarpg — главная база данных Календаря
6	calendar-notifyd	HTTP	relay — сервис для пересылки почтовых сообщений
7	calendar-notifyd	gRPC	calendarbot-api — API бота календаря Суперапп VK WorkSpace
8	calendarbot-api	HTTP	internal-api — API на стороне Суперапп VK WorkSpace

Развернутое описание потоков даных:

1. Сервис **calendar-notifyd** с установленной периодичностью отправляет запрос в **calendar-notifier-zubr** для получения шаблонов уведомлений и напоминаний для отправки.
2. Прокси-сервис **calendar-notifier-zubr** перенаправляет запрос в **calendar-notifytar** для получения данных из очереди.

> **Примечание**
>
> В шаблоне уведомлений хранится полная информация о событии, а в шаблоне напоминания — краткая. По данному признкаку **calendar-notifyd** определяет, уведоление это или напоминание.

3. Получив краткую информацию о событии из шаблона напоминания, **calendar-notifyd** отправляет запрос в сервис **calendarapi-internal** на получение полной информации о событии из **calendarpg** через прокси-сервис **calendar-storage**, а также проверяет информацию в шаблоне на актуальность.

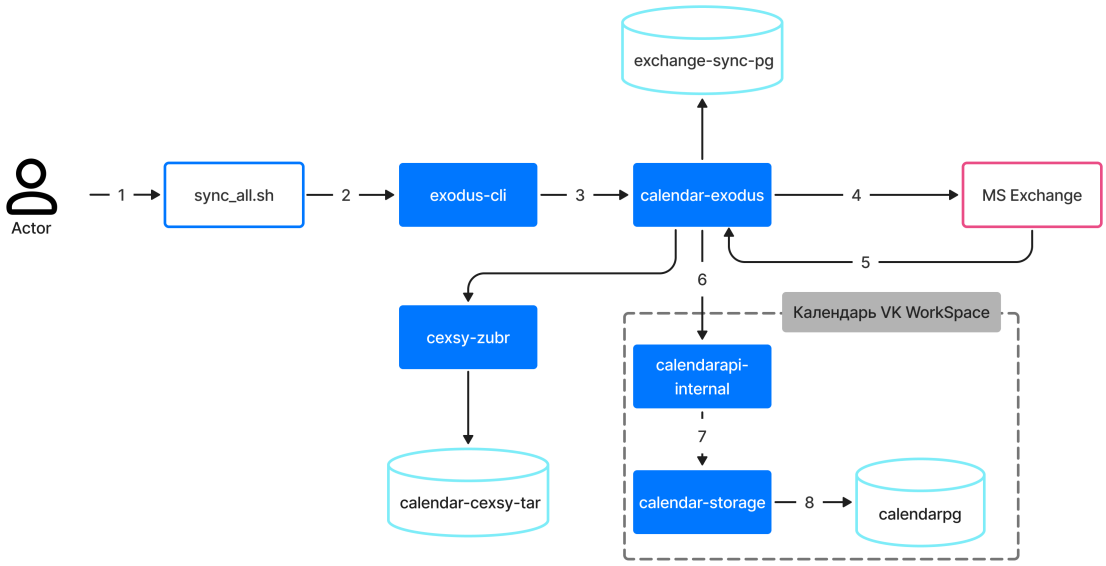
> **Примечание**
>
> Если встреча была перенесена или отменена, напоминания в указанное время отправлены не будут.

4. Далее сервис **calendar-notifyd** передает полученную информацию о событии в **calendarpg** через прокси-сервис **calendar-storage**.
5. Затем **calendar-notifyd** забирает полную информацию о событии, подготоваливает напоминая и передает в сервис **relay** для отправки участникам.
6. Для Супераппа VK WorkSpace напоминания отправляются через сервис **calendarbot-api**.
7. Далее **calendarbot-api** передает напоминания в сервис **internal-api** на стороне Супераппа VK WorkSpace.

Потоки данных при миграции каледарей по протоколу EWS

Синхронизация календарных событий

Схема взаимодействия сервисов приведена на рисунке ниже:



Примечание

Так как в **calendar-cexsy-tar** и **exchange-sync-pg** происходит множество обращений в различные моменты работы миграции, стрелки не пронумерованы.

Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Описание потока	Получатель
1	Actor	Запускает вручную скрипт	sync_all.sh
2	sync_all.sh	Передает почтовые адреса пользователей	exodus-cli — Клиент для работы с exodus
3	exodus-cli	Передает почтовые адреса пользователей	calendar-exodus — сервис для миграции календарей из Exchange и подписки на синхронизацию календарей
4	calendar-exodus	Передает данные на сервер MS Exchange	MS Exchange
5	MS Exchange	Передает события подписанных пользователей в Календарь VK WorkSpace	calendar-exodus — сервис для миграции календарей из Exchange и подписки на синхронизацию календарей
6			

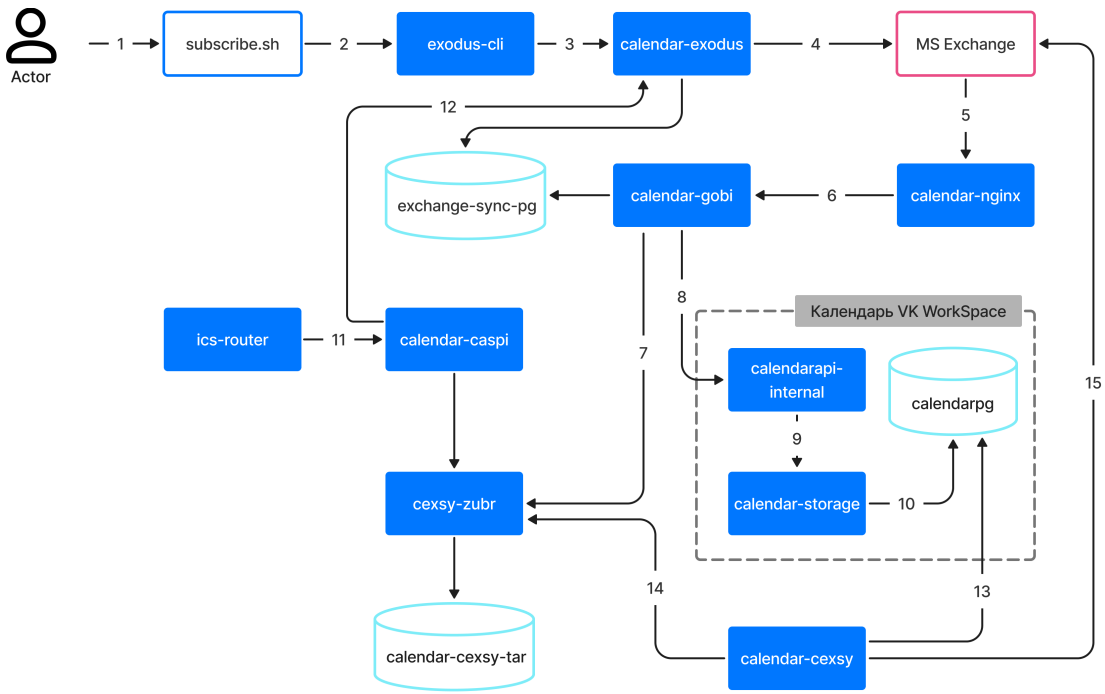
Номер потока	Источник	Описание потока	Получатель
	calendar-exodus	Отправляет данные в API Календаря	calendarapi-internal — API для межсерверного общения
7	calendarapi-internal	Сохраняет в БД	calendar-storage — прокси БД calendarpg
8	calendar-storage	Сохраняет в БД	calendarpg — главная база данных календаря

Описание схемы:

1. Для разового переноса данных календарей с MS Exchange используется скрипт **sync_all.sh**. В скрипте содержится список почтовых ящиков пользователей, календари которых необходимо мигрировать в Почту VK WorkSpace.
2. Данные из скрипта попадают в сервис **calendar-exodus** через **exodus-cli**.
3. После сохранения **calendar-exodus** преобразует информацию из скрипта в запросы к MS Exchange. От сервера **MS Exchange** возвращаются события по пользователям из скрипта обратно в **calendar-exodus**.
4. Все мигрированные события пользователей **calendar-exodus** также отправляет в **calendarapi-internal** для дальнейшей отправки уведомлений. **Calendarapi-internal**, в свою очередь, сохраняет информацию в **calendarpg** через сервис **calendar-storage**.

Подписка на календарные события

Схема взаимодействия сервисов приведена на рисунке ниже:



Примечание

Так как в **calendar-cexsy-tar** и **exchange-sync-pg** происходит множество обращений в различные моменты работы миграции, стрелки не пронумерованы.

Описание потоков данных приведно в таблице ниже:

Номер потока	Источник	Описание потока	Получатель
1	Actor	Запускает вручную скрипт	subscribe.sh
2	subscribe.sh	Передает данные о подписываемых пользователях	exodus-cli — Клиент для работы с exodus
3	exodus-cli	Сохраняет подписываемых пользователей в БД	calendar-exodus — сервис для миграции календарей из Exchange и подписки на синхронизацию календарей
4	calendar-exodus	Передает данные о подписках на сервер EWS	MS Exchange
5	сервер EWS	Передает события подписанных	calendar-nginx — является точкой входа в сервисы Календаря

Номер потока	Источник	Описание потока	Получатель
		пользователей в Календарь VK WorkSpace	
6	calendar-nginx	Проксирует запросы в Календарь	calendar-gobi — сервис для получения уведомлений о событиях и их дальнейшей синхронизации
7	calendar-gobi	Отправляет данные в БД	cexsy-zubr — прокси БД calendar-cexsy-tar
8	calendar-gobi	Отправляет данные в API Календаря	calendarapi-internal — API для межсерверного общения
9	calendarapi-internal	Передает в БД	calendar-storage — прокси БД calendarpg
10	calendar-storage	Сохраняет в БД	calendarpg — главная база данных календаря
11	ics-router	Запрашивает информацию о событии	calendar-caspi — служит публичным API, возвращает информацию о синхронизированных календарях
12	calendar-caspi	Отправляет запрос на выгрузку события	calendar-exodus — сервис для миграции календарей из Exchange и подписки на синк календарей
13	calendar-cexsy	Слушает БД	calendar-storage — прокси БД calendarpg
14	calendar-cexsy	Проверяет в БД наличие событий MS Exchange	cexsy-zubr — прокси БД calendar-cexsy-tar
15	calendar-cexsy	Отправляет данные в EWS	cexsy-zubr — прокси БД calendar-cexsy-tar

Описание схемы:

1. Администратор системы инициирует запуск скрипта **subscribe.sh**, в котором содержатся email-адреса для подписки. Данные из скрипта направляются в сервис **calendar-exodus** через **exodus-cli**.
2. В **calendar-exodus** запросы конвертируются в запросы к серверу MS Exchange. Также **calendar-exodus** проверяет наличие пользователей на стороне MS Exchange.
3. В MS Exchange реализован механизм callback, благодаря которому все события и действия с ними на стороне сервера MS Exchange передаются в сервис **calendar-gobi** через **calendar-nginx**.
4. Сервис **Calendar-gobi** отправляет запрос в **exchange-sync-pg** для получения пользователей, на которых создана подписка. Действия пользователей, отсутствующих в БД, игнорируются.
5. После синхронизации сервис **calendar-gobi** через сервис **cexsy-zubr** делает запись о синхронизации в **calendar-cexsy-tar**.
6. Все полученнные от MS Exchange события сервис **calendar-gobi** отправляет в **calendarapi-internal** для дальнейшей рассылки почтовых уведомлений. **Calendarapi-internal**, в свою очередь, сохраняет события в БД **calendarpg** через **calendar-storage**.

> **Примечание**
>
> **Calendar-cexsy-tar** является центральной базой данных в двусторонней миграции. Она проверяет, было ли событие уже синхронизировано, чтобы миграция событий из VK WorkSpace и обратно не становилась бесконечной. То есть, она отвечает за маппинг событий.

7. **Calendar-caspi** проверяет, есть ли событие на стороне VK WorkSpace. Если события нет, **calendar-caspi** отправляет запрос на выгрузку события из MS Exchange.
8. За отправку событий из Календаря VK WorkSpace в MS Exchange отвечает сервис **calendar-cexsy**. Он опрашивает сервис **calendarpg** на наличие событий, созданных в Календаре.
9. Если событие отсутствует на стороне MS Exchange, событие будет отправлено. В случае, если источником события был MS Exchange и по нему не было изменений, **calendar-cexsy** проигнорирует его.

Автор: Дерябин Дмитрий

16 февраля 2026 г.