Схема базы данных доступна по ссылке.

В случае, если есть расхождения с данной схемой, то информация здесь преобладает над схемой по ссылке.

1) Основные таблицы:

- **User**
- **UserEvent**
- **SubscribeWords**
- **Referral**
- **Invoice**
- **Payment**
- **SubscribePlan**
- **UserSubscribe**
- **Rubric**
- **RubricUserSubscribe**
- **Worker**
- **DigestNews**
- **Channel**
- **ChannelsNews**
- **OpenAiCost**
- **FeedBack** ### 2) Описание таблиц

User

Описание таблицы:
Таблица, хранящая информацию о пользователях системы.

Поля:

- `id` (BigInteger, Identity): Уникальный идентификатор пользователя. Это первичный ключ.
- `user_id` (BigInteger, unique): Уникальный идентификатор пользователя.
- `first_name` (Text): Имя пользователя.
- `username` (Text): Юзер пользователя.
- `chatgpt_flag` (Boolean, default=False): Флаг, указывающий, включен ли чат-бот GPT для пользователя.
- `created_at` (DateTime): Дата и время создания пользователя.
- `updated_at` (DateTime): Дата и время последнего обновления данных пользователя.
- `balance` (Float, default=0.0): Баланс пользователя.(не используется)
- `media_flag` (Boolean, default=True): Флаг, указывающий, включены ли медиа для пользователя.
- `recommendation_flag` (Boolean, default=False): Флаг, указывающий на активность рекомендации для пользователя.
- `channel_up` (Boolean, default=False): Флаг, указывающий, активирован ли канал пользователя.
- `language` (Text, default="ru"): Язык пользователя.(в данный момент функция не используется)
- `digest_time` (Text): Время получения дайджеста пользователем.
- `similar_news_filter` (Boolean, default=True): Флаг фильтрации похожих новостей.
- `utm_source` (Text): Источник UTM для пользователя(допустим media, referral).
- `ban_date` (DateTime): Дата блокировки бота пользователем.

Обязательные поля:

- `user_id`, остальные поля могут быть пустыми. ### UserEvent

Описание таблицы:
Таблица, хранящая информацию о событиях, связанных с пользователями.

Поля:

- `id` (BigInteger): Уникальный идентификатор события.
- `user_id` (ForeignKey): Внешний ключ, ссылающийся на пользователя, к которому относится событие.
- `event_type` (Text): Тип события.
- `event_timestamp` (DateTime, default=func.now() + timedelta(hours=3)): Время события.

Обязательные поля:

- `user_id`, `event_type`. ### SubscribeWords

Описание таблицы:
Таблица, связывающая подписки пользователей с определенными словами для фильтрации.

Поля:

- `id` (BigInteger, Identity): Уникальный идентификатор записи.
- `user_id` (ForeignKey): Внешний ключ, ссылающийся на пользователя.
- `subscribe_id` (ForeignKey): Внешний ключ, ссылающийся на подписку.
- `words` (Text): Слова, связанные с подпиской.

Обязательные поля:

- `user_id`, `subscribe_id`, `words`. ### Referral

Описание таблицы:
Таблица, хранящая информацию о рефералах (приглашенных пользователях).

Поля:

- `id` (BigInteger): Уникальный идентификатор.
- `referrer` (ForeignKey): Внешний ключ, ссылающийся на пользователя, который является реферером.
- `referral` (ForeignKey): Внешний ключ, ссылающийся на пользователя, который был приглашен.

- `type` (Enum(ReferralType)): Тип реферала (новый, активный, оплаченный).
- `created_at` (DateTime): Дата и время создания записи.
- `updated_at` (DateTime): Дата и время последнего обновления записи.

Обязательные поля:

- `referrer`, `referral`, `type`. ### Payment

Описание таблицы:
Таблица, хранящая информацию о платежах пользователей.

Поля:

- `id` (Integer): Уникальный идентификатор платежа.
- `order_id` (String, unique): Идентификатор заказа.
- `amount` (Float): Сумма платежа.
- `status` (Enum(PaymentStatusType)): Статус платежа.
- `created_at` (DateTime): Дата и время создания записи.
- `updated_at` (DateTime): Дата и время последнего обновления записи.
- `user_id` (BigInteger): Идентификатор пользователя.
- `subscribe_type` (Enum(SubscribePlanType)): Тип подписки.
- `days_count` (Integer): Количество дней.
- `is_gift` (Boolean, default=False): Флаг подарочного платежа.
- `for_username` (String): Логин пользователя, для которого был осуществлен платеж.

Обязательные поля:

- `order_id`, `amount`, `status`, `user_id`, `subscribe_type`, `days_count`. ### SubscribePlan

Описание таблицы:
Таблица, хранящая информацию о подписках пользователей.

Поля:

- `id` (Integer): Уникальный идентификатор.
- `payment_id` (Integer, ForeignKey): Внешний ключ, ссылающийся на платеж.
- `created_at` (DateTime): Дата и время создания подписки.
- `updated_at` (DateTime): Дата и время последнего обновления подписки.
- `user_id` (BigInteger, ForeignKey): Идентификатор пользователя, связанного с подпиской.
- `paid_by_user_id` (BigInteger, ForeignKey): Идентификатор пользователя, оплатившего подписку.
- `subscribe_type` (Enum(SubscribePlanType)): Тип подписки.
- `days_count` (Integer): Количество дней подписки.
- `for_username` (String): Логин пользователя.
- `is_gift` (Boolean, default=False): Флаг подарочной подписки.
- `gift_hash` (String, unique): Хеш подарочного кода.
- `is_activated` (Boolean, default=True): Флаг активации подписки.
- `start_at` (DateTime): Дата начала подписки.
- `expired_at` (DateTime): Дата окончания подписки.

Обязательные поля:

- `subscribe_type`, `days_count`.

UserSubscribe

Описание таблицы:
Таблица, хранящая информацию о подписках пользователей на каналы.

Поля:

- `id` (BigInteger, Identity): Уникальный идентификатор записи.
- `channel_id` (BigInteger): Идентификатор канала.
- `user_id` (BigInteger): Идентификатор пользователя, подписавшегося на канал.
- `channel_name` (String): Название канала.
- `channel_link` (String): Ссылка на канал.
- `last_message_is_sended` (Boolean, default=False): Флаг, указывающий, отправлено ли последнее сообщение.
- `worker_id` (Integer, default=2): Идентификатор клиента, связанного с подпиской.
- `created_at` (DateTime): Дата и время создания подписки.
- `foreign_agent` (Boolean, default=False): Флаг, указывающий, является ли агентом внешний агент.
- `category` (String): Категория канала.

Обязательные поля:

- `channel_id`, `user_id`, `channel_name`, `channel_link`. ### Rubric

Описание таблицы:
Таблица, хранящая информацию о рубриках, которые могут быть связаны с подписками пользователей.

Поля:

- `id` (BigInteger): Уникальный идентификатор рубрики.
- `name` (String): Название рубрики.
- `user_id` (BigInteger, ForeignKey): Идентификатор пользователя, который создал рубрику.
- `created_at` (DateTime): Дата и время создания рубрики.

Обязательные поля:

- `name`, `user_id`. ### RubricUserSubscribe

Описание таблицы:
Таблица, связывающая пользователей с рубриками через подписки.

Поля:

- `id` (BigInteger): Уникальный идентификатор записи.
- `user_subscribe_id` (BigInteger, ForeignKey): Идентификатор подписки пользователя.
- `rubric_id` (BigInteger, ForeignKey): Идентификатор рубрики.

Обязательные поля:

- `user_subscribe_id`, `rubric_id`. ### Worker

Описание таблицы:
Таблица, хранящая информацию о воркерах, их сессиях и других данных.

Поля:

- `id` (Integer): Уникальный идентификатор воркера.
- `session` (String): Уникальная сессия воркера.
- `api_id` (BigInteger): Идентификатор API воркера.
- `api_hash` (String): Хэш API воркера.
- `device_model` (String): Модель устройства воркера.
- `system_version` (String): Версия операционной системы.
- `has_proxy` (Boolean, default=False): Флаг, указывающий, используется ли прокси.
- `proxy_type` (String, default="http"): Тип прокси.
- `proxy_address` (String): Адрес прокси.
- `proxy_port` (Integer): Порт прокси.
- `proxy_username` (String): Имя пользователя для прокси.
- `proxy_password` (String): Пароль для прокси.
- `username` (Text): Имя воркера.
- `channels_count` (Integer, default=0): Количество каналов, с которыми работает воркера.
- `has_limit` (Boolean, default=False): Флаг, указывающий, есть ли лимит для воркера.
- `is_active` (Boolean, default=False): Флаг, указывающий, активен ли воркера.
- `created_at` (DateTime): Дата и время создания записи о воркера.
- `updated_at` (DateTime): Дата и время последнего обновления воркера.

Обязательные поля:

- `session`, `api_id`, `api_hash`, `device_model`, `system_version`. ### DigestNews

Описание таблицы:
Таблица, хранящая информацию о новостях дайджеста, отправленных пользователям.

Поля:

- `id` (Integer): Уникальный идентификатор новости.
- `user_id` (BigInteger): Идентификатор пользователя, для которого была отправлена новость.
- `text` (Text): Текст новости.
- `created_at` (DateTime): Дата и время создания новости.
- `sended` (Boolean, default=False): Флаг, указывающий, была ли новость отправлена.
- `theme` (Text): Тема новости.
- `symbol_difference` (BigInteger): Разница в символах.

Обязательные поля:

- `text`, `created_at`. ### Channel

Описание таблицы:
Таблица, хранящая информацию о каналах, которые были напаршены заранее с помощью tg_stat.

Поля:

- `id` (Integer): Уникальный идентификатор канала.
- `title` (Text): Название канала.
- `link` (Text): Ссылка на канал.
- `language` (Text): Язык канала.
- `category` (Text): Категория канала.

Обязательные поля:

- `link`. ### ChannelsNews(не используется) ### OpenAiCost

Описание таблицы:
Таблица, хранящая информацию о стоимости запросов к OpenAI.

Поля:

- `id` (Integer): Уникальный идентификатор записи.
- `user_id` (BigInteger): Идентификатор пользователя.
- `type_of_requests` (String(255)): Тип запросов к OpenAI.
- `input_tokens` (BigInteger): Количество токенов во входном запросе.
- `output_tokens` (BigInteger): Количество токенов в ответе.
- `created_at` (DateTime): Дата и время создания записи.
- `cost` (Float): Стоимость запроса.

Обязательные поля:

- `user_id`, `type_of_requests`, `input_tokens`, `output_tokens`, `cost`. ### FeedBack

Описание таблицы:
Таблица, хранящая информацию о отзывах пользователей.

Поля:

- `id` (Integer): Уникальный идентификатор отзыва.
- `user_id` (BigInteger): Идентификатор пользователя, оставившего отзыв.
- `added_channels` (Integer): Количество добавленных каналов.
- `digest_time` (Text): Время получения дайджеста.
- `text` (Text): Текст отзыва.
- `created_at` (DateTime): Дата и время создания отзыва.

Обязательные поля:

- `user_id`, `text`.

Тут будут идеи

1. Общая архитектура

База данных содержит 16 взаимосвязанных таблиц, реализующих: - Управление пользователями и их профилями - Систему подписок на каналы и рубрики - Платежи и реферальную программу - Работу с контентом и новостными рассылками - Интеграцию с внешними сервисами (Telegram, OpenAI)

2. Основные сущности

2.1 Пользователи (users)

Назначение: Хранение основной информации о пользователях системы
Ключевые поля:

- `chatgpt_flag` - доступ к ChatGPT
- `media_flag` - разрешение медиаконтента
- `balance` - баланс(не используется, в будущем будет удален)
- `digest_time` - предпочтительное время рассылки

Связи:

- С платежами (payments)
- С рефералами (referrals)
- С рубриками (rubrics) ### 2.2 Подписки (users_subscribes)

Назначение: Управление подписками пользователей на Telegram-каналы
Особенности:

- Распределение по воркерам (worker_id)
- Флаг иностранных агентов(является ли канал иностранным агентом) (foreign_agent)
- Система категорий и рубрик

Связи:

- Многие-ко-многим с rubrics
- Связь с workers через worker_id

2.3 Платежи и подписки (payments, subscribe_plans)

Назначение: Управление платными подписками и платежами
Жизненный цикл: 1. Создание инвойса (invoices) 2. Обработка платежа (payments) 3. Активация подписки (subscribe_plans) ### 2.4 Работа с контентом

Каналы (channels): - Каталог доступных для подписки каналов - Классификация по языкам и категориям

Новости (channels_news):

- Архив полученных новостей
- Связь с оригинальными постами через link_of_news

Дайджесты (digest_news): - Система отложенной отправки (sended flag) ## 3. Вспомогательные системы

3.1 Реферальная программа (referrals)

- Трехуровневая система статусов (new/active/paid)
- Защита от самоссылок (referrer ≠ referral) ### 3.2 Воркеры (workers)

Назначение: Управление Telegram-клиентами для сбора новостей
Конфигурация:

- Настройки прокси
- Лимиты каналов на воркер
- Статистика использования (channels_count) ### 3.3 Аналитика и метрики

UserEvent:
Трекинг действий пользователей (открытие приложения, настройки)

OpenAICost:
Учет расходов на генерацию контента с детализацией:

- Типы запросов
- Использование токенов
- Стоимость операций

Feedback:
Система сбора обратной связи с привязкой к:

- Добавленным каналам
- Времени рассылки
- Произвольным комментариям пользователя ## 4. Бизнес-логика ### 4.1 Механика рекомендаций

1. Анализ имеющихся подписок
2. Учет рубрик и категорий
3. Фильтрация через similar_news_filter
4. Формирование персонализированного дайджеста ### 4.2 Модерация контента

- Автоматическая маркировка foreign_agent
- Ручная категоризация каналов
- Фильтрация через subscribe_words ## 5. Особенности реализации

Временные метки:

- Все таблицы содержат created_at/updated_at
- Автоматическое обновление при изменении записи
- Учет московского времени (+3 часа)

Безопасность: - Отдельное хранение платежных данных (order_id) - Система банов через ban_date - Валидация gift_hash (уникальные 8-символьные хеши)

Производительность: - Индексы на внешних ключах - Партиционирование по датам для channels_news

Мониторинг

`monitor_channels` [Каналы] `monitor_new_subscriptions` [[Подписки]] `monitor_session` [[Сессии]]

Сообщения

`process_message` [[Обработка сообщения]] `send_media_group` [[Отправка медиа]] `send_message_to_user_with_media` [[Отправка сообщения]] `get_shortened_text` [[Сокращение текста]]

Каналы

`handle_new_post` [[Обработка нового поста]] `subscribe_to_channels` [[Подписка на канал]] `get_last_event_and_handle` [[Последнее сообщение из канала]]

Описание

Микросервис на основе TelegramClient, позволяет “воркерам” подписываться на каналы, обрабатывать поступающую информацию с них, сокращать тексты, выявлять рекламу, а также убирать похожие новости. Он использует модели машинного обучения и внешние API для обработки данных и улучшения качества взаимодействия с пользователями.

Какие модели используются в воркерах

- Для обработки похожих новостей, нахождения ключевых слов в тексте используется **spaCy** с моделью **ru_core_news_sm**.
- Для сокращения новостей, выделения главной сути и тезисов используется **API GPT-4o-mini**.

Глоссарий

- **Воркеры** — это Telegram клиенты, через которые мы подписываемся на каналы и обрабатываем полученные новости.
- **Телеграм-каналы** — каналы, на которые воркеры подписываются и из которых получают информацию.
- **Новости** — текстовые сообщения или медиафайлы, поступающие от каналов.
- **ChatGPT** — используется для сокращения текстов и выделения основных тезисов.
- **Реклама** — сообщения, содержащие определенные ключевые слова или признаки рекламных материалов.

Основные функции

1. **Подписка на каналы**: Воркеры подписываются на каналы, извлекая информацию из базы данных, и обновляют ID канала при необходимости.
2. **Обработка сообщений**: Включает в себя фильтрацию текста на наличие рекламы, определение тематики, сокращение текста и отправку сообщений пользователям.
3. **Проверка похожих новостей**: Проводится фильтрация поступающих новостей на основе схожести с ранее отправленными пользователями новостями.
4. **Работа с медиа**: Включает загрузку и отправку медиафайлов (фото, видео, документы) подписанным пользователям, с учетом их предпочтений и флагов.

Этот микросервис предоставляет возможность автоматически обрабатывать потоки информации и отправлять релевантные новости пользователям через Telegram.

Общая информация

handle_new_post — асинхронная функция, которая обрабатывает новое сообщение в канале, включая текст и медиафайл. Функция фильтрует сообщения по ключевым словам, сокращает текст, проверяет рекламу, и отправляет сообщения подписанным пользователям в зависимости от их настроек.

Что принимает?

- **w_id** (int): Идентификатор рабочего процесса для логирования.
- **tg_client** (TelegramClient): Объект клиента для взаимодействия с Telegram API.
- **event** (Event): Объект события, представляющий новое сообщение в чате.
- **for_user** (int, optional): Идентификатор пользователя, если нужно обработать сообщение для конкретного пользователя (по умолчанию `None`).

Логика работы

1. **Обработка групповых сообщений**:
   - Если сообщение является частью медиагруппы, проверяется, было ли оно уже обработано.
   - Если сообщение не обработано, оно добавляется в список обработанных.
2. **Получение информации о канале**:
   - Извлекается информация о канале (название и username).
   - Проверяется наличие канала в базе данных.
   - Если канал не найден, происходит выход из канала для рабочих процессов, кроме первого.
3. **Обработка текста и медиа**:
   - Вызывается функция **process_message** для обработки текста и медиа.
   - Если медиа — это фото, проверяется наличие рекламы с помощью функции **advertisment_recognizer**.
   - Если текст или медиа содержат стоп-слова, они исключаются.
4. **Получение подписанных пользователей**:
   - Извлекаются пользователи, подписанные на канал, и разделяются по типам (с ChatGPT или без, с медиа или без).
5. **Сокращение текста с помощью ChatGPT**:
   - Для пользователей с включенной функцией ChatGPT текст сокращается с помощью **get_shortened_text**.
   - Если текст слишком длинный или схож с предыдущими новостями, сообщение не отправляется.
6. **Отправка сообщений подписчикам**:
   - В зависимости от настроек пользователя (с медиа или без), отправляется соответствующее сообщение с текстом и/или медиа.
7. **Очистка файлов**:
   - После обработки медиафайлов вызывается функция **remove_files** для их удаления.

Возвращаемые значения

Функция не возвращает значения, так как выполняет побочные эффекты — обработку сообщений, отправку уведомлений и очистку ресурсов.

Общая информация

subscribe_to_channels — асинхронная функция для подписки на каналы, извлеченные из базы данных. Функция также обновляет информацию о канале, проверяет наличие дубликатов подписок и обрабатывает ошибки, связанные с подпиской на каналы.

Что принимает?

- **telegram_clients** (dict): Словарь клиентов Telegram, использующихся для подписки на каналы.

Логика работы

1. Извлекаются каналы с непрочитанными сообщениями из базы данных.
2. Для каждого канала:
   - Получается объект канала через **get_entity**.
   - Если канал не является трансляцией, он удаляется из базы данных.
   - Выполняется проверка на подписку пользователя на канал через **GetParticipantRequest**.
3. Если канал найден:
   - Проверяется наличие дубликатов подписок для пользователя.
   - При необходимости происходит обновление информации о канале (например, **worker_id**, **channel_id**).
   - Происходит подписка на канал через **JoinChannelRequest**.
4. Обновляется настройка уведомлений канала через **UpdateNotifySettingsRequest**.
5. При ошибке подписки или обновления, канал удаляется из базы данных.
6. Обрабатываются ошибки подписки (например, превышение лимита каналов для рабочего аккаунта, ошибка при получении данных канала).
7. Происходит случайная задержка между подписками, чтобы избежать блокировки.

Общая информация

get_last_event_and_handle — асинхронная функция, которая извлекает последнее сообщение из канала и обрабатывает его. Функция создает фейковое событие (fake event) на основе последнего сообщения и передает его для дальнейшей обработки в **handle_new_post**.

Что принимает?

- **w_id** (int): Идентификатор рабочего процесса для логирования.
- **tg_client** (TelegramClient): Объект клиента для взаимодействия с Telegram API.
- **channel** (Channel): Канал, из которого извлекается последнее сообщение.
- **username** (str): Имя пользователя канала.
- **for_user** (int): Идентификатор пользователя, для которого нужно обработать сообщение.

Логика работы

1. **Проверка канала**:
   - Проверяется наличие атрибутов **id** и **access_hash** у канала. Если один из них отсутствует, выбрасывается ошибка.
2. **Получение последнего сообщения**:
   - Используется метод **GetHistoryRequest** для получения последнего сообщения из канала.
3. **Создание фейкового события**:

    - Если сообщение найдено, создается объект **Event**, который имитирует событие, предоставляя информацию о сообщении, чате и клиенте.
4. **Обработка сообщения**:
    - С помощью функции **handle_new_post** передается фейковое событие для дальнейшей обработки.
5. **Обработка ошибок**:
    - При возникновении исключения логируется ошибка, и выполнение прерывается.

Возвращаемые значения

Функция не возвращает значения, так как выполняет побочные эффекты — извлекает и обрабатывает последнее сообщение из канала.

Общая информация

monitor_channels — асинхронная функция, которая настраивает обработчик событий для прослушивания новых сообщений в Telegram через клиента Telethon. При получении нового сообщения вызывается функция **handle_new_post**.

Что принимает?

- **w_id** (int): Идентификатор рабочей сессии, используемый для логирования и обработки ошибок.
- **tg_client** (TelegramClient): Объект клиента Telethon, который используется для взаимодействия с Telegram API.

Логика работы

1. Настраивает обработчик события для нового сообщения (`NewMessage`) с использованием декоратора **@tg_client.on(events.NewMessage)**.
2. При возникновении нового сообщения вызывает функцию **handle_new_post**, передавая ей **w_id**, **tg_client** и объект события.
3. После настройки обработчика функция продолжает работать, ожидая новых сообщений, пока не будет разорвано соединение с сервером (через вызов **tg_client.run_until_disconnected()**).
4. В случае возникновения исключения логируется ошибка и отправляется уведомление администратору с подробностями о проблеме.

Общая информация

monitor_new_subscriptions - функция, которая с некоторым периодом вызывает функцию **subscribe_to_channels** ## Что принимает? Функция принимает telegram_clients, interval

Логика работы

С определенным интервалом вызывает **subscribe_to_channels**

Общая информация

monitor_session — асинхронная функция, которая отслеживает состояние сессии клиента Telethon в бесконечном цикле. Она проверяет, подключен ли клиент и авторизован ли пользователь, и при необходимости выполняет отключение или перезапуск клиента. Функция также обрабатывает ошибки, такие как тайм-ауты или невалидные номера телефона.

Что принимает?

- **work_id** (int): Идентификатор рабочей сессии для логирования и отслеживания состояния.
- **client** (TelegramClient): Объект клиента Telethon, используемый для взаимодействия с Telegram API.

Логика работы

1. В бесконечном цикле функция выполняет проверку состояния клиента, вызывая методы **client.is_connected()** и **await client.is_user_authorized()**.
2. Если клиент не подключен или не авторизован:
    - Логируется ошибка, и администратору отправляется уведомление.
    - Клиент отключается с помощью **await client.disconnect()** и выбрасывается исключение с сообщением “Worker disconnected”.
3. Если клиент подключен и авторизован, выводится лог с информацией о текущем состоянии.
4. Обрабатываются различные ошибки:
    - **asyncio.TimeoutError** — при тайм-ауте.
    - **PhoneNumberInvalidError** — если номер телефона в сессии некорректен.
    - **asyncio.CancelledError** — если сессия была отменена.
    - В случае других ошибок, логируется сообщение и отправляется уведомление администратору.
5. Если произошла ошибка или клиент был отключен, функция завершает выполнение.
6. После каждой итерации цикл приостанавливается на 60 секунд с помощью **await asyncio.sleep(60)**.

Общая информация

process_message — асинхронная функция, которая обрабатывает новое сообщение в Telegram. Функция анализирует текст сообщения, медиафайлы (фото, документы, опросы), а также группу медиафайлов (если сообщение является частью группы). В зависимости от типа медиафайла или содержимого сообщения возвращает соответствующие данные.

Что принимает?

- **tg_client** (TelegramClient): Объект клиента Telethon для взаимодействия с Telegram API.
- **event** (Event): Объект события, представляющий новое сообщение, которое поступило в чат.

Логика работы

1. **Получение текста сообщения**:
   - Извлекается текст сообщения с помощью **event.message.message**. Если текста нет, он устанавливается как пустая строка.
2. **Обработка медиа-группы** (если сообщение является частью группы):
   - Если у сообщения есть **grouped_id**, функция получает все сообщения чата, связанные с этой группой, и обрабатывает до 10 последних.
   - Если текст сообщения пустой, то текст берется из первого доступного сообщения в группе.
   - Для каждого сообщения в группе, если оно содержит медиа, файл скачивается, если размер документа не превышает 50 MB.
   - Возвращает текст и пути к медиафайлам, а также пометку `"group"`.
3. **Обработка одиночных медиафайлов**:
   - Если сообщение содержит медиа, определяется его тип:
     - Если это **фото** (MessageMediaPhoto), устанавливается тип `"photo"`.
     - Если это **опрос** (MessageMediaPoll), возвращается текст `"Опрос"`.
     - Если это **документ** (MessageMediaDocument):
       - Если размер документа больше 50 MB, возвращается текст и метка `">50"`, чтобы указать, что файл слишком большой.
       - Если документ является стикером или имеет атрибут `round_message`, функция возвращает специальный код для этих случаев.
   - Если медиафайл не слишком большой и не является стикером, скачивается медиафайл и его путь возвращается.
4. **Возврат значений**:
   - Возвращает текст сообщения, путь к медиафайлу (или список путей, если это группа медиа) и тип медиа (или пометку типа “group” для медиа-группы).

Общая информация

send_media_group — функция, которая отправляет группу медиафайлов в Telegram чат. Она принимает список путей к медиафайлам (фото и видео) и текстовое сообщение, которое будет добавлено к первому медиафайлу. Функция обрабатывает различные типы медиафайлов, поддерживает видео с указанием размеров и отправляет их в чат.

Что принимает?

- **chat_id** (int): Идентификатор чата, куда нужно отправить медиафайлы.
- **media_paths** (list): Список путей к медиафайлам (фото и видео), которые будут отправлены.
- **text** (str): Текстовое сообщение, которое будет добавлено к первому медиафайлу.

Логика работы

1. Создается пустой список **input_media_files**, который будет содержать объекты медиафайлов.
2. Для каждого пути в **media_paths**:
   - Если файл является изображением (расширения .jpg, .jpeg, .png), создается объект **InputMediaPhoto**.
   - Если файл является видео (расширения .mp4, .mkv, .mov, .webm), вычисляются его размеры с помощью функции **get_video_dimensions()** и создается объект **InputMediaVideo** с указанием поддерживаемого стриминга.
   - В случае ошибок при обработке медиафайлов, они логируются.
3. Если список **input_media_files** содержит медиафайлы:
   - Текстовое сообщение добавляется к первому медиафайлу и устанавливается форматирование в **HTML**.
   - Группа медиафайлов отправляется в чат с помощью метода **bot.send_media_group**.
   - Логируется успешная отправка.
4. Если не удалось найти валидные медиафайлы, выводится предупреждение о том, что медиафайлы не были найдены для отправки.

Общая информация

send_message_to_user_with_media — асинхронная функция для отправки сообщения с медиафайлом пользователю. Функция позволяет отправлять текстовое сообщение с вложением (фото, документ или видеосообщение), а также форматирует сообщение с учетом условий.

Что принимает?

- **user_id** (int): Идентификатор пользователя, которому отправляется сообщение.
- **channel_up** (bool): Флаг, указывающий, отправляется ли сообщение от канала.
- **message_text** (str): Текст сообщения.
- **media_file_path** (str, optional): Путь к медиафайлу, который будет отправлен.
- **media_type** (str, optional): Тип медиафайла (“photo”, “document”, “round”, “group”).
- **channel_name** (str, optional): Название канала.
- **channel_link** (str, optional): Ссылка на канал.
- **send_media** (bool): Флаг, указывающий, нужно ли отправлять медиа.
- **is_foreign_agent** (bool): Флаг, указывающий, является ли сообщение от иностранного агента.
- **rubrics** (str, optional): Рубрика или категория, добавляемая к сообщению.

Логика работы

1. Проверка на наличие ключевых слов для пользователя. Если слова не найдены, сообщение не отправляется.
2. При необходимости добавляется текст для сообщений от иностранного агента.
3. Форматирование сообщения с учетом позиции канала:
   - Если **channel_up**: Название канала будет первым.
   - Если не **channel_up**: Ссылка на канал будет внизу.
4. Если длина текста больше 1024 символов:
   - Обрезается текст, добавляется ссылка на полный текст.
5. Отправка медиа:
   - Если **send_media** и есть медиафайл:
     - **photo** — отправка фото.
     - **document** — отправка документа (включая видео).
     - **round** — отправка видеосообщения.
     - **group** — отправка группы медиафайлов.
6. В случае ошибок при отправке сообщения (например, если бот был заблокирован), происходит удаление подписки пользователя.

Общая информация

get_shortened_text — асинхронная функция для сокращения текста и определения его тематики с помощью ChatGPT. Также проверяет наличие рекламы.

Что принимает?

- **text** (str): Текст для сокращения и анализа.
- **users_id** (str, optional): Идентификатор пользователя для вычисления токенов.

Логика работы

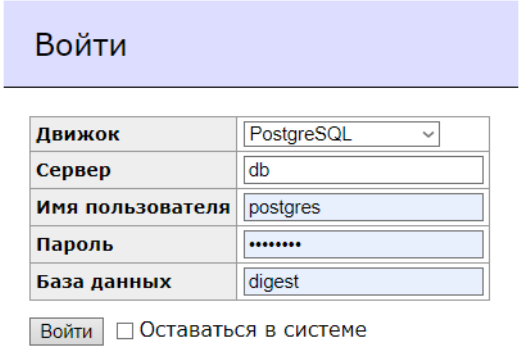
1. Если длина текста ≤ 150 символов, возвращается оригинальный текст и 0.
2. Иначе, вызывается ChatGPT для сокращения текста до 1-2 предложений и определения тематики (реклама или одна из предустановленных тем).
3. Возвращаются:
    - Сокращенный текст.
    - Тематика (например, “реклама”).
    - Длина сокращения.
4. При ошибке возвращается оригинальный текст, пустая тематика и 0.

Все взаимодействия с базой данных, происходят путем использования класса {username}DAO(Base), вот пример такого класса ```class UserDAO(BaseDAO):
model = User

```
@classmethod
def create_user(cls, user_id, first_name, username, utm_source):
    with session_maker() as session:
        query = insert(User).values(
            user_id=user_id,
            first_name=first_name,
            username=username,
            chatgpt_flag=False,
            balance=0.0,
            created_at=func.now() + timedelta(hours=3),
            updated_at=func.now() + timedelta(hours=3),
            utm_source=utm_source,
        )
        session.execute(query)
        session.commit()

        return cls.find_one_or_none(user_id=user_id)
```

```
Открывать сессию вне класса СТРОГО запрещено

[[Требования к бэкэнду]]
[[Как запустить проект в первый раз]]
[[Документация/Сервисы/Админка/Информация для бэкэнда Админки/Взаимодействие с базой данных]]

## Установка проекта
Клонируйте репозиторий и установите его в удобную вам папку. Далее запросите у коллег файл виртуального окружения(.env) для запуска проекта
Для запуска установите docker dekstop
## Запуск проекта
Проект можно запустить локально на компьютере таким образом:
#### 1. Создать файл postgres.yaml(заполнить его можно таким образом)
```

name: digest
services:
db:
container_name: db
image: postgres
restart: always
user: postgres
volumes:
- digest-db-data:/var/lib/postgresql/data
ports:
- “5432:5432”
command: -p 5432
environment:
- POSTGRES_DB=digest
- POSTGRES_PASSWORD=postgres
healthcheck:
test: [“CMD”, “pg_isready”]
interval: 1s
timeout: 1s
retries: 50
adminer:
image: adminer
restart: always
ports:
- “8080:8080”
environment:
- ADMINER_DEFAULT_SERVER=db
- ADMINER_DEFAULT_PORT=5432
volumes:
digest-db-data: ``` Этот файл запускает два контейнера: 1) База данных 2) Админка для регулировки этой базы данных(доступна по ссылке localhost:8080) Ниже пример входа



После того как он создан впишите команду в консоли `docker-compose -f postgres.yaml up -d` #### 2. Установить зависимости Пример для windows(команды вписать в консоли, в директории проекта)

```
python -m venv venv
pip install -r requirements.txt
```

3. Запустить проект

Запустить проект можно вписать в консоли(в корневой папке) команду `python .\main.py` #### 4. Подготовка базы данных

```
from sqlalchemy import create_engine

from models import Base


DATABASE_URL = "postgresql+psycopg2://postgres:postgres@localhost/digest"
engine = create_engine(DATABASE_URL)

Base.metadata.drop_all(engine)
Base.metadata.create_all(engine)

print("Таблицы успешно созданы!")
```

Запустите этот код, далее зайдите в базу данных и заполните таблицу workers, без нее бот не сможет обрабатывать каналы!

Данные требование необходимо соблюдать в будущем!

Код

1. Названия функций и переменных должно отражать их предназначение.
2. Названия всех функций и методов классов, которые должны быть приватными в пакете/классе должны начинаться либо с _, либо с __
3. Каждая функция, класс должны иметь документацию в определенном формате. Данный формат должен быть обсужден
4. Наименования функций, классов должно иметь единый стиль в пакете.
5. Не стоит зависеть от общих классов. Исключением может являться объект, являющийся репрезентацией сущности с которой мы работаем. Для каждого хендлера должен быть собственный класс запроса и ответа, даже если мы имеем одинаковые ответы в разных хендлерах, классы ответов должны быть разными.
6. Все функции и методы должны быть покрыты UNIT-тестами. Желательно иметь покрытие 60-80%.
7. Для каждого хендлера желательно иметь подробное описание всех возможных ответов с примером ответа. ## GIT
8. Все ветки должны иметь в своем названии информацию по типу ветки, краткому описанию и идентификатору задачи в таск менеджеру

- `{prefix}/{name}-{identifier}` Префикс может быть один из следующих:
- `feature`
- `fix`
- `update`

1. Перед выпуском обновления в prod необходимо провести старшему специалисту code review # Особенности кода
2. При взаимодействии со временем необходимо переводить UTC время в МСК. По умолчанию в проекте у нас время по МСК
3. При запуске проекта необходимо добавить воркера в базу данных, в ином случае проект запустится, но не сможет обрабатывать каналы

[[Команды]]

Функционал бота

Настройки

[[Отображение медиа]] [[Похожие публикации]] [[Рекомендации для пользователя]] [[Сокращенный текст новостей]]

Дайджест бот позволяет пользователям собирать персонализированную ленту на основе его подписок, присылать сокращенные новости, рекомендации, добавлять ключевые слова для фильтрации текста и многое другое, что будет расписано в дальнейших разделах документации (так же написать про админку)

Описание

Дайджест ко времени - позволяет пользователю получать сводку новостей

Значение по умолчанию

По умолчанию включено у пользователя ## Как работает? Добавляет медиа, к новости отсылаемой ботом, если оно существует(видео, фото, аудио)

Значение по умолчанию

По умолчанию включены у пользователя ## Как работает? Мы используем модель, которая отбирает новости по базе данных для пользователя, если находит похожую новость(в пределах 12 часов), то он не отправляет ее пользователю

Замечания на будущее

Пока что работает не всегда, требуется переработка функции

Значение по умолчанию

По умолчанию отключены у пользователя ## Как работает? Рекомендации берутся из нашей базы данных, бот пытается определить категорию канала и:

Если определяет категорию, то отправляет похожие каналы В ином случае, отправляет самые популярные каналы из бота(ориентируется на подписки пользователей)

Как мы находим каналы?

Для нахождения каналов используется сервис TG STAT, мы забираем около 80 каналов для каждой категории, чтобы выдавать их пользователю

Значение по умолчанию

По умолчанию отключено у пользователя

Как работает?

Используется GPT, для выделения основной мысли текста, которая поступает из новости.

[[Команды]]

[[Обработка сообщений]]

Каналы

`back_to_start` [[Возврат к кнопке start]]
`channel_guide` [[Инструкция по добавлению каналов]]
`show_channel_info` [[Информация о канале]]
`list_channels` [[Список каналов пользователя]]
`delete_channel` [[Удаление канала]]

Ключевые слова

`add_word` [[Добавление ключевого слова]]
`add_new_word` [[Добавление ключевого слова]]
`list_word` [[Список ключевых слов]]
`delete_word` [[Удаление ключевого слова]]

Общий файл callback

`account` [[Аккаунт]]
`pay_subscribe` [[Оплата подписки]]
`handle_second_mes_pagination` [[Пагинация видео]]
`handle_callback_query` [[Оплата подписки]]

Флаги

`toggle_media` [[Переключение флага медиа]]
`toggle_media2` [[Переключение флага медиа]]
`toggle_channel_up` [[Переключение флага название канала]]
`toggle_similar` [[Переключение флага похожие каналы]]
`toggle_chatgpt` [[Переключение флага gpt]]

Дайджест

`digest` [[Режим дайджеста]]
`digest_off` [[Режим дайджеста]]
`get_now` [[Режим дайджеста]]

Рекомендации

`toggle_recommendation` [Рекомендации]
`handle_subscription` [Рекомендации]

Меню

`back_to_account` [[Вернуться к аккаунту]]
`regulate_channel` [[Регулировка каналов]]
`regulate_settings` [[Регулировка настроек]]

Подписка

`recharge_balance` [[Стандартная подписка]]
`recharge_balance2` [[Подписка с другим планом]]
`recharge_balance3` [[Подписка с ограничением по времени]]
`recharge_balance_with_free_days` [[Подписка с бесплатными днями]]

Рефералы

`referrals_info` [[Информация о рефералах]]
`open_gift` [[Общий подарок]]
`recharge_balance (gift-invoice)` [[Оплата подарка]]
`personal_gift` [[Персональный подарок]]
`referrals_btn` [[Реферальная ссылка]]
`referrals_stat` [[Статистика по рефералам]]
`gift_type` [[Тип подарка]]

Рубрики

`rubric_add_channel` [[Добавление канала в рубрику]]
`add_rubric` [[Добавление рубрики]]
`show_rubric_info` [[Информация о рубрике]]
`rubric_channels_query` [[Каналы рубрики]]
`confirm_delete_rubric` [[Подтверждение удаления рубрики]]
`list_rubrics` [[Список рубрик]]
`handle_page_rubric` [[Список рубрик с пагинацией]]
`rubric_rem_channel` [[Удаление канала с рубрики]]
`delete_rubric` [[Удаление рубрики]]

1. Команда `/start`

Назначение: Первичная инициализация пользователя в системе
Логика работы:

1. Обработка UTM-меток:
   - `giftXXXX` - активация подарочной подписки
   - цифровой ID - реферальная программа
2. Создание нового пользователя с:
   - Пробной 7-дневной подпиской
   - Отложенными уведомлениями (через 4 часа и 3 дня)
   - Генерацией персонального изображения профиля
3. Для существующих пользователей:
   - Снятие бана при наличии
   - Отправка адаптированного приветствия

Особенности: - Автозагрузка видео-инструкции с fallback механизмом - Интеграция с реферальной системой - Поддержка разных сценариев входа (медийные источники/обычный) ## 2. Команда `/account`

Назначение: Управление персональным аккаунтом
Функционал: - Отображение: - Персонализированного изображения профиля - Основного меню управления - Статуса подписки **Обработка ошибок:** - Автоматическая перегенерация изображения при отсутствии - Логирование проблем с доступом к медиафайлам ## 3. Команда `/language`(не используется)

Назначение: Смена языка интерфейса
Реализация:

- Поддерживаемые языки:
  - □□ Русский (`set_language_ru`)
  - □□ English (`set_language_en`)
- Особенность: Требуется доработка механизма обновления интерфейса после смены ## 4. Команда `/search`(не используется)

Назначение: Активация голосового режима (экспериментальный)
Функции: - Перевод бота в состояние `VoiceStates.waiting_for_voice` - Предупреждение о “коварстве ИИ” □ - Используется для: - Голосового поиска каналов - Управления через аудиосообщения ## 5. Команда `/add`

Назначение: Добавление новых каналов
Логика: - Фиксация события `add_command` - Отправка инструкций по формату ## 6. Команда `/pay`

Назначение: Управление платными подписками
Сценарии использования: 1. Для активной подписки: - Показ вариантов продления - Кнопки “Месячный”/“Годовой” план 2. Для неактивной подписки: - Отображение условий оплаты - Выбор тарифного плана ## 7. Команда `/support`

Назначение: Связь с технической поддержкой
Реализация: - Перенаправление в Telegram-чат `@digest_support` - Фиксация события `support_command` ## 8. Команда `/options`

Назначение: Настройка параметров системы
Особенности: - Динамическое меню настроек: - Управление рассылками - Фильтры контента - Настройки ChatGPT - Система генерации изображения профиля: - Автоматическое создание при первом входе - Кэширование в `media/image/{user_id}.jpg` ## 9. Команда `/channels`

Назначение: Просмотр текущих подписок

Логика отображения: Генерация интерактивного списка - Обработка пустого списка: - Отправка инструкции по добавлению - Кнопка возврата в аккаунт ## 10. Команда `/refresh`

Назначение: Сброс аккаунта (только для тестирования)
Функционал:

- Полное удаление пользовательских данных
- Принудительный рестарт через `/start`
- Логирование операций в `TestUserRefresh` ## Системные особенности:

1. **Механизм событий:**
   - Все действия фиксируются в `UserEvent`
   - Примеры событий: `start_text`, `pay_command`, `get_settings`
2. **Работа с медиа:**
   - Видео кэшируется через `upload_video()`
   - Использование `file_id` Telegram для оптимизации
3. **Безопасность:**
   - Валидация пользователя при каждом запросе
   - Защита от SQL-инъекций через ORM
   - Изоляция тестовых данных
4. **Логирование:**
   - Детальный трекинг ошибок
   - Запись ключевых пользовательских действий
   - Использование структурированных логов

1. Обработка голосовых сообщений

Эндпоинт: `@register_message_handler(content_types=["voice"])`
Состояние: Требуется активный стейт (после `/search`) ### Логика работы: 1. Распознавание речи через ASR-сервис 2. Обработка запроса через GPT 3. Маршрутизация ответа: - **Дайджест**: `digest-тема` - **Поиск**: `search-ключевые_слова` - **Произвольный ответ**: прямая отправка **Особенности:** - Автоматическое удаление промежуточных сообщений - Поддержка Markdown в ответах(на данный момент) ## 2. Обработка подарков (username)

Состояние: `GiftUserStates.waiting_for_username`

Активация: После выбора “Персональный подарок” ### Валидация имени:

1. Удаление `@` в начале
2. Проверка на:
   - Спецсимволы
   - Пробелы
   - Совпадение с отправителем **Ошибки:**

- `gift_username_invalid` - неверный формат
- `gift_username_self_error` - попытка самоподарка ## 3. Обработка ключевых слов **Состояние:** `UserWordStates.waiting_for_word`
  Контекст: Добавление фильтров к каналу **Особенности:**
- Регистронезависимое хранение
- Максимум 5 фраз на канал
- Автоматическая тримминг пробелов ## 4. Создание рубрик

Состояние: `AddRubricStates.waiting_for_rubric_name`
Лимиты:

- Пагинация по 10 рубрик на пользователя **Интеграция:**

1. Создание рубрики
2. Привязка к существующим подпискам
3. Генерация клавиатуры управления ## 5. Обработка медиаконтента

Поддерживаемые типы: - Текст - Фото/Видео - Документы - Голосовые/Видеосообщения

Тут будет реализован план рефакторинга

back_to_start(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает кнопку с данными `back_to_start`. Он возвращает пользователя в начальное состояние с возможностью вернуться к инструкциям или выбрать другие действия.

Логика

- Проверяется язык пользователя.
- Если каналы не добавлены, пользователю показывается инструкция по добавлению канала или текст, в зависимости от UTM-метки.
- Если каналы есть, пользователь возвращается к стартовому экрану.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

add_word(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает на кнопку для добавления ключевого слова для канала (например, с данными `add_word|{channel_id}`). Он предоставляет интерфейс для добавления ключевого слова или отображения существующих.

Логика

- Проверяется, есть ли уже добавленные ключевые слова.
- Если нет ключевых слов, бот переходит в режим ожидания, где пользователь может ввести новое ключевое слово.
- Если ключевые слова уже есть, отправляется список доступных слов с возможностью их удалить или изменить.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

add_new_word(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает кнопку для добавления нового ключевого слова (например, с данными `add_new_word|{channel_id}`). Он проверяет, не превышает ли количество ключевых слов лимит (максимум 5) и позволяет добавить новое слово.

Логика

- Проверяется, не добавлено ли уже больше 5 ключевых слов для канала.
- Если лимит превышен, выводится сообщение с предупреждением.
- В противном случае, пользователь переходит в режим ожидания для ввода нового слова.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

channel_guide(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает кнопку с данными `channel_guide`. Он отображает инструкцию по добавлению канала, если каналы не были найдены, или список каналов, если они уже существуют.

Логика

- Если у пользователя нет подписанных каналов:
  - Отправляется инструкция по добавлению канала.
  - Добавляется кнопка для возврата в главное меню.
- Если каналы есть:
  - Отправляется список всех доступных каналов.
  - Добавляется возможность вернуться к списку каналов.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

show_channel_info(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает на канал для просмотра его информации (например, с кнопки типа `channel_{channel_id}`). Он отображает подробности канала с возможностью удалить канал или изменить ключевые слова.

Логика

- Извлекается информация о канале из базы данных.
- Пользователь может удалить канал или просматривать/добавлять ключевые слова для канала.
- Отправляется соответствующее сообщение с кнопками для дальнейших действий.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

Описание

Этот раздел описывает callback-обработчики, отвечающие за управление каналами в боте. Обработчики позволяют пользователям добавлять каналы, просматривать их информацию, а также удалять. # Общая информация

- Все обработчики используют информацию о пользователе для персонализированных сообщений.
- Логирование используется для отслеживания действий пользователя и выявления ошибок.
- Все обработчики взаимодействуют с базой данных для получения и изменения информации о каналах и ключевых словах.
- В будущем будет переработано

list_channels(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает кнопку с данными `list_channels`. Он отправляет список каналов, на которые подписан пользователь.

Логика

- Если у пользователя нет подписанных каналов:
  - Отправляется инструкция по добавлению канала.
  - Добавляется кнопка для возврата в аккаунт.
- Если каналы есть:
  - Отправляется список всех каналов пользователя. ### Параметры:
- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

list_word(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает на ключевое слово для получения подробной информации о нем (например, с данными `word_{word_id}`). Он отображает информацию о выбранном слове и предоставляет возможность его удалить или изменить.

Логика

- Загружается информация о ключевом слове.
- Отправляется сообщение с информацией о слове и возможностью для пользователя удалить или изменить его.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

delete_channel(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает кнопку для удаления канала (например, с данными `delete_channel_{channel_id}`). Он удаляет канал из базы данных и обновляет список доступных каналов.

Логика

- Из базы данных удаляется выбранный канал.
- Отправляется сообщение с подтверждением удаления канала.
- Обновляется список каналов пользователя.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

delete_word(call, bot)

Описание

Этот обработчик вызывается, когда пользователь нажимает кнопку для удаления ключевого слова (например, с данными `delete_word|{word_id}`). Он удаляет ключевое слово из базы данных и отображает соответствующее сообщение.

Логика

- Удаляется выбранное ключевое слово.
- Пользователь получает подтверждение о том, что слово было удалено.

Параметры:

- `call`: объект, содержащий информацию о callback-запросе от пользователя.
- `bot`: объект бота, через который отправляются сообщения.

back_to_account

Назначение:

Обработчик для callback'а, который отвечает за возвращение пользователя в главное меню аккаунта. Этот callback вызывается, когда пользователь нажимает кнопку для возврата в основное меню.

Действия:

1. Логирует начало выполнения.
2. Извлекает информацию о пользователе из базы данных.
3. Проверяет наличие фото пользователя. Если фото отсутствует, пытается загрузить его заново.
4. Отправляет обновленное сообщение с изображением профиля и основной клавиатурой меню.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `UserEventDAO`: Для логирования события пользователя (возвращение в аккаунт).
- `generate_main_menu_keyboard`: Генерация клавиатуры для главного меню.
- `update_image`: Функция для обновления изображения пользователя, если оно отсутствует.
- `types.InputMediaPhoto`: Отправка изображения профиля пользователя.

Описание

Этот набор callback-обработчиков управляет действиями пользователей в меню, связанными с настройками и каналами. В основном, они включают в себя навигацию по различным разделам меню и работу с изображениями пользователей.

Закомментированный Callback (В будущем возможно будет реализован)

Назначение:

Этот callback, в будущем, будет использоваться для обработки часто задаваемых вопросов (FAQ) и поддержки пользователей. Кнопка будет выводить текст с часто задаваемыми вопросами и поддерживающей клавиатурой.

Действия:

1. Получение информации о пользователе.
2. Отправка текста с часто задаваемыми вопросами.
3. Добавление клавиатуры с опциями для дальнейших действий пользователя.

Используемые компоненты:

- `get_support_menu_keyboard`: Генерация клавиатуры с поддержкой.
- `get_user_info`: Получение текста с информацией для пользователя.

regulate_channel

Назначение:

Обработчик для callback'а, который отвечает за управление каналами пользователя. Этот callback вызывается, когда пользователь нажимает на кнопку для настройки или выбора каналов.

Действия:

1. Логирует начало выполнения.
2. Извлекает информацию о пользователе из базы данных.
3. Проверяет наличие фото пользователя. Если фото отсутствует, пытается загрузить его заново.
4. Отправляет обновленное сообщение с изображением профиля и клавиатурой для управления каналами.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `generate_channels_menu_keyboard`: Генерация клавиатуры для меню каналов.
- `update_image`: Функция для обновления изображения пользователя, если оно отсутствует.
- `types.InputMediaPhoto`: Отправка изображения профиля пользователя.

regulate_settings

Назначение:

Обработчик для callback'а, который отвечает за отображение меню настроек пользователя. Этот callback вызывается, когда пользователь нажимает на кнопку для управления настройками.

Действия:

1. Логирует начало выполнения.
2. Извлекает информацию о пользователе из базы данных.
3. Проверяет наличие фото пользователя. Если фото отсутствует, пытается загрузить его заново.
4. Отправляет обновленное сообщение с изображением профиля и клавиатурой для настроек.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `UserEventDAO`: Для логирования события пользователя (получение настроек).
- `generate_settings_menu_keyboard`: Генерация клавиатуры для меню настроек.
- `update_image`: Функция для обновления изображения пользователя, если оно отсутствует.
- `types.InputMediaPhoto`: Отправка изображения профиля пользователя.

account

- **Назначение**: Отображение аккаунта пользователя с обновленным профилем.
- **Действия**:
  - Загружает и отображает основное меню с персонализированным изображением профиля.
  - Логирует запрос и отправляет ответ пользователю.

- **Используемые компоненты**:
  - `generate_main_menu_keyboard` для формирования меню.
  - `update_image` для обновления изображения. ## start_btn
- **Назначение**: Обработка нажатия кнопки старт.
- **Действия**:
  - Отправляет пользователю сообщение приветствия, выполняя команду `/start`.
- **Используемые компоненты**:
  - `send_welcome` для отправки приветственного сообщения.

Назначение

Обработчики команд в этом модуле реализуют действия при нажатии пользователем кнопок в интерфейсе бота, управляют состоянием пользователя и обновляют контент в чатах. ### Структура

Каждый обработчик взаимодействует с определенной кнопкой и выполняет действия в зависимости от состояния пользователя или команды. Обработчики используют `telebot` для работы с Telegram API и базы данных SQLAlchemy для хранения состояния пользователей.

Общие особенности

1. **Логирование**: Все обработчики логируют действия пользователей для дальнейшего анализа и улучшения функционала.
2. **Подписка**: Для большинства функций, таких как фильтрация новостей, рекомендации или использование ChatGPT, проверяется наличие активной премиум-подписки.
3. **Переводы**: Все сообщения и уведомления генерируются с учетом языка пользователя, что позволяет поддерживать локализацию для различных языков.
4. **Интерактивность**: Все команды сопровождаются интерактивными кнопками для управления состоянием и перехода между различными интерфейсами. ## set_language(не используется)

- **Назначение**: Установка языка для пользователя.
- **Действия**:
  - Изменяет язык пользователя в базе данных и вызывает отправку приветственного сообщения.
- **Используемые компоненты**:
  - `send_welcome` для отправки приветственного сообщения. # В будущем будет переработано

Назначение

Обработчики команд в этом модуле реализуют действия при нажатии пользователем кнопок в интерфейсе бота, управляют состоянием пользователя и обновляют контент в чатах. ### Структура

Каждый обработчик взаимодействует с определенной кнопкой и выполняет действия в зависимости от состояния пользователя или команды. Обработчики используют `telebot` для работы с Telegram API и базы данных SQLAlchemy для хранения состояния пользователей.

pay_subscribe

- **Назначение**: Платеж за подписку.
- **Действия**:
  - Проверяет наличие подписки и отображает опции для подписки (ежемесячная или ежегодная).
  - Логирует информацию о подписке.
  - Отправляет текст с деталями подписки и кнопками выбора.
- **Используемые компоненты**:
  - `InlineKeyboardButton` для создания кнопок подписки.
  - `translations` для локализации текста.(не используется)

handle_second_mes_pagination

- **Назначение**: Пагинация видео-сообщений в процессе подписки.
- **Действия**:
  - Загружает видео и текст, отображая их в зависимости от выбранной страницы.
  - Логирует действия пользователя.
- **Используемые компоненты**:
  - `upload_video` для загрузки видео.

handle_callback_query (channels_page_)

- **Назначение**: Обработка пагинации страниц канала.
- **Действия**:
  - Переход к следующей или предыдущей странице списка каналов.
  - Отправка обновленного списка каналов пользователю.
- **Используемые компоненты**:
  - `generate_channel_list` для формирования списка каналов.

toggle_chatgpt

- **Назначение**: Переключение флага использования ChatGPT.
- **Действия**:
  - Проверяет подписку и состояние дайджеста, затем переключает флаг `chatgpt_flag`.
  - Обновляет медиафайл профиля пользователя.
  - Логирует событие и обновляет интерфейс с изображением.
- **Используемые компоненты**:
  - `update_image` для обновления изображения профиля.
  - `generate_settings_menu_keyboard` для обновления интерфейса.

1. toggle_media(1 версия)

- **Назначение**: Переключение флага показа медиа-материалов.
- **Действия**:
  - Проверяет подписку пользователя и обновляет флаг `media_flag`.
  - Логирует событие и обновляет меню пользователя.
- **Используемые компоненты**: -`generate_settings_menu_keyboard` для обновления интерфейса.
  - `UserDAO.check_subscribe` для проверки подписки. ## 2. toggle_media2(2 версия)
- **Назначение**: Переключение флага отображения медиа-контента для пользователя.
- **Действия**:
  - Переключает значение флага `media_flag` пользователя.
  - Обновляет кнопку с текстом в зависимости от состояния флага.
  - Логирует событие и отправляет обновленную кнопку в чат.
- **Используемые компоненты**:
  - `UserEventDAO.create_user_event` для записи события.
  - `translations` для перевода текста.
  - `bot.edit_message_reply_markup` для обновления кнопки.

Определяет, где будет название канала, сверху или снизу ### toggle_channel_up

- **Назначение**: Переключение флага канала для пользователя.
- **Действия**:
  - Переключает флаг `channel_up` пользователя.
  - Логирует изменения и обновляет меню канала.
- **Используемые компоненты**:
  - `generate_channels_menu_keyboard` для обновления интерфейса.

toggle_similar

- **Назначение**: Переключение фильтра похожих новостей.
- **Действия**:
  - Проверяет наличие премиум-подписки у пользователя.
  - Переключает флаг фильтра похожих новостей.
  - Логирует изменение и обновляет интерфейс пользователя.
- **Используемые компоненты**:
  - `UserDAO.check_subscribe` для проверки подписки.
  - `generate_settings_menu_keyboard` для обновления интерфейса.
  - `translations` для сообщений об ошибке.

digest

- **Назначение**: Включение режима дайджеста.
- **Действия**:
  - Активирует режим дайджеста и отправляет информацию с обновленным меню.
- **Используемые компоненты**:
  - `generate_digest_keyboard` для формирования клавиатуры.

digest_off

- **Назначение**: Отключение режима дайджеста.
- **Действия**:
  - Отключает режим дайджеста и сбрасывает `digest_time`.
  - Отправляет обновленное сообщение.
- **Используемые компоненты**:
  - `digest_off_scheduler` для сброса таймера дайджеста.
  - `generate_digest_keyboard` для обновления интерфейса.

get_now

- **Назначение**: Получение дайджеста в текущий момент.
- **Действия**:
  - Отправляет дайджест пользователю.
- **Используемые компоненты**:
  - `send_digest` для отправки дайджеста.

toggle_recommendation

- **Назначение**: Переключение флага рекомендаций для пользователя.
- **Действия**:
  - Проверяет подписку пользователя и переключает флаг `recommendation_flag`.
  - Логирует событие и обновляет меню настроек.
- **Используемые компоненты**:
  - `generate_settings_menu_keyboard` для обновления интерфейса. ## handle_subscription
- **Назначение**: Обработка обновлений рекомендаций для пользователя.
- **Действия**:
  - Обновляет рекомендации для указанной категории (например, “наши каналы” или другая категория если найдена).
- **Используемые компоненты**:
  - `generate_our_recommendation_keyboard` и `generate_recommendation_keyboard` для обновления интерфейса.

Описание

Этот набор callback-обработчиков управляет действиями пользователей в подписке, связанной с покупкой/продлением тарифа. В основном, они включают в себя генерацию платежей

recharge_balance_with_free_days

Назначение:

Обработчик для callback'а, который управляет пополнением баланса с дополнительными бесплатными днями, предоставляемыми пользователю в рамках акции.

Действия:

1. Логирует начало выполнения пополнения баланса с бесплатными днями.
2. Проверяет, прошел ли необходимый срок для использования акции.
3. Извлекает информацию о пользователе из базы данных.
4. Генерирует уникальный платежный ID.
5. Определяет тип подписки и рассчитывает количество дней, включая бесплатные.
6. Добавляет запись о платеже в базу данных.
7. Создает ссылку на оплату и формирует сообщение с деталями для пользователя.
8. Отправляет сообщение пользователю с ссылкой на оплату.

Используемые компоненты:

- `logger`: Для логирования процесса пополнения.
- `UserDAO`: Для получения данных о пользователе.
- `UserEventDAO`: Для проверки, может ли пользователь воспользоваться акцией.
- `PaymentDAO`: Для добавления записи о платеже.
- `create_invoice`: Для генерации ссылки на оплату.
- `translations`: Для получения перевода сообщения для пользователя.
- `settings`: Для получения цены и характеристик плана с учетом бесплатных дней.

recharge_balance2

Назначение:

Обработчик для callback'а, который выполняет пополнение баланса по аналогии с первым обработчиком, но для другого типа подписки с возможностью использования другого плана с дополнительными параметрами.

Действия:

1. Логирует начало выполнения операции пополнения для второго типа плана.
2. Извлекает информацию о пользователе.
3. Генерирует уникальный платежный ID.
4. Определяет тип подписки и количество дней в зависимости от выбранного плана.
5. Добавляет информацию о платеже в базу данных.
6. Создает ссылку на оплату и формирует сообщение для пользователя.
7. Отправляет пользователю сообщение с ссылкой для оплаты.

Используемые компоненты:

- `logger`: Для логирования действий и информации о процессе пополнения.
- `UserDAO`: Для получения данных о пользователе.
- `PaymentDAO`: Для сохранения данных о платеже.
- `create_invoice`: Для генерации платежной ссылки.
- `translations`: Для получения перевода сообщения на язык пользователя.
- `settings`: Для определения цены и характеристик плана.

3. recharge_balance3

Назначение:

Обработчик для callback'а, который управляет пополнением баланса для подписки, при этом учитывает дополнительные условия акции (например, ограничение по времени для использования скидки).

Действия:

1. Логирует начало выполнения операции пополнения для третьего типа плана.
2. Проверяет, прошел ли необходимый срок для использования акции.
3. Извлекает информацию о пользователе из базы данных.
4. Генерирует уникальный ID для платежа.
5. Рассчитывает стоимость и количество дней в зависимости от выбранного плана.
6. Добавляет запись о платеже в базу данных.
7. Создает ссылку на оплату и формирует сообщение с подробностями для пользователя.
8. Отправляет сообщение пользователю с ссылкой на оплату.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения информации о пользователе.
- `UserEventDAO`: Для проверки доступности акции для пользователя.
- `PaymentDAO`: Для сохранения информации о платеже.
- `create_invoice`: Для генерации ссылки на оплату.
- `translations`: Для создания сообщения на языке пользователя.
- `settings`: Для получения цены подписки.

recharge_balance

Назначение:

Обработчик для callback'а, который отвечает за пополнение баланса пользователя с выбором между месячным или годовым планом. Он создает уникальный платежный ID, добавляет информацию о платеже в базу данных, генерирует ссылку для оплаты и отправляет сообщение пользователю с деталями оплаты.

Действия:

1. Логирует начало выполнения операции пополнения.
2. Извлекает информацию о пользователе из базы данных.
3. Генерирует уникальный платежный ID.
4. Определяет тип подписки (месячная или годовая) и рассчитывает количество дней для подписки.
5. Добавляет запись о платеже в базу данных.
6. Создает ссылку для оплаты и формирует сообщение для пользователя с подробностями платежа.
7. Отправляет пользователю сообщение с ссылкой на оплату и суммой.

Используемые компоненты:

- `logger`: Для логирования действий пользователя и информации о процессе оплаты.
- `UserDAO`: Для получения данных о пользователе (язык и ID).
- `PaymentDAO`: Для добавления записи о платеже в базу данных.
- `create_invoice`: Функция для генерации ссылки на оплату.
- `translations`: Для выбора текста сообщения в зависимости от языка пользователя.(не используется сейчас)
- `settings`: Для получения информации о ценах планов.

referrals_info

Назначение:

Обработчик для callback'а, который предоставляет информацию о рефералах пользователя, включая количество приглашенных друзей и ссылку для реферала.

Действия:

1. Логирует начало обработки запроса на информацию о рефералах.
2. Извлекает данные пользователя из базы данных.
3. Генерирует ссылку для реферала на основе ID пользователя.
4. Рассчитывает количество приглашенных рефералов.
5. Формирует сообщение с количеством рефералов и ссылкой для приглашений.
6. Отправляет обновленное сообщение с реферальной ссылкой и количеством рефералов, используя клавиатуру с управлением рефералами.

Используемые компоненты:

- `logger`: Для логирования событий и действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `ReferralDAO`: Для получения информации о рефералах пользователя.
- `translations`: Для формирования сообщения с учетом языка пользователя(не используется).
- `settings`: Для генерации реферальной ссылки.
- `get_referral_keyboard`: Для создания клавиатуры с возможностью управления рефералами.

Описание

Этот набор обработчиков callback'ов управляет действиями пользователей, связанными с реферальной программой и подарками

open_gift

Назначение:

Обработчик для callback'а, который предоставляет информацию о возможных планах и ценах, если пользователь решит взять "публичный" подарок.

Действия:

1. Логирует запрос на открытие подарка.
2. Извлекает данные пользователя.
3. Отправляет информацию о стоимости планов и их вариантах для подарка.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `translations`: Для формирования сообщения с учетом языка пользователя.(не используется)
- `get_gift_plans_kb`: Для создания клавиатуры с планами для подарков.

recharge_balance (gift-invoice)

Назначение:

Обработчик для callback'а, который обрабатывает пополнение баланса для подарков и создает платежные ссылки для пользователей, которые хотят подарить подписку другому пользователю.

Действия:

1. Логирует запрос на пополнение баланса для подарка.
2. Извлекает данные о выбранном плане и пользователе, которому будет подарена подписка.
3. Генерирует уникальный платежный ID.
4. Добавляет информацию о платеже в базу данных.
5. Создает ссылку на оплату для подарка и отправляет информацию пользователю.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `ReferralDAO`: Для получения информации о рефералах (если это необходимо).
- `PaymentDAO`: Для добавления записи о платеже в базу данных.
- `create_invoice`: Для генерации платежной ссылки.
- `translations`: Для формирования сообщения с учетом языка пользователя.(не используется)
- `settings`: Для получения информации о ценах планов.

personal_gift

Назначение:

Обработчик для callback'а, который запрашивает у пользователя персональные данные (например, имя) для оформления персонального подарка.

Действия:

1. Логирует запрос на персональный подарок.
2. Извлекает данные пользователя.
3. Запрашивает у пользователя информацию (например, имя) для подарка.
4. Изменяет состояние пользователя и отправляет сообщение с инструкциями для ввода.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `translations`: Для формирования сообщения с учетом языка пользователя.(не используется)
- `bot.set_state`: Для установки состояния пользователя для дальнейшего ввода.
- `get_return_to_gift_type_kb`: Для создания клавиатуры для возврата к выбору типа

referrals_btn

Назначение:

Обработчик для callback'а, который отправляет пользователю информацию о реферальной программе и предоставляет возможность использовать реферальные кнопки.

Действия:

1. Логирует событие клика по кнопке рефералов.
2. Извлекает данные пользователя из базы данных.
3. Генерирует реферальную ссылку для пользователя.
4. Отправляет сообщение с текстом о реферальной программе и кнопками для использования рефералов.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `translations`: Для формирования текста сообщения с учетом языка пользователя.(не используется)
- `settings`: Для получения информации о реферальной ссылке.
- `get_btn_referral_keyboard`: Для создания клавиатуры с кнопками для использования рефералов.

referrals_stat

Назначение:

Обработчик для callback'а, который предоставляет статистику о рефералах пользователя, включая количество рефералов, количество подписанных и платящих пользователей.

Действия:

1. Логирует запрос на статистику по рефералам.
2. Извлекает данные пользователя и информацию о рефералах.
3. Рассчитывает статистику по общему количеству рефералов, подписавшихся пользователей и платящих рефералов.
4. Формирует сообщение с статистикой и отправляет его пользователю.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `ReferralDAO`: Для получения статистики по рефералам.
- `translations`: Для формирования сообщения с учетом языка пользователя.(не используется)

gift_type

Назначение:

Обработчик для callback'а, который запрашивает у пользователя выбор типа подарка, предоставляя клавиатуру для дальнейших действий.

Действия:

1. Логирует начало запроса на выбор типа подарка.
2. Извлекает данные пользователя.
3. Отправляет пользователю сообщение с предложением выбрать тип подарка и отображает соответствующую клавиатуру.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `UserDAO`: Для получения данных о пользователе.
- `translations`: Для формирования сообщения с учетом языка пользователя.(не используется)
- `get_gift_type_keyboard`: Для создания клавиатуры с выбором типа подарка.

rubric_add_channel

Назначение:

Обработчик для callback'а, который добавляет канал в рубрику и обновляет список каналов для отображения.

Действия:

1. Логирует добавление канала в рубрику.
2. Добавляет канал в рубрику.
3. Обновляет сообщение с новым списком каналов рубрики.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `get_session`: Для получения сессии с базой данных.
- `RubricUserSubscribe`: Для добавления канала в рубрику.
- `rubric_generate_channel_list`: Для генерации обновленного списка каналов.

add_rubric

Назначение:

Обработчик для callback'а, который инициирует процесс создания новой рубрики.

Действия:

1. Логирует запрос на создание новой рубрики.
2. Отправляет сообщение с инструкциями по созданию рубрики.
3. Устанавливает состояние пользователя для ввода имени рубрики.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `translations`: Для локализации сообщения.(не используется)
- `bot.set_state`: Для установки состояния пользователя.

show_rubric_info

Назначение:

Обработчик для callback'а, который отображает информацию о рубрике, включая возможность редактировать каналы, удалить рубрику и вернуться к списку рубрик.

Действия:

1. Логирует запрос на информацию о рубрике.
2. Извлекает данные о рубрике и пользователе.
3. Отправляет сообщение с информацией о рубрике и кнопками для редактирования или удаления рубрики.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `get_session`: Для получения сессии с базой данных.
- `Rubric`: Для получения информации о рубрике.
- `User`: Для получения данных о пользователе.
- `translations`: Для локализации сообщения.(не используется)
- `types.InlineKeyboardButton`: Для создания кнопок с действиями по рубрике.

rubric_channels_query

Назначение:

Обработчик для callback'а, который отображает страницу каналов рубрики, генерирует список каналов и позволяет перейти на другие страницы с каналами.

Действия:

1. Логирует запрос страницы каналов.
2. Извлекает информацию о рубрике и пользователя.
3. Генерирует список каналов рубрики для текущей страницы.
4. Отправляет сообщение с каналами рубрики, используя клавиатуру для навигации между страницами.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `get_session`: Для получения сессии с базой данных.
- `Rubric`: Для получения информации о рубрике.
- `User`: Для получения данных о пользователе.
- `rubric_generate_channel_list`: Для генерации списка каналов рубрики.
- `translations`: Для локализации сообщения.(не используется)

Описание

Этот набор обработчиков callback'ов управляет действиями пользователей, связанными с рубриками и их каналами. Пользователи могут создавать, редактировать, удалять рубрики, а также добавлять и удалять каналы внутри рубрик. Эти обработчики обеспечивают навигацию по страницам рубрик, отображение информации о рубриках, добавление каналов в рубрики и их удаление. Также поддерживаются функции для подтверждения удаления рубрик и управления списками рубрик.

confirm_delete_rubric

Назначение:

Обработчик для callback'а, который подтверждает удаление рубрики.

Действия:

1. Логирует запрос на подтверждение удаления рубрики.
2. Удаляет рубрику из базы данных.
3. Отправляет сообщение с обновленным списком рубрик.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `get_session`: Для получения сессии с базой данных.
- `Rubric`: Для удаления рубрики.
- `translations`: Для локализации сообщения.(не используется)

handle_page_rubric

Назначение:

Обработчик для callback'а, который отображает страницу рубрик с возможностью перехода между страницами.

Действия:

1. Логирует запрос страницы рубрик.
2. Генерирует и отправляет список рубрик для указанной страницы.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `generate_rubric_list`: Для генерации списка рубрик.

list_rubrics

Назначение:

Обработчик для callback'а, который выводит список рубрик, если они существуют, и предлагает добавить новую рубрику, если их нет.

Действия:

1. Логирует запрос на вывод списка рубрик.
2. Извлекает данные о пользователе.
3. Отправляет сообщение с доступными рубриками или инструкциями по добавлению рубрики.

Используемые компоненты:

- `logger`: Для логирования действий пользователя.
- `get_session`: Для получения сессии с базой данных.
- `generate_rubric_list`: Для генерации списка рубрик.
- `translations`: Для локализации сообщения.(не используется)

rubric_rem_channel

Назначение:

Обработчик для callback'а, который удаляет канал из рубрики и обновляет список каналов для отображения.

Действия:

1. Логирует удаление канала из рубрики.
2. Удаляет канал из рубрики.
3. Обновляет сообщение с новым списком каналов рубрики.

Используемые компоненты:

- **`logger`**: Для логирования действий пользователя.
- **`get_session`**: Для получения сессии с базой данных.
- **`RubricUserSubscribe`**: Для удаления канала из рубрики.
- **`rubric_generate_channel_list`**: Для генерации обновленного списка каналов.

delete_rubric

Назначение:

Обработчик для callback'а, который инициирует процесс удаления рубрики.

Действия:

1. Логирует запрос на удаление рубрики.
2. Отправляет сообщение с запросом подтверждения удаления рубрики.

Используемые компоненты:

- **`logger`**: Для логирования действий пользователя.
- **`get_session`**: Для получения сессии с базой данных.
- **`Rubric`**: Для получения информации о рубрике.
- **`translations`**: Для локализации сообщения.(не используется)

Для управления данными пользователей, подписками, каналами, запросами к OpenAI и аналитикой реализована панель администратора на базе Django. Панель обеспечивает удобный доступ к данным бота и предоставляет гибкие возможности для фильтрации, сортировки, аналитики и управления. ## Основные возможности

1. Статистика (home):

URL: /
views.py::home()

Главная страница (`home`) отображает ключевые метрики:

- Общее количество пользователей (`User.objects.count()`).
- Пользователи с пробной подпиской (вычисляются через аннотацию Exists для оплаченных счетов).
- Активные подписчики (определяются как пользователи с `ban_date=None`).
- Графики:
  - `stats.get_paid_chart()` — анализ дохода.
  - `stats.get_active_chart()` — активные пользователи.
  - `stats.get_pro_chart()` — пользователи с подпиской.(планы MONTHLY, YEARLY, GIFT, REFERRAL_PAID)
  - `stats.get_new_exit_chart` — **График ушедших / пришедших пользователей:** отображает динамику регистрации и блокировок бота

Реализовано через подзапросы ORM (Subquery, OuterRef, Exists) и сторонние функции из stats.py.

2. Управление пользователями (UserListView):

URL: /users/ Класс представления `UserListView` предоставляет:

- **Поиск**: По `username, first_name, user_id` с использованием `Q()` и фильтра icontains.
- **Фильтры**:
  - `search` — поиск по имени, ID, названию или ссылке.
  - `plan` — фильтрация по типу подписки (`trial, paid, month, year, expired`(ушедшие платные)).
  - `ordering` — сортировка данных.
  - `utm` — фильтрация по источнику регистрации.
  - `per_page` — количество записей на странице.
- **Сортировка**: По -`created_at` (новые пользователи сверху) или параметру GET-запроса ordering.
- **Пагинация**: Настройка через параметр `per_page`.
- **Аннотации**:
  - Количество подписок на каналы через `Subquery`.
  - Тип последней подписки через `Subquery`.

3. Управление каналами (ChannelListView):

URL: /channels/ - Класс `ChannelListView`: - Отображает список каналов с: - Количеством подписчиков (аннотация `Count('channel_link')`). - Названием и ссылкой на канал. - Привязкой к воркерам. - Поддерживает поиск по названию или ссылке канала (`icontains`).

URL: /channles// - Класс `ChannelDetailView`: - Показывает пользователей, подписанных на конкретный канал и краткую информацию о канале

4. Детальная информация о пользователе (UserInfoView):

URL: /users// - Функция представления `user_info`(): - Выводит данные пользователя через `get_object_or_404`. - Отображает: - Список каналов пользователя с количеством подписчиков каждого канала. - Покупки пользователя. - Логика поиска каналов: - Аннотации с `Count и Subquery`.

5. Управление воркерами (WorkerListView):

URL: /workers/ - Класс `WorkerListView`: - Отображает список воркеров и их состояние. - Стандартная пагинация и фильтрация.

6. Авторизация:

URL: /login/ - Реализована через стандартные классы Django: - `Login` (наследуется от `LoginView`) — для авторизации. - `logout_view`() — для выхода из системы. - Для защиты доступа используется декоратор `@login_required` и `LoginRequiredMixin`.**

7. График затрат пользователей на запросы к OpenAI:

URL: /openaicost/ - Класс `OpenAiCostListView`: - Страница отображает общую сумму затрат пользователей на запросы к OpenAI. - **Фильтры по датам:** - `date_start` и `date_end` — фильтры по диапазону дат (например, для запросов к OpenAI). - **Сортировка по наибольшей сумме запросов:** - Список пользователей отсортирован по сумме запросов. - Реализовано через запросы ORM с агрегацией (Sum) и фильтрацией по дате.**

8. История покупок пользователя (InvoiceListView):

URL: /invoices/ - Для каждого пользователя отображается история его покупок: - Тип подписки (недельная, месячная, годовая). - Дата создания подписки. - Статус (`paid` или `pending`). - Фильтр по статусу (status):
- По умолчанию, Оплаченные подписки
- `all`, Все подписки, даже не оплаченные - Реализовано через связанные запросы ORM (`ForeignKey` на модель `Invoice`).

Методы представлений

- `home()` — отображение статистики.
- `UserListView.get_queryset()` — обработка фильтров, аннотаций и поиска.
- `UserInfoView.get_object()` — получение объекта пользователя.
- `ChannelListView.get_queryset()` — группировка каналов и подсчёт подписчиков.
- `get_paid_subscribers_by_date()` — получение подписчиков с оплаченной подпиской по указанной дате. ****

Структура URL

- `/` — главная страница статистики (`home`).
- `/login/` — авторизация (`Login`).
- `/logout/` — выход из системы (`logout_view`).
- `/users/` — список пользователей (`UserListView`).
- `/users/<user_id>/` — детальная информация о пользователе (`UserInfoView`).
- `/channels/` — список каналов (`ChannelListView`).
- `/channels/<channel_id>/` — пользователи канала (`ChannelDetailView`).
- `/workers/` — список воркеров (`WorkerListView`).
- `/invoices/` — список покупок (`InvoiceListView`).
- `/openaicots/` — топ пользователей по тратам токенов (`OpenAiCostListView`).

Все взаимодействия с базой данных, происходят путем использования встроенной ORM Django, пример модели:

```
class User(models.Model):
    id = models.BigAutoField(primary_key=True)
    user_id = models.BigIntegerField(unique=True)
    first_name = models.TextField(null=True, blank=True)
    username = models.TextField(null=True, blank=True)
    # newsletter_flag = models.BooleanField(default=True)
    chatgpt_flag = models.BooleanField(default=False)
    created_at = models.DateTimeField()
    updated_at = models.DateTimeField()
    balance = models.FloatField(default=0.0)
    media_flag = models.BooleanField(default=False)
    channel_up = models.BooleanField(default=False)
    # expired_at = models.DateTimeField(null=True, blank=True)
    language = models.TextField(default='ru')
    digest_time = models.TextField(null=True, blank=True)
    similar_news_filter = models.BooleanField(default=False)
    recommendation_flag = models.BooleanField(default=False)
    utm_source = models.CharField(null=True, blank=True)
    ban_date = models.DateTimeField()

    class Meta:
        db_table = '"public"."users"'
        managed = False
```

Для новых моделей обязательно добавляем класс Meta! (чтобы новая таблица не создавалась, т.к. создаём на другом уровне)

[[Требования к бэкэнду]] [[Как запустить проект в первый раз]] [[Документация/Сервисы/Админка/Информация для бэкэнда Админки/Взаимодействие с базой данных]]

Установка проекта

Клонируйте репозиторий и установите его в удобную вам папку. Далее запросите у коллег файл виртуального окружения(.env) для запуска проекта Для запуска установите docker dekstop ## Запуск проекта Проект можно запустить локально на компьютере таким образом: #### 1. Запустить файл compose.yml

В терминале docker desktop прописываем команду `docker-compose up --build`

```
services:
  web_admin:
    env_file:
      - .env
    ports:
      - "8080:8000"
    build:
      context: .
      dockerfile: Dockerfile
```

```
    command: "python3 digest_web/manage.py runserver 0.0.0.0:8000"
```

Переходим в терминал запущенного контейнера

1. В терминале переходим в директорию digest_web и применяем миграции

```
cd digest_web python manage.py migrate
```

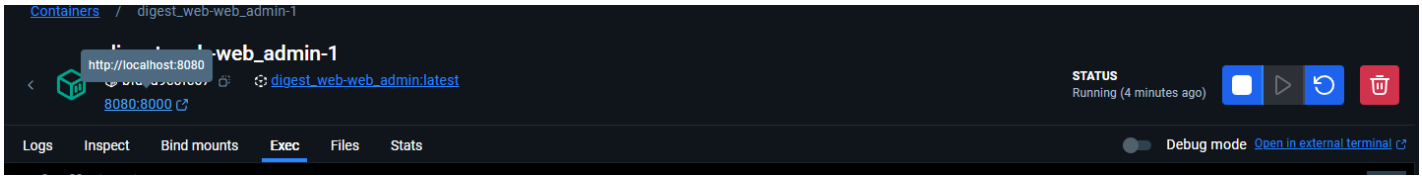
2. Создаём суперпользователя (через него осуществляется вход в админку)

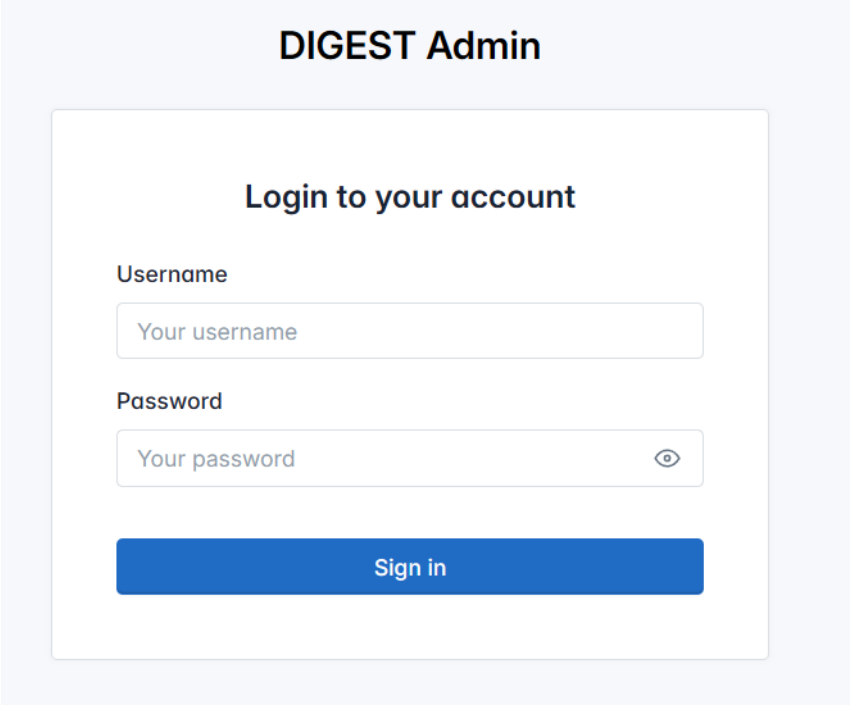
```
python manage.py createsuperuser
```

```
python: can't open file '/app/manage.py': [Errno 2] No such file or directory
# ls
Dockerfile  compose.yml  digest_web  requirements.txt
# cd digest_web
# ls
digest_web  manage.py  static  webadmin
# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions, webadmin
Running migrations:
  No migrations to apply.
# python manage.py createsuperuser
Username (leave blank to use 'root'):
Email address:
Password:
Password (again):
This password is too short. It must contain at least 8 characters.
This password is too common.
This password is entirely numeric.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

2. Переход в панель администратора

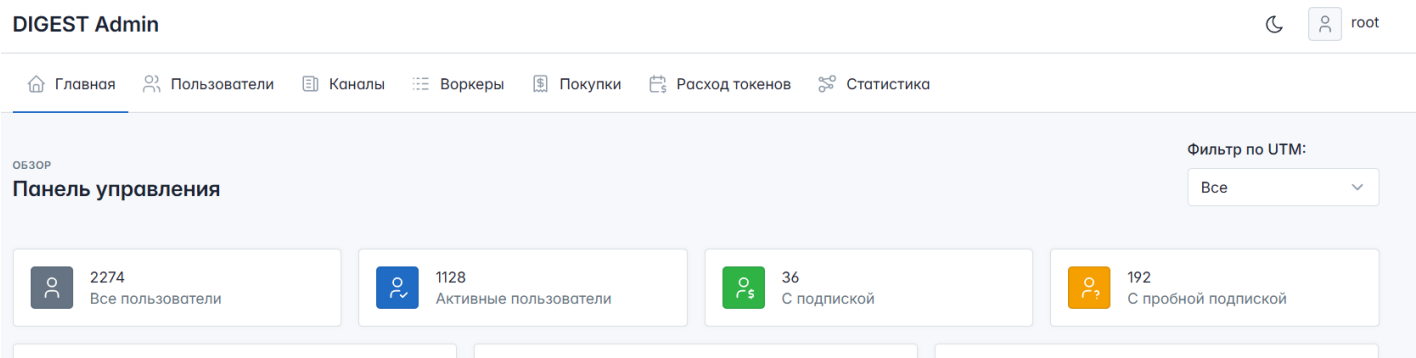
1. Переходим по адресу http://localhost:8080/login/





2. Вводим данные созданного суперпользователя и логинимся

Открывается главная страница админки:



Данные требование необходимо соблюдать в будущем!

Код

1. Названия функций и переменных должно отражать их предназначение.
2. Названия всех функций и методов классов, которые должны быть приватными в пакете/классе должны начинаться либо с _, либо с __
3. Каждая функция, класс должны иметь документацию в определенном формате. Данный формат должен быть обсужден
4. Наименования функций, классов должно иметь единый стиль в пакете.
5. Не стоит зависеть от общих классов. Исключением может являться объект, являющийся репрезентацией сущности с которой мы работаем.
6. Все функции и методы должны быть покрыты UNIT-тестами. Желательно иметь покрытие 60-80%.
7. Для каждого представления желательно иметь подробное описание взаимодействия с бд и шаблоном. ## GIT
8. Все ветки должны иметь в своем названии информацию по типу ветки, краткому описанию и идентификатору задачи в таск менеджеру

- `{prefix}/{name}-{identifier}` Префикс может быть один из следующих:
- `feature`
- `fix`
- `update`

1. Перед выпуском обновления в prod необходимо провести старшему специалисту code review # Особенности кода
2. При взаимодействии со временем необходимо переводить UTC время в МСК. По умолчанию в проекте у нас время по МСК
3. При запуске проекта необходимо добавить воркера в базу данных, в ином случае проект запустится, но не сможет обрабатывать каналы

Панель администратора (для авторизации запросите у коллег данные админ-пользователя) представляет собой аналитический инструмент с возможностью автоматизировать действия с пользователями. - Позволяет просматривать информацию по пользователям в Дайджест бот, их события в боте, всю информацию по их подпискам, и фильтровать под определённые критерии, которые помогут найти закономерности в событиях. - Также автоматизация действий (выдать подарочную подписку, сконструировать сообщение для рассылки)

Страница Воркеры

DIGEST Admin

admin

Главная | Пользователи | Каналы | Воркеры | Покупки | Расход токенов | Статистика

Воркеры

Показать 100 записей

	WORKER ID	СЕССИЯ	ТЕЛЕГРАМ	КОЛИЧЕСТВО ПОДПИСОК	ЛИМИТ	АКТИВЕН	ЗАГРУЖЕННОСТЬ
1	1	accounts/session_name_test	-	347	✓	✓	0
2	2	accounts/acc2-79789174653	-	262	✗	✓	0
3	3	accounts/acc3-79789718920	-	246	✗	✓	0
4	4	accounts/acc1-79789015846	-	259	✗	✓	0
5	5	accounts/acc5-79998737270	@zubovnic	157	✗	✓	0
6	6	accounts/acc6-79010025589	-	154	✗	✓	0
7	7	accounts/acc7-79522153847	-	149	✗	✓	0
8	8	accounts/acc8-79779170097	@maxereve	147	✗	✓	0
9	9	accounts/acc9-79910036162	@varvarshaa	83	✗	✓	0
11	11	accounts/acc11-79772614994	-	74	✗	✓	0
12	12	accounts/acc12-79015278244	-	13	✗	✓	0
13	13	accounts/acc13-79919525670	-	13	✗	✓	0

Обзор

Страница “Воркеры” предоставляет информацию о технических объектах (воркерах), которые обеспечивают сбор и обработку данных из Telegram-каналов. Воркеры — это технические аккаунты, которые непосредственно взаимодействуют с Telegram API для получения контента из каналов, на которые подписаны пользователи. Эта страница позволяет оценить текущее состояние технической инфраструктуры сервиса и понять возможные ограничения масштабирования.

Основные возможности

Список воркеров

DIGEST Admin

admin

Главная | Пользователи | Каналы | Воркеры | Покупки | Расход токенов | Статистика

Воркеры

Показать 100 записей

	WORKER ID	СЕССИЯ	ТЕЛЕГРАМ	КОЛИЧЕСТВО ПОДПИСОК	ЛИМИТ	АКТИВЕН	ЗАГРУЖЕННОСТЬ
1	1	accounts/session_name_test	-	347	✓	✓	0
2	2	accounts/acc2-79789174653	-	262	✗	✓	0
3	3	accounts/acc3-79789718920	-	246	✗	✓	0
4	4	accounts/acc1-79789015846	-	259	✗	✓	0
5	5	accounts/acc5-79998737270	@zubovnic	157	✗	✓	0
6	6	accounts/acc6-79010025589	-	154	✗	✓	0
7	7	accounts/acc7-79522153847	-	149	✗	✓	0
8	8	accounts/acc8-79779170097	@maxereve	147	✗	✓	0
9	9	accounts/acc9-79910036162	@varvarshaa	83	✗	✓	0
11	11	accounts/acc11-79772614994	-	74	✗	✓	0
12	12	accounts/acc12-79015278244	-	13	✗	✓	0
13	13	accounts/acc13-79919525670	-	13	✗	✓	0

На странице отображается таблица со следующими данными о каждом воркере: - ID воркера - Информация о сессии (технический идентификатор) - Телеграм аккаунт (если привязан) - Количество обслуживаемых подписок - Наличие лимита - Статус активности - Уровень загруженности

Цветовая индикация

Система использует цветовое кодирование для быстрой оценки состояния воркеров: - **Зеленый** - воркер активен или имеет оптимальное количество подписок (0) - **Красный** - воркер достиг лимита подписок или имеет другие ограничения - **Оранжевый** - воркер приближается к критической нагрузке (более 400 подписок)

Настройка отображения

Возможность выбора количества отображаемых воркеров на странице и удобная навигация по страницам.

Как использовать для маркетинговых задач

Оценка масштабируемости сервиса

1. **Анализ технических возможностей**:
   - Оценивайте общее количество воркеров и их загруженность для понимания инфраструктурных возможностей
   - Отслеживайте количество воркеров, достигших лимита, для прогнозирования возможных ограничений в будущем
2. **Прогнозирование расширения**:
   - Оценивайте соотношение количества активных воркеров к общему числу пользователей
   - Анализируйте возможность быстрого масштабирования при успешных маркетинговых кампаниях
   - Планируйте темпы роста пользовательской базы с учетом технических ограничений

Координация с техническим отделом

1. **Совместное планирование маркетинговых активностей**:
   - Перед запуском крупных рекламных кампаний проверяйте доступность технических ресурсов
   - Согласовывайте с техническим отделом график активностей, требующих значительного расширения аудитории
   - Устанавливайте реалистичные KPI с учетом технических возможностей системы
2. **Приоритизация технических улучшений**:
   - Используйте данные о загруженности воркеров для обсуждения необходимых улучшений инфраструктуры
   - Аргументируйте необходимость расширения технических возможностей для достижения маркетинговых целей
   - Совместно планируйте развитие продукта с учетом технических ограничений

Мониторинг качества сервиса

1. **Контроль стабильности работы**:
   - Отслеживайте количество активных воркеров для понимания общей работоспособности системы
   - При обнаружении снижения числа активных воркеров или повышения их загруженности корректируйте ожидания пользователей
   - Используйте эту информацию для проактивной коммуникации с пользователями при возможных задержках в работе сервиса
2. **Анализ пользовательского опыта**:
   - Отслеживайте корреляцию между загруженностью воркеров и показателями удовлетворенности пользователей
   - Выявляйте пороговые значения загруженности, при которых начинается ухудшение пользовательского опыта
   - Планируйте улучшения до достижения критических порогов

Практические кейсы использования

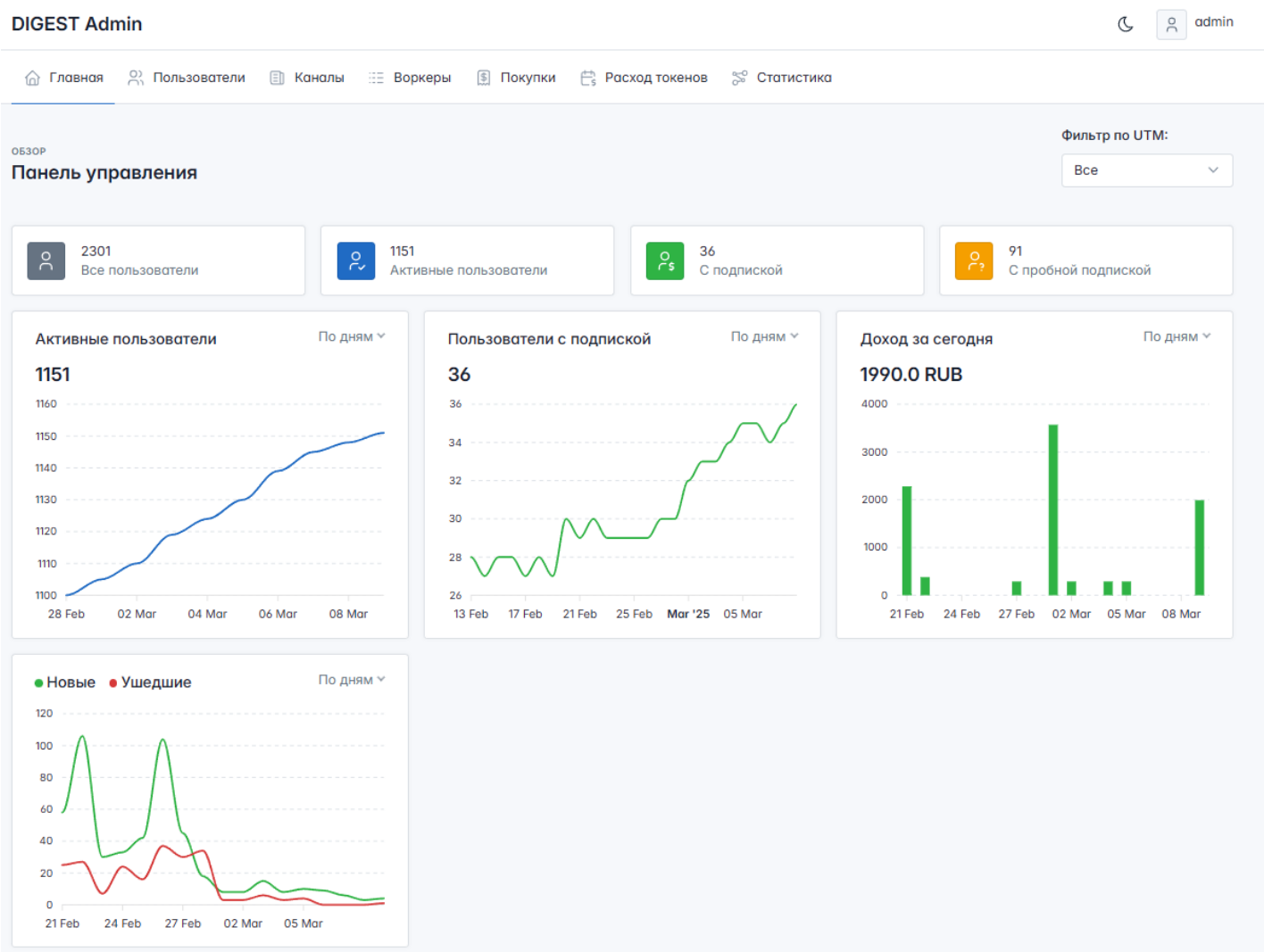
1. **Планирование рекламных кампаний**:
   - Перед запуском масштабной кампании проверьте текущую загруженность воркеров
   - Рассчитайте максимальное количество новых пользователей, которое система может обслужить без ухудшения качества
   - Планируйте поэтапный запуск кампании с мониторингом технических показателей
2. **Управление ожиданиями клиентов**:
   - При высокой загрузке системы корректируйте обещания по скорости обработки новостей
   - Подготовьте коммуникационную стратегию для периодов пиковой нагрузки
   - Используйте технические данные для обоснования временных ограничений некоторых функций
3. **Разработка ценовых стратегий**:
   - Учитывайте техническую стоимость обслуживания пользователя при разработке тарифных планов
   - Рассматривайте возможность введения премиальных тарифов с выделенными ресурсами для VIP-клиентов
   - Анализируйте эффективность различных тарифных моделей с учетом технических ограничений
4. **Оптимизация пользовательских потоков**:
   - Разрабатывайте стратегии перераспределения нагрузки между воркерами
   - Предлагайте пользователям альтернативные способы потребления контента в периоды пиковой нагрузки
   - Создавайте механизмы приоритизации для платящих пользователей при ограниченных ресурсах

Советы по эффективному использованию

- Регулярно мониторьте соотношение числа воркеров к количеству пользователей и подписок
- Устанавливайте пороговые значения нагрузки как триггеры для маркетинговых решений
- Координируйте маркетинговые активности с планами технического развития инфраструктуры
- Используйте информацию о технических ограничениях как аргумент при обосновании бюджетов на техническое развитие
- Разрабатывайте сценарии масштабирования с учетом текущего состояния воркеров
- Планируйте специальные предложения в периоды низкой загрузки системы

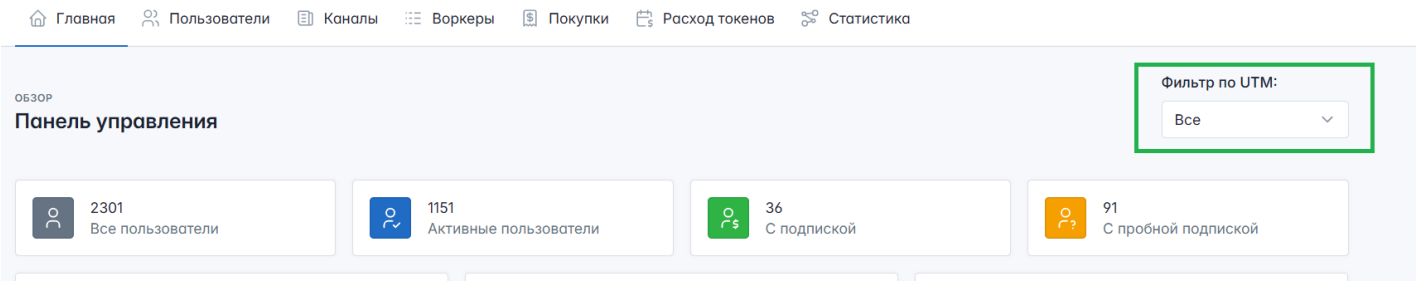
Хотя страница “Воркеры” имеет преимущественно техническую направленность, понимание этих данных позволяет маркетологам принимать обоснованные решения, реалистично планировать рост и обеспечивать согласованность между маркетинговыми обещаниями и техническими возможностями сервиса.

Страница Главная



Основные элементы интерфейса

Фильтр по UTM-меткам



В верхней части страницы расположен выпадающий список для фильтрации данных по UTM-меткам:

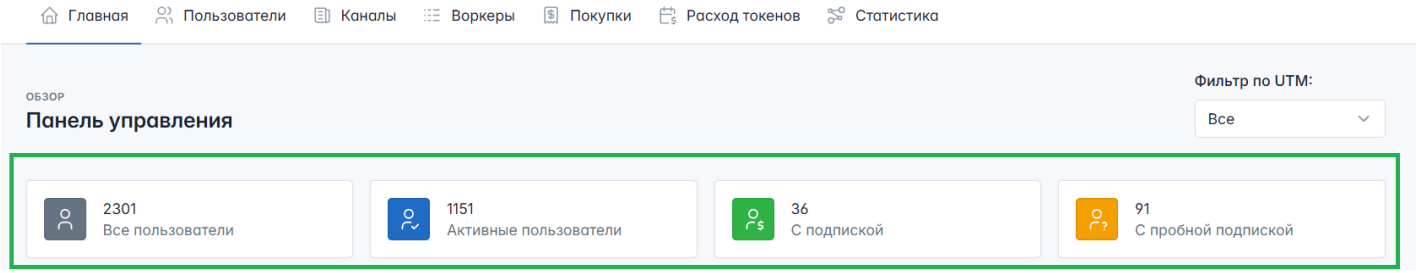
- Позволяет выбрать конкретный источник трафика для анализа
- Показывает количество пользователей по каждому источнику в скобках
- Опция “Все” отображает данные без фильтрации

Как использовать: Выберите нужную UTM-метку из выпадающего списка для анализа эффективности конкретного канала привлечения.

Польза: Быстрое сравнение эффективности различных маркетинговых каналов и кампаний.

Ключевые метрики

Четыре карточки в верхней части дашборда отображают основные показатели:



1. Все пользователи

- Общее количество пользователей в системе
- При нажатии переходит к полному списку пользователей

2. Активные пользователи

- Количество пользователей, активных на текущий момент
- При нажатии переходит к списку активных пользователей

3. Пользователи с подпиской

- Количество пользователей с действующей подпиской (Месячный/Годовой/Подарочный/Реферальный план)
- При нажатии переходит к списку пользователей с подпиской

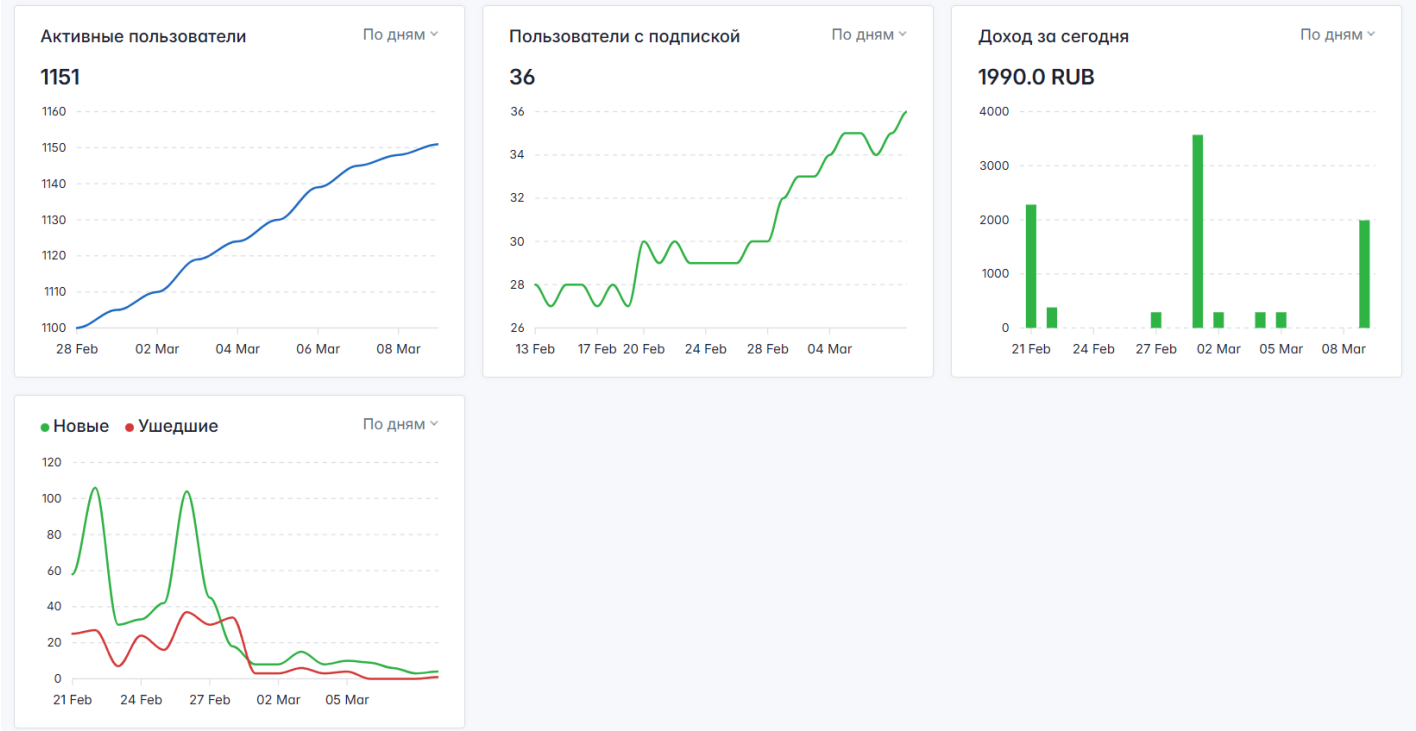
4. Пользователи с пробной подпиской

- Количество пользователей на пробном периоде
- При нажатии переходит к списку пользователей с триальной подпиской

Как использовать: Мониторинг этих показателей позволяет быстро оценить текущее состояние и динамику роста пользовательской базы.

Графики и аналитика

Дашборд содержит четыре информативных графика:



1. График активности пользователей - Отображает динамику активных пользователей во времени (кто не заблокировал бота) - Помогает определить тренды активности и оценить эффективность удержания

2. График пользователей с подпиской

- Показывает изменение количества пользователей с подпиской (Месячный/Годовой/Подарочный/Реферальный план)
- Позволяет оценить конверсию из триальных пользователей в платящих

3. График прибыли

- Визуализирует финансовые показатели от подписок
- Помогает оценить ROI маркетинговых кампаний

4. График новых и ушедших пользователей

- Сравнение притока новых пользователей с оттоком существующих
- Позволяет оценить устойчивость роста и эффективность удержания

Как использовать: Анализируйте графики для выявления трендов, сезонности и корреляции с проводимыми маркетинговыми активностями.

Польза: Глубокий анализ эффективности маркетинговых кампаний, выявление проблемных мест в воронке конверсии.

Практическое применение дашборда

Анализ эффективности каналов

1. Фильтруйте данные по разным UTM-меткам
2. Сравнивайте количество привлеченных пользователей, их активность и конверсию в платящих

3. Выявляйте наиболее эффективные каналы

Оценка ROI маркетинговых кампаний

1. Сопоставляйте всплески новых пользователей с запусками кампаний
2. Отслеживайте конверсию в платящих пользователей по каждому каналу

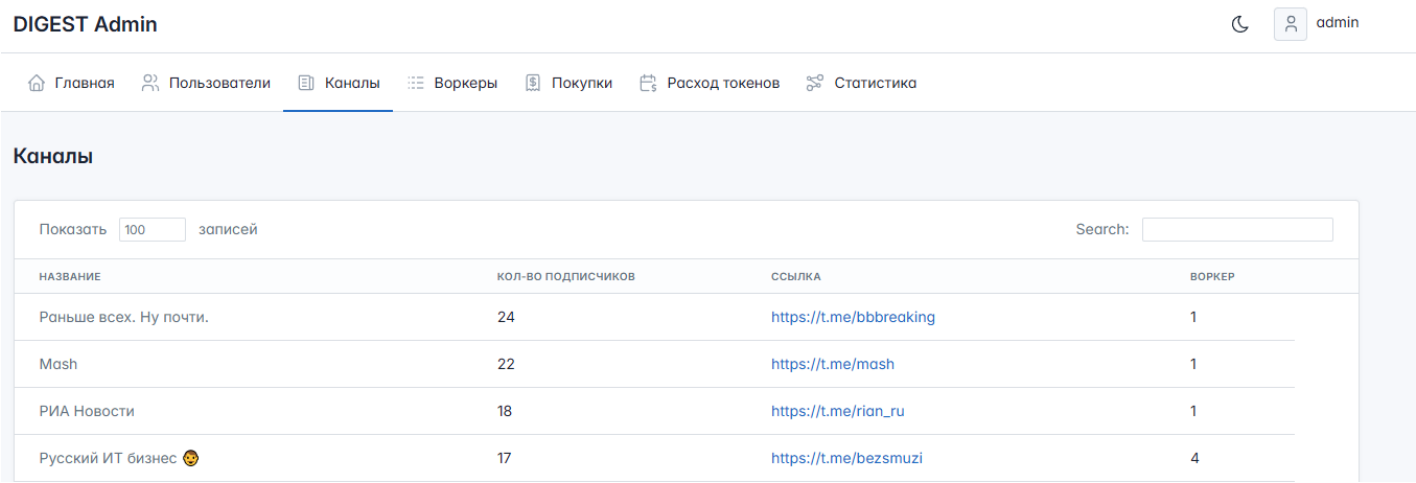
Оптимизация стратегии удержания

1. Анализируйте отток пользователей после пробного периода
2. Выявляйте сегменты с высоким показателем конверсии в платящих пользователей
3. Разрабатывайте целевые предложения для повышения удержания

Рекомендации по использованию

- Регулярно (еженедельно/ежемесячно) анализируйте динамику ключевых показателей
- Синхронизируйте анализ с календарем маркетинговых активностей
- Используйте дополнительную фильтрацию по UTM для детального анализа эффективности отдельных кампаний
- Сравнивайте показатели разных периодов для выявления тенденций

Страница Каналы

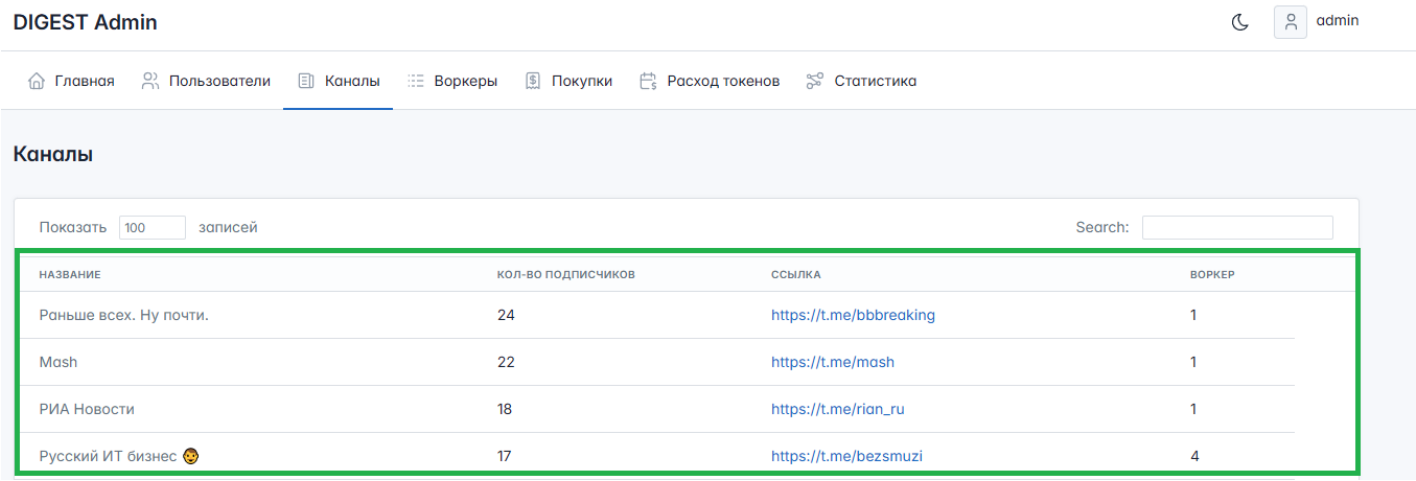


Обзор

Страница “Каналы” предоставляет полный обзор всех каналов, на которые подписаны пользователи бота. Этот раздел позволяет анализировать популярность различных каналов, определять тренды в предпочтениях аудитории и выявлять наиболее перспективные направления для маркетинговых активностей.

Основные возможности

Список каналов



На странице отображается таблица со следующими данными о каналах: - Название канала - Количество подписчиков (пользователей бота, подписанных на этот канал) - Ссылка на канал в Телеграме - Идентификатор воркера (технический работник, обслуживающий этот канал)

Функциональность страницы

DIGEST Admin

admin

Главная | Пользователи | Каналы | Воркеры | Покупки | Расход токенов | Статистика

Каналы

Показать 100 записей

Search: Mash

НАЗВАНИЕ	КОЛ-ВО ПОДПИСЧИКОВ	ССЫЛКА	ВОРКЕР
Mash	22	https://t.me/mash	1
Маша Полуянова	2	https://t.me/mashapoluyanova	4
Masha Ivakova	1	https://t.me/mashaivakova	1
Mash на Мойке	1	https://t.me/mashmoyka	6
Babr Mash	1	https://t.me/babr_mash	6
Ural Mash	1	https://t.me/url_mash	5
Well-reading club by Masha ...	1	https://t.me/wellreadingclub	1
Mashkka про Data Science	1	https://t.me/mashkka_ds	8
@masha_tg_ads	1	https://t.me/masha_tg_ads	1
Ni Mash	1	https://t.me/mash_nimash	7

Страница 1 из 1

prev 1 next

- **Поиск**: Быстрый поиск по названию канала - **Настройка отображения**: Возможность выбора количества отображаемых каналов на странице - **Пагинация**: Удобная навигация по страницам с большим количеством каналов - **Переход к детальной информации**: По клику на название канала открывается страница с подробной информацией

Детальная страница канала

DIGEST Admin

admin

Главная | Пользователи | Каналы | Воркеры | Покупки | Расход токенов | Статистика

Информация о канале

-1001117628569

НАЗВАНИЕ	ССЫЛКА	ПОДПИСЧИКИ	ВОРКЕР
Mash	https://t.me/mash	22	1

Подписчики

Все | Пробный план | С платной подпиской | Месячный план | Годовой план

Показать 100 записей

Search:

UTM: Все

	USER ID	ИМЯ	НИК	ДАТА РЕГИСТРАЦИИ	ДАТА ОКОНЧАНИЯ ПОДПИСКИ	СТАТУС	ПОДПИСКИ	ИСТОЧНИК
KO	1147932229	Константин	@von_waterloo	26-02-2025 20:42	21-03-2025 00:26	Пробная	6	-
AH	386147583	Анна	@afmakarova	26-02-2025 10:13	05-03-2025 10:13	Без подписки	10	media1
SE	194773805	Sergio	@pescadotravel	26-02-2025 09:23	05-03-2025 09:23	Без подписки	7	media1
IG	206287931	Igor	@ТАРАКАНАТОР	22-02-2025 17:46	01-03-2025 17:46	Без подписки	9	rabynagalerah
AL	1076142082	Alex	@chydo_dev	19-02-2025 19:36	14-03-2025 08:33	Пробная	10	-

При переходе к информации о конкретном канале вы получите доступ к: - Детальной информации о канале (ID, название, ссылка, количество подписчиков, id воркера) - Списку пользователей, подписанных на этот канал, с возможностью фильтрации

Как использовать для маркетинговых задач

Анализ популярности контента

1. **Выявление популярных тематик**:
   - Проанализируйте, какие тематики привлекают наибольшее число пользователей
   - Используйте эти данные для формирования контент-стратегии

Сегментация аудитории по интересам

1. **Создание групп пользователей по интересам**:
   - Изучите пересечения аудиторий разных тематических каналов
   - Формируйте сегменты на основе комбинаций подписок пользователей
2. **Анализ предпочтений различных сегментов**:
   - Сравните предпочтения пользователей с разными типами подписок (пробная/платная)
   - Выявите каналы, подписчики которых чаще всего конвертируются в платящих пользователей

Оптимизация партнерских программ

1. **Выбор каналов для партнерства**:
   - Определите наиболее популярные каналы для размещения партнерских материалов
   - Отслеживайте эффективность различных партнерских размещений
2. **Мониторинг активности каналов**:
   - Выявляйте неактивные или малоэффективные каналы для исключения из партнерской программы

Практические кейсы использования

1. **Разработка контент-плана**:
   - Изучите топ-10 популярных каналов среди ваших пользователей
   - Проанализируйте типы и форматы контента в этих каналах
   - Адаптируйте успешные форматы для вашего контента
2. **Таргетирование рекламных кампаний**:
   - Используйте информацию о подписках для более точного таргетинга рекламы
   - Создавайте look-alike аудитории на основе подписчиков определенных каналов
3. **Разработка специальных предложений**:
   - Создавайте таргетированные предложения для подписчиков определенных каналов
   - Тестируйте различные форматы акций для разных сегментов аудитории

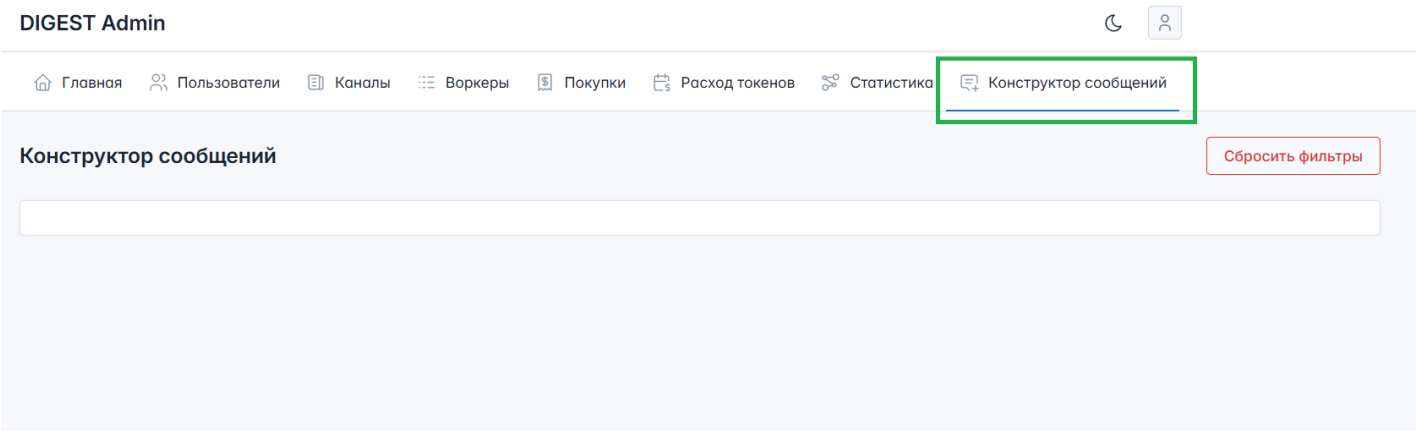
Советы по эффективному использованию

- Регулярно анализируйте изменения в распределении подписчиков между каналами
- Сопоставляйте данные о популярных каналах с данными о конверсии в платные подписки
- Используйте информацию о подписках для улучшения рекомендательной системы
- Периодически проверяйте активность наиболее популярных каналов
- Отслеживайте появление новых каналов с быстрорастущей аудиторией для раннего выявления трендов

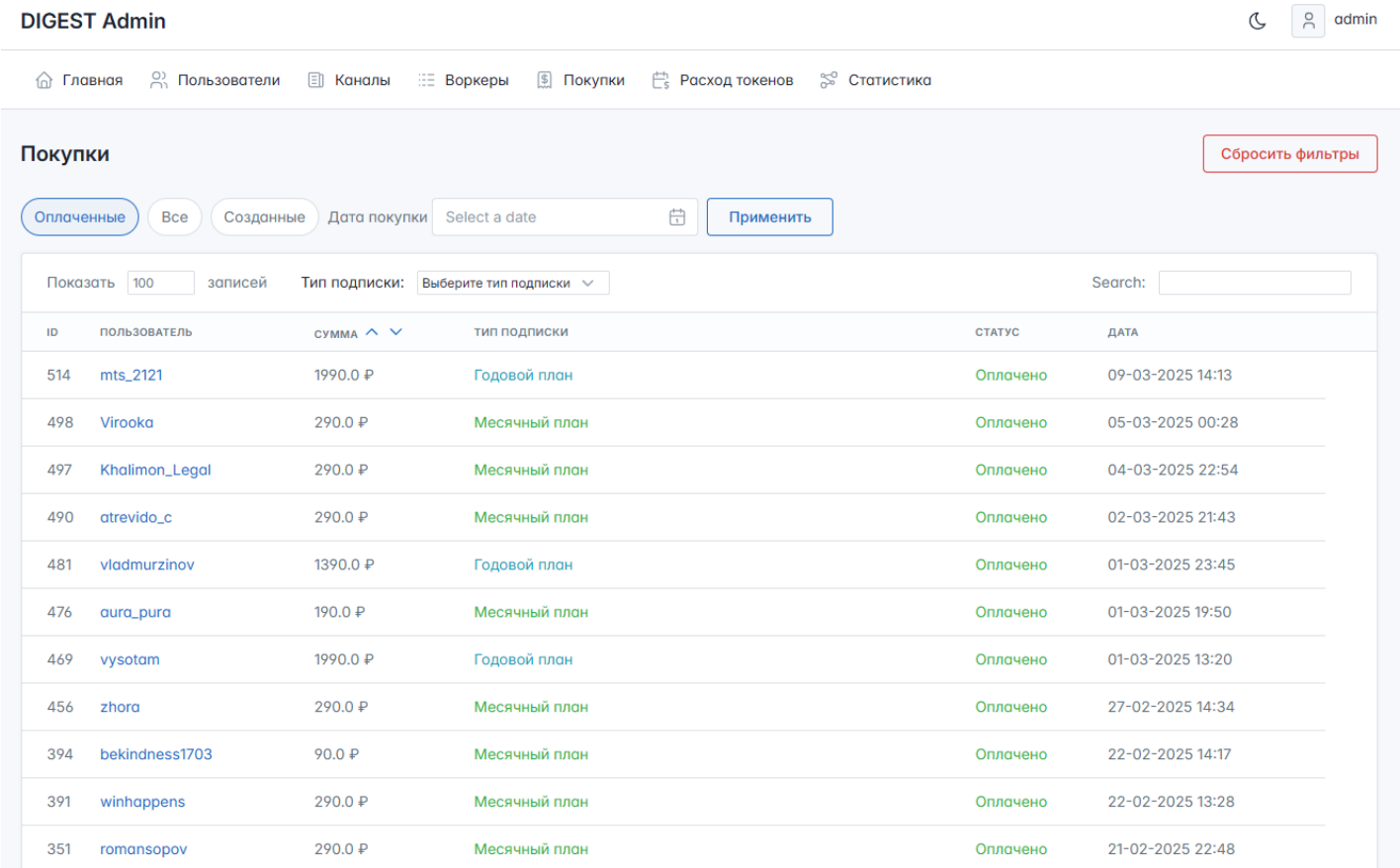
Страница “Каналы” предоставляет ценные данные для понимания интересов аудитории, что позволяет более эффективно настраивать маркетинговые кампании, улучшать контент-стратегию и оптимизировать расходы на привлечение новых пользователей.

Страница Конструктор сообщений

На данный момент страница в разработке



Страница Покупки

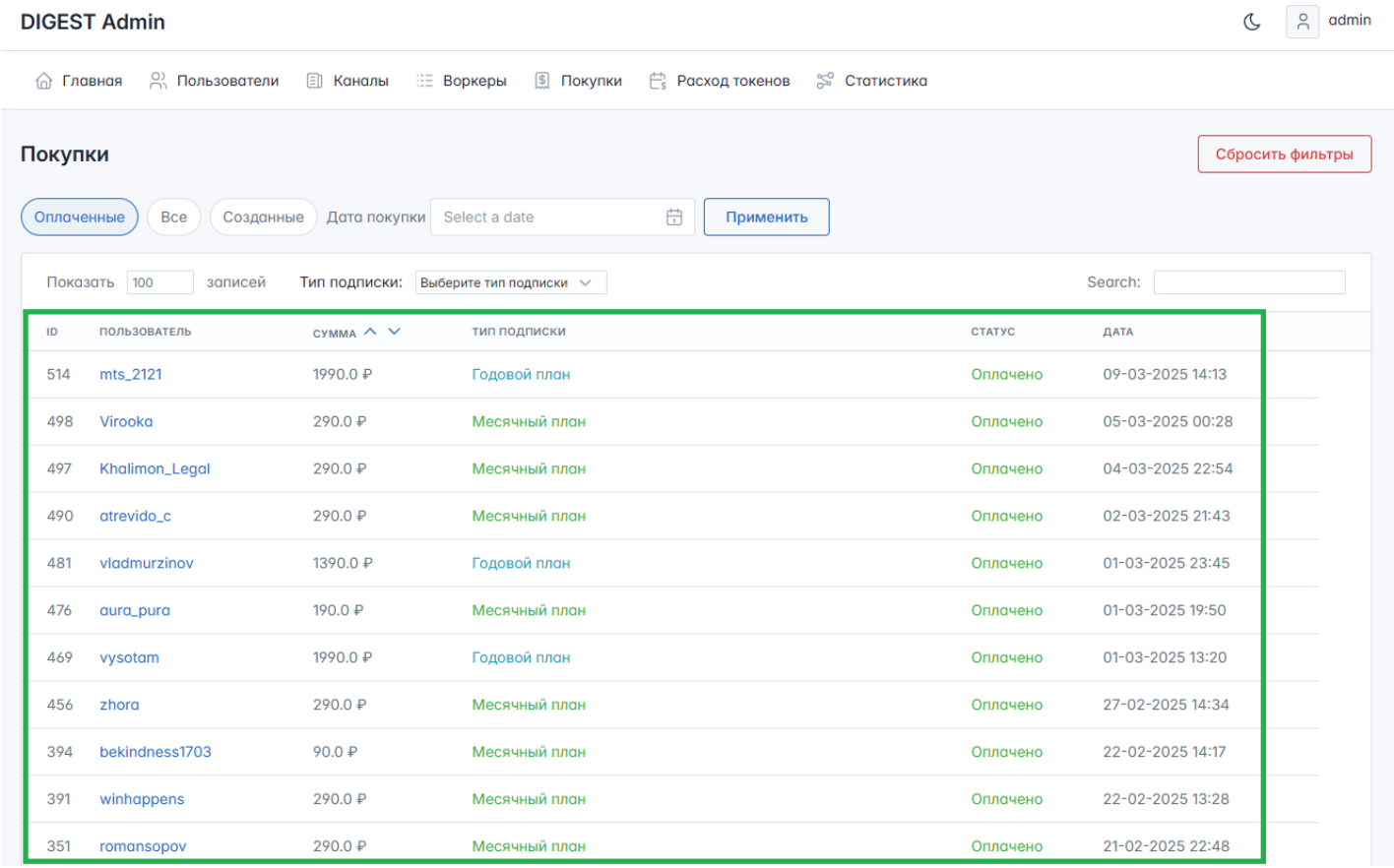


Обзор

Страница “Покупки” предоставляет полную информацию о финансовых транзакциях пользователей в системе. Этот раздел является ключевым инструментом для финансового анализа, оценки эффективности ценовой политики и мониторинга доходов от различных типов подписок. Здесь собраны все платежи пользователей с подробной информацией о каждой транзакции.

Основные возможности

Список платежей



На странице отображается таблица со следующими данными о каждой транзакции: - ID платежа - Пользователь (с прямой ссылкой на профиль) -

Сумма платежа в рублях - Тип подписки - Статус платежа (оплачено, создано) - Дата и время создания платежа

Фильтрация платежей

По статусу платежа

DIGEST Admin

admin

Главная | Пользователи | Каналы | Воркеры | Покупки | Расход токенов | Статистика

Покупки

Сбросить фильтры

Оплаченные | Все | Созданные | Дата покупки | Select a date | Применить

Показать 100 записей | Тип подписки: Выберите тип подписки | Search:

ID	ПОЛЬЗОВАТЕЛЬ	СУММА	ТИП ПОДПИСКИ	СТАТУС	ДАТА
514	mts_2121	1990.0 ₽	Годовой план	Оплачено	09-03-2025 14:13
513	artyrvaleev	1990.0 ₽	Годовой план	new	09-03-2025 14:07
512	VVCUUM	290.0 ₽	Месячный план	new	08-03-2025 13:56
511	endorforce	290.0 ₽	Месячный план	new	07-03-2025 09:49
510	mrnikbur	290.0 ₽	Месячный план	new	07-03-2025 08:32
509	wildestduck	290.0 ₽	Месячный план	new	07-03-2025 03:57
508	ZZZurich	290.0 ₽	Месячный план	new	05-03-2025 21:22
507	kssrak	290.0 ₽	Месячный план	new	05-03-2025 18:30
503	oculii	1990.0 ₽	Годовой план	new	05-03-2025 15:45
502	oculii	290.0 ₽	Месячный план	new	05-03-2025 15:36
501	Juravlptica	290.0 ₽	Месячный план	new	05-03-2025 11:25
500	webgromov	290.0 ₽	Месячный план	new	05-03-2025 03:47
499	Chief_train	1990.0 ₽	Годовой план	new	05-03-2025 01:26
498	Virooka	290.0 ₽	Месячный план	Оплачено	05-03-2025 00:28

- **Оплаченные** - только успешно завершенные транзакции (выбран по умолчанию) - **Все** - все платежи без фильтрации по статусу - **Созданные** - платежи, которые были инициированы, но еще не завершены

По дате покупки

DIGEST Admin

admin

Главная | Пользователи | Каналы | Воркеры | Покупки | Расход токенов | Статистика

Покупки

Сбросить фильтры

Оплаченные | Все | Созданные | Дата покупки | 2025-03-01 | Применить

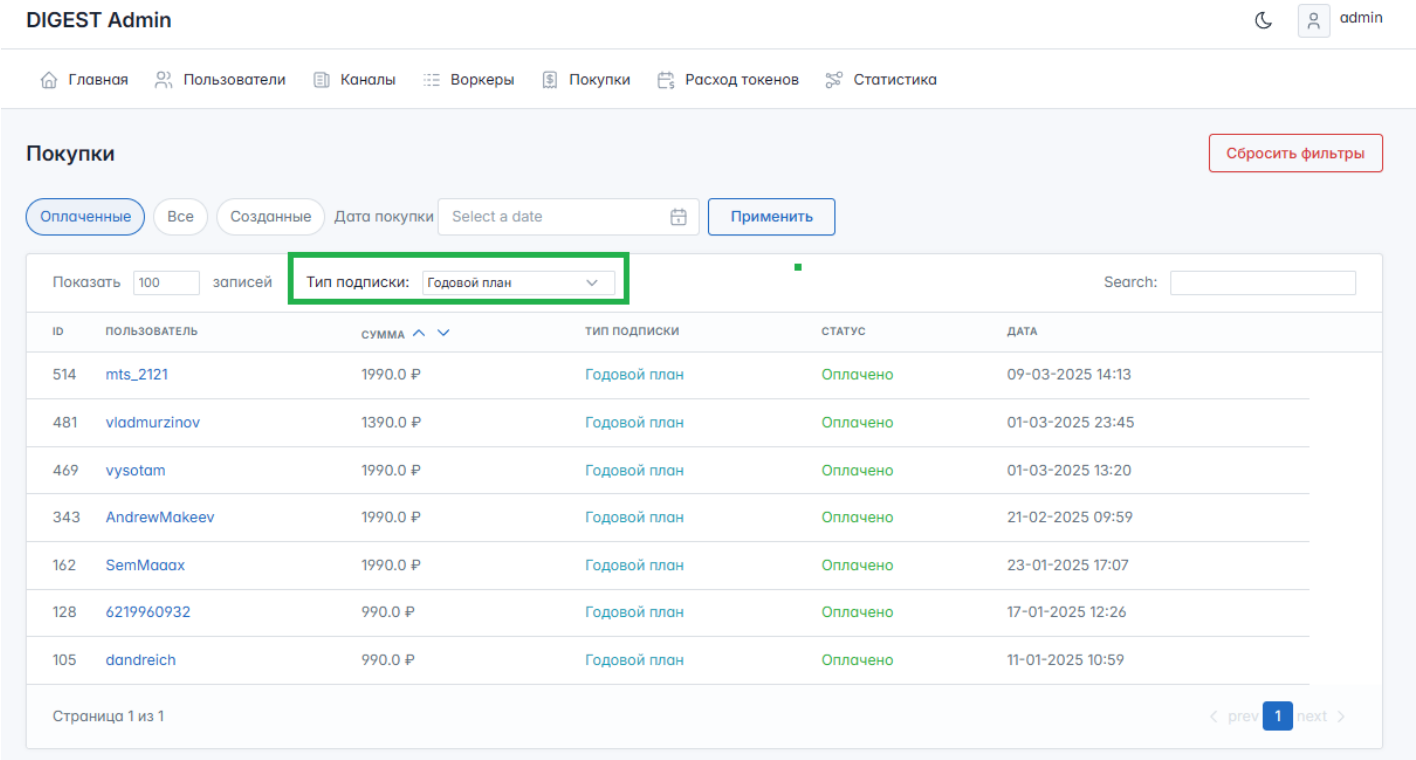
Показать 100 записей | Тип подписки: Выберите тип подписки | Search:

ID	ПОЛЬЗОВАТЕЛЬ	СУММА	ТИП ПОДПИСКИ	СТАТУС	ДАТА
481	vladmurzinov	1390.0 ₽	Годовой план	Оплачено	01-03-2025 23:45
476	aura_pura	190.0 ₽	Месячный план	Оплачено	01-03-2025 19:50
469	vysotam	1990.0 ₽	Годовой план	Оплачено	01-03-2025 13:20

Страница 1 из 1 | prev | 1 | next

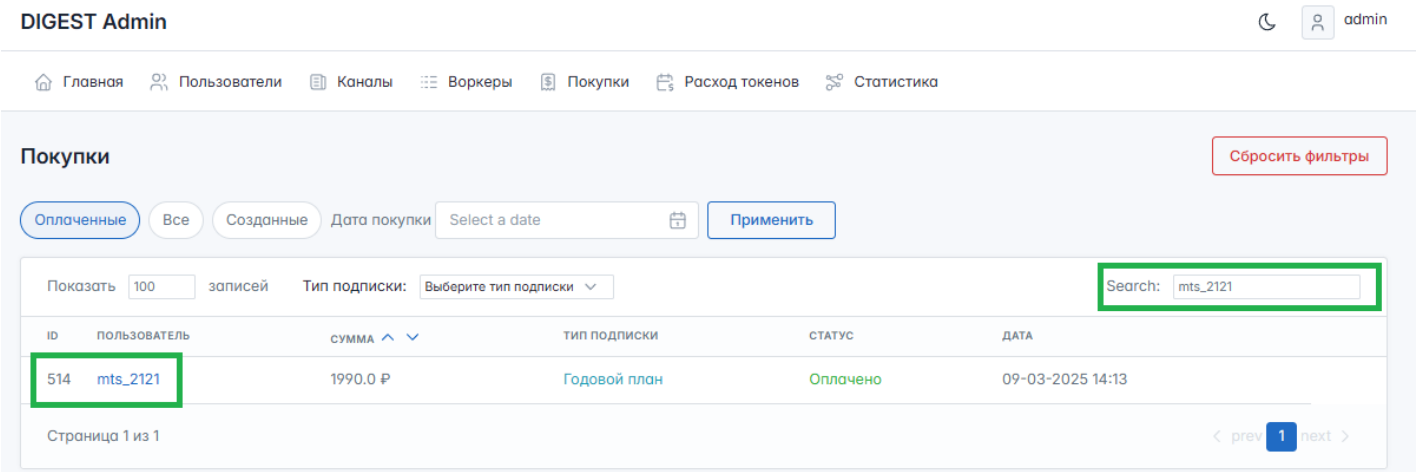
Возможность фильтрации платежей по конкретной дате с помощью календаря.

По типу подписки

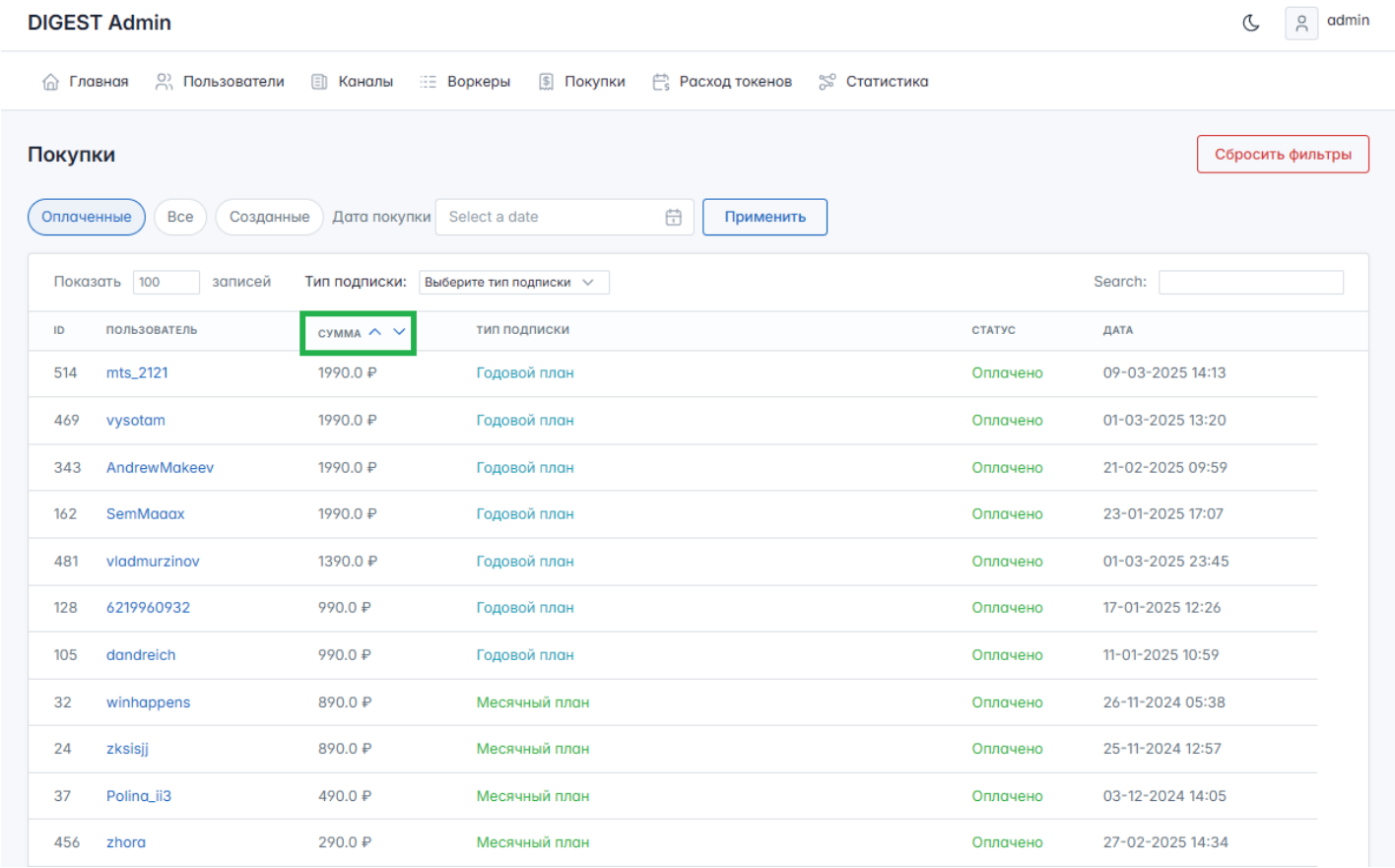


- **Месячный план** - платежи за месячную подписку - **Годовой план** - платежи за годовую подписку

Функции поиска и сортировки



- **Поиск по пользователю** - возможность найти платежи конкретного пользователя



- **Сортировка по сумме** - возможность сортировать платежи по возрастанию или убыванию суммы - **Настройка отображения** - выбор количества записей на странице

Как использовать для маркетинговых задач

Анализ продаж и доходов

1. **Мониторинг продаж**:
   - Отслеживайте ежедневные, еженедельные и месячные продажи
   - Анализируйте динамику продаж по типам подписок
   - Выявляйте дни недели или периоды с наибольшим количеством покупок
2. **Оценка эффективности ценовой политики**:
   - Сравнивайте популярность различных типов подписок (месячная vs годовая)
   - Анализируйте средний чек и частоту покупок
   - Оценивайте соотношение годовых и месячных подписок для понимания лояльности пользователей

Оценка эффективности маркетинговых кампаний

1. **Измерение ROI рекламных кампаний**:
   - Фильтруйте продажи по датам проведения рекламных кампаний
   - Сопоставляйте затраты на рекламу с полученным доходом в эти периоды
   - Оценивайте эффективность скидок и специальных предложений
2. **Анализ сезонности продаж**:
   - Выявляйте сезонные тренды в продажах
   - Планируйте маркетинговые активности с учетом выявленных закономерностей
   - Корректируйте рекламный бюджет в соответствии с периодами повышенного/пониженного спроса

Анализ пользовательского поведения

1. **Изучение платежного поведения**:
   - Анализируйте, какие пользователи предпочитают более дорогие подписки
   - Определяйте временной промежуток между активацией аккаунта и первой покупкой
2. **Выявление проблем в платежном процессе**:
   - Мониторьте соотношение созданных и оплаченных платежей
   - Выявляйте этапы, на которых пользователи чаще всего отказываются от покупки
   - Идентифицируйте технические проблемы, которые могут препятствовать завершению платежа

Практические кейсы использования

1. **Анализ воздействия акций и скидок**:
   - Сравните количество и сумму платежей до, во время и после проведения акции
   - Оцените, какие типы подписок пользуются большим спросом во время скидок
   - Определите оптимальный размер скидки для максимизации дохода
2. **Прогнозирование доходов**:
   - Используйте исторические данные о платежах для прогнозирования будущих доходов
   - Анализируйте сезонные тренды для более точного планирования бюджета
   - Создавайте финансовые модели с учетом различных маркетинговых сценариев
3. **Оптимизация ценовой стратегии**:

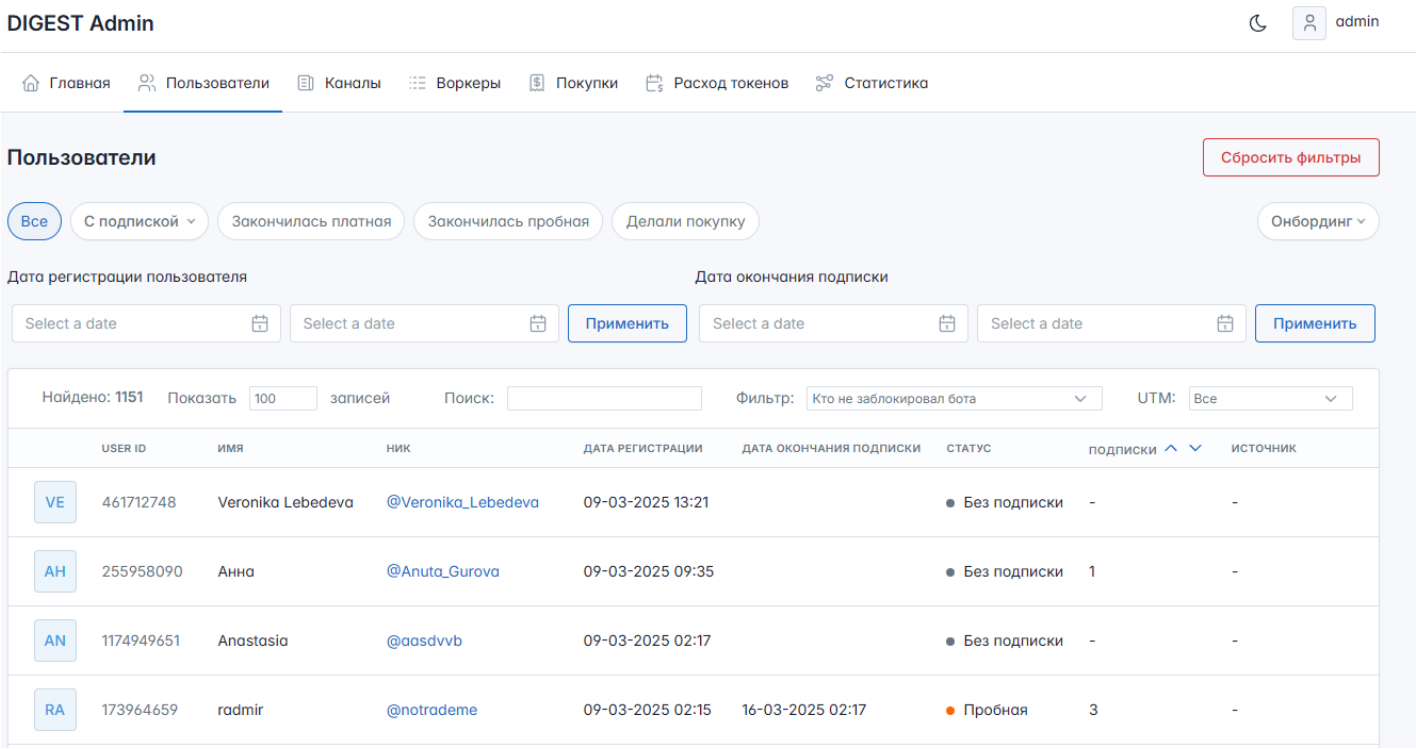
    - Исследуйте отношение пользователей к различным ценовым предложениям
    - Анализируйте влияние изменения цен на объем продаж
    - Тестируйте различные ценовые сегменты для нахождения оптимальной цены
4. **Анализ пользователей с высокой стоимостью**:
    - Идентифицируйте пользователей, которые совершают наиболее дорогие покупки
    - Изучите их характеристики и поведение
    - Создавайте таргетированные кампании для привлечения похожих пользователей

Советы по эффективному использованию

- Регулярно сопоставляйте данные о продажах с другими метриками (активность пользователей, источники трафика и т.д.)
- Создавайте еженедельные и ежемесячные отчеты для отслеживания динамики продаж
- Используйте фильтр по датам для анализа результатов конкретных маркетинговых кампаний
- Обращайте внимание на соотношение типов подписок для корректировки маркетинговой стратегии
- Отслеживайте незавершенные платежи для выявления возможных проблем в процессе оплаты

Страница “Покупки” является мощным инструментом для финансового анализа и оптимизации маркетинговой стратегии. Правильное использование этого инструмента позволяет не только отслеживать текущие доходы, но и принимать обоснованные решения по ценообразованию, проведению акций и распределению рекламного бюджета.

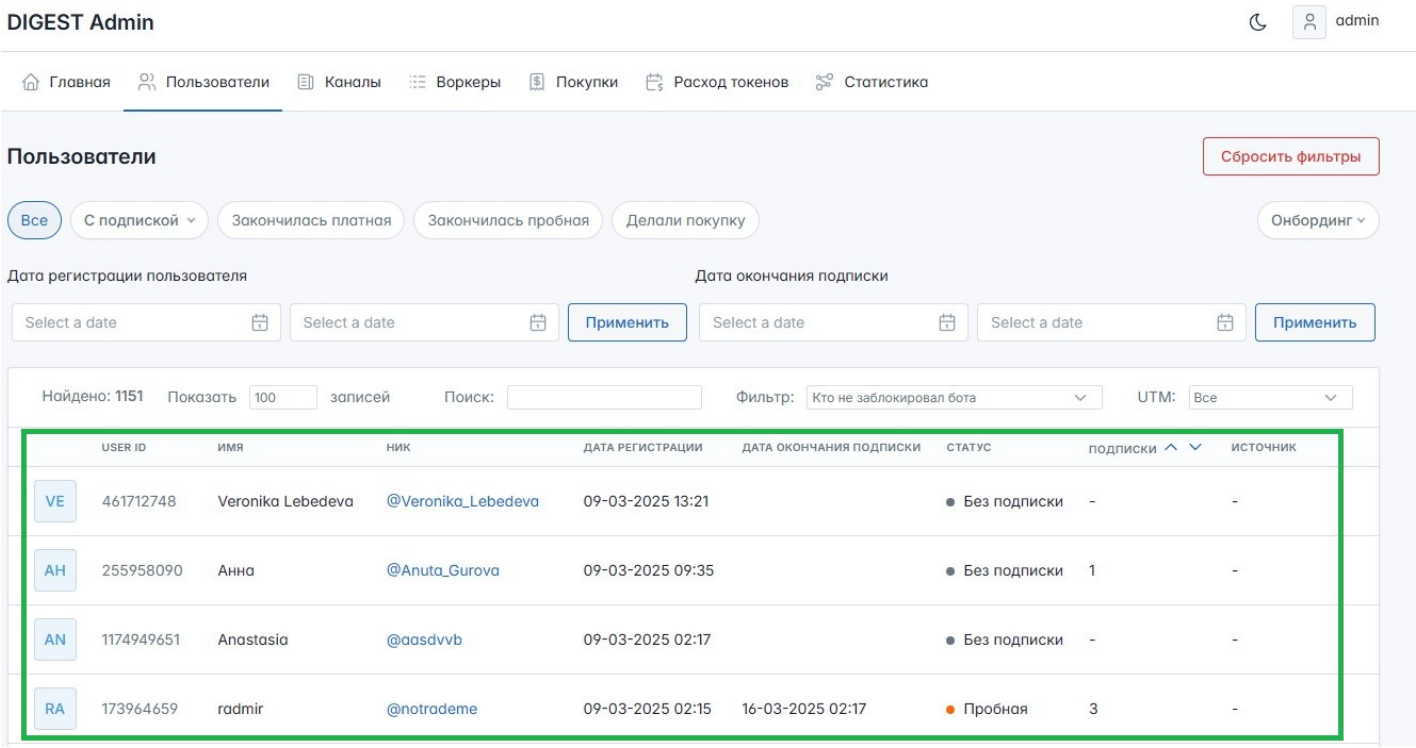
Страница Пользователи



Страница “Пользователи” предоставляет полный доступ к базе пользователей бота и позволяет проводить детальный анализ аудитории. Эта страница является ключевым инструментом для маркетинговой аналитики, сегментации пользователей и отслеживания эффективности кампаний.

Основные возможности

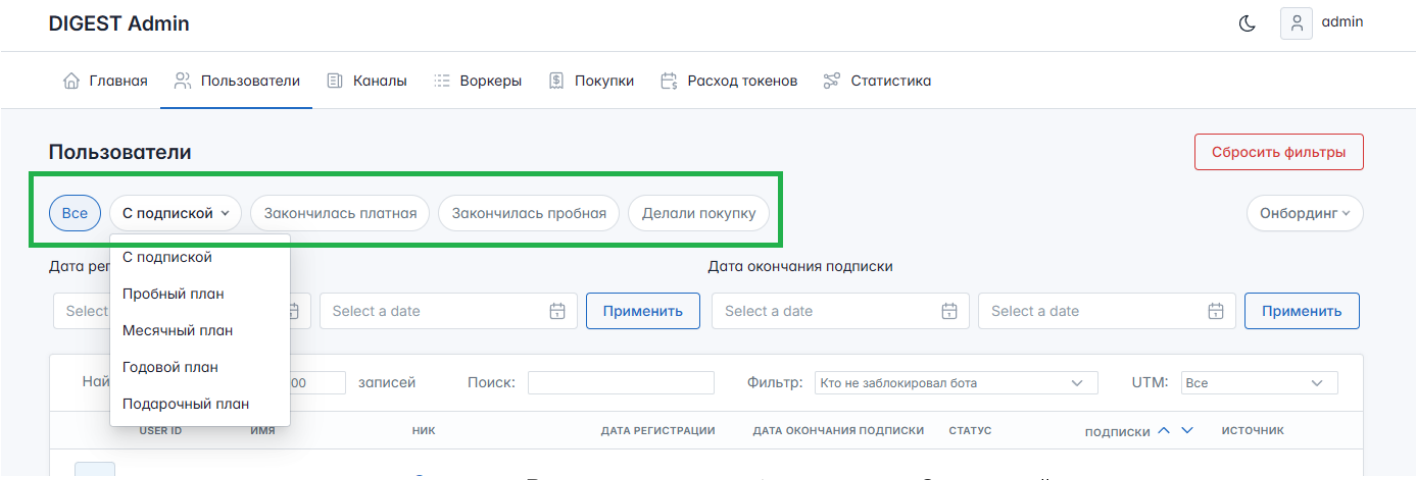
Список пользователей



На странице отображается таблица со следующими данными о пользователях: - ID пользователя - Имя пользователя - Ник в Телеграм (с прямой ссылкой на профиль) - Дата регистрации - Дата окончания подписки - Статус подписки - Количество подписок на каналы - Источник привлечения (UTM-метка)

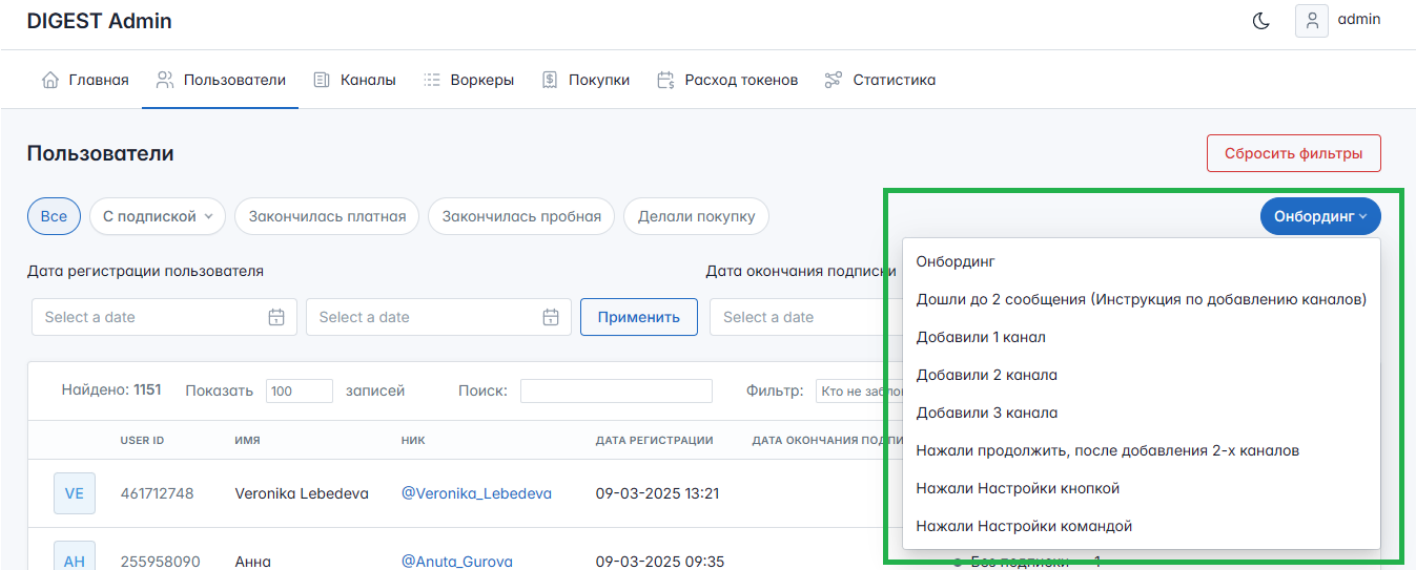
Фильтрация пользователей

По типу подписки



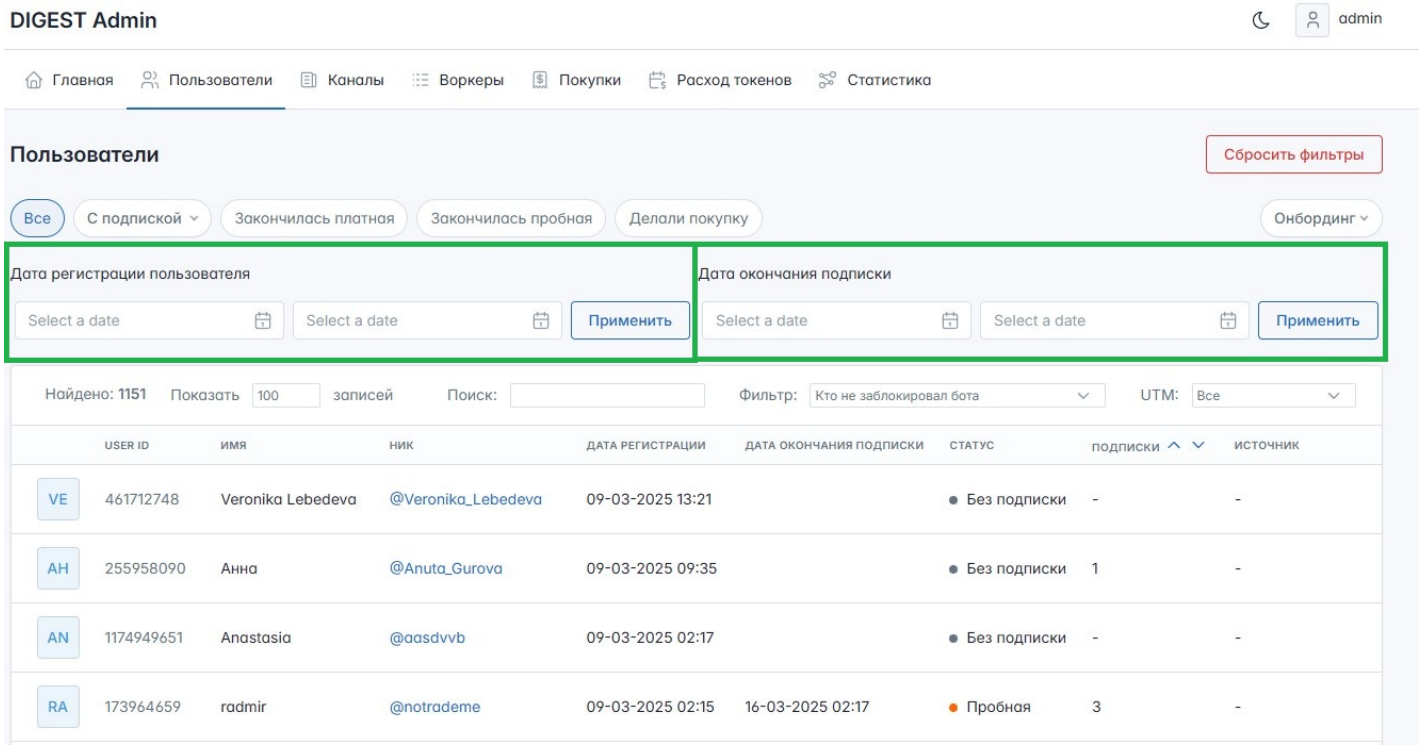
Возможность фильтрации по следующим параметрам: - **Все** - все пользователи без фильтрации - **С подпиской** - пользователи с активной подпиской любого типа (Месячный/Годовой/Подарочный/Реферальный план) - **Пробный план** - пользователи с активной пробной подпиской - **Месячный план** - пользователи с активной месячной подпиской - **Годовой план** - пользователи с активной годовой подпиской - **Подарочный план** - пользователи с подарочной подпиской - **Закончилась платная** - пользователи, у которых закончилась платная подписка - **Закончилась пробная** - пользователи, у которых закончилась только пробная подписка - **Делали покупку** - пользователи, которые хотя бы раз покупали подписку

По этапам онбординга



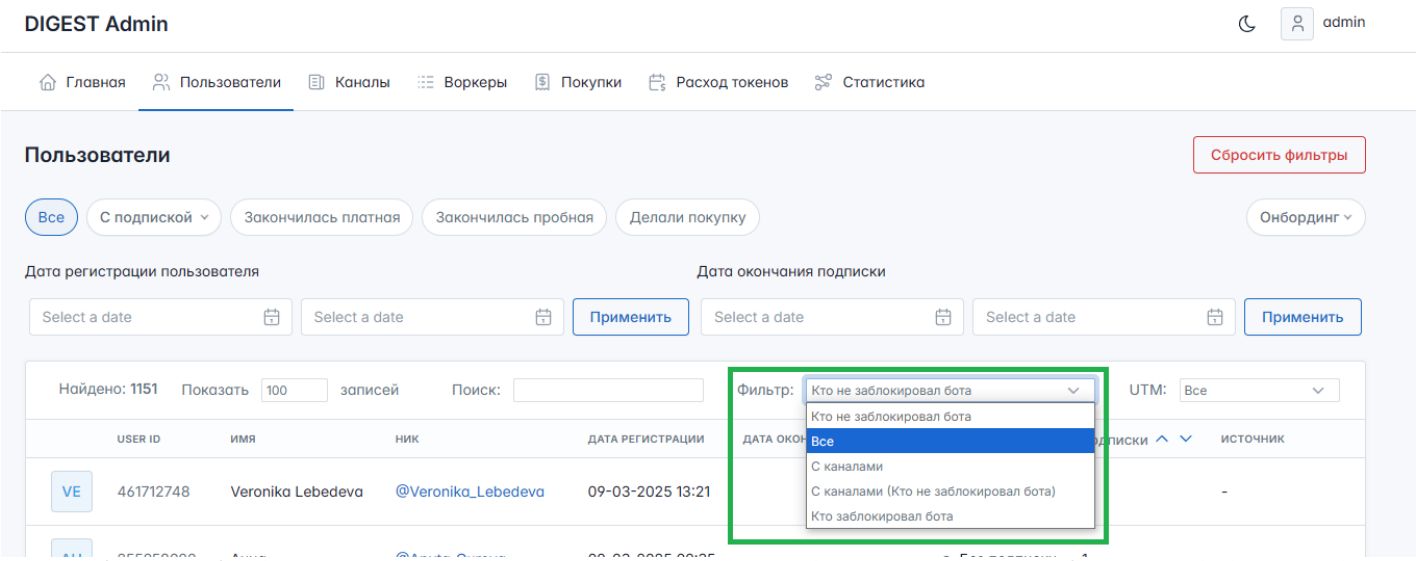
Позволяет анализировать воронку онбординга и отслеживать, на каком этапе пользователи выпадают: - Уникальные нажатия /start - Дошли до 2 сообщения (инструкция по добавлению каналов) - Добавили 1, 2, 3 канала - Нажали продолжить после добавления каналов - Нажали “Настройки” (кнопкой или командой)

По датам



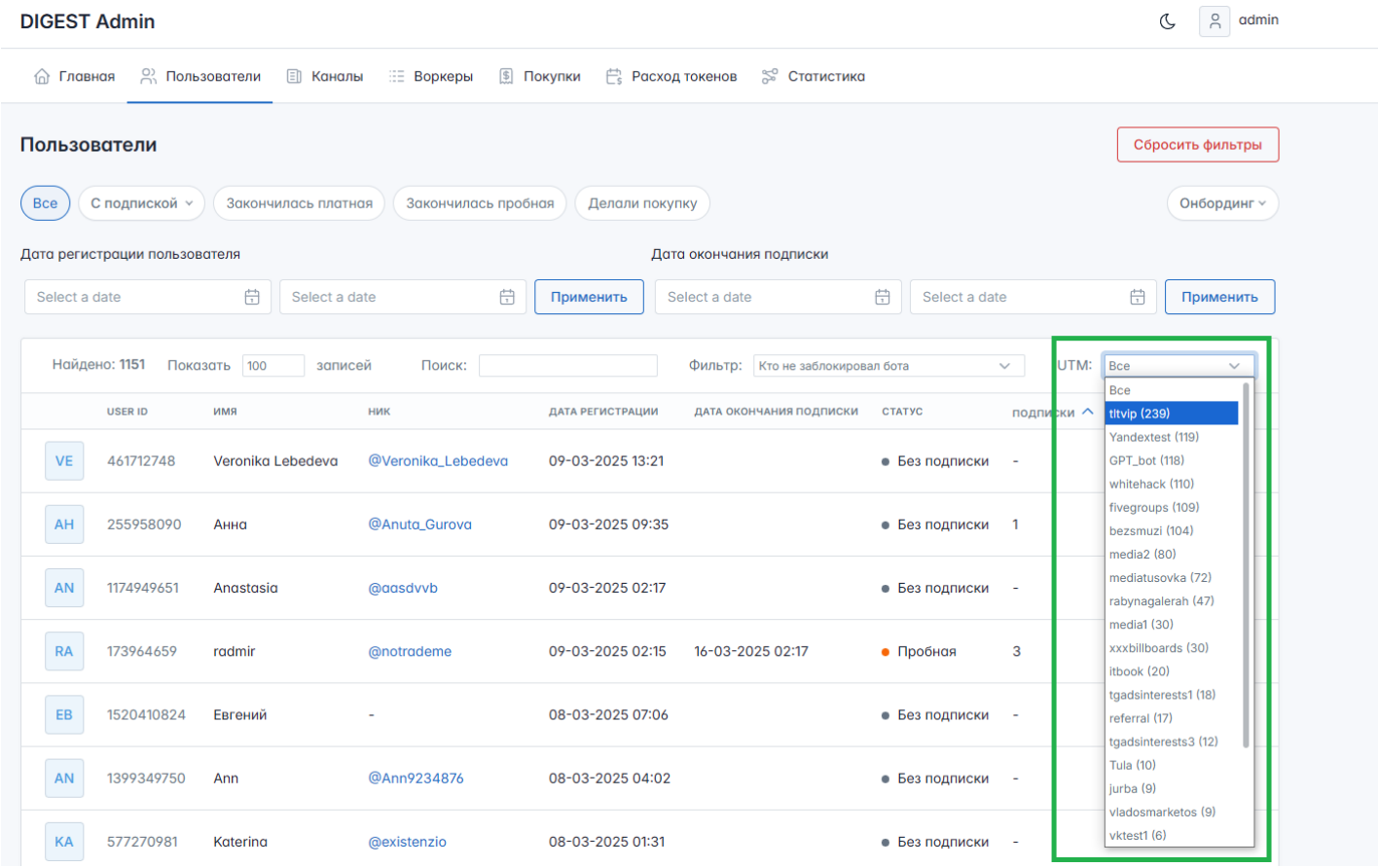
- **Дата регистрации пользователя** - диапазон дат для анализа новых пользователей - **Дата окончания подписки** - диапазон дат для прогнозирования оттока

По статусу пользователя



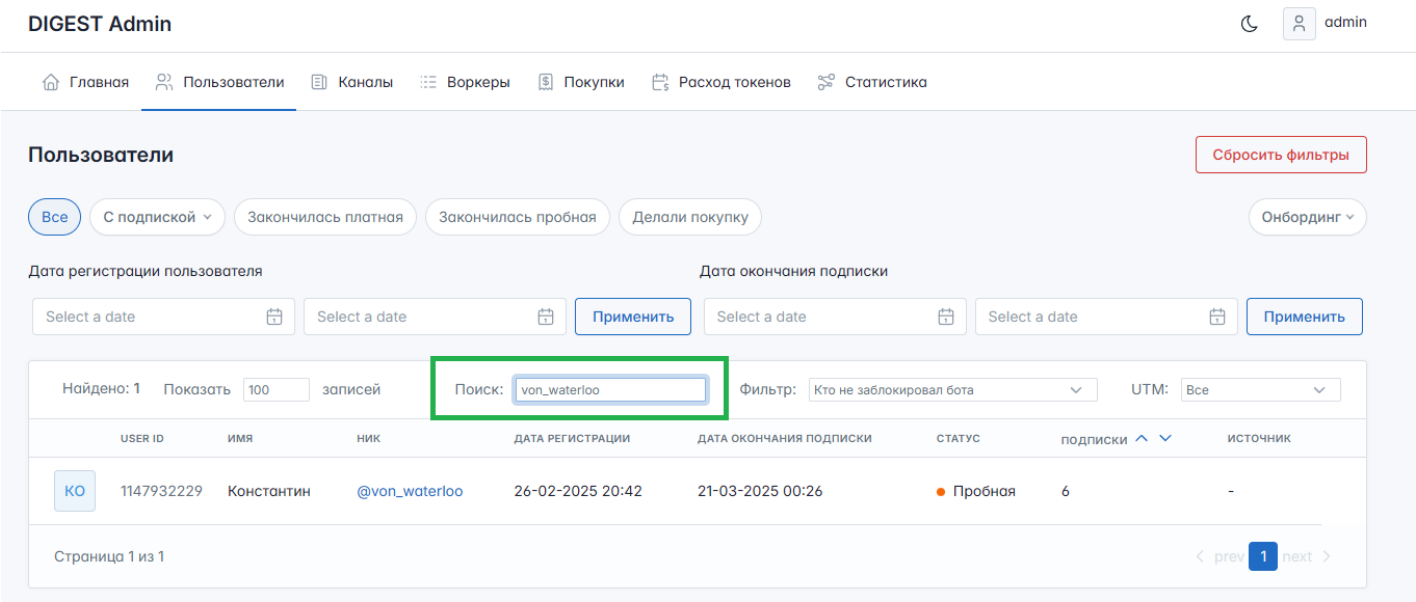
- Кто не заблокировал бота (активные пользователи) - Все пользователи - С каналами (пользователи, которые добавили каналы) - С каналами (кто не заблокировал бота) - Кто заблокировал бота (для анализа причин отказа)

По источнику привлечения (UTM)



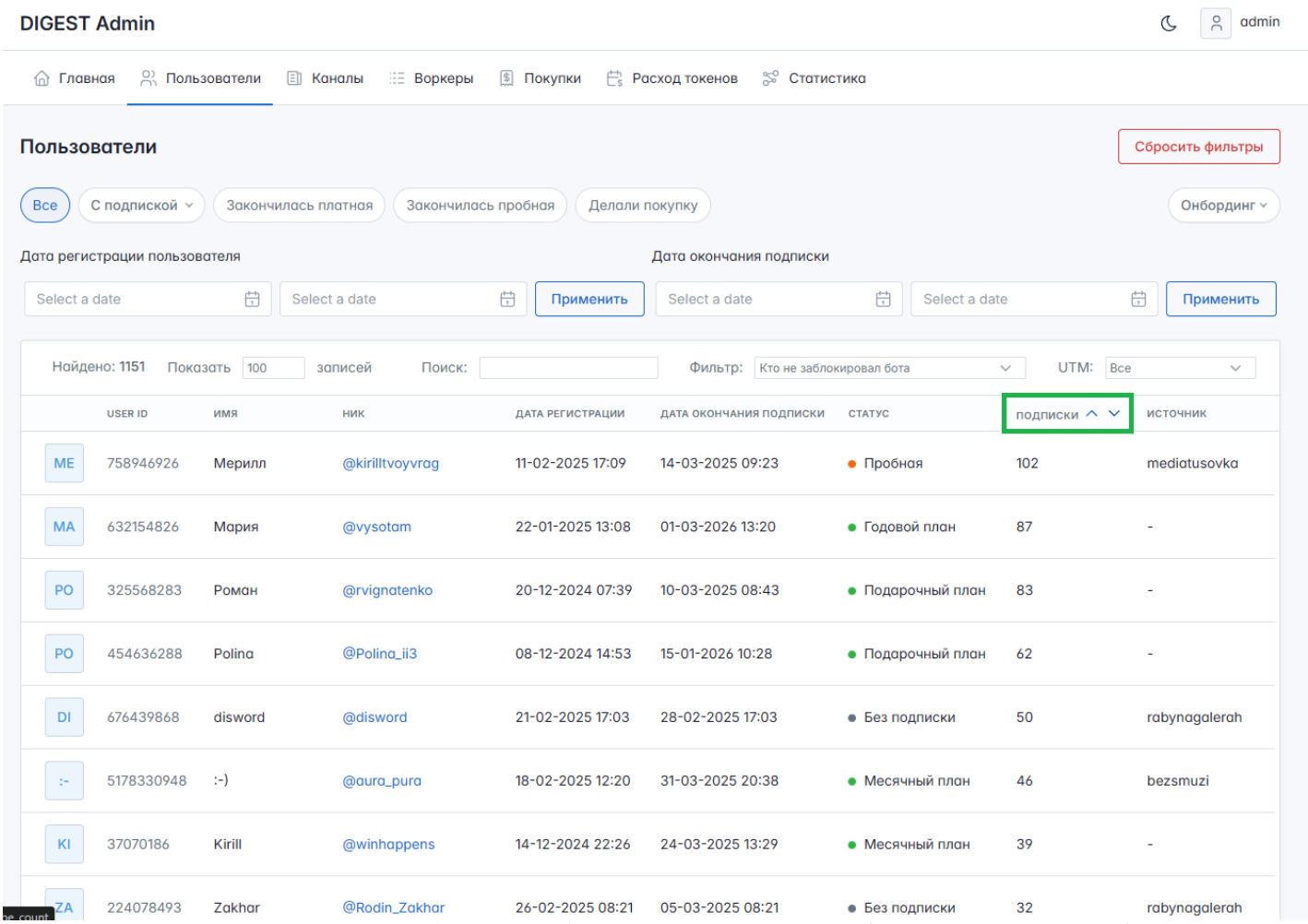
Позволяет анализировать эффективность кампаний по различным UTM-меткам.

Поиск



Возможность быстрого поиска пользователей по имени, ID или нику в Телеграм.

Сортировка



Возможность сортировки по количеству подписок (возрастание/убывание). Позволяет выявлять амбассадоров продукта, кто наиболее заинтересован в использовании.

Как использовать для маркетинговых задач

Анализ эффективности кампаний

1. Используйте фильтр по UTM-меткам, чтобы сравнить конверсию из разных источников
2. Отслеживайте, какие источники привлекают пользователей, которые чаще покупают платные подписки

Анализ воронки онбординга

1. Используйте фильтры по этапам онбординга, чтобы понять, где происходит наибольший отток пользователей
2. Сравнивайте поведение пользователей из разных источников на этапах онбординга

Планирование ретаргетинга

1. Найдите пользователей с истекшими платными подписками для таргетирования специальных предложений
2. Выделите пользователей, которые только попробовали пробный период, но не стали платить

Сегментация аудитории

1. Создавайте сегменты на основе типа подписки и активности (количество добавленных каналов)
2. Анализируйте поведение пользователей в зависимости от типа подписки

Мониторинг оттока

1. Отслеживайте пользователей, которые заблокировали бота
2. Анализируйте, из каких источников чаще всего приходят пользователи, блокирующие бота

Практические кейсы использования

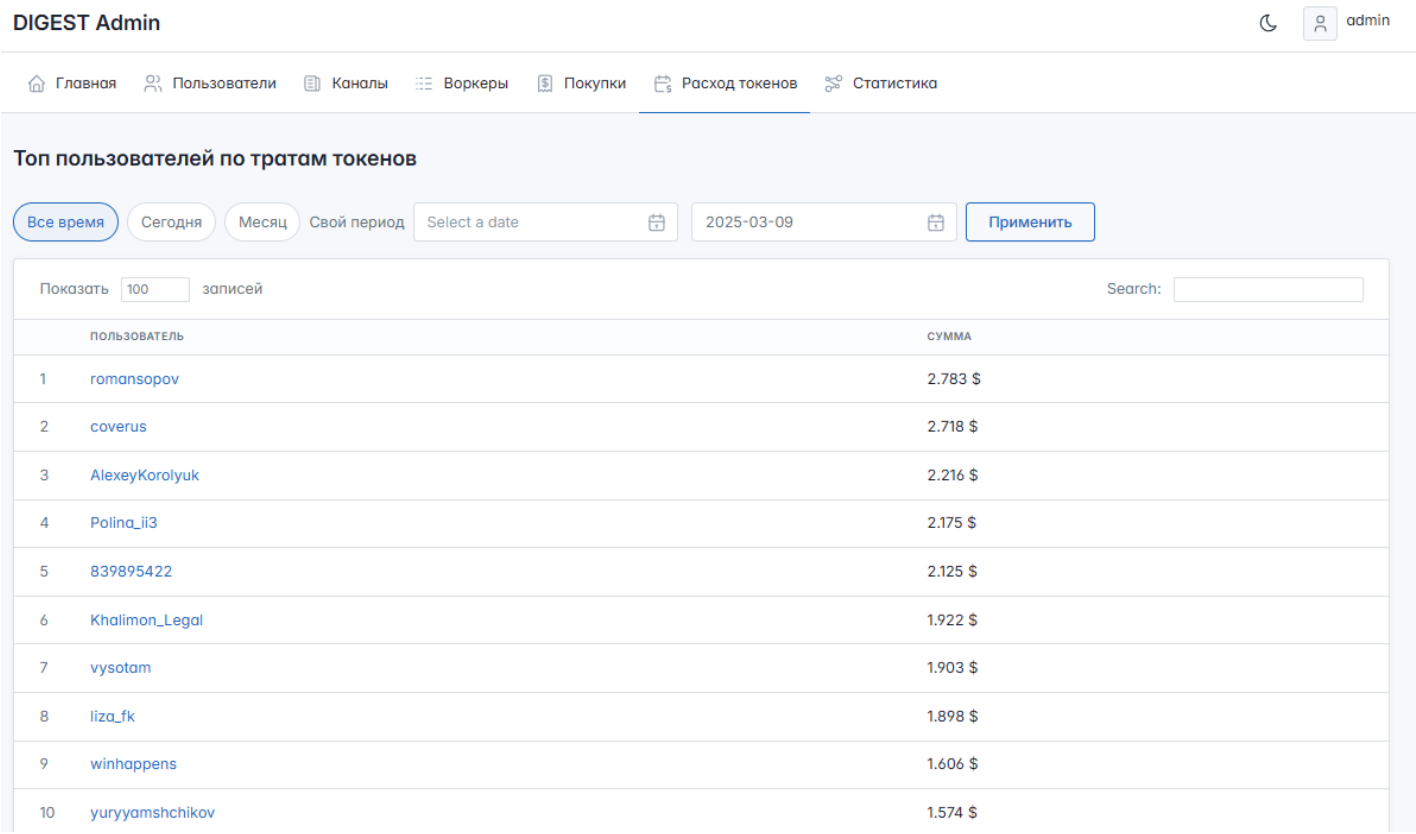
1. **Оценка ROI рекламных кампаний**:
   - Отфильтруйте пользователей по определенной UTM-метке
   - Посмотрите, сколько из них перешли на платную подписку
2. **Выявление проблемных мест в онбординге**:
   - Сравните количество пользователей на разных этапах онбординга
   - Обнаружьте этапы с наибольшим оттоком для дальнейшей оптимизации
3. **Планирование рассылок**:
   - Найдите пользователей с истекающими подписками для отправки предложений о продлении
   - Сегментируйте аудиторию по типу подписки для персонализированных предложений
4. **Анализ лояльности**:
   - Изучите пользователей с годовой подпиской для выявления их особенностей
   - Используйте эти данные для улучшения конверсии других сегментов

Советы по эффективному использованию

- Регулярно анализируйте соотношение пробных и платных подписок по разным каналам привлечения
- Отслеживайте изменение конверсии после внесения изменений в онбординг
- Сравнивайте показатели удержания для разных сегментов пользователей
- Используйте фильтры по датам для оценки сезонных трендов

Страница “Пользователи” является мощным инструментом для маркетингового анализа, позволяющим глубоко понимать аудиторию, отслеживать эффективность кампаний и оптимизировать стратегию привлечения и удержания пользователей.

Страница Расход токенов



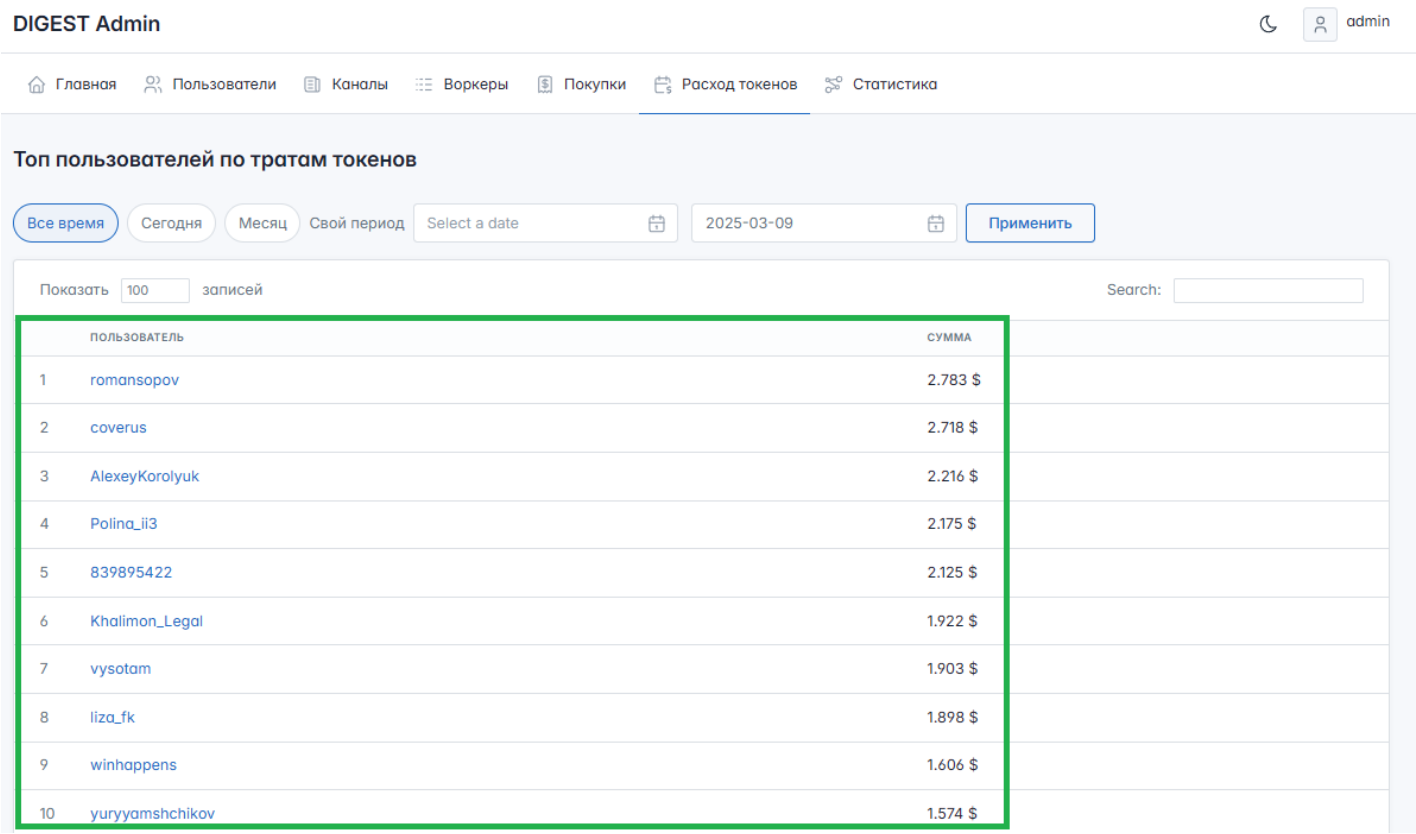
Обзор

Страница “Расход токенов” предоставляет детальную аналитику затрат на сокращение новостей для каждого пользователя. Здесь отображается, сколько компания тратит на обработку и сокращение новостного контента, который потребляет каждый пользователь. Эта информация критически

важна для анализа рентабельности сервиса, оптимизации расходов на AI-обработку новостей и понимания паттернов потребления контента.

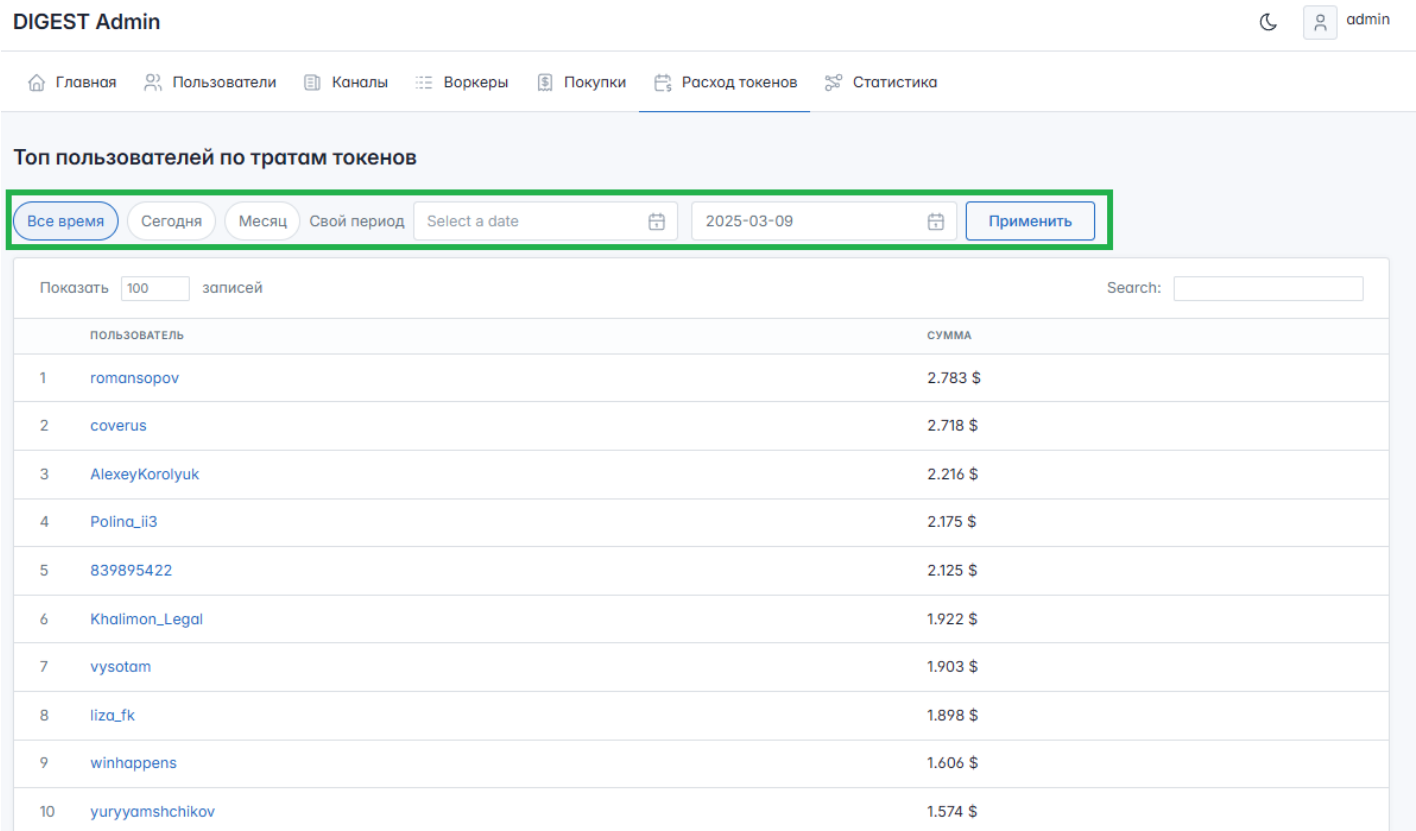
Основные возможности

Рейтинг пользователей по затратам



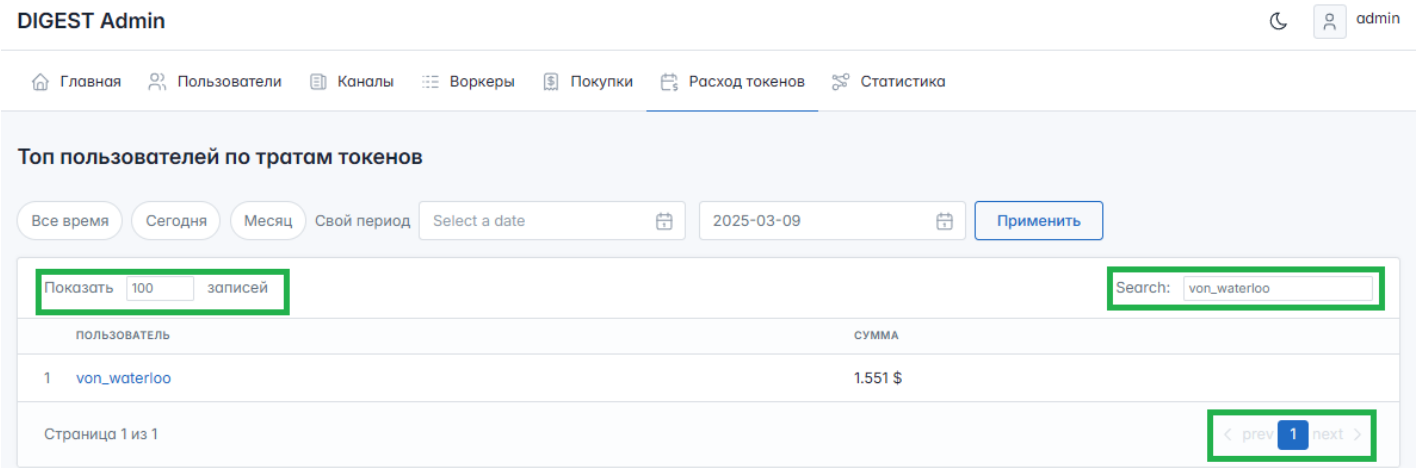
На странице отображается отсортированный список пользователей с указанием: - Порядкового номера в рейтинге - Имени пользователя (с прямой ссылкой на профиль) - Суммы затрат на обработку новостей в долларах США

Фильтрация по периодам



Гибкие возможности выбора временного периода для анализа: - **Все время** - полная статистика без ограничения по датам - **Сегодня** - анализ только за текущий день - **Месяц** - статистика за последние 30 дней - **Свой период** - возможность задать произвольный диапазон дат для анализа

Функции поиска и отображения



- **Поиск по пользователю** - быстрый поиск конкретного пользователя в списке - **Настройка количества записей** - возможность отображать от десятков до сотен записей на одной странице - **Пагинация** - удобная навигация по страницам при просмотре большого количества данных

Как использовать для маркетинговых задач

Анализ эффективности бизнес-модели

1. **Оценка рентабельности пользователей**:
   - Сопоставляйте затраты на обработку новостей с доходом от каждого пользователя
   - Выявляйте сегменты пользователей с наиболее благоприятным соотношением дохода к затратам
   - Определяйте “границу окупаемости” для различных типов подписок
2. **Оптимизация расходов на сокращение новостей**:
   - Отслеживайте общий объем затрат на обработку новостей за различные периоды
   - Выявляйте аномалии и всплески потребления для своевременной реакции
   - Разрабатывайте стратегии оптимизации затрат без ущерба для качества сервиса

Сегментация пользователей по потреблению контента

1. **Выделение групп по интенсивности потребления новостей**:
   - Создавайте сегменты пользователей на основе объема потребляемых новостей
   - Анализируйте характеристики “тяжелых потребителей” новостного контента
   - Разрабатывайте таргетированные предложения для разных сегментов
2. **Корреляция потребления новостей и удержания**:
   - Исследуйте связь между объемом потребления новостей и продолжительностью использования сервиса
   - Выявляйте оптимальный уровень вовлеченности для максимального удержания
   - Создавайте стратегии, направленные на достижение этого оптимального уровня

Оптимизация ценообразования

1. **Разработка справедливой модели ценообразования**:
   - Используйте данные о затратах для формирования более точной структуры тарифов
   - Оценивайте возможность введения лимитов на объем обрабатываемых новостей в различных тарифах
   - Рассчитывайте оптимальную стоимость подписок с учетом фактических затрат
2. **Планирование специальных предложений**:
   - Определяйте периоды с наибольшим и наименьшим потреблением новостей
   - Разрабатывайте специальные акции и скидки с учетом этих данных
   - Оценивайте потенциальное влияние промо-акций на затраты по обработке новостей

Практические кейсы использования

1. **Анализ сезонности потребления новостей**:
   - Изучайте, как меняется потребление новостного контента в разные периоды
   - Планируйте маркетинговые активности с учетом этих сезонных паттернов
   - Корректируйте прогнозы затрат на основе выявленных сезонных трендов
2. **Создание устойчивой бизнес-модели**:
   - Используйте данные о затратах для прогнозирования долгосрочной рентабельности
   - Разрабатывайте планы масштабирования с учетом ожидаемого роста затрат
   - Балансируйте между качеством сервиса и экономической эффективностью
3. **Оценка влияния изменений в продукте**:
   - Сравнивайте затраты до и после внедрения новых функций сокращения новостей
   - Измеряйте эффективность различных алгоритмов обработки с точки зрения затрат
   - Принимайте решения о развитии функциональности на основе этих данных
4. **Выявление пользователей с аномальным потреблением**:
   - Идентифицируйте пользователей с чрезмерно высокими затратами на обработку новостей
   - Анализируйте причины такого потребления и разрабатывайте соответствующие меры
   - При необходимости внедряйте механизмы предотвращения злоупотреблений

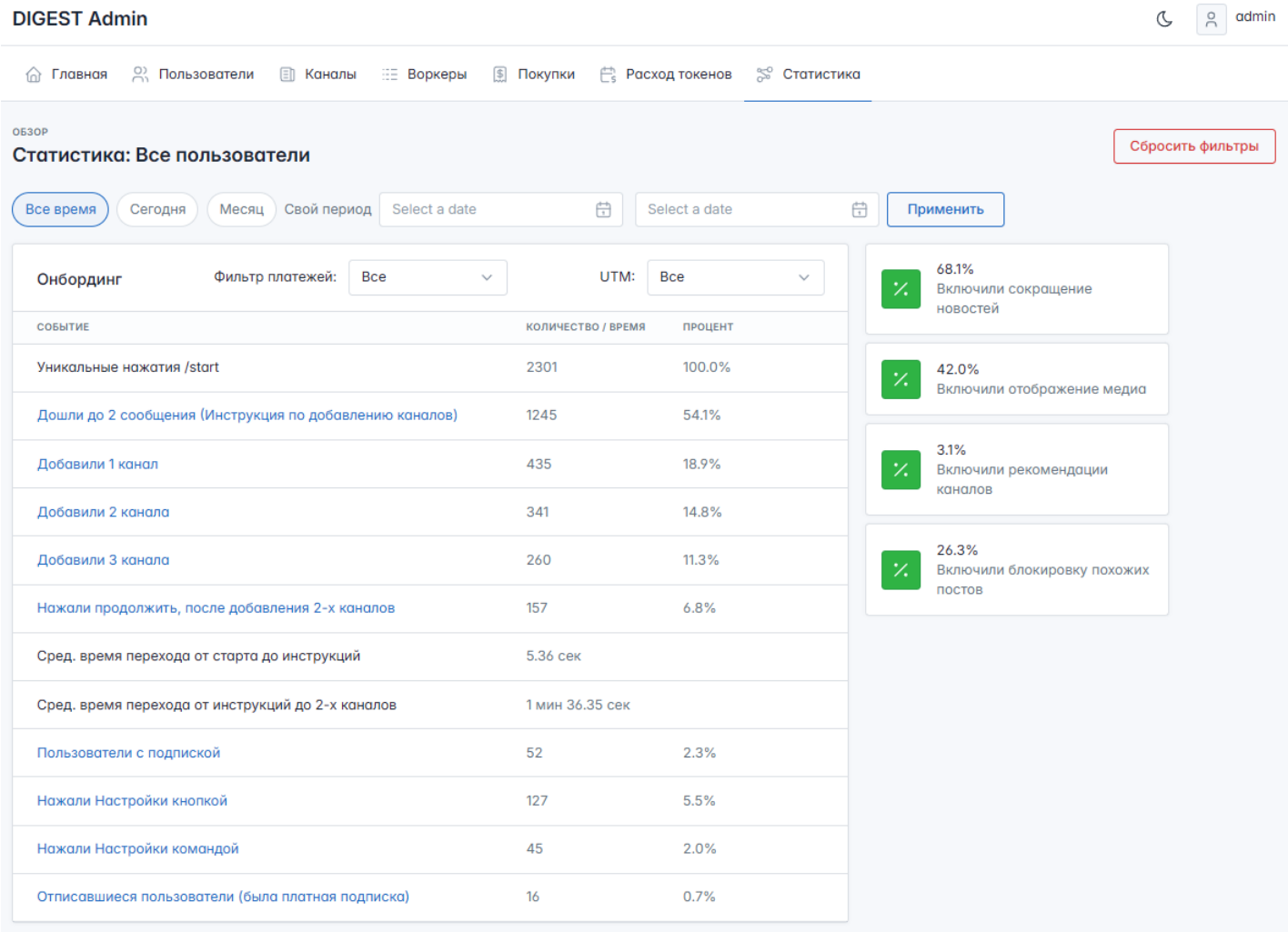
Советы по эффективному использованию

- Регулярно анализируйте соотношение затрат на сокращение новостей к доходу от пользователей
- Используйте фильтрацию по периодам для выявления трендов в потреблении новостей
- Сравнивайте профили пользователей с высокими и низкими затратами для понимания различий в их поведении
- Отслеживайте, как изменения в UI или функциональности сервиса влияют на объемы потребления новостей

- Изучайте корреляцию между затратами на новости и различными показателями удержания пользователей
- Анализируйте эффективность различных источников привлечения с точки зрения последующих затрат на обработку новостей

Страница “Расход токенов” является важнейшим инструментом для анализа экономической эффективности сервиса по сокращению новостей. Правильное использование этих данных позволяет построить устойчивую бизнес-модель, оптимизировать структуру тарифов и максимизировать долгосрочную рентабельность, обеспечивая при этом высокое качество сервиса для всех категорий пользователей.

Страница Статистика

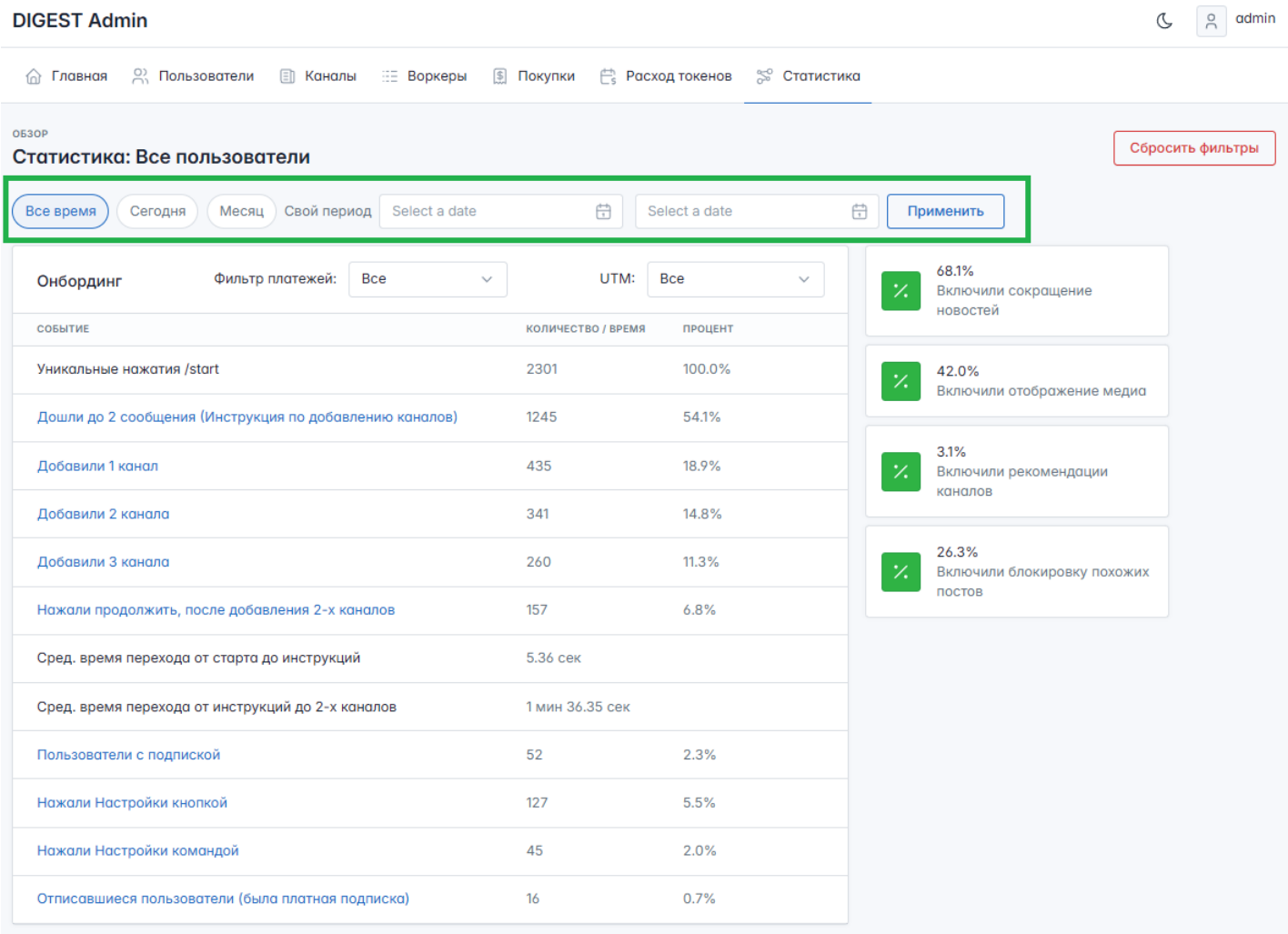


Обзор

Страница “Статистика” представляет собой комплексный аналитический инструмент, позволяющий отслеживать ключевые показатели онбординга, конверсии и удержания пользователей. Эта страница является одной из центральных для анализа эффективности маркетинговых кампаний и принятия стратегических решений на основе данных.

Основные возможности

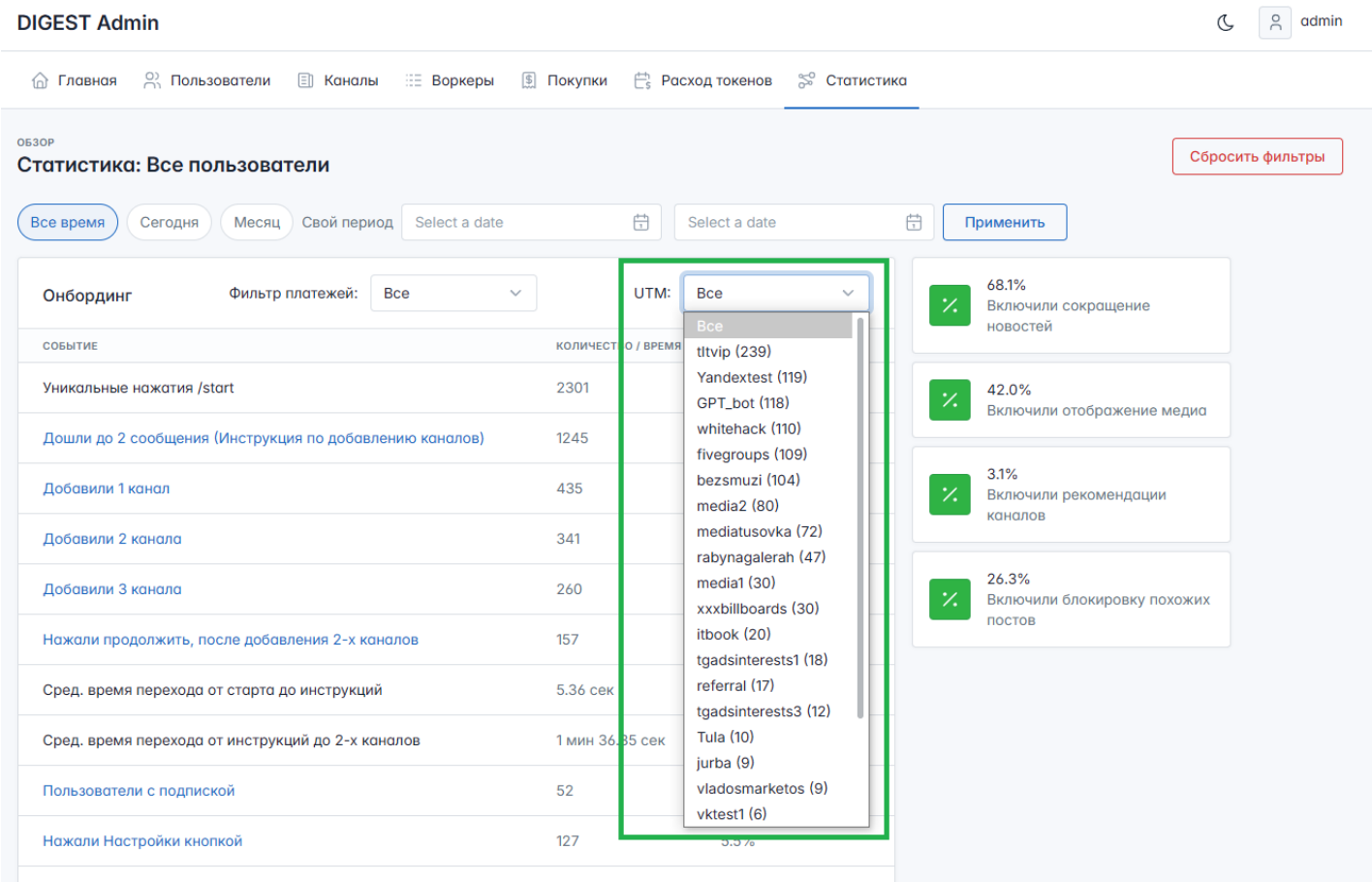
Фильтрация по периодам



Страница позволяет анализировать данные за различные временные интервалы: - **Все время** - полная статистика без ограничения по датам - **Сегодня** - анализ только за текущий день - **Месяц** - статистика за последние 30 дней - **Свой период** - возможность задать произвольный диапазон дат для анализа

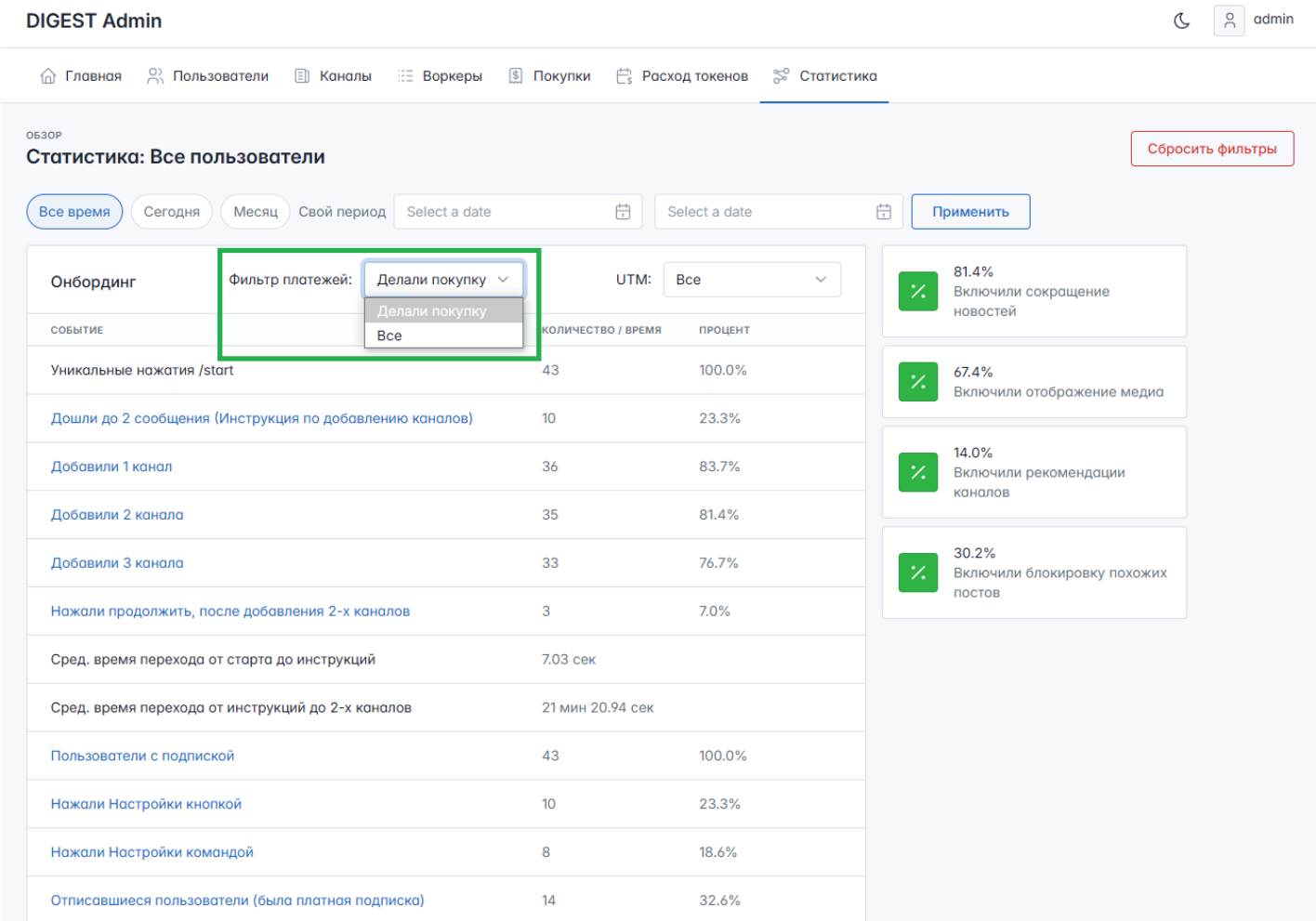
Сегментация данных

По источникам привлечения (UTM)



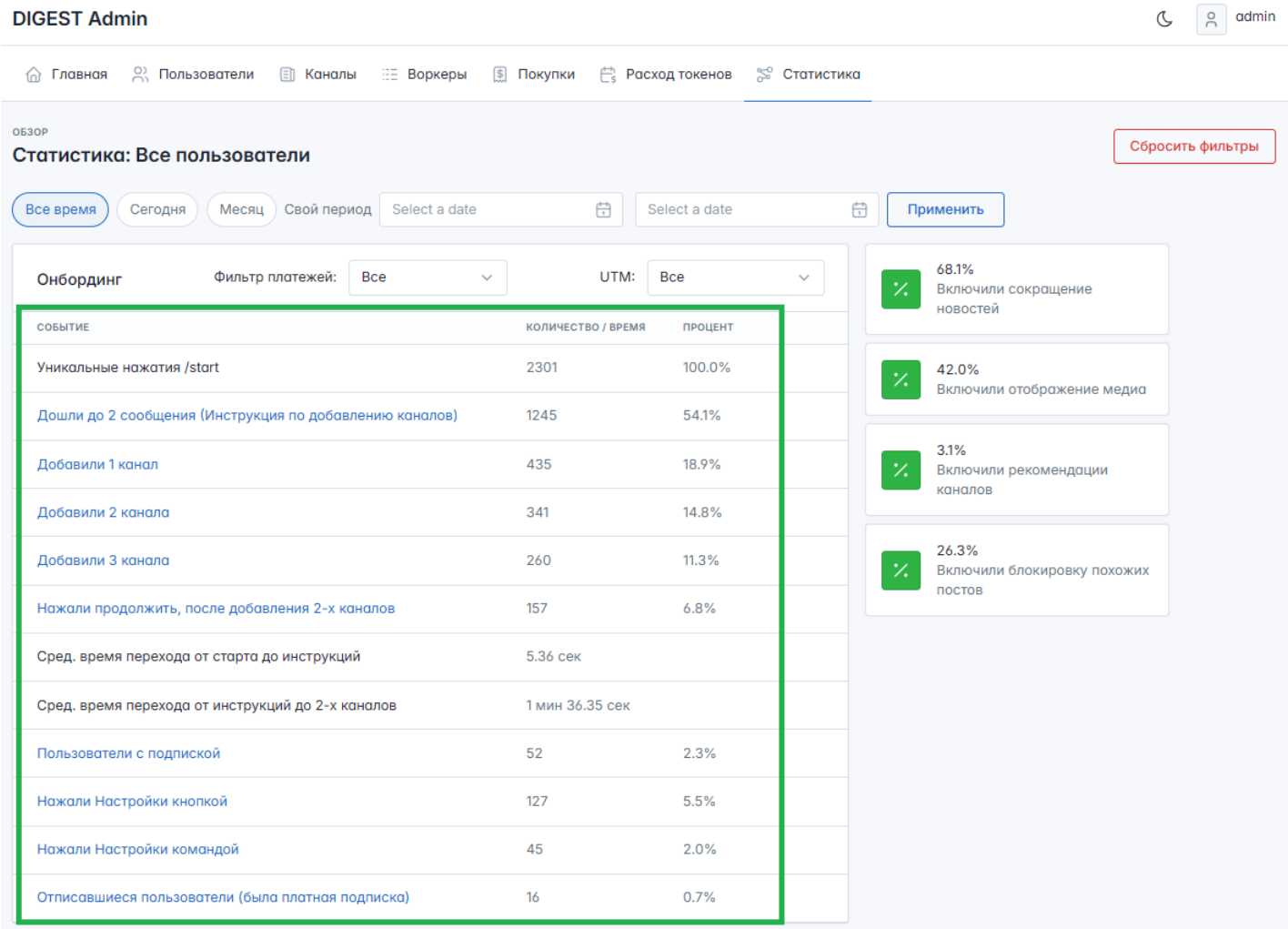
- Выпадающий список с возможностью выбора конкретного UTM-источника - Для каждого источника показывается количество привлеченных пользователей - Возможность сравнения показателей разных рекламных кампаний

По платежному поведению



- **Все пользователи** - статистика по всей базе пользователей - **Делали покупку** - фильтр для анализа пользователей, которые делали покупки

Аналитика онбординга

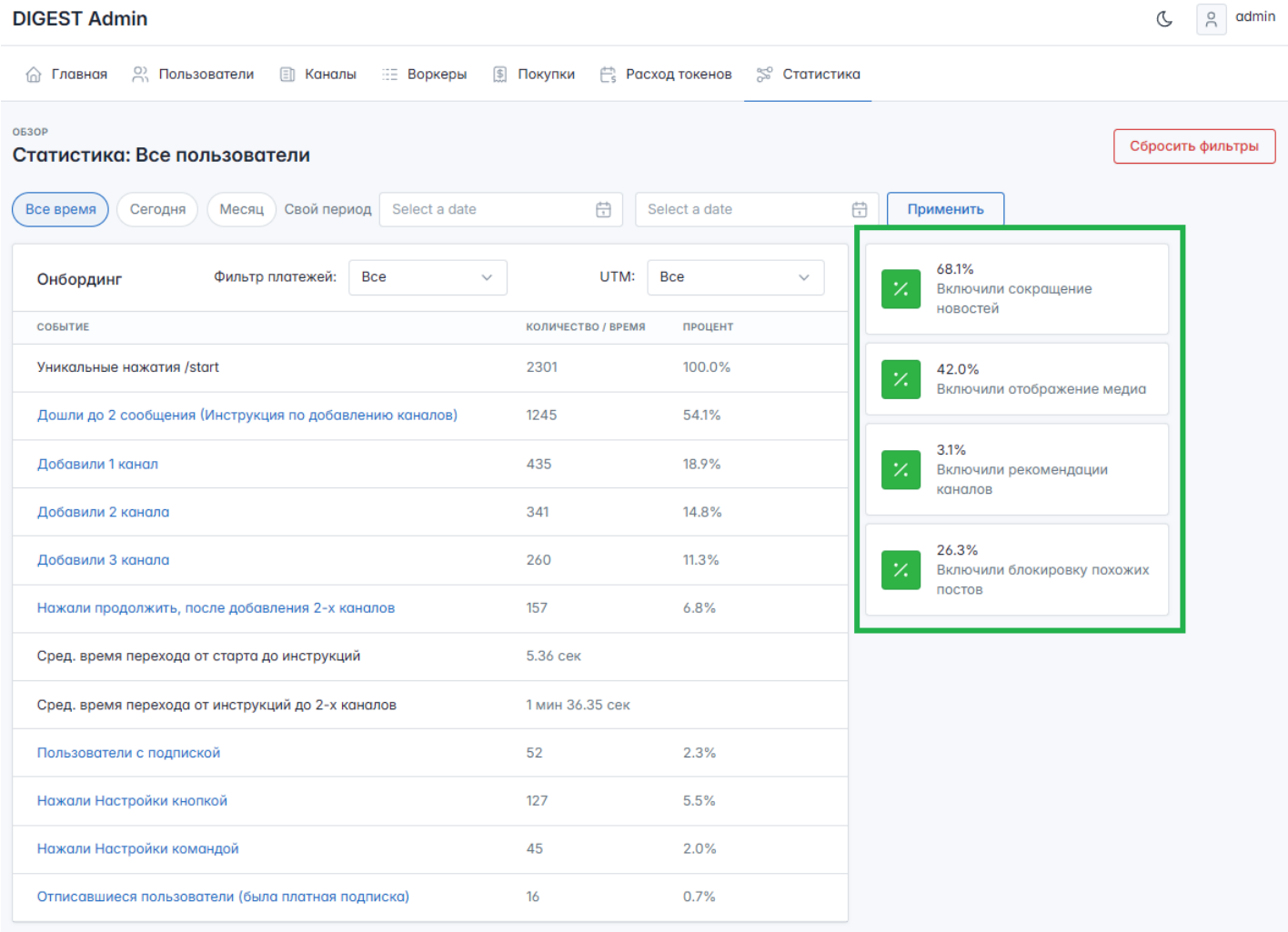


Центральная таблица страницы содержит детальную информацию о прохождении пользователями каждого этапа воронки:

- **Количество пользователей** на каждом этапе
- **Процент конверсии** от общего числа пользователей
- Динамика изменения показателей (визуализация)
- Интерактивные ссылки для перехода к спискам пользователей на конкретных этапах

Ключевые этапы онбординга, представленные в таблице: 1. **Дошли до 2 сообщения** (инструкция по добавлению каналов) 2. **Добавили 1 канал** 3. **Добавили 2 канала** 4. **Добавили 3 канала** 5. **Нажали продолжить** после добавления каналов 6. **Взаимодействия с настройками** (через кнопку или команду) 7. **Пользователи с подпиской** 8. **Отписавшиеся пользователи** (была платная подписка)

Ключевые метрики пользователей



Дополнительный блок справа отображает важные процентные показатели: - Включили сокращение новостей - Включили отображение медиа - Включили рекомендации каналов - Включили блокировку похожих постов

Как использовать для маркетинговых задач

Анализ эффективности воронки конверсии

1. **Выявление проблемных мест в онбординге**:
   - Определите этапы с наибольшим оттоком пользователей
   - Сравните процентные показатели между этапами
   - Проанализируйте, на каком шаге теряется наибольшее количество потенциальных клиентов
2. **Оптимизация шагов воронки**:
   - Используйте данные о медианном времени между этапами для оптимизации
   - Фокусируйтесь на улучшении этапов с самой низкой конверсией
   - Отслеживайте изменения конверсии после внесения улучшений

Оценка эффективности рекламных кампаний

1. **Сравнение источников привлечения**:
   - Используйте фильтр по UTM для анализа разных каналов привлечения
   - Сравните не только объемы привлеченных пользователей, но и их конверсию на каждом этапе
   - Определите, какие источники дают наиболее качественных пользователей
2. **Расчет ROI рекламных кампаний**:
   - Анализируйте процент пользователей, доходящих до платной подписки, по каждому источнику

Временной анализ

1. **Отслеживание динамики показателей**:
   - Используйте фильтры по периодам для сравнения эффективности в разные временные интервалы
   - Анализируйте сезонные тренды и их влияние на конверсию
   - Оценивайте эффективность внесенных изменений, сравнивая периоды до и после
2. **Прогнозирование**:
   - На основе исторических данных стройте прогнозы конверсии
   - Планируйте маркетинговые активности с учетом выявленных тенденций

Практические кейсы использования

1. **Анализ новой рекламной кампании**:
   - Установите фильтр на период проведения кампании
   - Выберите соответствующий UTM-источник
   - Проанализируйте конверсию на каждом этапе онбординга
   - Сравните с показателями других кампаний

2. **Оптимизация онбординга**:
   - Определите этап с наименьшей конверсией
   - Перейдите по ссылке на список пользователей, остановившихся на этом этапе
   - Проведите дополнительный анализ или опрос для выявления причин отказа
   - Внесите изменения и отслеживайте эффект
3. **Анализ платящей аудитории**:
   - Установите фильтр “Делали покупку”
   - Изучите особенности онбординга платящих пользователей
   - Выявите закономерности и используйте их для улучшения конверсии
4. **Оценка качества трафика**:
   - Сравните процент пользователей, доходящих до добавления каналов, из разных источников
   - Определите источники с наиболее вовлеченной аудиторией
   - Скорректируйте таргетинг рекламы с учетом полученных данных

Советы по эффективному использованию

- Регулярно (например, еженедельно) анализируйте динамику ключевых показателей
- Всегда сравнивайте данные за сопоставимые периоды при оценке эффективности изменений
- Используйте комбинацию фильтров для глубокого анализа конкретных сегментов
- Сопоставляйте данные статистики с изменениями в продукте и маркетинговых активностях

Страница “Статистика” — это мощный инструмент для принятия обоснованных маркетинговых решений. Регулярный анализ представленных на ней данных позволяет оптимизировать маркетинговый бюджет, улучшать пользовательский опыт и максимизировать конверсию на каждом этапе воронки.