07. Кастомные таблицы

Хранение прикладных данных и работа с таблицами внутри платформы.

- Кастомные таблицы
- Связи между таблицами
- Инструкция по обновлению таблицы department

Кастомные таблицы

«После того, как бизнес-данные были централизованы и интегрированы, ценность базы данных больше, чем сумма ранее существовавших частей».
— Ларри Эллисон, основатель Oracle

Кастомные таблицы — удобное и практичное хранилище данных, которое вы можете использовать как составную часть решения, создаваемого на Metabot, так и как отдельный сервис, доступный по API из других приложений и платформ.

Кастомные таблицы от Metabot — это альтернатива (или дополнение) к таким продуктам как Google Sheets, AirTable и Notion, созданное специально для разработки приложений с бизнес-логикой.

Примеры использования:

- Таблица пользователей для регистрации/авторизации и личного кабинета в мессенджере;
- Таблицы заявок или заказов, заполняемая ботом, как замена формы;
- Таблицы, представляющие операционную модель вашего бизнеса (ERP);
- Таблицы для учёта выдачи материалов на мероприятии.

Используя кастомные таблицы в сочетании с диалоговыми сценариями, JavaScript и API конструктором вы сможете создать фронтенд и бэкэнд, которая заменят вам мобильное приложение и серверный бэкэнд.

Таблицы незаменимы при реализации глубоких интеграций мессенджеров в процессы предприятия, диалогового интерфейса к системам предприятия и диалогового подключенного опыта (CCX) — таблицы используются для хранения данных для синхронизации систем.

Обращаться к таблицам вы можете прямо в диалоговых сценариях вашего бота, через методы JS: вы сможете выполнять операции работы с таблицей, такие как, поиска записей, их редактирования, удаления.

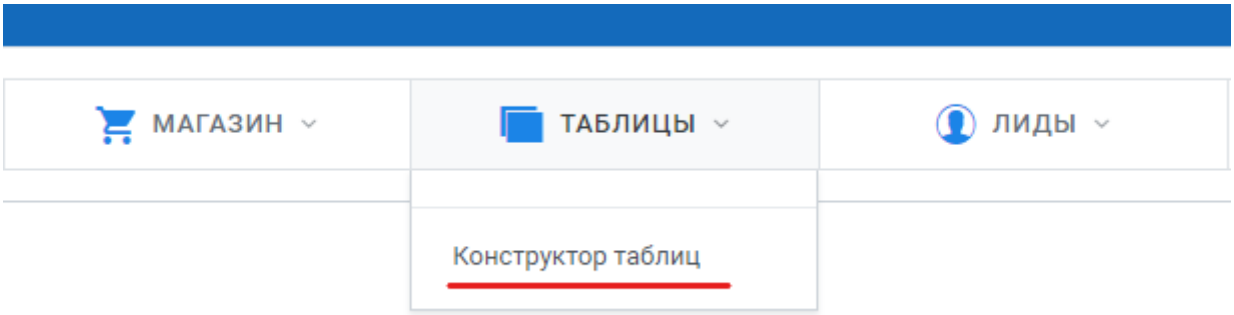
Если вам необходимо обратиться к таблицам Metabot из внешних систем, для этого вы можете создать внутренние API эндпоинты.

С помощью плагина вы сможете расширить методы работы с таблицами, например, сделав прямые обращение к базе данных и вызов SQL.

Работа с SQL доступна на выделенном сервере и в on-premise коробочной версии.

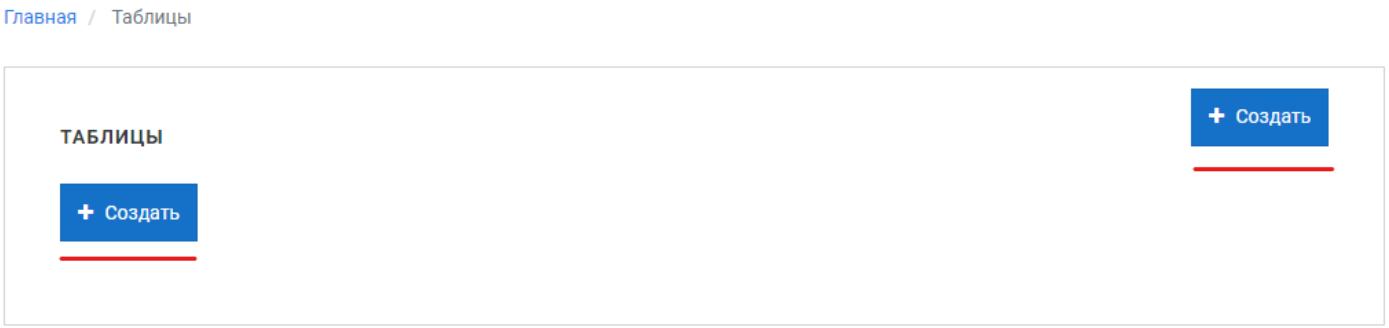
Конструктор таблиц

Конструктор таблиц находится в верхнем меню бота в разделе **Таблицы**:

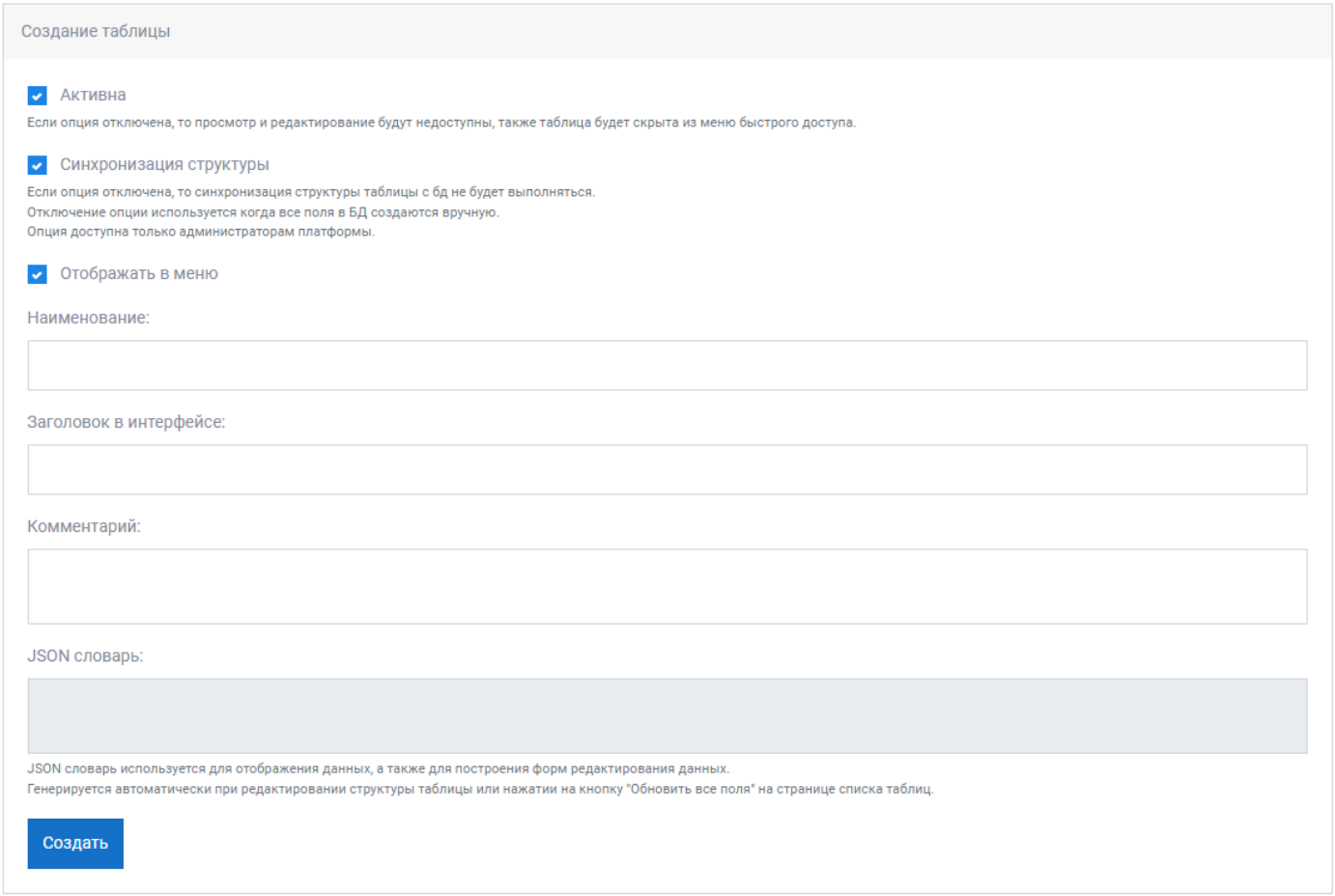


Он предназначен для работы с таблицами, например, для создания таблиц, их форматирования и удаления.

Для создания таблицы необходимо нажать кнопку **Создать**:



В открывшемся окне **Создание таблицы** можно заполнить информацию о создаваемой таблице:



- **Активна** — если опция отключена, то просмотр и редактирование будут недоступны, также таблица будет скрыта из меню быстрого доступа;
- **Синхронизация структуры** — если опция отключена, то синхронизация структуры таблицы с базой данных (БД) не будет выполняться. Отключение опции используется когда все поля в БД создаются вручную;

> Опция **Синхронизация структуры** доступна только администраторам платформы.

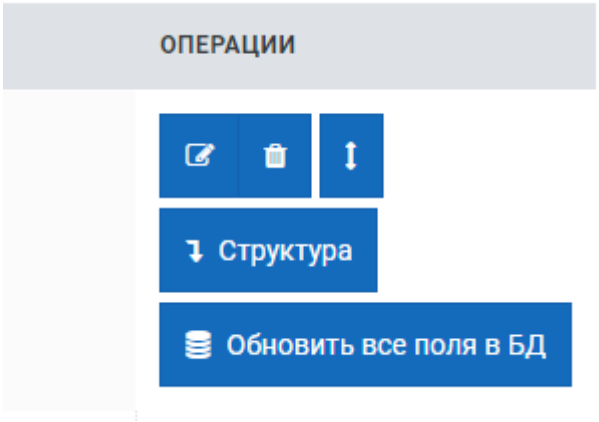
- **Отображать в меню** — если данная опция включена, то таблица будет отображаться в выпадающем списке раздела **Таблицы** верхнего меню бота;
- **Наименование** — название таблицы, которое будет использоваться при обращении к ней методами;

> В имени таблицы можно указывать только латиницу (только нижний регистр), цифры и подчеркивание.

- **Заголовок в интерфейсе** — название таблицы, которое будет отображаться в интерфейсе;
- **Комментарий** — пояснение к таблице;
- **JSON словарь** — JSON словарь используется для отображения данных, а также для построения форм редактирования данных. Генерируется автоматически при

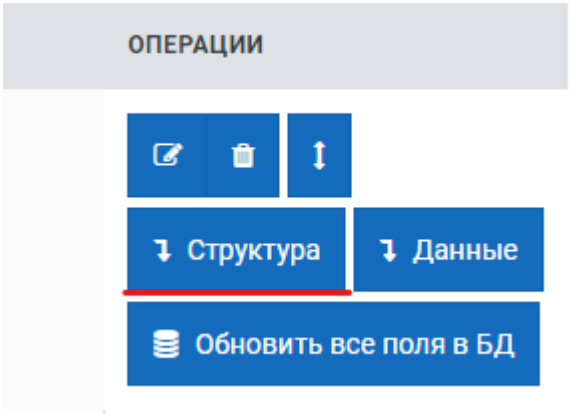
редактировании структуры таблицы или нажатии на кнопку "Обновить все поля" на странице списка таблиц.

После создания таблицы появляются функции ее редактирования, удаления, создания полей и их обновления в БД:



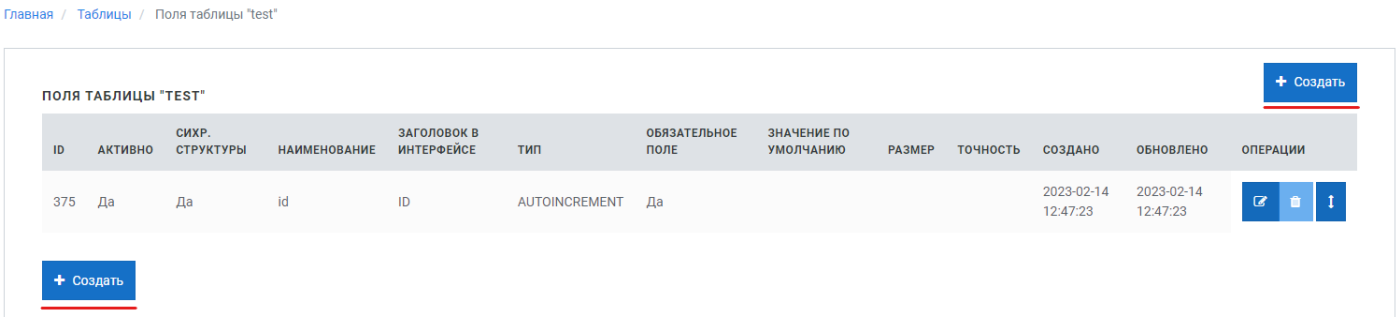
Конструктор полей

Чтобы заполнить таблицу данными, нужно для начала создать ей соответствующие поля. Это можно сделать при помощи операции **Структура:**

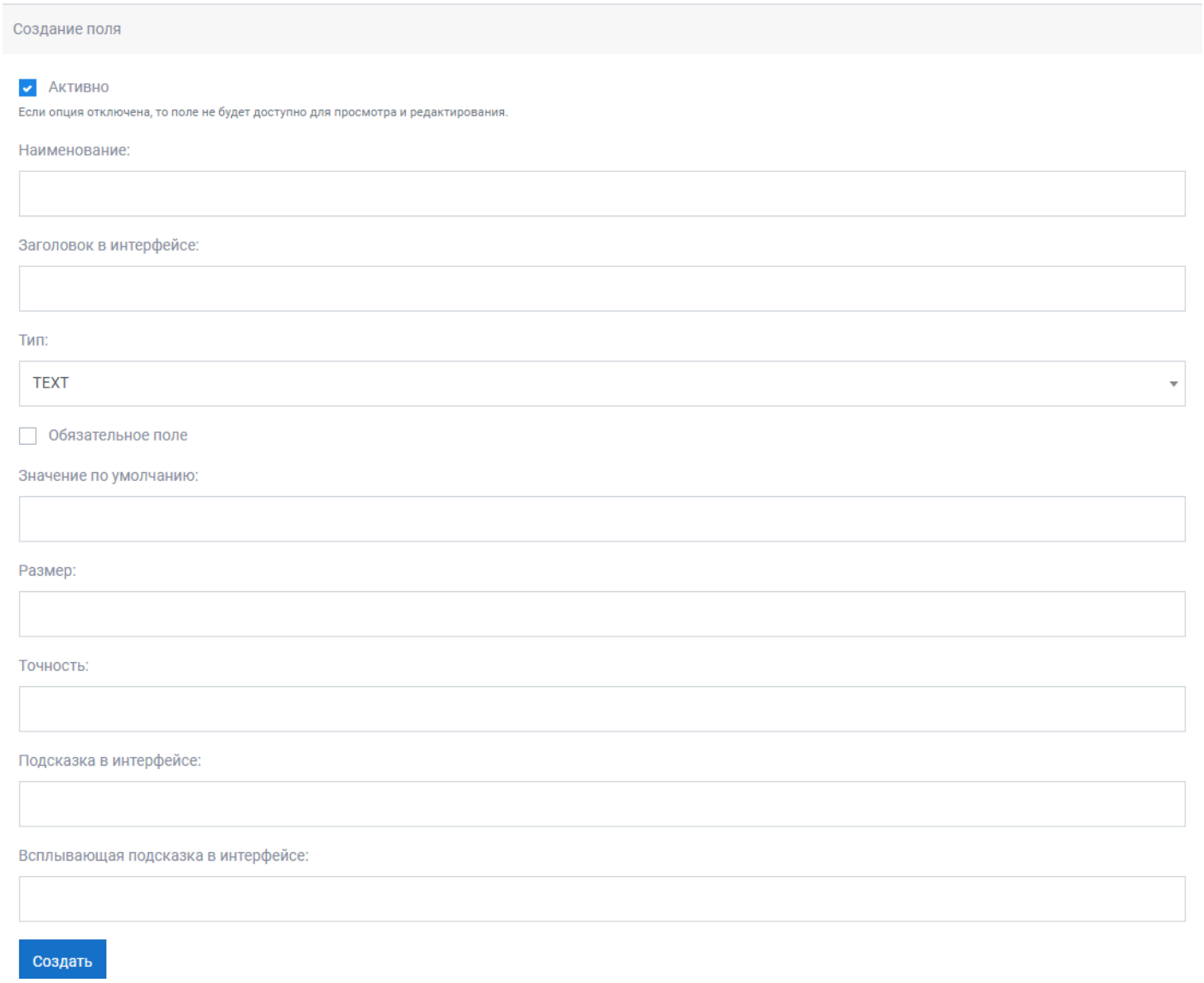


Первым полем любой таблицы всегда является ID строки. Это поле создается автоматически при создании таблицы и не подлежит удалению. По этому полю вы сможете идентифицировать отдельные строки таблицы, а также искать и редактировать записи через JS.

Остальные необходимые поля можно добавить щелкнув по кнопке **Создать:**



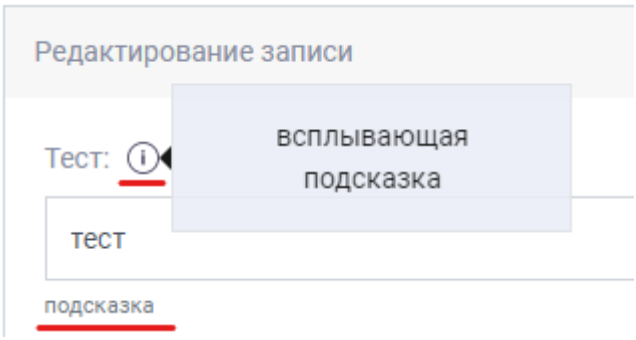
В открывшемся окне **Создание поля** необходимо заполнить информацию о создаваемом поле:



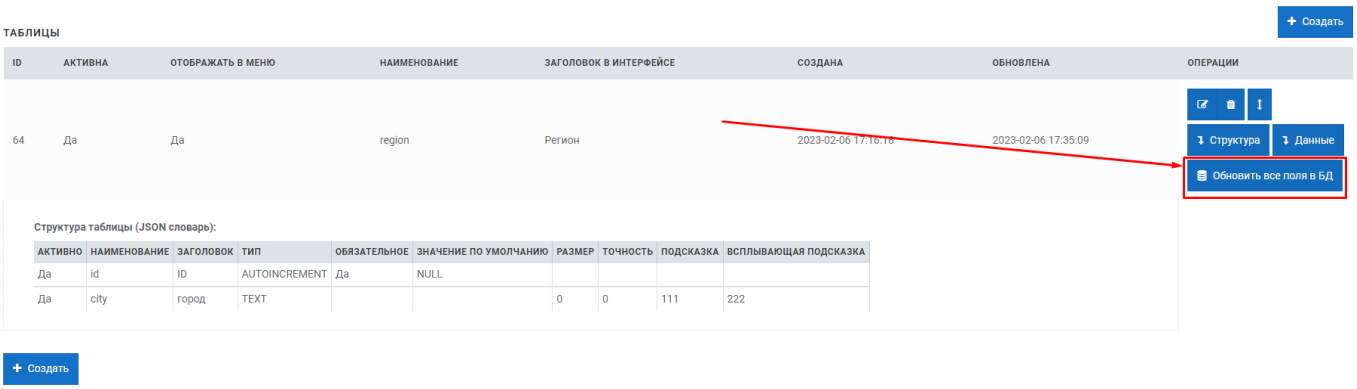
- **Активно** — если опция отключена, то поле не будет доступно для просмотра и редактирования;
- **Наименование** — название поля, которое будет использоваться при обращении к нему методами;

> В имени поля можно указывать только латиницу (только нижний регистр), цифры и подчеркивание.

- **Заголовок в интерфейсе** — название поля, которое будет отображаться в интерфейсе;
- **Тип** — тип данных поля;
  - **Text** — строковый тип данных;
  - **TextArea** — строковый тип данных с возможностью переноса строк;
  - **Int** — числа в диапазоне **от -2 147 483 648 до 2 147 483 647**;
  - **Big_Int** — числа в диапазоне **от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807**;
  - **Tiny_Int** — числа в диапазоне **от -32768 до 32767**;
  - **Decimal** — числа с плавающей точкой;
  - **DateTime** — дата в формате **дд.мм.гггг --:--:--** ;
  - **Date** — дата в формате **дд.мм.гггг** ;
  - **Vector** — динамический массив;
  - **Created_At** — дата и время создания (не редактируется);
  - **Updated_At** — дата и время обновления (не редактируется);
  - **Deleted_At** — дата и время удаления (не редактируется);
- **Обязательное поле** — если данная опция включена, то в поле не может быть задано значение NULL;
- **Значение по умолчанию** — предустановленное значение поля;
- **Размер** — максимальное количество символов в данном поле;
- **Точность** — количество символов в числе после точки;
- **Подсказка в интерфейсе** — подсказка отображаемая в интерфейса в режиме просмотра и редактирования данных;
- **Всплывающая подсказка в интерфейсе** — подсказка в интерфейсе, возникающая при наведении мышкой на знак вопроса рядом с полем.



После создания и настройки всех ячеек таблицы можно провести обновление таблицы в базе данных платформы. Для это необходимо в конструкторе таблиц нажать **Обновить все поля в БД**:



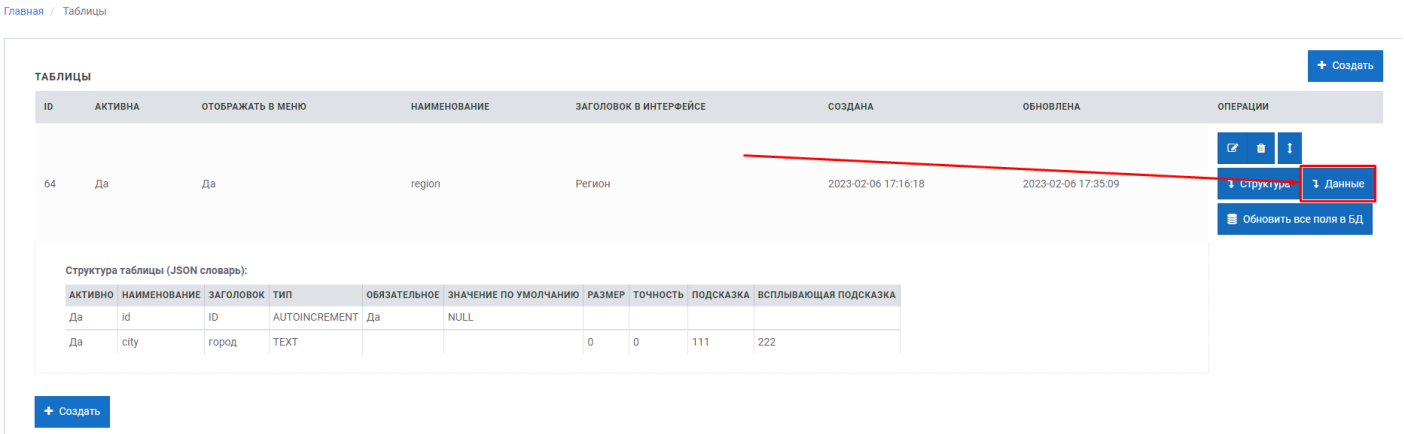
Это синхронизирует и обновит таблицу в базе данных, а так же сгенерирует ее **JSON словарь** (Структуру таблицы).

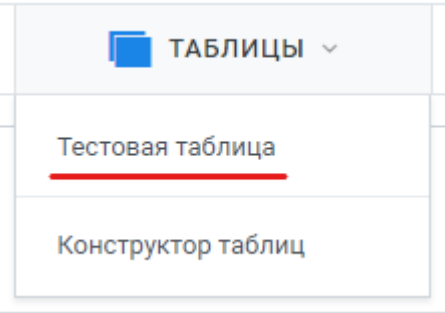
> На текущий момент, в качестве базы данных платформа Metabot поддерживает PostgreSQL.

> Формирование модели данных через текстовое описание мета-словаря на базе JSON в текущей версии не доступно.

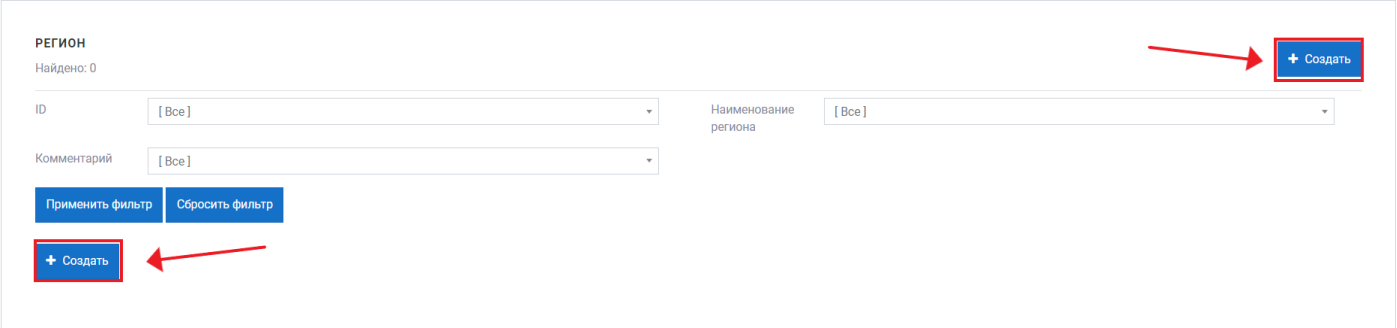
Просмотр и редактирование данных

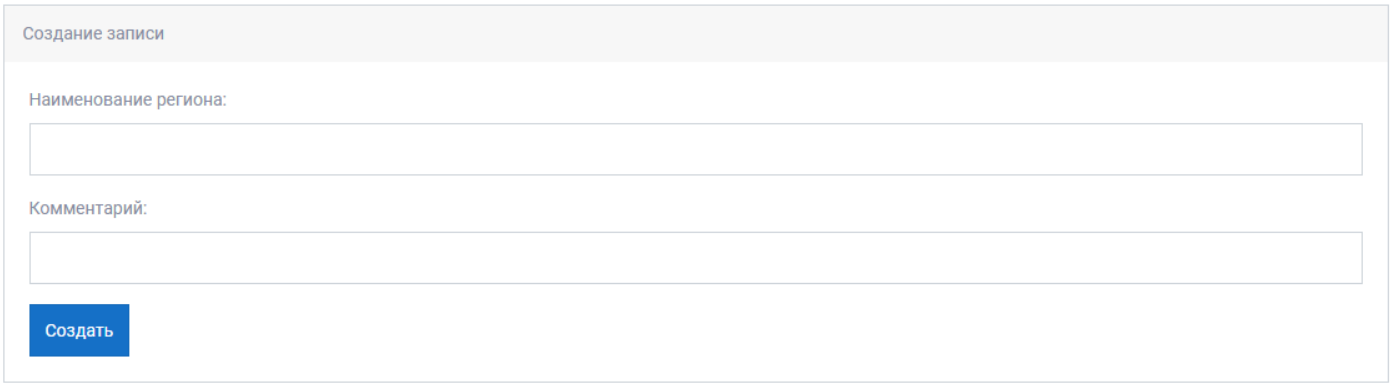
После создания структуры таблицы можно вручную заполнить ее данными. Для этого необходимо зайти в **Данные** в конструкторе таблиц либо, если вы включали опцию **Отображать в меню**, то нажать на название таблицы в выпадающем списке раздела **Таблицы** верхнего меню бота.





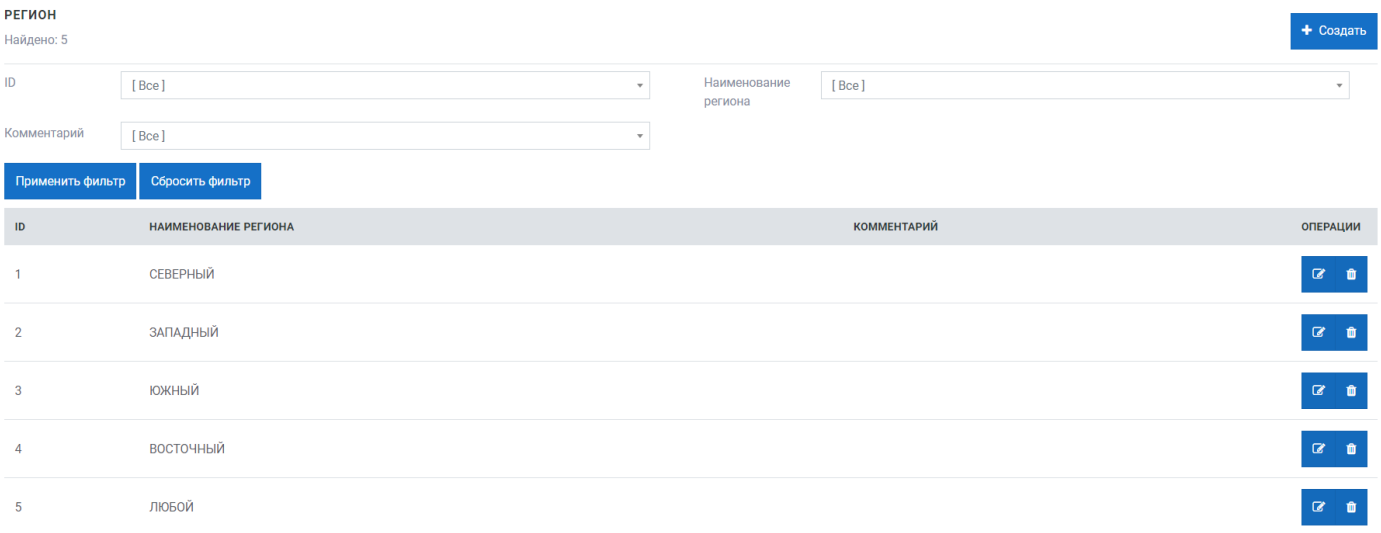
После этого нажмите на кнопку **Создать** и добавьте необходимые данные.





После создания запись появится в таблице. Записи так же можно редактировать и удалять.

По итогу у вас получится таблица, которая может выглядеть, например, так:

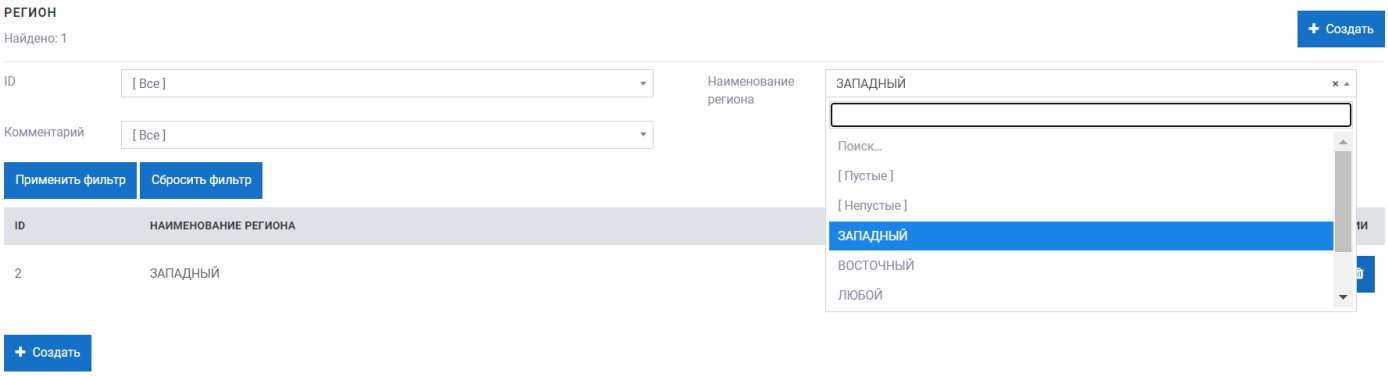


> Данная таблица**"Регионы"** будет использоваться для примеров использования методов в **Справочнике функций JS**.

Фильтры

Для ID и всех кастомных полей таблиц предусмотрены фильтры. В зависимости от созданных полей таблицы, будут автоматически созданы фильтры в виде выпадающих списков.

При задании фильтров и нажатии на кнопку **Применить фильтр** будут выведены соответствующие записи в таблице:



> С кастомными таблицами так же можно работать с помощью JS методов. Все методы и их описание вы можете посмотреть в **Справочнике по функциям JS**

Связи между таблицами

Представьте ситуацию: У вас есть интернет-магазин, и вам необходимо получить детальную информацию о заказе. Помимо базовых данных (таких как сумма и дата), вам нужны сведения о клиенте, товарах и их характеристиках. Раньше для этого требовалось писать сложные SQL-запросы или реализовывать дополнительную логику в коде.

Теперь в Metabot достаточно один раз настроить связи между сущностями через интерфейс. После этого вы можете обращаться к связанным данным в коде так же просто, как к обычным полям.

Преимущества такого подхода:

- Упрощение работы с данными за счет автоматизации связей.
- Минимизация ручного кодирования и SQL-запросов.
- Быстрый доступ к связанным данным через стандартные поля.

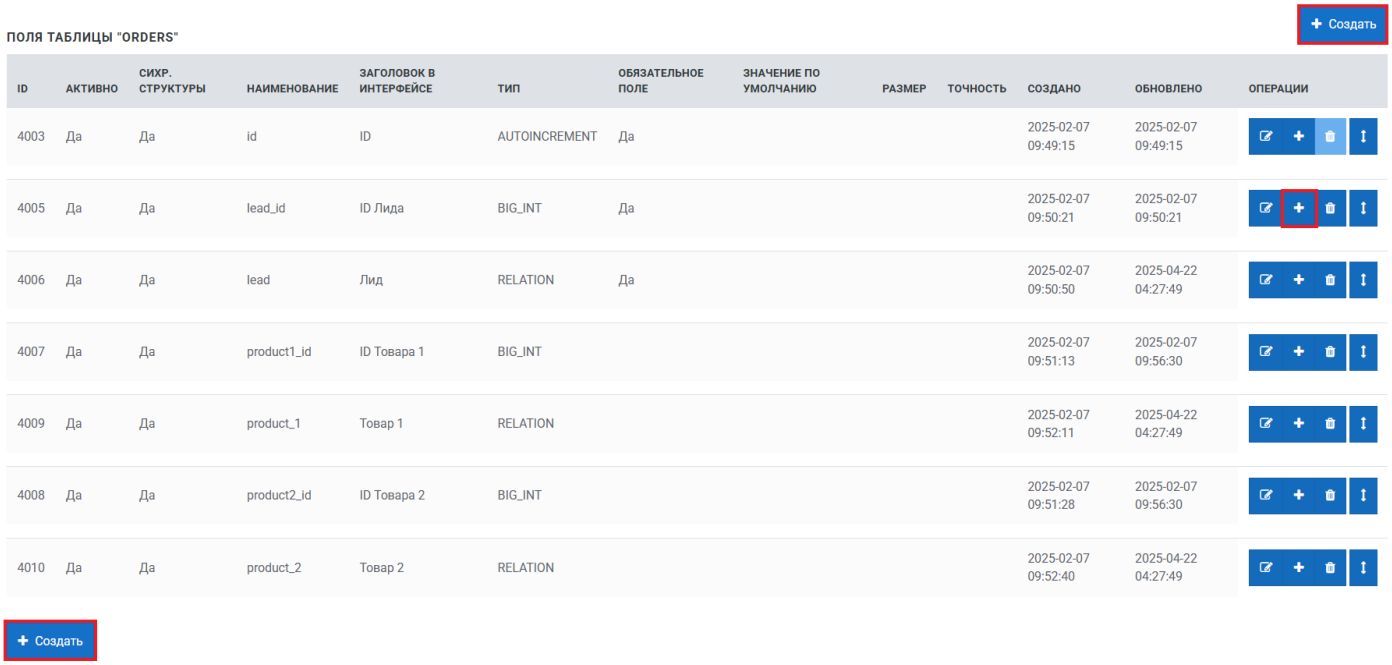
Создание связи

Связи между таблицами создаются при помощи JOIN полей.

JOIN поля создаются в меню **Структура** таблицы.



Существует два способа создания JOIN поля:



- Нажмите на любую из кнопок **Создать**. В открывшемся окне выберите тип поля **RELATION** и наименование связываемого поля.
- Нажмите на кнопу **+** напротив связываемого поля. Тип поля и наименование связываемого поля заполнятся автоматически.



Помимо уже названных полей в окне создания можно заполнить следующие поля:

Создание поля

Активно

Если опция отключена, то поле не будет доступно для просмотра и редактирования.

Наименование:

Заголовок в интерфейсе:

Тип:

RELATION

Обязательное поле

Связываемая таблица:

[Не выбрано]

Алиас связываемой таблицы:

Поле текущей таблицы:

lead_id

Поле связываемой таблицы:

Использовать данную связь при редактировании поля

Отображаемое поле 1 связываемой таблицы:

Заголовок отображаемого поля 1:

Отображаемое поле 2 связываемой таблицы:

Заголовок отображаемого поля 2:

Отображаемое поле 3 связываемой таблицы:

Заголовок отображаемого поля 3:

Создавать внешний ключ для связи

Действие при обновлении:

[Не выбрано]

Действие при удалении:

[Не выбрано]



- **Активно** — если опция отключена, то поле не будет доступно для просмотра и редактирования;
- **Наименование** — название поля, которое будет использоваться при обращении к нему методами;

> В имени поля можно указывать только латиницу (только нижний регистр), цифры и подчеркивание.

- **Заголовок в интерфейсе** — название поля, которое будет отображаться в интерфейсе;
- **Тип** — тип данных поля **RELATION**;
- **Обязательное поле** — если данная опция включена, то в поле не может быть задано значение NULL;
- **Связываемая таблица** — выпадающий список с кастомными таблицами бизнеса и доступными системными таблицами. Выбор таблицы из списка автоматически заполняет все остальные поля;
- **Алиас связываемой таблицы** — алиас необходим, так как в разных связываемых таблицах фигурируют одинаковые имена полей. Через него можно будет обращаться к полям связанной таблицы (алиас.поле);
- **Поле текущей таблицы** — имя связываемого поля текущей таблицы;
- **Поле связываемой таблицы** — поле по которому будет искаться соответствие с полем текущей таблицы;
- **Использовать данную связь при редактировании поля** — если опция включена, то при редактировании значения физического поля таблицы появляется выпадающий список со значениями из связанной таблицы;

> НЕ включайте опцию **Использовать данную связь при редактировании поля** для нескольких связей одного и того же поля текущей таблицы. НЕ включайте данную опцию для JOIN полей второго уровня, так как у такого поля нет физического поля, для его редактирования в текущей таблице.

- **Отображаемое поле 1-3 связываемой таблицы** — имя поля которое тянется из связанной таблицы;
- **Заголовок отображаемого поля 1-3** — заголовок поля которое тянется из связанной таблицы;
- **Создавать внешний ключ для связи** — Foreign Key для связи таблиц;

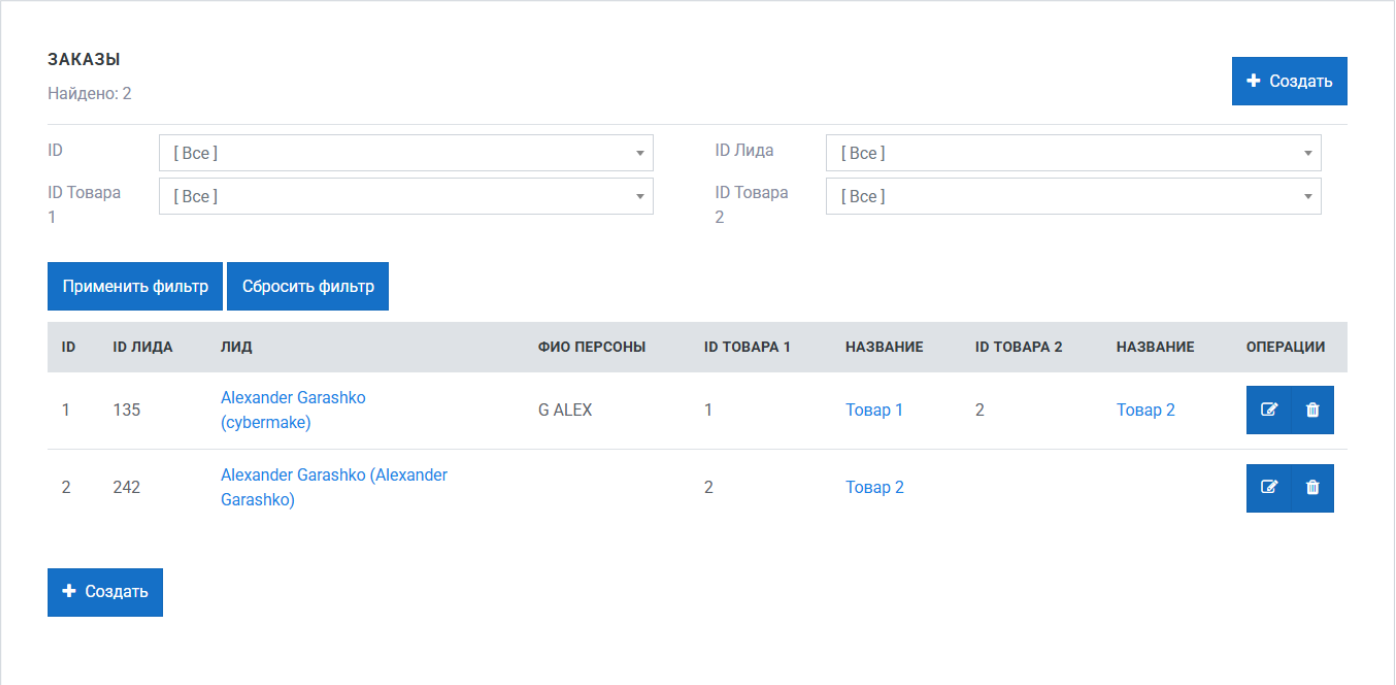
> НЕ включайте опцию **Создавать внешний ключ для связи** для нескольких связей одного и того же поля текущей таблицы. НЕ включайте данную опцию для JOIN полей второго уровня.

- **Действие при обновлении**:
  - **Каскад** — автоматически обновлять ссылающиеся друг на друга записи в связанных таблицах;
  - **Запрет** — невозможно обновить запись в связанной таблице, если есть записи ссылающиеся на этот товар в другой таблице;
  - **Установить в NULL** — установить значение NULL при обновлении связанной записи;
- **Действие при удалении**:
  - **Каскад** — автоматически удалять ссылающиеся друг на друга записи в связанных таблицах;
  - **Запрет** — невозможно удалить запись в связанной таблице, если есть записи ссылающиеся на этот товар в другой таблице;
  - **Установить в NULL** — установить значение NULL при удалении связанной записи;
- **Значение по умолчанию** — предустановленное значение поля;
- **Размер** — максимальное количество символов в данном поле;
- **Точность** — количество символов в числе после точки;
- **Подсказка в интерфейсе** — подсказка отображаемая в интерфейса в режиме просмотра и редактирования данных;
- **Всплывающая подсказка в интерфейсе** — подсказка в интерфейсе, возникающая при наведении мышкой на знак вопроса рядом с полем.

После заполнения всех обязательных полей (Наименование, Заголовок в интерфейсе, Связываемая таблица, Алиас, Поле связываемой таблицы) нажмите на кнопку **Сохранить**. Поле появится в таблице **Структура**.

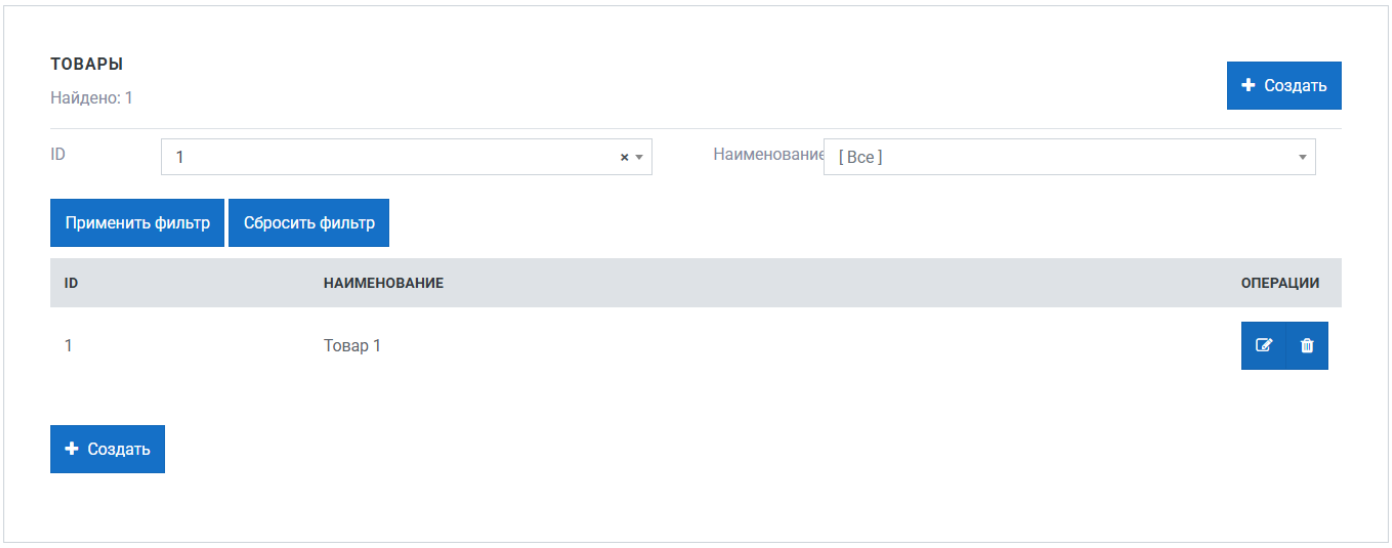
> Если вы хотите создать несколько связей с одной и той же таблицей, добавьте уникальное для таблицы число в алиасе. Например: leads, leads2...

Записи в таблицах имеют кликабельные ссылки на записи, на которые они ссылаются. Поля с ссылками выделены синим цветом.

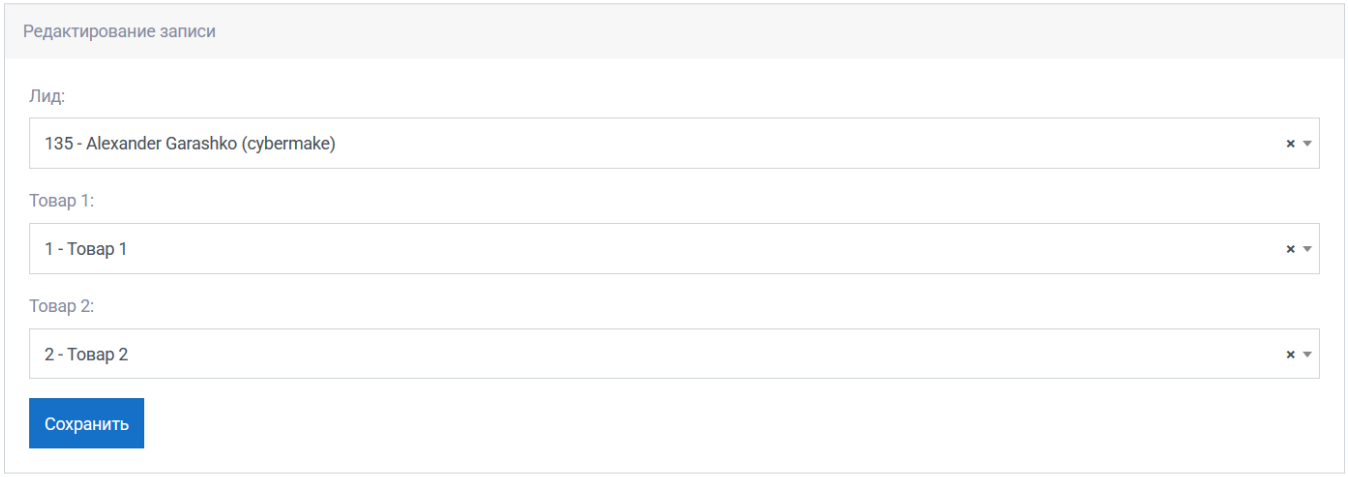


> При выключении активности физического поля, связанные с ним RELATION поля **продолжат отображаться** в данных таблицы.

Так на скриншоте выше поле **Лид** содержит ссылку на соответствующую запись в системной таблице лидов, а поле **Название** ссылку на соответствующую запись о товаре.



При редактировании записи в ссылающейся таблице связанные поля изменяются с помощью выпадающих списков.



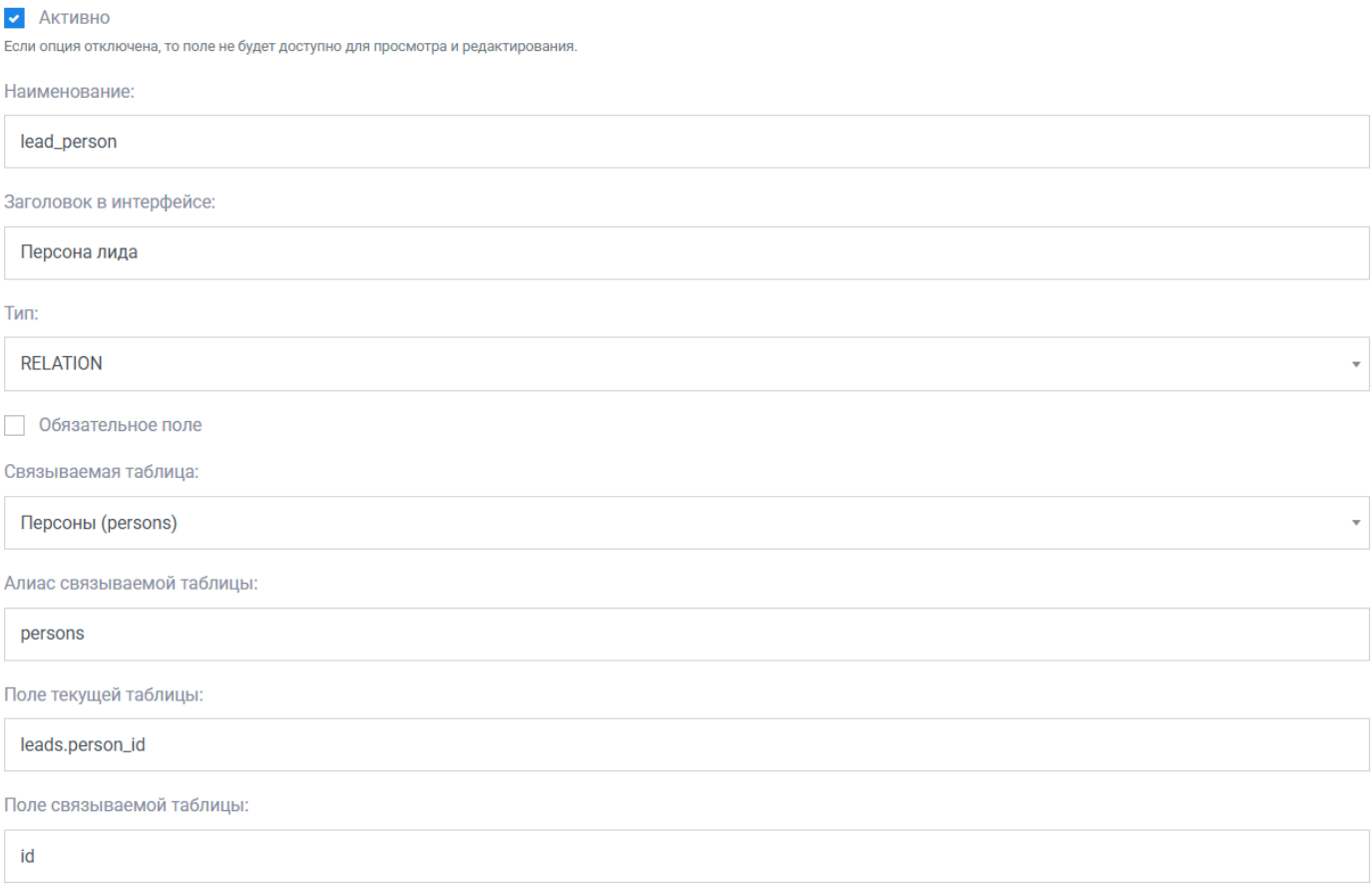
> Если что-то настроено некорректно, то при просмотре или редактировании записей таблицы может возникнуть ошибка 500. Попробуйте отключить лишние опции **Использовать данную связь при редактировании поля** и **Создать внешний ключ** , проверьте алиасы и имена полей. Если не получается исправить проблему, обратитесь в поддержку Metabot.

Уровни вложенности

Связанные поля имеют неограниченный уровень вложенности. Рассмотрим на примере:

- В таблице **orders** есть поле lead_id;
- В той же таблице создается RELATION поле **lead** ссылающееся на таблицу **leads** поле **id**, которое обращается к leads.name и **leads.person_id**;
- В той же таблице **orders** создается RELATION поле **person** ссылающееся на таблицу **person** поле **id** на основе подтянутого связанного поля **leads.person_id**.

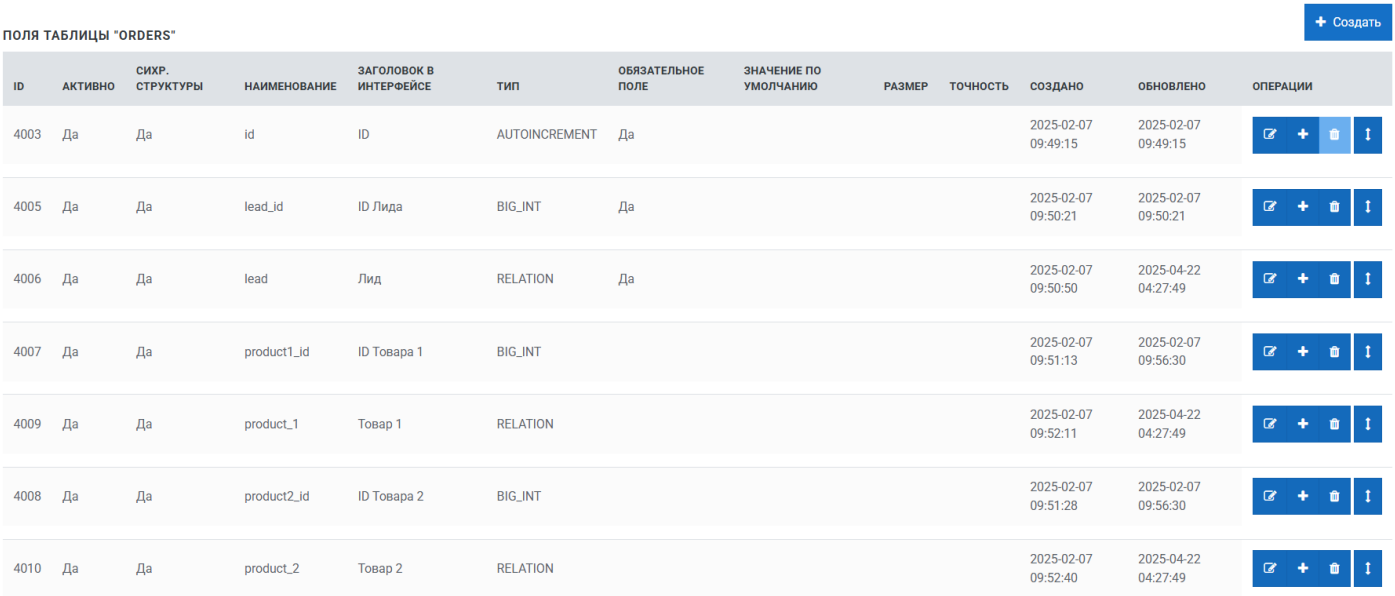
Так будут выглядеть настройки данного поля:



В **Поле текущей таблицы** записывается не физическое поле таблицы, а поле доставаемое из таблицы лидов. В таких многоуровневых JOIN полях обязательно должны быть выключены опции **Использовать данную связь при редактировании поля** и **Создать внешний ключ**.

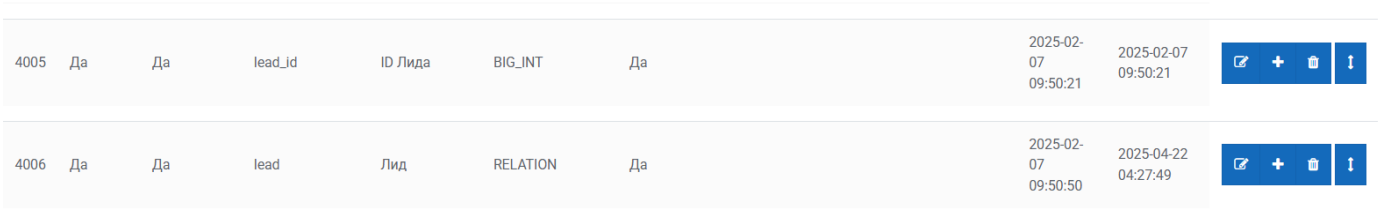
Пример

1. В вашем боте есть таблица заказов **orders**.



2. Вам требуется узнать id персоны лида через эту таблицу. Чтобы это сделать можно связать поле **lead_id** с системной таблицей лидов.

3. Для этого создается JOIN поле **lead** с типом **RELATION** описывающее связь между полем **lead_id** таблицы заказов и полем **id** в системной таблице лидов.



4. Теперь при обращении из JS методов вы можете обратиться к данным лида напрямую из таблицы заказов.

> Со связываемыми полями так же можно работать с помощью JS методов. Все методы и их описание вы можете посмотреть в **Справочнике по функциям JS**

Инструкция по обновлению таблицы department

При добавлении нового адреса, отделения или структурного подразделения должны обновляться некоторые скрипты и таблицы.

1. Обновление таблицы отделений

В данных таблицы **department** необходимо внести новые записи в случае изменений адресов, отделений или струк. подразделений.

Запись должна состоять из:

- **Отделения** - например, Врачебная амбулатория с.Наумовка.
- **Структурного подразделения** - например, Врачебные амбулатории.
- **Адреса** - например, Районная поликлиника.

> Адрес и структурное подразделение **не должно отличаться** по написанию от данных взятых в скрипте из кнопок. Следите за **лишними символами** (например, пробелы в конце строки) и правильностью написания.

1212 Да Да Да department Отделения Нет

Структура таблицы:

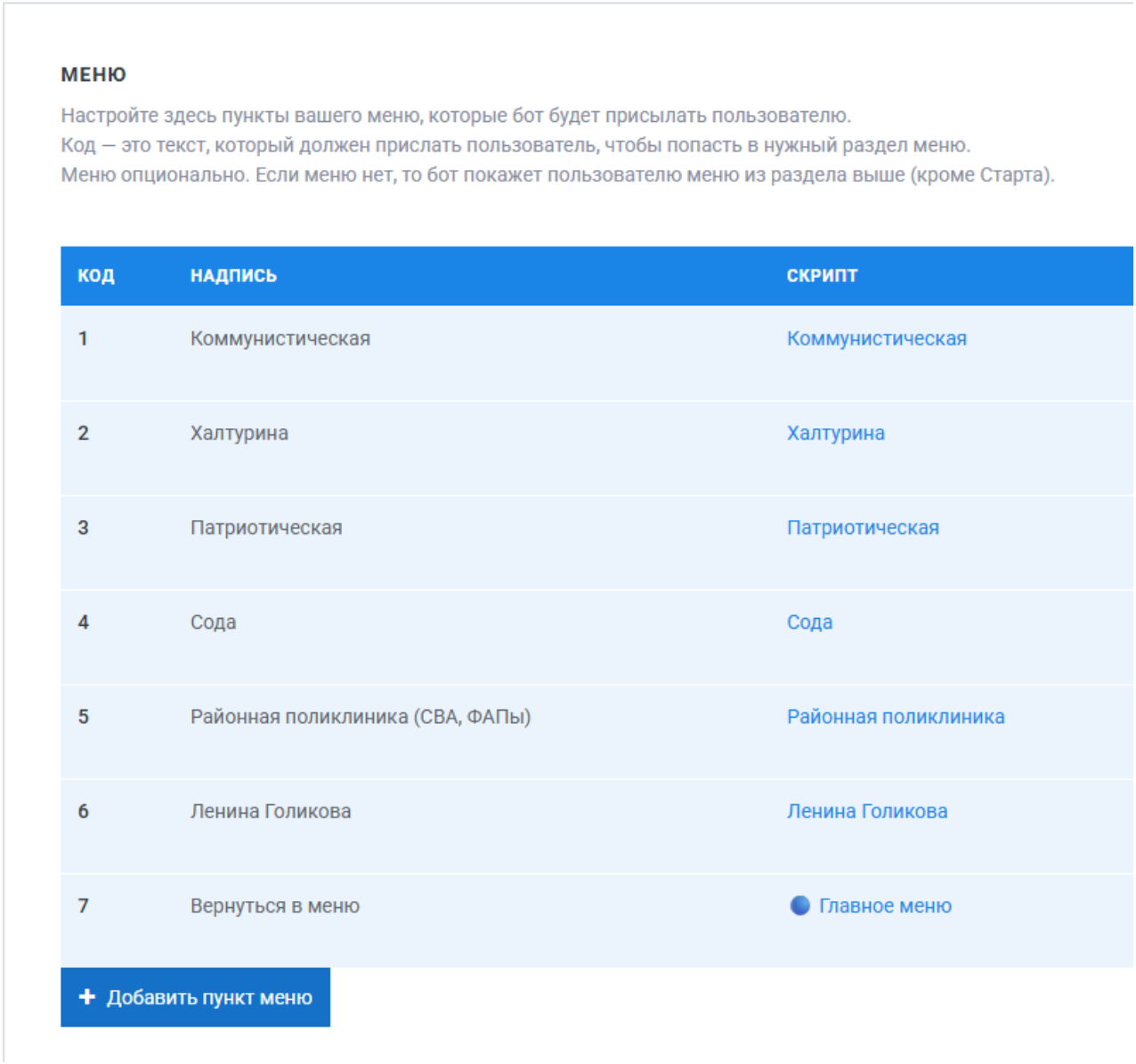
АКТИВНО	СИХР. СТРУКТУРЫ	НАИМЕНОВАНИЕ	ЗАГОЛОВОК	ТИП	СВЯЗЫВАЕМАЯ ТАБЛИЦА	ОБЯЗАТЕЛЬНОЕ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ
Да	Да	id	ID	AUTOINCREMENT		Да	NULL
Да	Да	department	Отделение	TEXT			
Да	Да	structural_unit	Структурное подразделение	TEXT			
Да	Да	adress	Адрес	TEXT			

2. Обновление Адреса

При добавлении нового Адреса его необходимо добавить в виде кнопки в Скрипт:

> https://app.metabot24.com/menubuilder/46480

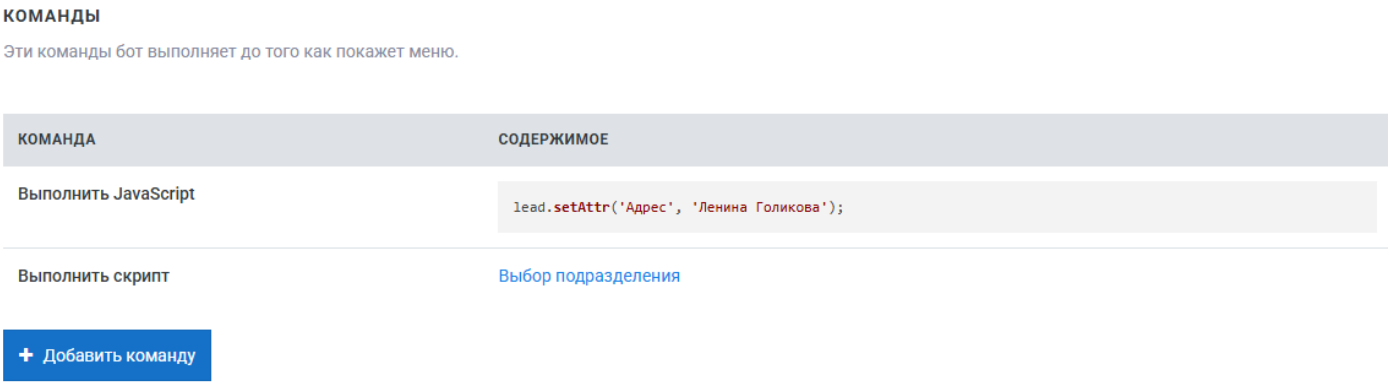
В Меню нажмите на кнопку **Добавить пункт меню**:



В открывшемся окне заполните **Код, ожидаемый в ответе пользователя** и **Подпись пункта меню**. Нажмите **Создать**.

У созданной кнопки нажмите на ссылку в поле скрипт.

В открывшемся скрипте создайте команды по кнопке **Добавить команду**:



В команде **Выполнить JavaScript** напишите код:

```
lead.setAttr('Адрес', "Название адреса");
```

"Название адреса" замените на адрес, который хотите добавить.

В команде **Выполнить скрипт** обязательно укажите скрипт **Выбор подразделения**.

3. Обновление Отделения

При добавлении нового Отделения его необходимо добавить в виде кнопки в Скрипт:

> https://app.metabot24.com/menubuilder/60178

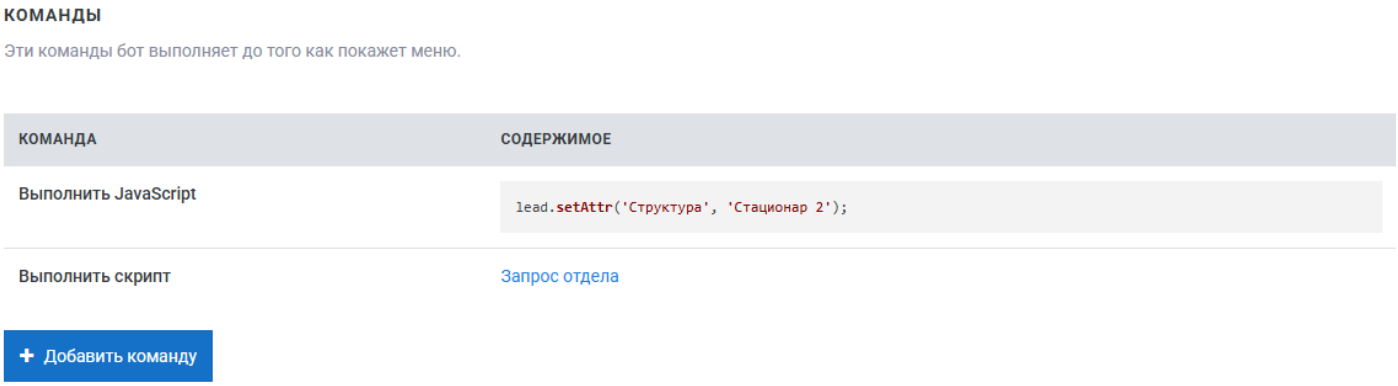
В **Меню** добавьте кнопку по аналогии с адресом. Дополнительно поставьте галочку возле **Использовать условие** и добавьте код:

```
let tasks = table.find('department', [], [["adress", "=", lead.getAttr('Адрес')],
["structural_unit", "=", "Отделение"]]);


return tasks.length
```

"Отделение" замените на отделение, которое хотите добавить.

Перейдите по ссылке на кнопке и создайте команды по аналогии с адресом.



В команде **Выполнить JavaScript** напишите код:

```
lead.setAttr('Структура', 'Отделение');
```

"Отделение" замените на отделение, которое хотите добавить.

В команде **Выполнить скрипт** обязательно укажите скрипт **Запрос отдела**.