Read the Docs Template Documentation

Выпуск 1.0

Read the Docs

февр. 12, 2026

Оглавление

1 Введение **1**
1.1 Пример ордера на разработку бизнес-процесса . . . 1

2 Создание приложения **3**
2.1 Шаги . . . 3
2.2 Результат . . . 4

3 Создание и настройка формы **9**
3.1 Создание формы заявки . . . 9
3.2 Настройка формы . . . 12
3.3 Добавление полей формы . . . 17
3.4 Сохранение формы и финальные штрихи . . . 33
3.5 Условные действия . . . 35

4 Реестры **43**
4.1 Общая информация о реестрах . . . 43
4.2 Создание реестра . . . 44
4.3 Основные настройки реестра . . . 44
4.4 Поля и колонки реестра . . . 46
4.5 Права доступа к реестру . . . 47

5 Маршруты **53**
5.1 Маршруты реестра: назначение и различия . . . 53
5.2 Маршрут «Активация»: интерфейс и логика настройки . . . 55

6 Архитектурные и эксплуатационные рекомендации **91**
6.1 Архитектурные рекомендации . . . 91
6.2 Рекомендации по эксплуатации . . . 94

Глава 1

Введение

Данный курс посвящен работе с платформой Synergy и предназначен для ознакомления с основными принципами построения бизнес-процессов с использованием инструментов системы.

Synergy является low-code платформой, предназначенной для автоматизации бизнес-процессов и создания прикладных решений без необходимости разработки программного кода.

В рамках курса рассматривается последовательность действий, необходимых для реализации бизнес-процесса на основе ордера, начиная с создания приложения и заканчивая настройкой маршрутов и хранения данных.

Материал курса построен таким образом, чтобы последовательно провести пользователя через основные этапы работы с платформой и дать общее понимание логики взаимодействия форм, реестров и маршрутов внутри одного процесса.

1.1 Пример ордера на разработку бизнес-процесса

Ниже представлен пример ордера на разработку бизнес-процесса, который используется в рамках данного курса.

Скачать ордер №1 на разработку процесса: `Ордер №1`

Примечание: В ходе курса данный ордер будет использоваться в качестве примера для поэтапного разбора и реализации бизнес-процесса в платформе Synergy.

Ордер на разработку процесса

Договор (Заказ):	ATLAS
Номер/дата ордера:	№1 от _____2025г.
Наименование процесса:	Автоматизация процесса онлайн заказа услуги
Вход процесса:	Заявка на оказание услуги ATLAS
Выход процесса:	Зарегистрированная заявка на услугу
Уровень сложности:	20 story points (40 чел/часов)
Признак выполнения работы (результат):	Реализована автоматизированная подача заявки на услуги ATLAS через онлайн-форму с присвоением уникального номера, регистрацией в реестре и отправкой уведомления клиенту.
Профиль нагрузки:	3 000 пользователей / заявок ежемесячно (≈ 100 заявок в день, включая действующих и новых клиентов).

Рис. 1: Пример ордера на разработку бизнес-процесса

Глава 2

Создание приложения

Работа над бизнес-процессом в Synergy начинается с создания приложения.

На этом шаге мы создаём приложение — логический контейнер для всех объектов будущего бизнес-процесса. Приложение объединяет формы, реестры, маршруты, справочники и скрипты, относящиеся к одному проекту.

На выходе этого шага у нас появится пустое приложение, внутри которого мы будем дальше создавать форму заявки и весь процесс.

> Приложение — это отдельный рабочий проект или папка, в которой собрано всё, что нужно для выполнения каждого из бизнес-процессов. Приложение может включать в себя несколько бизнес-процессов. Каждый бизнес-процесс может состоять из одной или нескольких форм, реестров и др.объектов приложения
>
> - объекты одного приложения не «ломают» другие приложения;
> - изменения внутри приложения не влияют на чужие процессы;
> - проще тестировать, дорабатывать и переносить решения.

2.1 Шаги

Для того, чтобы создать приложение в системе, нам необходимо:

1) Перейти по адресу проекта: `http://<адрес_сервера>/designer`

После перехода по адресу выполняется авторизация пользователя в системе.

2) После авторизации система предложит создать новое приложение для проекта (если оно еще не создано). Если уже создано одно или несколько приложений, то в открывшемся окне открываем настройки рядом с названием приложения – **«Создать»** (новое) или **«Открыть»** и выбрать уже созданное из дерева:

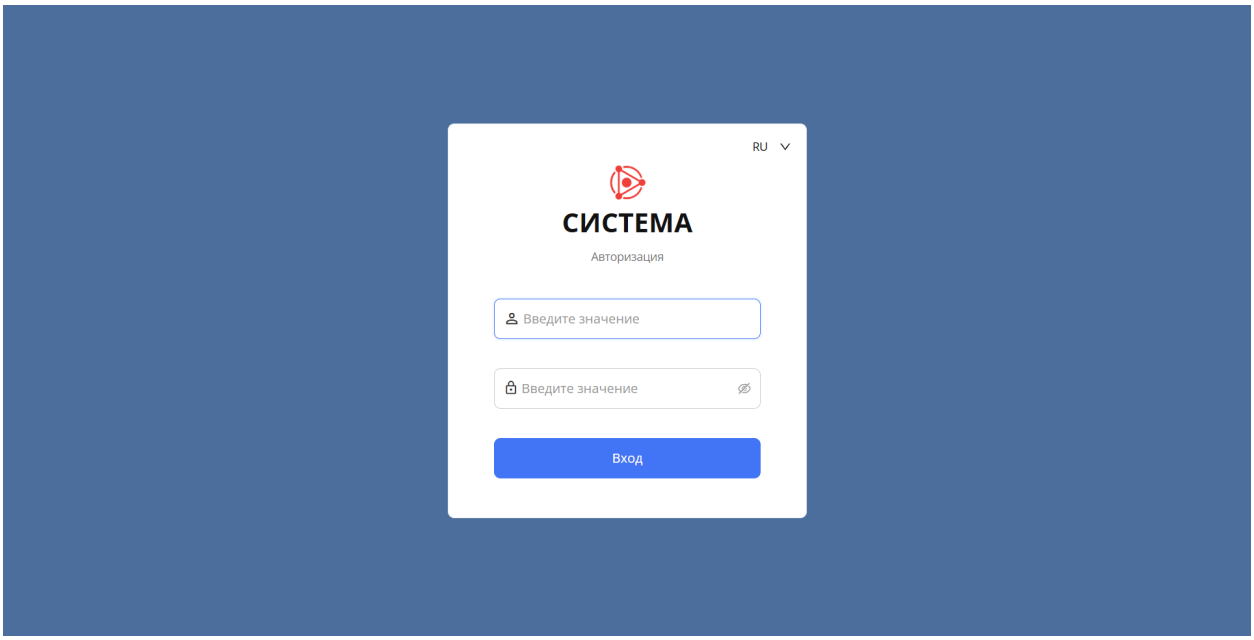


Рис. 1: Авторизация в среде проектирования Synergy

2.1.1 Создание нового приложения

В открывшемся окне указываем:

- **Наименование приложения** – человеко-читаемое название приложения (проекта), например «ATLAS»
- **Код** - название приложения на английском, используя «_» вместо пробелов
- При необходимости отмечаем пункт «Использовать структуру по умолчанию», чтобы система автоматически создала базовые папки. Если структура по умолчанию не выбрана, будет создана только одна корневая папка, внутри которой пользователь сможет самостоятельно формировать структуру приложения.

2.1.2 Выбор приложения из уже созданных

- Кликнуть по иконке меню рядом с названием текущего приложения
- В открывшемся списке выбрать пункт «Открыть»
- В появившемся окне по поиску или вручную выбрать нужное приложение, затем нажать кнопку «Открыть»

2.2 Результат

Приложение создано и открыто. Теперь все дальнейшие элементы процесса будут создаваться внутри него.

См.также:

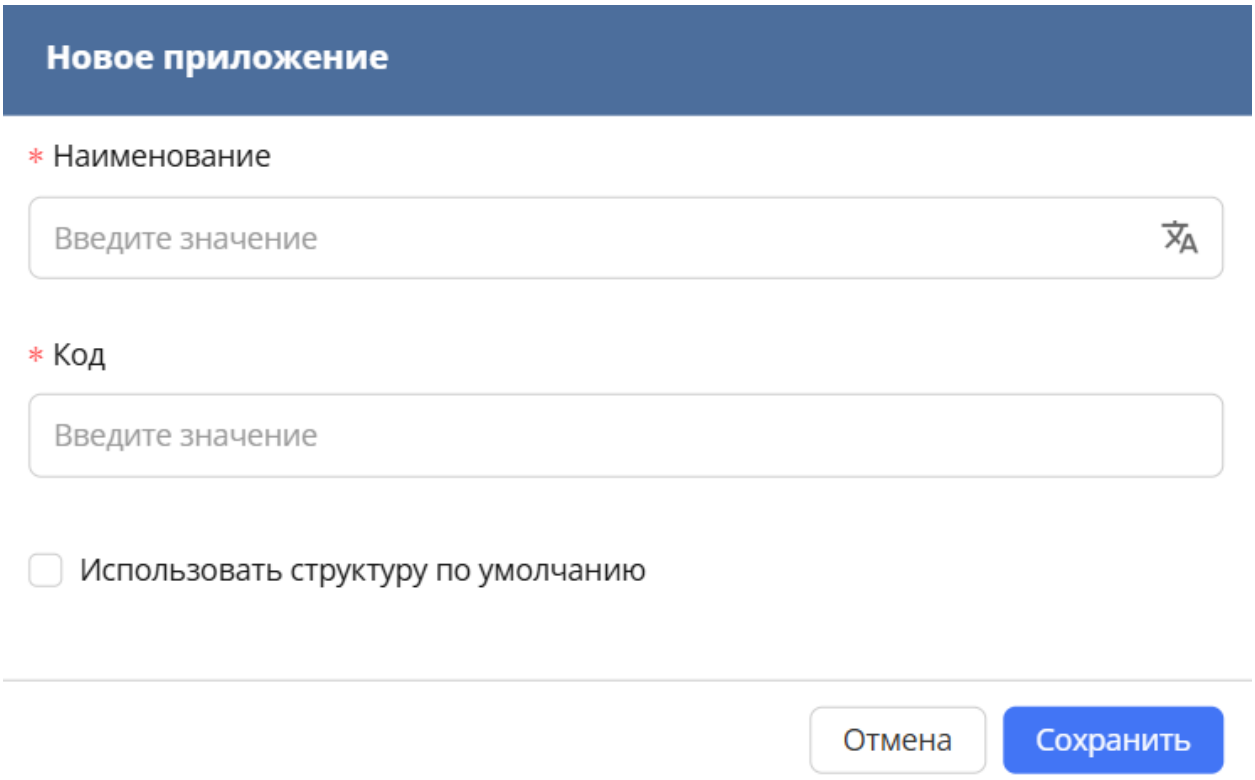
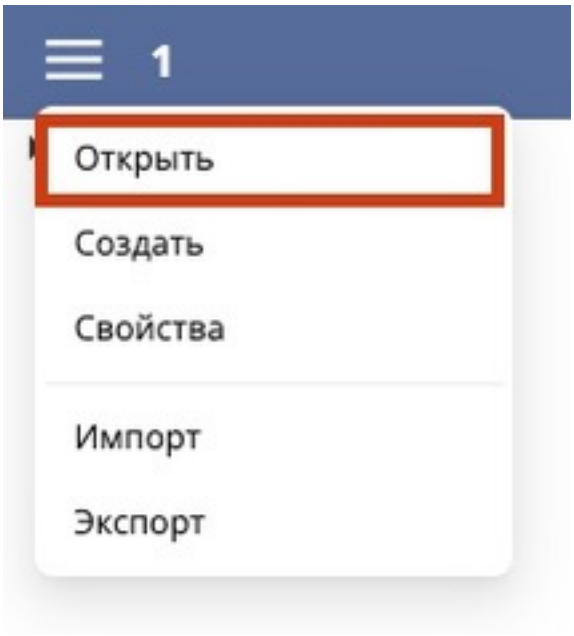
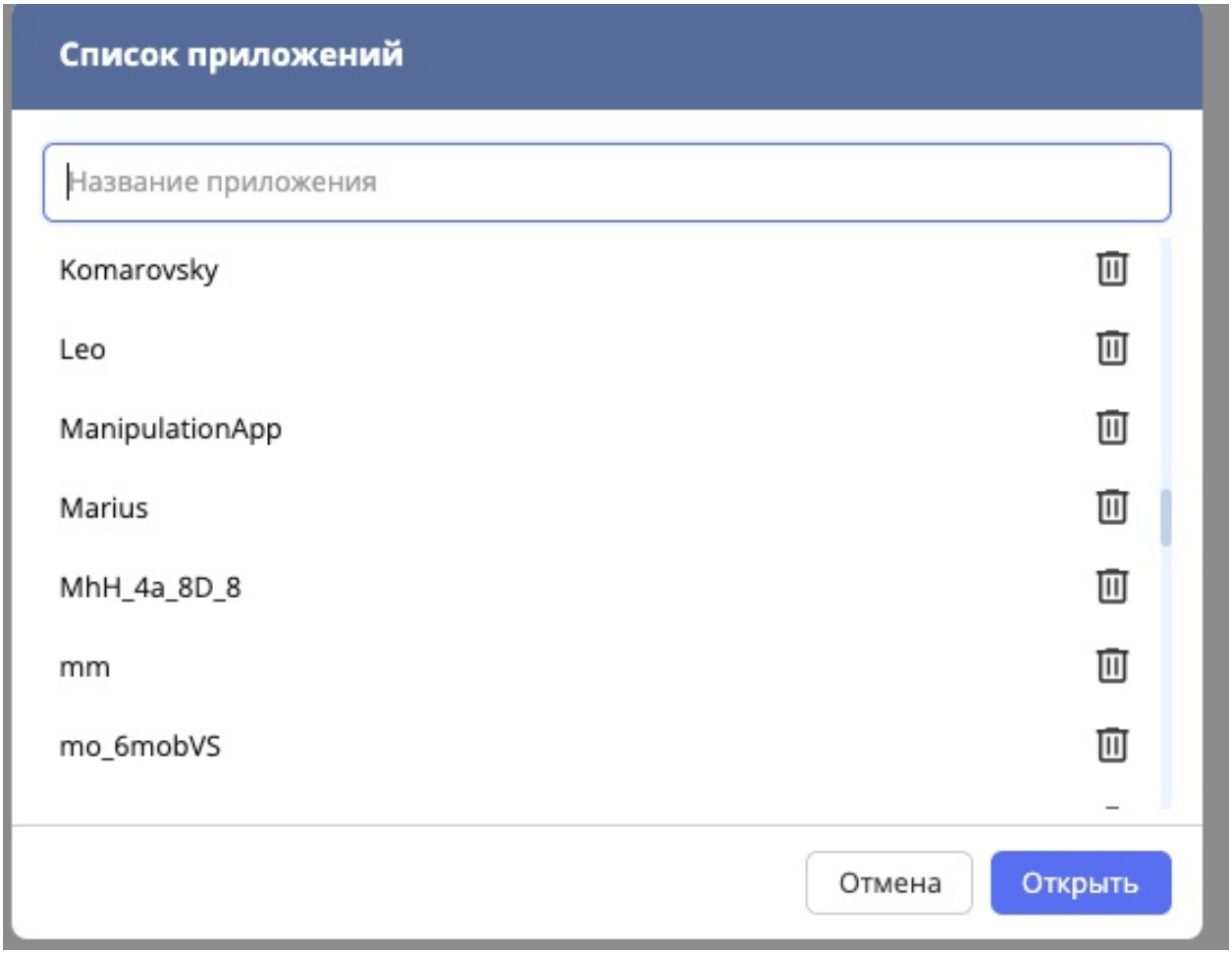


Рис. 2: Создание приложения


1
Открыть
Создать
Свойства
Импорт
Экспорт


Список приложений
Название приложения
Komarovsky
Leo
ManipulationApp
Marius
MhH_4a_8D_8
mm
mo_6mobVS
Отмена
Открыть

Дополнительную информацию можно найти в официальной документации: http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/app.html

Глава 3

Создание и настройка формы

3.1 Создание формы заявки

Форма — это пользовательский интерфейс заявки и вход бизнес-процесса. Форма определяет структуру данных процесса: какие поля существуют, какие обязательны, какие значения в них могут быть введены.

На выходе этого шага у нас будет создана пустая форма, готовая к наполнению полями.

3.1.1 Шаги создания формы

Шаг 1. В дереве приложения выбираем папку, в которой будет храниться форма.

Шаг 2. Кликнуть правой кнопкой мыши по нужной папке → Добавить → Базовые сущности → Форма

Шаг 3. Открывается окно создания формы. Во вкладке «Форма» указываем:

- **Наименование формы** например, «Заявка на услуги компании ATLAS».
- **Код формы** - рекомендуется задавать вручную, чтобы код был читаемым и логичным.

3.1.2 Результат

После задания основных параметров форма готова к наполнению компонентами.

На данном этапе:

- форма создана;
- заданы ее основные параметры;
- можно переходить к настройке структуры и добавлению компонентов.

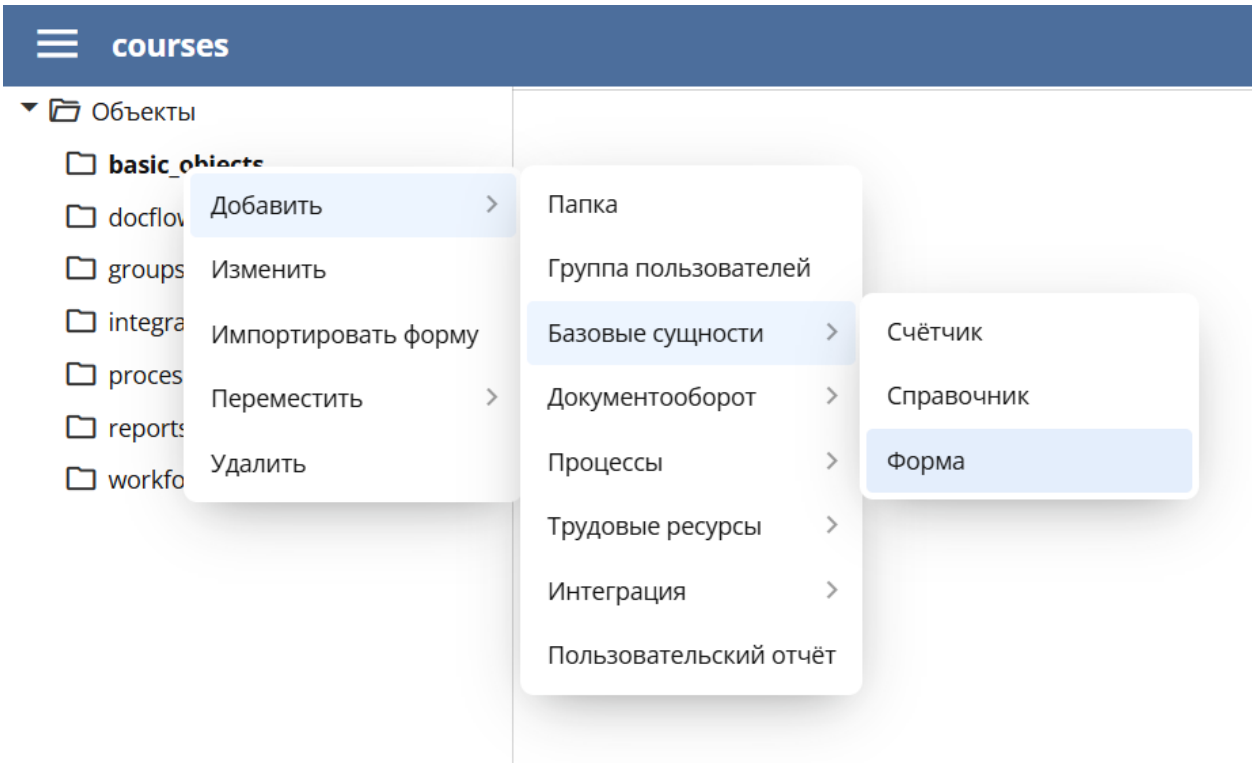


Рис. 1: Добавление формы в структуре приложения

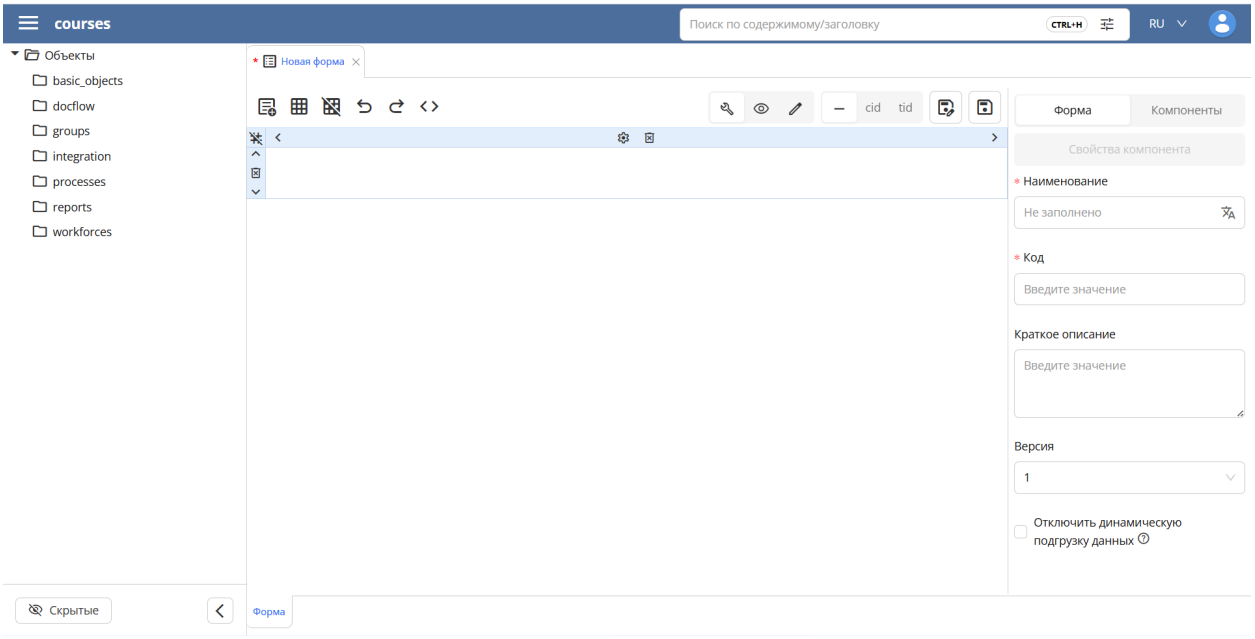


Рис. 2: Пустая форма после создания

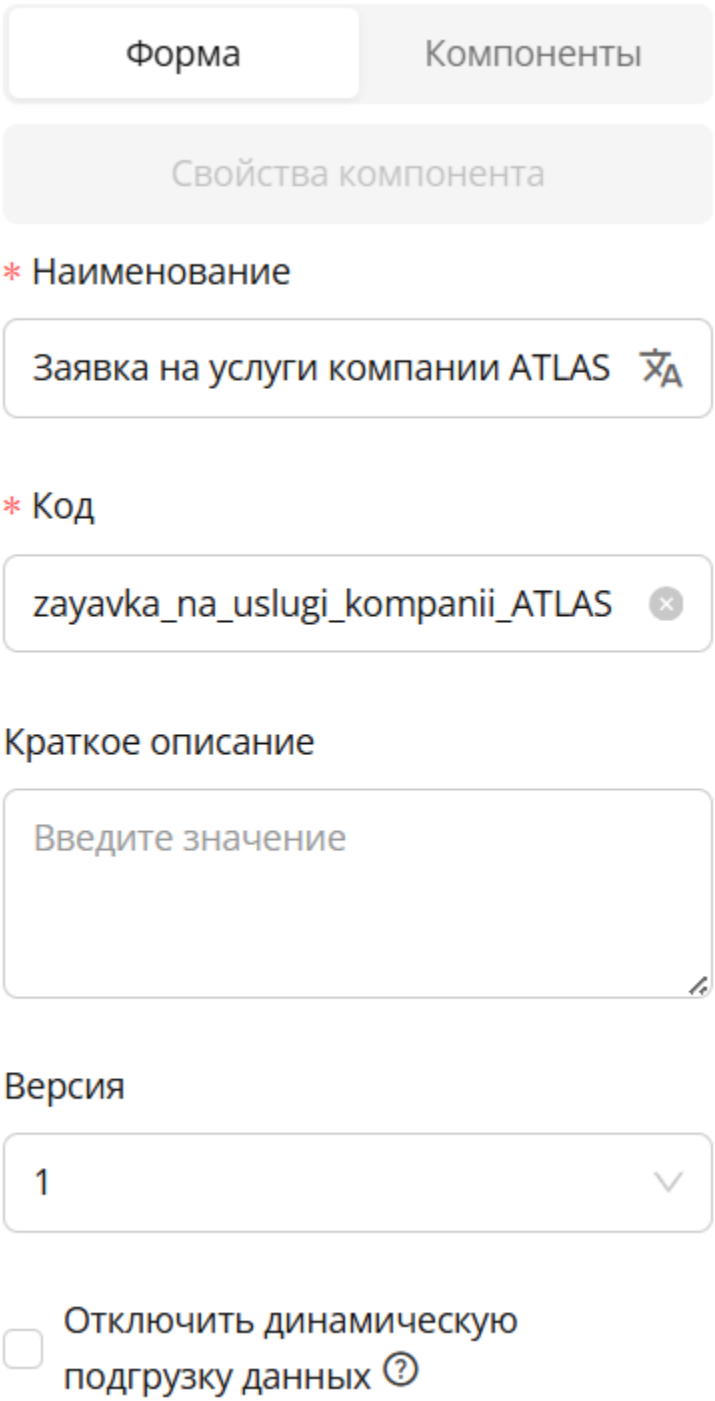


Рис. 3: Основные параметры формы

См.также:

Подробное описание формы, её компонентов и настроек можно найти в официальной документации: http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/form.html

3.2 Настройка формы

После создания формы необходимо выполнить ее настройку. На данном этапе форма подготавливается к дальнейшему наполнению полями в соответствии со структурой ордера.

Перед добавлением полей важно подготовить визуальную структуру формы. Как правило, форма по ордеру делится на логические блоки: данные заявителя, параметры услуги, вложения и т.д.

3.2.1 Переход к добавлению компонентов

Для дальнейшей работы необходимо открыть вкладку **«Компоненты»**.

Во вкладке «Компоненты» осуществляется добавление всех элементов формы в соответствии со структурой данных заявки и требованиями к пользовательскому интерфейсу, указанными в ордере.

Добавление компонентов выполняется непосредственно на форме: сначала выбирается нужная ячейка, после чего из списка компонентов кликом выбирается необходимый элемент.

3.2.2 Шаги настройки формы

Шаг 1. Переходим во вкладку «Компоненты» → раздел «Структура».

Шаг 2. Выделяем ячейку рабочей области и кликом добавляем компонент «Таблица» на форму.

Согласно ордеру форма заявки должна быть организована определенным образом.

Визуально форма делится:

- **вертикально** — на два столбца:
  - левый — наименования полей;
  - правый — поля для ввода данных;
- **горизонтально** — на несколько логических блоков, каждый из которых объединяет связанные между собой поля.

Такая структура необходима для удобства заполнения заявки пользователем и наглядного разделения информации.

Шаг 3. Создаём таблицу с двумя колонками:

- левая колонка — названия полей;
- правая колонка — поля ввода.

Чтобы сделать две колонки — нажимаем стрелку влево/вправо, добавляя нужные столбцы.

Компонент «Таблица» используется для разметки формы, структурирования элементов внутри формы, а также для создания динамических таблиц с возможностью добавления строк во время заполнения.

Шаг 4. Настраиваем ширину колонок через иконку шестерёнки (px - в пикселяъ / % - в процентах / Авто).

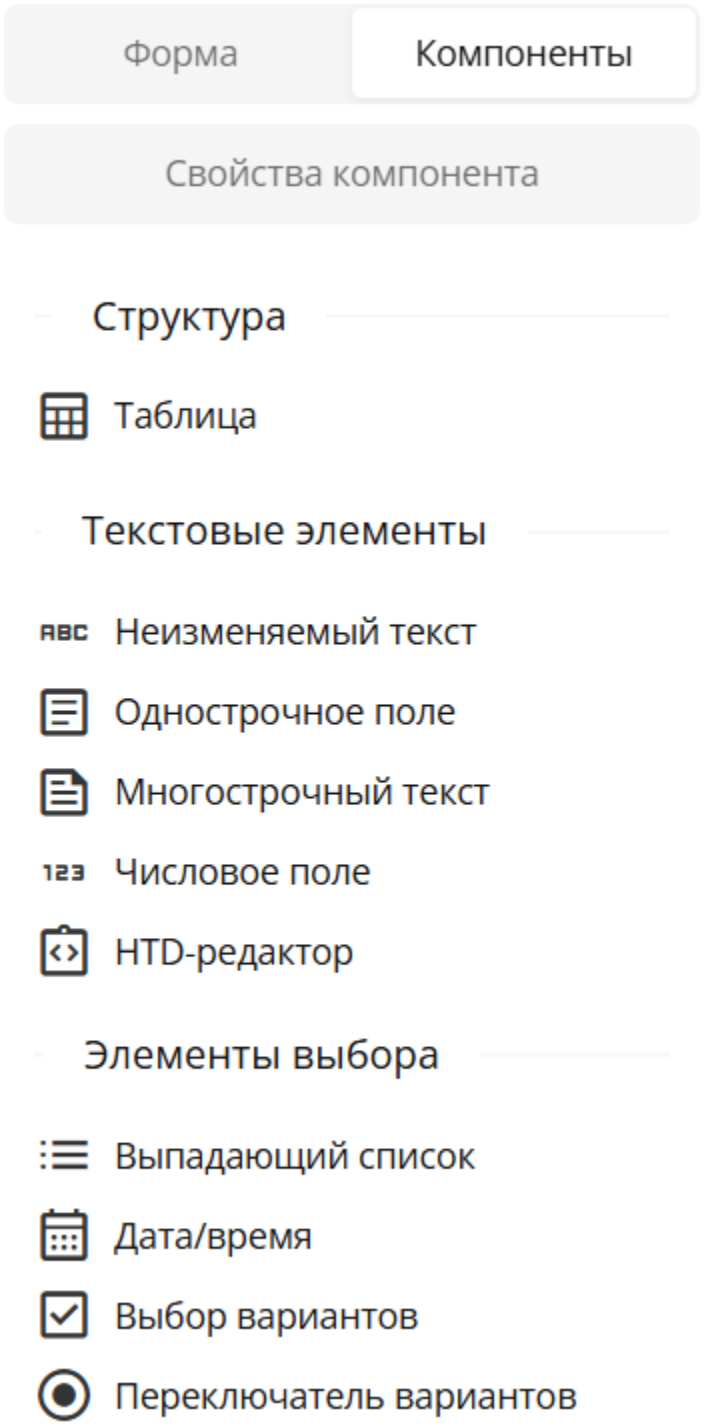


Рис. 4: Вкладка «Компоненты»

3. Пользовательский интерфейс (UI/UX)

3.1. Пользовательский интерфейс «Заявка на оказание услуги ATLAS»

Рис. 1 – Пользовательский интерфейс «Заявка на оказание услуги ATLAS»

Рис. 5: Логика разметки формы согласно ордеру

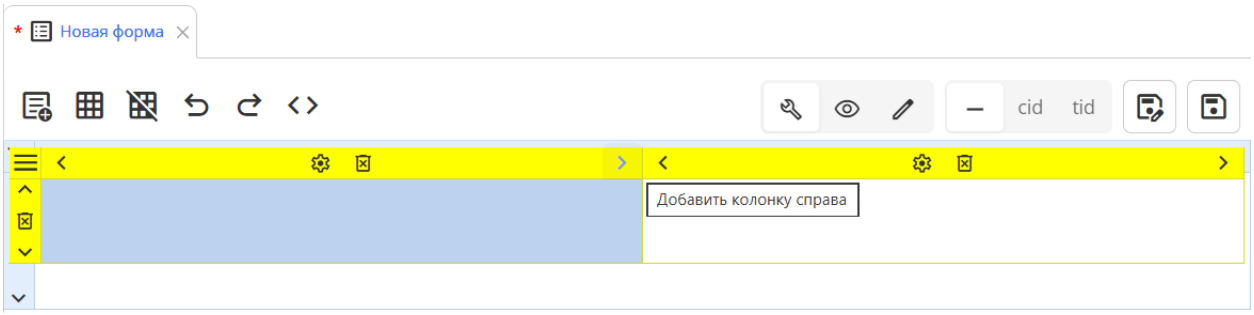


Рис. 6: Добавление компонента «Таблица» и создание колонок

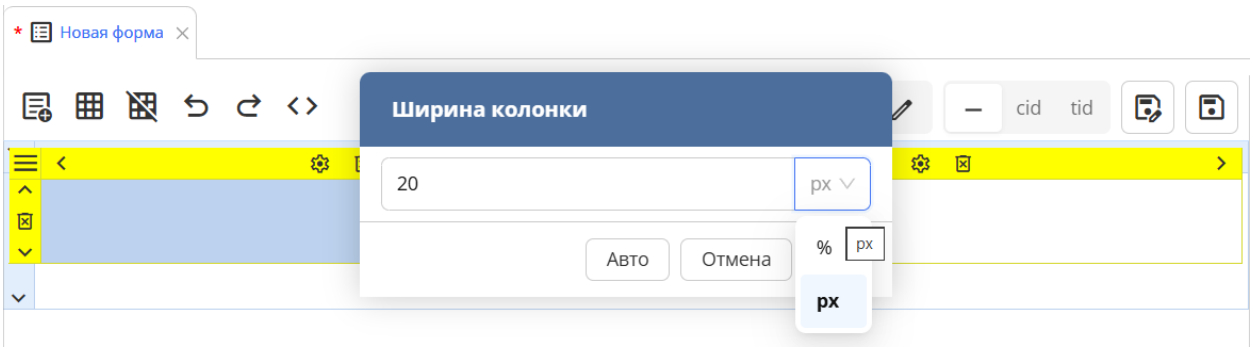


Рис. 7: Настройка ширины столбца таблицы

Шаг 5. Для заголовков секций:

- выделяем две ячейки в строке при помощи shift;
- объединяем их при помощи кнопки «объединить»

- добавляем компонент «Неизменяемый текст»;
- выравниваем текст по центру.

3.2.3 Настройки компонента «Таблица»

Каждый компонент формы, включая таблицу, имеет собственные настройки, а также уникальный код, параметры стиля, выравнивания и шрифта.

Компонент **«Таблица»** поддерживает следующие настройки:

- **Отобразить границы** — отображение внешних и внутренних линий таблицы;
- **Фиксированные размеры таблицы** — фиксирует размеры столбцов, при этом вложенные компоненты подстраиваются под заданную ширину;
- **Выводить содержимое в виде абзаца при просмотре/печати** — отображает данные таблицы в виде текста;
- **Добавлять строки в режиме заполнения** — делает таблицу динамической;
- **Добавить заголовок динамической таблицы** — отображает постоянный заголовок над динамическими строками.

3.2.4 Результат этапа

После выполнения данного этапа:

- заданы основные параметры формы;
- выполнена разметка формы в соответствии с ордером;

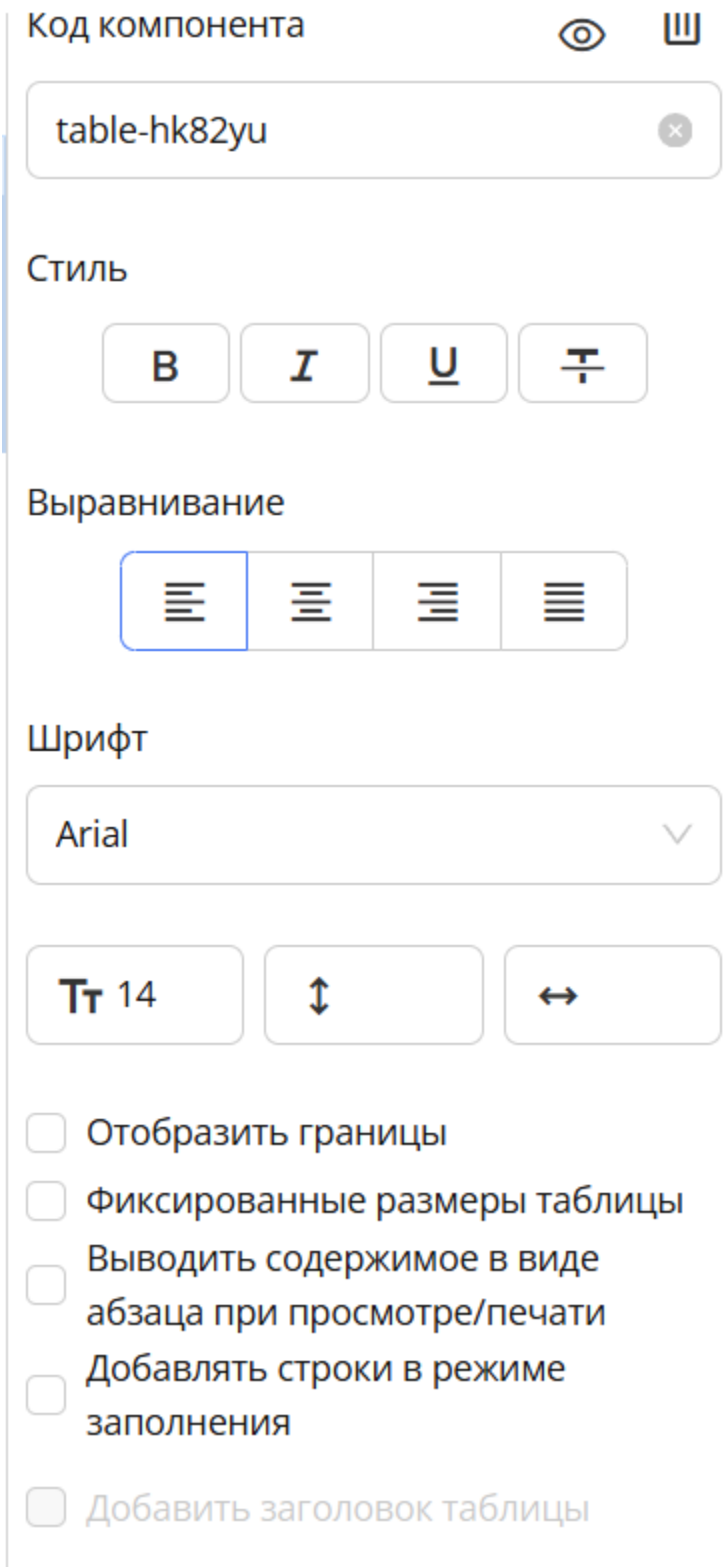


Рис. 8: Настройки компонента «Таблица»

- форма подготовлена к добавлению полей заявки.

В следующем этапе будет выполнено добавление полей формы в соответствии со структурой данных ордера.

См.также:

Дополнительную информацию можно найти в официальной документации: http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/form/form-structure.html

3.3 Добавление полей формы

На этом шаге мы наполняем форму полями ввода. Поля добавляются в соответствии со структурой данных ордера.

Для каждого поля на форме используется определенный тип компонента, в зависимости от типа данных.

Важно: Важно отметить, что при создании формы обязательным является присвоение компонентам интуитивно понятного кода. Для того, чтобы сменить автоматически сгенерированный системой код компонента, необходимо выделить нужный компонент и поменять его код в поле «код компонента» - предпочтительно в виде «тип_название»

3.3.1 Общий принцип добавления полей

Добавление любого поля на форму выполняется по одному и тому же принципу:

1. Выделяется нужная ячейка таблицы.
2. В панели **«Компоненты»** кликом выбирается необходимый компонент.
3. Компонент добавляется в выбранную ячейку.
4. В свойствах компонента выполняется его настройка.

Здесь мы рассмотрим создание полей на нескольких примерах прямо из ордера.

3.3.2 Шаги добавления полей на форму

Шаг 1. В левую колонку таблицы добавляем компонент **«Неизменяемый текст»**. Для ускорения процесса можно добавить сначала сразу все наименования полей, затем перейти к компонентам.

Шаг 2. В правую колонку добавляем компонент в зависимости от типа данных, указанного в ордере:

- Номер заявки — компонент **«Номер»**.
- Справочные значения — **«Выпадающий список»**.
- Выбор одного варианта — **«Переключатель вариантов»**.
- Текстовые данные — **«Однострочное поле»**.
- Комментарии — **«Многострочный текст»**.
- Файлы - компонент **«Файл»**

Шаг 3. Для каждого поля во вкладке **«Свойства»**:

- Задаем код поля (читаемый и содержащий смысловую нагрузку предназначения поля)
- При необходимости отмечаем обязательность

3.3.3 Поле 1. Порядковый номер заявки

Шаг 1. В левой ячейке таблицы добавляем компонент **«Неизменяемый текст»**, который используется для отображения названия поля.

В поле **«Надпись»** указывается название **«Порядковый номер заявки»**. При необходимости добавляем перевод.

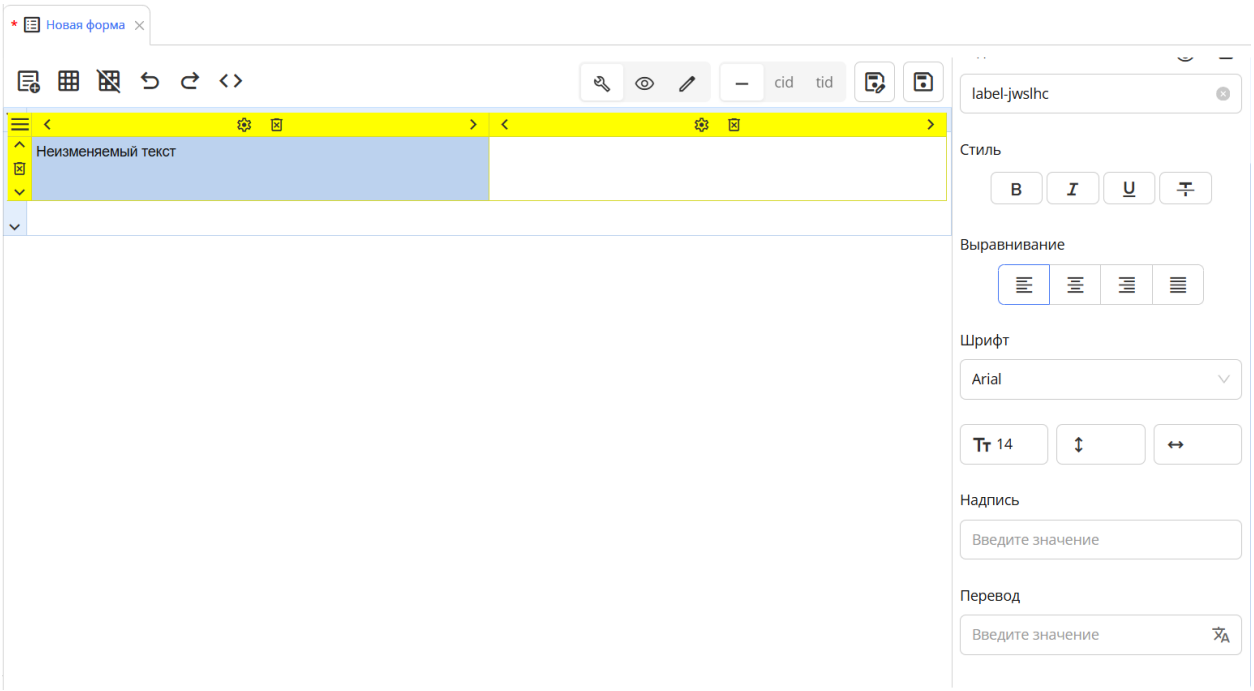


Рис. 9: Название поля с использованием компонента «Неизменяемый текст»

Шаг 2. В правой ячейке добавляется компонент **«Номер»** из раздела **«Специальные»**.

Данный компонент будет использовать шаблон номера, к созданию которого мы перейдем позже.

Не забываем поменять код поля на читаемый и содержащий смысл.

3.3.4 Добавление строк в таблицу

Для добавления следующего поля необходимо добавить новую строку в таблицу.

Добавление строки выполняется с помощью стрелки **«вниз»**, расположенной внизу таблицы.

3.3.5 Поле 2. Статус заявки

Поле «Статус заявки» имеет тип данных **«Справочник»**.

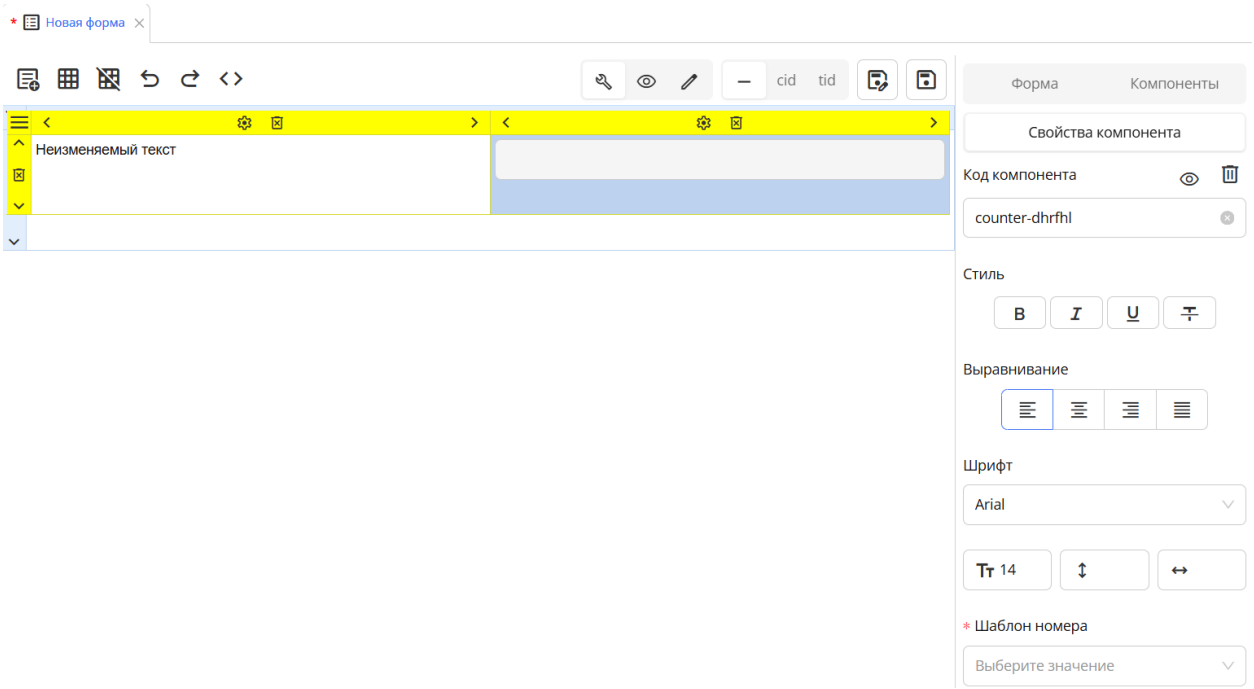


Рис. 10: Компонент «Номер» для порядкового номера заявки

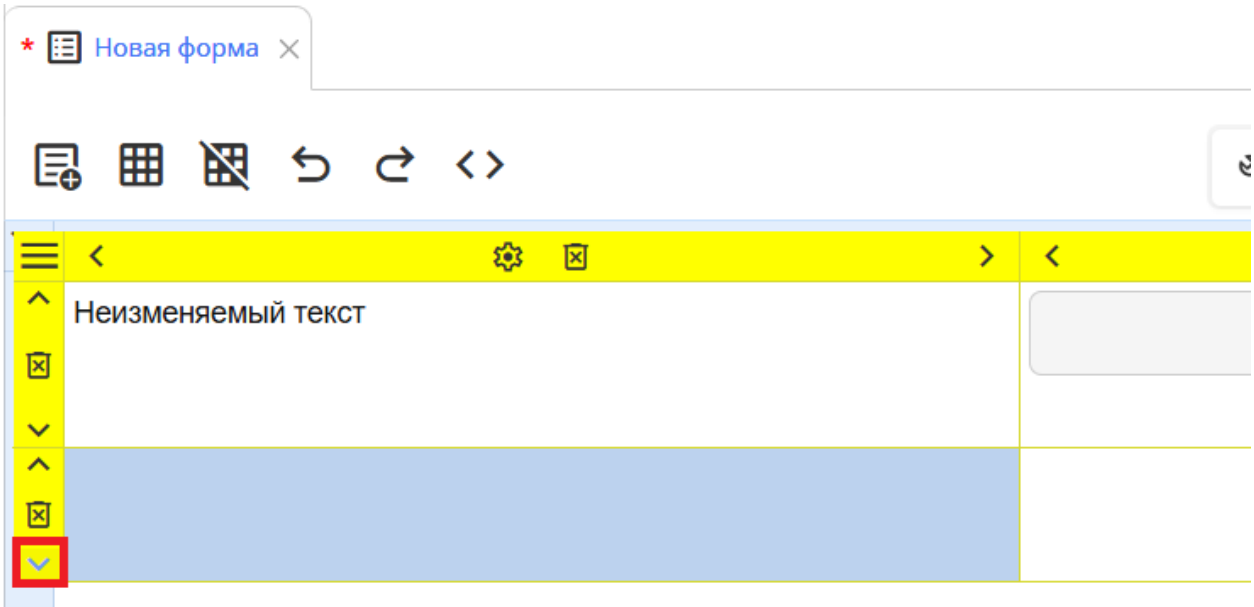


Рис. 11: Добавление новой строки в таблицу

Шаг 1. В левую ячейку добавляем компонент **«Неизменяемый текст»** и указываем название поле «Статус заявки»

Шаг 2. В правую ячейку из раздела «Элементы выбора» добавляем компонент **«Выпадающий список»**

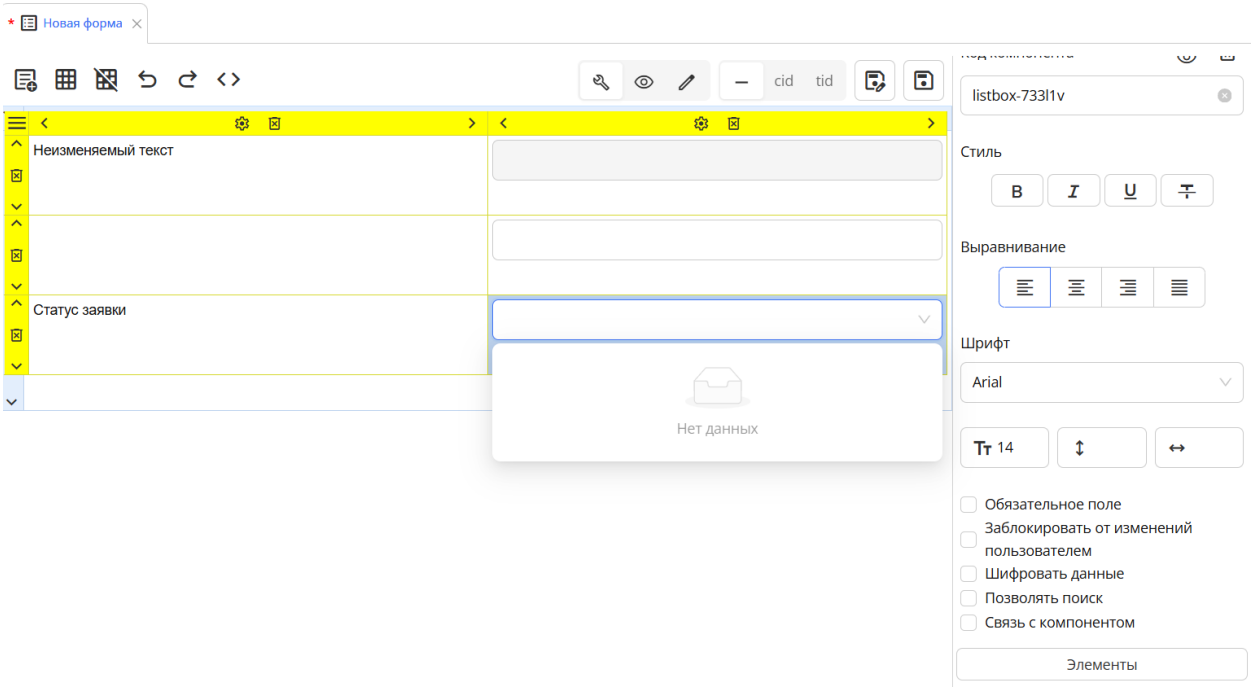


Рис. 12: Компонент «Выпадающий список»

Настройка возможных значений выполняется во вкладке **«Элементы»**.

Список значений может быть:

- создан вручную с помощью кнопки «Добавить ряд данных»;
- выбран из уже существующих справочников системы.

В режиме заполнения пользователю отображаются значения из столбца «Наименование», а системным значением является значение из столбца «Значение».

Компонент «Выпадающий список» имеет следующие настройки:

- **«Обязательное поле»** - пока не работает
- **«Заблокировать от изменений пользователем»** - блокирует выбор значения из справочника пользователем во время редактирования заявки.
- **«Шифровать данные»** используется для защиты конфиденциальной информации. Если опция включена, значения поля сохраняются в системе в зашифрованном виде и недоступны для чтения вне платформы. Шифрование включается галочкой в настройках компонента формы. Для пользователей работа с полем не меняется - данные автоматически шифруются при сохранении и расшифровываются при просмотре. Примечание: Для работы функционала необходимо попросить администратора платформы включить использование шифрования.
- **«Позволять поиск»** позволяет найти нужное значение в справочнике

- **«Связь с компонентом»** позволяет связывать справочники между собой, для фильтрации определенных значений в зависимости от другого справочника.

Объект приложения «Справочник»

Справочник — это отдельный объект приложения, представляющий собой таблицу со списком значений, которая используется в компонентах выбора (выпадающие списки, переключатели вариантов и т.д.).

Справочник нужен, чтобы:

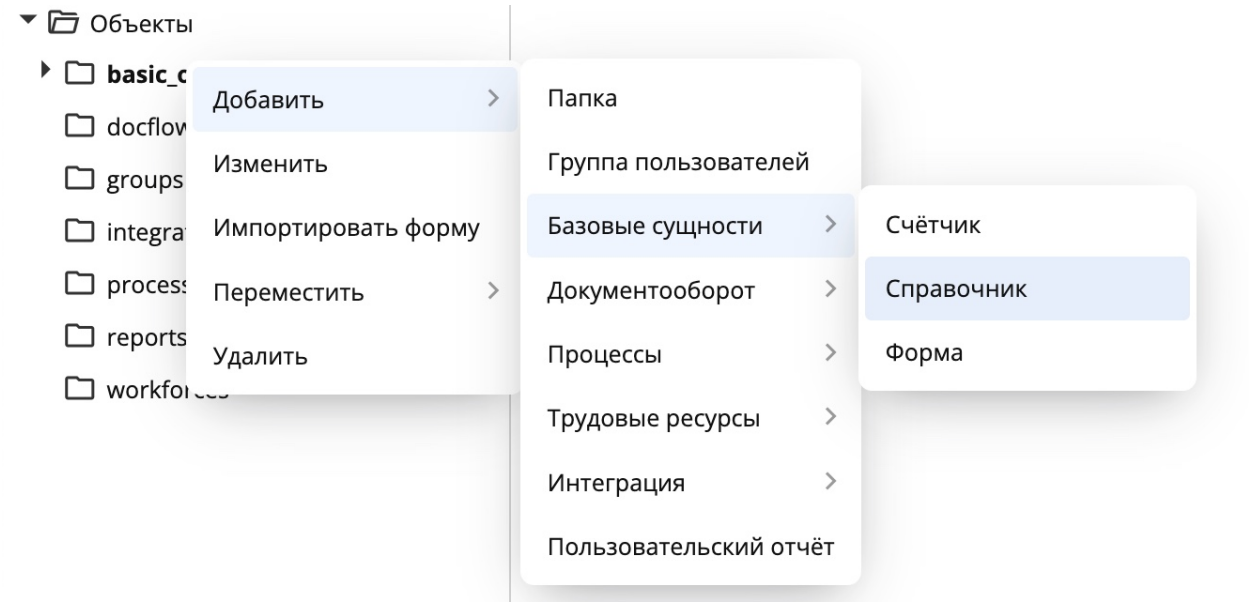
- не вводить одни и те же значения каждый раз;
- избежать ошибок и опечаток;
- использовать единые значения во всех формах и процессах;
- управлять логикой процесса (условия, переходы, фильтрация).

В этом шаге мы создадим статичный справочник, который будет универсальным для всего приложения.

Создание справочника:

Шаг 1. В дереве приложения кликаем правой кнопкой мыши по нужной папке.

Шаг 2. Выбираем: Добавить → Базовые сущности → Справочник



Шаг 3. В открывшемся окне указываем:

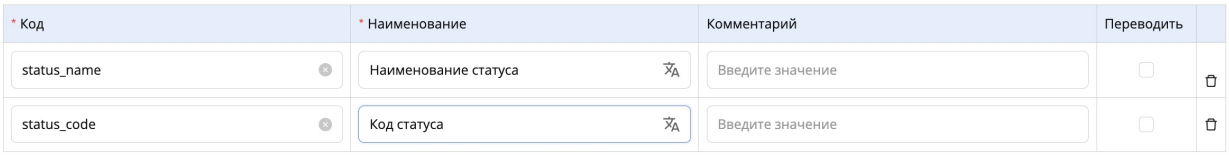
- **Наименование справочника** (например, «Статус заявки»);
- **Код справочника** — интуитивно понятный (можно оставить автоматически сгенерированный транслит).

Шаг 4. Ниже отображается таблица колонок справочника. Эта таблица определяет, какие колонки будет иметь справочник. Добавляем минимум одну строку таблицы, указав:

- Наименование колонки;
- Код колонки.

Пример:

- Наименование → status_name
- Код → status_code



При необходимости, отмечаем галочкой столбец «Перевод» если нужно добавить переводы значений.

Шаг 5. Сохраняем справочник

Справочник создан. После сохранения рядом с кнопкой «Сохранить» появляется вкладка «Элементы».

Переходим к добавлению значений в справочник

Добавление значений в справочник:

На этом этапе мы наполняем справочник конкретными значениями, которые позже будут отображаться пользователю в форме.

Шаг 1. Переходим во вкладку **«Элементы»**

Шаг 2. Нажимаем кнопку «+» на верхней панели – добавляются пустые строки

Шаг 3. Заполняем значения в созданных колонках

Привязка справочника к полю на форме:

Шаг 1. Возвращаемся в форму заявки

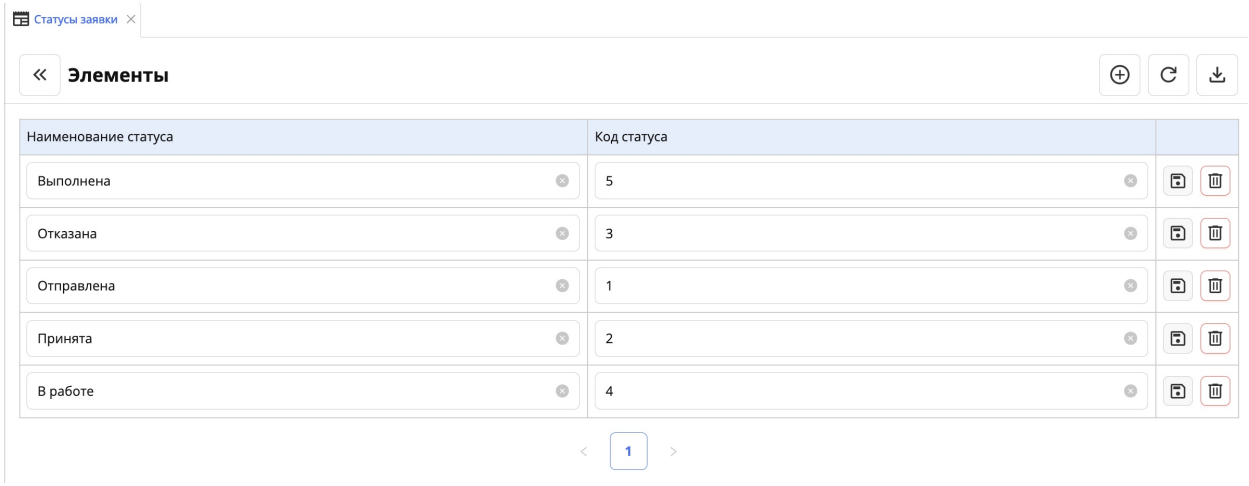
Шаг 2. Кликаем по компоненту «Выпадающий список» в соответствующем поле (например «Статус заявки»)

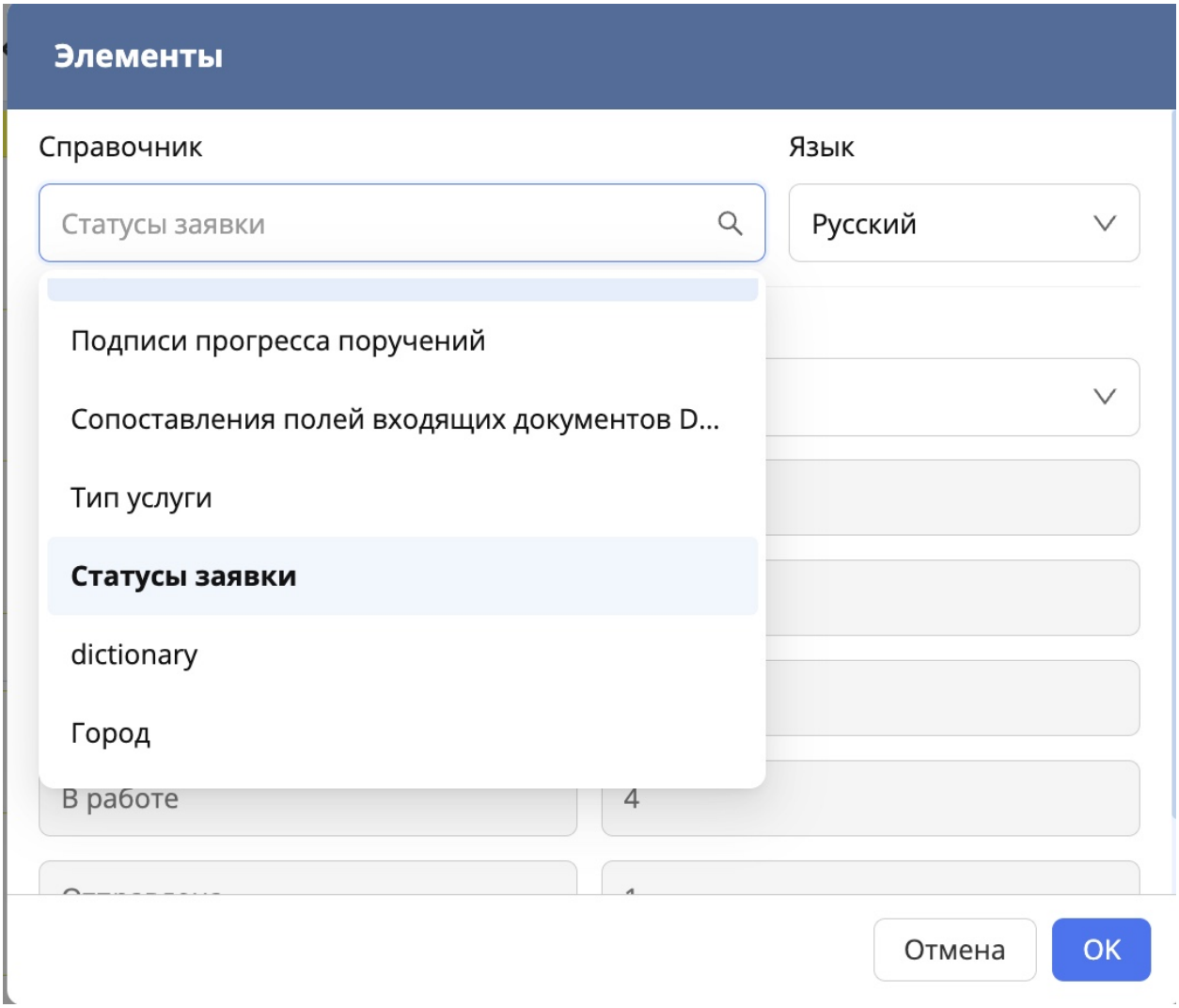
Шаг 3. Переходим во вкладку «Элементы» в свойствах компонента

Шаг 4. В выпадающем списке выбираем созданный справочник.

- В столбце «Наименование» выбираем колонку «Наименование статуса».
- В столбце «Значение» выбираем колонку «Код статуса»

Шаг 5. Нажимаем «ОК»

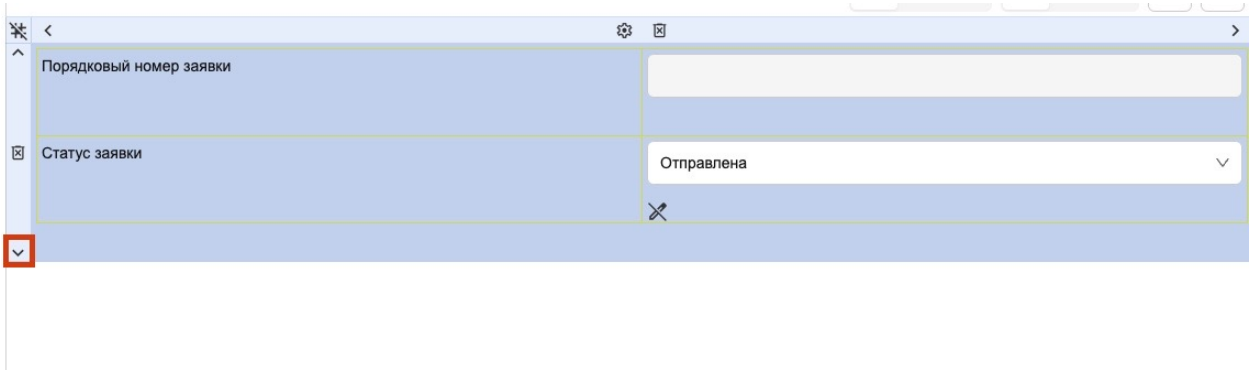

Статусы заявки
Элементы
Наименование статуса
Код статуса
Выполнена
5
Отказана
3
Отправлена
1
Принята
2
В работе
4
1


Элементы
Справочник
Статусы заявки
Язык
Русский
Подписи прогресса поручений
Сопоставления полей входящих документов D...
Тип услуги
Статусы заявки
dictionary
Город
В работе
4
Отмена
OK

3.3.6 Визуальное разделение

Для того чтобы визуально отделить основные данные заявки от раздела «Анкета», используем еще один компонент «Таблица» с заголовком.

Шаг 1. Добавляем еще одну свободную строку на основной форме



Шаг 2. В новую строку добавляем компонент «Таблица»

Шаг 3. Разделяем таблицу на две колонки и настриваем ширину колонок аналогично первой таблице.

Шаг 4. Создание заголовка:

- Выделяем две верхние ячейки таблицы, зажав клавишу Shift.
- Нажимаем на иконку «Объединить» в левом верхнем углу редактора формы

Шаг 5. В объединенную ячейку добавляем компонент **«Неизменяемый текст»**:

- Заполняем текст заголовка «Анкета»
- Делаем текст жирным
- Выравниваем по центру

В новые ячейки таблицы продолжаем добавлять поля этого раздела.

Таким образом мы будем отделять разделы заявки согласно ордеру.

3.3.7 Поле 3. Тип заявителя

Для поля «Тип заявителя» мы используем компонент **«Переключатель вариантов»**. Т.к в данном случае предполагается использование компонента, позволяющее выбрать только один возможный вариант.

Переключатель вариантов по своим настройкам похож на компонент «Выпадающий список», внутри него так же используется вложенный справочник, либо возможные значения заполняются вручную по нажатию на вкладку **«Элементы»**

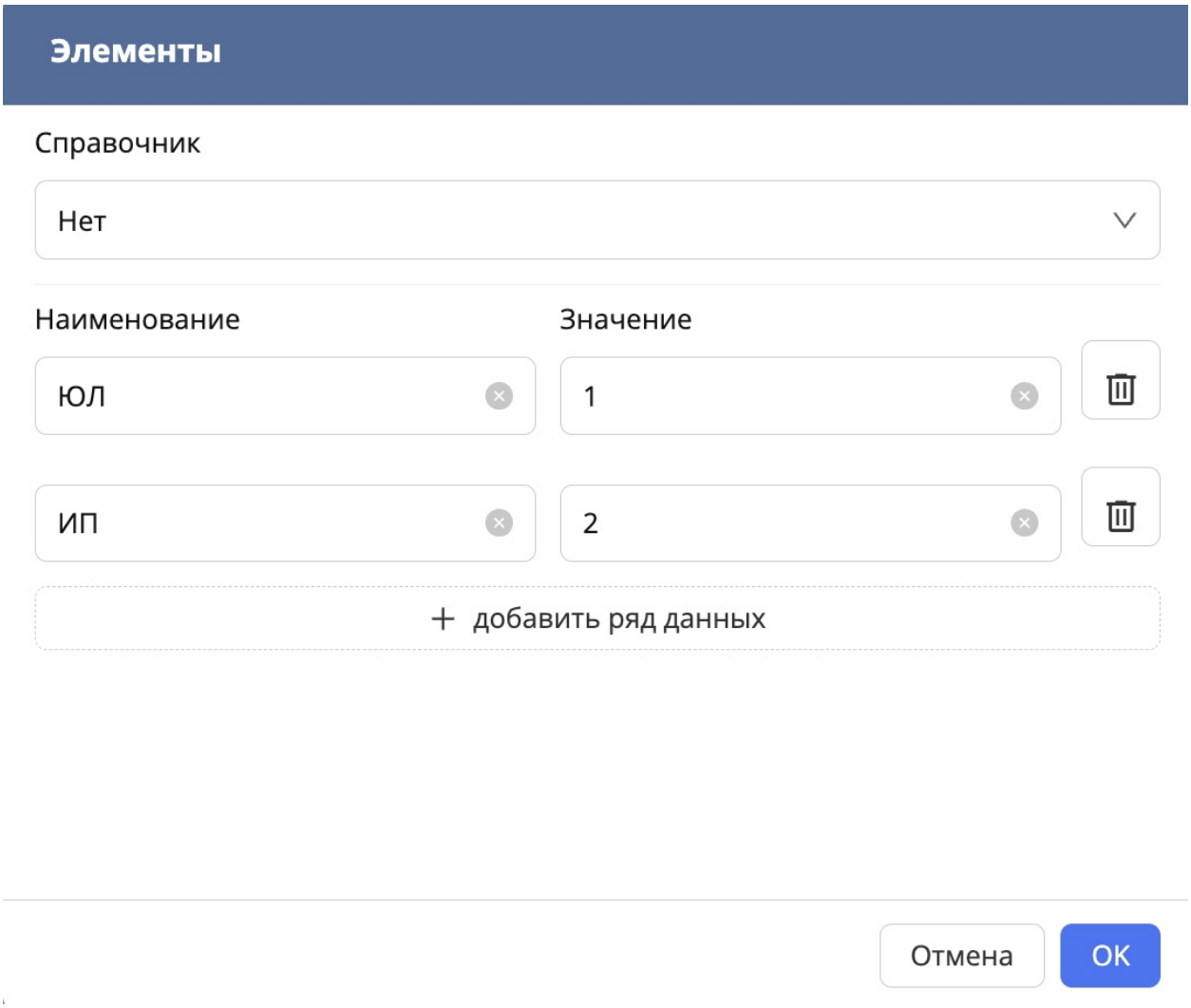
Шаг 1. На форме в нужной строке добавляем компонент **«Переключатель вариантов»**.

Шаг 2. Кликаем по компоненту и переходим в его настройки.

Шаг 3. Открываем вкладку **«Элементы»**.

Шаг 4. В таблице ниже добавляем возможные варианты:

- указываем наименование значения («ЮЛ», «ИП»);
- задаём соответствующее значение (код, может быть как числовым так и буквенным).



Шаг 5. После указания всех необходимых значений нажимаем «ОК».

3.3.8 Поля 4,5,6,7,8,9: «БИН организации», «ИИН индивидуального предпринимателя», «Наименование организации или индивидуального предпринимателя», «ФИО Руководителя организации», «Номер телефона», «Адрес электронной почты»

Поля 4–9 предназначены для ввода идентификационных и контактных данных заявителя. Все эти значения представляют собой короткий текст, вводимый в одну строку, поэтому для них используется компонент **«Однострочное поле**.

Компонент **«Однострочное поле»** позволяет вводить и отображать произвольное текстовое значение в одну строку (без абзацев) и подходит для таких данных, как:

- идентификаторы (БИН, ИИН);
- наименования и ФИО;
- номер телефона;
- адрес электронной почты.

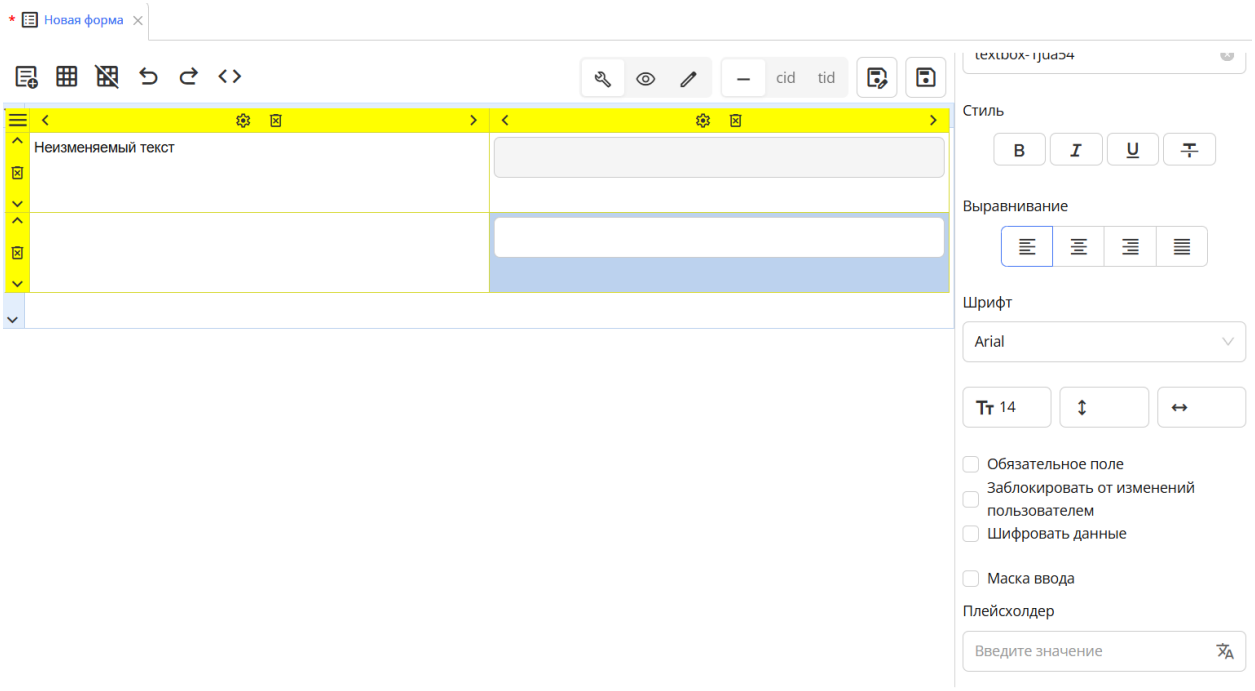


Рис. 13: Компонент «Однострочное поле»

Для компонента «Однострочное поле» доступны следующие настройки:

- **Обязательное поле** — делает поле обязательным для заполнения;
- **Заблокировать от изменений** — запрещает пользователю изменять значение;
- **Шифровать данные** — шифрует введенные значения;
- **Маска ввода** — ограничивает формат вводимых данных;
- **Плейсхолдер** — отображает подсказку внутри пустого поля.

Шаг 1. Для каждого из полей 4-9 добавляем в правую ячейку компонент **«Однострочное поле»**

Шаг 2. В настройках каждого компонента присваем читаемый и логичный код поля.

Шаг 3. Отмечаем галочку «Обязательное поле» — согласно звёздочке в ордере рядом с названием поля.

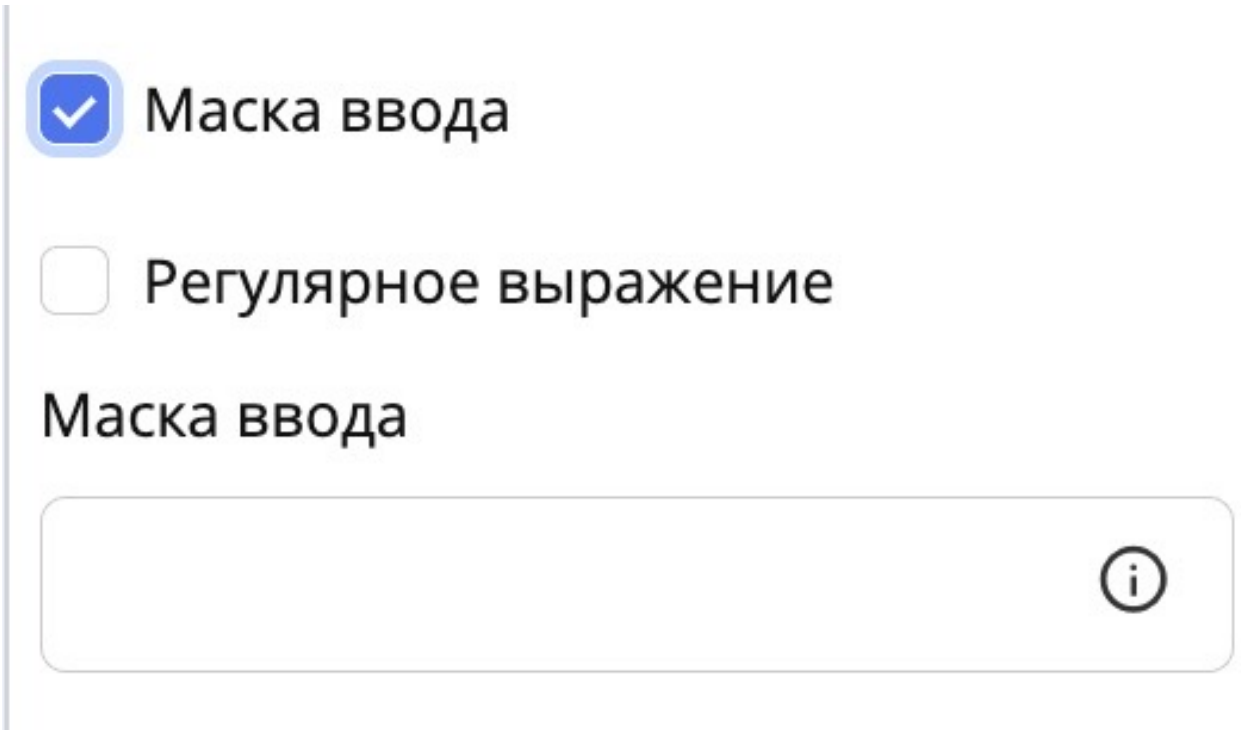
Шаг 4. Отмечаем галочку «Заблокировать от изменений» тем полям, у которых в примечании ордера указано что поля заполняются автоматически.

Настройки формата (маска ввода и регулярные выражения)

Маска ввода настраивается отдельно для каждого компонента.

1. Формат полей **«ИИН индивидуального предпринимателя» и «БИН организации»**

Шаг 1. В настройках компонента включаем галочку **«Маска ввода»**



Шаг 2. В октрывшемся поле открываем настройки маски ввода кликнув по иконке настроек

Шаг 3. В таблице с настройками для задания формата 12-значного числового значения используем маску ##########.

Маску можно:

- Ввести вручную
- Кликом по строке в предложенной таблице

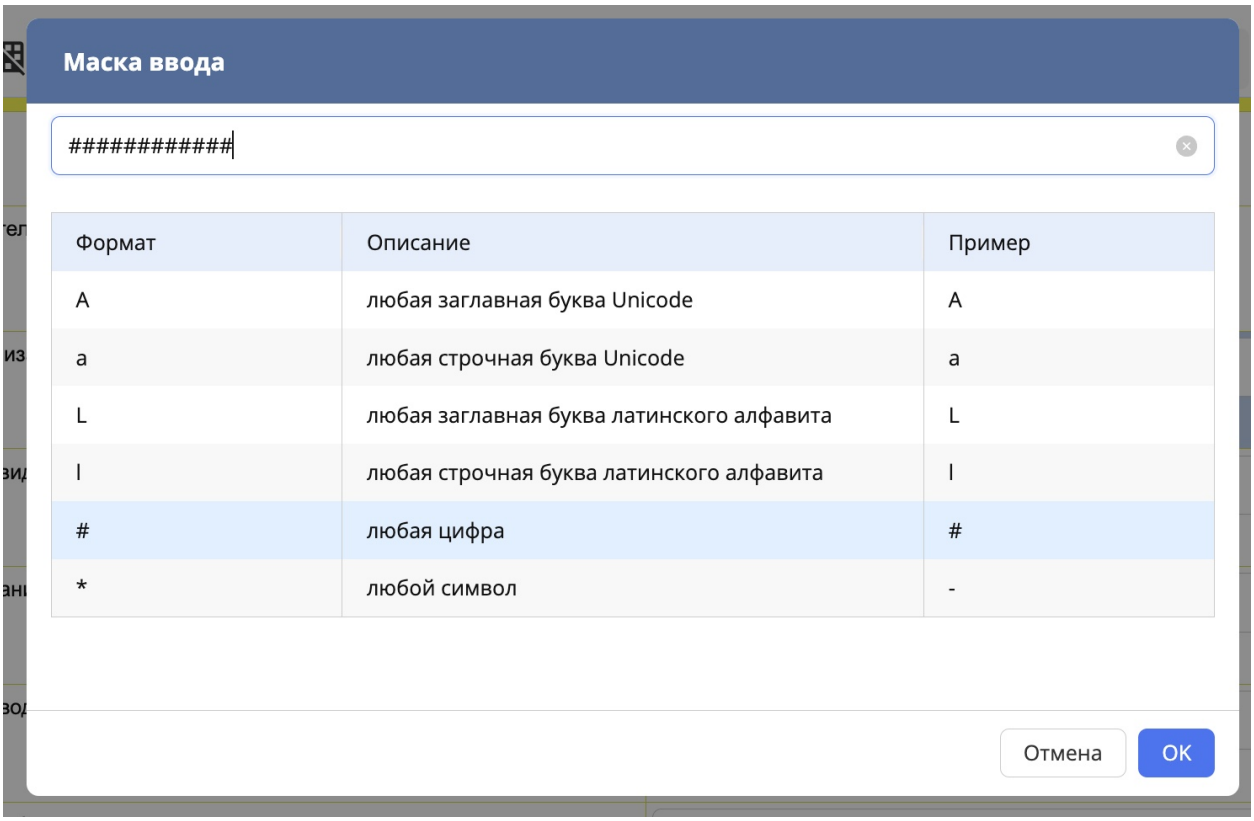
2. Формат поля **«Номер телефона»**

Шаг 1. Для поля «Номер телефона» также включаем настройку «Маска ввода».

Шаг 2. В маске указываем формат с фиксированными символами, например: +7 (###) ###-##-##

Фиксированные символы (например, +7) будут:

- автоматически подставляться;



- недоступны для удаления или изменения пользователем.

Это позволяет обеспечить корректный формат ввода номера телефона.

3. Настройки формата поля **«Электронная почта»**

Для поля «Адрес электронной почты» требуется строгая проверка формата.

Шаг 1. Включаем настройку «Маска ввода».

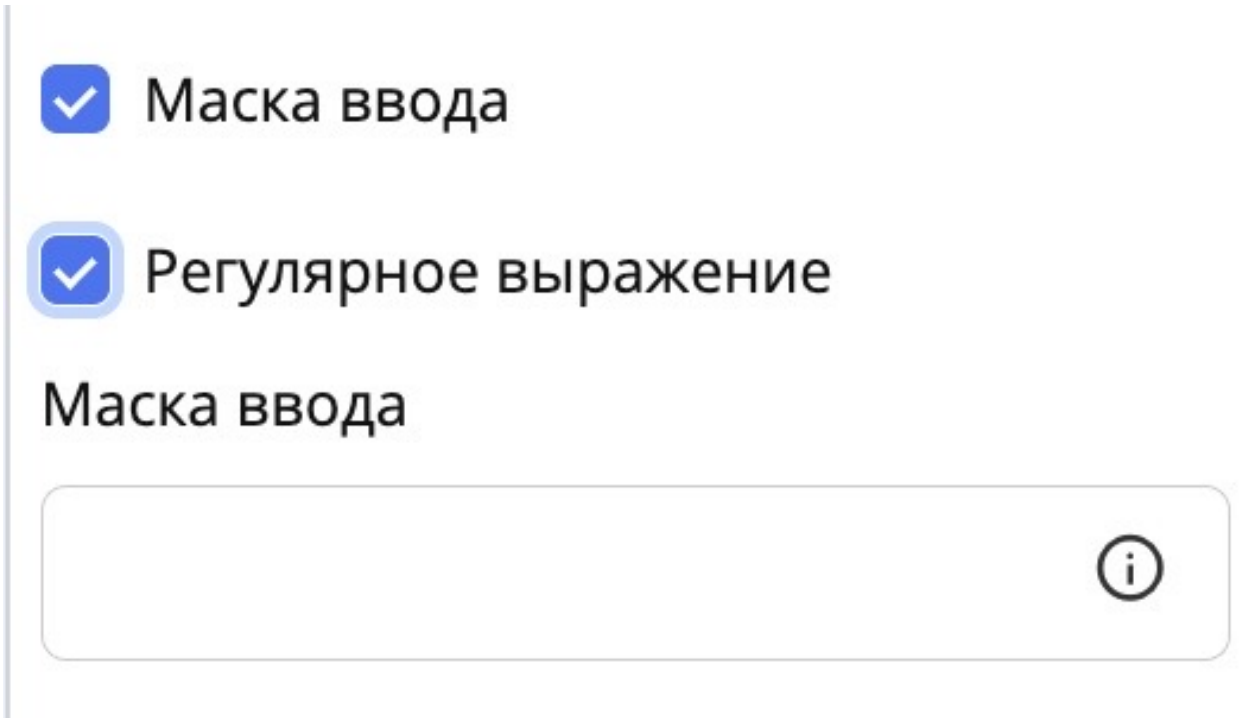
Шаг 2. После этого становится доступна дополнительная галочка **Регулярное выражение** - отмечаем её

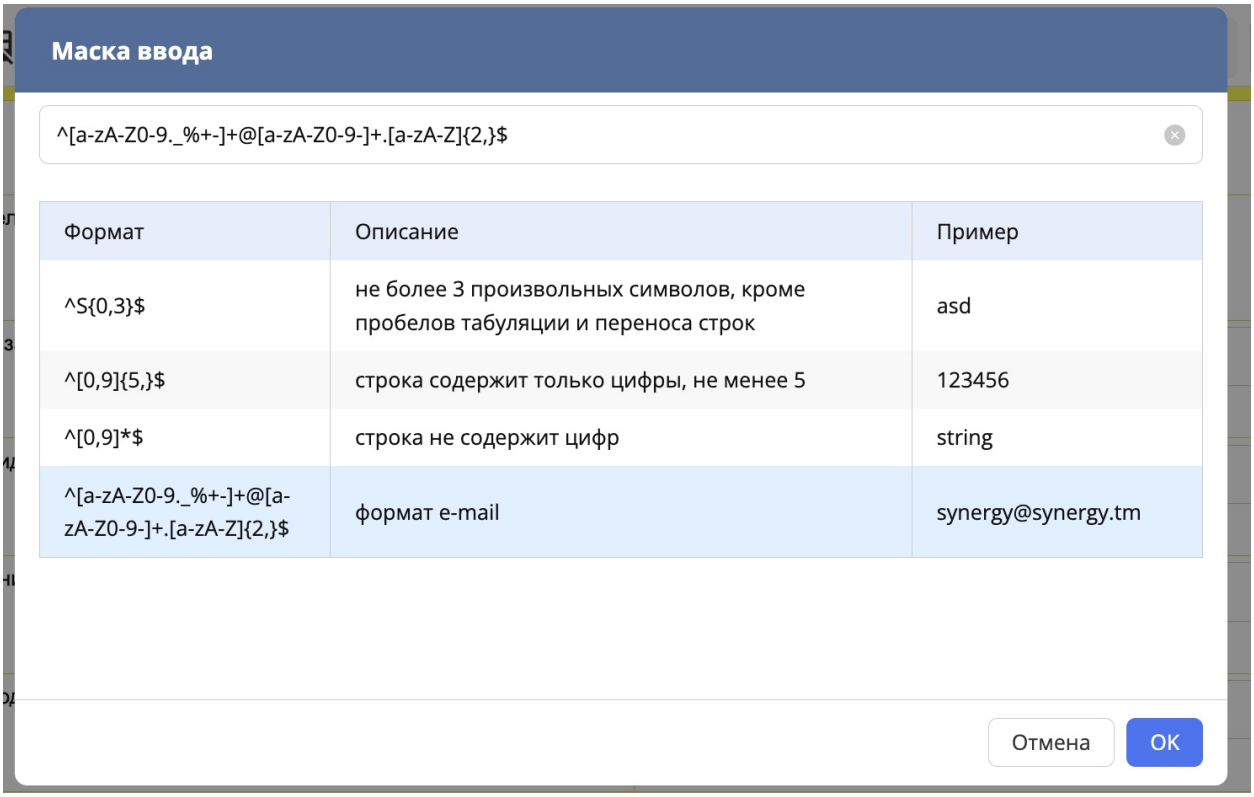
Шаг 3. Нажимаем на иконку настроек в поле **Маска ввода**

Шаг 4. В открывшемся окне выбираем готовый шаблон формата e-mail из списка доступных регулярных выражений.

Шаг 5. Нажимаем «Ок»

Примечание: При ручном редактировании регулярного выражения для применения изменений необходимо нажать Enter.


Маска ввода
Регулярное выражение
Маска ввода


Маска ввода
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9-]+.[a-zA-Z]{2,}$
Формат
Описание
Пример
^S{0,3}$
не более 3 произвольных символов, кроме пробелов табуляции и переноса строк
asd
^[0,9]{5,}$
строка содержит только цифры, не менее 5
123456
^[0,9]*$
строка не содержит цифр
string
^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9-]+.[a-zA-Z]{2,}$
формат e-mail
synergy@synergy.tm
Отмена
OK

3.3.9 Поле 10, 11. Регион и Вид услуги

Для полей «Регион» и «Вид услуги» нам необходим компонент **«Выпадающий список»**, добавляем его и настраиваем уже известным нам способом - аналогично полю «Статус заявки».

3.3.10 Поле 12. Тип подписки

Справочником, помимо выпадающего списка, может также являться список отдельных записей реестра. Для этого:

Шаг 1. Добавить компонент **«Ссылка на реестр»** из раздела «Специальные»

Шаг 2. Создать в системе реестр, который будет использоваться как справочный (к этому мы перейдем чуть позже)

3.3.11 Поле 13. Комментарий клиента

Поле **«Комментарий клиента»** предназначено для ввода развернутого текстового описания со стороны заявителя. В отличие от других текстовых полей, здесь предполагается возможность ввода нескольких строк текста с абзацами.

Для этого используется компонент «Многострочный текст», который позволяет вводить неограниченное количество текста и сохранять структуру комментария.

Шаг 1. В ячейку добавляем компонент «Многострочный текст»

Шаг 2. При необходимости включаем настройку:

«Не удалять пробелы в начале строки», если важно сохранить форматирование и структуру текста, вводимого пользователем.

Компонент «Многострочный текст» предназначен для работы с большим объёмом текста и содержит следующие настройки:

- **Обязательное поле** — делает поле обязательным для заполнения.
- **Заблокировать от изменений пользователем** — запрещает редактирование поля пользователем в режиме изменения заявки.
- **Шифровать данные** — шифрует введенные значения.
- **Не удалять пробелы в начале строки** — позволяет сохранить форматирование текста и структуру абзацев, не объединяя текст в одну сплошную строку.

3.3.12 Поле 14. Способ связи

Поле **«Способ связи»** предназначено для указания предпочтительных каналов связи с заявителем (например, телефон, электронная почта и т.д.).

По логике процесса данное поле должно позволять выбор нескольких значений одновременно, поэтому тип данных поля — чекбокс.

Для реализации такого поведения используется компонент **«Выбор вариантов»**.

Шаг 1. Рядом с названием поля «Способ связи» добавляем компонент **«Выбор вариантов»**

Шаг 2. Кликаем по компоненту и переходим в его свойства

Шаг 3. Открываем вкладку «Элементы»

Шаг 4. В выпадающем списке «Справочник»:

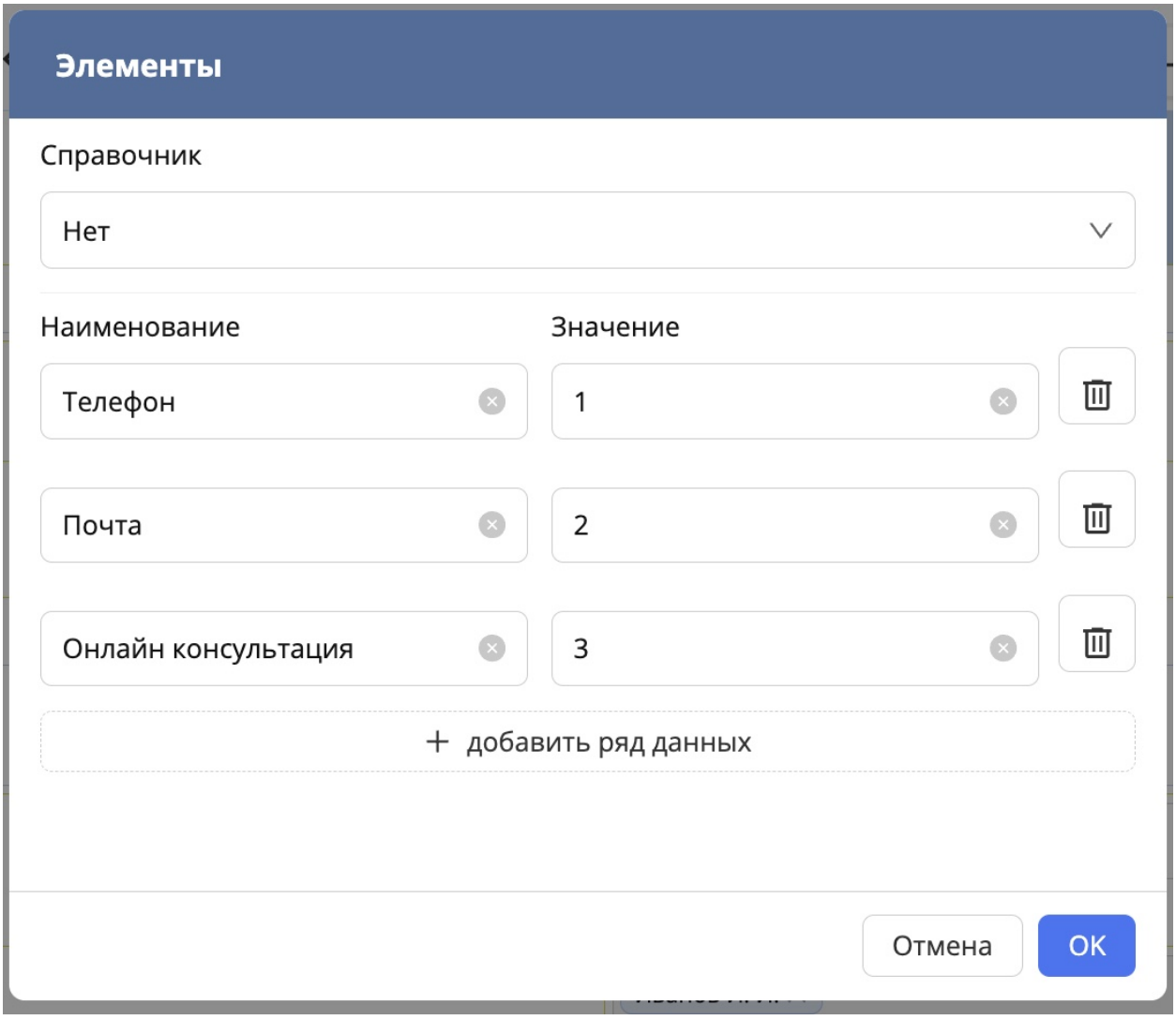
- выбираем существующий справочник со способами связи,

или

- оставляем значение «Нет» и добавляем варианты вручную.

Шаг 5. При ручном добавлении:

- указываем наименование каждого варианта;
- задаём значение (код, буквенный или числовой).



Шаг 6. После указания всех необходимых вариантов нажимаем «ОК».

Примечание: Для понимания логики выбора компонента важно различать элементы выбора: **Выпадающий список:** Представляет собой раскрывающийся список и позволяет выбрать только одно значение из предложенных вариантов.

Переключатель вариантов: Представляет собой видимый список вариантов и также позволяет выбрать только одно значение, переключаясь между ними.

Выбор вариантов: Представляет собой видимый список вариантов и позволяет выбрать несколько значений одновременно, отмечая их галочками.

3.3.13 Поле 15. Прикрепить документы (документы компании/ТЗ/отчеты и т.д)

Поле **«Прикрепить документы»** предназначено для загрузки одного или нескольких файлов, относящихся к заявке (документы компании, техническое задание, отчёты и другие вложения).

Согласно ордеру, данное поле должно позволять динамически добавлять несколько файлов, поэтому оно реализуется в виде динамической таблицы.

Для этого поле выносится в отдельный компонент «Таблица», внутри которого будет размещён компонент «Файл».

Шаг 1. Добавляем на форму компонент «Таблица»

Шаг 2. В данном случае ячейки таблицы можно не делить на колонки — достаточно одной колонки.

Шаг 3. В настройках таблицы включаем параметры:

- **«Добавлять строки в режиме заполнения»;**
- **«Добавить заголовок динамической таблицы».**

Шаг 4. В появившейся строке заголовка:

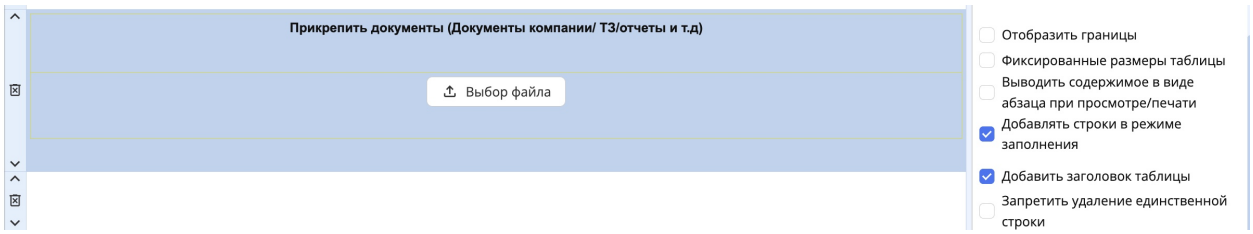
- вводим название поля «Прикрепить документы»;
- выравниваем текст по центру.

Шаг 5. Во вторую строку таблицы добавляем компонент **«Файл»** и также выравниваем его по центру.

Шаг 6. В настройках компонента «Файл» включаем:

- **«Открывать выбор файла с устройства»**,

так как у пользователей будет остуствовать доступ к хранилищу.



Шаг 7. В настройке **«Расширенные форматы файлов»** указываем допустимые форматы:

- DOCX
- PDF
- XLSX

Настройки компонента «Файл»

Компонент содержит следующие настройки:

- **Обязательное поле** — делает загрузку файла обязательной.
- **Заблокировать от изменений пользователем** — запрещает изменение файла при редактировании заявки.
- **Отображать полный путь к файлу при загрузке из хранилища** — отображает полный путь к файлу из хранилища проекта.
- **Отображать содержимое загруженного файла** — вместо ссылки отображает содержимое файла (актуально для изображений).
- **Открывать выбор файла с устройства** — ограничивает выбор источника только файлами с устройства.
- **Открывать выбор файла с хранилища** — ограничивает выбор источника только файлами из хранилища.
- **Позволять создание нового документа** — открывает модальное окно для создания нового файла.
- **Расширенные форматы файлов** — позволяет ограничить список допустимых форматов файлов.

См.также:

Дополнительную информацию касательно всех полей и их особенностей можно найти в официальной документации: http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/form/components.html

3.3.14 Результат этапа

По завершении данного этапа:

- все поля заявки добавлены на форму;
- каждому полю соответствует правильный тип компонента;
- форма готова к настройке форматных ограничений и логики отображения.

3.4 Сохранение формы и финальные штрихи

После добавления всех компонентов и полей формы необходимо сохранить выполненные изменения и выполнить завершающие настройки формы.

3.4.1 Сохранение формы

После завершения настройки формы необходимо сохранить изменения.

Для этого используется кнопка **«Сохранить»**, расположенная в верхней части редактора формы.

Важно: Рекомендуется сохранять форму регулярно в процессе работы, чтобы избежать потери внесенных изменений.

Рис. 14: Кнопка сохранения формы

3.4.2 Проверка корректности формы

После сохранения формы необходимо визуально проверить ее структуру.

Следует убедиться, что:

- все поля добавлены в соответствии со структурой ордера;
- названия полей отображаются корректно;
- поля расположены логично и удобно для заполнения;
- отсутствуют пустые или лишние элементы.

На данном этапе важно проверить форму именно с точки зрения пользователя, который будет создавать заявку.

3.4.3 Финальные штрихи

Перед переходом к следующему этапу рекомендуется обратить внимание на несколько важных моментов.

Коды компонентов

Каждый компонент формы имеет собственный код.

Для удобства дальнейшей работы рекомендуется задавать понятные и осмысленные коды компонентов.

Код компонента задается в его свойствах и используется при настройке логики формы и бизнес-процесса.

Визуальное оформление

При необходимости можно скорректировать:

- выравнивание текста;
- шрифт и размер;
- отступы внутри ячеек таблицы.

Эти настройки не влияют на логику работы формы, но повышают удобство ее использования.

3.4.4 Результат этапа

По завершении данного этапа:

- форма полностью настроена;
- все изменения сохранены;

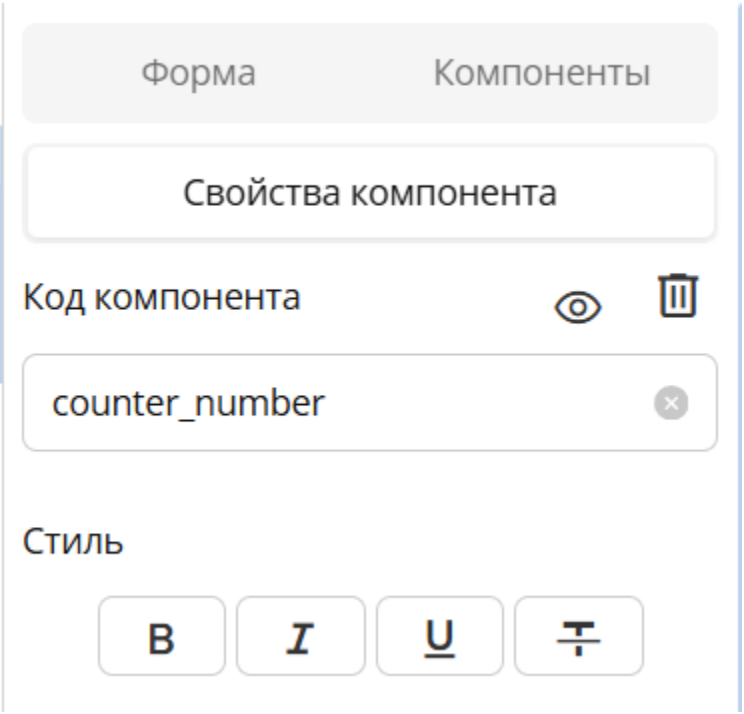


Рис. 15: Код компонента в свойствах

- форма готова к использованию;
- можно переходить к настройке логики поведения формы.

В следующем этапе будет выполнена настройка условных действий, которые управляют отображением полей в зависимости от введенных данных.

3.5 Условные действия

После сохранения формы и приведения кодов компонентов в порядок можно переходить к настройке условных действий.

Условные действия используются для управления отображением и поведением элементов формы в зависимости от значений, которые вводит пользователь.

Примечание: Условные действия не являются отдельной сущностью приложения и привязаны к конкретной форме.

3.5.1 Переход к разделу «Условные действия»

1. Откройте ранее созданную форму.
2. На левой панели выберите вкладку **«Условные действия»**.

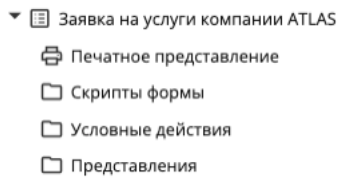


Рис. 16: Раздел «Условные действия» формы

3.5.2 Создание нового условного действия

Для добавления условного действия выполните следующие шаги:

Шаг 1. Нажмите правой кнопкой мыши по папке **«Условные действия»**

Шаг 2. Выберите **«Добавить группу действий»**.

Шаг 3. В открывшемся окне латинскими буквами укажите наименование условных действий.

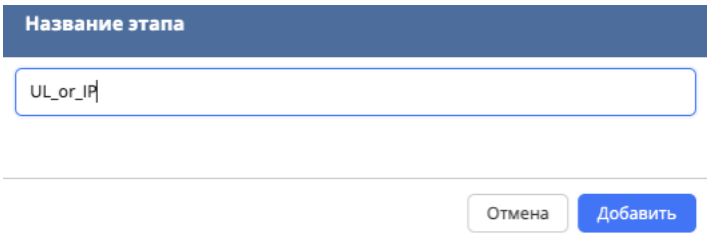


Рис. 17: Создание нового условного действия

Шаг 4. В открывшейся странице нажимаем кнопку «+» для добавления условных действий

Каждое условное действие состоит из двух логических частей:

- условия (когда выполняется);
- действий (что происходит при выполнении условия).

3.5.3 Логика условия

В первую очередь необходимо указать условие, при котором будет срабатывать правило.

Шаг 1. В поле выбора компонента укажите компонент формы, значение которого будет проверяться.

Шаг 2. Выберите тип условия (равно, не равно и т.д.).

Шаг 3. Укажите значение, при котором будет соблюдаться условие.

В рамках текущего процесса в качестве условия используется поле **«Тип заявителя»**.

Пример логики условия:

Что требуется по ордеру:

- поле **«БИН организации»** должно отображаться только при выборе типа заявителя **«ЮЛ»**
- поле **«ИИН индивидуального предпринимателя»** должно отображаться только при выборе типа заявителя **«ИП»**

Как это работает через условные действия:

Если Тип заявителя = **«ЮЛ»**

То показать поле **«БИН организации»** и скрыть поле **«ИИН индивидуального предпринимателя»**

Если Тип заявителя = **«ИП»**

То показать поле **«ИИН индивидуального предпринимателя»** и скрыть поле **«БИН организации»**

3.5.4 Настройка условия

После нажатия на кнопку «+» на странице нам открывается возможность настроить нужные нам условные действия

Шаг 1. В строке **«Если»** выбираем поле, которое будет являться условием для срабатывания действия. В нашем случае это **«Тип заявителя»**, выбираем из списка код этого поля.

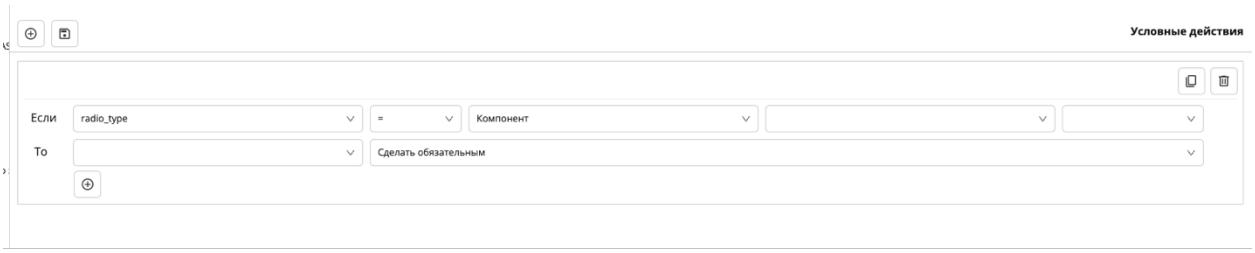


Рис. 18: Настройка условия

Шаг 2. Теперь нам необходимо указать при выборе какого значения мы будем производить действие с полем. В нашем случае пользователь должен отметить тип заявителя **«ЮЛ»**, значит обращаемся к справочнику поля **«Тип заявителя»**

В нашем случае, вариант **«ЮЛ»** = значению **«1»**

Значит в условных действиях выбираем операнд «=», а в следующем поле тип компонента выбираем **«Текстовое значение»**

И в открывшемся правом поле указываем нужное нам значение справочника.

3.5.5 Настройка действия

В строке **«То»** указываем с каким полем и что именно должно произойти

В нашем случае, нужно отобразить поле **«БИН организации»** и при этом скрыть поле **«ИИН индивидуального предпринимателя»**.

Значит в строке **«То»** выбираем из списка код поля **«БИН организации»** - в нашем случае это **«textbox_bin»** и выбираем из списка действие, которое должно произойти с полем. В нашем случае **«Отобразить»**

Для того чтобы добавить еще одно действие к тому же условию, под строкой **«То»** нажимаем на кнопку добавить «+»

И аналогичным образом указываем, что уже поле **«ИИН индивидуального предпринимателя»** должно быть скрыто.

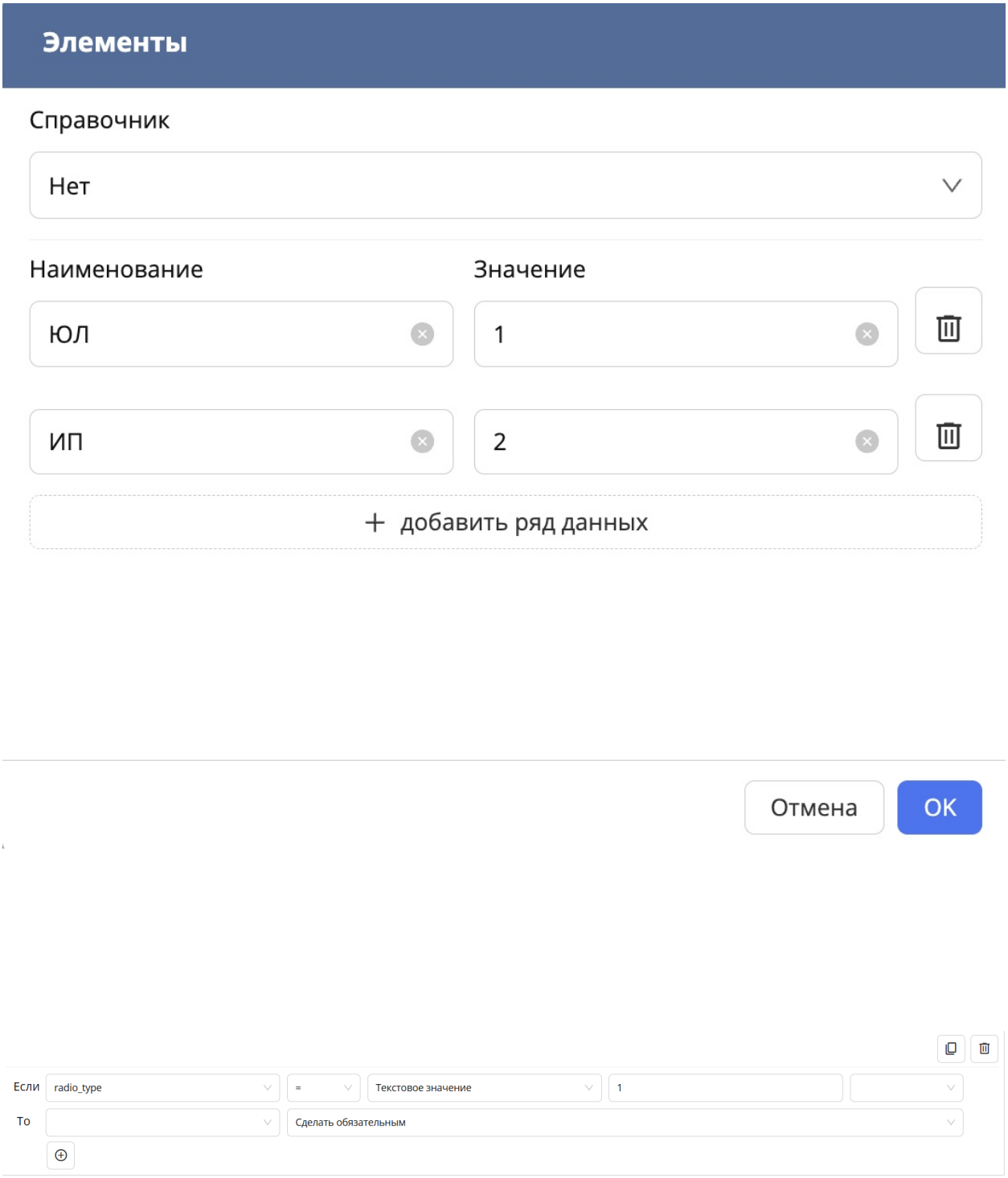


Рис. 19: Настройка условия

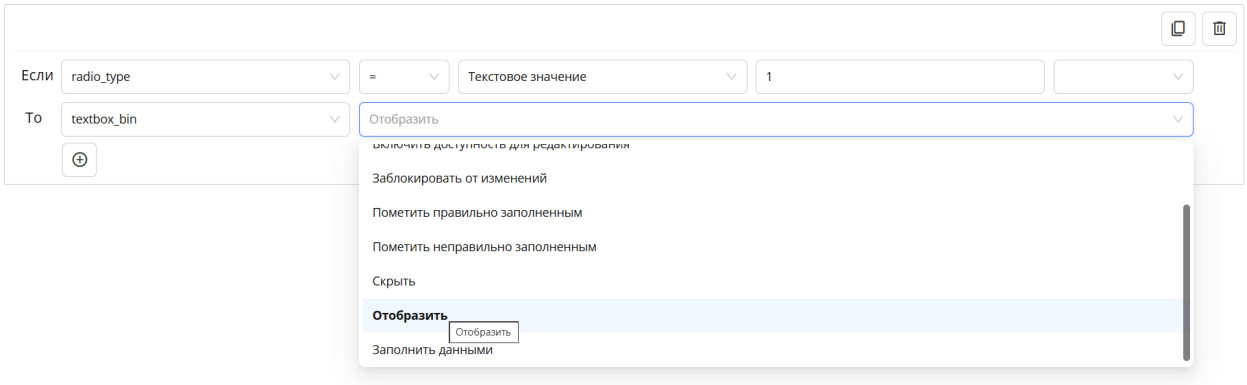


Рис. 20: Настройка действия

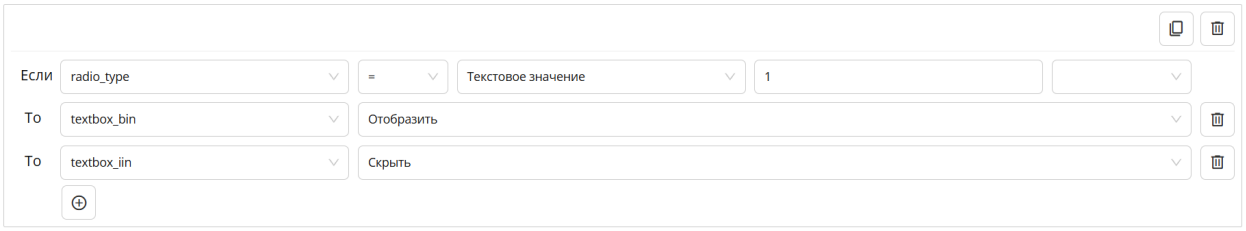
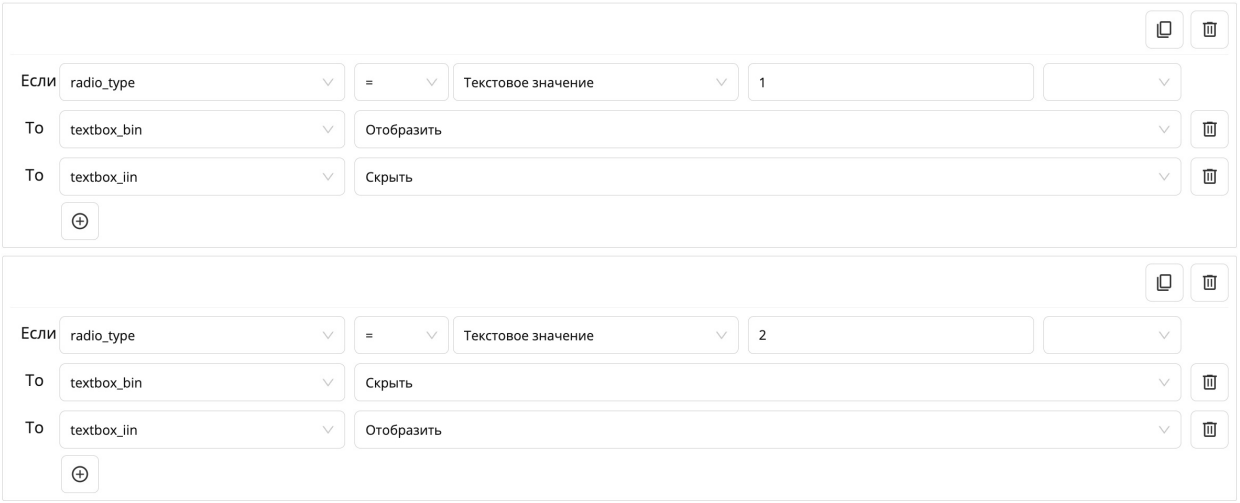


Рис. 21: Первое условное действие

Первое условное действие готово, теперь нам необходимо настроить зеркальное действия для второго варианта справочника, когда выбран Тип заявителя **«ИП»**.

Шаги создания второго условного действия аналогичны созданию первого варианта, но с другими условиями.



Аналогично необходимо настроить **Названия полей** (надписи)

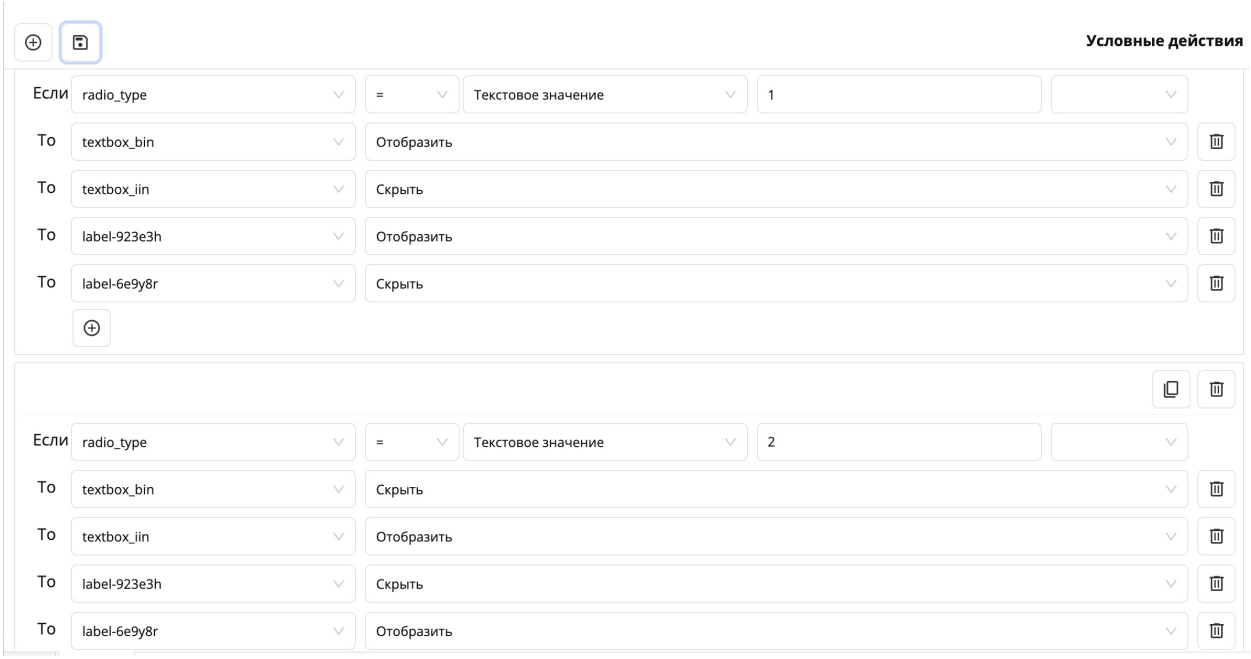
Шаг 1. В первый блок условий в раздел «То»

- добавить код поля в котором находится надпись «БИН организации» и выбрать действие «Отобразить»

- добавить код поля в котором находится надпись «ИИН индивидуального предпринимателя» и выбрать действие «Скрыть»

Шаг 2. Аналогично, во втором блоке условий настроить зеркально

- добавить код поля в котором находится надпись «БИН организации» и выбрать действие «Скрыть»
- добавить код поля в котором находится надпись «ИИН индивидуального предпринимателя» и выбрать действие «Отобразить»



Ожидаемый результат:

При выборе Типа заявителя «ЮЛ»:

- скрывается строка «ИИН индивидуального предпринимателя» и поле для ввода ИИН
- отображается строка «БИН Организации» и поле для ввода БИН

При выборе Типа заявителя «ИП»:

- скрывается строка «БИН Организации» и поле для ввода БИН
- отображается строка «ИИН индивидуального предпринимателя» и поле для ввода ИИН

3.5.6 Проверка работы условных действий

После настройки условных действий необходимо проверить их работу.

Для этого:

1. Перейдите в режим редактирования формы нажав на значок редактирования в панеле редактора

2. Измените значение поля, участвующего в условии.
3. Убедитесь, что соответствующие поля отображаются или скрываются корректно.

Если логика работает ожидаемым образом, условные действия настроены корректно.

3.5.7 Результат этапа

По завершении данного этапа:

- форма адаптируется под разные сценарии заполнения;
- пользователь видит только релевантные поля;
- логика заполнения заявки соответствует требованиям ордера.

После настройки условных действий можно переходить к следующему этапу - созданию реестра для хранения заявок.

Глава 4

Реестры

4.1 Общая информация о реестрах

После создания формы бизнес-процесса необходимо определить, где будут храниться документы, созданные по этой форме, и в каком виде пользователь будет работать с ними в системе.

Для этих целей используется сущность **«Реестр»**.

Реестр представляет собой структурированный каталог записей, созданных по форме, с возможностью:

- просмотра списка документов;
- фильтрации и сортировки;
- разграничения доступа между пользователями;
- запуска маршрутов бизнес-процесса.

Каждая запись в реестре соответствует одному документу, созданному на основе формы.

4.1.1 Роль реестра в бизнес-процессе

Реестр является центральной точкой работы с документами. Именно через него пользователь:

- создает новые заявки;
- отслеживает ранее созданные документы;
- взаимодействует с маршрутом обработки;
- получает доступ к данным в рамках своих прав.

Корректная настройка реестра напрямую влияет на удобство работы пользователей и корректность бизнес-процесса.

4.2 Создание реестра

После создания формы следующим шагом является создание реестра, который будет хранить документы по данной форме.

4.2.1 Переход к созданию реестра

Для создания реестра необходимо:

1. В структуре приложения выбрать папку, в которой будет располагаться реестр (рекомендуется использовать ту же папку, где находится форма).
2. Кликнуть правой кнопкой мыши.
3. В контекстном меню выбрать: **Добавить** → **Процессы** → **Реестр**.

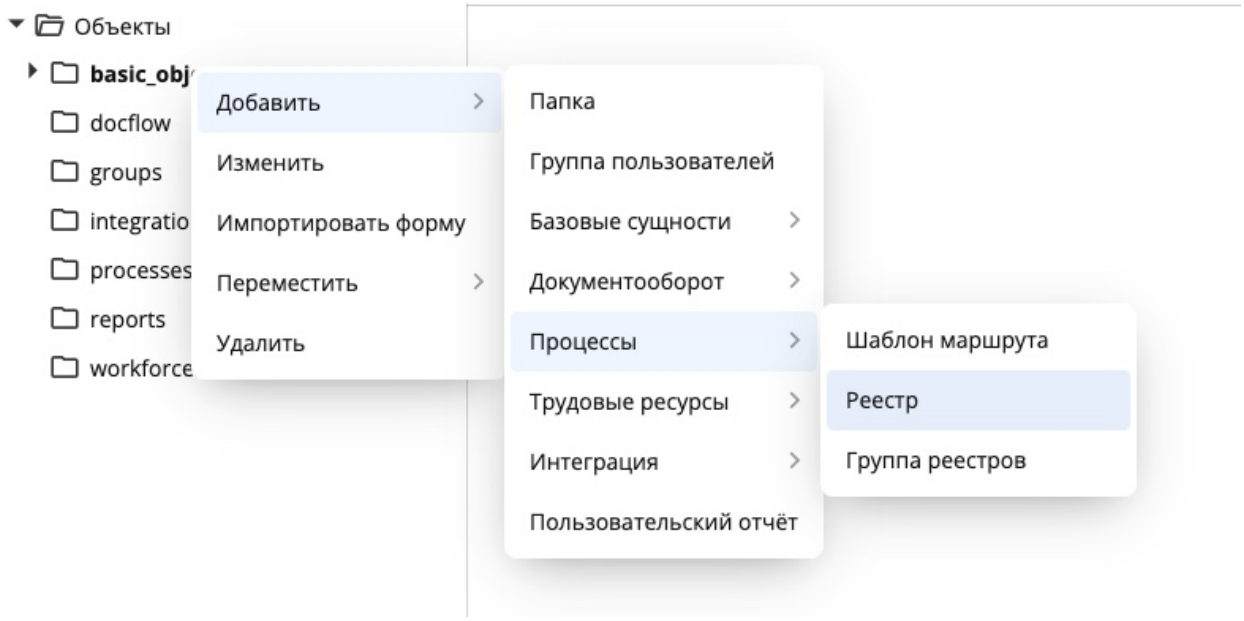


Рис. 1: Создание реестра

После выполнения указанных действий откроется созданный реестр, который необходимо будет настроить.

4.3 Основные настройки реестра

В окне создания реестра необходимо заполнить основные параметры, которые определяют связь реестра с формой и поведение бизнес-процесса.

4.3.1 Заполнение основных полей

В процессе создания реестра указываются следующие параметры:

- **Наименование реестра** - задается в соответствии с логикой бизнес-процесса.

- **Код** - может быть задан вручную. Либо оставлен автоматически сформированный транслит.
- **Форма документа** - выбирается форма, по которой будут создаваться записи. При большом количестве форм рекомендуется использование поиска.
- **Название действия по документу** - текст кнопки, по нажатию на которую запускается бизнес-процесс (например: «Создать», «Отправить»).

Рис. 2: Основные параметры реестра

Таким образом реестр связывается с конкретной формой и получает точку входа для запуска маршрута.

Примечание: Реестр, помимо основных, также имеет в себе следующие настройки:

- «Тип документа» не обязательно для заполнения. Содержит типы документов текущего приложения в исходящих и внутренних журналах. Необходимо заполнять при наличии в маршруте типа работы «регистрация».
- «Сортировать по» позволяет настроить сортировку записей реестра по двум вариантам: «Дате создания» (используется по умолчанию). «Полю реестра» отображает доступные поля формы, по которым будет произведена сортировка. (кроме «Динамическая таблица», «Выбор вариантов», «Ссылка на документ», «Свойства документа» и «Ссылка на файл в хранилище»)
- «Направление сортировки» позволяет настроить сортировку по возрастанию или убыванию выбранного значения.
- «Название действия по документу» название кнопки для запуска маршрута (отправить, создать, сохранить и т.д). Поле является обязательным.
- «Отображать документы по реестру в разделе «Мои»» определяет, будет ли добавляться документ, созданный пользователем по реестру, в разделе «Все» и «Мои» документов этого пользователя.
- «Закрыть окно документа после отправки на активацию»
- «Реестр ответа» Настройка является необязательной. Указывается, какой реестр будет использоваться для ответа на документ данного реестра. Реестр для ответа выбирается из общего списка реестров текущего приложения.

- «Название действия по реестру ответа» для названия предпочтительной («зеленой») кнопки действия с документом. Поле доступно только для выбранного реестра ответа. Значение поля по умолчанию - «Создать ответ»
- «Создавать корневую работу для маршрута по реестру» это настройка, которая включает создание одной общей “главной работы” для записи реестра, когда по ней запускается маршрут (активация, изменение, удаление).

4.4 Поля и колонки реестра

После создания реестра необходимо определить, какие поля формы будут отображаться в списке документов.

Если компонентам формы были заранее присвоены корректные коды, это существенно упрощает настройку полей реестра.

4.4.1 Назначение настроек полей

Настройки полей реестра определяют:

- какие колонки отображаются в списке;
- порядок их расположения;
- наименования колонок;
- какие поля считаются значащими.

4.4.2 Порядок отображения колонок

Поле **«№ п/п»** определяет очередность отображения колонок.

Принцип работы простой:

- значение **1** — колонка отображается первой;
- значение **2** — второй и т.д.;
- если значение не указано, порядок определяется системой автоматически.

Это позволяет гибко управлять визуальным представлением реестра.

4.4.3 Отображение и наименование

- **Отображать колонку** — определяет, будет ли поле видно в списке реестра. По умолчанию все флажки выключены. Для сохранения реестра необходимо включить хотя бы один.
- **Наименование колонки** — задает отображаемое название поля, понятное пользователю.

4.4.4 Значащее содержимое

Настройка **«Значащее содержимое»** определяет, является ли значение поля важным для системы.

Если параметр включен, значение поля:

- отображается в кратком содержании записи;
- участвует в поиске;
- используется в названиях работ при запуске маршрутов.

Поля формы реестра

№ п/п	Отображать колонку	Идентификатор	Наименование колонки	Значащее содержимое	Примечание
	☐	textbox_name_organization	Введите значение	☐	-
	☐	textbox_head	Введите значение	☐	-
	☐	entity_manager	Введите значение	☐	-
	☐	textbox_phone	Введите значение	☐	-
	☐	textbox_iin	Введите значение	☐	-
	☐	entity_responsible	Введите значение	☐	-
	☐	counter_number	Введите значение	☐	-
	☐	textbox_bin	Введите значение	☐	-
	☐	listbox_region	Введите значение	☐	-

Рис. 3: Настройка полей реестра

После этого, созданный реестр можно сохранить, нажав на иконку сохранения на панели реестра.

Рис. 4: Кнопка сохранения

4.5 Права доступа к реестру

После настройки полей необходимо определить, какие пользователи и группы будут иметь доступ к реестру.

Для этого используется вкладка **«Права на реестр»**.

4.5.1 Назначение прав доступа

Права доступа определяют, кто может создавать, просматривать, изменять и удалять записи реестра.

Настройка прав выполняется на уровне:

- групп пользователей;

- департаментов;
- подразделений;
- отдельных организационных единиц.

4.5.2 Виды прав

Реестр поддерживает следующие типы прав доступа:

- **«Все»** - наличие всех либо частичных прав на реестр
- **«Просмотр списка»** - просмотр списка записей по форме реестра
- **«Просмотр данных»** - просмотр конкретных записей по форме реестра
- **«Создание»** - создание записей по форме реестра
- **«Редактирование»** - изменение записей по форме реестра (для всех статусов, кроме «Активная»)
- **«Изменение»** - изменение записи со статусом «Активная» (и запуск соответствующего маршрута)
- **«Удаление»** - удаление записей по форме реестра

Примечание:

- Право просмотра данных автоматически предусматривает право просмотра списка.
- Права на редактирование / изменение / удаление автоматически предусматривают права на просмотр списка и данных.

4.5.3 Группы пользователей

Для удобства управления доступом рекомендуется использовать **группы пользователей**.

Группы позволяют:

- централизованно управлять правами;
- назначать доступ сразу нескольким пользователям;
- упростить дальнейшее сопровождение системы.

Перед назначением прав на реестр необходимо создать хотя бы одну группу пользователей.

4.5.4 Создание группы пользователей

Для создания группы пользователей выполните следующие шаги:

1. В дереве проекта выберите папку, в которой будут храниться группы пользователей.
2. Кликните правой кнопкой мыши по папке.
3. В контекстном меню выберите: **Добавить** → **Группа пользователей**.

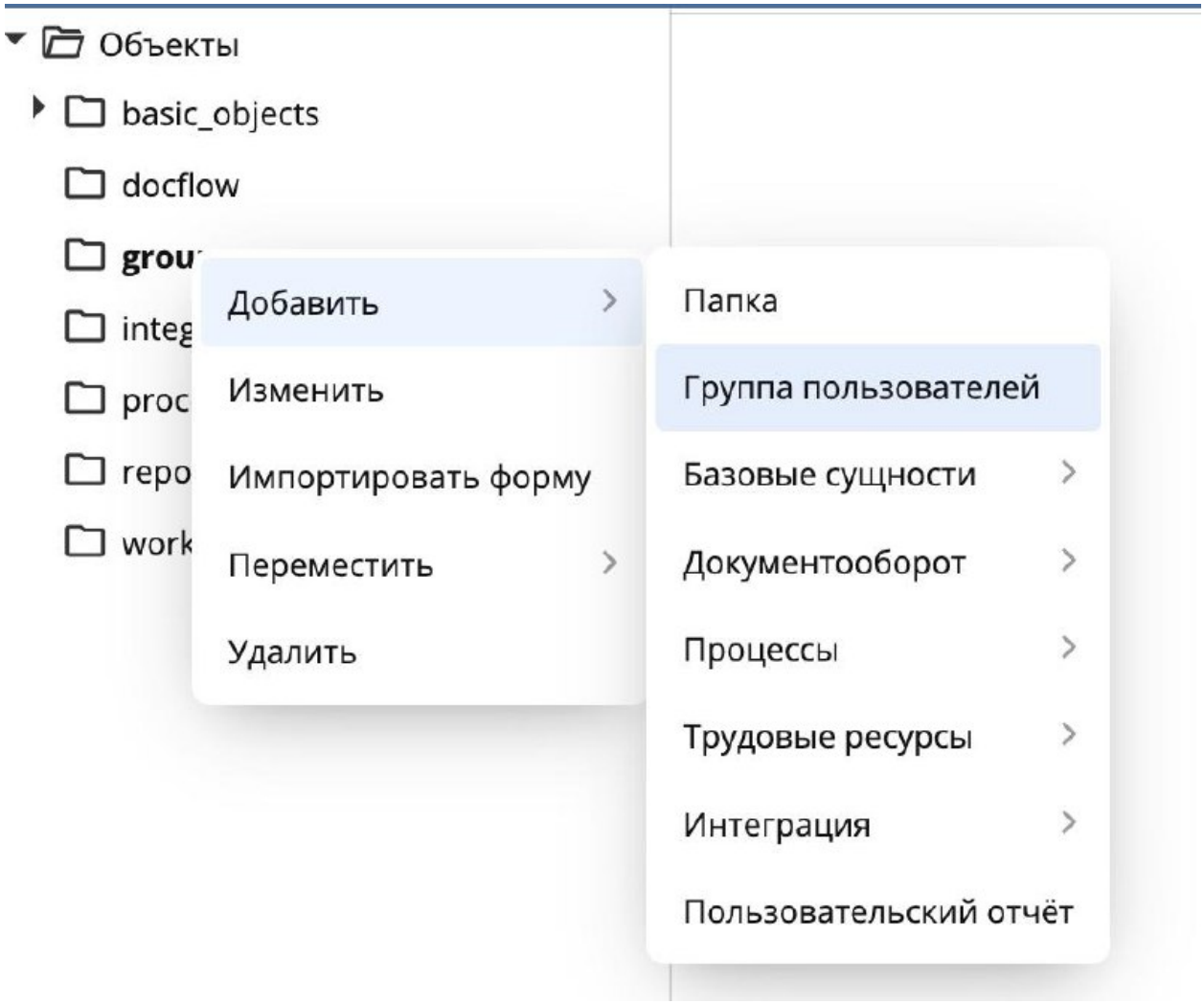


Рис. 5: Создание группы пользователей

В открывшемся окне:

1. Укажите **Наименование группы пользователей**.
2. В поле **«Пользователи»** выберите зарегистрированных в системе пользователей, которые будут входить в данную группу.
3. Сохраните группу.

После сохранения группа станет доступна для назначения прав на реестры и другие сущности системы.

4.5.5 Назначение прав группе пользователей

После создания групп можно перейти непосредственно к настройке прав доступа к реестру.

Для этого:

1. Откройте реестр и перейдите во вкладку **«Права на реестр»**.
2. Нажмите кнопку «+» для добавления доступа.
3. В открывшемся окне выберите:
   - группу пользователей;
   - либо департамент / подразделение.
4. Отметьте галочками необходимые права.
5. Сохраните изменения.

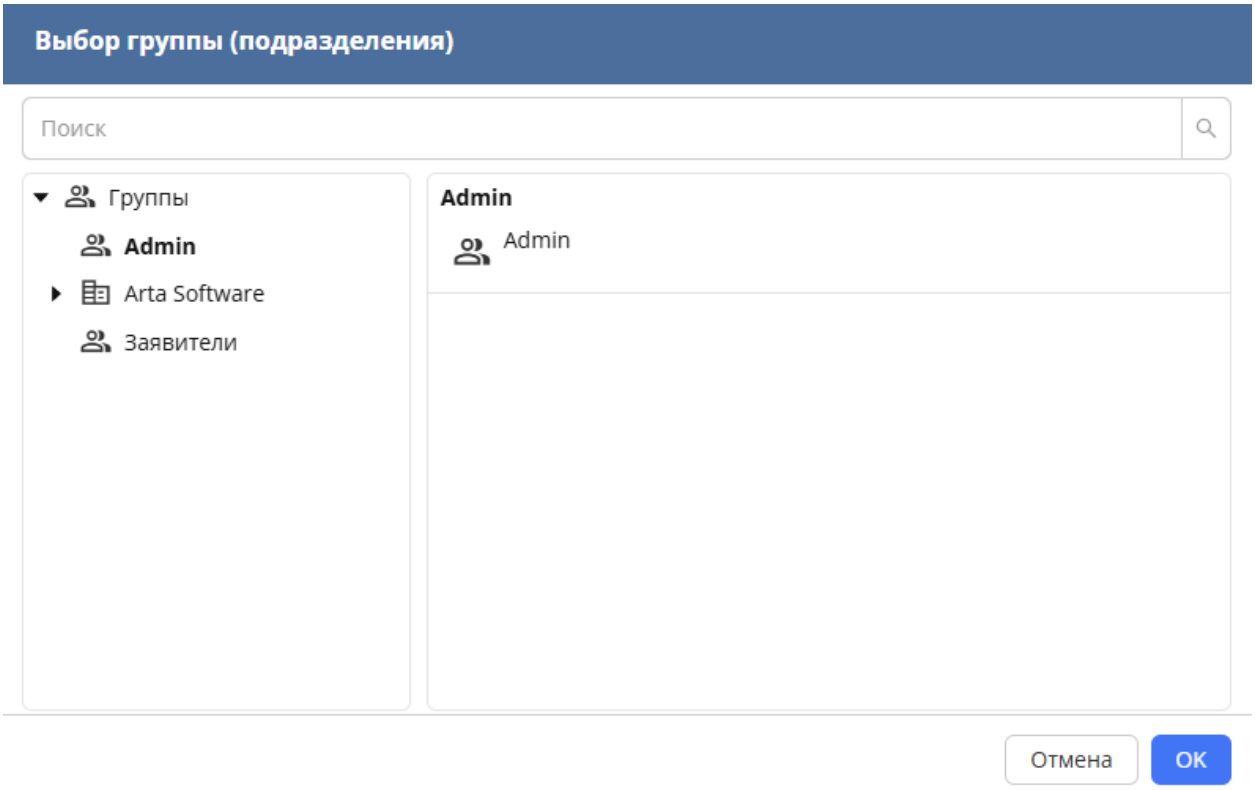


Рис. 6: Выбор группы пользователей(подразделения)

4.5.6 Пример настройки прав

Например:

- группе **Admin** можно назначить все права на реестр: просмотр, создание, редактирование, изменение и удаление;
- группе **Заявители** можно предоставить: * создание записей; * просмотр списка; * просмотр данных.

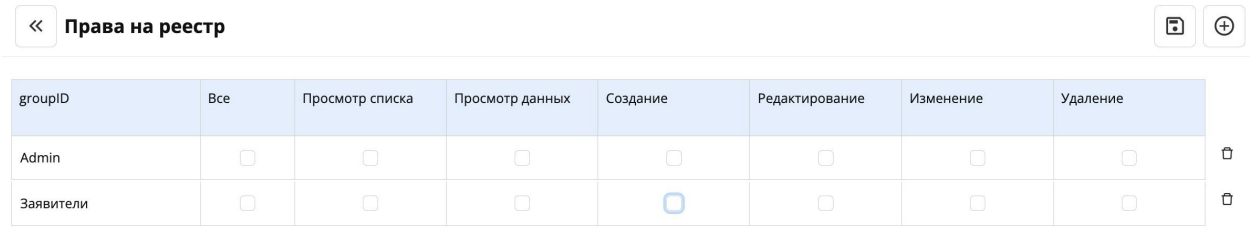


Рис. 7: Назначение прав группе пользователей

Следует учитывать, что на данном этапе права применяются ко всем записям реестра. Ограничение доступа только к «своим» записям настраивается отдельно с помощью фильтров.

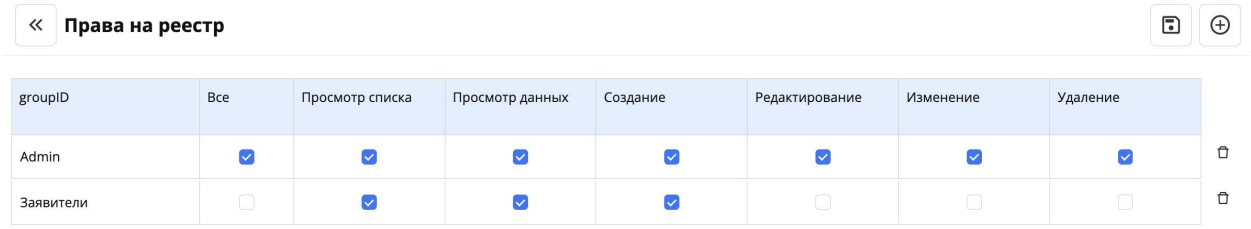


Рис. 8: Пример настройки прав доступа

4.5.7 Завершение настройки

После назначения прав:

1. Нажмите кнопку сохранения в правом верхнем углу.
2. Для перехода к следующей вкладке вернитесь назад с помощью стрелки **«Назад»**.

На этом настройка прав доступа к реестру завершена.

Глава 5

Маршруты

5.1 Маршруты реестра: назначение и различия

В рамках одного реестра может существовать несколько маршрутов. Тип маршрута определяет, в каком случае и по какому сценарию будет обрабатываться заявка.

Для добавления маршрута необходимо перейти во вкладку **«Маршруты»**

по нажатию на кнопку «+» нам открывается список возможных видов маршрута.

5.1.1 Виды маршрутов

Активация

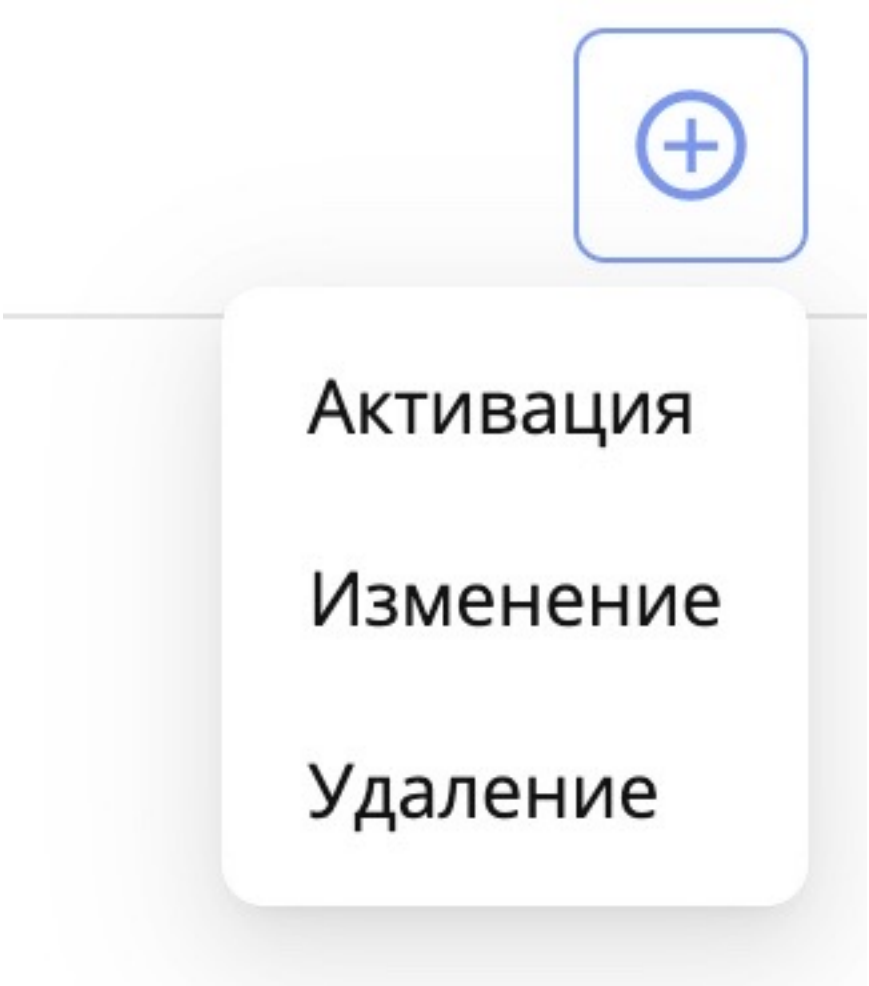
Определяет путь, по которому заявка проходит **после создания и первичной активации**.

Изменение

Определяет маршрут заявки **после внесения изменений** в уже активированную (завершённую) заявку.

Удаление

Определяет маршрут заявки **при попытке её удаления**. Используется, например, для согласования удаления с руководителем или отправки уведомлений о попытке удаления записи.


Активация
Изменение
Удаление

Для реестра может быть настроено несколько маршрутов, каждый из которых отрабатывает в своём контексте: при создании, изменении или удалении записи.

5.2 Маршрут «Активация»: интерфейс и логика настройки

На этом этапе мы настраиваем путь заявки при её подаче, то есть маршрут, по которому заявка пойдёт после создания и активации.

Для этого используется маршрут типа **«Активация»**.

Основная рабочая область.

Редактор маршрутов состоит из трёх панелей, каждая из которых отвечает за свой тип этапов маршрута.

Панели редактора маршрутов

Предварительные этапы - содержат только стандартные процессы, которые выполняются до основных действий:

- Работа
- Согласование
- Утверждение
- Ознакомление
- Отправка документа
- Регистрация
- Маршрут

Действия - ключевая часть маршрута.

Этапы в этой панели:

- могут автоматически получать данные из формы (по кодам компонентов);
- изменяют объекты и состояние процесса в системе;
- выполняются только после успешного завершения предварительных этапов (если они есть).

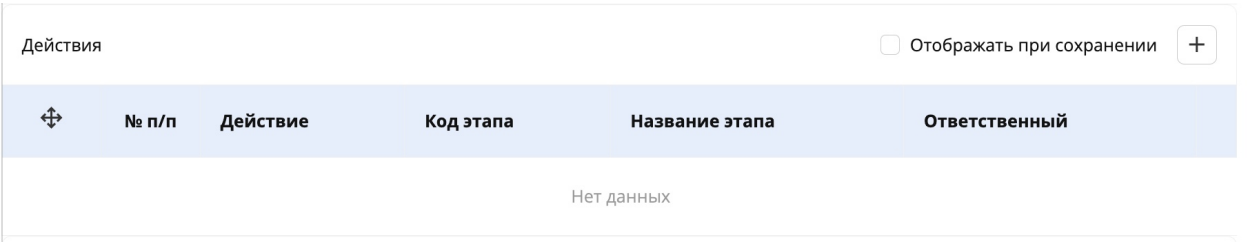
Последующие этапы - также содержат только стандартные процессы и выполняются после действий (при необходимости).

5.2.1 Переход к созданию маршрута

Шаг 1. По нажатию на кнопку «+» в разделе **«Маршруты реестра»**, в выпаждающем списке выбрать тип маршрута **«Активация»**

Шаг 2. В открывшемся окне перейти к редактированию маршрута активации нажав кнопку редактирования:

Шаг 3. На панеле «Действия» нажимаем кнопку «+» чтобы добавить новый этап



Шаг 4. В открывшейся справа панели настройки этапа необходимо настроить этап в соответствии с ордером.

Каждый этап имеет свой результат выполнения (подписание, ввод комментария, автоматическое системное действие и другие операции, необходимые для бизнес-процесса)

Для того, чтобы понять какой именно тип действия нам необходим на каждом этапе, мы обращаемся к пункту ордера **2. Диаграмма процесса и шаги процесса**

№	Шаг процесса	Вход операции	**Выход операции**	**Роль**
1	Заполнение заявки	Разработанная в системе форма заявки с данными клиента	Заполненная по форме заявка на услугу	Клиент
3	Отправка заявки	Подписание заявки с использованием ЭЦП	Сформированная и подписанная в системе заявка на услугу	Клиент
4	Регистрация заявки	Сформированная и подписанная в системе заявка на услугу	Зарегистрированная в системе заявка на услугу с присвоенным регистрационным номером	Система
5	Отправка уведомления о номере заказа	Зарегистрированная в системе заявка на услугу с номером заказа	Отправка уведомления на почту клиента «Ваша заявка на ‘название услуги’ зарегистрирована под номером ‘номер заказа’»	Система

4

Для того чтобы правильно построить бизнес-логику процесса, нам необходимо обращать внимание на вход и выход шага, и роль, которая его выполняет

5.2.2 Этап 1. Заполнение заявки

Вход - форма заявки, разработанная в системе и данные клиента, которые он вносит при заполнении формы.

Выход - заполненная по форме заявка на услугу.

На этом этапе не нужен дополнительный шаг процесса, т.к отправка заявки и будет являться первым шагом процесса.

5.2.3 Этап 2. Отправка заявки.

Вход - подписание заявки при помощи ЭЦП.

Выход - заявка подписанная в системе.

Для реализации данного этапа нам потребуется тип действия **«Работа по форме»** с типом **«Согласование»**

Данный тип этапа требует указания поля из заявки, в котором будет находится текущий пользователь, создающий заявку.

Предварительная настройка

Шаг 1. Возвращаемся в форму заявки

Шаг 2. Добавляем на форму компонент таблицы, и включаем для нее настройки скрытности, чтобы системные поля не отображались пользователю

Шаг 3. Внутрь таблицы добавляем компонент **«Объекты Synergy»**

- задаем ему код например «entity_author»
- включаем настройку **«Заполнять текущим пользователем»**

Шаг 4. Полю задаем наименование «Автор»

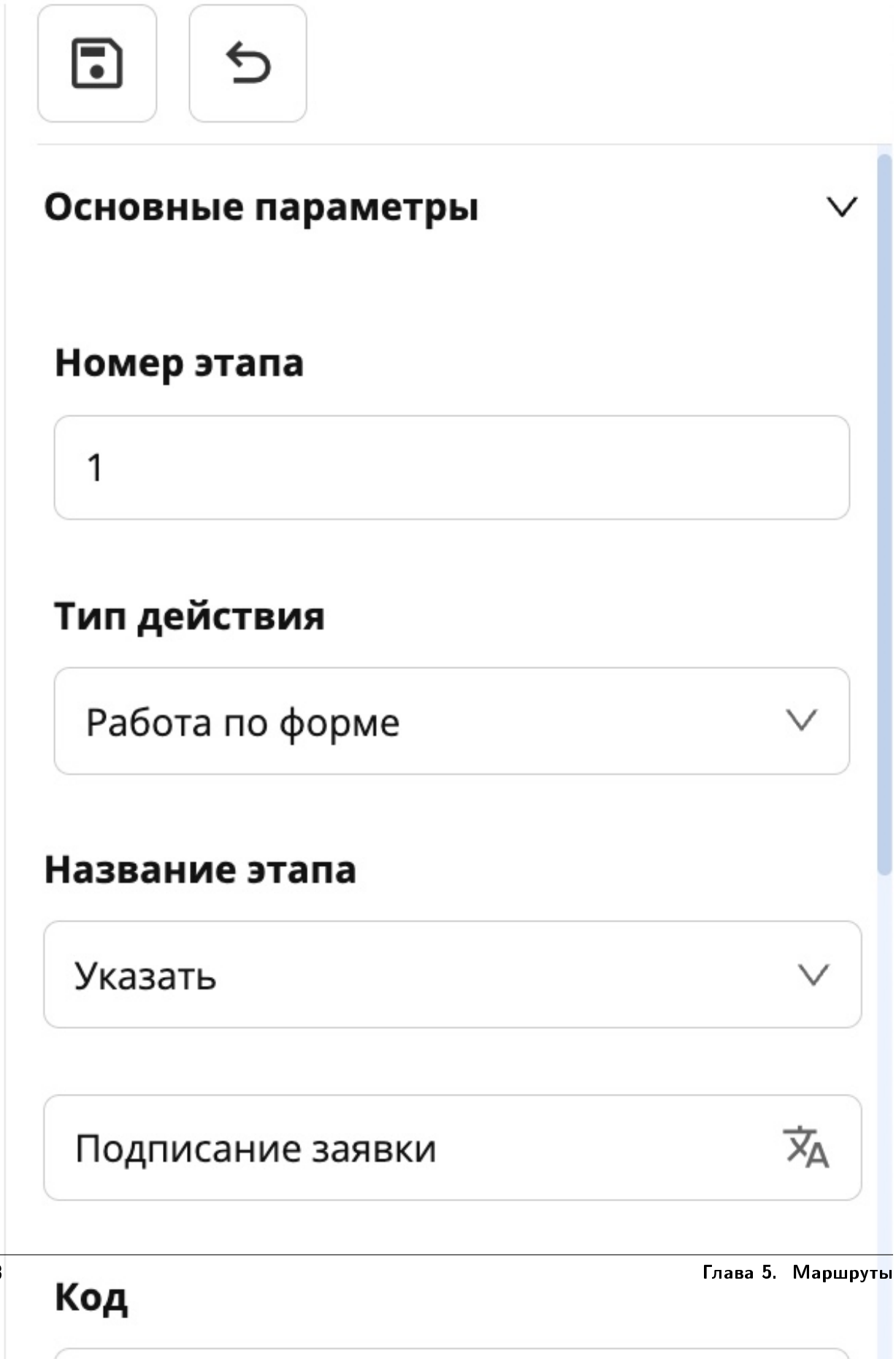
Шаг 5. Сохраняем изменения

В результате этого, поле автоматически будет заполняться учетной записью пользователя, создавшего заявку.

Создание этапа маршрута

Шаг 1. В настройках этапа указываем

- Тип действия - «Работа по форме»
- Название этапа - указываем согласно логике шага, например «Подписание заявки»
- Код этапа - необязательно, используется при обращении к этапу в скриптах или условиях.
- Ответственный - здесь необходимо указать код поля в котором будет находится автор заявки (entity_author) его мы создали в предварительном шаге
- Тип работы - согласование
- Возврат - в этом поле указывается этап маршрута, к которому необходимо вернуться в случае отказа в согласовании (при необходимости). В нашем случае может оставить пустым.


Основные параметры
Номер этапа
1
Тип действия
Работа по форме
Название этапа
Указать
Подписание заявки
Код

Шаг 2. Сохраняем этап, нажав на иконку дискеты в панели настроек этапа.

Примечание: Дополнительно о том как подключить модуль подписания ЭЦП http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/signing.html

---

5.2.4 Этап 3. Регистрация заявки

Вход - подписанная в системе заявка

Выход - заявка со сформированным регистрационным номером

Формирование номера заявки в системе осуществляется на основе **счётчика**.

Примечание: **Счётчик** — это базовая сущность системы, предназначенная для генерации последовательных числовых значений. Он используется для нумерации заявок, документов и других объектов.

Создание счетчика

Шаг 1. В дереве приложений выбираем папку в которой будут храниться счетчики

Шаг 2. Кликаем правой кнопкой мыши и выбираем Добавить → Базовые сущности → Счётчик

Шаг 3. Открывается панель редактирования информации о счётчике:

- **Код -** Обязательное поле. Используется для обращения к счётчику в шаблоне номера.
- **Начальное значение -** значение, с которого начинается нумерация.
- **Следующее значение -** должно быть не меньше начального.
- **Период сброса -** определяет, когда следующее значение будет сбрасываться к начальному.

Возможные варианты:

- Никогда (по умолчанию)
- Каждый день

- Каждую неделю
- Каждый месяц
- Каждый год

Шаг 4. Вводим необходимые значения и нажимаем кнопку «Сохранить»

После создания счетчика нам необходимо привязать его к шаблону номера, который мы потом поместим на форму.

Привязка счетчика к шаблону номера

Шаг 1. Выбрав нужную папку кликаем по ней правой кнопкой мыши Добавить → Документооборот → Шаблон номера

Шаг 2. В открывшемся окне создания шаблона номера указываем:

- Наименование - название счетчика или его назначение (например «Номер заявки на услуги ATLAS»)
- Код - ставим логический код или оставляем автоматически сгенерированный транслит
- Значение - помещаем сюда код ранее созданного нами счетчика в формате {код_счетчика}

Шаг 3. Сохраняем шаблон номера.

Добавление шаблона номера на форму

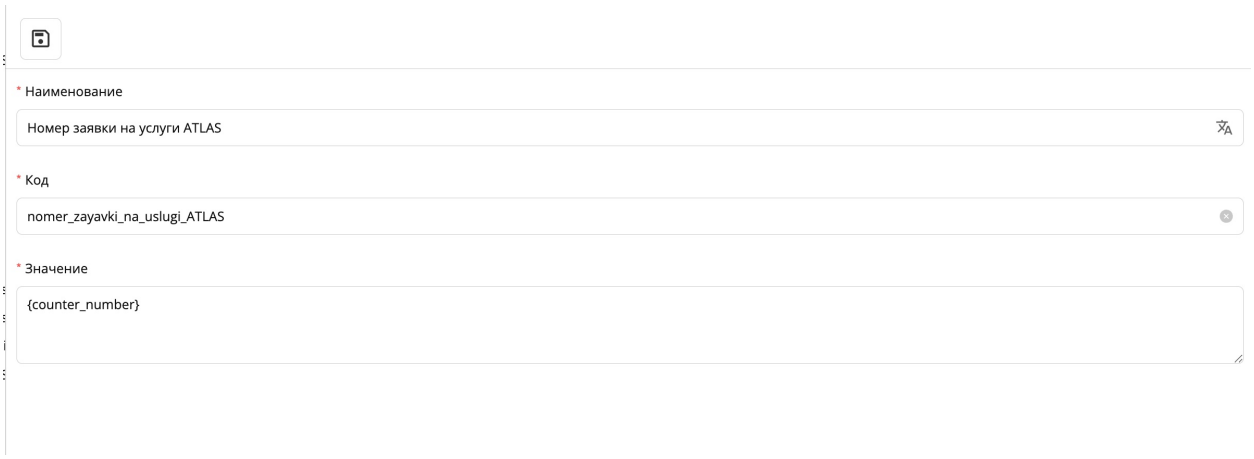
Шаг 1. Возвращаемся к форме заявки

Шаг 2. В поле **«Порядковый номер заявки»** выделяем его компонент **«Номер»**

Шаг 3. В строке **«Шаблон номера»** выбираем созданный нами шаблон

Шаг 4. Сохраняем форму

Сгенирированный автоматически номер будет являться регистрацией заявки в системе.



5.2.5 Этап 3. Отправка уведомления о номере заказа.

Вход - подписанная ЭЦП и зарегистрированная в системе заявка на услугу

Выход - уведомление на почту клиента

> На данном этапе на почту клиента отправляется уведомление о регистрации заявки с указанием её номера.

Шаг 1. В редакторе маршрута нажимаем кнопку «+» для добавления нового этапа

Шаг 2. В настройках этапа выбираем тип действия **«Отправка письма на почту»**

Шаг 3. Указываем:

- Наименование этапа, например **«Отправка уведомления о номере заказа»**
- Код - указывается при необходимости
- Код поля на форме - подразумевает поле на форме с почтой клиента, на которую необходимо отправить уведомление. У нас например это textbox_mail
- Тема письма с поддержкой HTML разметки - будет содержать тему письма
- Тело письма с поддержкой HTML разметки - будет содержать основной текст письма с данными из нужных нам полей

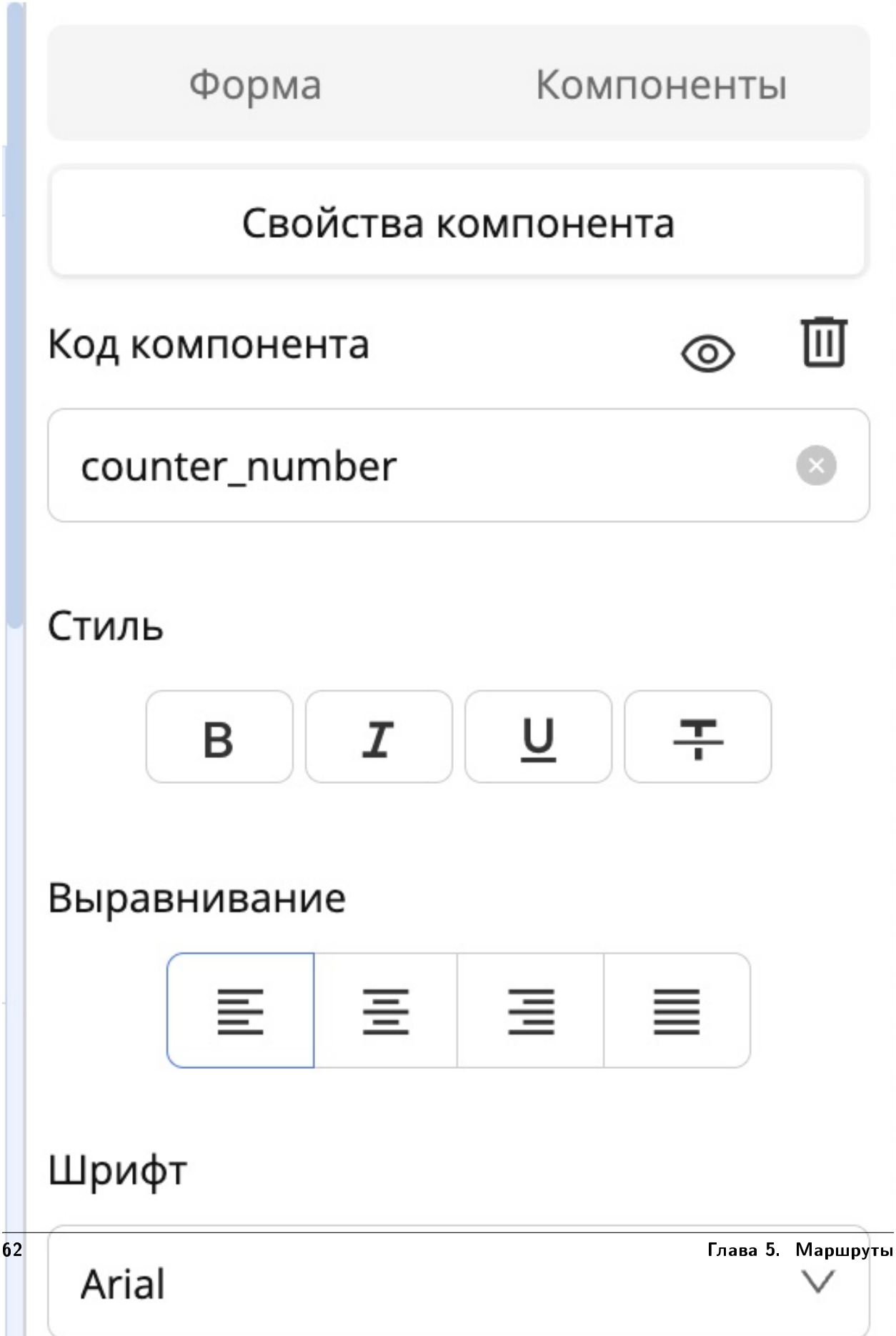
В нашем случае, согласно ТЗ (ордера) HTML шаблон письма будет выглядеть следующим образом:

```
Ваша заявка на “ + $listbox_type + ” зарегистрирована под номером “ +
$counter_number + ”.
```

- где `listbox_type` - код поля **«Тип заявки»**
- `counter_number` - код поля **«Номер заявки»**
- **«Ваша заявка на»** и **«зарегистрирована под номером»** - это статический текст, который будет одинаковым в каждом письме.
- Символ $ - обращение к переменной.

Символ $ обязательный и означает, что далее указывается переменная, значение которой будет взято из данных заявки:

- $listbox_type — значение поля Тип заявки
- $counter_number — значение номера заявки, сформированного счётчиком


Форма
Компоненты
Свойства компонента
Код компонента
counter_number
Стиль
B
I
U
Выравнивание
Шрифт
Arial

Во время отправки письма система автоматически подставляет фактические значения этих полей из данных заявки.

5.2.6 Этап 4. Назначение исполнителя

Вход - заявка поступившая менеджеру

Выход - заявка с назначенным по ней исполнителем

Для реализации данного шага, будем использовать тип **«Работа по форме»**

Назначение исполнителя выполняется менеджером. Так как конкретный исполнитель может отличаться от заявки к заявке, его необходимо указать вручную в процессе выполнения этапа.

Для начала нам необходимо создать на форме поле, в котором будет указан актуальный менеджерский состав для назначения исполнителя.

Предварительная настройка

Шаг 1. Возвращаемся на форму

Шаг 2. В скрытой таблице под полем «Автор» добавляем еще одно поле **«Менеджер»** с компонентом **«Объекты Synergy»**

Шаг 3. В настройках компонента присваиваем ему код содержащий смысловую нагрузку (например entity_manager) и в настройке **«Тип данных»** выбираем тип «Пользователь», «Должности» либо «Подразделения».

Примечание: Если мы хотим чтобы заявка на данном этапе падала конкретному человеку – мы прикрепляем к заявке конкретно его учетную запись, выбрав для этого тип данных **«Пользователь»** и выбрав в поле конкретного человека.

Если же, в системе имеется должность, с назначенными на нее пользователями, отвечающими за определенную работу, то используется тип данных **«Должности»**, в котором указывается должность и работа будет падать всем пользователям, состоящим на данной должности.

Если необходимо направлять работу целому подразделению, с вложенными в него должностями, то используется тип данных **«Подразделения»**, тогда работа будет поступать всем людям, состоящим на всех вложенных должностях выбранного подразделения.

Шаг 4. Сохраняем форму

Создание этапа маршрута

Шаг 1. В редакторе маршрута нажимаем кнопку «+» на панели «Действия»

Шаг 2. В настройках этапа указываем:

- Тип действия - работа по форме
- Название этапа - назначение исполнителя
- Ответственный - указываем код созданного нами поля с менеджером (например у нас это entity_manager)
- Тип работы - «Работа»
- Включаем настройку «Использовать форму завершения»

* Код поля на форме

textbox_mail

* Тема письма с поддержкой HTML разметки

Заявка на услугу компании ATLAS

* Тело письма с поддержкой HTML разметки

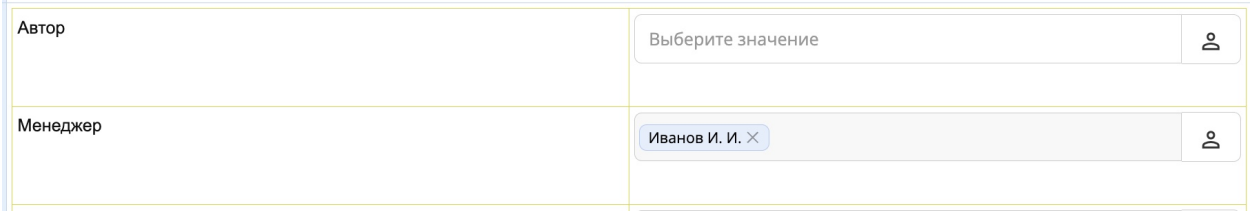


Рис. 2: Пример созданного поля с предзаполненным пользователем

Форма завершения.

Форма завершения — это модальное окно, которое открывается при выполнении работы и определяет, что должен сделать пользователь для завершения этапа. Существует несколько типов форм завершения:

- Комментарий — результатом является комментарий.
- Файл — результатом является файл (с устройства, из хранилища или из приложений работы).
- Документ — выбор или создание документа, связанного с работой.
- Форма — заполнение отдельной формы (используется в нашем случае).
- Без результата — завершение без результата.

Подробнее о формах завершения http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/completion_form.html

---

Создание формы завершения

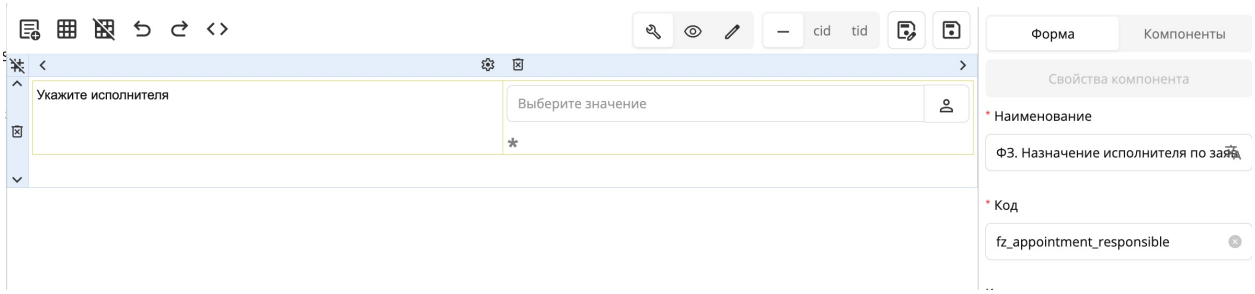
Шаг 1. Создаем отдельную форму кликнув правой кнопкой мыши по нужной папке → Добавить → Базовые сущности → Форма.

Шаг 2. Задаем форме:

- понятное наименование, отличающее её от основной формы
- аналогичный код

Шаг 3. Добавляем поле для указания исполнителя:

- Наименование - Исполнитель
- компонент «Объекты Synergy»



делаем поле обязательным, так как исполнитель является обязательным участником следующего шага процесса.

Шаг 4. Сохраняем форму.

Привязка формы завершения к этапу

Шаг 1. Возвращаемся к настройке этапа **«Назначение исполнителя»**

Шаг 2. В строке **«Форма завершения»** нажимаем кнопку «+». В открывшемся окне:

- выбираем тип завершения «Форма»;
- задаём код и наименование (для удобства делаем их аналогичными созданной форме);
- в поле «Форма» выбираем созданную форму.

Шаг 3. Сохраняем этап.

Важно: Если работа направляется на **должность** или **подразделение** необходимо включить настройку: **«Прервать выполнение параллельных этапов после завершения одного из них»**. Иначе каждому пользователю потребуется выполнить назначение исполнителя, прежде чем маршрут продолжится.

ПРИМЕЧАНИЕ

Т.к форма завершения является отдельной сущностью, данные, введенные в форму завершения не попадают в основную форму автоматически. Для дальнейшего использования данных, введенных в форму завершения, их нужно перенести в основную форму при помощи отдельного этапа **«Блокирующий процесс»**. Подробно будет описано в этапе 5

5.2.7 Этап 5. Смена статуса заявки

Вход - заявка с назначенным по ней исполнителем

Выход - заявка со статусом «в работе»

Смена статуса выполняется автоматически с помощью скрипта интерпретатора, запускаемого этапом маршрута типа **«Блокирующий процесс»**.

Примечание: **Скрипт интерпретатора** - это программный код на JavaScript внутри системы Synergy, который выполняется без компиляции и позволяет автоматизировать логику работы системы. Он используется для расчетов, обработки данных форм и карточек, а также для реакции на внутренние события системы. Скрипт выполняется сразу при запуске или событии и возвращает результат выполнения системе.

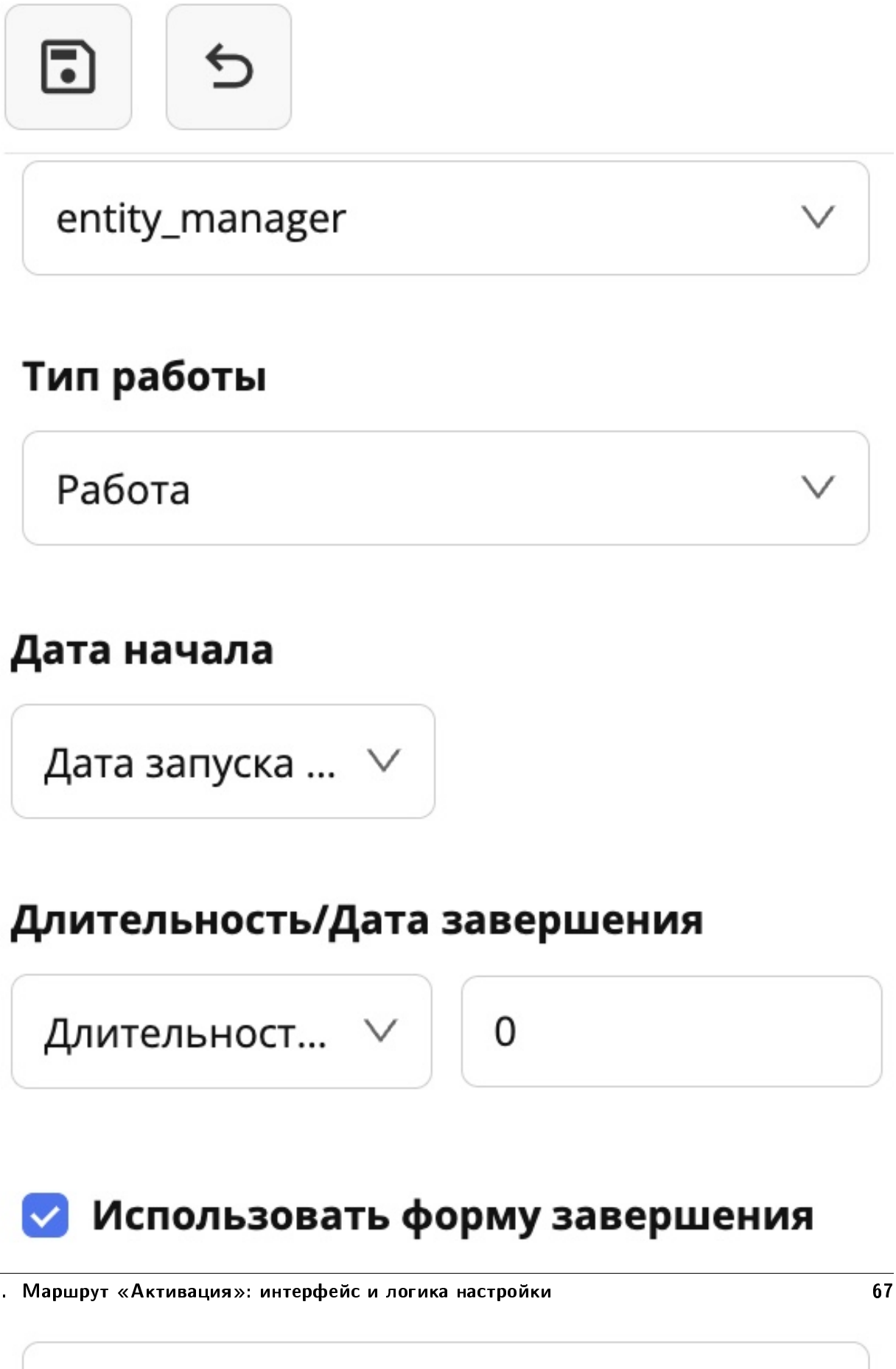
Подробнее о том, что такое скрипт интерпретатора http://rtd.lan.arta.kz/docs/guide/ru/minsky/interpreter.html

---

Создание скрипта интерпретатора

Шаг 1. В дереве приложения выбираем папку для хранения скриптов. Для удобства это может быть:

- папка integrations,
- либо можно создать отдельную папку block_processes.


entity_manager
Тип работы
Работа
Дата начала
Дата запуска ...
Длительность/Дата завершения
Длительност...
0
Использовать форму завершения

Шаг 2. Кликаем по папке правой кнопкой мыши → Добавить → Интеграция → Скрипт интерпретатора.

Шаг 3. В открывшемся окне редактирования кода указываем:

- Наименование - Система автоматически задаёт базовый префикс `event.blocking.interpreter`

Необходимо дополнить наименование и код, используя латинские буквы и отражая логику работы скрипта. Например `event.blocking.interpreter.change.status_work`

- Код - формируется автоматически из наименования
- Описание - добавляется при необходимости для пояснения логики работы скрипта
- Комментарий по умолчанию - обязательная настройка, это сообщение которое система выведет после выполнения скрипта. (например «ОК» или «Статус изменен»)
- **Авторизация - скрипту для срабатывания необходимы права доступа. Здесь имеется два варианта**
  - По логину и паролю - понадобится логин и пароль учетной записи с правами администратора
  - По ключу - понадобится ключ администратора

Шаг 4. Выбираем тип авторизации **«По логину и паролю»** и указываем учетную запись администратора

Важно: При смене логина и пароля учетной записи администратора логин и пароль нужно будет обновить во всех скриптах интерпретатора, использующих эту учетную запись

Шаг 5. В редактор кода вставляем стандартный скрипт для смены статуса

```
var result = true;
var message = 'Смена значения статуса';
 try {
   var form = platform.getFormsManager().getFormData(dataUUID);
   form.load();
   form.setValue('код справочника', 'код значения');
   form.save();
} catch (e) {
  result = false;
  message = e.message;
}
```

Шаг 3. Вместо слов:

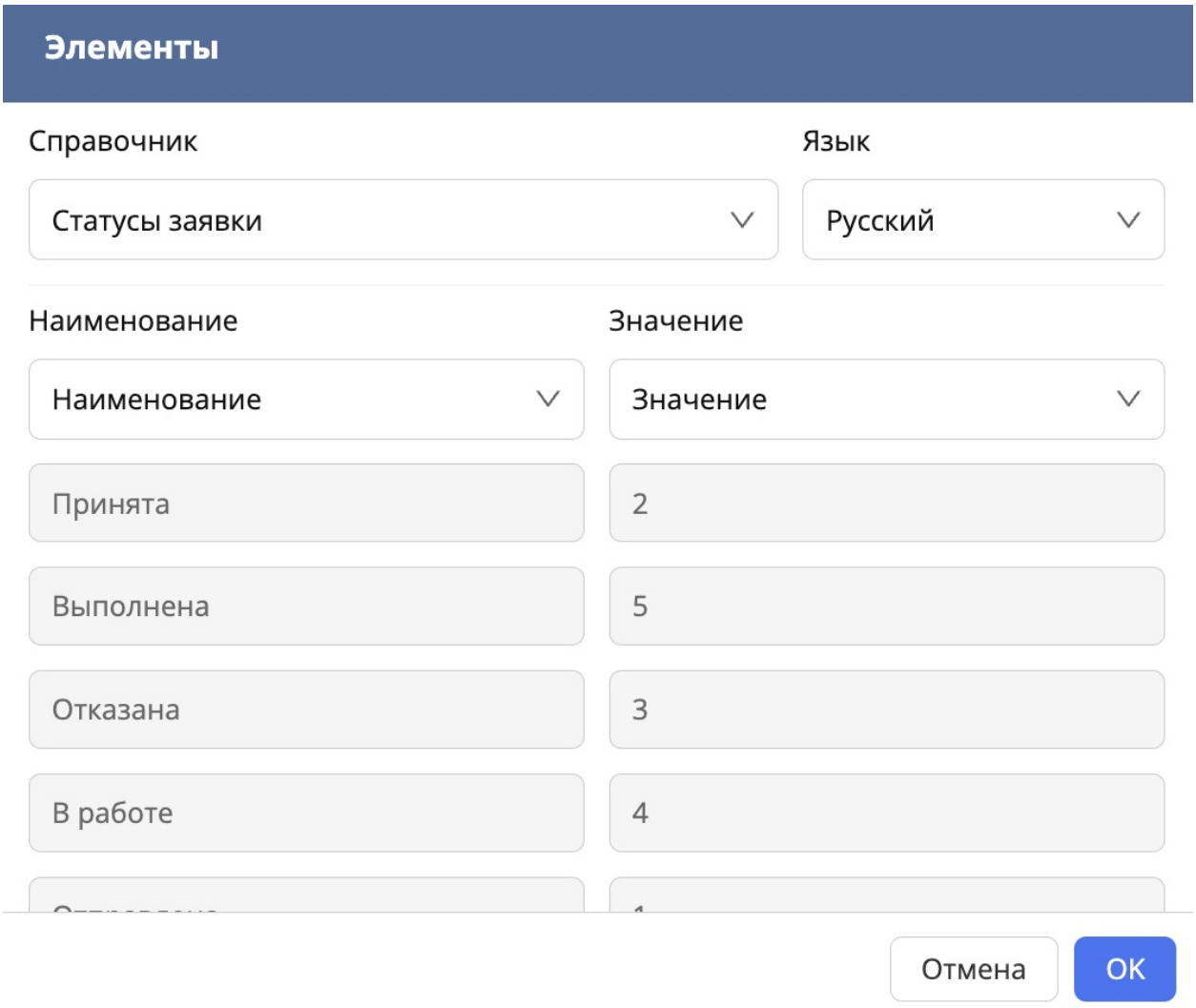
- **«код справочника»** - указываем код поля в котором находится выпадающий список статусов, например у нас это `listbox_status`
- **«код значения»** - значение, на которое должен смениться статус, согласно справочнику у нас это значение 4 («В работе»)

Если все сделано правильно, код будет выглядеть так:

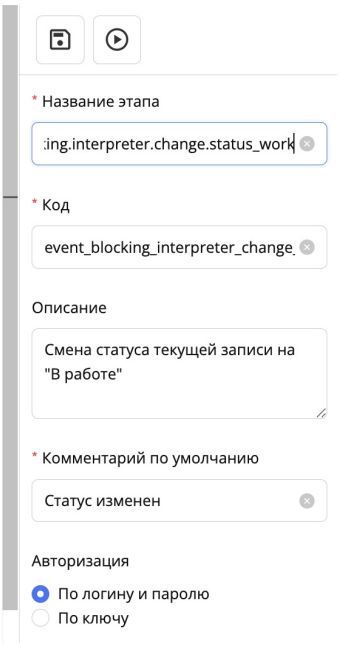
Шаг 4. Сохраняем скрипт интерпретатора и возвращаемся к настройке этапа

Создание этапа маршрута

Шаг 1. В маршруте добавляем новый этап.




```
var result = true;
var message = 'Смена значения статуса';

try {
    var form = platform.getFormsManager().getFormData(dataUUID);
    form.load();
    form.setValue('listbox_status', '4');
    form.save();
} catch (e) {
    result = false;
    message = e.message;
}
```



Шаг 2. В настройках этапа:

- указываем наименование этапа (например, «Смена статуса на «В работе»»);
- в поле **«Событие»** вставляем название созданного скрипта интерпретатора.

Шаг 3. Сохраняем этап.

Скрипт интерпретатора для этапа 4

Теперь когда мы умеем добавлять блокирующий процесс в маршрут, создаем еще один скрипт интерпретатора для переноса данных из формы завершения этапа 4 в основную форму.

Шаг 1. Для начала добавляем на основную форму заявки поле **«Исполнитель»** из формы завершения. Поле должно иметь аналогичный код.

Шаг 2. Сохраняем форму заявки

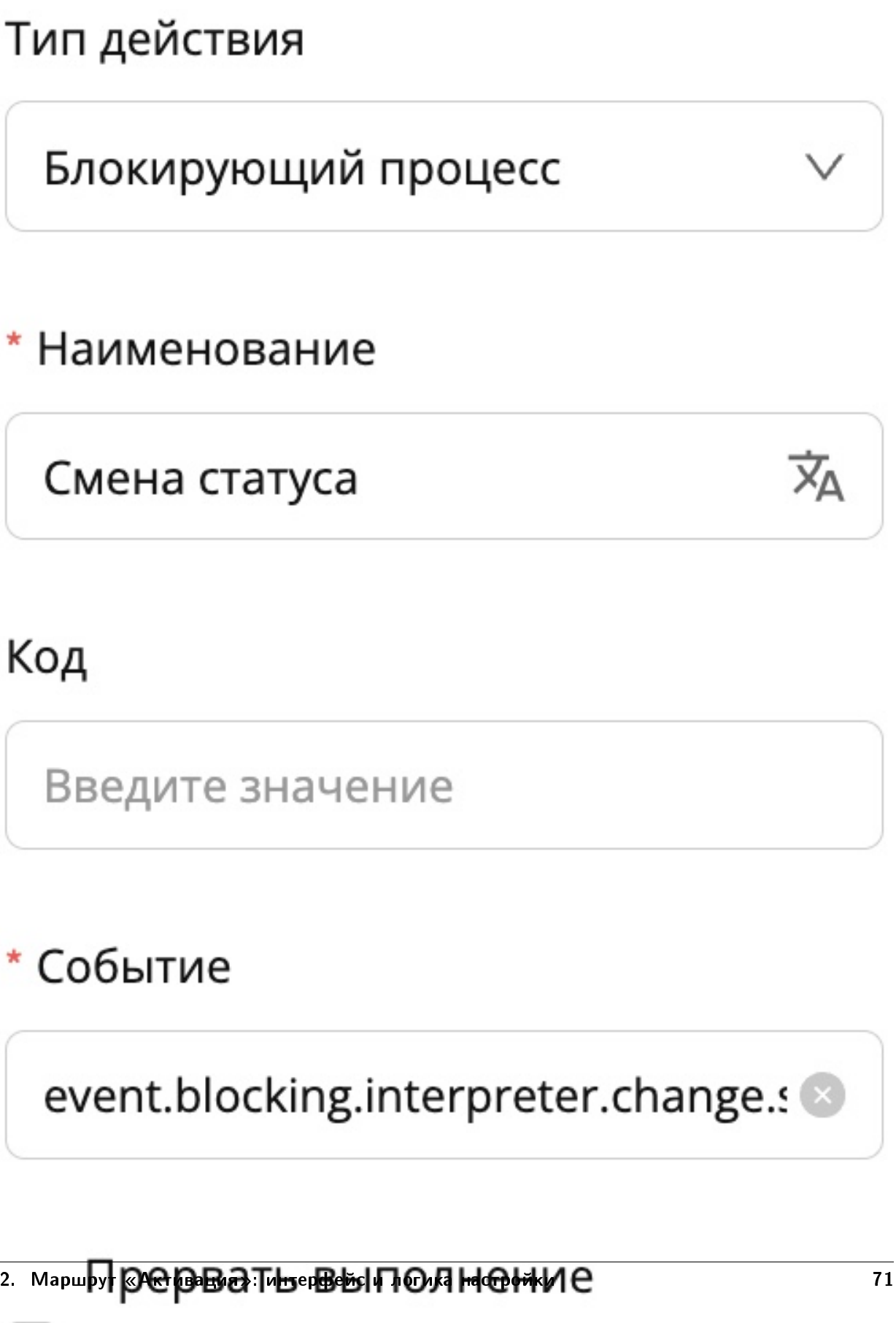
Шаг 3. Правой кнопкой мыши по нужной папке → добавить → интеграция → скрипт интерпретатора

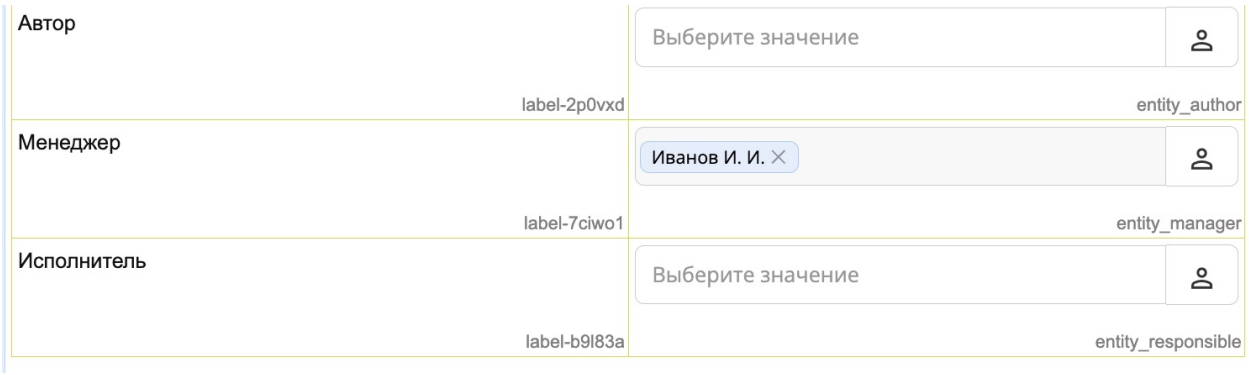
Шаг 4. В редактор кода добавляем скрипт, который переносит данные с формы завершения на основную форму:

Пример такого блокирующего процесса:

```
var result = true;
var message = 'ok';

function getHttpClient() {
  let client = new org.apache.commons.httpclient.HttpClient();
  let creds = new org.apache.commons.httpclient.UsernamePasswordCredentials(login, password);
  client.getParams().setAuthenticationPreemptive(true);
  client.getState().setCredentials(org.apache.commons.httpclient.auth.AuthScope.ANY, creds);
  return client;
}

function httpGetMethod(methods, type) {
  let client = getHttpClient();
  let get = new org.apache.commons.httpclient.methods.GetMethod(
    "http://127.0.0.1:8080/Synergy/" + methods
  );
  get.setRequestHeader("Content-type", "application/json");
  client.executeMethod(get);
  let resp = get.getResponseBodyAsString();
  get.releaseConnection();
  return type == 'text' ? resp : JSON.parse(resp);
}

function httpPostMethod(methods, params, contentType) {
  let client = getHttpClient();
  let post = new org.apache.commons.httpclient.methods.PostMethod(
    "http://127.0.0.1:8080/Synergy/" + methods
  );

  if (contentType) {
    post.setRequestBody(JSON.stringify(params));
  } else {
    for (let key in params) post.addParameter(key, params[key]);
  }
```

(continues on next page)


Тип действия
Блокирующий процесс
* Наименование
Смена статуса
Код
Введите значение
* Событие
event.blocking.interpreter.change.s
Прервать выполнение
параллельных этапов после



(продолжение с предыдущей страницы)

```
  post.setRequestHeader(
    "Content-type",
    contentType || "application/x-www-form-urlencoded; charset=utf-8"
  );

  let resp = client.executeMethod(post);

  if (contentType) {
    resp = JSON.parse(post.getResponseBodyAsString());
  }

  post.releaseConnection();
  return resp;
}

function getProcesses(documentID) {
  return httpGetMethod("rest/api/workflow/get_execution_process?documentID=" + documentID);
}

function getWorkCompletionData(workID) {
  return httpGetMethod("rest/api/workflow/work/get_completion_data?workID=" + workID);
}

function getFormData(asfDataId) {
  return httpGetMethod("rest/api/asforms/data/" + asfDataId);
}

function mergeFormData(uuid, data) {
  return httpPostMethod(
    "rest/api/asforms/data/merge",
    {
      uuid: uuid,
      data: data
    },
    "application/json; charset=utf-8"
  );
}

let UTILS = {
  createField: function (fieldData) {
    let field = {};
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    for (let key in fieldData) field[key] = fieldData[key];
    return field;
  },

  getValue: function (data, cmpID) {
    data = data.data ? data.data : data;
    for (let i = 0; i < data.length; i++) {
      if (data[i].id === cmpID) return data[i];
    }
    return null;
  },

  setValue: function (asfData, cmpID, data) {
    let field = this.getValue(asfData, cmpID);

    if (field) {
      for (let key in data) {
        if (key === 'id' || key === 'type') continue;
        field[key] = data[key];
      }
      return field;
    } else {
      asfData = asfData.data ? asfData.data : asfData;
      field = this.createField(data);
      field.id = cmpID;
      asfData.push(field);
      return field;
    }
  }
};

function processesFilter(processes) {
  let result = [];

  function search(p) {
    p.forEach(function (x) {
      if (x.typeID == 'ASSIGNMENT_ITEM' && x.finished) result.push(x);
      if (x.subProcesses.length > 0) search(x.subProcesses);
    });
  }

  search(processes);

  return result.sort(function (a, b) {
    return new Date(b.finished) - new Date(a.finished);
  });
}

try {
  let processes = processesFilter(getProcesses(documentID));

  if (!processes.length) throw new Error('Не найденно завершенной работы');

  let resultFormWork = getWorkCompletionData(processes[0].actionID);

  if (!resultFormWork || !resultFormWork.result.hasOwnProperty('dataUUID')) {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    throw new Error('Не найдена форма завершения');
  }

  let completionFormData = getFormData(resultFormWork.result.dataUUID);

  let newFormData = [];
  let matching = [];

  // поля для сопоставления с фз на основную форму
  matching.push({ from: 'поле на форме завершения', to: 'поле основной формы' });

  matching.forEach(function (id) {
    let fromData = UTILS.getValue(completionFormData, id.from);
    if (fromData) UTILS.setValue(newFormData, id.to, fromData);
  });

  mergeFormData(dataUUID, newFormData);

} catch (err) {
  message = err.message;
}
```

Шаг 5. Находим в скрипте блок сопоставления полей matching и заменяем текст внутри кавычек

- from - код поля «Исполнитель» в форме завершения
- to - код поля «Исполнитель» в осноной форме, куда переносим значение

Пример:

До:

```
113     // поля для сопоставления с фз на основную форму
114     matching.push({ from: 'поле на форме завершения', to: 'поле основной формы' });
```

После:

```
113   //поля для сопоставления с фз на основную форму
114     matching.push({ from: 'entity_responsible', to: 'entity_responsible'});
```

Шаг 6. Присваиваем скрипту интерпретатора наименование, например `event.blocking.interpreter.completion_form_responsible`

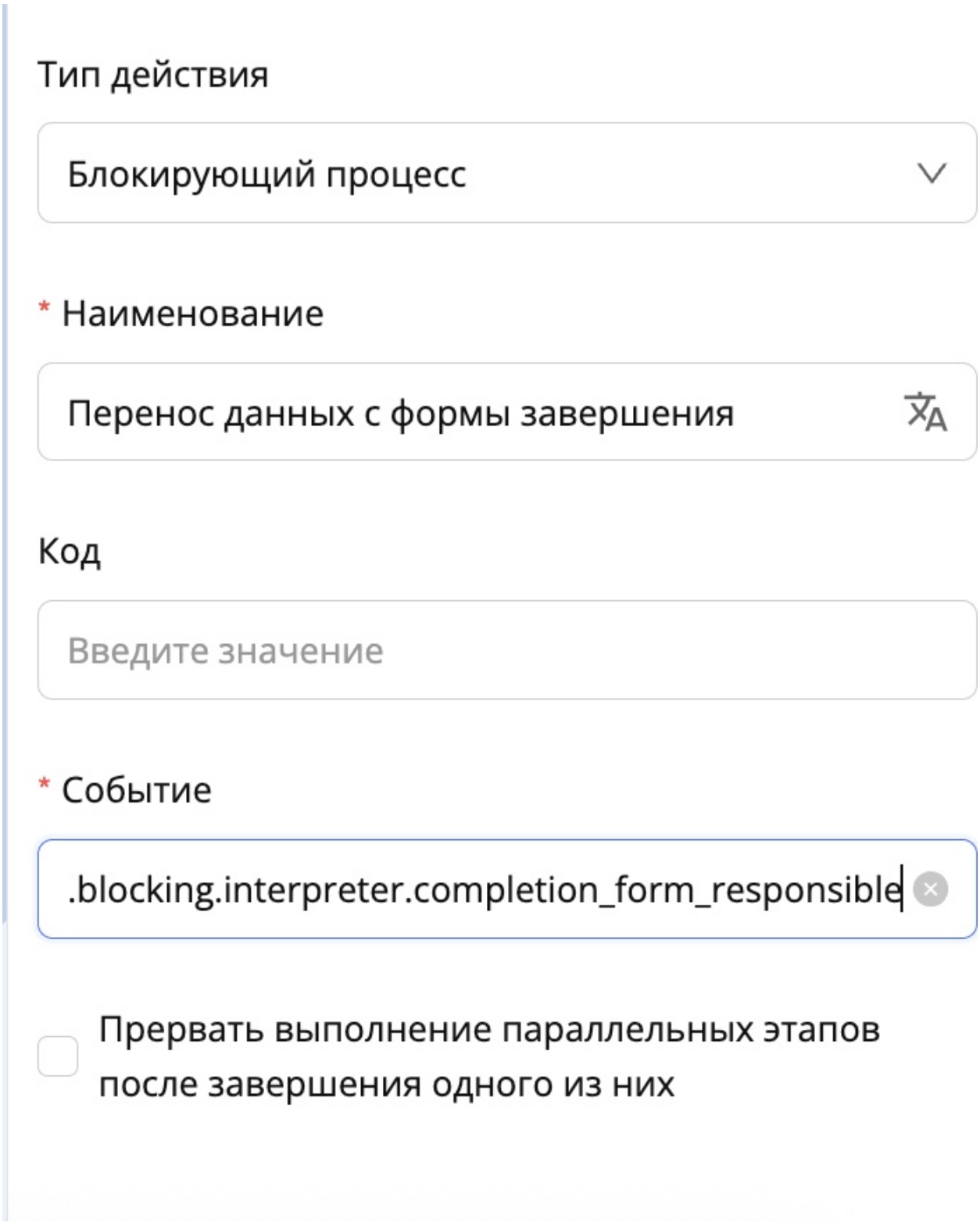
Шаг 7. При необходимости добавляем описание и указываем авторизационные данные как мы это делали ранее

Шаг 8. После того как все готово, сохраняем сущность и переходим к добавлению этапа в маршрут.

5.2.8 Добавление дополнительного шага для этапа 4

Шаг 1. Добавляем этап с типом действия «Блокирующий процесс»:

- Наименование - перенос данных с формы завершения


Тип действия
Блокирующий процесс
* Наименование
Перенос данных с формы завершения
Код
Введите значение
* Событие
.blocking.interpreter.completion_form_responsible
Прервать выполнение параллельных этапов
после завершения одного из них

- Событие - вставляем код созданного нами скрипта интерпретатора для переноса данных из формы завершения (`event.blocking.interpreter.completion_form_responsible`)

Шаг 2. Сохраняем этап и переходим к формированию правильного порядка этапов. Нам необходимо передвинуть этап с переносом формы завершения непосредственно под этап с формой завершения.

Шаг 3. Для переноса этапа зажимаем кнопку переноса этапа рядом с цифрой этапа

и двигая строку в нужное место

	№ п/п	Действие	Код этапа	Название этапа	Ответственный
⠿	1	Работа по форме		Прошу подписать заявку с использованием ЭЦП	-
⠿	2	Отправка письма на почту		Отправка уведомления о номере заказа	-
⠿	3	Работа по форме		Назначение исполнителя	-
⠿	4	Блокирующий процесс		Перенос данных с формы завершения	-
⠿	5	Блокирующий процесс		Смена статуса	-

Теперь все этапы располагаются в правильном порядке.

5.2.9 Этап 6. Отправка уведомления заявителю о смене статуса заявки

Вход - заявка со статусом «В работе»

Выход - уведомление заявителю о смене статуса

Здесь мы используем уже знакомый нам этап **«Отправка письма на почту»**

Шаг 1. В редакторе маршрута нажимаем кнопку «+» для добавления нового этапа.

Шаг 2. В настройках этапа выбираем:

- Тип действия — Отправка письма на почту;
- Название этапа — «Уведомление о смене статуса заявки»;
- Код этапа — при необходимости.
- Код поля на форме- Указываем поле формы, содержащее адрес электронной почты заявителя. В нашем случае — `textbox_mail`.
- Тема письма (с поддержкой HTML-разметки) - «Изменен статус заявки на услуги компании Atlas»
- Тело письма (с поддержкой HTML-разметки) - «Ваша заявка на » + $listbox_type + » под номером » + $counter_number + » принята в работу»

* Код поля на форме

textbox_mail

* Тема письма с поддержкой HTML разметки

Ваша заявка на услуги компании ATLAS принята в работу

* Тело письма с поддержкой HTML разметки

Шрифт Размер ...

Ваша заявка на “ + $reglink_type + ” под номером “ + $counter_number + ” приянта в работу

Шаг 3. Сохраняем этап

5.2.10 Этап 7. Проверка типа услуги

На данном этапе система автоматически определяет тип услуги, указанный в заявке, и направляет её по соответствующему сценарию обработки.

Вход - поданная в системе заявка с указанным типом услуги.

Выход -

- если тип услуги — «Подписка на обслуживание», заявка направляется на этап создания договора
- если другой тип услуги - направляется в работу исполнителю

Этот этап используется для реализации ветвления бизнес-процесса в зависимости от значений полей формы заявки.

Для этого применяется тип действия **«Условный переход»**. Данный тип действия выполняется системой и подразумевает собой некую «проверку», считывающую условия на форме заявки (по умолчанию) либо с формы завершения, и отправляющую заявку на один из этапов маршрута.

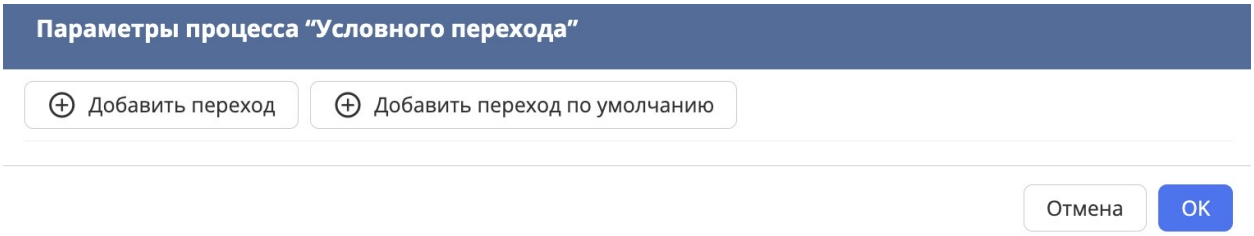
Шаг 1. В редакторе маршрута добавляем новый этап, нажав кнопку «+».

Шаг 2. В настройках этапа выбираем:

- Тип действия — Условный переход;
- Название этапа — Проверка типа услуги;
- Код этапа — при необходимости

Шаг 3. Открываем вкладку «Переходы». Вкладка «Переходы» содержит:

- кнопку добавить переход;
- добавить переход по умолчанию.



Шаг 4. Во вкладке «Переходы» нажимаем **Добавить переход»**

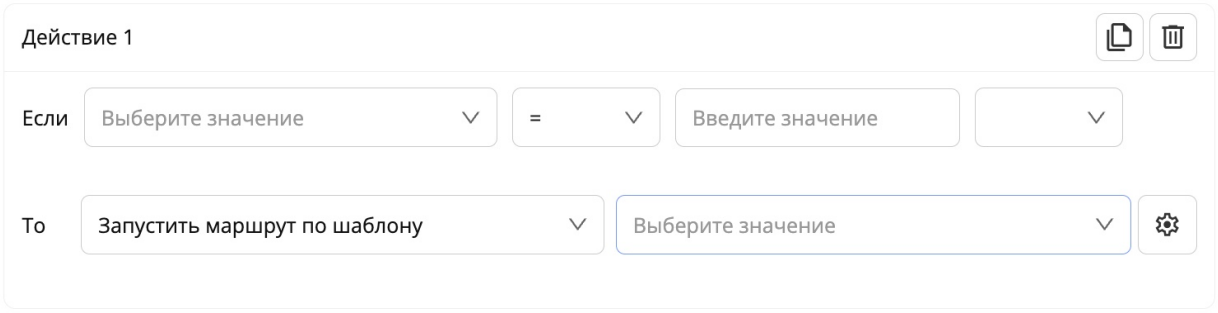
Шаг 5. В левом операнде указываем код поля, которое проверяем — мы проверяем, в данном случае **«Вид услуги»**.

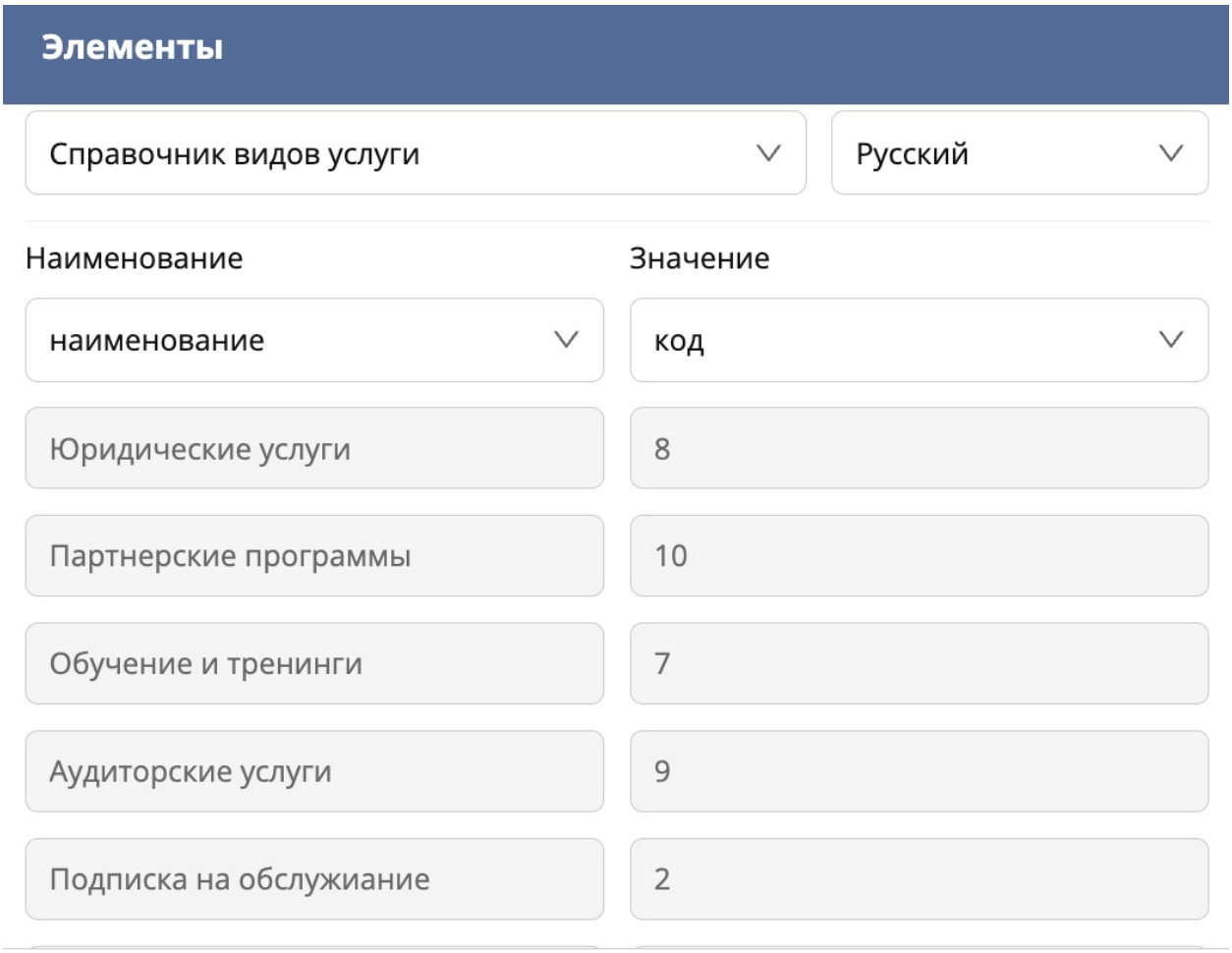
Шаг 6. В операторе сравнения выбираем **«=»**, т.к нам нужно осуществлять переход случае если Вид услуги = определенному какому-то значению

Шаг 7. В правом операнде нам необходимо указать значение, которое должно быть выбрано в нашем поле **«Вид услуги»**, для осуществления перехода. Для этого мы обращаемся к колонке справочника, в которой указано значение услуги.

- если выбрано значение 2 – Подписка на обслуживание, нам нужно отправить заявку на создание договора.

Переходы


Действие 1
Если
Выберите значение
=
Введите значение
То
Запустить маршрут по шаблону
Выберите значение


Элементы
Справочник видов услуги
Русский
Наименование
Значение
наименование
код
Юридические услуги
8
Партнерские программы
10
Обучение и тренинги
7
Аудиторские услуги
9
Подписка на обслужиание
2

Для этого в правом операнде мы указываем значение **2**

Шаг 8. Настройка действия **«То»**. При выполнении условия доступны два варианта:

- Запустить маршрут по шаблону;
- Перейти к этапу.

В нашем случае требуется переход к этапу создания договора, поэтому:

- выбираем **«Перейти к этапу»**;
- указываем код будущего этапа, например: `create_agreement`

Что происходит при такой настройке

система считывает значение поля **«Вид услуги»**;

если значение = **«Подписка на обслуживание»**, маршрут переходит к этапу создания договора.

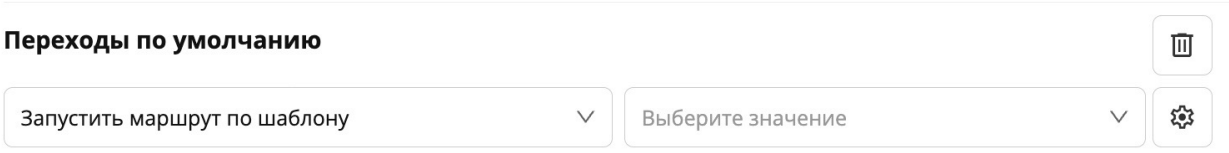
Шаг 9. Во вкладке **«Переходы»** нажимаем **«Добавить переход по умолчанию»**.

Зачем нужен переход по умолчанию

Если указано только одно условие, а оно не выполняется, маршрут не продолжится. Поэтому необходимо задать альтернативный путь.

Шаг 10. В переходе по умолчанию указываем действие **«перейти к этапу»**

Шаг 11. Вводим код этапа будущей работы исполнителя, например: `work_executor`



Шаг 12. Сохраняем настройки условных переходов и сам этап.

Примечание: Подробнее об условных переходах http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/conditional_transitions.html

---

5.2.11 Этап 8. Формирование договора

Вход - заявка на услугу с типом «Подписка на обслуживание».

Выход - созданный в системе договор на оказание услуг.

На данном этапе система автоматически формирует договор на оказание услуг. Для этого будем использовать тип действия маршрута **«Создать запись в реестре»**

Этап используется для создания новой записи в отдельном реестре договоров на основании данных заявки.

Шаг 1. В редакторе маршрута добавляем новый этап, нажав кнопку «+».

Шаг 2. В настройках этапа указываем:

- Тип действия - Создать запись в реестре;
- Наименование - Формирование договора;
- Код - `create_agreement` (тот же код, который был указан в этапе условного перехода).
- Поле **«Реестр, в котором необходимо создать запись»** на данном этапе оставляем пустым.

Оно будет заполнено после создания формы и реестра договоров.

- В поле **«От кого совершается действие»** указываем пользователя, от имени которого будут создаваться записи в реестре договоров. Как правило, используется учетная записи с правами администратора.
- Отмечаем галочкой настройку **«Активировать запись реестра»**, если необходимо чтобы запись после создания отправлялась по маршруту

5.2.12 Этап 8.1. Завершение работы исполнителем

Вход - заявка, обработанная исполнителем.

Выход - завершённая заявка и отправка уведомления на электронную почту.

Для удобства, вынесем эту ветвь процесса в отдельный **«Шаблон маршрута»**.

Шаблон маршрута:

Представляет собой вложенный маршрут, который может использоваться одновременно в нескольких реестрах. Настройки маршрута в шаблоне почти не отличается от настройки маршрута в реестре, за исключением отсутствия предварительных и последующих этапов.

Подробнее о том что такое шаблон маршрута http://rtd.lan.arta.kz/docs/docs-po-platforme-arta-synergy/ru/latest/route_template.html

---

Шаг 1. Кликаем правой кнопкой мыши по удобной для нас папке и выбираем: Добавить → Процессы → Шаблон маршрута.

Шаг 2. В открывшейся странице указываем:

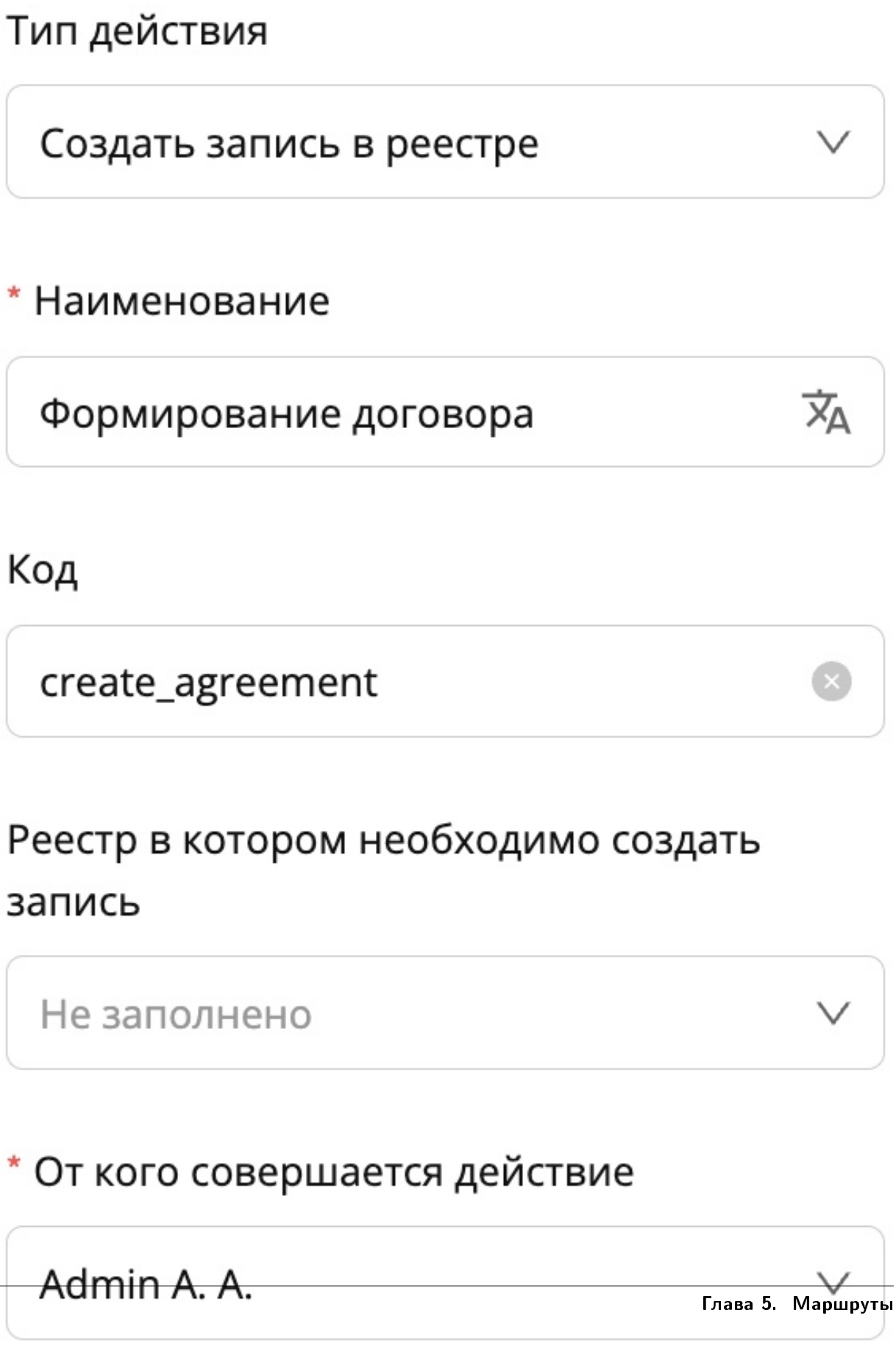
- наименование шаблона маршрута
- соответствующий код

Шаг 3. Сохраняем получившееся. После сохранения становится доступна уже знакомая нам панель создания маршрута.

Шаг 4. Нажимаем «+» и добавляем этап работы исполнителя:

- Тип действия - работа по форме
- Ответственный - код поля, в котором указывается исполнитель, назначенный менеджером.
- Тип работы - работа
- включаем настройку «Использовать форму завершения» и выбираем форму завершения «Комментарий»

Шаг 5. Сохраняем этап.


Тип действия
Создать запись в реестре
* Наименование
Формирование договора
Код
create_agreement
Реестр в котором необходимо создать запись
Не заполнено
* От кого совершается действие
Admin A. A.

5.2.13 Этап 8.2. Смена статуса заявки на «Выполнена»

Вход - заявка с выполненной работой исполнителя.

Выход - заявка со статусом **«Выполнена»**.

Для смены статуса используется уже известный нам скрипт интерпретатора.

Для удобства возьмем уже созданный скрипт интерпретатора с похожей логикой и сделаем его копию с некоторыми изменениями.

Шаг 1. В дереве объектов находим ранее созданный скрипт интерпретатора смены статуса.

Шаг 2. Кликаем по нему правой кнопкой мыши и выбираем «Сделать копию».

Шаг 3. Открываем созданную копию и в коде скрипта меняем значение статуса на нужное, в нашем случае нужно поменять значение 4 (статус в работе) на 5 (выполнена).

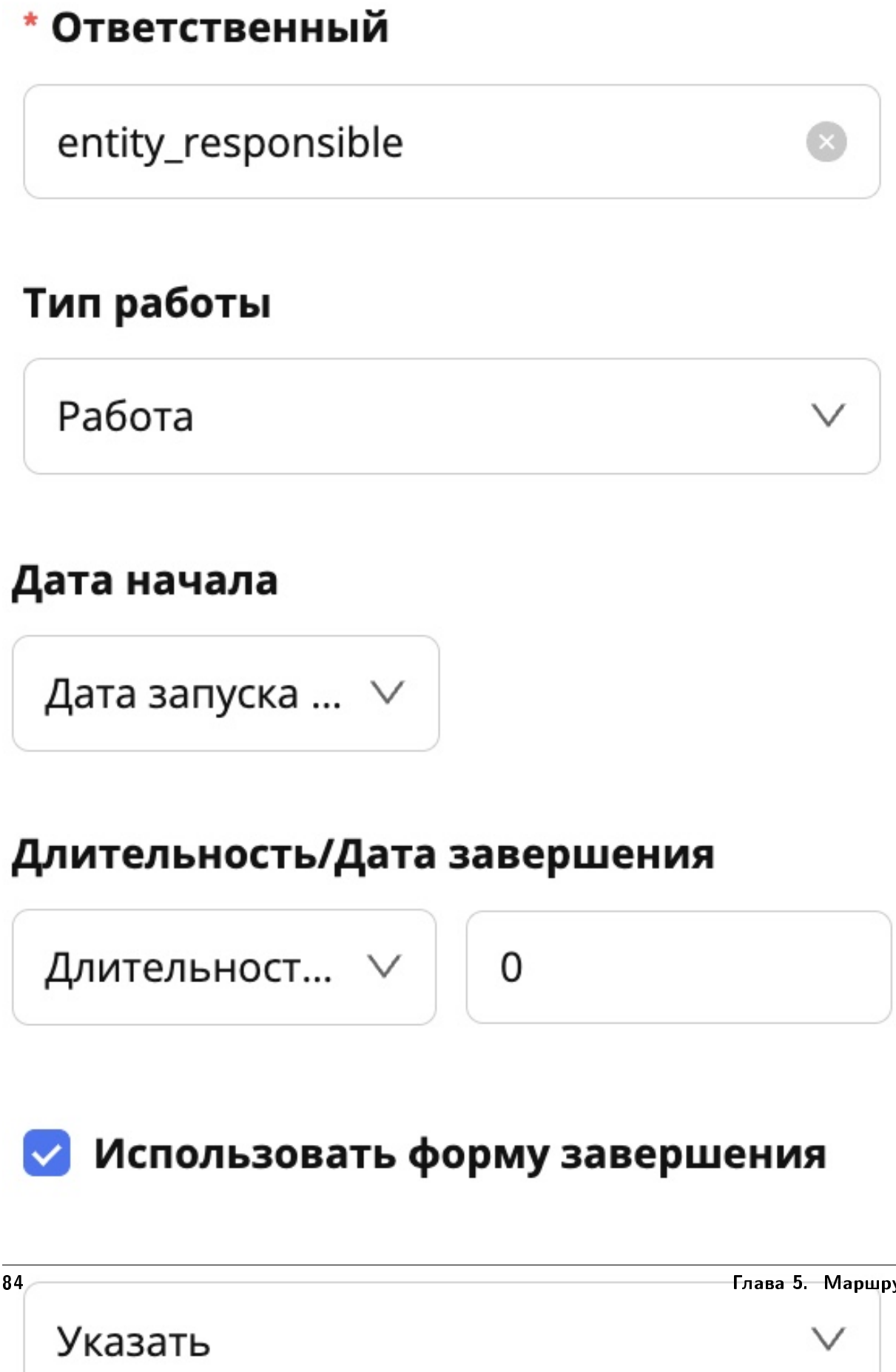
Шаг 4. Изменяем наименование и код скрипта, на соответствующее логике например `event.blocking.interpreter.change.status_ready`

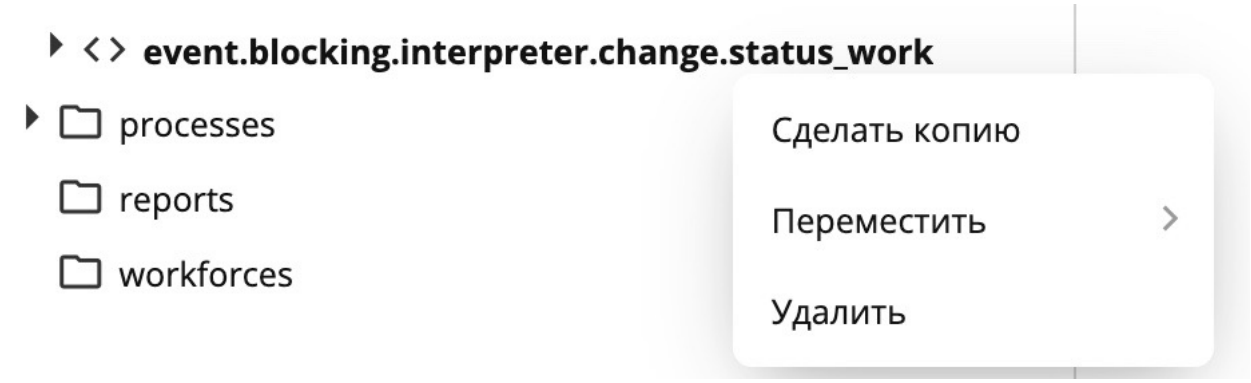
Шаг 5. Сохраняем скрипт.

Шаг 6. Возвращаемся в шаблон маршрута и добавляем новый этап:

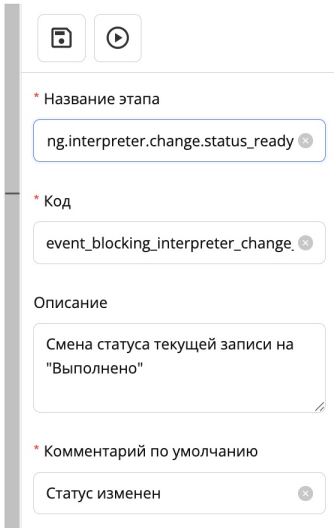
- Тип действия - блокирующий процесс
- Наименование - «Смена статуса заявки на «Выполнена»«
- Событие - вставляем наименование созданного нами скрипта

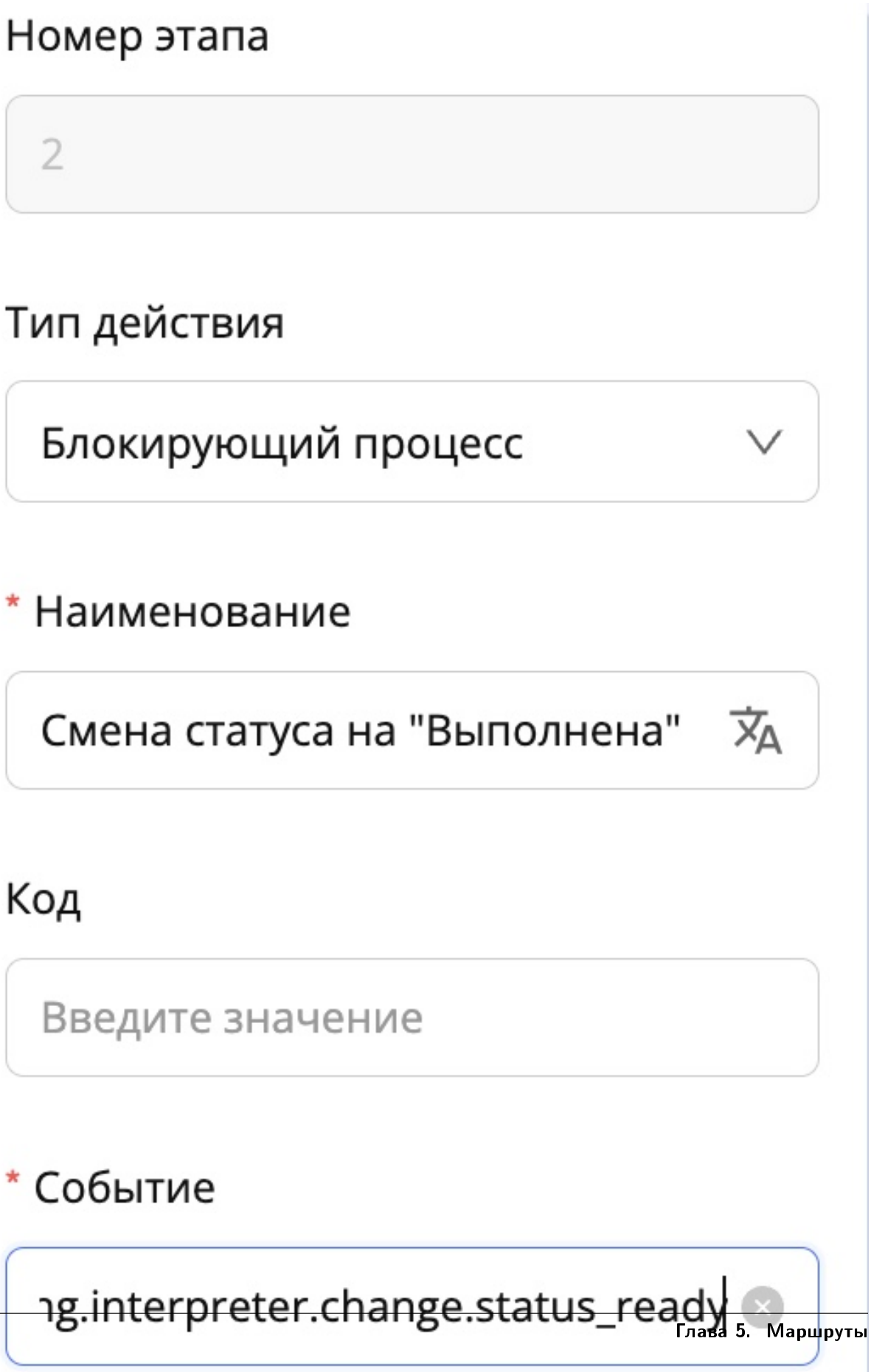
Шаг 7. Сохраняем этап


* Ответственный
entity_responsible
Тип работы
Работа
Дата начала
Дата запуска ...
Длительность/Дата завершения
Длительност...
0
Использовать форму завершения
Указать




```
var result = true;
var message = 'Смена значения статуса';

try {
    var form = platform.getFormsManager().getFormData(dataUUID);
    form.load();
    form.setValue('listbox_status', '5');
    form.save();
} catch (e) {
    result = false;
    message = e.message;
}
```




Номер этапа
2
Тип действия
Блокирующий процесс
* Наименование
Смена статуса на "Выполнена"
Код
Введите значение
* Событие
ng.interpreter.change.status_ready

5.2.14 Этап 8.3. Отправка уведомления о статусе заявки

После смены статуса необходимо уведомить заявителя, что его заявка выполнена.

Шаг 1. В том же шаблоне маршрута добавляем новый этап:

- Тип действия - «Отправка письма на почту»
- Код поля формы с почтой заявителя — `textbox_mail`
- Тема письма - «Ваша заявка на услуги компании ATLAS выполнена»
- Тело письма - «Ваша заявка на » + $listbox_type + «под номером » + $counter_number + » выполнена.»

Шаг 2. Сохраняем этап и шаблон маршрута.

5.2.15 Привязка шаблона маршрута к реестру

Теперь созданный шаблон необходимо подключить к основному маршруту заявки через условный переход.

Шаг 1. Возвращаемя в основной маршрут и выделяем этап «Условный переход».

Шаг 2. Открываем вкладку «Переходы»

Шаг 3. В разделе «Переход по умолчанию» выбираем вариант «Запустить маршрут по шаблону»

Шаг 4. Из открывшегося списка выбираем созданный нами шаблон маршрута

Шаг 5. Переходим во вкладку настроек рядом с этапом условного перехода

(иконка)

Шаг 6. В параметре «После выполнения» выбираем «Перейти к этапу» и указываем код этапа `end`.

Примечание: Так мы уточняем что после завершения маршрута в шаблоне, основной маршрут заявки должен перейти к опредленному этапу (в данном случае - в конец), а не на следующие по порядку этапы.

Шаг 7. Нажимаем «ОК» и сохраняем этап.

5.2.16 Этап 9. Создание подпроцесса «Формирование договора»

В ордере №2 описан подпроцесс создания договора на основании заявки.

Подготовка формы и реестра договора

Создаем форму и реестр будущего подпроцесса согласно ордеру №2 (Процесс создания формы и реестра уже был рассмотрен ранее.)

Шаг 1. Возвращаемся в маршрут основной заявки к этапу «Создать запись в реестре»:

- Тип действия - «Создать запись в реестре»
- Код этапа - create_agreement (указанный нами ранее в условном переходе)
- Реестр в котором необходимо создать запись - указываем созданный нами реестр договора.

* Код поля на форме

textbox_mail

* Тема письма с поддержкой HTML разметки

Ваша заявка на услуги компании ATLAS выполнена

* Тело письма с поддержкой HTML разметки

- Если нам необходимо чтобы созданный документ сразу после создания отправлялся по маршруту, делаем активной галочку **“Активировать запись реестра”**

Шаг 2. Ниже переходим во вкладку **«Настроить сопоставления»**

Шаг 3. В открывшемся окне нажимаем «+ **Добавить сопоставление».**

Шаг 4. В открывшихся выпадающих списках выбираем поля откуда и куда попадут данные:

- Слева - поля заявки (from - откуда)
- Справа - поля договора (to - куда)

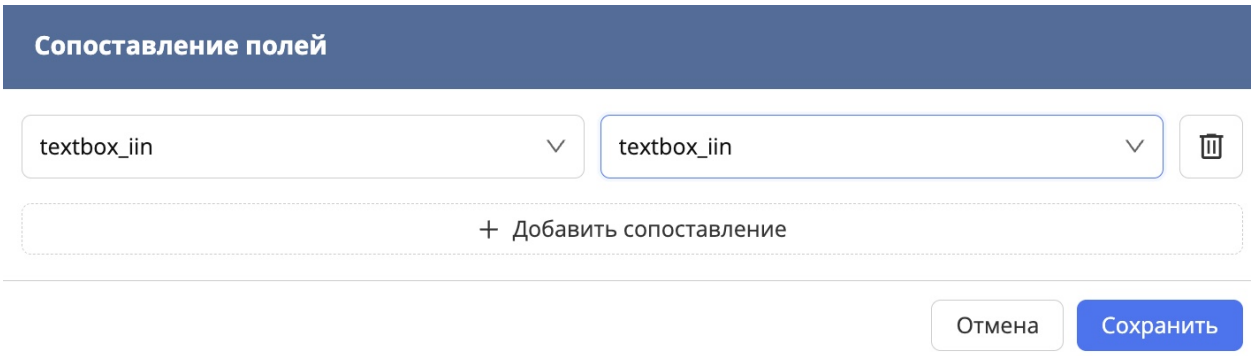


Рис. 3: Пример сопоставления ИИН

Шаг 5. После настройки всех сопоставлений нажимаем «Сохранить» в модальном окне.

Шаг 6. Сохраняем этап маршрута.

Этап 10. Конец маршрута.

Вход - заявка, прошедшая все предусмотренные этапы маршрута. **Выход -** завершённый процесс.

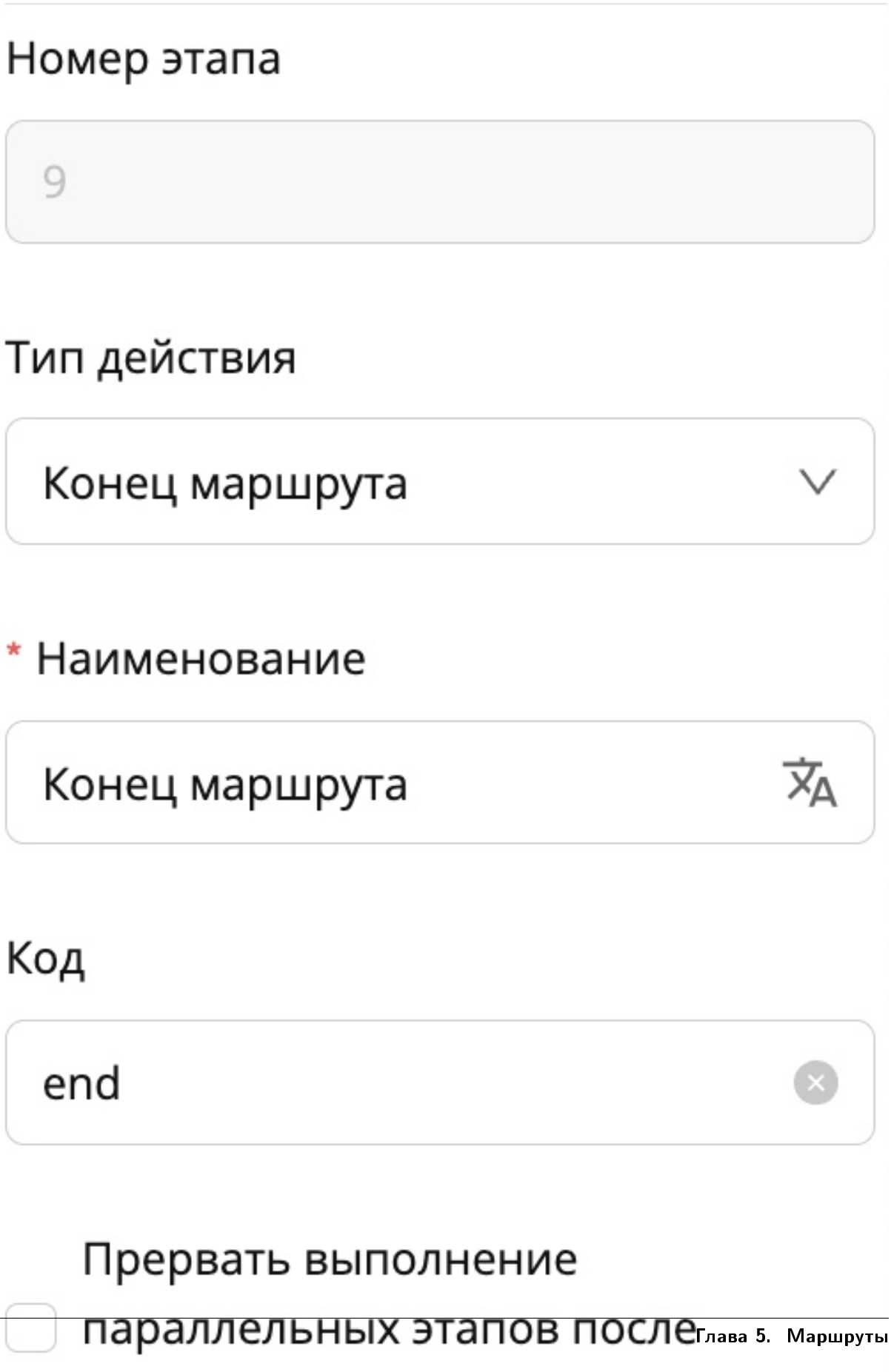
Данный этап завершает выполнение бизнес-процесса заявки.

Шаг 1. В редакторе маршрута нажимаем кнопку «+» для добавления нового этапа.

Шаг 2. В настройках этапа указываем:

- Тип действия — Конец маршрута;
- Наименование — Конец маршрута;
- Код этапа — end (указанный нами ранее в условном переходе)

Шаг 3. Сохраняем этап.


Номер этапа
9
Тип действия
Конец маршрута
* Наименование
Конец маршрута
Код
end
Прервать выполнение параллельных этапов после завершения одного из них

Глава 6

Архитектурные и эксплуатационные рекомендации

6.1 Архитектурные рекомендации

Данный раздел содержит архитектурные рекомендации, основанные на практическом опыте эксплуатации платформы и экономически обоснованных требованиях к аппаратному и программному обеспечению.

Рекомендации ориентированы на разные сценарии использования и учитывают как нагрузочные характеристики, так и требования к масштабируемости и надежности.

6.1.1 Минимальные требования к инфраструктуре

При выборе инфраструктуры рекомендуется исходить из целевого сценария использования платформы и предполагаемой нагрузки.

Enterprise Instance

Enterprise Instance предназначен для промышленной эксплуатации и рассчитан на нагрузку до **1000 именованных пользователей** и до **300 конкурентных пользователей**.

Целевой показатель SLA — время отклика до **3 секунд** при стабильной нагрузке.

Минимальные требования к инфраструктуре:

- **CPU**: 16 vCPU
- **RAM**: 64 GB
- **Storage**: 1 TB NVMe
- **Network**: до 20 TB трафика

Оценочная стоимость аренды виртуальной машины с такими характеристиками составляет порядка **150–200 USD в месяц**.

Данный вариант рекомендуется для production-окружений и корпоративных внедрений.

Dev Instance

Dev Instance предназначен для:

- разработки;
- демонстрации решений;
- показа кейсов и прототипов.

Рассчитан на нагрузку до **5 конкурентных пользователей**.

Минимальные требования:

- **CPU**: 4 vCPU
- **RAM**: 8 GB
- **Storage**: ~25 GB
- **Network**: минимальные, зависят от сценария использования

В качестве Dev Instance может использоваться рабочий ноутбук разработчика или недорогая виртуальная машина.

6.1.2 Мультиинстанс-архитектура (process-to-process)

В крупных организациях с выраженной организационной структурой (головная компания, дочерние организации, филиалы, департаменты) рекомендуется использовать мультиинстанс-подход.

Суть подхода заключается в том, что экземпляры Synergy разворачиваются по логике организационной структуры: отдельный инстанс - на отдельную организационную единицу (например, на каждую дочернюю организацию или крупный контур).

Данный подход позволяет:

- изолировать процессы и данные между организационными единицами;
- повысить уровень безопасности за счет разделения контуров доступа;
- повысить отказоустойчивость (сбой в одном инстансе не останавливает работу остальных);
- упростить масштабирование за счет распределения нагрузки по нескольким инстансам;
- поддерживать независимые циклы изменений и сопровождения для разных контуров.

При использовании мультиинстанс-архитектуры допускается построение process-to-process взаимодействия между инстансами, когда выполнение процесса в одном инстансе может инициировать продолжение или отдельный участок процесса в другом инстансе (в рамках общей бизнес-логики компании).

Сводный инстанс

Дополнительно может быть выделен **сводный инстанс**, который используется как общий уровень для:

- централизованных справочников;
- сводной отчетности и аналитики;
- консолидации результатов процессов из дочерних контуров.

Сводный инстанс позволяет обеспечить единые источники данных и единый слой отчетности, не нарушая при этом изоляцию и автономность рабочих инстансов.

Примечание: Выбор мультиинстанс-архитектуры рекомендуется при наличии нескольких организационных контуров и необходимости разделения доступа, ответственности и нагрузки.

6.1.3 Архитектурные рекомендации по пользовательским порталам

При проектировании внешних пользовательских порталов рекомендуется учитывать количество конкурентных пользователей и профиль нагрузки.

Сценарий до 1000 конкурентных пользователей

При нагрузке до **1000 конкурентных пользователей** рекомендуется использовать стандартные средства платформы Synergy.

В данном сценарии:

- пользовательский портал реализуется непосредственно на Synergy;
- формы, бизнес-валидации, маршруты и транзакции выполняются внутри платформы;
- инфраструктурные требования минимальны;
- профиль нагрузки предсказуем.

Такой подход обеспечивает:

- быстрый **time-to-market**;
- упрощенную архитектуру;
- снижение эксплуатационных затрат.

Сценарий свыше 1000 конкурентных пользователей

Нагрузка свыше **1000 конкурентных пользователей** относится к классу **highload** и требует отдельного архитектурного проектирования.

В данном случае рекомендуется:

- разрабатывать внешний пользовательский портал как самостоятельное highload-приложение;
- применять горизонтальное масштабирование;
- проектировать архитектуру под конкретный профиль нагрузки.

Взаимодействие с Synergy в этом сценарии должно выполняться:

- асинхронно;
- через очереди сообщений;
- с обязательным использованием **rate limiting**;
- исключительно через API и события.

В данной архитектуре Synergy выступает в роли **процессного и рабочего ядра**, а внешний портал берет на себя всю нагрузку пользовательского взаимодействия.

6.1.4 Общие рекомендации

При выборе архитектурного подхода рекомендуется:

- четко определять целевой профиль нагрузки;
- избегать преждевременной оптимизации;
- использовать стандартные средства Synergy при умеренных нагрузках;
- выносить highload-компоненты во внешние сервисы только при наличии объективной необходимости.

Соблюдение данных рекомендаций позволяет достичь оптимального баланса между стоимостью владения, производительностью и масштабируемостью системы.

6.2 Рекомендации по эксплуатации

Данный раздел содержит рекомендации по эксплуатации приложений, развернутых на платформе Synergy.

Рекомендации охватывают ключевые эксплуатационные аспекты, такие как перенос приложений между средами, резервное копирование и восстановление, а также общие подходы к сопровождению системы в промышленной эксплуатации.

6.2.1 Импорт и экспорт приложений

Платформа Synergy предоставляет встроенные механизмы импорта и экспорта приложений, предназначенные для переноса конфигурации между различными окружениями (например: $dev \rightarrow test \rightarrow prod$).

Экспорт приложения позволяет:

- сохранить структуру приложения и все объекты конфигурации;
- зафиксировать состояние приложения на определенной версии;
- перенести приложение на другой экземпляр платформы;
- использовать экспорт как элемент процессов CI/CD.

Импорт приложения обеспечивает:

- восстановление приложения из архива;
- развертывание приложения на новом сервере;
- синхронизацию конфигурации между средами;
- перенос решений между инсталляциями заказчика.

Поддерживаемые сценарии

Механизм импорта и экспорта поддерживает следующие сценарии:

- экспорт приложения полностью или частично;
- экспорт с данными или без данных;
- контроль целостности выгрузки;

- импорт приложения с сохранением зависимостей и структуры.

Детальное описание механизма, пошаговые инструкции и ограничения приведены в официальной документации платформы Synergy:

http://tdd.lan.arta.kz/docs/synergy/tags/hamming/release-notes/html/app_export_import.html

Рекомендации по использованию

При эксплуатации рекомендуется:

- использовать экспорт приложений как основной способ переноса конфигурации между средами;
- не выполнять ручные изменения конфигурации напрямую в production-среде;
- хранить экспортируемые архивы приложений в системе контроля версий или защищенном хранилище;
- привязывать экспорт приложений к версиям релизов и этапам CI/CD.

6.2.2 Резервное копирование и восстановление

Резервное копирование является обязательным элементом эксплуатации платформы Synergy и направлено на обеспечение сохранности данных и возможности восстановления системы в случае сбоев или аварий.

Платформа поддерживает резервное копирование всех критически важных компонентов системы, включая данные, хранилища и конфигурацию.

Объекты резервного копирования

В резервное копирование должны входить:

- реляционные базы данных платформы;
- файловое хранилище (JackRabbit или Cassandra в зависимости от типа инсталляции);
- индексы поиска (Lucene или Elasticsearch);
- конфигурационные файлы платформы и окружения.

Общие рекомендации

При организации резервного копирования рекомендуется:

- выполнять резервное копирование при остановленных сервисах платформы для обеспечения целостности данных;
- хранить резервные копии с привязкой к дате и версии системы;
- регулярно проверять возможность восстановления из резервных копий;
- разделять резервные копии production- и тестовых окружений.

Подробные процедуры резервного копирования и восстановления системы описаны в эксплуатационной документации платформы.

6.2.3 Эксплуатационные практики

Для стабильной и предсказуемой работы системы рекомендуется:

- разделять окружения разработки, тестирования и эксплуатации;
- минимизировать прямые изменения в production;
- использовать импорт/экспорт приложений как основной механизм обновлений;
- регулярно контролировать состояние хранилищ и индексов;
- планировать резервное копирование как часть регламентных работ.

Соблюдение данных рекомендаций позволяет снизить эксплуатационные риски, обеспечить устойчивость системы и упростить сопровождение приложений на протяжении всего жизненного цикла.