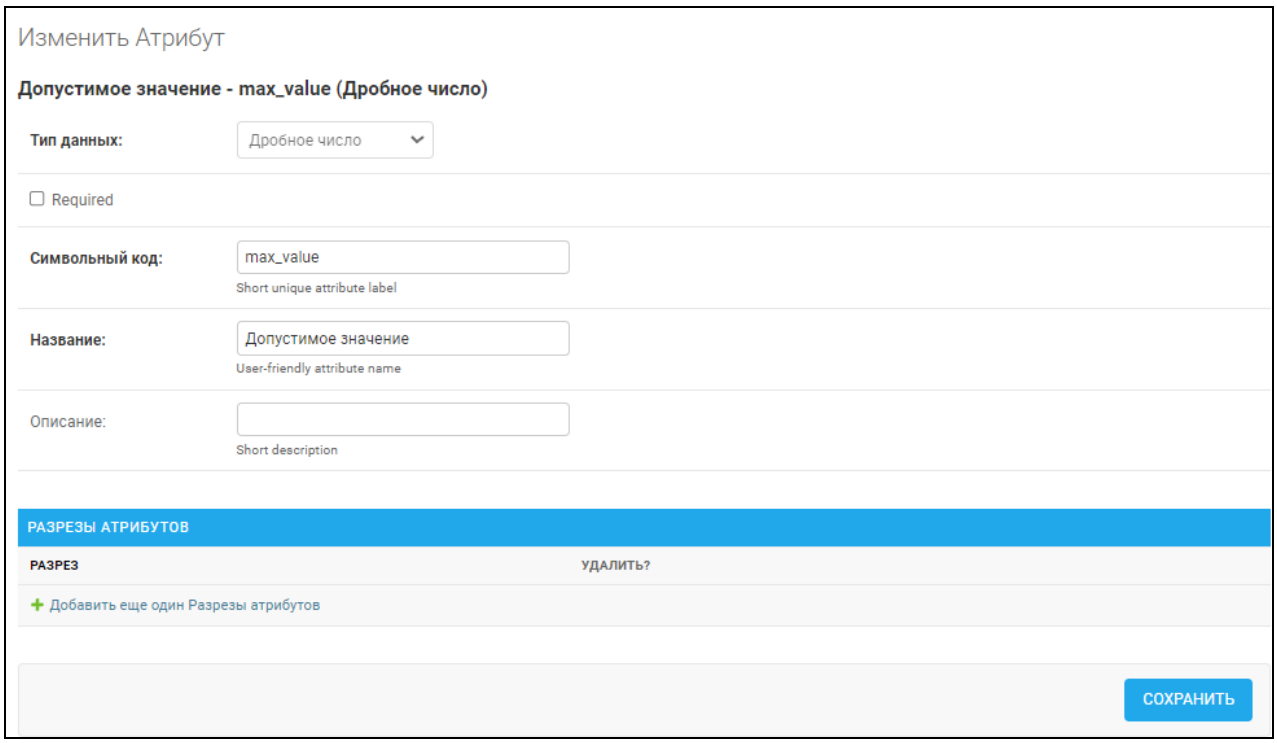


Выбор группы из списка

Рис. 5

5. Сохраните изменения в форме создания типа представления.

4.3.3. Создание и настройка формы списка

В форме списка отображается список объектов выбранного типа представления.

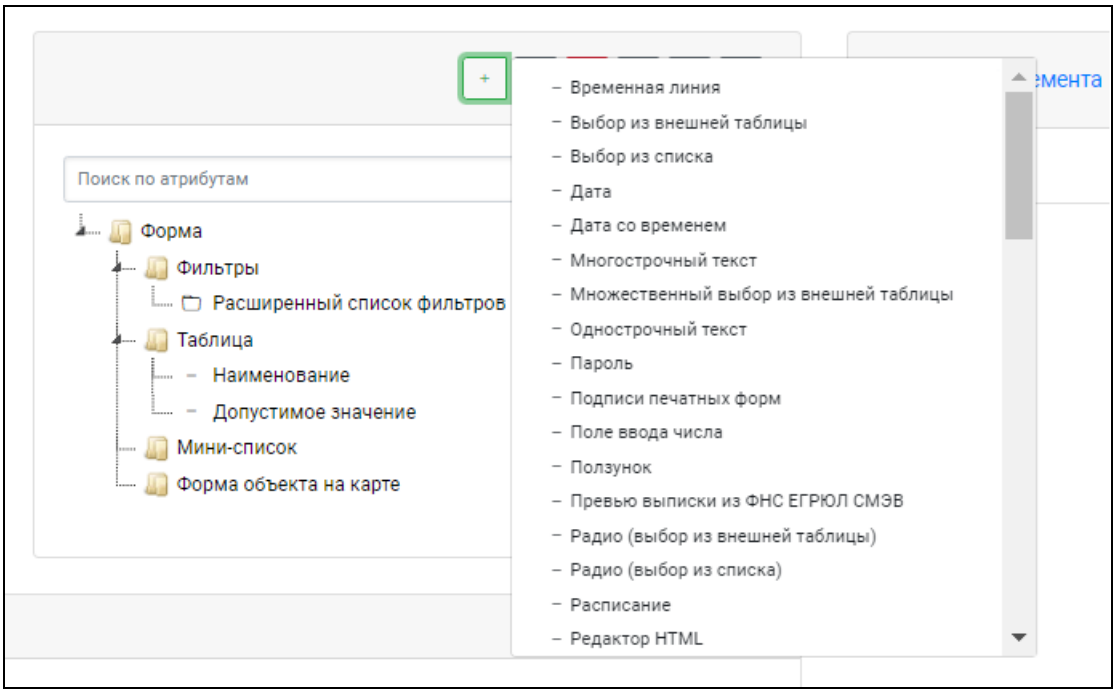
Форма списка создается автоматически после создания типа представления. Для того чтобы настроить форму списка, необходимо выполнить следующие действия:

1. Перейдите в раздел конструктора «Конструктор форм → 1. Формы». Найдите форму списка по созданному ранее типу представления.

2. В подразделе формы «Конструктор» в блоке «Структура данных» отображаются атрибуты типа представления, доступные для добавления на форму списка. В блоке посередине отражается древовидная структура формы. Чтобы вывести атрибут на форму в виде поля:

– в среднем блоке нажмите на папку в структуре, куда необходимо добавить атрибут.

При этом можно создавать новые группы и добавлять новые элементы (рис. 6);

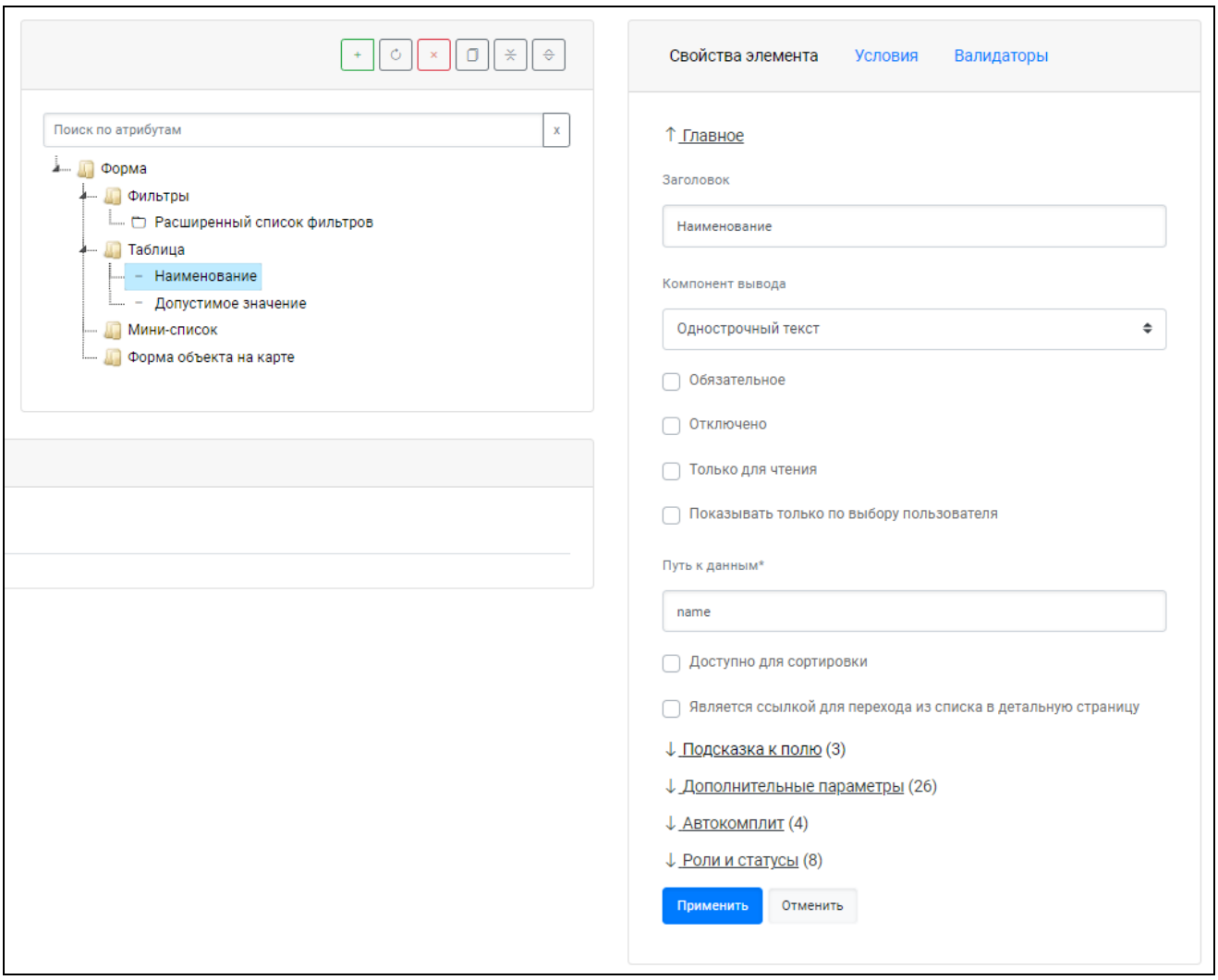


Форма создания новой группы и добавления нового элемента

Рис. 6

– в блоке «Структура данных» найдите атрибут, который необходимо добавить на форму списка. Двойным кликом по атрибуту добавьте его на форму. Атрибут отобразится в среднем блоке в папке, которая была выбрана до добавления. При этом атрибуты можно перемещать по структуре перетаскиванием.

3. Нажмите на добавленный атрибут и настройте его отображение (поле в форме) в третьем блоке конструктора формы (рис. 7). Описание настроек отображения полей приведено в п. 4.3.1 настоящего документа.



Настройка отображения добавленного атрибута

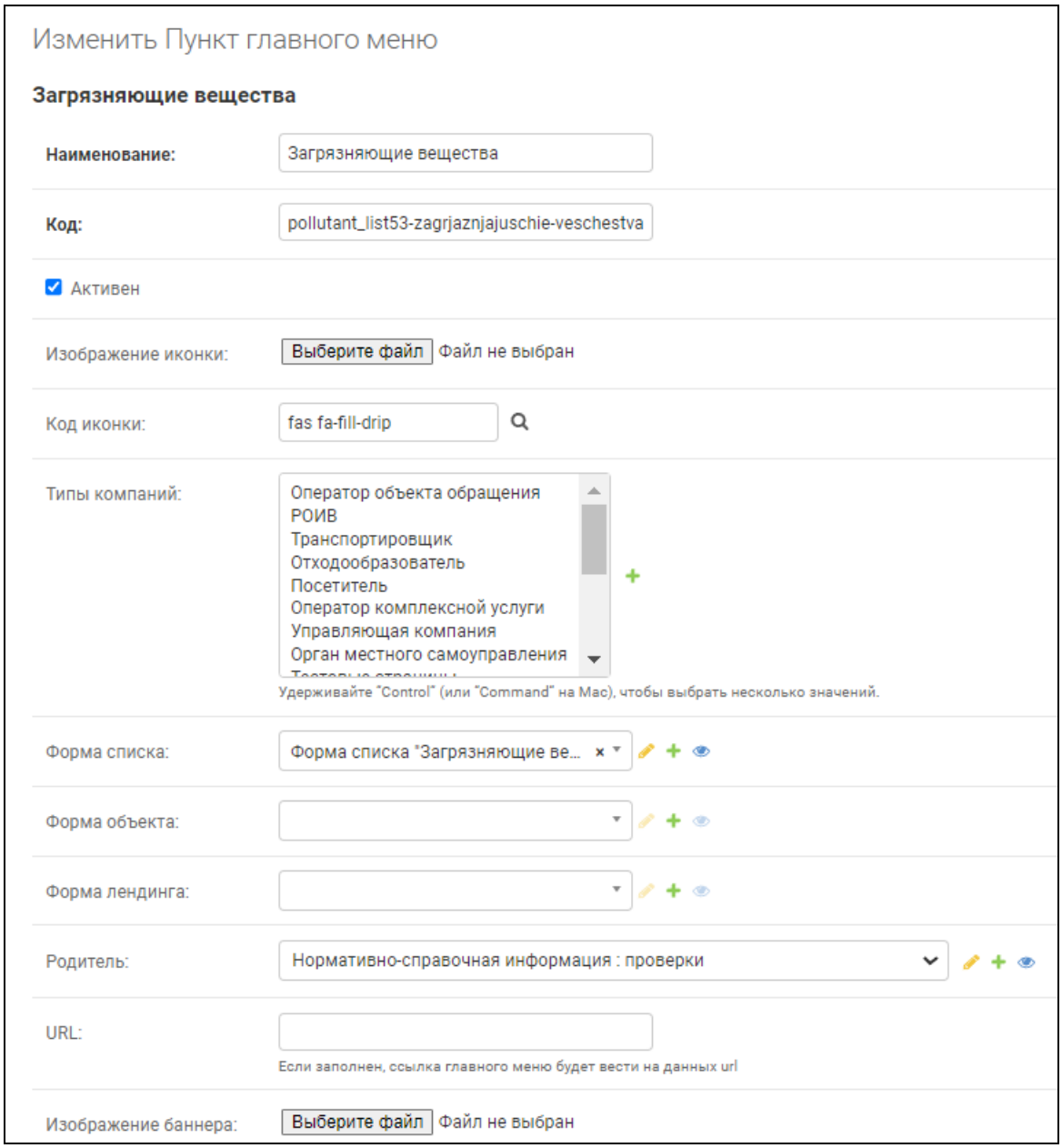
Рис. 7

4. Выведите форму в меню Личного кабинета. Для этого:

– перейдите в раздел конструктора «Конструктор форм → 2. Главное меню»;

– нажмите кнопку «Добавить пункт главного меню»;

– заполните форму (рис. 8):



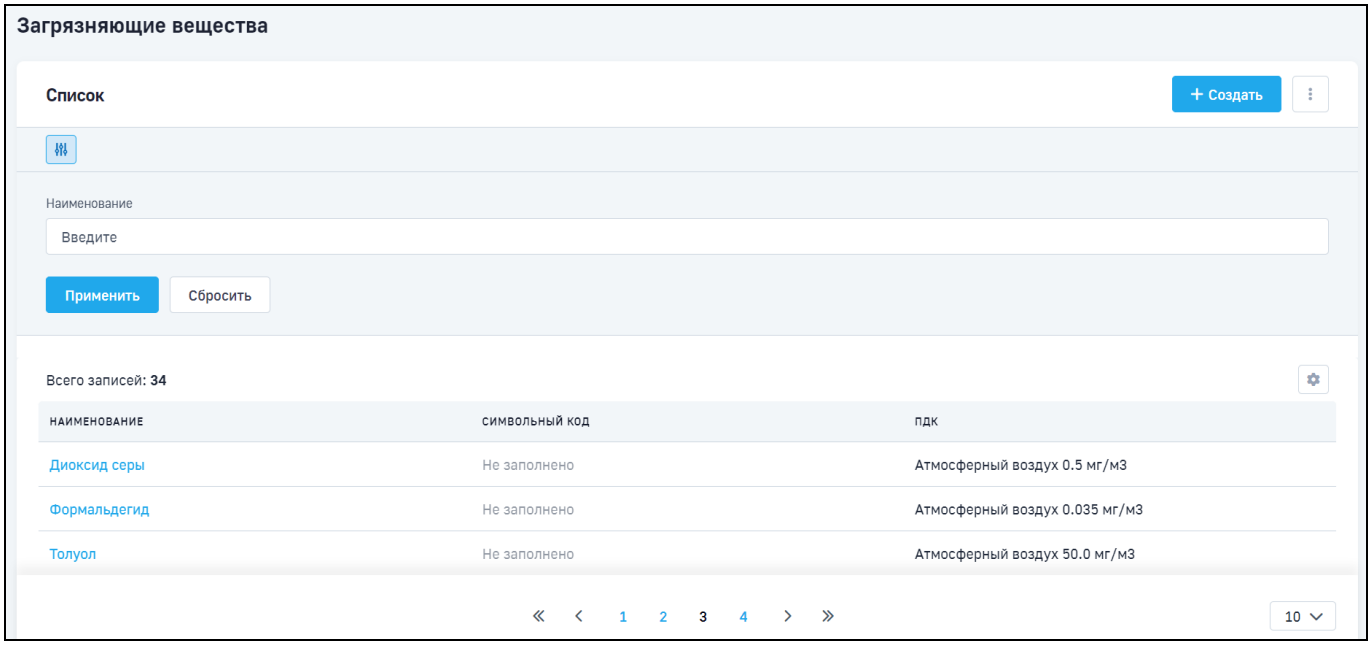
Пример заполнения формы пункта главного меню

Рис. 8

- в поле «Наименование» укажите удобное для пользователя название пункта меню, которое будет отображаться в интерфейсе,
- в поле «Код» укажите уникальный код пункта меню, используя буквы латинского алфавита,
- в поле «Активен» поставьте отметку, чтобы пункт меню отображался в интерфейсе,
- в поле «Код иконки», нажав кнопку «🔍» можно выбрать иконку из уже доступных в Платформе или загрузить новую,

- в поле «Типы компаний» выберите все типы компаний, для которых будет доступен данный пункт меню,
- в поле «Форма списка» выберите форму списка, которая должна открываться при переходе по пункту меню,
- в поле «Родитель» выберите родительский пункт меню, если необходимо, чтобы создаваемый пункт меню был вложенным (не первого уровня),
- в поле «URL» при необходимости укажите адрес (можно как внутри информационной системы, так и внешний), на который будет осуществляться переход из пункта меню,
- в поле «Индекс сортировки» укажите число, от которого будет зависеть расположение пункта меню среди других пунктов. Чем меньше число, тем выше пункт в списке,
- в поле «Только для ролей» выберите все роли, для которых будет доступен данный пункт меню.

5. Форму списка теперь можно просматривать в личном кабинете (рис. 9). При нажатии на элемент списка будет открываться детальная страница, если она создана для данного типа представления.



Просмотр формы списка через личный кабинет пользователя

Рис. 9

4.3.4. Создание и настройка детальной страницы

Детальная страница отражает данные по одному объекту выбранного типа представления.

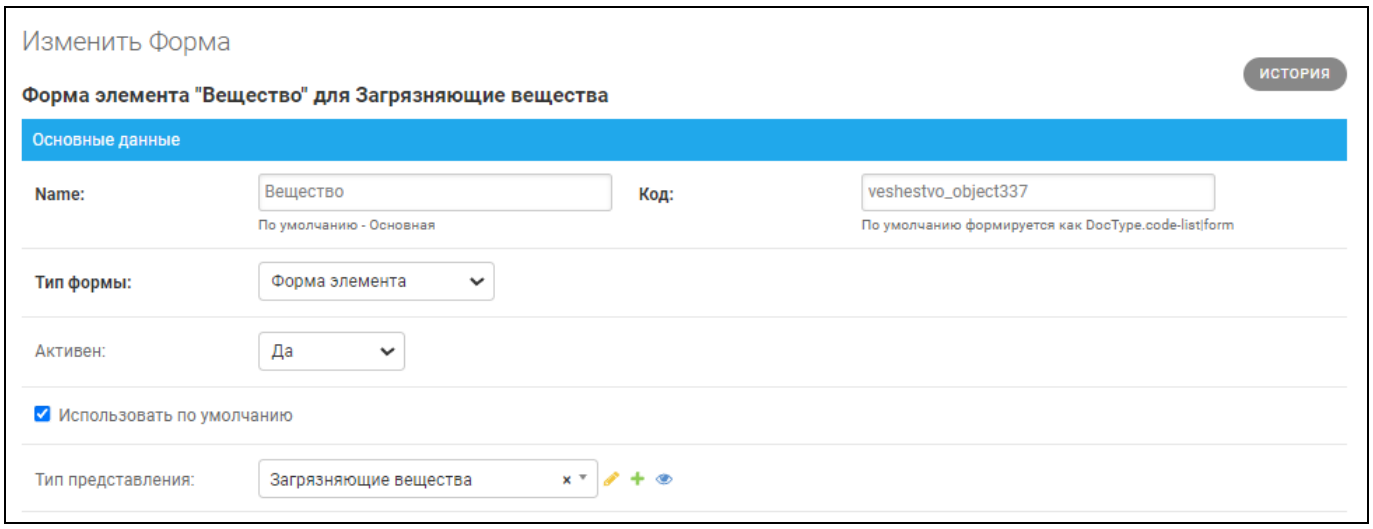
Детальная страница также создается автоматически после создания типа представления и привязывается к форме списка, также созданной автоматически. Т.е. данная форма будет открываться при нажатии на какую-либо запись в форме списка.

Кроме этого, для одного типа представления можно создать несколько детальных страниц и привязать в форме списка нужную детальную страницу.

Для создания детальной страницы необходимо выполнить следующие действия:

1. Перейдите в раздел конструктора «Конструктор форм → 1. Формы».

2. Нажмите кнопку «Добавить форму». Заполните форму (рис. 10):



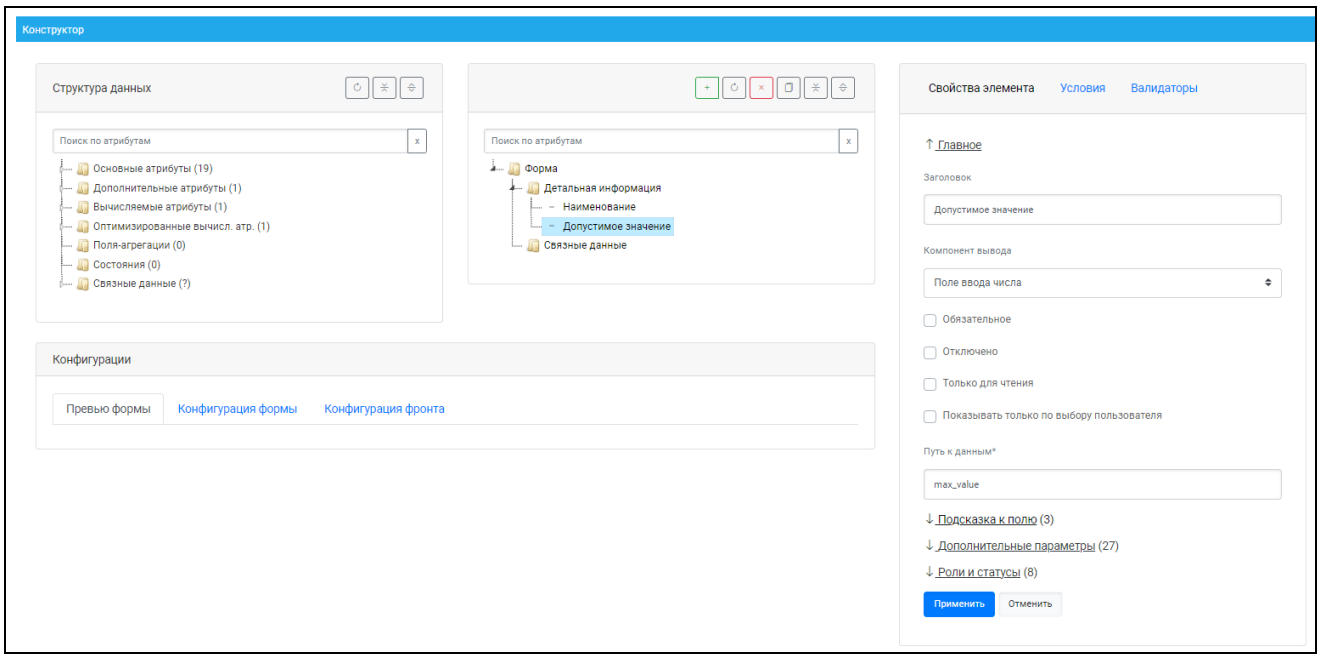
Заполнение формы при создании детальной страницы

Рис. 10

– в поле «Name» укажите удобное для пользователя название формы;
– в поле «Код» можно указать уникальный код формы или оставить значение, сформированное автоматически;
– в поле «Тип формы» выберите «Форма элемента»;
– в поле «Активен» оставьте выбранное по умолчанию значение «Да», если необходимо, чтобы форма отображалась в интерфейсе;
– в поле «Тип представления» выберите тип представления, для которого необходимо создать детальную страницу.

3. Сохраните изменения.

4. После сохранения формы станут доступны расширенные настройки формы. В разделе «Конструктор» добавьте и настройте необходимые поля в форме (рис. 11). Описание настроек отображения полей приведено в п. 4.3.1 настоящего документа.



Настройка полей в форме детальной страницы

Рис. 11

5. Снова сохраните изменения.

4.4. Использование формул в конструкторе

4.4.1. Общее описание и контексты данных

В конструкторе реализована возможность обработки данных формы с помощью формул. Например, можно вывести значение поля в зависимости от выполнения условий; вычислить значение поля, используя различные арифметические, статистические функции; можно скрыть поле, сделать его обязательным, изменить его внешний вид в зависимости от значений других полей и т.п.

За обработку формул в конструкторе отвечает фронтенд-модуль «Агент формул». Поскольку формулы выполняются на серверной части Платформы, то изменения, которые вносит пользователь, отражаются в информационной системе сразу.

Формулы могут быть добавлены в свойства и условия атрибутов.

Формулы оформляются в фигурных скобках с соблюдением синтаксиса библиотеки math.js.

В формулах можно обращаться к различным данным из форм и объектов (например, использовать значение другого поля формы или получить данные пользователя). Для обращения к таким данным используются контексты данных.

Контекст данных – объекты, в которых хранятся данные, к которым формулы имеют доступ. В разных типах форм доступны разные контексты данных. На текущий момент, в формах доступны следующие контексты:

Форма сущности (форма и поля формы):

data – текущая сущность;

user – объект текущего пользователя;

app – контекст для определения мобильной версии.

Форма перехода статуса сущности (поля формы):

data – текущая сущность;

user – объект текущего пользователя;

app – контекст для определения мобильной версии.

Форма списка (только форма):

data – родительская сущность списка, если есть (иначе контекст недоступен и при обращении к нему будет появляться ошибка);

user – объект текущего пользователя;

app – контекст для определения мобильной версии;

filter – форма фильтра;

filterSaved – сохраненные данные формы фильтра;

parentFilter – родительская форма фильтра на мультикарте, если есть (иначе контекст недоступен и при обращении к нему будет появляться ошибка);

item – сущность элемента списка (недоступен в формулах, но доступен в коде для того, чтобы получить зависимости от данных элемента списка).

Форма элемента списка (форма и поля формы):

data – форма текущего элемента списка;

user – объект текущего пользователя;

app – контекст для определения мобильной версии;

parent – родительская сущность списка, если есть (иначе контекст недоступен и при обращении к нему будет появляться ошибка);

filter – форма фильтра списка;

filterSaved – сохраненные данные формы фильтра;

parentFilter – родительская форма фильтра на мультикарте, если есть (иначе контекст недоступен и при обращении к нему будет появляться ошибка).

Форма фильтра списка (поля формы):

data – форма фильтра;

user – объект текущего пользователя;

app – контекст для определения мобильной версии;

filterSaved – сохраненные данные формы фильтра;

parent – родительская сущность списка, если есть (иначе контекст недоступен и при обращении к нему будет появляться ошибка);

parentFilter – родительская форма фильтра на мультикарте, если есть (иначе контекст недоступен и при обращении к нему будет появляться ошибка).

Форма родительского фильтра списка (поля формы):

data – форма фильтра;

user – объект текущего пользователя;

app – контекст для определения мобильной версии.

Форма операции (поля формы):

data – форма операции;

user – объект текущего пользователя;

app – контекст для определения мобильной версии.

В большинстве случаев одноименные контексты несут одну и ту же логику. Однако необходимо обращать внимание на тип формы, в котором создаются формулы, т.к. в некоторых случаях логика может отличаться:

data – в большинстве случаев означает контекст текущей формы, за исключением случаев, когда у формы есть родительская форма. В контекст data попадает модель данных, которая описывается ObjectModel формы;

user – во всех формах означает объект текущего пользователя. В контекст попадает объект пользователя с данными, которые получены после запроса на api/auth;

app – контекст app доступен во всех формах и полях форм (также, как контекст user);

filter – все, что попадает в контекст data в форме фильтра, является контекстом filter в остальных формах, где доступен данный контекст. В контекст попадают данные, которые введены в форму фильтра и при этом еще не применены. При вводе значения в поле фильтра агент формул сразу пересчитывает формулы, в которых используется данное поле;

filterSaved – в данный контекст попадают значения полей фильтра после применения. Если после применения фильтра будут внесены изменения в поля фильтра, но без повторного применения, то будет меняться только контекст filter;

parent – все, что попадает в контекст data в форме сущности, является контекстом parent, если сущность является родительской для сущности списка. В некоторых видах форм такие данные находятся в контексте data. Если у формы нет родительской сущности, то при обращении к данному контексту будет выводиться ошибка;

parentFilter – контекст доступен только для мультикарт (страниц, на которых находится несколько форм списков). Включает в себя фильтр, доступный для всех форм списков на мультикарте. Основные операторы, доступные в формулах.

На Платформе при создании формул можно использовать любые операторы из библиотеки math.js.

В таблице 4 представлены основные операторы, которые могут быть полезны при работе с Платформой.

Таблица 4 – Основные операторы данных

Оператор	Название	Пример использования
,	Разделитель параметров и элементов	max(2, 1, 5)
a(b)	Вызов метода	str(1, 2) → "12"
{}	Создание объекта	{a: 1, b: 2}
[]	Создание массива	[1, 2, 3]
(a)	Группировка части выражения	2 * (3 + 4) → 14
a.b	Обращение к свойствам	obj={a: 12}; obj.a → 12
a[b]	Динамическое обращение к свойствам	obj={a: 12}; x='a'; obj[x] → 12
a ? b : c	Условное выражение, где: a – условие, b – значение, если условие истинно, c – если ложно	15 > 100 ? 1 : -1 → -1
a and b	Конъюнкция	true and false → false
a or b	Дизъюнкция	true or false → true

Оператор	Название	Пример использования
a xor b	Строгая дизъюнкция	true xor true → false
not a	Отрицание	not true → false
a + b	Сложение	4 + 5 → 9
a - b	Вычитание	7 - 3 → 4
a * b	Умножение	2 * 3 → 6
a / b	Деление	6 / 2 → 3
a ^ b	Возведение в степень	2 ^ 3 → 8
a % b	Остаток	8 % 3 → 2
a%	Процент	8% → 0.08
+a	Унарный плюс. Приведение к числу	+" 17.56" → 17.56
-a	Унарный минус	-" 17.56" → -17.56
a == b	Равенство	2 == 4 - 2 → true
a != b	Неравенство	2 != 3 → true
a < b	Меньше	2 < 3 → true
a > b	Больше	2 > 3 → false
a <= b	Меньше или равно	4 <= 3 → false
a >= b	Больше или равно	2 + 4 >= 6 → true

4.4.2. Доступные методы в формулах

На Платформе при создании формул используются большинство методов из библиотеки math.js. Кроме этого, имеются методы, реализованные непосредственно на Платформе.

В таблице 5 представлены основные методы в формулах, которые могут быть полезны при работе с Платформой.

Таблица 5 – Основные методы в формулах

Функция	Название	Параметры	Пример использования
str(x, y, z…): string	Воспринимает все переданные аргументы как строки и конкатенирует их в одну строку, которая возвращается в результате. Конвертирует переданные аргументы в строку, если это не строки	x, y, z – параметры в формате строк, чисел, обращений к объектам и т.п.	str('Строка') // 'Строка'; str('Строка1', 'Строка2') // 'Строка1Строка2'; str(user.firstName, ' ', user.lastName) // 'Иван Иванов'
date(raw): FormcyMoment	Создание объекта FormcyMoment	Необязательный параметр **raw** – значение в формате date, datetime или local_moment. Если ничего не указано, то будет использована текущая дата и время	
has(object: object, prop: string): boolean	Проверяет, что в объекте есть свойство. Вернет true, если в object есть свойство с названием, указанным в prop. Иначе вернет false		has(user.roles, 'admin') // true/false, has(data.roles, 0) // вернет true, если есть хотя бы один элемент в списке data.roles
print(template, values): string	Интерполирует значения values в строковый шаблон template	**template**: string – строка, содержащая плейхолдеры переменных; **values**: Object/Array/Matrix – объект или массив, содержащий переменные, которые будут заполнены в шаблоне	print('За год было проведено $num проверок', {num: 1235}) // 'За год было проведено 1235 проверок'
boolean(x): boolean / Array / Matrix	Создает логическое значение или преобразует строку или число в логическое значение	**x**: string / number / boolean / Array / Matrix / null	boolean([1, 0, 1, 1]) // [true, false, true, true]
number(x): number / Array / Matrix	Создает число или преобразует строку, логическое значение или единицу измерения в число	**x**: string / number / BigNumber / Fraction / boolean / Array / Matrix / Unit / null	number([true, false, true, true]) // [1, 0, 1, 1]
distance([x1, y1], [x2, y2]):	Вычисляет евклидово	**x**: Array / Matrix / Object – координаты первой	distance({pointOneX: 0,

Функция	Название	Параметры	Пример использования
Number / BigNumber	расстояние между двумя точками в N-мерных пространствах, расстояние между точкой и линией в 2-х и 3-х мерных пространствах, попарное расстояние между набором 2D- или 3D-точек	точки, **y**: Array / Matrix / Object – координаты второй точки	pointOneY: 0}, {pointTwoX: 10, pointTwoY: 10}) // 14.142135623730951
pickRandom(array):number / Array	Случайный выбор одного или нескольких значений из одномерного массива. Элементы массива выбираются случайным образом с равномерным или взвешенным распределением	**array** – одномерный массив	pickRandom([3, 6, 12, 2]) // возвращает одно из значений массива
random([min, max]): number / Array / Matrix	Используя равномерное распределение, возвращает случайное число, большее или равное min и меньшее max	**min**: number – минимальная граница случайного значения, включая; **max**: number – максимальная граница случайного значения, исключая	random(30, 40) // возвращает случайное число от 30 до 40
randomInt([min, max])	Используя равномерное распределение, возвращает случайное целое число, большее или равное min и меньшее max	**min**: number – минимальная граница случайного значения, включая; **max**: number – максимальная граница случайного значения, исключая	randomInt(30, 40) // возвращает случайное целое число от 30 до 40
Арифметические функции	см. Приложение А настоящего документа		
Статистические функции	см. Приложение Б настоящего документа		
Тригонометрические функции	см. Приложение В настоящего документа		

4.4.3. Доступные методы в объекте FormcyMoment

FormcyMoment – объект, который связан с какой-то конкретной датой, временем и таймзоной. Объект FormcyMoment имеет несколько полезных методов для манипуляции и форматирования даты и времени.

Можно использовать большинство методов из библиотеки moment.js. Также есть методы, реализованные непосредственно на Платформе.

В таблице 6 представлены основные методы в объекте FormcyMoment, которые могут быть полезны при работе с Платформой.

Таблица 6 – Основные методы в объекте FormcyMoment

Функция	Название	Параметры	Пример использования
format(string): string	Форматирование даты в удобочитаемый вид с учетом текущей локали для интернационализации	**format** – формат данных, доступный в Angular DatePipe. Это может быть настраиваемый формат или один из готовых пресетов. Готовые пресеты форматов учитывают интернационализацию с учетом текущей локали. По умолчанию «short», если не указано	date('2022-01-01T12:30Z').format('short') // '01.01.2022, 12:30'
toSrc(‘datetime’ | ‘date’): string	Форматирование даты в формат конкретного типа данных	**format** – тип данных, который может использоваться для дат в моделях данных, т.е. в хранилище данных и для обмена между сервером и клиентской части Платформы. Поддерживаемые типы: «date», «datetime». По умолчанию «datetime», если не указано или указано неверно	date('2022-01-01T12:30Z').toSrc('date') // '2022-01-01'
add(number, string): FormcyMoment	Добавляет время к времени объекта FormcyMoment	**number** – количество, которое необходимо добавить; **string** – единица времени, которое необходимо добавить	add(7, 'days') // добавит 7 дней
subtract(number, string): FormcyMoment	Вычитает время из времени объекта FormcyMoment	**number** – количество, которое необходимо вычесть; **string** – единица времени, которое необходимо вычесть	subtract(7, 'days') // вычтет 7 дней
startOf(string): FormcyMoment	Устанавливает время объекта FormcyMoment на начало единицы времени	**string** – единица времени в строковом формате	startOf('year')// устанавливает дату на 1 января, 00:00 этого года
endOf(string): FormcyMoment	Устанавливает время объекта FormcyMoment на	**string** – единица времени в строковом формате	endOf('year')// устанавливает дату на 31 декабря,

Функция	Название	Параметры	Пример использования
	конец единицы времени		23:59:59.999 этого года
get(string)	Возвращает значение указанной единицы времени в объекте FormcyMoment	**string** – единица времени в строковом формате	date('2022-01-01T12:30Z').get('year')// возвращает число 2022
set(string, int): FormcyMoment	Устанавливает указанное значение единицы времени объекту FormcyMoment	**string** – единица времени в строковом формате; **int** – значение, которое нужно установить единице времени	set('year', 2013)// устанавливает в объекте FormcyMoment 2013 год
millisecond()	Задает или получает миллисекунды времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается указанное число миллисекунд. Принимаются числа от 0 до 999. Если диапазон превышен, он будет увеличиваться до секунд. Если параметр не задан, то возвращает миллисекунды времени объекта FormcyMoment	millisecond(123)// устанавливает в объекте FormcyMoment значение миллисекунд = 123 date([2010, 1, 14, 15, 25, 50, 33]).millisecond()// возвращает число 33
second()	Задает или получает секунды времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается указанное число секунд. Принимаются числа от 0 до 59. Если диапазон превышен, он будет увеличиваться до минут. Если параметр не задан, то возвращает секунды времени объекта FormcyMoment	second(42)// устанавливает в объекте FormcyMoment значение секунд = 42 date([2010, 1, 14, 15, 25, 50, 33]).second()// возвращает число 50
minute()	Задает или получает минуты времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается указанное число минут. Принимаются числа от 0 до 59. Если диапазон превышен, он будет увеличиваться до часов. Если параметр не задан, то возвращает минуты	minute(42)// устанавливает в объекте FormcyMoment значение минут = 42 date([2010, 1, 14, 15, 25, 50, 33]).second()// возвращает число 25

Функция	Название	Параметры	Пример использования
		времени объекта FormcyMoment	
hour()	Задает или получает часы времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается указанное число часов. Принимаются числа от 0 до 23. Если диапазон превышен, он будет увеличиваться до дней. Если параметр не задан, то возвращает часы времени объекта FormcyMoment	hour(2)// устанавливает в объекте FormcyMoment значение часа =2 date([2010, 1, 14, 15, 25, 50, 33]).second()// возвращает число 15
date()	Задает или получает день месяца	Если передается параметр **number**, то в день месяца устанавливается указанное число. Принимаются числа от 1 до 31. Если диапазон превышен, он будет увеличиваться до месяцев. Если параметр не задан, то возвращает день месяца времени объекта FormcyMoment	date(2)// устанавливает в объекте FormcyMoment значение числа месяца = 2 date([2010, 1, 14, 15, 25, 50, 33]).second()// возвращает число 14
dayOfWeek()	Получает или задает день недели времени объекта FormcyMoment по стандарту ISO, где 1 – понедельник, а 7 – воскресенье	Если передается параметр **number**, то устанавливается соответствующий день недели. Принимаются числа от 1 до 7. Если параметр не задан, то возвращает день недели времени объекта FormcyMoment	dayOfWeek(1) // устанавливает в объекте FormcyMoment значение дня недели = Понедельник. При этом дата автоматически устанавливается в соответствии с заданным днем недели (устанавливается дата этого дня недели на неделе времени объекта FormcyMoment). date([2010, 1, 14, 15, 25, 50, 33]).dayOfWeek()// возвращает число 7
dayOfYear()	Получает или задает день	Если передается параметр **number**, то	dayOfYear(13)// устанавливает

Функция	Название	Параметры	Пример использования
	года времени объекта FormcyMoment	устанавливается соответствующий день года. Принимаются числа от 1 до 366. Если параметр не задан, то возвращает день года времени объекта FormcyMoment	в объекте FormcyMoment дату 13 января date([2010, 1, 14, 15, 25, 50, 33]).dayOfYear()// возвращает число 45
weekOfYear()	Получает или задает неделю года времени объекта FormcyMoment по стандарту ISO	Если передается параметр **number**, то устанавливается соответствующая неделя года. Если параметр не задан, то возвращает неделю года времени объекта FormcyMoment	weekOfYear(1) // устанавливает в объекте FormcyMoment последнюю дату 1й недели года (7 января) date([2010, 1, 14, 15, 25, 50, 33]).weekOfYear()// возвращает число 6
month()	Получает или задает месяц времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается соответствующий месяц. Принимаются числа от 0 до 11 (0 – январь, 11 – декабрь). Если параметр не задан, то возвращает месяц времени объекта FormcyMoment	month(10) // устанавливает в объекте FormcyMoment месяц ноябрь date([2010, 1, 14, 15, 25, 50, 33]).month()// возвращает число 1
quarter()	Получает или задает квартал времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается соответствующий квартал. Принимаются числа от 1 до 4. Если параметр не задан, то возвращает квартал времени объекта FormcyMoment	quarter(2) // устанавливает в объекте FormcyMoment первый месяц 2го квартала (апрель) date([2010, 1, 14, 15, 25, 50, 33]).quarter()// возвращает число 1
year()	Получает или задает год времени объекта FormcyMoment	Если передается параметр **number**, то устанавливается соответствующий год. Принимаются числа от -270000 до 270000. Если параметр не задан, то возвращает год времени объекта FormcyMoment	year(2023) устанавливает в объекте FormcyMoment год = 2023 date([2010, 1, 14, 15, 25, 50, 33]).year() // возвращает число 2010
weeksInYear(): number	Получает количество недель года времени		date([2010, 1, 14, 15, 25, 50, 33]).weeksInYear() //

Функция	Название	Параметры	Пример использования
	объекта FormcyMoment по стандарту ISO		возвращает число 52
weekday()	Получает или задает день недели времени объекта FormcyMoment в соответствии с локалью	Если передается параметр **number**, то устанавливается соответствующий день недели в соответствии с локалью. Если параметр не задан, то возвращает день недели времени объекта FormcyMoment в соответствии с локалью	date([2022, 9, 7]).weekday()// возвращает число 5 date([2022, 9, 7]).weekday(-7) // устанавливает дату на 25 сентября 2022
day()	Получает или задает день недели времени объекта FormcyMoment. Этот метод можно использовать для установки дня недели, где воскресенье – 0, а суббота – 6.	Если передается параметр **number**, то устанавливается соответствующий день недели. Если параметр не задан, то возвращает день недели времени объекта FormcyMoment	date([2022, 9, 7]).day()// возвращает число 5 date([2022, 9, 7]).day(-7) // устанавливает дату на 25 сентября 2022
week()	Получает или задает неделю года времени объекта FormcyMoment в соответствии с локалью	Если передается параметр **number**, то устанавливается соответствующая неделя года в соответствии с локалью. Если параметр не задан, то возвращает неделю года времени объекта FormcyMoment в соответствии с локалью	date([2022, 9, 7]).week()// возвращает число 41 date([2022, 9, 7]).week(1) // устанавливает дату на 31 декабря 2021
toNow(): string	Возвращает время от времени объекта FormcyMoment до текущего момента		date([2007, 0, 29]).toNow() // возвращает строку ‘in 16 years’ date([2007, 0, 29]).toNow(true) // возвращает строку ‘16 years’
to()	Возвращает время от времени объекта FormcyMoment до указанного	Если параметр не задан, то используется текущая дата	date().to([2022, 6, 29]) // возвращает строку ‘2 month ago’ date().to([2022, 11, 29]) // возвращает строку ‘in 3 month’ date([2007, 0, 29]).to()

Функция	Название	Параметры	Пример использования
			// возвращает строку ‘in 16 years’
locale()	Получает или задает локаль (язык, на котором будут выводиться ответы методов)	По умолчанию Moment.js выдает ответы на английском языке. Для смены локали необходимо указать в параметре двухбуквенный код языка	date().locale() //возвращает строку ‘en’ date().locale('ru')// возвращает строку ‘ru’ и меняет локаль date().to([2022, 11, 29]) // возвращает строку ‘через 3 месяца’
isLeapYear()	Возвращает true, если этот год високосный, false – если нет		date([2000]).isLeapYear() // true
localeData()	Возвращает словарь языка (короткие названия месяцев, дней недель и т.п.)		

4.4.4. Проверка работоспособности формул

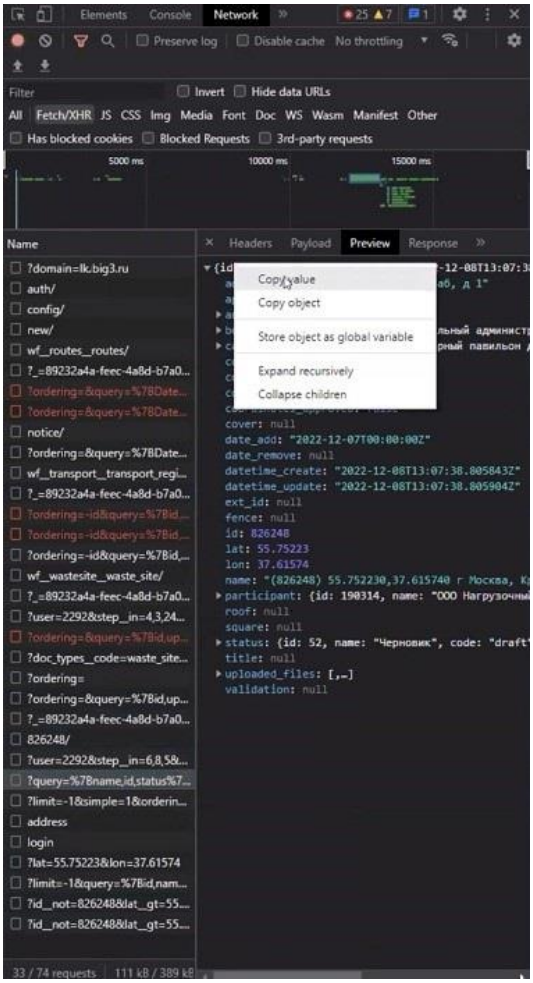
Для удобной проверки работоспособности формул реализована песочница формул. Песочница формул доступна только для пользователя с ролью «Администратор».

Песочница формул («Профиль → Dev: Инструменты») позволяет скопировать готовый результат из формы в контекст data и протестировать работу формул.

Для проверки работоспособности формул необходимо выполнить следующие действия:

1. Откройте любую детальную форму.

2. Откройте инструменты разработчика в браузере, перейдите на вкладку «Network». Найдите запрос и скопируйте значение (рис. 12).



Инструменты разработчика в браузере. Вкладка «Network»

Рис. 12

2. В песочнице формул в поле «Данные» укажите контекст data (рис. 13)

Поле «Данные» песочницы формул

Рис. 13

3. Вставьте скопированное значение из формы.

Теперь можно проверять работу формул (рис. 14).

Пример проверки работы формул

Рис. 14

4.5. Работа с бизнес-процессами со статусами

На Платформе предусмотрена возможность настройки двух типов бизнес-процессов.

Первый тип бизнес-процессов – это «машина состояний» – бизнес-процесс, основанный на смене состояний объектов учета. Для каждой группы сущностей в информационной системе можно настроить перечень доступных состояний (статусов) и граф переходов между ними. После

этого пользователям информационной системы в интерфейсе становятся доступны кнопки переходов между статусами (при наличии соответствующих разрешений, разрешения настраиваются в разделе «Права на переходы»). С помощью этих кнопок пользователи меняют состояние объекта учета.

Администратор Платформы может задавать в административной панели правила, согласно которым страница отображения объекта учета будет меняться в зависимости от его состояния. В зависимости от состояния объекта разные роли пользователей могут иметь разные права на действия с объектом.

Второй тип бизнес-процессов – это система настраиваемых шаблонов событий и реакций на эти события. В админ-панели можно задать условия, при которых в базе данных будет возникать запись о событии (например, событие должно возникать при обновлении записи в реестре МНО с категорией «Контейнерная площадка»). Запись о событии будет содержать также данные объекта, с которым данное событие произошло. Далее в админ-панели можно задать перечень реакций на каждое такое событие, среди реакций может быть отправка письма по e-mail, создание новой записи в каком-либо реестре, операция, переход статуса или http-запрос. Таким образом можно задавать цепочки событий и реакций информационной системы на эти события, создавая дерево автоматизированных процессов.

Объекты в Платформе могут иметь разные статусы. Пользователи могут производить операции над объектами, при которых объект будет переходить из одного статуса в другой. При этом для различных статусов объекта можно настраивать доступность данного объекта для пользователей, перечень действий, которые можно производить с объектом, и т.п. (см. п. 4.3.1 настоящего документа)

Статусы неразрывно связаны с ролевой моделью и правами и разрешениями на Платформе (см. п. 4.3.2 настоящего документа)

4.5.1. Создание бизнес-процесса со статусами

Для создания бизнес-процесса со статусами для объектов типа представления необходимо выполнить следующие действия:

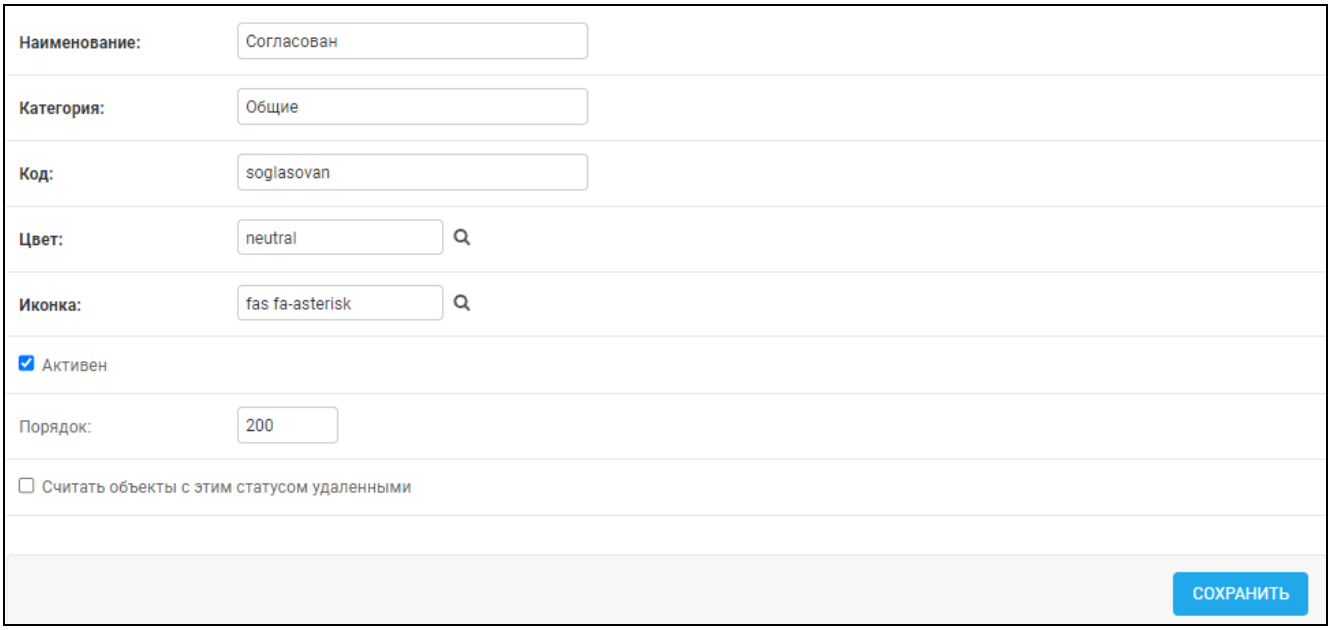
1. Перейдите в раздел конструктора «Документооборот → 1. Типы представлений».

2. Откройте тип представления, для которого необходимо настроить бизнес-процесс со статусами.

3. В подразделе формы «Параметры» в поле «Доступные статусы» добавьте в правый блок все статусы, которые должны быть у объектов данного типа представления. В блоке слева

отображаются все созданные в информационной системе статусы (в том числе для других типов представлений).

Если вы не нашли нужный статус, нажмите кнопку «➕» напротив поля и заполните форму создания нового статуса (рис. 15):



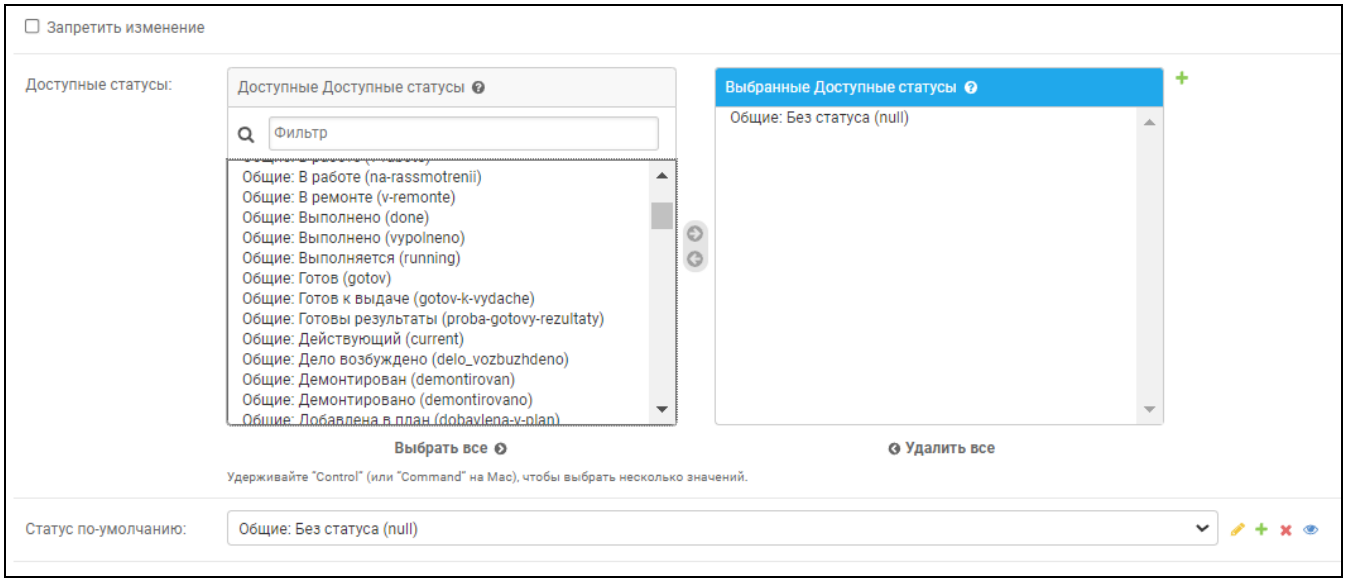
Пример заполнения формы создания нового статуса

Рис. 15

- в поле «Наименование» укажите удобное для пользователя название статуса;
- в поле «Категория» укажите удобное для аналитика название категории. Если такой категории еще нет, то она будет создана;
- в поле «Код» укажите уникальный код статуса или оставьте код, сформированный автоматически;
- в поле «Цвет» выберите цвет из доступных;
- в поле «Иконка» выберите иконку статуса из доступных;
- в поле «Активен» поставьте отметку, если необходимо, чтобы статус отображался в интерфейсе;
- в поле «Порядок» при необходимости укажите числовое значение, которое будет использоваться при сортировке доступных статусов. Но обратите внимание, что это значение не влияет напрямую на сортировку статусов во всех формах, а может использоваться, как параметр при задании условий отображения в конкретной форме;
- в поле «Считать объекты с этим статусом удаленными» можно поставить отметку, если требуется настройка скрытия элементов с таким статусом, не прибегая к коду. Например, статусы «Завершил действие» и «Удален» могут иметь такую отметку, а

аналитик может настроить дашборд таким образом, чтобы не учитывать объекты с такими статусами.

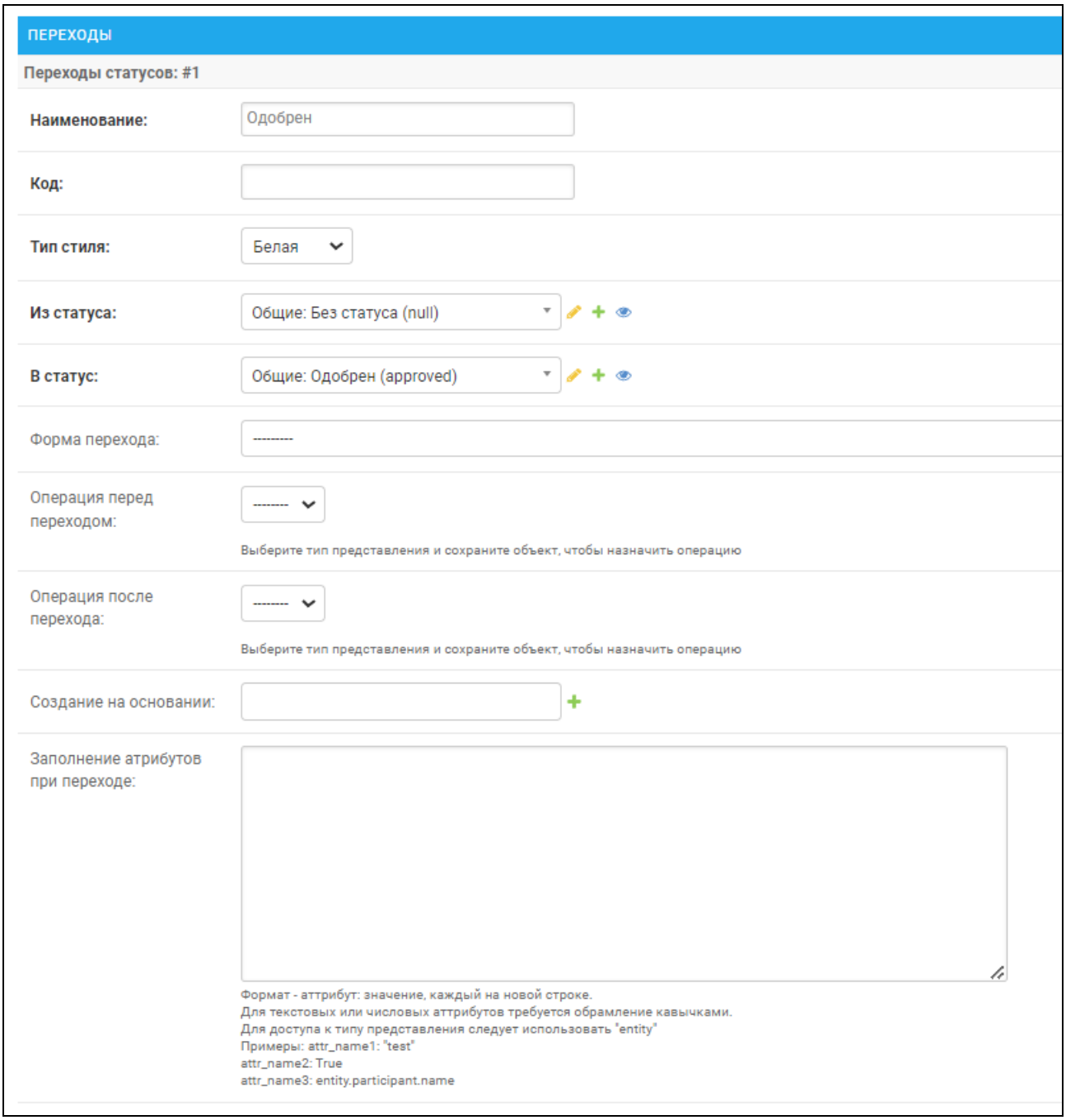
4. В форме типа представления в поле «Статус по умолчанию» можно также выбрать статус, в который будут переводиться все объекты типа представления при создании (рис. 16).



Выбор статуса в форме типа представления

Рис. 16

5. Создайте переходы статусов, т.е. действия, после запуска которых будет осуществляться смена статуса. Для этого в подразделе «Переходы» формы типа представления нажмите «Добавить еще один Переходы статусов» и заполните данные (рис. 17):



Создание переходов статусов

Рис. 17

– в поле «Наименование» укажите наименование кнопки, которая будет отображаться в интерфейсе пользователю, и при нажатии на которую будет происходить смена статуса;
– в поле «Код» укажите уникальный код перехода или оставьте код, сформированный автоматически;
– в поле «Тип стиля» выберите цвет кнопки перехода статуса, которая будет отображаться в интерфейсе пользователю;