Read the Docs Template Documentation

Выпуск 1.0

Read the Docs

нояб. 14, 2022

Оглавление

Глава 1

Введение

Конструктор - подсистема платформы ARTA SYNERGY, предназначенная для создания, настройки и модификации веб-приложений, интегрированных с платформой. Конструктор позволяет создавать приложения без программирования, но с возможностью подключения дополнительных ресурсов (CSS/HTML/JS), обеспечивая тем самым широкий спектр возможностей при проектировании решений.

Конструктор позволяет:

1. Создавать структуру приложения: страницы и переходы между ними, модальные окна и их поведение, обработчики событий и прочее.
2. Оформлять дизайн приложения встроенными инструментами, а также посредством подключения CSS файлов.
3. Бесшовно интегрировать приложение с объектами платформы: формами, реестрами, хранилищем, потоками работ.
4. Подключать внешние (кастомные) источники данных, компоненты, ресурсы и прочее.

Глава 2

Терминология

В настоящем документе используются следующие определения, сокращения и аббревиатуры:

Приложение - интернет программа, созданная для решения определенной бизнес-задачи пользователя и настроенная в соответствии с его требованиями. Приложение состоит из набора страниц.

Страница - документ, представляющий собой набор компонентов различных типов (текст, графические изображения, видео, и т.д.), определенный общим набором свойств: урл, требованием к авторизации и др.

Модальное окно - особый вид страницы, обладающей специфичными свойствами и обязательно являющейся дочерней к одной из основных страниц. Модальное окно открывается «поверх» своей родительской страницы.

Компоненты - элемент, расположенный на странице приложения, характеризующийся своим кодом и типом.

Ресурсы - дополнительные средства, необходимые для разработки приложения (например, таблицы стилей css)

Кейсы - сочетание «Условие» - «Событие» - «Действие», позволяющее реализовать большую часть логики приложения без использования скриптинга. Например, для кнопки «Вход» может быть добавлен кейс - «при клике по кнопку, если пользователь авторизован, перейти на другую страницу».

Условия кейса - предварительные требования, необходимые для выполнения кейса. Например, для кнопки «Вход» из предыдущего примера условием может быть «Авторизация пользователя»

Событие кейса - событие текущего компонента, которое будет обрабатываться в кейсе. Например, клик по кнопке.

Действия кейса - результат, который должен быть получен после наступления события и истинности заданных условий. Например, после клика по кнопке (при условии, что пользователь авторизован) - переход на заданную страницу приложения. Для удобства навигации при создании кейса все действия объединены в **Группы действий**.

Параметры - свойства компонента или страницы, которые могут быть переданы другому компоненту или странице.

Глава 3

Рабочее место пользователя подсистемы Конструктор

3.1 Доступ к подсистеме

Для того, чтобы получить доступ к подсистеме конструктор, необходимо:

1. В подсистеме http://ip:8080/SynergyAdmin (раздел «Управление пользователями» - карточка пользователя) предоставить пользователю доступ **«Конструктор web-клиента»**.
2. Перейти по адресу http://ip:8080/constructor и авторизоваться в подсистеме Конструктор.

Доступ в конструктор может быть также предоставлен к отдельному приложению (одному или нескольким) без общего доступа в конструктор. Для того, чтобы получить доступ к одному или нескольким приложениям, необходимо чтобы пользователь, уже обладающий таким доступом, в свойствах приложения предоставил соответствующее право.

рис. Настройка доступа к редактированию приложения в конструкторе

3.2 Рабочее место пользователя

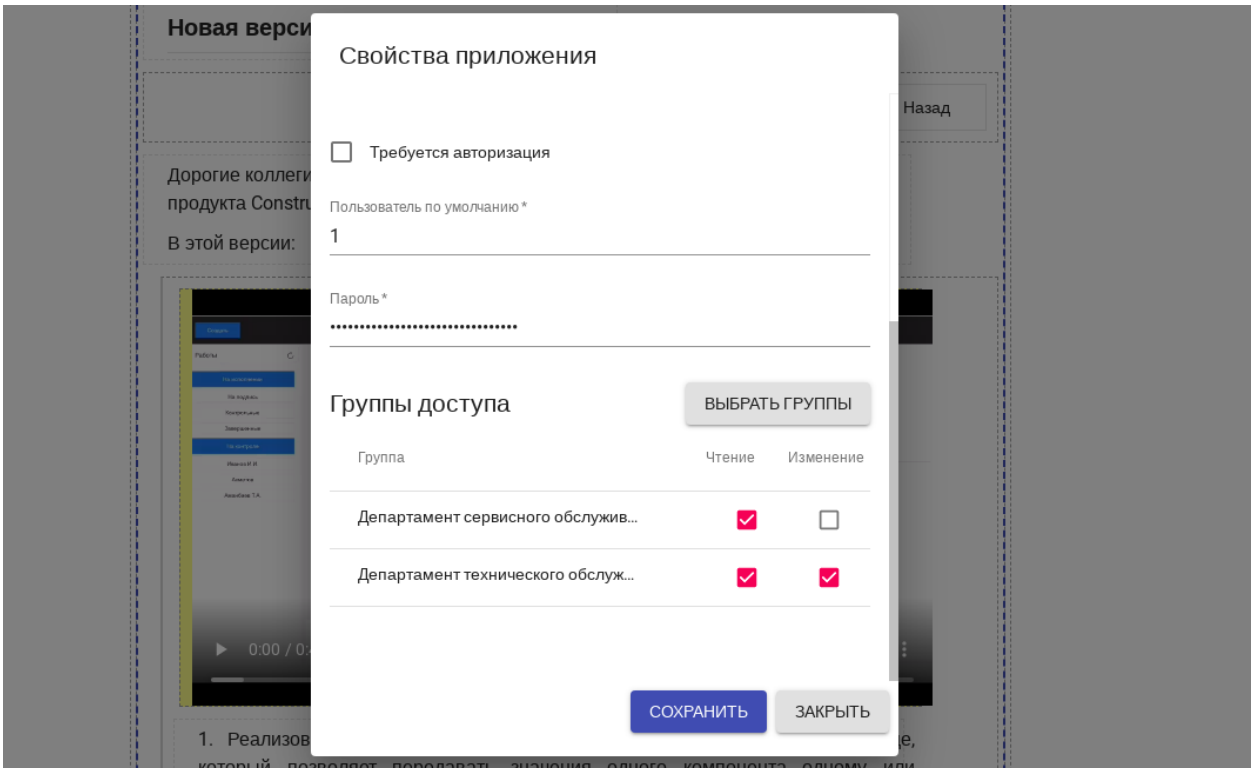
Интерфейс подсистемы Конструктор представлен четырьмя основными панелями:

1. Верхняя панель, состоящая из четырех элементов:

- Кнопка «Меню» - открывает меню для управления приложениями конструктора.
- Название открытого приложения.
- Кнопки отмены и повтора действий.
- Переключатель режима конструктора - при включенной опции отображается режим редактирования, в противном случае - режим просмотра.

2. Левая панель - «Навигатор»:

- Страницы - отображает структуру приложения: дерево папок, страниц и модальных окон. Позволяет создавать, управлять и удалять соответствующие объекты.



- **Компоненты:**
  - Стандартные - позволяет добавлять на страницу стандартные компоненты.
  - Пользовательские - позволяет создавать, управлять и удалять пользовательские компоненты.
- **Ресурсы**
  - Медиафайлы - позволяет добавлять фото: png,jpg(jpeg,jpe),bpm, gif и видео:mp4,avi,mov,mpg - для дальнейшего их использования в соответствующих компонентах.
  - Общий раздел - позволяет добавлять файлы в формате css или js, расширяя возможности конструктора.
- Дерево компонентов - позволяет осуществлять навигацию по всем компонентам открытой страницы.

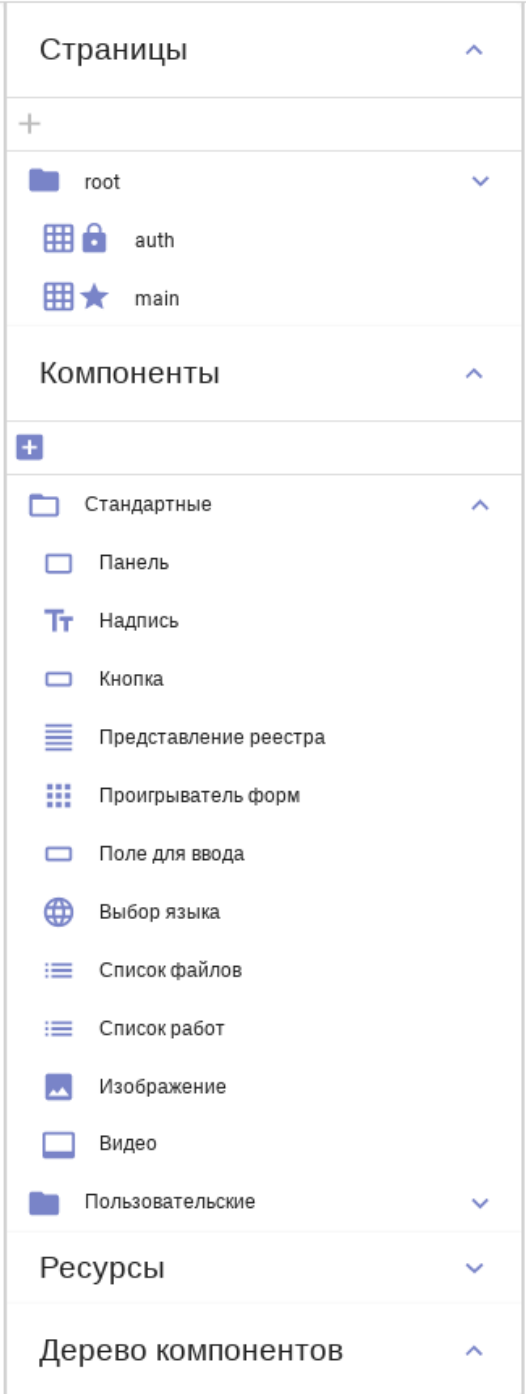
рис. Левая панель конструктора

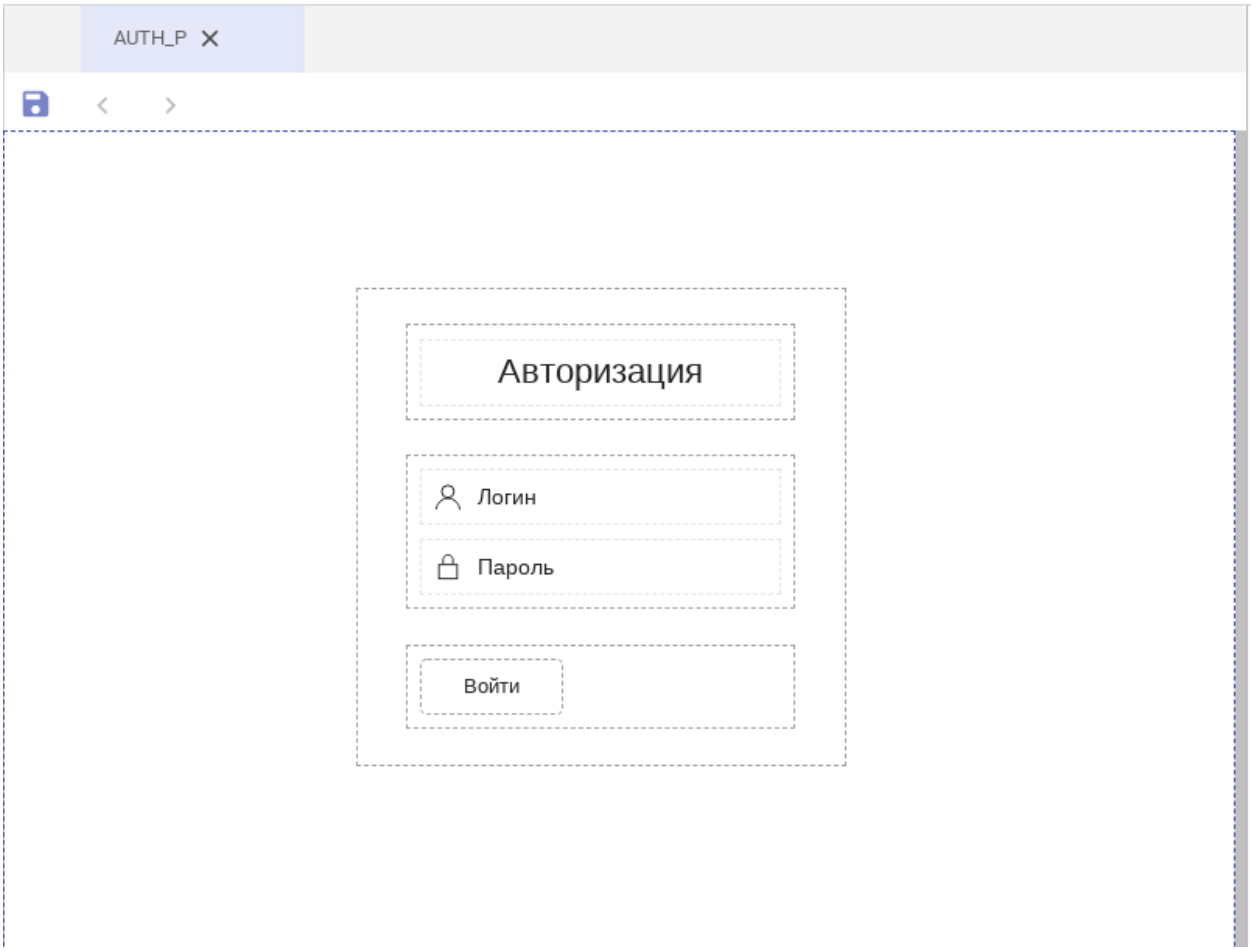
3. Основная рабочая область

- Панель навигации между открытыми объектами: страницами/ресурсами/пользовательскими компонентами.
- Панель действий над страницами: кнопка сохранения, кнопки перемещения компонентов «выше»/ «ниже» относительно других компонентов в рамках той же панели.
- Область редактирования объекта: страницы, ресурса, пользовательского компонента.

рис. Основная рабочая область

4. Правая панель - Панель «Свойств»


Страницы
root
auth
main
Компоненты
Стандартные
Панель
Надпись
Кнопка
Представление реестра
Проигрыватель форм
Поле для ввода
Выбор языка
Список файлов
Список работ
Изображение
Видео
Пользовательские
Ресурсы
Дерево компонентов


AUTH_P
Авторизация
Логин
Пароль
Войти

- Свойства - отображает перечень стандартных и специфичных свойств выделенного в основной рабочей области компонента.
- Кейсы - позволяет управлять кейсами (настраивать сочетания «Условие»-«Событие»-«Действие»), для реализации логики поведения выделенного в основной рабочей области компонента.
- Параметры - позволяет настроить получение текущим компонентом одного или нескольких параметров страницы (для этого параметры должны быть предварительно определены в свойствах страниц).

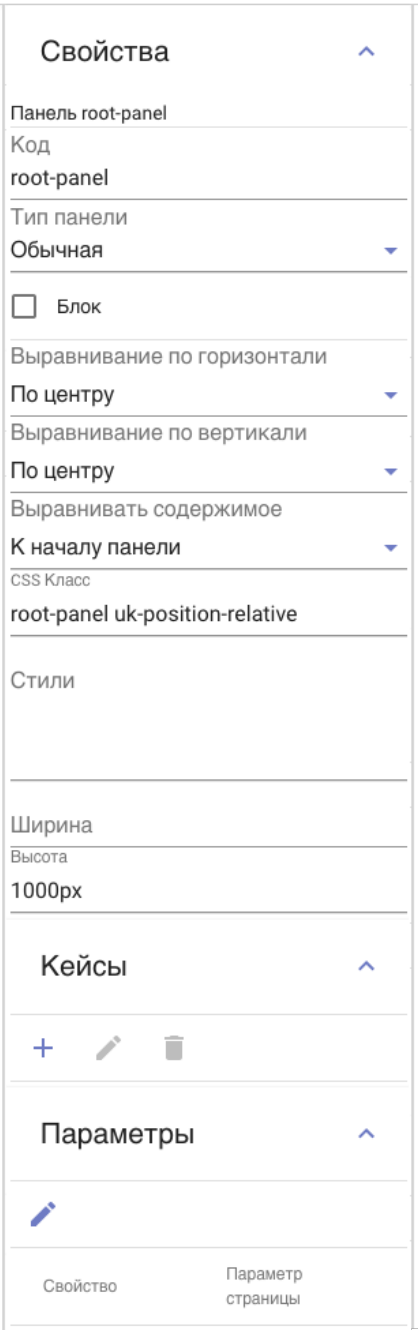


рис. Панель свойств

3.3 Основные подразделы интерфейса

3.3.1 Меню

Подраздел, позволяющий управлять приложениями, созданными на конструкторе. Подраздел позволяет осуществлять следующие действия:

Создать - открывает модальное окно для создания нового приложения в конструкторе.

Открыть приложение - открывает модальное окно со списком всех существующих приложений. В этом окне представленное в списке приложение можно или открыть на редактирование или удалить, при необходимости.

Свойства - открывает окно редактирования свойств текущего открытого приложения.

Раздел «Свойства» отображает следующие свойства приложения:

- **Код** - уникальный код приложения, используется в том числе при экспорте/импорте приложения (если код совпадает, то не создается новое приложение, а обновляется версия существующего с таким же кодом).
- **URL** - постфикс в адресе приложения, по которому оно будет доступно. В общем случае, ссылка формируется по принципу: http://ip:8080/*значение_указанное_в_этом_поле*
- **Номер ревизии** - автоматически заполняется (увеличивается +1) при сохранении новой версии приложении
- **Требуется авторизация** - если опция включена, в приложении обязательно должна присутствовать страница авторизации, если опция отключена - ниже обязательно должен быть указан пользователь по умолчанию (пользователь, от имени которого будет проверяться доступ к объектам SYNERGY, если они используются в приложении).
- **Пользователь по умолчанию (логин/пароль)** - поля доступны только если отключена опция «Требуется авторизация»
- **Группы доступа** - даёт доступ к редактированию отдельных приложений.

Помощь - открывает инструкцию по созданию приложения на конструктор.

Импорт - позволяет импортировать json - файл приложения, созданного на конструкторе.

Деплой - создает и деплоит war-файл приложения в папке /opt/synergy/jboss/standalone/deployments. После деплоя приложение становится доступным для пользователя по адресу http://ip:8080/*значение_указанное_в_поле_URL_приложения*

Экспорт - позволяет экспортировать и скачать json-файл приложения.

WAR - позволяет скачать war-файл приложения.

Выход - осуществляет выход авторизованного пользователя в конструкторе.

3.3.2 Страницы

Подраздел, позволяющий управлять страницами текущего открытого приложения. Подраздел позволяет осуществлять следующие действия:

1. Создавать/редактировать/удалять папки.
2. Создавать/редактировать/удалять страницы.
3. Импортировать и экспортировать страницы.

4. Создавать/редактировать/удалять модальные окна для страниц.
5. Просматривать дерево страниц.
6. Просматривать версии страниц и модальных окон.

Свойства папки

Новая папка может быть добавлена через кнопку «+» и выбор пункта контекстного меню «Добавить папку» (активно только при выделенной ноде папки в навигаторе). В качестве свойств папки необходимо указать название.

Свойства страницы

Новая страница в приложении может быть создана через кнопку «+» и выбор пункта контекстного меню «Добавить страницу» (активно только при выделенной ноде папки в навигаторе). Также страница может быть импортирована при выборе пункта «Импортировать страницу».

При создании или при выборе пункта «Изменить» контекстного меню страницы отображаются следующие свойства страницы:

- **Название** - отображается в навигаторе конструктора
- **Код** - уникальный код страницы, используется в том числе при экспорте/импорте страницы (если код совпадает, то не создается новая страница, а обновляется версия существующей с таким же кодом).
- **Требуется авторизация** - позволяет настраивать авторизацию на каждую отдельно страницу (данная настройка над страницей приоритетнее аналогичной настройки над всем приложением).
- **URL** - *на текущий момент настройка не используется*
- **Путь** - *на текущий момент настройка не используется*
- **Принимаемые параметры** - Механизм позволяющий передавать значения между компонентами на разных страницах.

Свойства модального окна

Модальное окно является особым видом страницы, со своими специфичными свойствами и обязательно являющейся дочерней к одной из основных страниц. Модальное окно может быть создано через кнопку «+» и выбор пункта контекстного меню «Модальное окно» (активно только при выдленное ноде **страницы** в навигаторе).

Свойства модального окна:

Основные (редактируютя через «Изменить» в контекстном меню модального окна):

- **Название**
- **Код**

Специфичные (редактируются в разделе «Свойства» корневой панели):

- **Заголовок модального окна** - текстовое поле с локализацией.
- **Вертикальный скролл** - да/нет
- **Горизонтальный скролл** - да/нет
- **Отображать кнопку закрыть** - да/нет

Прим. Стандартные поля ширины и высоты влияют на размеры модального окна, открываемого «поверх» ее родительской страницы. Для открытия модального окна должен быть настроен соответствующий кейс, например, по клику на кнопку.

3.3.3 Компоненты

Подраздел позволяющий добавлять на страницы компоненты: стандартные и пользовательские, а также создавать/редактировать/удалять пользовательские компоненты.

Список стандартных компонентов и их свойств определен в разделе:

На текущий момент перечень пользовательских компонентов является сквозным для всех приложений данного конструктора (т.е. при добавлении ПК в одном приложении, он автоматически становится доступен во всех других).

3.3.4 Кейсы

Подраздел, позволяющий управлять кейсами текущего выделенного компонента. Подраздел позволяет осуществлять следующие действия:

1. Создавать кейс
2. Удалять кейс
3. Редактировать кейс

Процесс создания кейса

1. Для создании кейса необходимо определить сочетание: Условие - Событие - Действие
2. Список возможных условий стандартный для каждого компонента и состоит из:

- Авторизация пользователя (два значения: авторизован, не авторизован)
- Локаль (русский, казахский, английский)
- Сравнить
- Свойства компонента
- Свойства компонента из итератора

3. Список событий для каждого компонента свой и определен в разделе «Перечень компонентов и их атрибутов»
4. Список возможных действий представлен в таблице ниже.

Список **специальных** действий для каждого компонента свой и определен в разделе «Перечень компонентов и их атрибутов».

Действие	Параметры действия
Перейти на страницу	Код страницы Принимаемы параметры
Выполнить на странице	Код компонента (над которым будет осуществляться действие) Действие: • Скрыть/показать • Одно из **Специальных действий над компонентом** Параметры выбранного действия
Выполнить на итераторе Выполнить на странице	Код итератора Код компонента (над которым будет осуществляться действие) **Специальные действия над компонентом** Параметры этого действия
Открыть URL	Источник URL: • Произвольный текст (поле для ввода URL) • Параметры страницы (поле для выбора параметра) • Значения компонента (поле для выбора компонента) • Локализованный текст (поле для ввода URL) • Параметр итератора (поле для ввода параметра) • Кастомный источник
Показать сообщение	Источник сообщения: • Произвольный текст (поле для ввода URL) • Параметры страницы (поле для выбора параметра) • Значения компонента (поле для выбора компонента) • Локализованный текст (поле для ввода URL) • Параметр итератора (поле для ввода параметра) • Кастомный источник Тип сообщения • Успешное сообщение (отобразиться зеленым шрифтом) • Сообщение (отобразиться синим шрифтом) • Ошибка (отобразиться красным шрифтом) Время отображения (в секундах)
Логин	• Логин (выбор одного из текстовых полей с отключенной опцией «Не отображать вводимые символы») • Пароль (выбор одного из текстовых полей со включенной опцией «Не отображать вводимые символы»)
Выход из системы	
Вернуться со страницы логина	
Выполнить API	Входной параметр (произвольный или параметр страницы) Метод (Данные по форме, Данные документ) Выходной параметр
Модальное окно	Код модального окна Скрыть/Показать

Глава 4

Алгоритм создания приложения

4.1 Приложение

Алгоритм работы в конструкторе рассмотрим на примере создания простейшего приложения с двумя страницами (страницей авторизации и «Главной»).

1. Для создания нового приложения необходимо выбрать пункт меню «Создать»
2. В появившемся окне необходимо указать:

- Название - будет отображаться на верхней панели, при работе с приложением и в списке при выборе пункта меню «Открыть приложение»
- Код - уникальный код приложения. Может содержать латинские буквы, цифры и символ «_» (нижнее подчеркивание)
- URL - уникальный адрес, по которому можно будет открыть приложение.
- Поставить галочку «Требуется авторизация» (или указать логин и пароль для пользователя по умолчанию).

В нашем примере мы будем создавать приложение со страницей авторизацией, соответственно данная опция включена.

рис. Окно свойств приложения

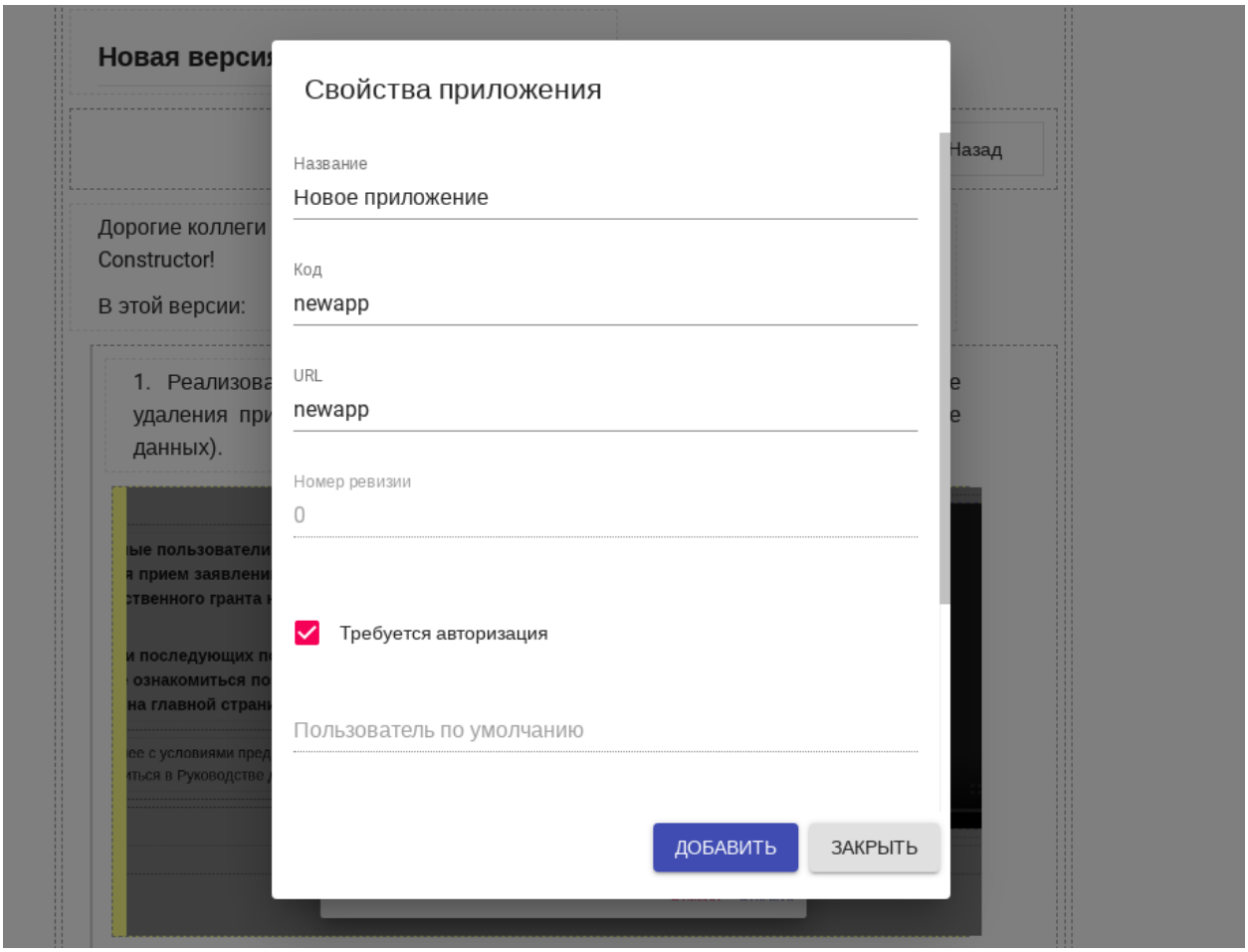
3. Выделить корневую папку root, нажать на «+» и выбрать пункт «Добавить страницу»:

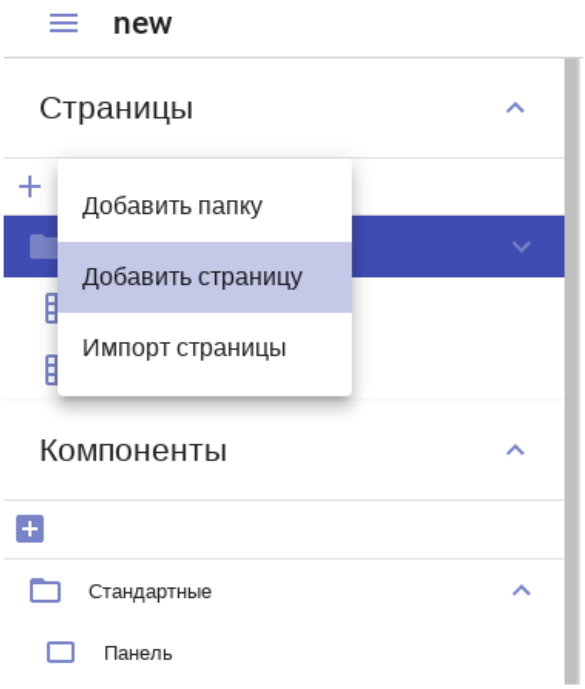
рис. Добавление новой страницы приложения

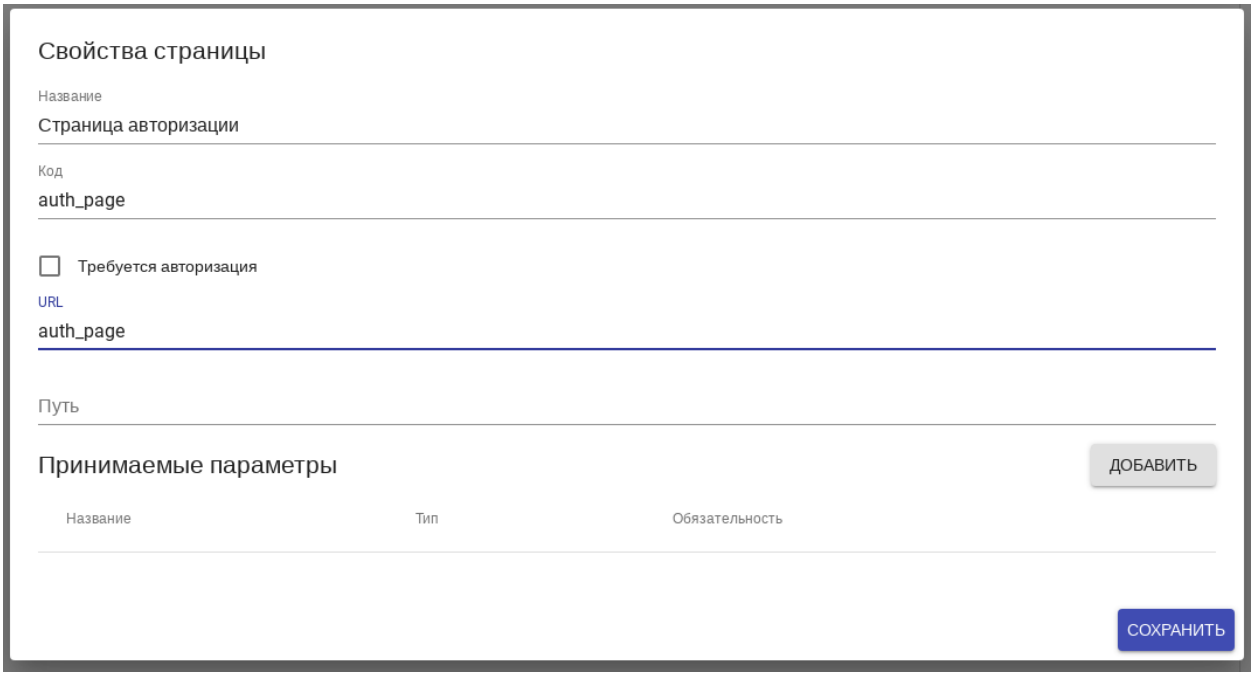
4. При создании нужно указать свойства для каждой страницы, в нашем случае страницы авторизации (заполняем поля согласно рис. Окно создания страницы «Авторизация»)

рис. Окно создания страницы «Авторизация»

5. В нашем примере мы будем создавать приложение соостоящее из двух страниц: страницы авторизации и «Главной». Таким образом, необходимо повторить процесс добавления страницы - для Главной.


Свойства приложения
Название
Новое приложение
Код
newapp
URL
newapp
Номер ревизии
0
Требуется авторизация
Пользователь по умолчанию
ДОБАВИТЬ
ЗАКРЫТЬ


new
Страницы
Добавить папку
Добавить страницу
Импорт страницы
Компоненты
Стандартные
Панель


Свойства страницы
Название
Страница авторизации
Код
auth_page
Требуется авторизация
URL
auth_page
Путь
Принимаемые параметры
ДОБАВИТЬ
Название
Тип
Обязательность
СОХРАНИТЬ

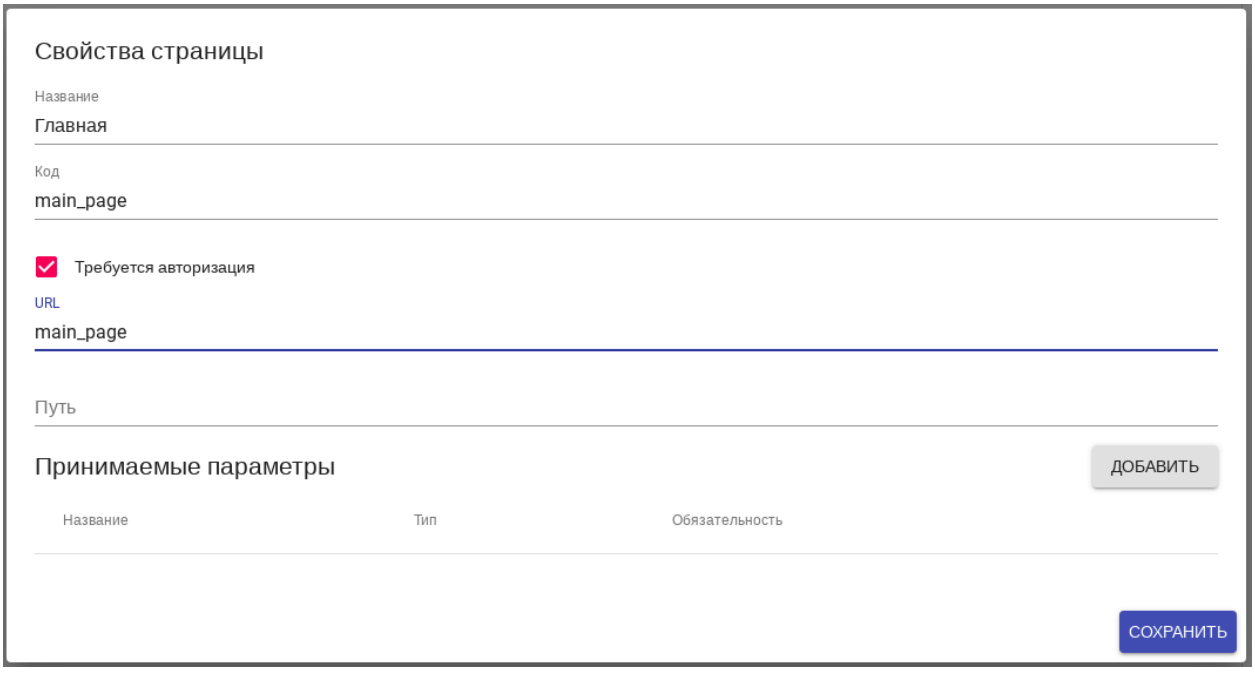

Свойства страницы
Название
Главная
Код
main_page
Требуется авторизация
URL
main_page
Путь
Принимаемые параметры
ДОБАВИТЬ
Название
Тип
Обязательность
СОХРАНИТЬ

рис. Окно создания страницы «Главная»

6. Затем можно приступать к оформлению каждой страницы.

4.2 Страница авторизации

1. Для начала нужно указать, что эта страница является стартовой для приложения и является страницей авторизации. Для этого нужно кликнуть правой кнопкой по названию в блоке «Страницы»:

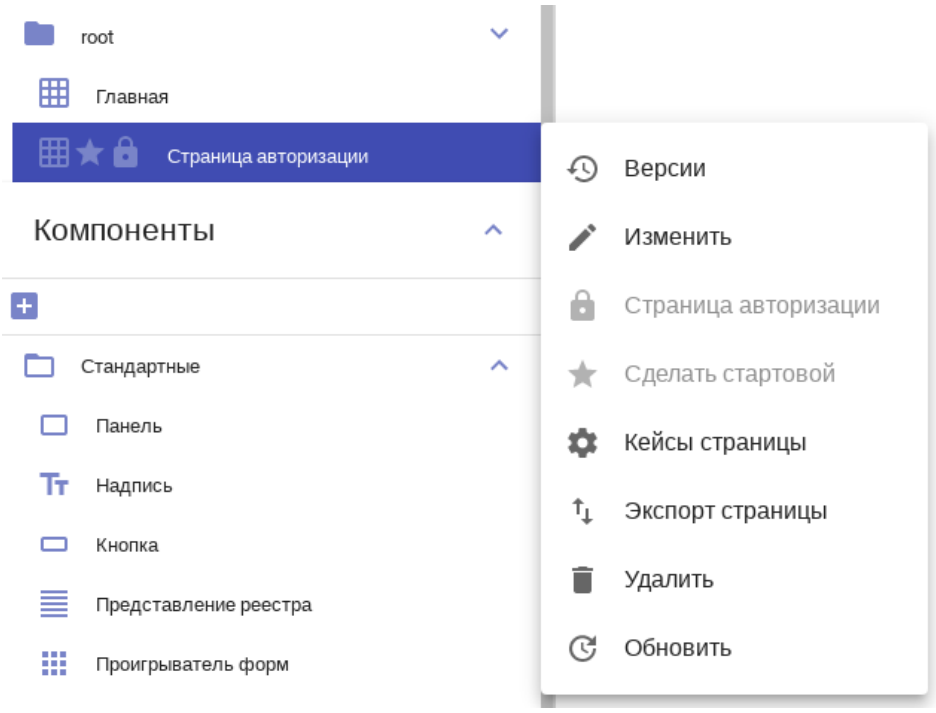


рис. Окно свойств страницы

2. Далее открываем страницу на редактирование (двойным кликом по названию или иконке в блоке «Страницы»).

3. По умолчанию каждая страница содержит компонент `root-panel` - панель, в которой будут размещены все остальные компоненты страницы. Перед тем, как добавлять на страницу новый компонент, необходимо выделить панель, на которой он будет расположен, кликом левой кнопки мыши. После этого справа отобразится блок свойств выделенной панели:

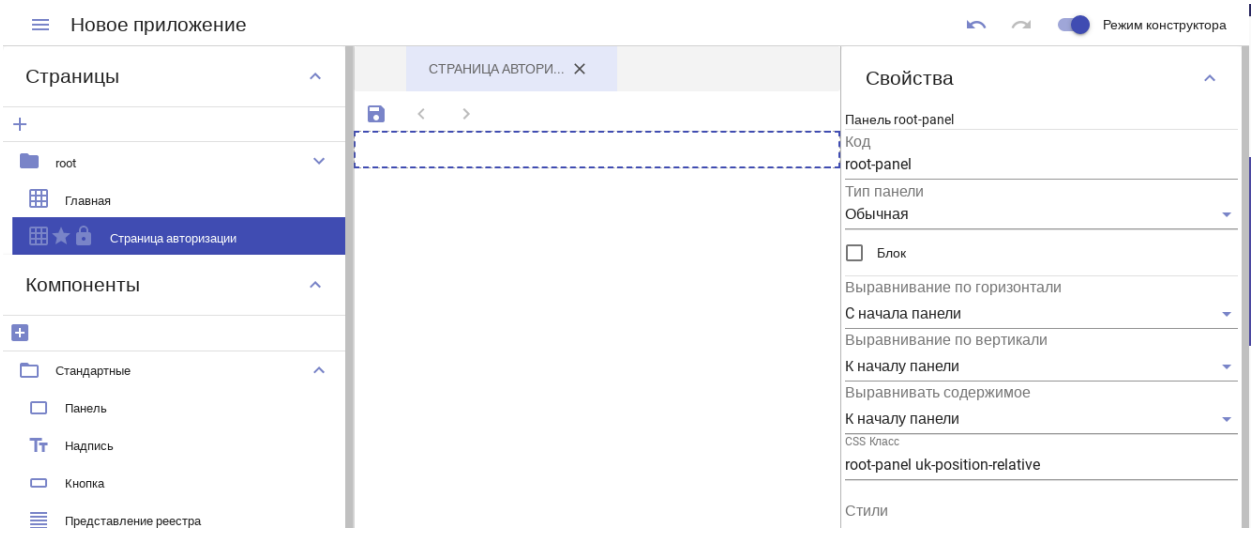
рис. Редактирование корневой панели страницы

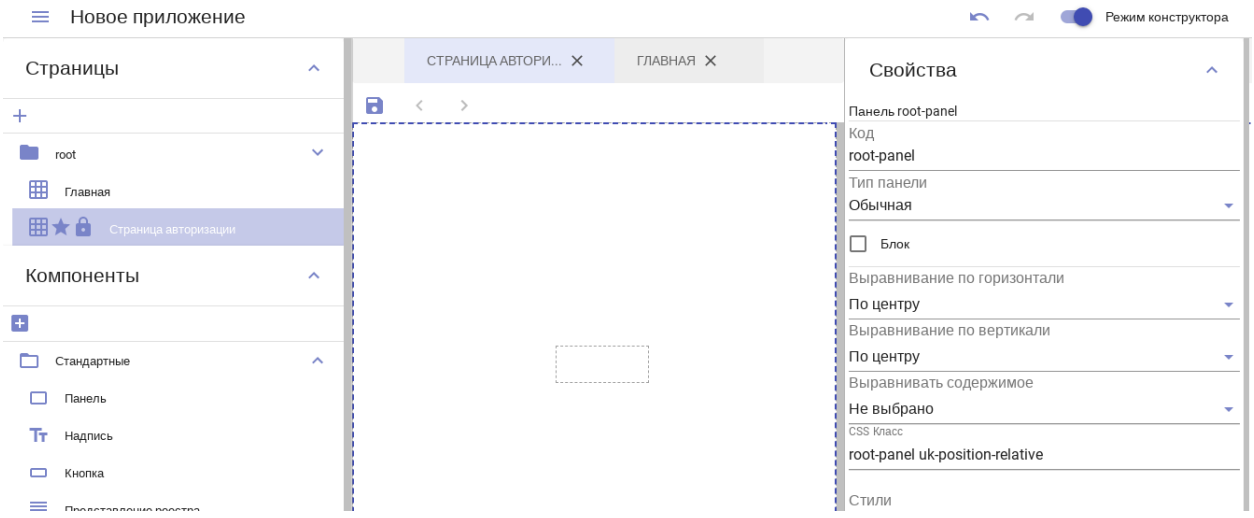
4. Настроим свойства корневой панели:

- Выравнивание по горизонтале - **по центру**
- Выравнивание по вертикали - **по центру**
- Стили - **height: 100vh;** (Растягивает панель на всю высоту экрана. Возможно, нужно сохранить и переоткрыть страницу, чтобы изменения применились).

5. Добавим на `root-panel` компонент «Панель» - это будет наше окно авторизации.

рис. Создание панели для окна авторизации


Новое приложение
Режим конструктора
Страницы
root
Главная
Страница авторизации
Компоненты
Стандартные
Панель
Надпись
Кнопка
Представление реестра
СТРАНИЦА АВТОРИ...
Свойства
Панель root-panel
Код
root-panel
Тип панели
Обычная
Блок
Выравнивание по горизонтали
С начала панели
Выравнивание по вертикали
К началу панели
Выравнивать содержимое
К началу панели
CSS Класс
root-panel uk-position-relative
Стили


Новое приложение
Режим конструктора
Страницы
root
Главная
Страница авторизации
Компоненты
Стандартные
Панель
Надпись
Кнопка
СТРАНИЦА АВТОРИ...
ГЛАВНАЯ
Свойства
Панель root-panel
Код
root-panel
Тип панели
Обычная
Блок
Выравнивание по горизонтали
По центру
Выравнивание по вертикали
По центру
Выравнивать содержимое
Не выбрано
CSS Класс
root-panel uk-position-relative
Стили

6. Выделив только что созданную панель, настроим ее свойства:

- Тип - **Вертикальная**
- Выравнивание по горизонтале - **по центру**
- Выравнивание по вертикали - **по центру**

7. добавим в нее еще три панели:

- В верхнюю из добавленных трёх панелей добавляем компонент - Надпись, в Свойствах указываем текст надписи - «Авторизация»
- В среднюю добавим два компонента «Поле для ввода» - для логина и пароля. Выбираем иконки и прописываем текст для плейсхолдера
- В нижнюю добавим компонент «Кнопка» с текстом «Войти»

Страница авторизации почти готова:

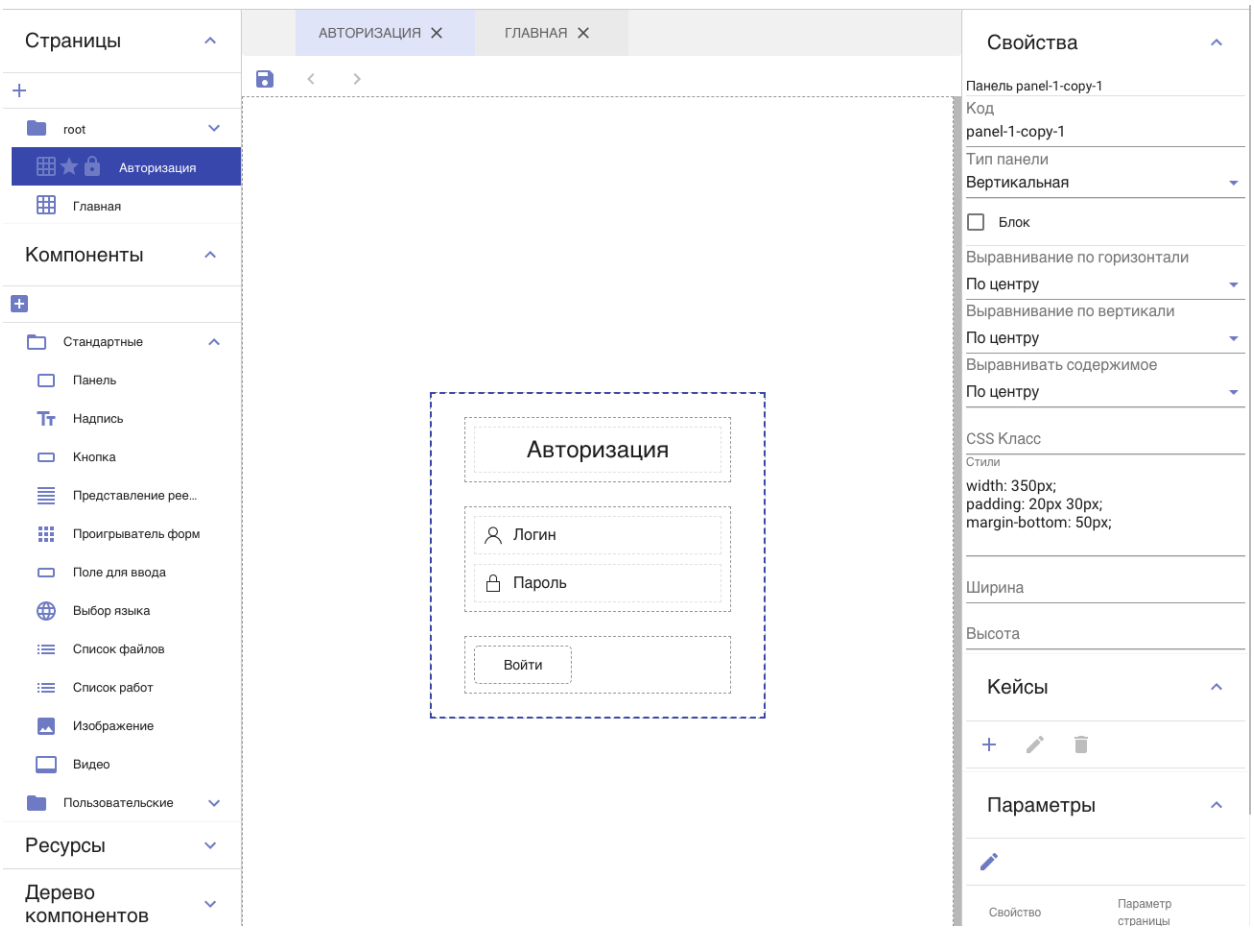
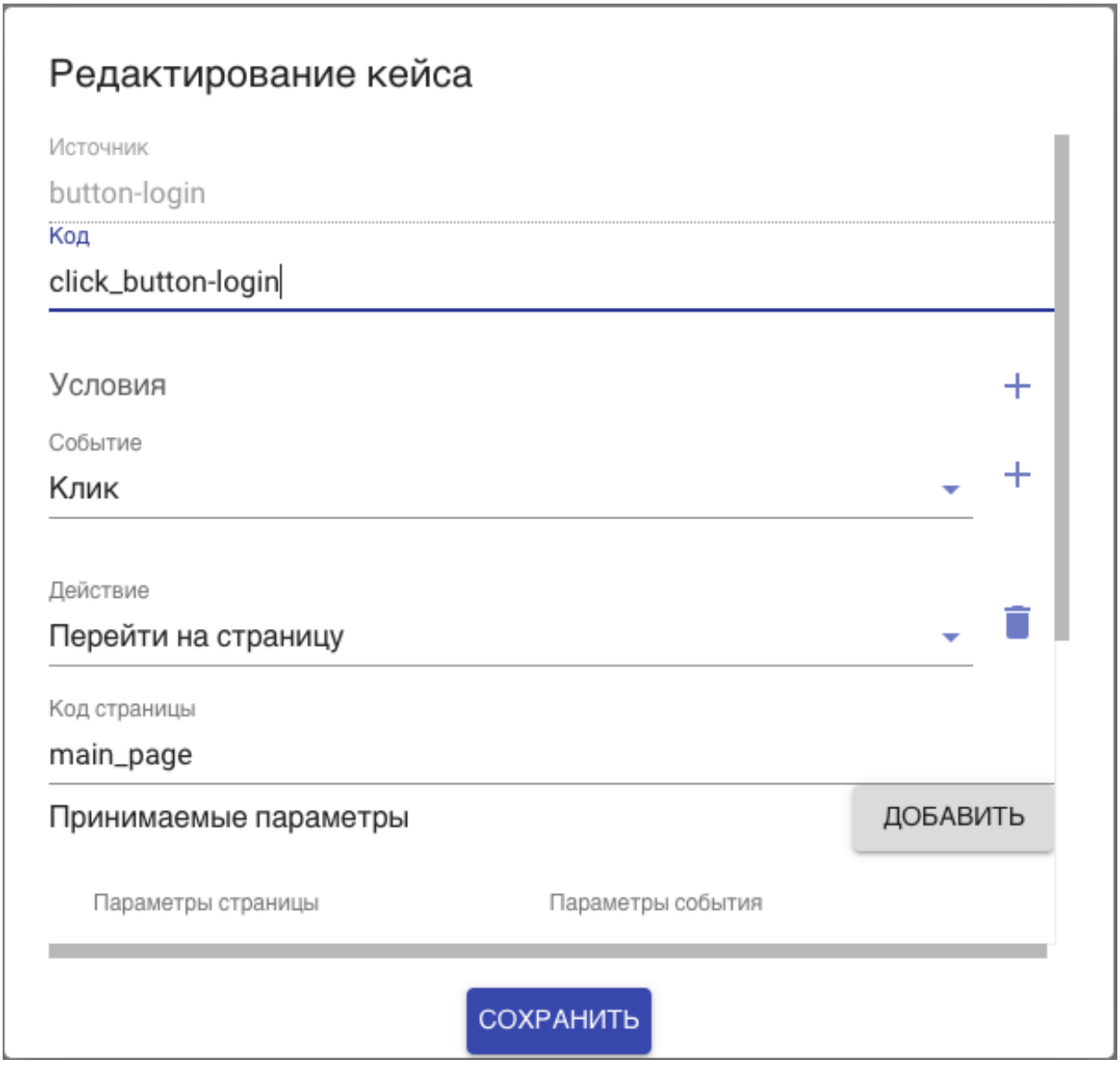


рис. Окно авторизации

Добавим кейс для кнопки «Войти»:

рис. Кейс перехода на главную страницу

5. Перейдём к Главной странице. Для начала добавим на нее логотип: компонент Панель, внутрь вложим компонент Изображение:


Редактирование кейса
Источник
button-login
Код
click_button-login
Условия
Событие
Клик
Действие
Перейти на страницу
Код страницы
main_page
Принимаемые параметры
ДОБАВИТЬ
Параметры страницы
Параметры события
СОХРАНИТЬ

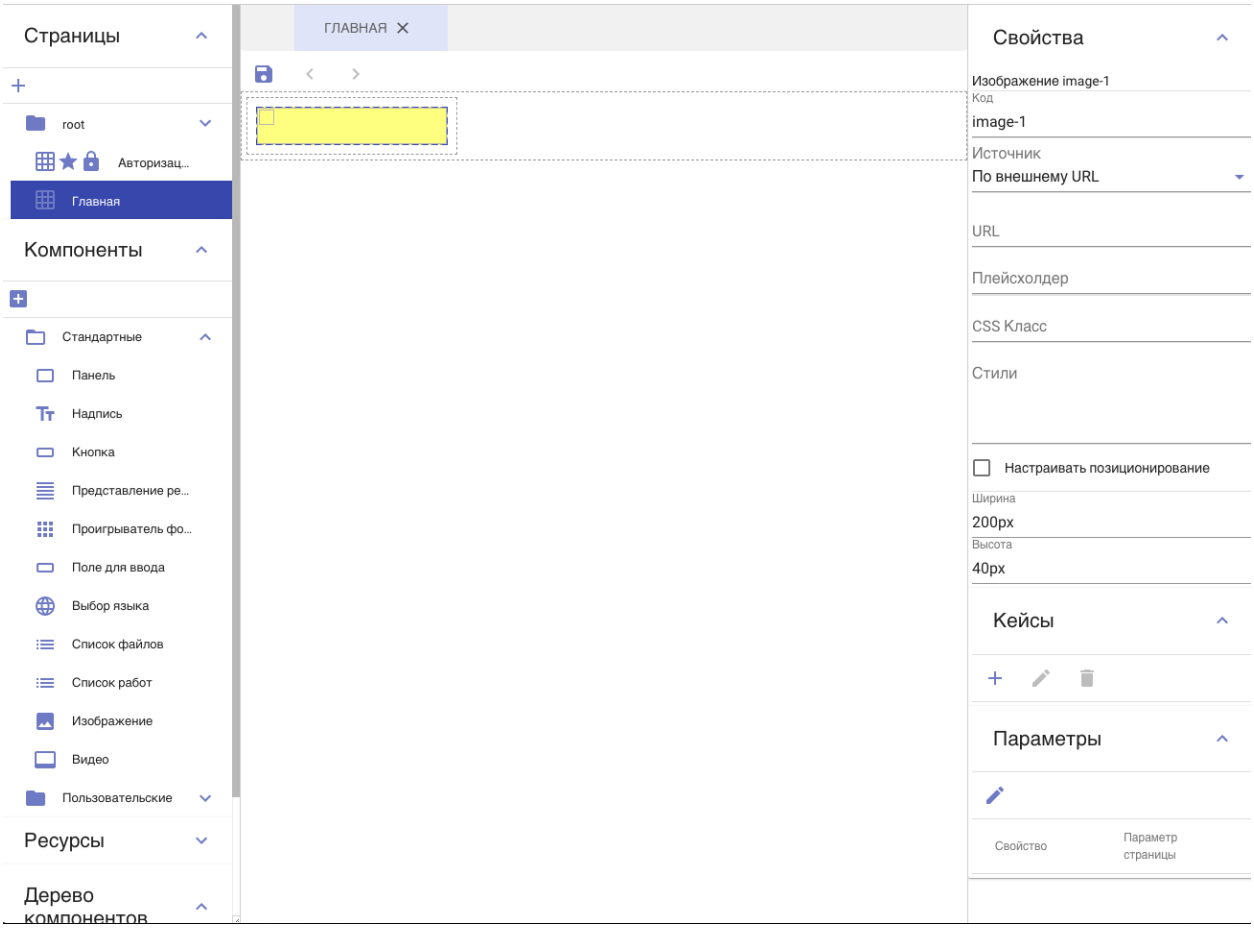

Страницы
root
Авторизац...
Главная
Компоненты
Стандартные
Панель
Надпись
Кнопка
Представление ре...
Проигрыватель фо...
Поле для ввода
Выбор языка
Список файлов
Список работ
Изображение
Видео
Пользовательские
Ресурсы
Дерево компонентов
ГЛАВНАЯ
Свойства
Изображение image-1
Код
image-1
Источник
По внешнему URL
URL
Плейсхолдер
CSS Класс
Стили
Настраивать позиционирование
Ширина
200px
Высота
40px
Кейсы
Параметры
Свойство
Параметр страницы

рис. Создание панели с логотипом

Выделяем компонент Изображение, указываем Источник - Из хранилища. Затем переходим в Хранилище Synergy, открываем нужный файл и его свойства:

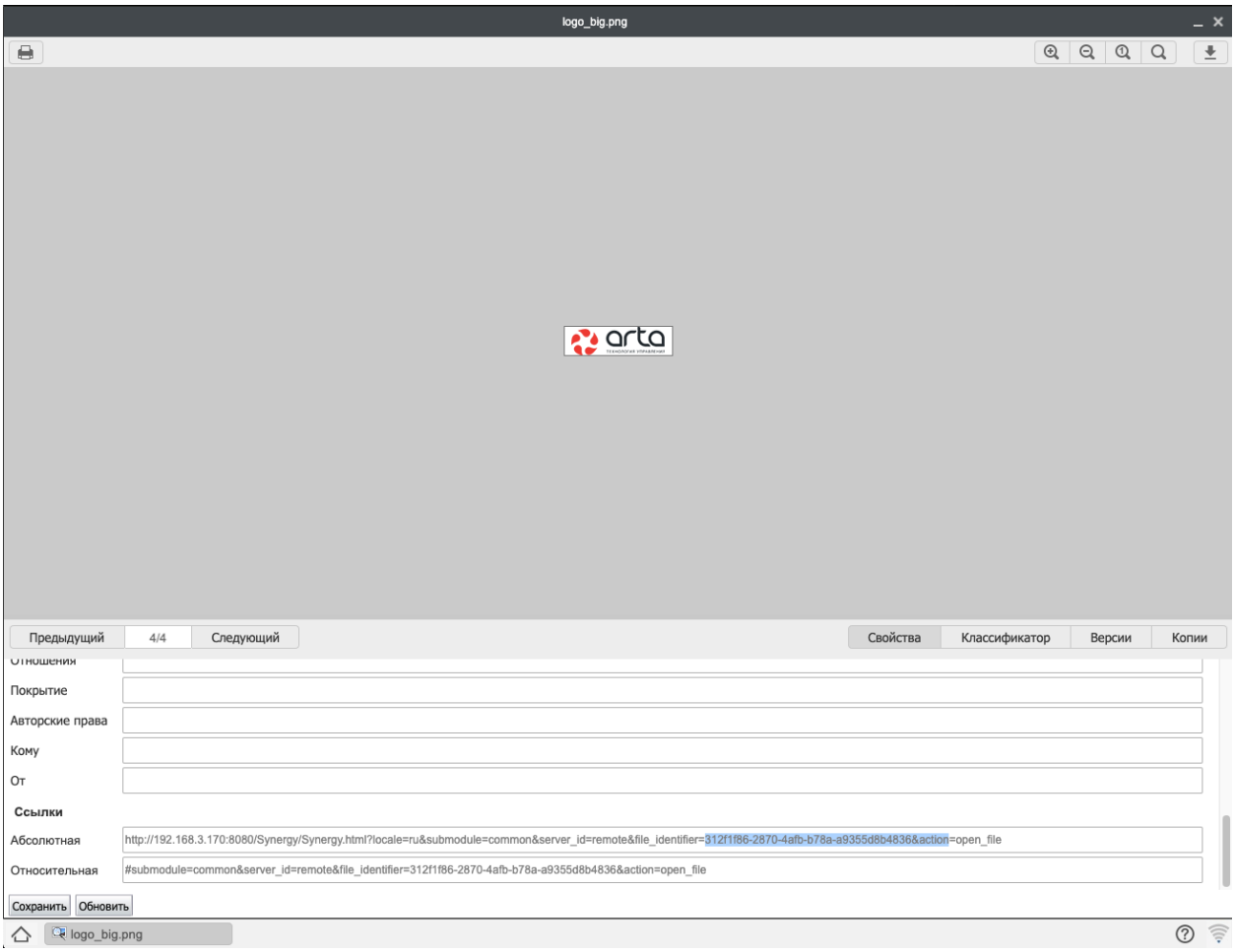


рис. Копируем идентификатор изображения

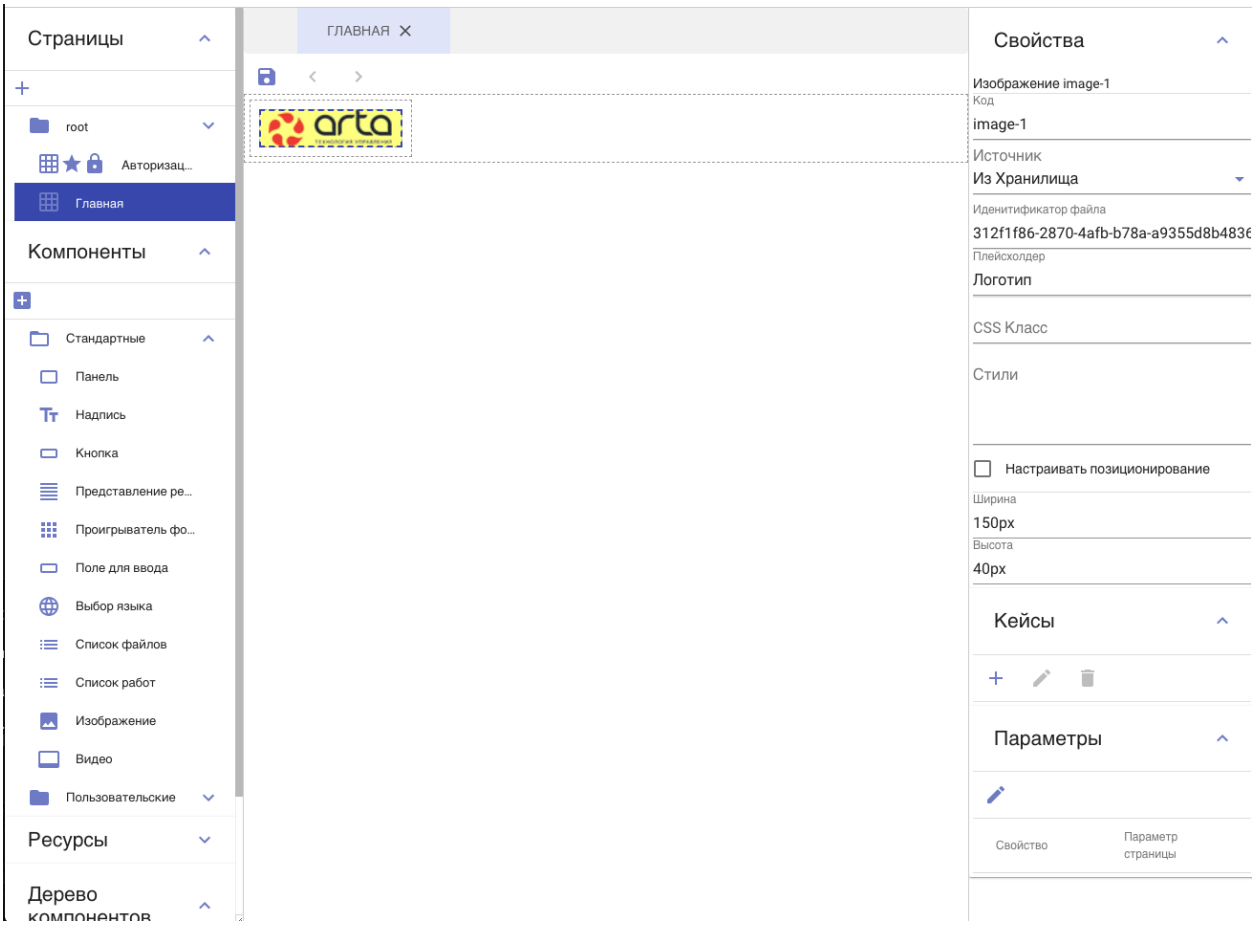
Вставляем скопированный идентификатор изображения в Свойства компонента:

рис. Предпросмотр добавленного изображения

На панель с логотипом добавим компонент «Выбор языка». Чтобы сместить его в правый угол, сначала добавим еще одну панель и перенесем компонент при помощи ctrl+x - ctrl+v:

рис. Добавленная панель с выбором языка

Затем укажем ширину панелей, на которых располагаются компоненты: 20% для логотипа и 80% для локали. Также выбираем выравнивание для этой панели:


Страницы
root
Авторизац...
Главная
Компоненты
Стандартные
Панель
Надпись
Кнопка
Представление ре...
Проигрыватель фо...
Поле для ввода
Выбор языка
Список файлов
Список работ
Изображение
Видео
Пользовательские
Ресурсы
Дерево компонентов
ГЛАВНАЯ
Свойства
Изображение image-1
Код
image-1
Источник
Из Хранилища
Идентификатор файла
312f1f86-2870-4afb-b78a-a9355d8b4836
Плейсхолдер
Логотип
CSS Класс
Стили
Настраивать позиционирование
Ширина
150px
Высота
40px
Кейсы
Параметры
Свойство
Параметр страницы


ГЛАВНАЯ
ru

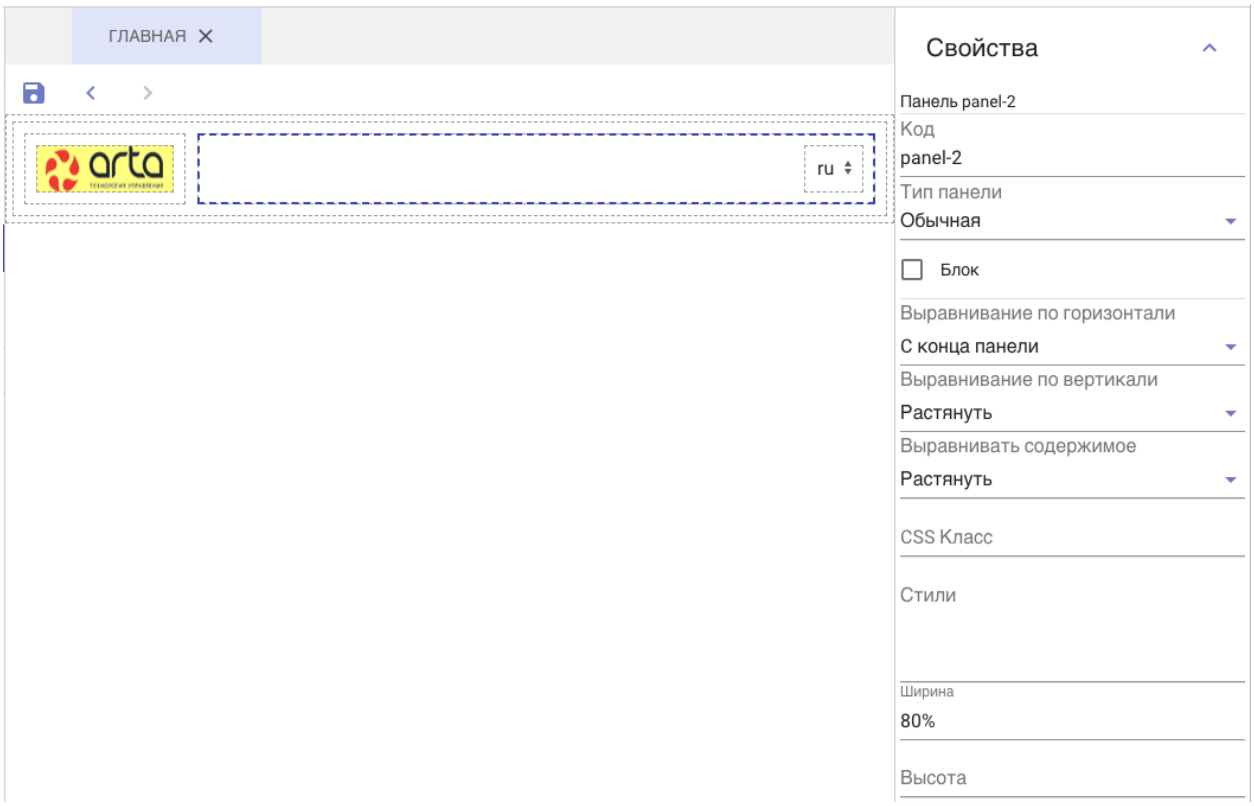


рис. Выравнивание для панели с выбором языка

Приветствие: добавляем панель и в нее компонент «Надпись», указываем текст надписи и ее свойства:

рис. Панель приветствия

Выделив корневую панель, добавим компонент «Видео», указав его источник:

рис. Добавление видео

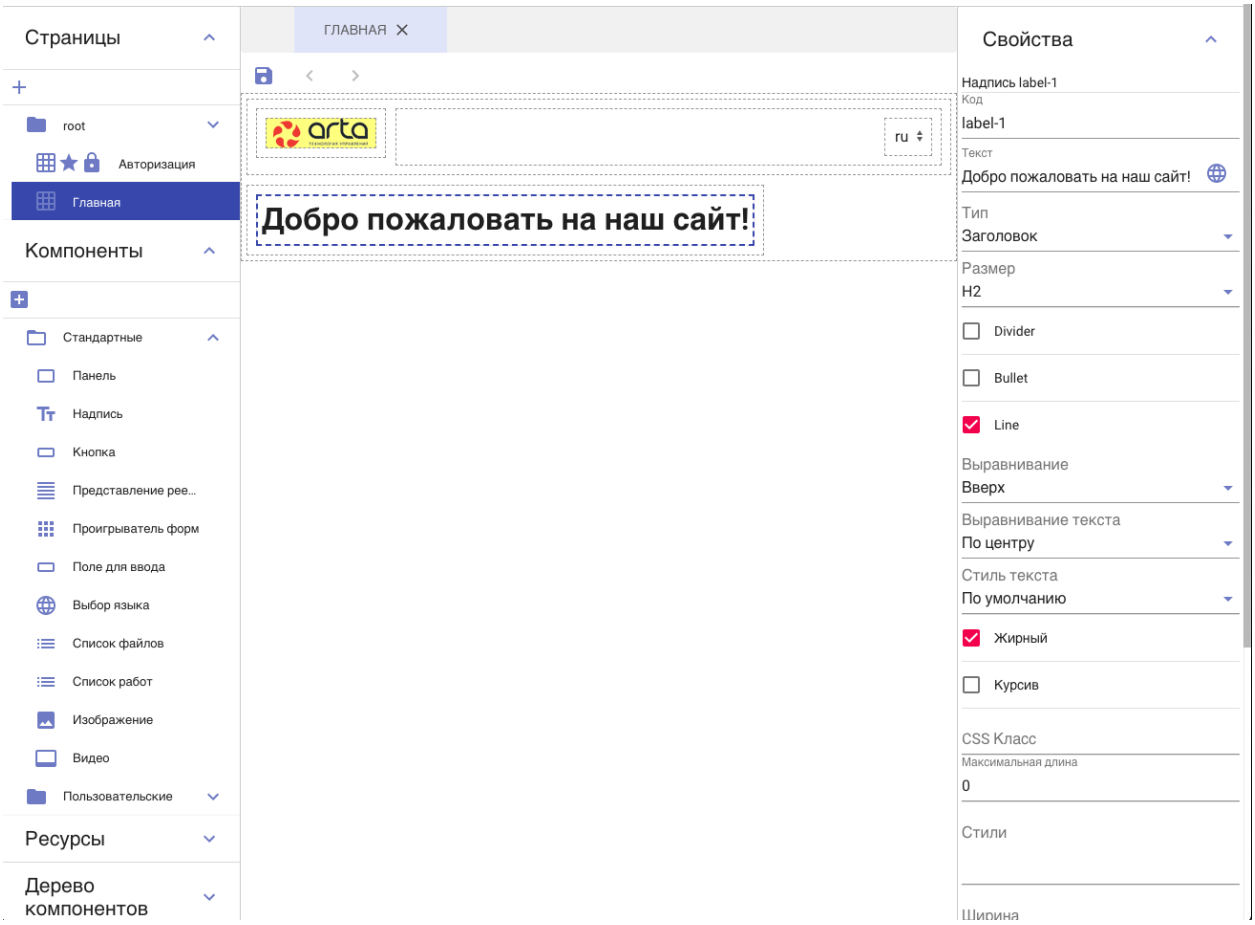
Сохраняем страницу, переходим в меню приложения и нажимаем «Деплой». По адресу, указанному в свойствах приложения, можно открыть наше приложение (http://ip:8080/url):

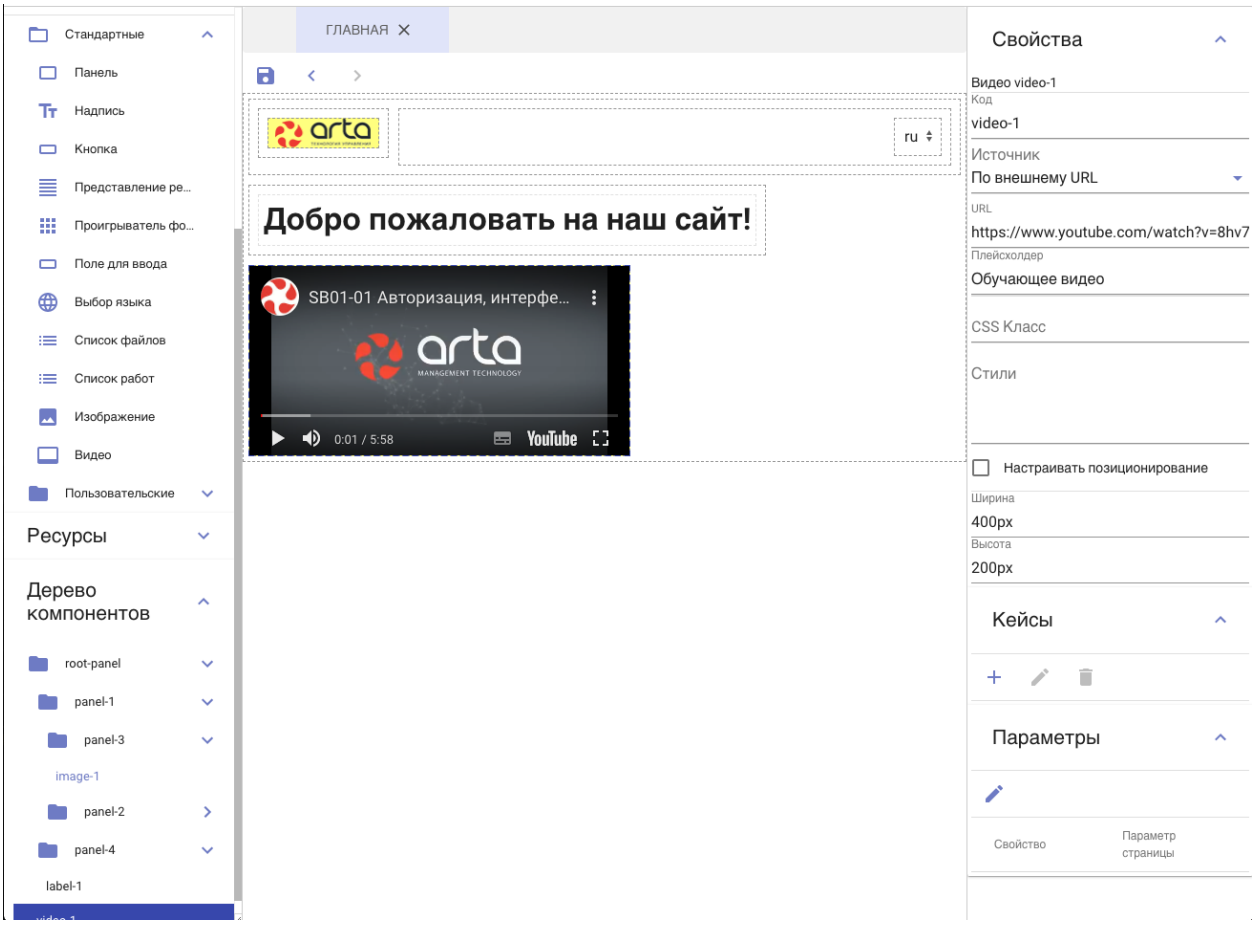
рис. Страница авторизации приложения

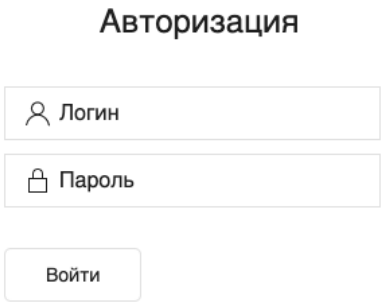
После авторизации попадаем на главную страницу:

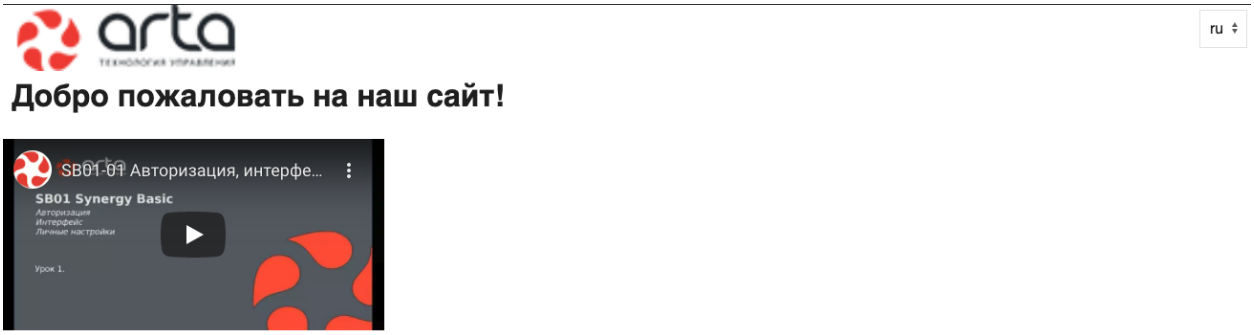
рис. Главная страница приложения

Приложение готово!


Страницы
root
Авторизация
Главная
Компоненты
Стандартные
Панель
Надпись
Кнопка
Представление рее...
Проигрыватель форм
Поле для ввода
Выбор языка
Список файлов
Список работ
Изображение
Видео
Пользовательские
Ресурсы
Дерево компонентов
ГЛАВНАЯ
ru
Добро пожаловать на наш сайт!
Свойства
Надпись label-1
Код
label-1
Текст
Добро пожаловать на наш сайт!
Тип
Заголовок
Размер
H2
Divider
Bullet
Line
Выравнивание
Вверх
Выравнивание текста
По центру
Стиль текста
По умолчанию
Жирный
Курсив
CSS Класс
Максимальная длина
0
Стили
Ширина


Стандартные
Панель
Надпись
Кнопка
Представление ре...
Проигрыватель фо...
Поле для ввода
Выбор языка
Список файлов
Список работ
Изображение
Видео
Пользовательские
Ресурсы
Дерево компонентов
root-panel
panel-1
panel-3
image-1
panel-2
panel-4
label-1
ГЛАВНАЯ
ru
Добро пожаловать на наш сайт!
SB01-01 Авторизация, интерфе...
arta
MANAGEMENT TECHNOLOGY
0:01 / 5:58
YouTube
Свойства
Видео video-1
Код
video-1
Источник
По внешнему URL
URL
https://www.youtube.com/watch?v=8hv7
Плейсхолдер
Обучающее видео
CSS Класс
Стили
Настраивать позиционирование
Ширина
400px
Высота
200px
Кейсы
Параметры
Свойство
Параметр страницы


Авторизация
Логин
Пароль
Войти


ru
Добро пожаловать на наш сайт!
SB01-01 Авторизация, интерфе...
SB01 Synergy Basic
Авторизация
Интерфейс
Личные настройки
Урок 1.

Глава 5

Перечень компонентов и их атрибутов

5.1 Виды компонентов

5.1.1 Стандартные

- **Панель** - специальный тип компонента, который может содержать в себе вложенные компоненты, в том числе и другие панели.
- **Надпись** - позволяет задать текстовый заголовок на компоненте.
- **Кнопка** - интерактивный компонент, при нажатии на который запускается связанное с ним событие или действие.
- **Представление реестра** - компонент, отображающий записи реестра в Synergy.
- **Проигрыватель форм** - компонент, отображающий форму документа из Synergy.
- **Поле для ввода** - компонент, позволяющий вводить текстовые данные (например, логин и пароль на странице авторизации)
- **Выбор языка** - компонент для смены локали приложения.
- **Список файлов** - компонент, отображающий список файлов папки в Хранилище Synergy.
- **Список файлов работ** - компонент, отображающий список файлов, приложенных к работе.
- **Список работ** - компонент, отображающий список работ авторизованного пользователя из Synergy.
- **Изображение** - добавление графических файлов на страницу приложения.
- **Видео** - добавление видеофайлов на страницу приложения
- **Дерево** - компонент, предназначенный для представления иерархической структуры записей.

5.1.2 Пользовательские

Компоненты, разработанные специально для одного или нескольких приложений конструктора.

5.2 Описание компонентов

5.2.1 Панель

Описание - компонент предназначен для группировки и настройки способа отображения компонентов и других панелей на странице. Сама страница представляет собой панель с кодом „root-panel“.

Специальные свойства

- Тип панели
  - Обычная
  - Горизонтальная
  - Вертикальная
  - Итератор

Для Обычной/Горизонтальная/Вертикальная:

- Выравнивание по горизонтали
- Выравнивание по вертикали
- Выравнивать содержимое

Для итератора:

- Источник
- URL

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

Не имеет собственных значений

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

Не имеет собственных значений

5.2.2 Надпись

Описание - компонент предназначен для размещения на странице статичного текста, недоступного пользователю на редактирование, но имеющего возможность меняться под влиянием внешних обработчиков событий.

Свойства

- Текст (с переводами)

- Тип (Текст/Заголовок)
- Стиль
- Размер
- Выравнивание
- Выравнивание текста
- Стиль текста

Входные параметры

- Текст
- Локализованный текст

Выходные параметры

- Текст
- Локализованный текст

Событие

- Клик

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить текст

Параметры действия: Источник:

- Произвольный текст (поле для ввода URL)
- Параметры страницы (поле для выбора параметра)
- Значения компонента (поле для выбора компонента)
- Локализованный текст (поле для ввода URL)
- Параметр итератора (поле для ввода параметра)
- Кастомный источник

5.2.3 Кнопка

Описание - компонент предназначен для осуществления назначенных на него событий или действий.

Свойства

- Текст (с переводами)
- Иконка
- Расположение иконки
- Тултип
- Заблокировано
- Размер
- Тип

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Клик

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Скрыть/показать
- Включить/Выключить
- Изменить подпись кнопки
- Изменить размер кнопки
- Изменить тип кнопки
- Изменить URL (для типа кнопки «ссылка»)

5.2.4 Поле для ввода

Описание - компонент предназначен для ввода пользователем текста на странице.

Свойства

- Значение по умолчанию
- Лейбл
- Иконка
- Плейсхолдер
- Настройки (Иконка справа/Не отображать вводимые символы/Обязательное/Многострочный/Заблокировано)
- Корректное значение (маска ввода)

Входные параметры

- Значение
- Значение по умолчанию
- Лейбл
- Плейсхолдер

Выходные параметры

- Значение
- Значение по умолчанию
- Лейбл
- Плейсхолдер

Событие

- Фокусировка на поле ввода

- Пустое поле ввода
- Заполнение значения
- Изменение значения
- Корректное значение
- Некорректное значение
- Нажатие кнопки Enter

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Включить/Выключить
- Отобразить/Скрыть вводимые символы
- Обязательное/Не обязательное
- Корректное/Не корректное значение

5.2.5 Выбор языка

Описание - компонент предназначен для смены локали

Свойства

- код локали

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Локаль

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Скрыть/показать

5.2.6 Изображение

Описание - компонент предназначен для добавления графических изображений на страницу.

Свойства

- Источник
- Плейсхолдер

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Клик

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить ссылку
- Изменить id файла из хранилища
- Скрыть/показать
- Изменить UUID записи по форме
- Изменить код компонента формы

5.2.7 Видео

Описание - компонент предназначен для добавления медиаконтента на страницу.

Свойства

- Источник
- Плейсхолдер

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Клик

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить ссылку
- Изменить id файла из хранилища
- Скрыть/показать
- Изменить UUID записи по форме
- Изменить код компонента формы

5.2.8 Представление реестра

Описание - компонент предназначен для отображения существующих реестров Synergy.

Свойства

- Код реестра
- Код фильтра
- Количество записей на странице
- Скрыть пагинатор

- Положение пагинатора
- Заголовки жирным шрифтом
- Чередующаяся заливка строк
- Отключить фильтр поиска

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Выбрать запись реестра
- Существует предыдущая страница
- Отсутствует предыдущая страница
- Существует следующая страница
- Отсутствует следующая страница

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить реестр
- Изменить фильтр реестра
- Изменить количество отображаемых записей
- Отобразить следующую страницу
- Отобразить предыдущую страницу
- Скрыть/показать
- Обновить данные
- registry_header_bold
- registry_paginator_position
- registry_hide_paginator
- registry_visible_cols
- registry_disable_filter_input
- registry_color_filling

5.2.9 Проигрыватель форм

Описание - компонент предназначен для отображения существующих форм Synergy.

Свойства

- Код формы
- Код представления
- Идентификатор данных по форме

- Открывать в режиме редактирования
- Создать запись реестра
- Активировать запись реестра
- Код реестра

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Успешная загрузка данных
- Изменение записи по форме
- Корректно заполнены поля
- Некорректно заполнены поля
- Успешное создание записи по форме
- Неуспешное создание записи по форме
- Успешное сохранение данных по форме
- Неуспешное сохранение данных по форме

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить форму
- Изменить запись
- Выбрать представление формы
- Изменить режим отображения
- Сохранить форму
- Создать запись по форме
- Скрыть/показать

5.2.10 Список файлов

Описание - компонент предназначен для отображения файлов из хранилища Synergy

Свойства

- Папка в хранилище
- Отображать название
- Отображать размер
- Отображать иконку
- Отображать дату изменения
- Отображать загрузившего

- Чередующаяся заливка строк
- Сортировать по полю
- Направление сортировки

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Выбрать запись из списка

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить папку
- Изменить сортировку
- Отображение даты изменения
- Отображение загрузившего
- Отображение размера
- Обновить данные
- Отображение названия
- Отображение иконок
- file_list_color_filling
- Скрыть/показать

5.2.11 Список файлов работ

Описание - компонент предназначен для отображения файлов, приложенных к работе

Свойства

- Идентификатор работы
- Отображать название
- Отображать иконку
- Отображать дату создания
- Чередующаяся заливка строк
- Сортировать по полю
- Направление сортировки

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

Не имеет собственных значений

5.2.12 Список работ

Описание - компонент предназначен для отображения существующих работ из Потоков работ Synergy.

Свойства

- Количество работ на страницце
- Фильтр работ
- Отображать удаленные
- Отображать завершенные
- Чередующаяся заливка строк
- Период работ

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

- Клик по определенной работе
- Двойной клик по определенной работе

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

- Изменить фильтр работ
- Изменить код фильтра
- Изменить период работ
- Изменить количество отображаемых записей
- Отображать завершенные работы
- Отображать удаленные работы
- Обновить список работ
- worklist_color_filling
- Скрыть/показать

5.2.13 Дерево

Описание - компонент предназначен для представления иерархической структуры записей.

Свойства

- Источник
- Отображение по умолчанию

- Отобразить счетчики

Входные параметры

Не имеет собственных значений

Выходные параметры

Не имеет собственных значений

Событие

Клик по элементу дерева

Специальные Действия над компонентом (для группы действий «Выполнить на странице»)

Скрыть/показать

Глава 6

Примеры

Глава 7

События конструктора

7.1 Общие события для компонентов

7.1.1 Событие скрытия компонента

```
  Событие скрытия компонента, может быть использовано для любого компонента.
  Пример генерации события:
  let event = {
      type: 'set_hidden',
      hidden: false
  };
  fire(event, 'код компонента, который хотим скрыть')
 /
SET_HIDDEN = 'set_hidden',
```

7.1.2 Событие блокировки компонента

```
  Событие блокировки компонента, может быть использовано для "Поле для ввода" (input) и "Кнопка"␣
↪(button)
  Пример генерации события:
  let event = {
      type: 'set_disabled',
      disabled: false
  }
  fire(event, 'код компонента, который хотим скрыть')
 /
SET_DISABLED = 'set_disabled',
```

7.1.3 Событие для изменения store Пользовательского компонента

```
  Событие для изменения store Пользовательского компонента
  Пример генерации события:
  let event = {
      type: 'change_custom_comp_store',
      store: {}
  };
  fire(event, 'код экземпляра пользовательского компонента, у которого нужно изменить store')
 /
CHANGE_CUSTOM_COMP_STORE = 'change_custom_comp_store',
```

7.1.4 Событие рендера кастомного компонента

```
  Событие рендера кастомного компонента
  Пример подписки на событие:
  addListener('render_custom_comp', 'my_form_code', (e) => { // ...Your Code })
 /
RENDER_CUSTOM_COMP = 'render_custom_comp',
```

7.1.5 Событие для отображения сообщения

```
  Событие для отображения сообщения. Принимает 2 параметра:
  @messageType 'error' | 'success' | 'info'
  @text: текст сообщения
  Пример генерации события:
  let event = {
      type: 'show_message',
      text: 'Текст сообщения',
      messageType: 'info'
  }
  fire(event, 'код компонента')
 /
SHOW_MESSAGE = 'show_message',
```

7.1.6 Событие для обновлении данных

Создан для компонентов, работающих со внешними ресурсами

```
  Событие для обновлении данных. Создан для тех компонент которые работают с внешними ресурсами
  Итератор, Преставление реестра, Список работ, Список файлов
  Пример генерации события:

  let event = {
      type: 'reload_data',
  }
  fire(event, 'my_reloadable_component');
 /
RELOAD_DATA = 'reload_data',
```

7.2 События для компонента «Список работ»

7.2.1 Событие для изменения отображаемого фильтра (централизованного)

```
  Событие для изменения отображаемого фильтра (пользовательского) в компоненте "Список работ"
  Пример генерации события:
  let event = {
      type: 'worklist_filter_code_change',
      userFilterCode: ''
  };
  fire(event, 'код компонента "Список работ", в котором нужно сменить фильтр')
 /
WORKLIST_FILTER_CODE_CHANGE = 'worklist_filter_code_change',
```

7.2.2 Изменение периода отображаемых работ

```
  Изменение периода отображаемых работ в компоненте "Список работ".
  Периоды:
   'inProgress'
   'lastQuarter'
   'lastMonth'
   'lastWeek'
   'today'
   'nextWeek'
   'nextMonth'
   'nextQuarter'
   'anyPeriod'
  Пример генерации события:
  let event = {
      type: 'worklist_period_change',
      period: 'anyPeriod'
  };
  fire(event, 'код компонента "Список работ"')
 /
WORKLIST_PERIOD_CHANGE = 'worklist_period_change',
```

7.2.3 Событие изменения сортировки списка работ

```
  Событие изменения сортировки списка работ.
  В качестве параметра принимает поле сортировки orderBy, которое может принимать значения:
       author
       left
       percent
       responsible
       start_date
       finish_date
  Пример генерации события:
  let event = {
      type: 'worklist_table_sort',
      order: 'author',
      orderBy: 'author'
  }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  fire(event, 'код компонента "Список работ"')
 /
WORKLIST_TABLE_SORT = 'worklist_table_sort',
```

7.2.4 Событие отображения удаленных работ

```
  Событие отображения удаленных работ в списке работ
  Пример генерации события:
  let event = {
      type: 'worklist_ext_show_deleted_change',
      showDeleted: 'showDeleted',
  }
  fire(event, 'код компонента "Список работ"')
 /
WORKLIST_SHOW_DELETED_CHANGE = 'worklist_ext_show_deleted_change',
```

7.2.5 Событие отображения завершенных работ

```
  Событие отображения завершенных работ в списке работ
  Пример генерации события:
  let event = {
      type: 'worklist_ext_show_completed_change',
      showCompleted: 'showCompleted',
  }
  fire(event, 'код компонента "Список работ"')
 /
WORKLIST_SHOW_COMPLETED_CHANGE = 'worklist_ext_show_completed_change',
```

7.2.6 Событие клика на конкретную работу

```
  Событие клика на конкретную работу в списке работ
  Пример подписки на событие:
  addListener('worklist_item_click', 'my_work_list_comp', (e) => { console.log(e) })
 /
WORKLIST_ITEM_CLICK = 'worklist_item_click',
```

7.2.7 Событие двойного клика по работе

```
 Событие клика на конкретную работу в списке работ
 Пример подписки на событие:
 addListener('worklist_item_dbl_click', 'workList-1', (e) => {
console.log('dblclick', e)
})
```

7.2.8 Событие клика на конкретный файл в компоненте «Список файлов работ»

```
Пример подписки на событие:
     addListener('work_file_list_select_item', 'my_work_file_list_code', (e) => { console.log(
↪"Selected file ", e.UUID) })
    /
   WORK_FILE_LIST_SELECT_ITEM = 'work_file_list_select_item',
```

7.2.9 Событие двойного клика на конкретный файл в компоненте «Список файлов работ»

```
Пример подписки на событие:
     addListener('work_file_list_dbl_select_item', 'my_work_file_list_code', (e) => { console.log(
↪"Selected file ", e.UUID) })
    /
   WORK_FILE_LIST_DBL_SELECT_ITEM = 'work_file_list_dbl_select_item',
```

7.2.10 Событие изменения количества отображаемых работ

```
  Событие изменения количества отображаемых работ на странице списка работ
  Пример генерации события:
  let event = {
      type: 'worklist_row_count_on_page',
      rowsPerPage: 10,
  }
  fire(event, 'код компонента "Список работ"')
 /
WORKLIST_ROW_COUNT_ON_PAGE_CHANGE = 'worklist_row_count_on_page',
```

7.2.11 Событие обновления списка работ

```
  Событие обновления списка работ
  Пример генерации события:
  let event = {
      type: 'worklist_refresh',
      shouldLoadData: true,
  }
  fire(event, 'код компонента "Список работ"')
 /
WORKLIST_REFRESH = 'worklist_refresh',

// TODO пока еще нет такого кейса
WORKLIST_COLOR_FILLING  = 'worklist_color_filling',
```

7.2.12 Событие для изменения идентификатора работы в компоненте «Список файлов работы»

```
Пример генерации события:

     let event = {
         type: 'work_file_list_change_uuid',
         actionID: 'actionID',
     }
     fire(event, 'код компонента "Список файлов работы"');
```

7.2.13 Событие для использования кастомного источника данных «Список работ»

```
response - результат выполнения "api/workflow/works/list_ext"
     Пример генерации события:
     let event = {
         type: 'worklist_add_custom_data',
         customData: response
     };
     fire(event, 'код компонента "Список работ"')
    /
   WORKLIST_ADD_CUSTOM_DATA = 'worklist_add_custom_data',
```

7.2.14 Событие для изменения источника для компонента «Список работ»

```
Пример генерации события:
     let event = {
         type: 'worklist_change_source',
         isCustom: boolean
     };
     fire(event, 'код компонента "Список работ"')
    /
   WORKLIST_CHANGE_SOURCE = 'worklist_change_source',
```

7.3 События для компонента «Представление реестра»

7.3.1 Событие для изменения отображаемого реестра по коду

```
  Событие для изменения отображаемого реестра по коду
  Пример генерации события:
  let event = {
      type: 'registry_code_change',
      code: 'example',
  }
  fire(event, 'код компонента "Представление реестра"')
 /
REGISTRY_CODE_CHANGE = 'registry_code_change',
```

7.3.2 Событие клика на конкретную запись

```
  Событие клика на конкретную запись в представлении реестра
  Пример подписки на событие:
  addListener('registry_item_click', 'my_form_code', (e) => {
       // e - объект события, в котором хранится информация о выбранной записи реестра
   })
 /
REGISTRY_ITEM_CLICK = 'registry_item_click',
```

7.3.3 Событие клика на конкретный файл в компоненте «Список файлов работ»

```

```

7.3.4 Событие успешной загрузки записей реестра

```
  Событие успешной загрузки записей реестра в представлении реестра
  Пример подписки на событие:
  addListener('registry_data_loaded', 'my_form_code', (e) => {...your code})
 /
REGISTRY_DATA_LOADED = 'registry_data_loaded',
```

7.3.5 Событие для перехода на указанную страницу

```
  Событие для перехода на указанную страницу в представлении реестра
  Пример генерации события:
  let event = {
      type: 'registry_page_change',
      page: 4,
  }
  fire(event, 'код компонента "Представление реестра"')
 /
REGISTRY_PAGE_CHANGE = 'registry_page_change',
```

7.3.6 Событие для перехода на следущую страницу

```
  Событие для перехода на следущую страницу в представлении реестра
  Пример генерации события:
  let event = {
      type: 'registry_page_next',
  }
  fire(event, 'код компонента "Представление реестра"')
 /
REGISTRY_PAGE_NEXT = 'registry_page_next',
```

7.3.7 Событие для перехода на предыдущую страницу

```
  Событие для перехода на предыдущую страницу в представлении реестра
  Пример генерации события:
  let event = {
      type: 'registry_page_back',
  }
  fire(event, 'код компонента "Представление реестра"')
 /
REGISTRY_PAGE_BACK = 'registry_page_back',
```

7.3.8 Событие для изменения количества отображаемых записей

```
  Событие для изменения количества отображаемых записей на одной странице в представлении реестра
  Пример генерации события:
  let event = {
      type: 'registry_row_count_change',
      rowCount: 10
  }
  fire(event, 'код компонента который хотим изменить')
 /
REGISTRY_ROW_COUNT_CHANGE = 'registry_row_count_change',

// TODO пока еще нет такого кейса
REGISTRY_HEADER_BOLD = 'registry_header_bold',

// TODO пока еще нет такого кейса
REGISTRY_PAGINATOR_POSITION  = 'registry_paginator_position',

// TODO пока еще нет такого кейса
REGISTRY_HIDE_PAGINATOR  = 'registry_hide_paginator',

// TODO пока еще нет такого кейса
REGISTRY_DISABLE_FILTER_INPUT  = 'registry_disable_filter_input',

// TODO пока еще нет такого кейса
REGISTRY_COLOR_FILLING  = 'registry_color_filling',
```

7.3.9 Событие изменения отображаемого кода фильтра реестра

```
  Событие изменения отображаемого кода фильтра реестра в представлении реестра
  Пример генерации события:
  let event = {
      type: 'registry_filter_change',
      filterCode: 'only_admin_users_records'
  }
  fire(event, 'код компонента который хотим изменить')
 /
REGISTRY_FILTER_CHANGE = 'registry_filter_change',
```

7.3.10 Событие, генерируемое при наличии предыдущей страницы

```
  Событие, генерируемое при наличии предыдущей страницы в представлении реестра
  Пример подписки на событие:
  addListener('has_previous_page', 'my_form_code', (e) => {...your code})
 /
HAS_PREVIOUS_PAGE = 'has_previous_page',
```

7.3.11 Событие, генерируемое при отсутствии предыдущей страницы

```
  Событие, генерируемое при отсутствии предыдущей страницы в представлении реестра
  Пример подписки на событие:
  addListener('no_previous_page', 'my_form_code', (e) => {...your code})
 /
NO_PREVIOUS_PAGE = 'no_previous_page',
```

7.3.12 Событие, генерируемое при наличии следующей страницы

```
  Событие, генерируемое при наличии следующей страницы в представлении реестра
  Пример подписки на событие
  addListener('has_next_page', 'my_form_code', (e) => {...your code})
 /
HAS_NEXT_PAGE = 'has_next_page',
```

7.3.13 Событие, генерируемое при отсутствии следующей страницы

```
  Событие, генерируемое при отсутствии следующей страницы в представлении реестра
  Примеры использовани в редакторе
  addListener('no_next_page', 'my_form_code', (e) => {...your code})
 /
NO_NEXT_PAGE = 'no_next_page',
```

7.4 События для страницы

7.4.1 Событие загрузки страницы

```
  Событие загрузки страницы.
  Пример подписки на событие:
  addListener('page_load', 'any_comp_code', (e) => { // ...Your Code })
 /
PAGE_LOAD = 'page_load',
```

7.4.2 Событие для перехода на другую страницу приложения

```
  Событие для перехода на другую страницу приложения с передачей ей параметров
  Пример генерации события:
  let event = {
      type: 'goto_page',
      pageCode: 'second_page',
      pageParams: [
          {
              pageParamName: 'Имя параметра страницы second_page',
              value: 'Значение параметра страницы'
          },
          {
              pageParamName: 'Имя параметра страницы second_page',,
              value: 'Значение параметра страницы'
          },
          ...
      ]
  }
  fire(event, 'any_comp_code')
 /
GOTO_PAGE = 'goto_page',
```

7.4.3 Событие для перехода на страницу авторизации

```
  Событие для перехода на страницу авторизации с передачей ей целевой страницы
  Пример генерации события:
  let event = {
      type: 'goto_page',
      sourcePageCode: 'код целевой страницы, на которую нужно перейти в случае успешной авторизации
↪'
  }
  fire(event, 'any_comp_code')
 /
GOTO_LOGIN_PAGE = 'goto_login_page',
```

7.4.4 Служебное событие заполнения параметров страницы

Не рекомендуется подписываться на это событие.

```
  Служебное событие заполнения параметров страницы, не рекомендуется подписываться на него.
 /
FILL_PARAM_DATA = 'fill_param_data',
```

7.5 События для компонентов «Надпись» и «Кнопка»

7.5.1 Событие для изменения текста, отображаемого в компоненте «Надпись»

```
  Событие для изменения текста, отображаемого в компоненте "Надпись". Параметр text принимает␣
↪локализированный текст.
  Пример генерации события:
  let event = {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      type: 'change_label',
      text: localizedText("test","test","test","test"),
  }
  fire(event, 'код компонента "Надпись"')
 /
CHANGE_LABEL = 'change_label',
```

7.5.2 Событие для имитации клика на компонент «Надпись»

```
  Событие для имитации клика на компонент "Надпись".
  Пример генерации события:
  let event = {
      type: 'label_click'
  }
  fire(event, 'код компонента "Надпись"')
 /
LABEL_CLICK = 'label_click',
```

7.5.3 Событие для имитации клика на компонент «Кнопка»

```
  Событие для имитации клика на компонент "Кнопка".
  Пример генерации события:
  let event = {
      type: 'button_click'
  }
  fire(event, 'код компонента "Кнопка"')
 /
BUTTON_CLICK = 'button_click',
```

7.5.4 Событие для изменения подписи компонента «Кнопка»

```
  Событие для изменения подписи компонента "Кнопка". Параметр text принимает локализированный␣
↪текст.
  Пример генерации события:
  let event = {
      type: 'button_change_text',
      text: localizedText("test","test","test","test"),
  }
  fire(event, 'код компонента "Кнопка"')
 /
BUTTON_CHANGE_TEXT = 'button_change_text',
```

7.5.5 Событие для изменения размера компонента «Кнопка»

```
  Событие для изменения размера компонента "Кнопка". Параметр size может принимать значения:␣
↪(default | uk-button-small | uk-button-large)
  Пример генерации события:
  let event = {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      type: 'button_change_size',
      text: 'uk-button-small',
  }
  fire(event, 'код компонента "Кнопка"')
 /
BUTTON_CHANGE_SIZE = 'button_change_size',
```

7.5.6 Событие для изменения типа компонента «Кнопка»

```
  Событие для изменения типа компонента "Кнопка".
  Параметр type может принимать значения: (uk-button-text | uk-button-default | uk-button-primary␣
↪| uk-button-secondary | uk-button-danger | uk-button-link)
  Пример генерации события:
  let event = {
      type: 'button_change_size',
      text: 'uk-button-text',
  }
  fire(event, 'код компонента "Кнопка")
 /
BUTTON_CHANGE_TYPE = 'button_change_type',

// TODO пока еще нет такого кейса
BUTTON_CHANGE_ICON = 'button_change_icon',

// TODO пока еще нет такого кейса
BUTTON_CHANGE_ICON_POSITION = 'button_change_icon_position',

// TODO пока еще нет такого кейса
BUTTON_CHANGE_URL = 'button_change_url',
```

7.6 События для компонента «Проигрыватель форм»

7.6.1 Событие для отображения указанной записи по ее идентификатору

```
  Событие для отображения указанной записи по ее идентификатору в проигрывателе форм
  Пример генерации события:
  let event = {
      type: 'show_form_data',
      dataId: '155',
  }
  fire(event, 'код компонента "Проигрыватель форм"')
 /
SHOW_FORM_DATA = 'show_form_data',
```

7.6.2 Событие для отображения указанной формы по ее коду

```
  Событие для отображения указанной формы по ее коду в проигрывателе форм
  Пример генерации события:
  let event = {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      type: 'show_form',
      formCode: 'reg_form',
  }
  fire(event, 'код компонента "Проигрыватель форм"')
 /
SHOW_FORM = 'show_form',
```

7.6.3 Событие для отображения указанного представления текущей формы

```
  Событие для отображения указанного представления текущей формы в проигрывателе форм
  Пример генерации события:
  let event = {
      type: 'show_form_view',
      viewCode: 'reg_form_view_for_mobile',
  }
  fire(event, 'код компонента "Проигрыватель форм"')
 /
SHOW_FORM_VIEW = 'show_form_view',
```

7.6.4 Событие для изменения режима отображения формы

```
  Событие для изменения режима отображения формы в проигрывателе форм: включение доступности␣
↪редактирования данных
  Пример генерации события:
  let event = {
      type: 'set_form_editable',
      editable: false,
  }
  fire(event, 'код компонента "Проигрыватель форм"')
 /
SET_FORM_EDITABLE = 'set_form_editable',
```

7.6.5 Событие изменения записи по форме

```
  Событие изменения записи по форме в проигрывателе форм
  Пример подписки на событие:
  addListener('changed_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
CHANGED_FORM_DATA = 'changed_form_data',
```

7.6.6 Событие успешной загрузки данных

```
  Событие успешной загрузки данных в проигрывателе форм
  Пример подписки на событие:
  addListener('loaded_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
LOADED_FORM_DATA = 'loaded_form_data',
```

7.6.7 Событие для сохранения данных по форме

```
  Событие для сохранения данных по форме, применяется при изменении уже существующей записи.␣
↪Входные параметры:
  @success: (dataId: string, documentId: string) => void;
  @error: (status: number, error: string) => void;
  Пример генерации события:

  let event = {
      type: 'save_form_data',
      success: (dataId, documentId) => { console.log("SUCCESS! ", dataId); },
      error: (status, error) => {cosnole.error("FAILED! ", error)}
  }
  fire(event, 'код компонента "Проигрыватель форм"');
 /
SAVE_FORM_DATA = 'save_form_data',
```

7.6.8 Событие для создания новой записи

```
  Событие для создания новой записи по форме. Выполняет создание и сохранение новой записи реестра␣
↪по форме. Входные параметры:
  @registryCode?: string;
  @activate?: boolean;
  @success: (dataId: string, documentId: string) => void;
  @error: (status: number, error: string) => void;
  Пример генерации события:

  let event = {
      type: 'create_form_data',
      registryCode: 'my_registry',
      activate: true,
      success: (dataId, documentId) => { console.log("SUCCESS! ", dataId); },
      error: (status, error) => {cosnole.error("FAILED! ", error)}
  }
  fire(event, 'код компонента "Проигрыватель форм"');
 /
CREATE_FORM_DATA = 'create_form_data',
```

7.6.9 Событие для проверки валидности данных

```
  Событие для проверки валидности данных формы, отображенной в проигрывателе форм. Входные␣
↪параметры:
  @errors: (errors: array) => void;
  Пример генерации события:

  let event = {
      type: 'validate_form_data',
      callback: (errors) => { errors.map(err => console.error(err); },
  }
  fire(event, 'код компонента "Проигрыватель форм"');
 /
VALIDATE_FORM_DATA = 'validate_form_data',
```

7.6.10 Событие успешного сохранения существующей записи

```
  Событие успешного сохранения существующей записи по форме, генерируется в проигрывателе форм.
  Пример подписки на событие:
  addListener('saved_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
SAVED_FORM_DATA = 'saved_form_data',
```

7.6.11 Событие неуспешного сохранения существующей записи

```
  Событие неуспешного сохранения существующей записи по форме, генерируется в проигрывателе форм
  Пример подписки на событие:
  addListener('not_saved_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
NOT_SAVED_FORM_DATA = 'not_saved_form_data',
```

7.6.12 Событие успешного создания новой записи

```
  Событие успешного создания новой записи по форме, генерируется в проигрывателе форм.
  Пример подписки на событие:
  addListener('created_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
CREATED_FORM_DATA = 'created_form_data',
```

7.6.13 Событие неуспешного создания новой записи

```
  Событие неуспешного создания новой записи по форме, генерируется в проигрывателе форм.
  Пример подписки на событие:
  addListener('not_created_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
NOT_CREATED_FORM_DATA = 'not_created_form_data',
```

7.6.14 Событие валидности данных

```
  Событие валидности данных в проигрывателе форм. Генерируется в тот момент, когда все поля формы␣
↪в проигрывателе становятся валидными.
  Пример подписки на событие:
  addListener('valid_form_data', 'my_form_code', (e) => { // ...Your Code })
 /
VALID_FORM_DATA = 'valid_form_data',
```

7.6.15 Событие невалидности данных

```
  Событие невалидности данных в проигрывателе форм. Генерируется в тот момент, когда хотя бы одно␣
↪поле формы в проигрывателе становится невалидным.
  Пример подписки на событие:
  addListener('not_valid_form_data', 'my_form_code', (e) => { // ...Your Code })
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 /
NOT_VALID_FORM_DATA = 'not_valid_form_data',
```

7.7 События для компонента «Поле ввода»

7.7.1 Событие установки фокуса на компонент «Поле ввода»

```
  Событие установки фокуса на компонент "Поле ввода"
  Пример подписки на событие:
  addListener('focused_input', 'my_input_code', (e) => { // ...Your Code })
 /
FOCUSED_INPUT = 'focused_input',
```

7.7.2 Событие заполнения поля ввода

```
  Событие заполнения поля ввода. Вызывается в тот момент, когда при изменении данных компонента
↪"Поле ввода" в него вводятся символы.
  Пример подписки на событие:
  addListener('filled_input', 'my_input_code', (e) => { // ...Your Code })
 /
FILLED_INPUT = 'filled_input',
```

7.7.3 Событие очистки поля ввода

```
  Событие очистки поля ввода. Вызывается в тот момент, когда при изменении данных компонента "Поле␣
↪ввода" в нем не осталось символов.
  Пример подписки на событие:
  addListener('filled_input', 'my_input_code', (e) => { // ...Your Code })
 /
NOT_FILLED_INPUT = 'not_filled_input',
```

7.7.4 Событие изменения содержимого

```
  Событие изменения содержинмого в компоненте "Поле ввода".
  Пример подписки на событие:
  addListener('value_changed_input', 'my_input_code', (e) => { // ...Your Code })
 /
VALUE_CHANGED_INPUT = 'value_changed_input',
```

7.7.5 Событие валидности данных

```
  Событие валидности данных в компоненте "Поле ввода"
  Пример подписки на событие:
  addListener('valid_input', 'my_input_code', (e) => { // ...Your Code })
 /
VALID_INPUT = 'valid_input',
```

7.7.6 Событие невалидности данных

```
  Событие невалидности данных в компоненте "Поле ввода"
  Пример подписки на событие:
  addListener('valid_input', 'my_input_code', (e) => { // ...Your Code })
 /
NOT_VALID_INPUT = 'not_valid_input',
```

7.7.7 Событие нажатия на клавишу ENTER при фокусировке на поле ввода

```
  Событие нажатия на клавишу ENTER при фокусировке на поле ввода.
  Пример подписки на событие:
  addListener('text_input_key_down_enter', 'my_input_code', (e) => { // ...Your Code })
 /
TEXT_INPUT_KEY_DOWN_ENTER = 'text_input_key_down_enter',
```

7.7.8 Событие для изменения заголовка компонента

```
  Событие для изменения заголовка компонента "Поле ввода"
  Пример генерации события:

  let event = {
      type: 'input_label_change',
      label: localizedText('defaultText', 'Rus', 'Kaz', 'Eng'),
  }
  fire(event, 'my_input_code');
 /
INPUT_LABEL_CHANGE = 'input_label_change',

// TODO пока еще нет такого кейса
INPUT_ICON_CHANGE = 'input_icon_change',
```

7.7.9 Событие для изменения плейсхолдера

```
  Событие для изменения плейсхолдера в компоненте "Поле ввода"
  Пример генерации события.

  let event = {
      type: 'input_placeholder_change',
      placeholder: localizedText('defaultText', 'Rus', 'Kaz', 'Eng'),
  }
  fire(event, 'my_input_code');
 /
INPUT_PLACEHOLDER_CHANGE = 'input_placeholder_change',
```

7.7.10 Событие для изменения содержимого по умолчанию

```
  Событие для изменения содержимого по умолчанию в компоненте "Поле ввода"
  Пример генерации события:

  let event = {
      type: 'input_default_change',
      defaultValue: 'Любой текст',
  }
  fire(event, 'my_input_code');
 /
INPUT_DEFAULT_CHANGE = 'input_default_change',
```

7.7.11 Событие для скрытия/отображения введенных символов в компоненте «Поле ввода»

```
  Событие для скрытия/отображения введенных символов в компоненте "Поле ввода"
  Пример генерации события:

  let event = {
      type: 'input_password_change',
      password: false,
  }
  fire(event, 'my_input_code');
 /
INPUT_PASSWORD_CHANGE = 'input_password_change',
```

7.7.12 Событие для включения/отключения обязательности в компоненте «Поле ввода»

```
  Событие для включения/отключения обязательности в компоненте "Поле ввода"
  Пример генерации события:

  let event = {
      type: 'input_required_change',
      required: true,
  }
  fire(event, 'my_input_code');
 /
INPUT_REQUIRED_CHANGE = 'input_required_change',
```

7.7.13 Событие для включения/отключения подсветки компонента «Поле ввода»

```
  Событие для включения/отключения подсветки компонента "Поле ввода"
  Пример генерации события:

  let event = {
      type: 'input_highlight',
      error: true,
  }
  fire(event, 'my_input_code');
```

(continues on next page)

(продолжение с предыдущей страницы)

```
 /
INPUT_HIGHLIGHT = 'input_highlight',
```

7.8 События для компонента «Список файлов»

7.8.1 Событие для изменения отображаемой папки

```
  Событие для изменения отображаемой папки в компоненте "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_uuid',
      error: true,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_UUID = 'file_list_change_uuid',
```

7.8.2 Событие для включения/отключения отображения иконок

```
  Событие для включения/отключения отображения иконок в компоненте "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_show_icon',
      showIcon: false,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_SHOW_ICON = 'file_list_change_show_icon',
```

7.8.3 Событие для включения/отключения отображения названий файлов

```
  Событие для включения/отключения отображения названий файлов в компоненте "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_show_name',
      showName: false,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_SHOW_NAME = 'file_list_change_show_name',
```

7.8.4 Событие для включения/отключения отображения авторов файлов

```
  Событие для включения/отключения отображения авторов файлов в компоненте "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_show_size',
      showSize: false,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_SHOW_SIZE = 'file_list_change_show_size',
```

7.8.5 Событие для включения/отключения отображения авторов файлов в компоненте «Список файлов»

```
  Событие для включения/отключения отображения авторов файлов в компоненте "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_show_size',
      showUploader: false,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_SHOW_UPLOADER = 'file_list_change_show_uploader',
```

7.8.6 Событие для включения/отключения отображения даты изменения файлов в компоненте «Список файлов»

```
  Событие для включения/отключения отображения даты изменения файлов в компоненте "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_show_date',
      showDate: false,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_SHOW_DATE = 'file_list_change_show_date',
```

7.8.7 Событие для изменения видимости компонента «Список файлов»

```
  Событие для изменения видимости компонента "Список файлов"
  Пример генерации события:

  let event = {
      type: 'file_list_change_hidden',
      hidden: false,
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_HIDDEN = 'file_list_change_hidden',
```

7.8.8 Событие для изменения сортировки файлов в компоненте «Список файлов»

```
  Событие для изменения сортировки файлов в компоненте "Список файлов". Принимает параметры:
  @sortBy 'name'|'uploaded'|'dateChanged'|'format'
  @sortType 1 = По возрастанию, 0 = По убыванию
  Пример генерации события:

  let event = {
      type: 'file_list_change_sort',
      sortBy: 'name',
      sortType: 1
  }
  fire(event, 'код компонента "Список файлов"');
 /
FILE_LIST_CHANGE_SORT = 'file_list_change_sort',
```

7.8.9 Событие клика на конкретный файл в компоненте «Список файлов»

```
  Событие клика на конкретный файл в компоненте "Список файлов"
  Пример подписки на событие:
  addListener('file_list_select_item', 'my_file_list_code', (e) => { console.log("Selected file ",␣
↪e.UUID) })
 /
FILE_LIST_SELECT_ITEM = 'file_list_select_item',

// TODO пока еще нет такого кейса
FILE_LIST_COLOR_FILLING  = 'file_list_color_filling',
```

7.9 События при авторизации пользователя

7.9.1 Событие успешной авторизации пользователя в приложении

```
  Событие успешной авторизации пользователя в приложении
  Пример подписки на событие:
  addListener('auth_success', 'any_comp_code', (e) => { console.log("Auth Success") })
 /
AUTH_SUCCESS = 'auth_success',
```

7.9.2 Событие неуспешной авторизации пользователя в приложении

```
  Событие неуспешной авторизации пользователя в приложении
  Пример подписки на событие:
  addListener('auth_failure', 'any_comp_code', (e) => { console.log("Auth Fail!", e) })
 /
AUTH_FAILURE = 'auth_failure',
```

7.10 События для компонента «Изображение»

7.10.1 Событие для изменения ссылки на изображение

```
  Событие для изменения ссылки на изображение, отображаемое в компоненте "Изображение"
  Пример генерации события:

  let event = {
      type: 'image_change_url',
      url: 'http://example.com/example.jpg'
  }
  fire(event, 'my_image_comp_code');
 /
IMAGE_CHANGE_URL = 'image_change_url',
```

7.10.2 Событие для изменения идентификатора записи по форме

```
  Событие для изменения идентификатора записи по форме, в которой расположено изображение,␣
↪отображаемое в компоненте "Изображение"
  Пример генерации события:

  let event = {
      type: 'image_change_form_uuid',
      uuid: '100'
  }
  fire(event, 'my_img_code');
 /
IMAGE_CHANGE_FORM_UUID = 'image_change_form_uuid',
```

7.10.3 Событие для изменения кода компонента формы

```
  Событие для изменения кода компонента формы, в котором расположено изображение, отображаемое в␣
↪компоненте "Изображение"
  Пример генерации события:

  let event = {
      type: 'image_change_form_data_id',
      identifier: 'cmp-code'
  }
  fire(event, 'my_img_code');
 /
IMAGE_CHANGE_FORM_DATA_ID = 'image_change_form_data_id',
```

7.10.4 Событие для изменения идентификатора файла изображения

```
  Событие для изменения идентификатора файла изображения, отображаемого в компоненте "Изображение"
  Пример генерации события:

  let event = {
      type: 'image_change_store_id',
```

(continues on next page)

(продолжение с предыдущей страницы)

```
      identifier: 'xxx-xxxxx-xxxx-xxx-xxxxx'
  }
  fire(event, 'my_img_code');
 /
IMAGE_CHANGE_STORE_ID = 'image_change_store_id',
```

7.10.5 Событие клика по компоненту «Изображение»

```
  Событие клика по компоненту "Изображение"
  Пример подписки на событие:
  addListener('image_event_on_click', 'img_comp_code', (e) => { console.log("Auth Fail!", e) })
 /
IMAGE_EVENT_CLICK = 'image_event_on_click',
```

7.11 События для модальной панели и итератора

7.11.1 Событие клика вне модальной панели

```
  Событие клика вне модальной панели.
  Примеры использовани в редакторе
  addListener('out_click', 'modal_panel_code', (e) => { console.log("Auth Fail!", e) })
 /
OUT_CLICK = 'out_click',
```

7.11.2 Событие открытия модального окна через скрипт

```
fire({
  type: 'display_view_modal',
  hidden: false,
}, 'код_модального_окна');
```

7.11.3 Событие для изменения кода реестра

```
  Событие для изменения кода реестра, являющегося источником данных для компонента "Итератор"
  Пример генерации события:

  let event = {
      type: 'change_repeater_registry_code',
      registryCode: 'my_reg_code'
  }
  fire(event, 'my_repeater_code');
 /
CHANGE_REPEATER_REGISTRY_CODE = 'change_repeater_registry_code',
```

7.11.4 Событие для изменения кода фильтра реестра

```
  Событие для изменения кода фильтра реестра, являющегося источником данных для компонента
↪"Итератор"
  Пример генерации события:

  let event = {
      type: 'change_repeater_filter_code',
      filterCode: 'my_filter_code'
  }
  fire(event, 'my_repeater_code');
 /
CHANGE_REPEATER_FILTER_CODE = 'change_repeater_filter_code',
```

7.11.5 Событие для изменения кода компонента - динамической таблицы

```
  Событие для изменения кода компонента - динамической таблицы, являющейся источником данных для␣
↪компонента "Итератор"
  Пример генерации события:

  let event = {
      type: 'change_repeater_table_id',
      tableId: 'my_dyntable_code'
  }
  fire(event, 'my_repeater_code');
 /
CHANGE_REPEATER_TABLE_ID = 'change_repeater_table_id',
```

7.11.6 Событие для изменения идентификатора записи по форме

```
  Событие для изменения идентификатора записи по форме, динамическая таблица в которой является␣
↪источником данных для компонента "Итератор"
  Пример генерации события:

  let event = {
      type: 'change_repeater_data_id',
      dataId: '5'
  }
  fire(event, 'my_repeater_code');
 /
CHANGE_REPEATER_DATA_ID = 'change_repeater_data_id',
```

7.11.7 Событие для изменения внешней ссылки на источник данных

```
  Событие для изменения внешней ссылки на источник данных для компонента "Итератор"
  Пример генерации события:

  let event = {
      type: 'change_repeater_url',
      url: 'http://example.com/arrays.json'
  }
```

(continues on next page)

(продолжение с предыдущей страницы)

```
  fire(event, 'my_repeater_code');
 /
CHANGE_REPEATER_URL = 'change_repeater_url',
```

7.11.8 Событие для изменения параметров поиска элементов

```
  Событие для изменения параметров поиска элементов, отображаемых в компоненте "Итератор"
  Пример генерации события:

  let event = {
      type: 'change_repeater_search_params',
      params: '?onlyImages=true&bigText=false'
  }
  fire(event, 'my_repeater_code');
 /
CHANGE_REPEATER_SEARCH_PARAMS = 'change_repeater_search_params',
```

7.12 События для компонента «дерево»

7.12.1 Событие обновления дерева

```
  Пример вызова события:

  fire({
      type: 'reload_tree'
  }, 'my_tree_comp')
 /
RELOAD_TREE = 'reload_tree',
```

7.12.2 Событие клика по элементу дерева

```
  Пример вызова события

  fire({
      type: 'tree_item_click',
      item: '311d1bd0-4f45-442c-ac5d-daac30352213' || {
           id,
           children,
           filterData: {
               filterType,
           },
           hasChildren,
           name,
           ...
       },
       data: getCompByCode('my_tree_comp').data
  }, 'my_tree_comp')
 /
TREE_ITEM_CLICK = 'tree_item_click',
```

7.12.3 Событие установки данных дерева (изменяет всю структуру)

```
  Пример вызова события

  const treeData = [{
       id: 'myTreeItem',
       name: 'Первый президент',
       hasChildren: true,
       children: [{
           id: 'myTreeItem-child',
           name: 'Дочка первого президента',
           hasChildren: false,
           children: [],
           counters: undefined
       }],
       counters: {
           total: 10,
           new: 2
       } || undefined
  }]

  Если counters === undefined, то его можно не передавать в объекте

  fire({
      type: 'tree_set_data',
      data: treeData
  }, 'my_tree_comp')
 /
TREE_SET_DATA = 'tree_set_data',
```

7.12.4 Событие для установки дочерних элементов элементам дерева (изменяет структуру только указанных элементов - parentId)

```
  Пример вызова события

const childElements = [{
 id: 'childElement',
 name: 'Дочерний элемент',
 hasChildren: false,
 children: []
 }]

  fire({
      type: 'tree_set_children',
      parentId: '311d1bd0-4f45-442c-ac5d-daac30352213',
      data: childElements
  }, 'my_tree_comp')
 /
TREE_SET_CHILDREN = 'tree_set_children'
```

7.12.5 Событие для установки массива ID открытых эелментов дерева

```
  Пример вызова события

  fire({
      type: 'tree_set_expanded',
      expanded: ['1', '2', '33']
  })
 /
TREE_SET_EXPANDED = 'tree_set_expanded'
```