Документация

Выпуск 1.08

Руководство разработчика XSQUARE - RAD 2.7

мая 18, 2023

Содержание

ГЛАВА 1

Общие сведения

Данное руководство описывает как использовать среду разработки XRAD для построения веб приложений.

1.1 Для кого это руководство

Руководство предназначено для разработчиков, целью которых является построение веб приложения с дата-центрической архитектурой (основа - БД) с помощью инструментов разработки XRAD. В руководстве описано как использовать инструменты для построения, отладки, управления и разверстки приложения.

От читателя ожидается базовое знание концепции работы реляционных баз данных, знание основ html и js.

ГЛАВА 2

Быстрый старт

В данном разделе Вы познакомитесь с общей концепцией XRAD. Данный раздел описывает основные компоненты среды разработки, как поменять информацию в своем профиле и настройки среды.

2.1 Введение в XRAD

XRAD обеспечивает разработчика всеми инструментами для построение приложения в единой, расширяемой платформе на базе PostgreSQL.

2.1.1 Что такое XRAD?

В современном мире для разработки веб приложения часто требуется наличие нескольких разработчиков, каждый из которых отвечал бы за отдельные компоненты системы. Эти компоненты делятся на пользовательский интерфейс (frontend), базу данных (backend) и связующее звено, которое отвечает за логику приложения (middleware). Все эти компоненты взаимозависимы друг от друга и без четко определенных границ ответственности могут нарушать логику работы приложения. Данные границы и зоны ответственности определяет архитектор приложения, человек от которого требуется высокий уровень компетенции в понимании работы каждого из компонентов. Не стоит также забывать про администрирование такой системы и поддержании работоспособности, что требует наличия системного администратора.

Наличие такой команды однозначно увеличивает цену и уменьшает скорость разработки приложения.

Платформа XRAD предлагает кардинально другой подход к разработке. XRAD - это платформа для быстрой разработки приложений, разработанная ООО “Хи-Квадрат” (Xsquare Rapid Application Development).

XRAD - декларативная среда для разработки и развертывания веб-приложений, ориентированных на базы данных. Благодаря встроенным функциям, таким как темы пользовательского интерфейса, элементы управления навигацией, обработчики форм и гибкие отчеты, XRAD ускоряет процесс разработки приложений.

2.1.2 Как работает XRAD?

XRAD предоставляет все инструменты, необходимые для создания приложений на единой расширяемой платформе, которая работает на PostgreSQL.

XRAD использует простую 3-уровневую архитектуру, в которой запросы отправляются из браузера через веб-сервер в базу данных. Вся обработка, манипулирование данными и бизнес-логика выполняются в базе данных. Эта архитектура гарантирует доступ к данным с нулевой задержкой, высочайшую производительность и масштабируемость «из коробки».

XRAD устанавливается в виде наборов компонентов:

- Веб сервер, который отвечает за обработку пользовательских запросов и подготовку структуры страницы для рендеринга на клиенте.
- Веб приложение, которое обрабатывает полученную от веб сервера структуру страницы и отрисовывает её.
- Отдельное веб приложение, для разработчика, которое предоставляет инструменты разработки.
- Отдельная база PostgreSQL, которая хранит метаданные приложения.

Независимо от того, запускаете ли вы среду разработки XRAD или приложение, созданное с использованием XRAD - процесс один и тот же. Ваш браузер отправляет запрос, который преобразуется в соответствующий вызов кода на стороне БД. После того, как база данных обработает код, результаты будут переданы обратно в ваш браузер в виде JSON структуры, на базе которой веб приложение отрисует страницу. Этот цикл происходит каждый раз, когда вы запрашиваете или отправляете страницу.

Также платформа XRAD осуществляет следующие задачи:

- Управление состоянием сессии
- Производить аутентификацию и авторизацию
- Управляет поведением перехода по страницам
- Осуществляет проверку введенных данных

2.2 Введение в среду разработки

Среда разработки предоставляет функционал для разработки страниц и управления приложения.

2.2.1 Аутентификация

Для входа в приложение разработчику необходимо ввести свой логин и пароль. Для входа в приложение:

1. Откройте страницу аутентификации в браузере:

```
https://hostname/#/login/
```

Где hostname - наименование домена где был установлен XRAD.

Откроется страница аутентификации.

Введите следующие данные

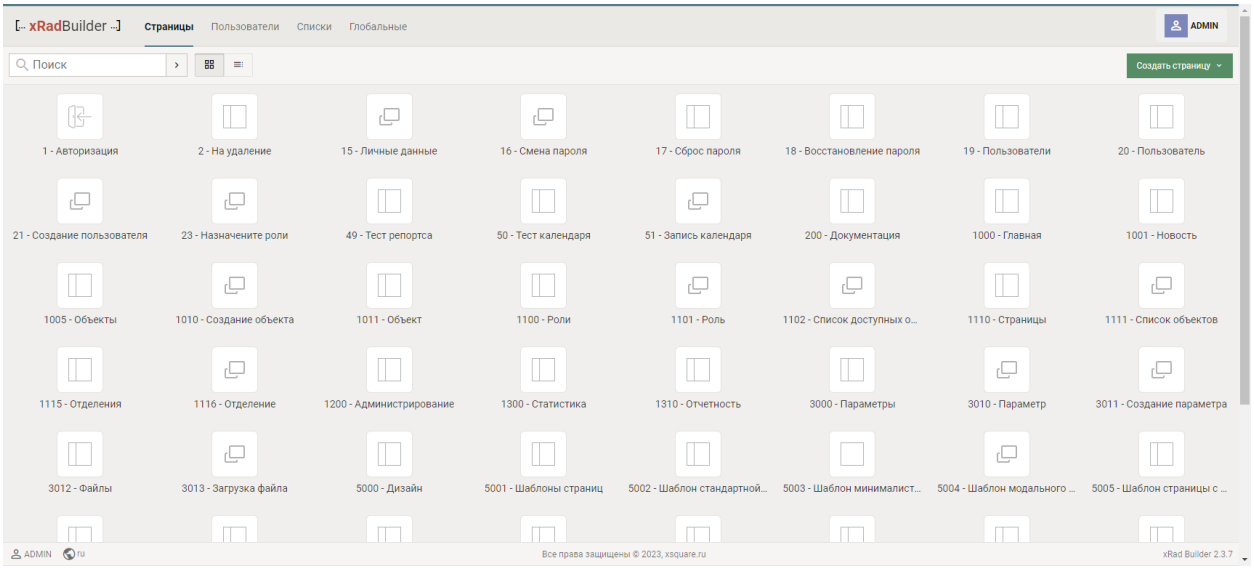
- Имя пользователя - укажите свой логин
- Пароль - укажите свой пароль

Нажмите кнопку “Авторизация”

В случае успешной авторизации откроется главная страница среды разработки

2.2.2 Основные элементы управления среды

После успешной авторизации откроется главная страница среды разработки. На главной странице предоставлен список страниц приложения и основные элементы навигации по среде. Далее рассмотри более подробно каждый элемент управления.



Верхняя панель управления включает в себя следующие вкладки :

- Страницы - перечень страниц приложения
- Пользователи - список пользователей среды разработки
- Списки - перечень предопределенных списков, используемых для работы компонентов приложения и навигации.
- Глобальные - настройки и глобальные параметры

Вкладка страницы

На данной вкладке располагается перечень всех разработанных страниц приложения. На данной вкладке разработчику доступен поиск страниц по наименованию, создание новой страницы

Для поиска страницы необходимо ввести название искомой страницы в поле поиск и нажать на кнопку поиска («>»). После чего система отфильтрует перечень страниц по заданному критерию поиска.

Система позволяет переключить вид списка страниц на табличное представление. Для этого необходимо нажать на кнопку табличного представления рядом с кнопкой поиска.

Для создания новой страницы необходимо нажать на кнопку “Создать страницу”. Подробнее о процессе создания см. *Управление страницами*.

Вкладка пользователи

На данной вкладке перечислен список пользователей среды разработки. Тут представлены возможности по созданию/редактированию/удалению пользователей. Подробнее см. *Управление пользователями*.

Вкладка списки

На данной вкладке располагается информация о предопределенных списка системы. Списки используются для построения навигации внутри приложения, отображения компонентов и прочего. Более подробно см. *Управление списками*.

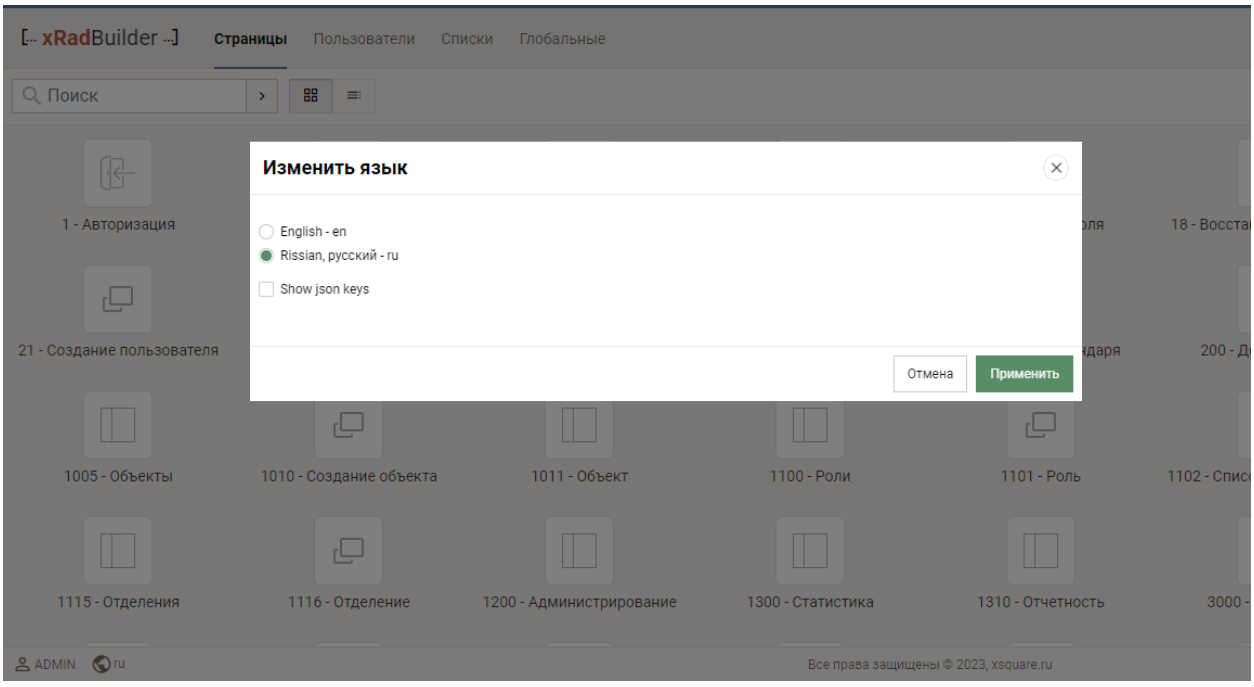
Вкладка глобальные

На данной вкладке располагаются глобальные компоненты приложения такие как:

- *Настройки приложения*
- *Глобальные процессы*
- *Глобальные переменные*
- *Схемы аутентификации*
- *Схемы авторизации*
- Экспорт/импорт

2.2.3 Смена локализации приложения

Разработчику доступна выбора локализации приложения. Для смены локализации необходимо в левом нижнем углу нажать на иконку глобуса, после чего выбрать необходимую локализацию.



ГЛАВА 3

Основные концепции среды разработки

Чтобы эффективно использовать XRAD разработчики должны понимать некоторые ключевые понятия, включая управление дизайном пользовательского интерфейса, просмотр визуализированных страниц приложений, понимание обработки и визуализации страниц, управление состоянием сеанса, управление значениями и состояния сеанса.

3.1 Общая концепция приложения

3.1.1 Что такое приложение?

Приложение XRAD - это HTML интерфейс, который существует поверх объектов базы данных: таблиц и процедур. Приложение в среде XRAD логически и физически разделяется на две базы данных: база данных отвечающая за бизнес логику и база данных с набором метаданных интерфейса приложения. Среда исполнения XRAD (pghs) объединяет данные двух баз и предоставляет конечному пользователю веб интерфейс, который отображает бизнес данные и процессы в соответствии с тем как было описано разработчиком.

Для конечного пользователя приложение это набор страниц, которые отображают и позволяют вводить данные посредством различных компонентом.

3.1.2 Что такое страница?

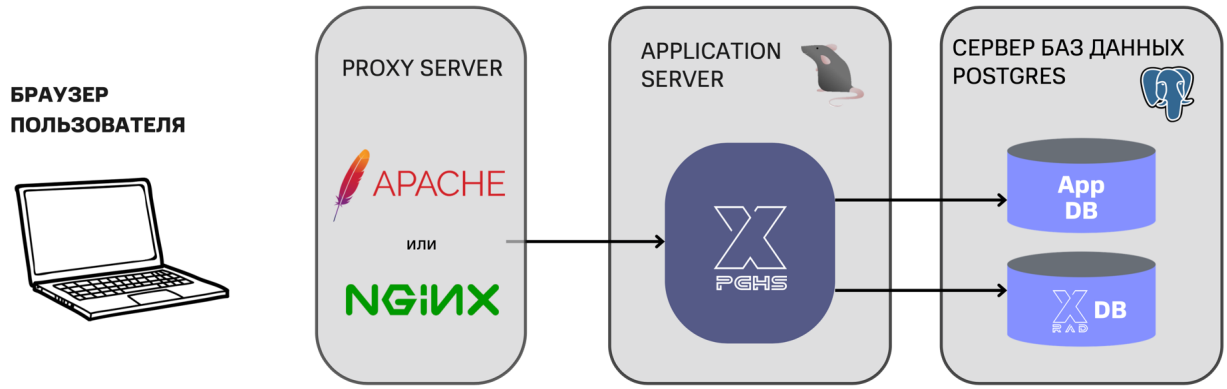
Страницу можно представить как строительный блок приложения. Приложение может содержать всего одну а скорее всего множество таких блоков. Каждая страница может иметь кнопки и поля для ввода (называемые **элементами**), которые группируются посредством размещения в контейнер, который называется **регион**. Страницы могут включать в себя логику обработки данных, называемую **процессами**. Связь страниц друг с другом обеспечивают ссылки или **ветви**. Страницы могут включать в себя проверки пользовательских данных, разнообразные отчеты, графики и прочие элементы. Разработка страницы осуществляется в редакторе страниц.

3.2 Как работает рендеринг страницы и обработка ввода

3.2.1 Как XRAD осуществляет рендеринг страницы и обработку ввода

XRAD осуществляет динамическое построение страниц и обработку данных, основываясь на метаданных в базе данных PostgreSQL. Процесс рендеринга и обработки страницы основан на взаимодействии трех компонентов системы.

- БД XRAD - базы данных с метаданными страницы. В ней хранится информация о компонентах запрашиваемой страницы, процессах которые должны произойти во время обработки страницы.
- PGHS - серверная среда исполнения. Выступает в роли http сервера, который принимает запросы от клиента, формирует страницу на основе данных в БД XRAD, вызывает логику обработки страницы.
- Веб приложение - веб приложение XRAD, которое отвечает за отрисовку компонентов переданных в управляющей структуре http сервером PGHS.



Для просмотра приложения необходимо вызвать страницу приложения XRAD по определенной ссылке, на которой размещается приложение. Когда вы запускаете приложение, XRAD вызывает два ключевых процесса:

- **showPage** - процесс рендеринга страницы, который собирает все атрибуты страницы (включая регионы, кнопки и элементы) в один управляющий файл json. После чего среда исполнения передаёт эту управляющую структуру в веб браузер клиента, на котором происходит отрисовка описанных элементов приложения. За отрисовку компонентов на клиенте отвечает клиентская среда исполнения XRAD. Когда вы запрашиваете страницу, переходя по ссылке XRAD вызывает процесс **showPage**.
- **processPage** - процесс обработки страницы. Отвечает за обработку введенных данных, включая в себя выполнения процессов, проверок данных и переходов на другие страницы. Процесс вызывается после отправки страницы на сервер, путем нажатия кнопки или вызова соответствующего метода jsAPI. После отправки все данные указанные в элементах страницы записываются в сессию и подставляются в соответствующие процессы.

3.2.2 Как работает транзакция

При вызове процессов **showPage** или **processPage** XRAD открывает транзакцию в БД с бизнес данными. Эта транзакция существует на протяжения всего времени выполнения процессов **showPage** или **processPage**. В случае успешного выполнения процессов транзакция завершается с вызовом commit. Если возникает ошибка в процессах обработки данных - транзакция будет прервана вызовом **rollback** на момент начала выполнения процесса.В связи с этим разработчику необходимо учитывать, что данные, которые будут переданы в базу данных посредством выполнения того или иного процесса будут доступны только при успешном завершении транзакции. Доступность данных в момент выполнения той или иной процедуры полностью зависит от того уровня изолированности транзакции настроенного на БД с бизнес данными. Напоминаем что по умолчанию уровень изолированности в PostgreSQL установлен в **Read committed**.

В связи с ограничениями связанными с подтранзакциями в PostgreSQL, вызов **commit** в бизнес процессе приведет к ошибке. Для обеспечения выполнения отдельного процесса вне зависимости от успешности выполнения обработки всей страницы предлагается в настройках процесса указывать вызов commit после его выполнения. Подробнее см. *Процессинг страницы*.

Значения сессии переданные в ходе вызова процессов **showPage** или **processPage**, также будут записаны в БД только после успешного выполнения обработки страницы. Однако их значения доступны в ходе выполнения этих процессов. Для передачи значений элементов ввода в бизнес процессы см. *Процессинг страницы*.

3.3 Сессии и их состояние

3.3.1 Что такое состояние сессии

Состояние сессии - это механизм который позволяет разработчикам хранить и получать значения для пользователя, даже когда он переходит по разным страницам приложения.

Протокол передачи гипертекста (HTTP), протокол, по которому чаще всего доставляются HTML-страницы, является протоколом без сохранения состояния. Веб-браузер подключается к серверу только на время, необходимое для загрузки полной страницы. Каждый запрос страницы обрабатывается сервером как независимое событие, не связанное с какими-либо запросами страницы, которые произошли ранее или могут произойти в будущем. Чтобы получить доступ к значениям формы, введенным на одной странице, на следующей странице, значения должны быть сохранены в состояние сессии. XRAD предоставляет разработчикам возможность получать и устанавливать значения состояния сессии с любой страницы приложения.

3.3.2 Что такое сессия

Сеанс — это логическая конструкция, которая устанавливает постоянство (или поведение с отслеживанием состояния) при просмотре страниц.

Каждой сессии присваивается уникальный идентификатор. XRAD использует этот идентификатор для хранения и извлечения рабочего набора данных приложения до и после каждого просмотра страницы. Поскольку сеансы полностью независимы друг от друга, в базе данных может существовать любое количество сеансов одновременно. Пользователь также может запускать несколько экземпляров приложения одновременно в разных программах браузера.

Сеансы логически и физически отличаются от сеансов базы данных, используемых для обслуживания запросов страниц. Пользователь запускает приложение в одном сеансе XRAD от входа до выхода с типичной продолжительностью, измеряемой в минутах или часах. Каждая страница, запрошенная во время этого сеанса, приводит к тому, что XRAD создает или повторно использует сеанс базы данных для доступа к ресурсам базы данных. Часто такие сеансы базы данных длятся всего доли секунды.

Значение идентификатора сессии хранится в cookie браузера. Время жизни данного cookie определяется настройками приложения и дополнительно контролируются средой исполнения PGHS.

3.4 Управление значениями сессии

Для хранения и получения значений для пользователя используется состояние сессии.

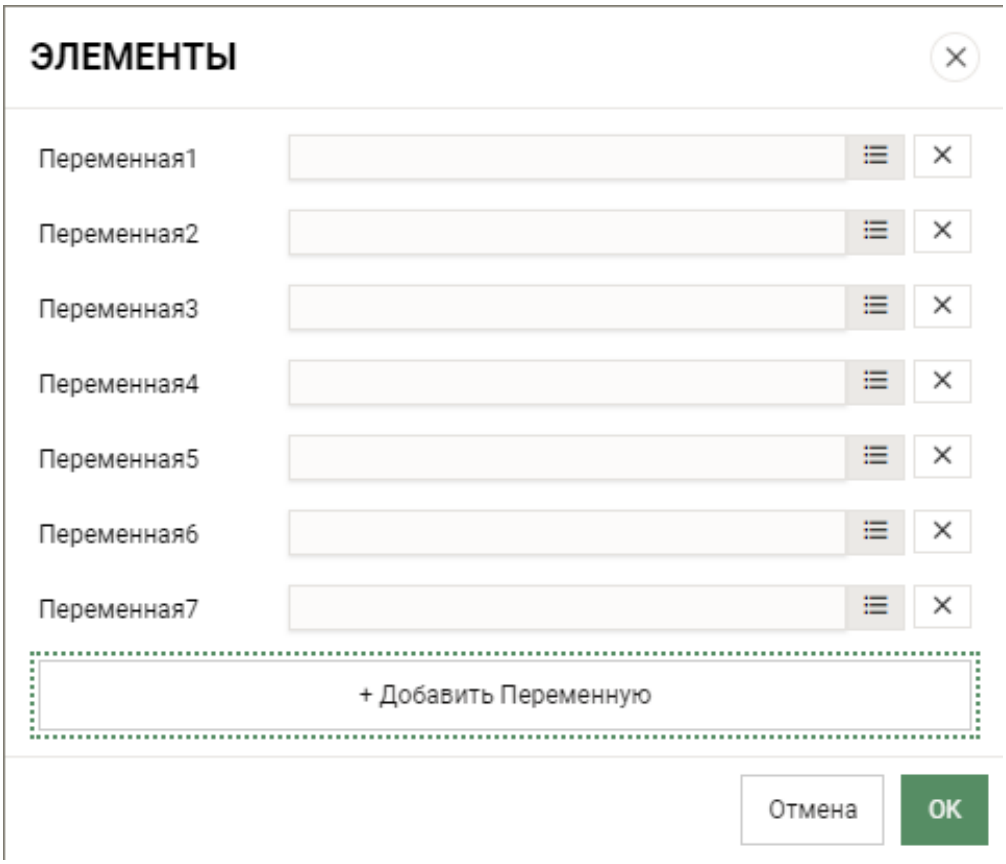
При создании интерактивных веб-приложений, управляемых данными, возможность доступа к значениям состояния сеанса и управления ими имеет решающее значение. В XRAD состояние сеанса автоматически управляется для каждой страницы, и на него легко ссылаться в статическом HTML или логических элементах управления, таких как процессы или проверки.

3.4.1 Как получить значение сессионной переменной

Значения контрольных элементов хранятся в состоянии сеанса в регионах, вычислениях, процессах, проверках и ветвях. Элемент может быть полем, текстовой областью, паролем, списком выбора и прочим элементом ввода.

Передача значений в исполняемый код

Для получения значения элемента и передачи его в исполняемый код, среда разработки предоставляет специальные поля для ввода наименований элементов. Эти поля подставляются в код как позиционные переменные. Код должен быть выражением SQL.



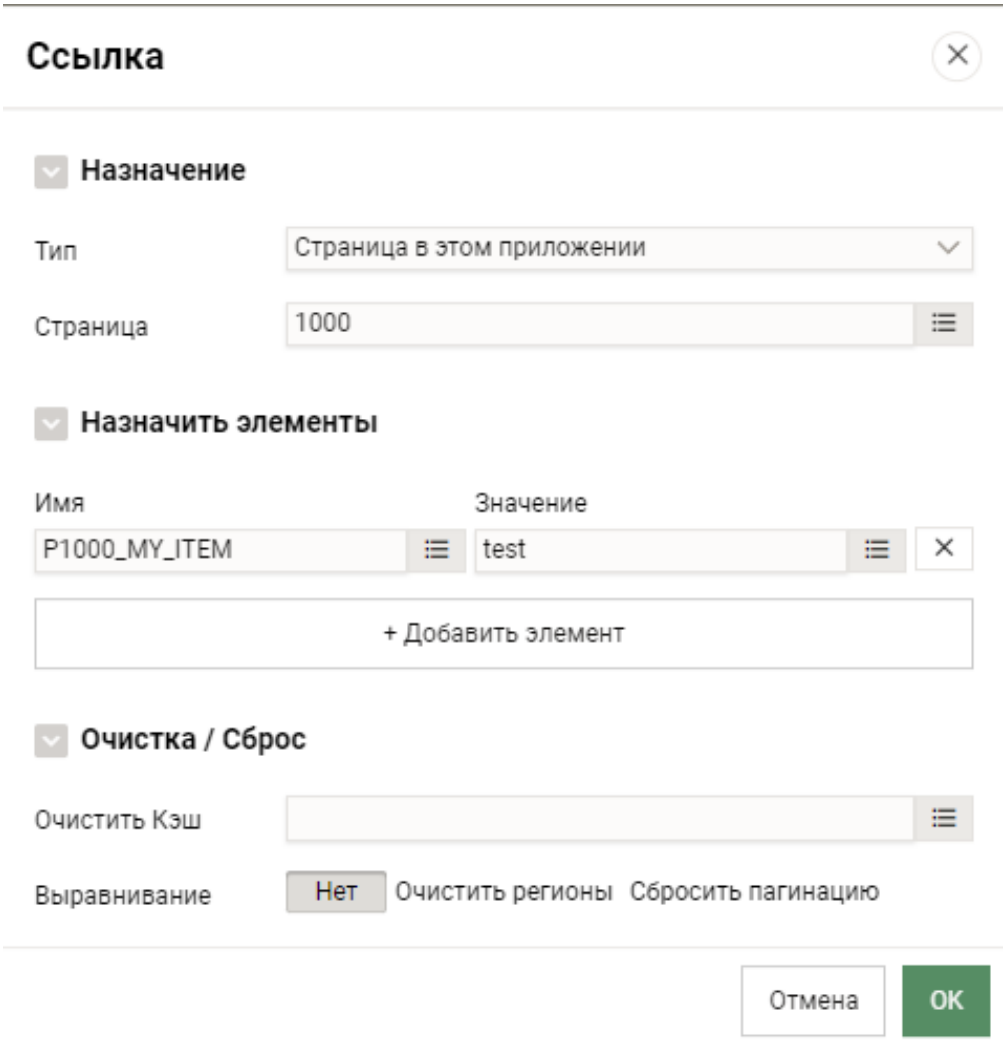
Передача значений для точной замены или в ссылку

Для статического текста или точной замены используйте условное обозначение &ITEM_NAME., за которым следует точка (.).

3.4.2 Как установить значение сессионной переменной

Значение сессионной переменной устанавливается автоматически после успешного выполнения процесса обработки страницы. Введенные пользователем данные в элементы ввода попадают в сессионные переменные, которые носят название указанное разработчиком в среде разработки.

Для передачи значений на форму пре переходе между страницами необходимо указать значение переменных в ссылке.



Ссылка страницу будет выглядеть следующим образом

```
/content/1000?P1000_MY_ITEM=test
```

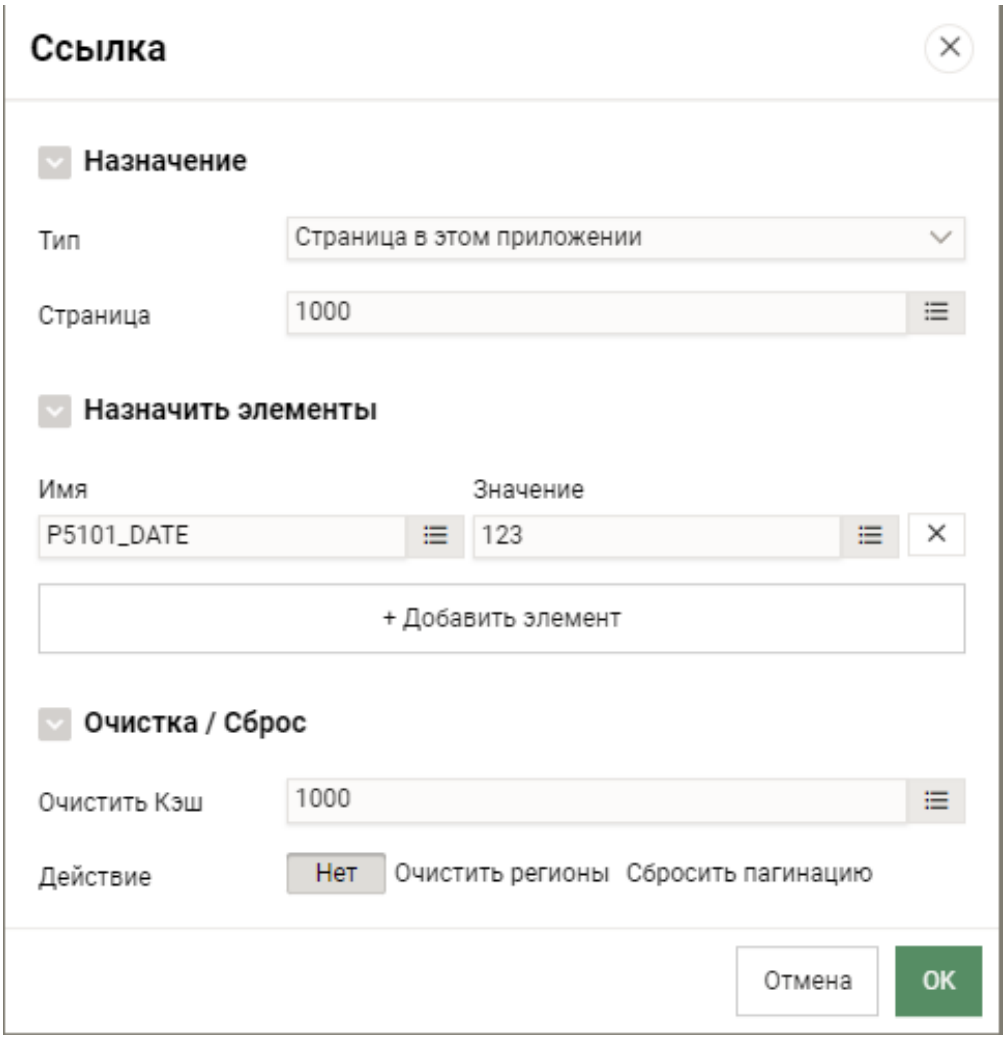
3.4.3 Как очистить состояние сессии

Для очистки состояния сессии используйте управляющее слово clear в ссылке на страницу

```
/content/1000?clear=1000
```

Состояние сессии элементов указанных на странице в параметре clear будет сброшено. Можно указать несколько страниц через запятую.

Данный параметр указывается на форме редактирования ссылки



Дополнительно можно указать действие, которое позволит сбросить пагинацию регионов с типом «Отчет» или очистить в нем критерии поиска.

3.4.4 Глобальные переменные и их значения

Глобальные переменные позволяют хранить значения привязанные к самому приложению а не к определенной странице. Более подробно см. 8. *Глобальные переменные*.

ГЛАВА 4

Управление страницами

Страницы — это основной блоки приложения. Разработчики управляют страницами в приложении, используя **Редактор страниц** (См. *редактор страниц*).

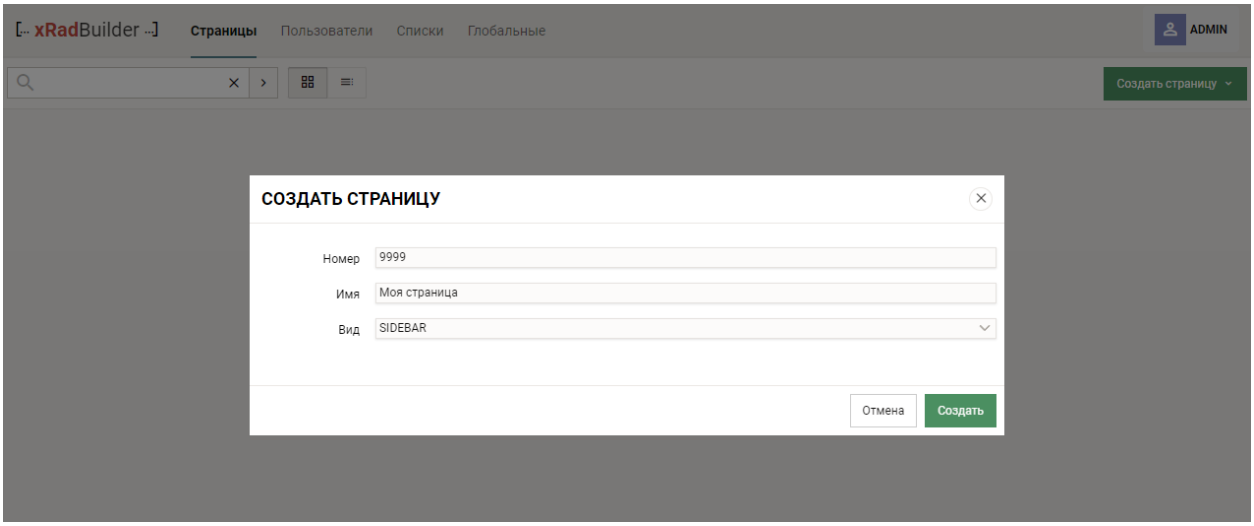
4.1 Добавление новой страницы

Для добавления страницы проделайте следующие шаги:

1. Выберите пункт **Страницы** на верхней панели управления
2. Нажмите **Создать страницу**
3. Выберите пункт выпадающего меню **Создать страницу**
4. Введите атрибуты страницы:

- **Номер** - целочисленное положительное число, которое идентифицирует страницу
- **Имя** - введите наименование страницы
- **Режим** - шаблон представления страницы

5. Нажмите кнопку **Создать**
6. После создания откроется **Редактор страниц**



Режимы страниц:

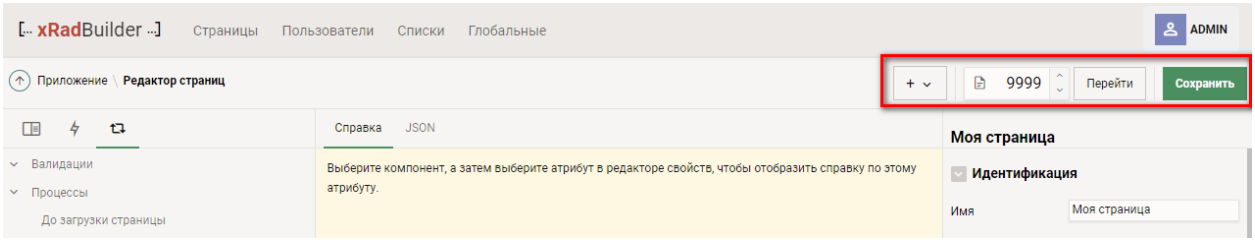
SIDEBAR	Стандартный	Присутствует боковое меню. Присутствуют левая и правая боковые панели (опционально)
MINIMAL	Минималистический	Отсутствуют боковое меню и боковые панели
MODAL	Модальный	
LOGIN	Страница для аутентификации	

4.2 Редактор страниц

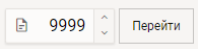
Разработчики могут просматривать и редактировать страницы в **Редакторе страниц**. **Редактор страниц** — это интегрированная среда разработки, которая включает в себя **Панель инструментов**, а также разделена на левую, центральную и правую **области**.

4.2.1 Панель инструментов

Панель инструментов отображается в верхней части **Редактора страниц** и содержит как кнопки, так и пункты меню.

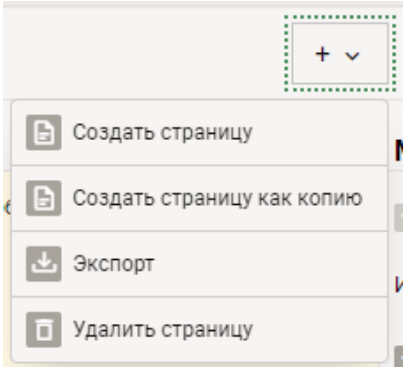


Селектор страниц



Селектор страниц отображает текущую страницу. Для поиска страницы нажмите на иконку страницы, либо введите номер страницы в поле и нажмите **Перейти**. Чтобы перейти на предыдущую страницы нажмите **стрелку вниз**, на следующую страницу - **стрелку вверх**.

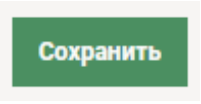
Меню “Действия”



Опции меню «**Действия**» включают:

- **Создать страницу** - открывается диалоговое окно создания новой страницы. См. раздел *Добавление новой страницы*.
- **Создать страницу как копию** - открывается диалоговое окно копирования страницы. См. раздел *Копирование страницы*.
- **Экспорт** - страница выгружается в виде SQL файла.
- **Удалить страницу** - Удаление текущей страницы. См. раздел *Удаление страницы*.

Кнопку “Сохранить ”



Для фиксации любых изменений на странице необходимо нажать кнопку “**Сохранить**”.

4.2.2 Левая область

Левая область содержит три вкладки:

- **Визуализация**
- **Динамические действия**
- **Обработка**

На каждой вкладке отображается список соответствующих типов компонентов, созданных на текущей странице.

Основные функции **левой области** включают в себя:

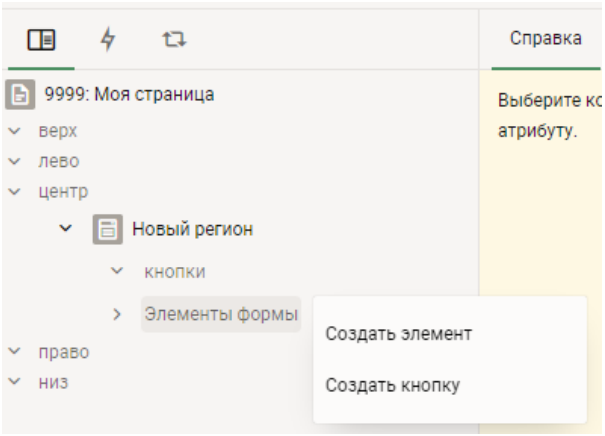
- **Контекстные меню**. Нажмите правой кнопкой мыши компонент или элемент управления, чтобы отобразить контекстное меню.
- **Быстрый доступ к редактору свойств**. Выберите компонент, чтобы отобразить соответствующие свойства в **Редакторе свойств** на **Правой области**.

Визуализация

На вкладке **Визуализация** отображаются компоненты, отображаемые на странице. Компоненты включают **регионы**, **элементы формы** и **кнопки**.

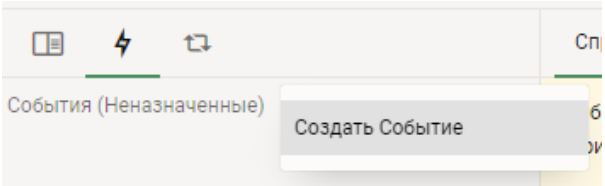
Компоненты могут быть размещены на уровне страницы в определенной позиции страницы или могут быть вложены в регионы в определенной позиции региона.

Чтобы создать новый компонент, нажмите правой кнопкой мыши и выберите параметр в контекстном меню.



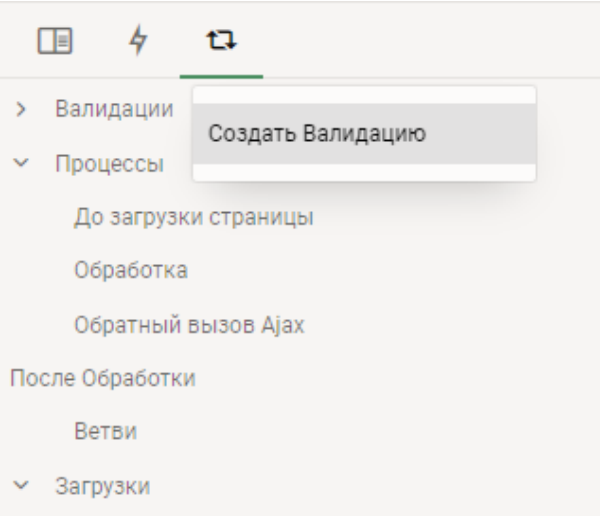
Динамические действия

На вкладке **Динамические действия** отображаются динамические действия, определенные на текущей странице. Создавая динамическое действие, вы можете определить поведение приложения на стороне клиента. Чтобы создать новое динамическое действие, нажмите правой кнопкой мыши и выберите параметр в контекстном меню.



Обработка

На вкладке **Обработка** отображается логика приложения, определенная на странице, а также группируются и упорядочиваются компоненты в зависимости от того, как их обрабатывает XRAD. Чтобы создать новый процесс, нажмите правой кнопкой мыши и выберите параметр в контекстном меню.



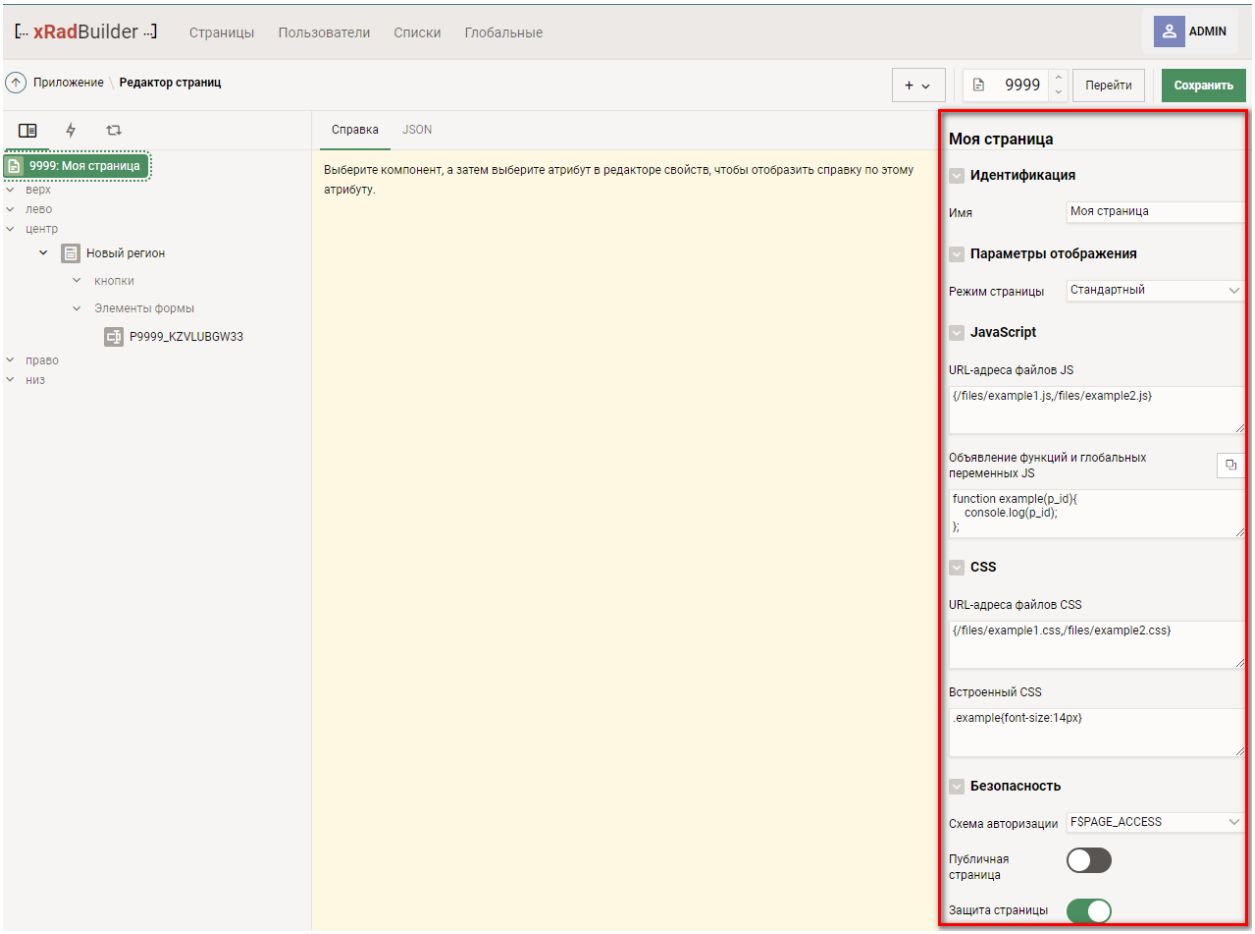
4.2.3 Центральная область

Центральная область содержит следующие вкладки:

- **Справка** -отображает справочную информацию по каждому свойству выбранного компонента.
- **JSON** -отображает техническую информацию о текущей странице.

4.2.4 Правая область

Правая область отображает **Редактор свойств**. **Редактор свойств** используется для редактирования атрибутов выбранного компонента. **Редактор свойств** организует свойства в функциональные группы.



4.3 Свойства страниц

Для страниц определены свойства:

Свойство	Примичание
Имя	Наименование страницы
Режим страницы	Определяет шаблон представления страницы. См раздел *Добавление новой страницы*
URL-адреса файлов JS	Используется для подключения внешних JS файлов
Объявление функций и глобальный переменных JS	Используется для определения JS переменных и функций на уровне страницы
URL-адреса файлов CSS	Используется для подключения внешних CSS файлов
Встроенный CSS	Используется для определения стилей на уровне страницы
Модальная ширина	Определяет ширину модального окна. Доступно только для режима страницы “Модальный”
Схема авторизации	Определяет доступность страницы конкретному пользователю. См. раздел *Схемы авторизации*
Публичная страница	Определяет необходимость аутентификации пользователя. См. раздел *Схемы аутентификации*
Защита страницы	См. раздел *Контрольные суммы*

4.4 Создание модальной страницы

При создании новой страницы, предлагается выбрать режим страницы. Режим страницы определяет, является ли страница обычной страницей или модальной.

Модальная страница представляет собой наложенное окно, расположенное в том же окне браузера. Модальная страница остается активной до тех пор, пока пользователь не завершит работу с ним и не закроет его. Пока модальная страница активна, пользователь не может взаимодействовать с остальной частью страницы, до тех пор пока модальная страница не будет закрыта.

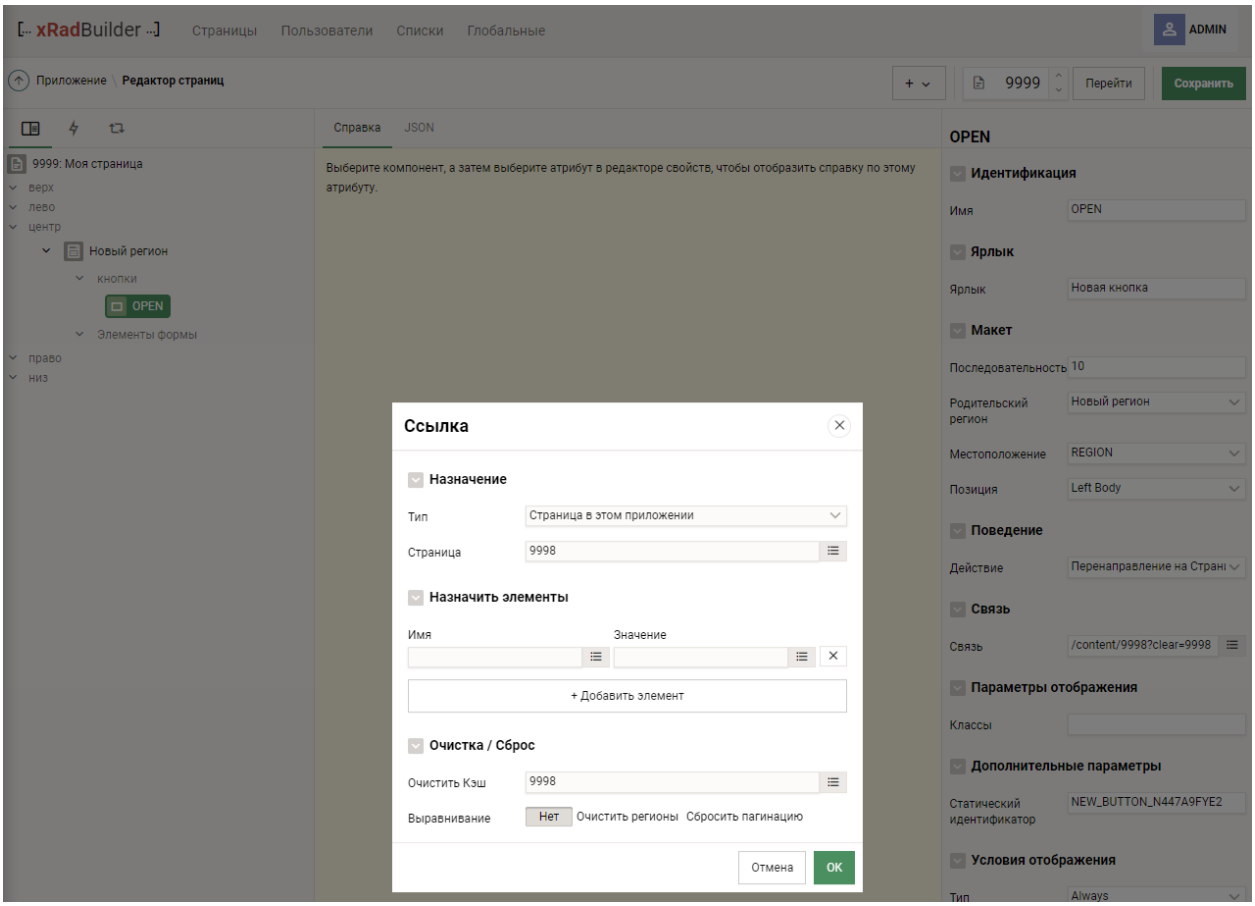
Для того, чтобы сделать страницу модальной необходимо выбрать режим **модальное окно** при добавлении новой страницы (См. раздел *Добавление новой страницы*), либо изменить свойство **режим страницы** в **Редакторе страниц**.

4.4.1 Вызов модальной страницы

Вызвать модальную страницу возможно 2 способами:

- **Ссылка**
- **Javascript**

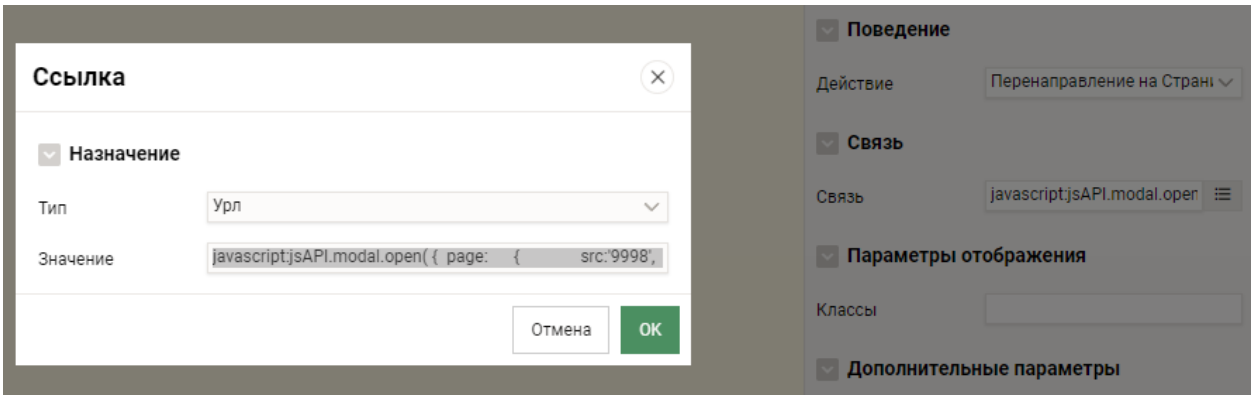
При вызове по ссылке необходимо установить ссылку (вручную, либо воспользовавшись конструктором ссылки) на модальную страницу в свойстве **Связь**. Данный способ не предусматривает обработку результата выполнения модальной страницы.



Вызов модальной страницы с помощью Javascript выполняется следующим образом:

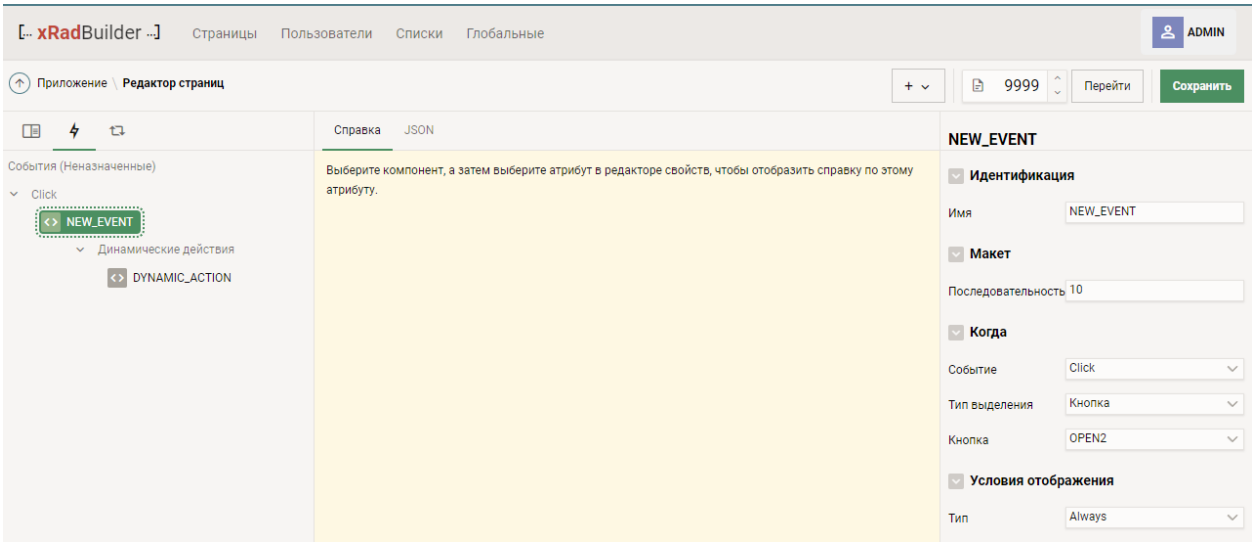
```
jsAPI.modal.open(
{
        page:
        {
                src:'9998',
                query:
                {
                        P9998_ID:jsAPI.getItem('P9999_ID'),
                        clear:'9998'
                }
        },
        title:'Моя модальная страница'
});
```

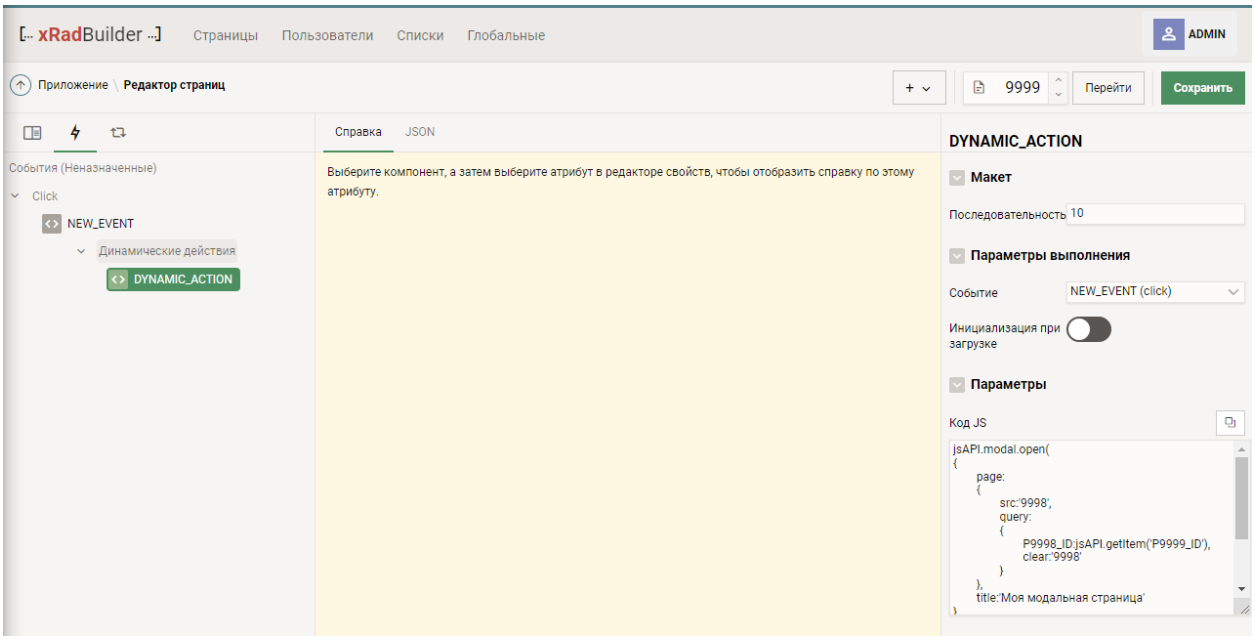
Вызов возможно осуществить как через действие **Перенаправление на страницу** по типу **Ссылка**:



так и через **Динамический действия**:

1. Создать **Событие** типа **Click** на выбранную кнопку
2. Создать **Динамическое действие** к созданному событию, и указать там свойство **Код JS**





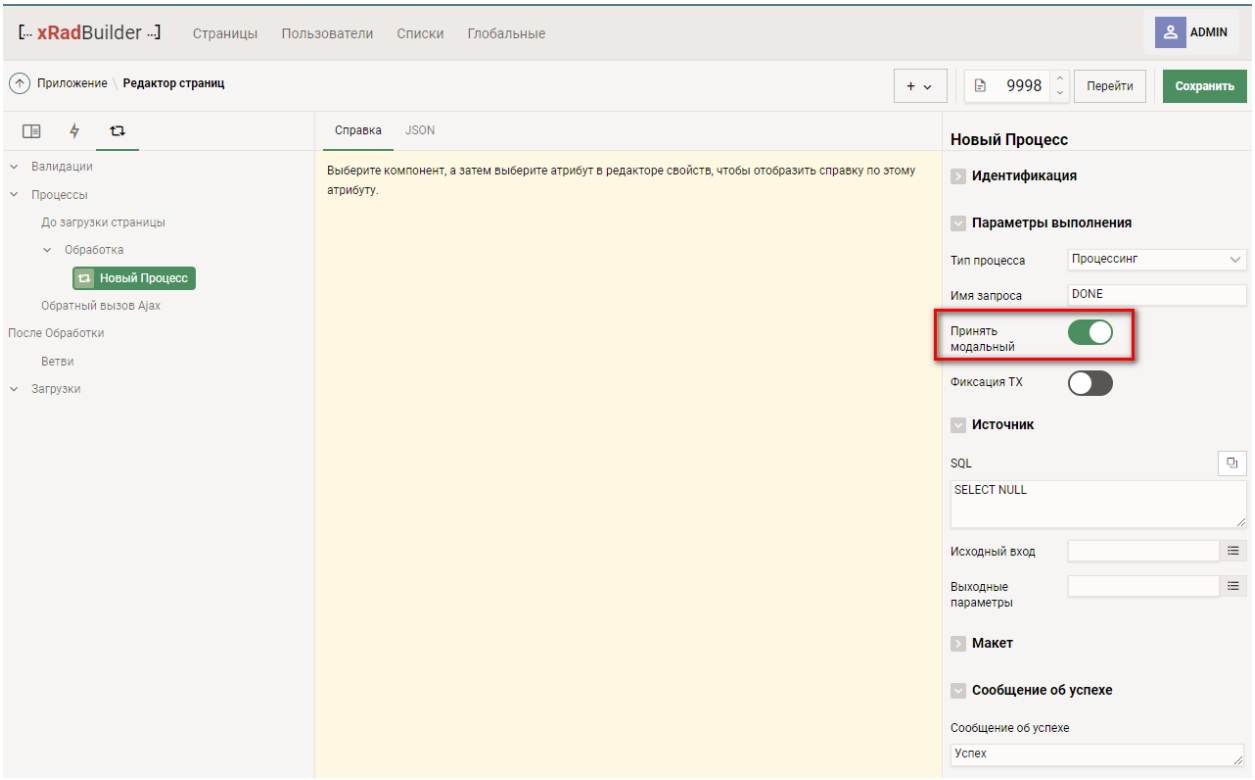
4.4.2 Закрытие модальной страницы (событие “Отмена”)

Закрытие модальной страницы выполняется через Javascript код, выполняющийся на текущей модальной странице:

```
jsAPI.modal.close()
```

4.4.3 Закрытие модальной страницы через процесс

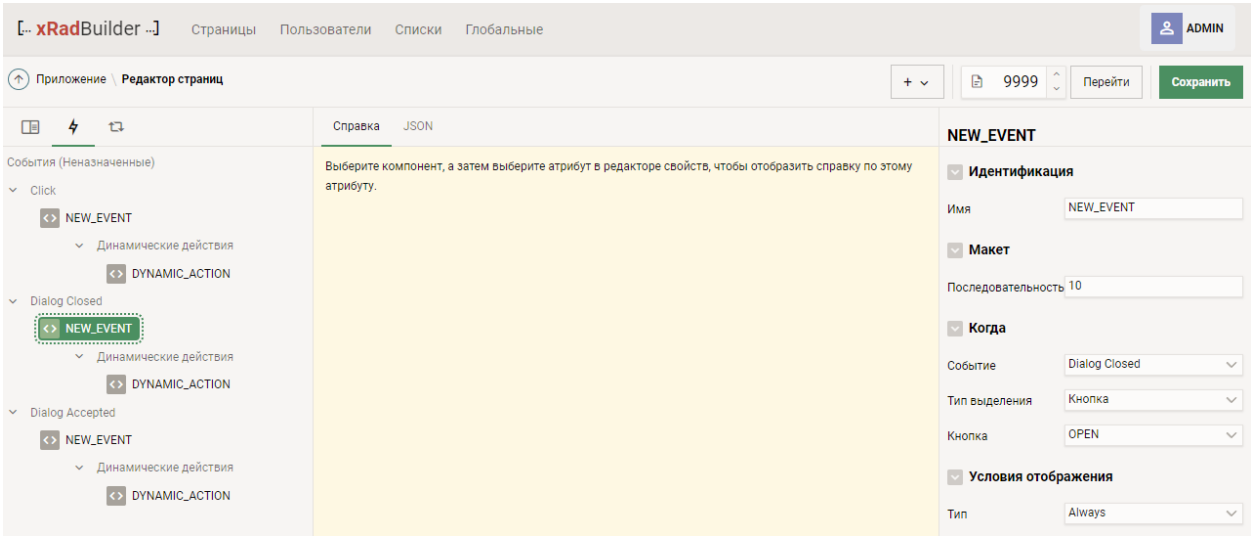
Для того, чтобы закрыть модальную страницу после выполнения процесса необходимо установить соответствующее свойство процесса:

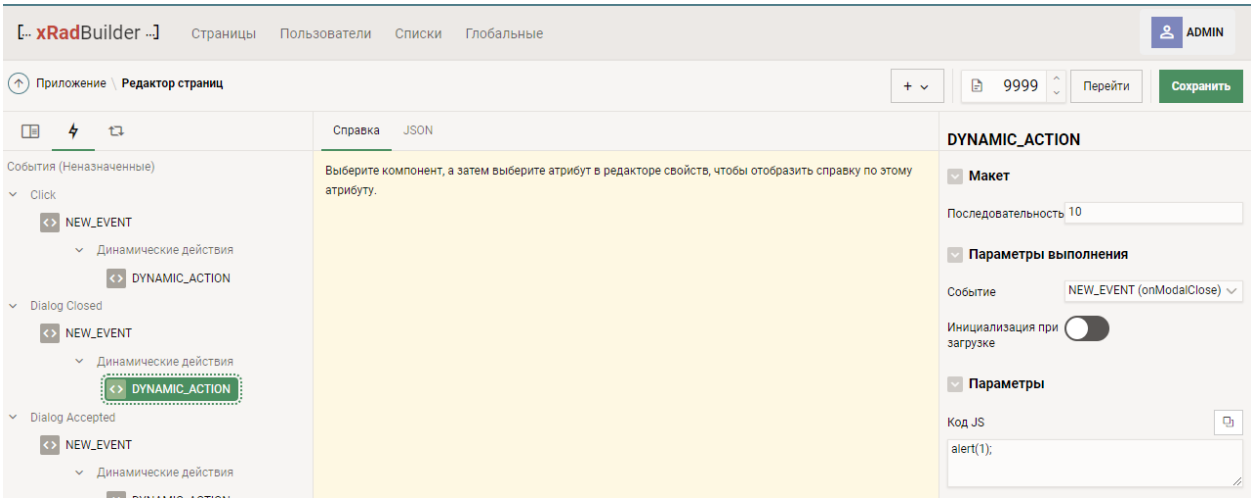


4.4.4 Обновление данных на родительской форме

Для того, чтобы обработать закрытие модальной страницы по необходимому событию необходимо:

1. Создать **Событие** типа **Dialog Closed** (при закрытии через событие “Отмена”) или **Dialog Accept** (при закрытии через процесс) на элемент с которого вызывается модальная страница.
2. Создать **Динамическое действие** к созданному событию, и указать там свойство **Код JS**





Обработку результат выполнения модальной страницы можно также определить при вызове модальной страницы в блоках onClose, onAccept:

```
jsAPI.modal.open(
{
        page:
        {
                src:'9998',
                query:
                {
                        P9998_ID:jsAPI.getItem('P9999_ID'),
                        clear:'9998'
                }
        },
        title:'Моя модальная страница'
},
{
        onAccept:function(e,i)
        {
                jsAPI.setItem('P9999_ID',i.P9998_ID);
        },
        onClose:function(e,i)
        {
                alert(3);
        }
});
```

4.5 Копирование страницы

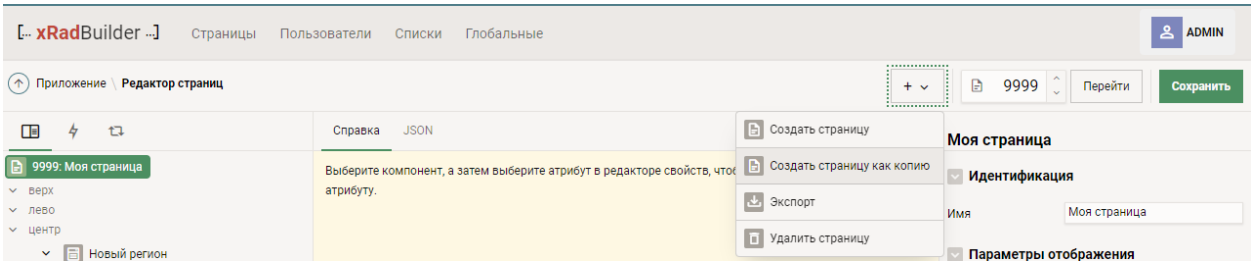
Для ускорения разработки приложения XRAD предоставляет возможность создавать новую страницу как копию с другой страницы.

Чтобы скопировать страницу:

1. Выберите пункт **Страницы** на верхней панели управления
2. Нажмите **Создать страницу**
3. Выберите пункт выпадающего меню **Создать страницу как копию**

4. Укажите номер страницы с которой необходимо создать копию, а также введите атрибуты новой страницы:
   - **Номер** - целочисленное положительное число, которое идентифицирует страницу
   - **Имя** - введите наименование страницы
5. Нажмите кнопку **Создать**
6. После создания откроется **Редактор страниц**

Также копирование страницы доступно в **Редакторе страниц** из меню **Действия**.



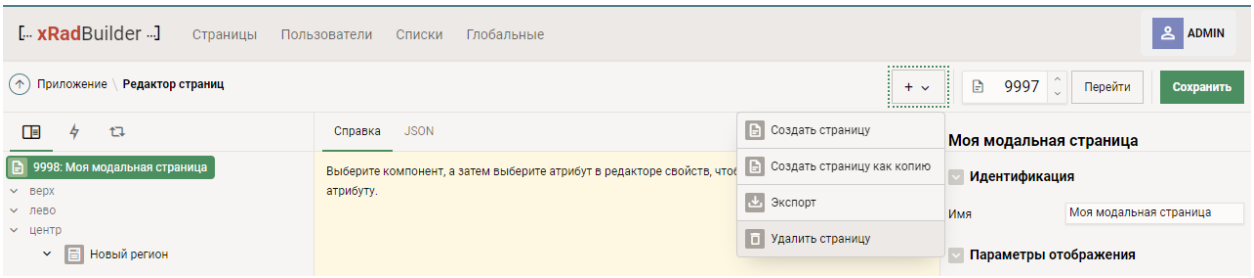
Созданная как копия страница унаследует все компоненты с копируемой страницы.

4.6 Удаление страницы

Удаление страницы осуществляется в **Редакторе страниц** из меню **Действия**.

Чтобы удалить страницу:

1. Нажмите меню **Действия**
2. Выберите пункт **Удалить страницу**
3. Подтвердите действие во всплывающем диалоговом окне
4. После удаления текущей страницы, в **Редакторе страниц** откроется страница со следующем номером



ВНИМАНИЕ! Страница будет удалена без возможности восстановления.

ГЛАВА 5

Разработка форм и отчётов

Визуальные компоненты - основные элементы пользовательского интерфейса. Основными визуальными компонентами являются регионы - контейнеры элементов и средства вывода данных. Основным контейнером элементов является форма (см. *FORM*), основным средством вывода данных - отчёт (см. *REPORT*).

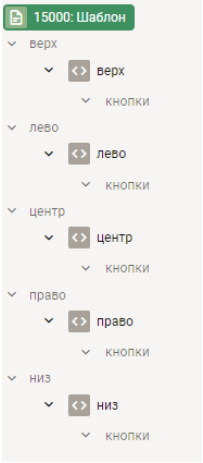
5.1 Регионы и компоненты

Компоненты страницы делятся на два типа: визуальные и невизуальные. К невизуальным относятся процессы, валидации и динамические события, в данном разделе будут рассмотрены визуальные, в числе которых:

1. Верхнее меню и меню навигации слева. Компоненты глобальной страницы, настройки которых определяются на уровне приложения и не меняются от страницы к странице.
2. Регионы - контейнеры элементов. Могут содержать в себе поля для ввода, кнопки и другие регионы, либо непосредственно выводить данные в виде отчётов, графиков и других способов предоставления информации. Являются основными компонентами страницы.
3. Прикрепляющиеся к регионам элементы можно разбить на следующие группы:

- Элементы ввода.
- Элементы выбора.
- Кнопки.

5.1.1 Расположение компонентов на странице



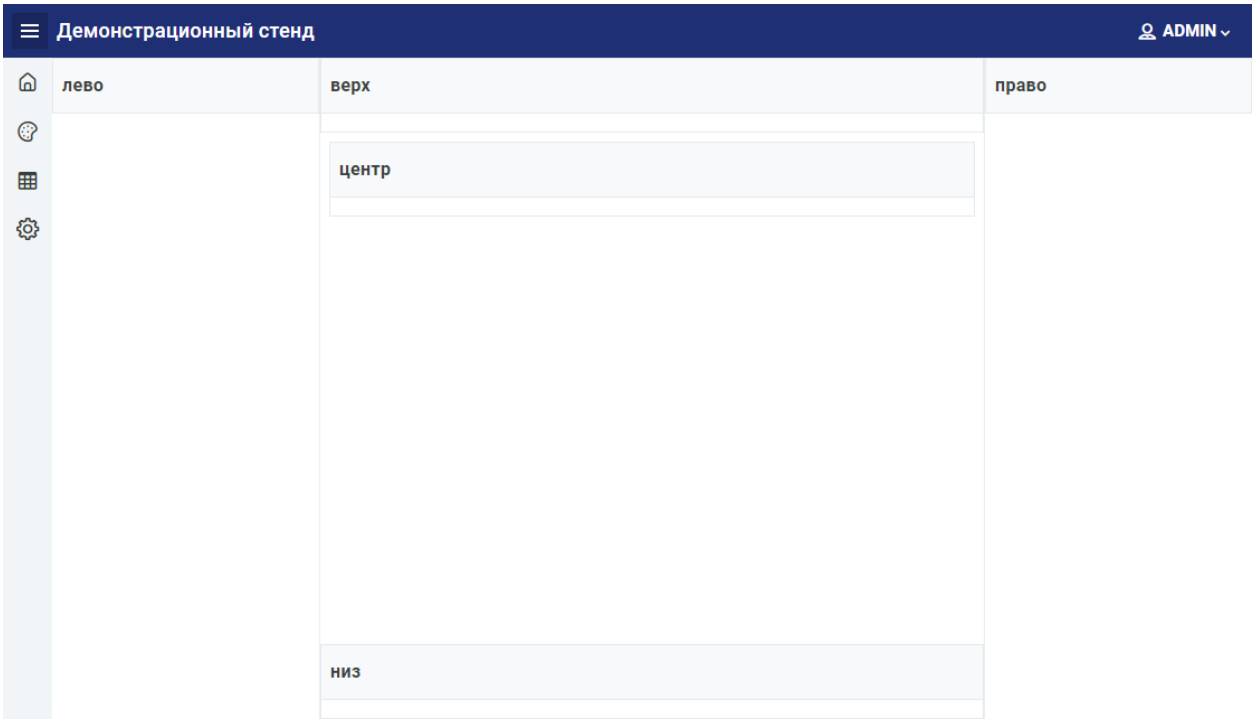
Для региона обязательным параметром является положение на странице.

В XRAD представлен выбор из пяти вариантов расположения:

- верх (top). Здесь удобно расположить регион типа “хлебные крошки”, либо другой вариант заголовка страницы
- лево (left). Хороший вариант расположить навигационную панель, дерево объектов, либо форму фильтров.
- центр (body). Здесь регионы с основными содержимым страницы: отчёты, формы ввода и т.д.
- право (right). Удобный вариант для панели дополнительных инструментов.
- низ (footer). На страницах вида модальное окно здесь располагаются кнопки.

Выбор варианта расположения происходит в момент создания региона на левой панели дерева объектов в конструкторе. В последующем регион можно переместить в другую область страницы, поменяв ему в параметрах свойство “местоположение”(Position).

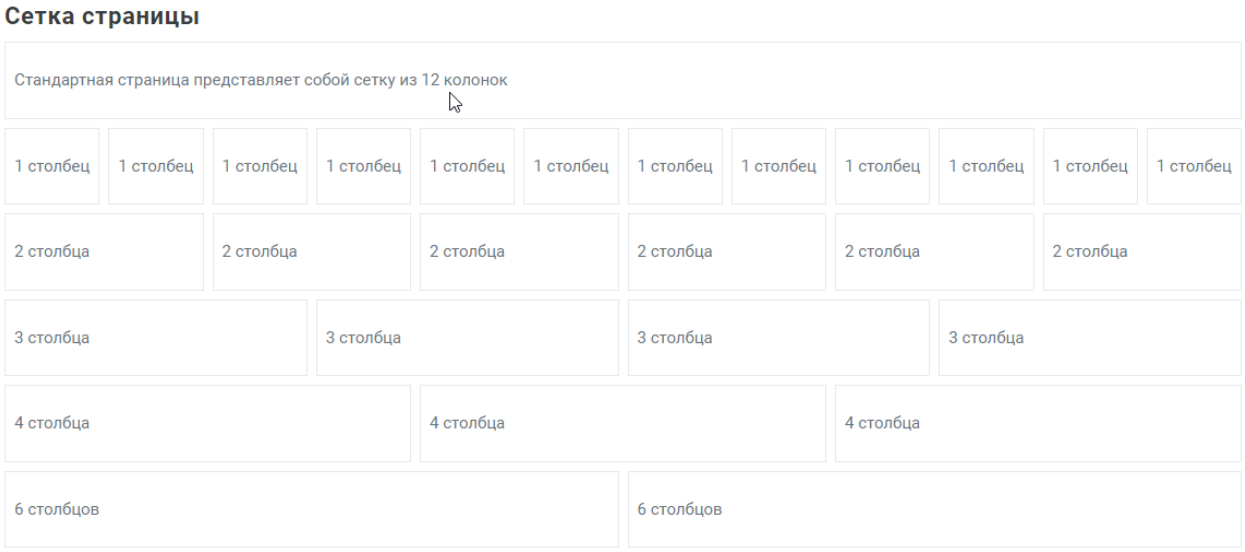
На рисунке продемонстрированы варианты расположения регионов на странице относительно друг друга и глобальных визуальных компонентов:



5.1.2 Расположение компонентов на сетке

Визуальные компоненты страницы - регионы и элементы формы (элементы ввода, выбора и кнопки формы) располагаются в сетке из 12 колонок.

Сетка для регионов верхнего уровня (в качестве родителя указана страница) определяет положение региона внутри области расположения, для вложенных регионов сетка определяет положение внутри родительского региона, как и для элементов формы.

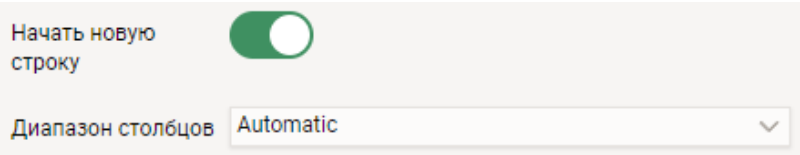


В конструкторе XRAD положение компонента на сетке регулируется следующими параметрами:

- Диапазон столбцов (Column Span). Принимает значение от 1 до 12, либо Automatic. Указывает количество

ячеек сетки, занимаемых компонентом. По умолчанию значение 12 для регионов и Automatic для элементов формы.

- Начать новую строку (Start New Row). Переключатель, принимающий значение true / false. Указывает на то, должен ли компонент занять указанное количество ячеек слева начиная с новой строки, или должен продолжить строку, вслед за предыдущим компонентом по порядку, определяющемуся свойством последовательность.



В одной строке сетки не может быть заполнено более 12 ячеек. В случае, если сумма диапазонов столбцов для компонентов без переноса строки превысит 12, сохранить страницу не получится. В случае, если сумма компонентов одной строки составит менее 12, оставшиеся ячейки составят свободное пространство.

5.1.3 Параметры регионов

В таблице данного раздела перечислены все параметры регионов. В зависимости от вида региона, их набор может различаться.

5.2 Форма (FORM)

Для создания элементов ввода на странице разработчику предлагается воспользоваться регионом с типом компонента “Форма”. Данный тип предлагает функционал по созданию разнообразных элементов ввода пользовательской информации.

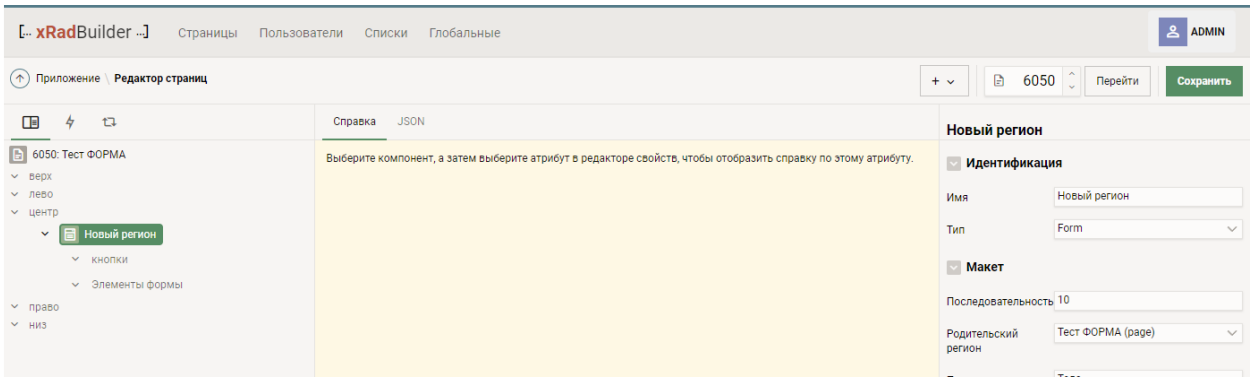
5.2.1 Создание компонента на странице

Для создания формы проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - FORM.



5.2.2 Элементы формы

После создания региона с типом «Form» разработчику становится доступен функционал по наполнению формы элементами ввода. Для создания нового элемента формы необходимо выделить в дереве пункт “Элементы формы”, принадлежащий необходимому региону, нажать правой кнопкой мыши и выбрать пункт выпадающего меню “Создать элемент”.

Основные атрибуты элемента

Рассмотри основные атрибуты, которые присутствуют вне зависимости от типа элемента ввода

Типы элементов

Для выбора типа элемента доступны следующие значения:

5.3 Отчет (REPORT)

Основной регион для вывода данных в виде таблицы. Формируется на основе sql-запроса к базе данных postgres.

5.3.1 Создание компонента на странице

Для создания отчёта проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.

7. Укажите тип региона - REPORT.
8. Введите запрос к БД в после источник данных SQL.
1. В случае наличия переменных подстановки в запросе, укажите входящие параметры.

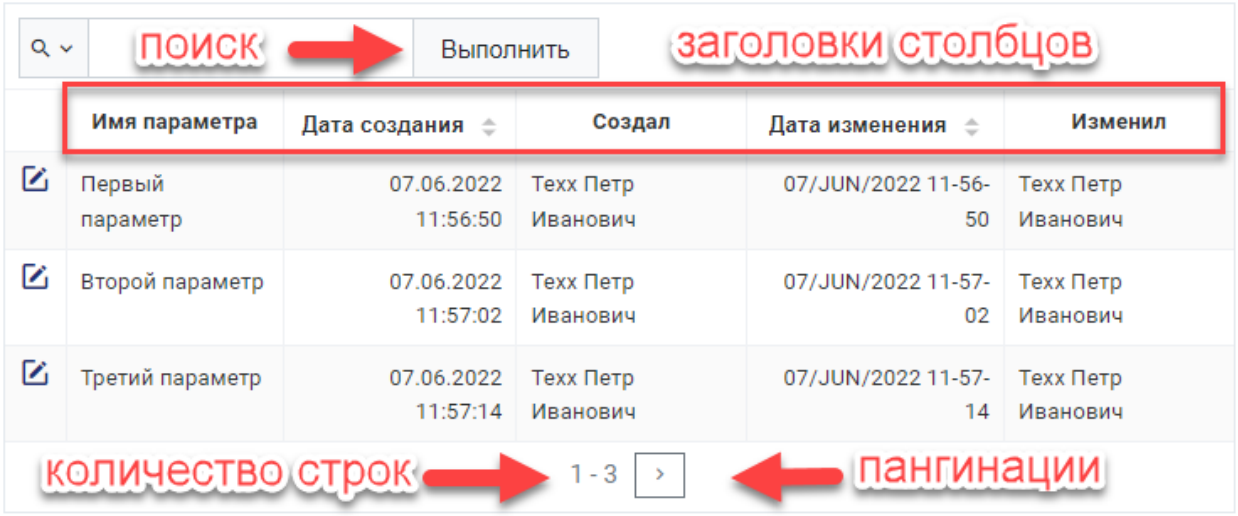
Столбцы отчёта сгенерируются автоматически.

5.3.2 Настройка компонента

Доступны настройки трёх уровней: параметры региона, атрибуты отчёта и параметры столбцов репорта. Рассмотрим подробнее каждую группу.

Атрибуты отчёта

Группа специальных атрибутов, характерных для данного вида регионов.



На рисунке видно как отражаются настройки в атрибутах отчёта, на его отображение на странице.

Параметры региона

Отличительной особенностью региона типа “Отчёт” является необходимость обязательного заполнения поля Источник данных - SQL. Запрос, на основе которого формируется список колонок репорта, и выбираются данные для вывода на форму. более подробно обо всех параметрах региона в п. *Параметры регионов*.

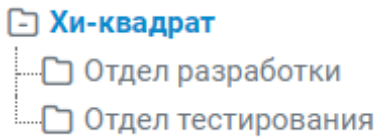
Параметры столбцов

Сгенерированный список столбцов отобразится в дереве объектов сразу после успешной валидации запроса в поле Источник данных SQL. Каждый из столбцов является самостоятельным объектом с индивидуальными параметрами. В таблице перечислен полный набор параметров компонента столбец отчёта.

5.4 Дерево (TREE)

Регион для вывода древовидной структуры данных.

В качестве источника данных используется рекурсивный запрос.



5.4.1 Создание компонента на странице

Для создания дерева проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “лево”.
7. Укажите тип региона - TREE.
8. Введите запрос к БД в после источник данных SQL.

1. В случае наличия переменных подстановки в запросе, укажите входящие параметры.

5.4.2 Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты дерева. Общие настройки параметров региона описаны в п. *Параметры регионов* Здесь рассмотрим собственные атрибуты дерева

5.5 График (CHART)

Регионы типа “График” служат для графического представления численных данных в виде диаграмм.

В системе представлено три типа диаграмм:

- Кольцевая диаграмма
- Столбчатая диаграмма
- Гистограмма

5.5.1 Создание компонента на странице

Для создания диаграммы проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - CHART.
8. Введите запрос к БД в после источник данных SQL.

1. В случае наличия переменных подстановки в запросе, укажите входящие параметры.

9. В дереве объектов щёлкните правой кнопкой мыши по строчке “серии” под атрибутами компонента, и в выпадающем меню нажмите “Создать серию”

5.5.2 Настройка компонента

Доступны настройки трёх уровней: параметры региона, атрибуты диаграммы и параметры серии. Рассмотрим подробнее каждую группу.

Атрибуты диаграммы

Группа специальных атрибутов, характерных для данного вида регионов.

Параметры региона

Отличительной особенностью региона типа “График” является необходимость обязательного заполнения поля Источник данных - SQL. Более подробно обо всех параметрах региона в п. *Параметры регионов*.

Параметры серий

5.6 Хлебные крошки (BREADCRUMBS)

Регион для вывода навигационной цепочки и заголовка страницы. Формируется на основе списка.

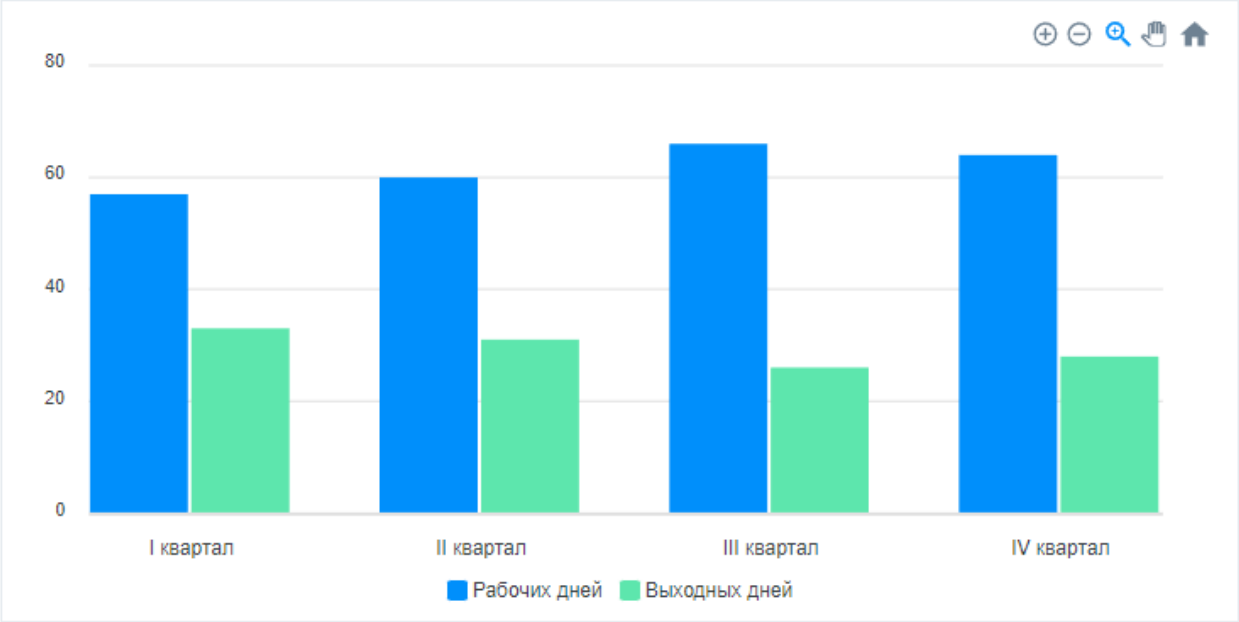


Рис. 1: Рис 1 - Вертикальная диаграмма

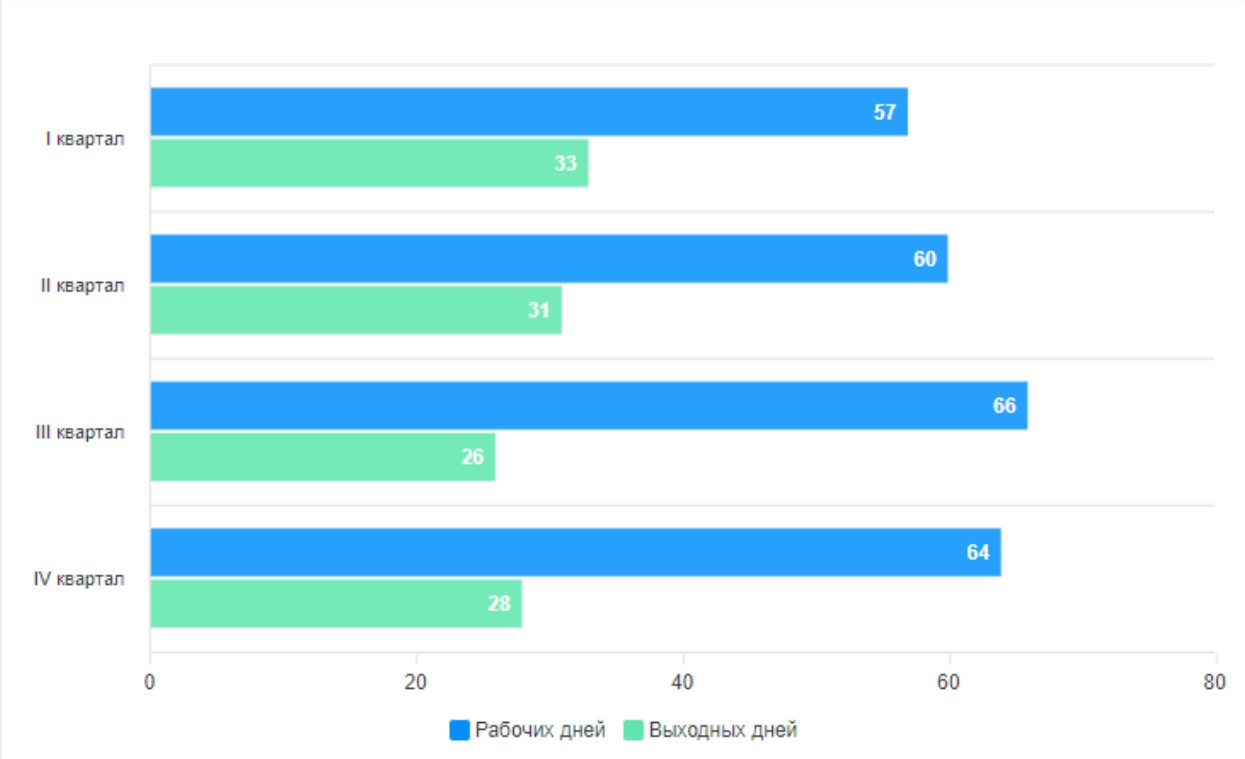


Рис. 2: Рис. 2 - Гистограмма

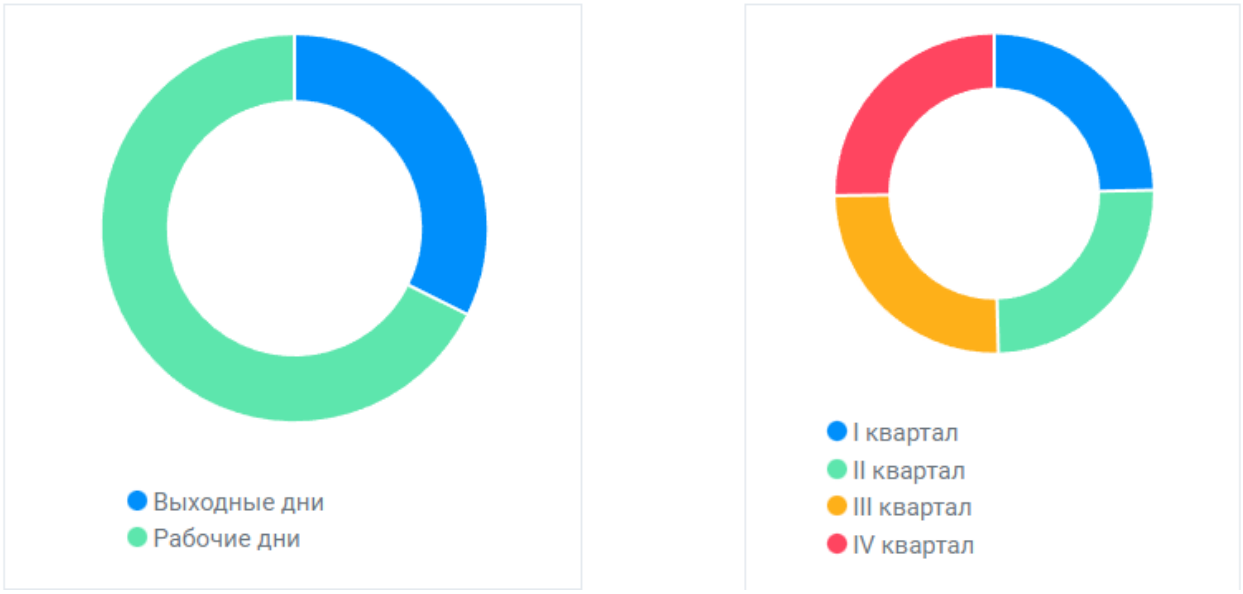


Рис. 3: Рис. 3 - Кольцевая диаграмма

5.6.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “верх”.
7. Укажите тип региона - BREADCRUMB.
8. Укажите список в поле Источник - Список.

Создание элемента списка

Для вывода навигационной цепочки на компоненте необходимо добавить соответствующий странице элемент в список типа Breadcrumb, привязанный к региону (рис.1).

1. В списке типа BREADCRUMB создайте новый элемент.
2. В настройках элемента введите:

- родительский элемент - предыдущий узел навигационной цепочки;
- лейбл - отображаемый заголовок на странице;

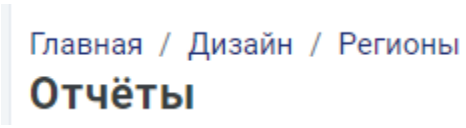


Рис. 4: Рис 1. - навигационная цепочка

- страница - привязка к странице для вывода на ней заголовка
- ссылка - ссылка на страницу из навигационной цепочки.

Подробнее о работе со списками в п. *Управление списками*.

5.6.2 Настройка компонента

Помимо необходимости указания списка-источника, остальные параметры компонента стандартные для региона. Подробнее с полным списком параметров региона можно ознакомиться в п. *Параметры регионов*.

5.7 Карточки (CARDS)

Регион для графического представления списка в виде карточек. Карточки имеют собственный цвет, картинку, а также ссылку на страницу, или скрипт.

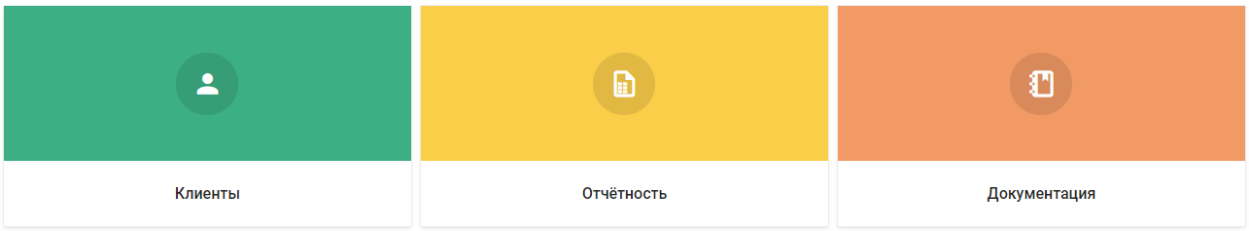


Рис. 5: Рис. 1 - Карточки

5.7.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.

7. Укажите тип региона - CARDS.
8. Укажите список в поле Источник - Список.

Создание элемента списка

Для вывода карточки на компоненте необходимо добавить элемент в список, привязанный к региону.

1. В списке создайте новый элемент.
2. В настройках элемента введите:

- последовательность - численный порядок элемента, определяющий его позицию;
- CSS Класс иконки - иконка для карточки;
- лейбл - отображаемый заголовок;
- ссылка - ссылка на страницу или скрипт при клике на карточку;
- атрибут 2 - RGB -цвет карточки в формате #ffffff

Подробнее о работе со списками в п. *Управление списками*.

5.7.2 Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты. Рассмотрим подробнее каждую группу.

Атрибуты

Регион типа CARDS имеет два специальных атрибута

Атрибут	Тип	Описание
Тема	Список	Шаблон, определяющий отображение карточек. На выбор три варианта: • Блочная. (Рис.1) • Функциональная. (Рис.2) • Базовая. (Рис.3)
Колонки	Список	Количество карточек в ряд. В случае если элементов в списке больше, чем указанное здесь число, карточки расположатся в несколько рядов. Принимает значение от одного до двенадцати.

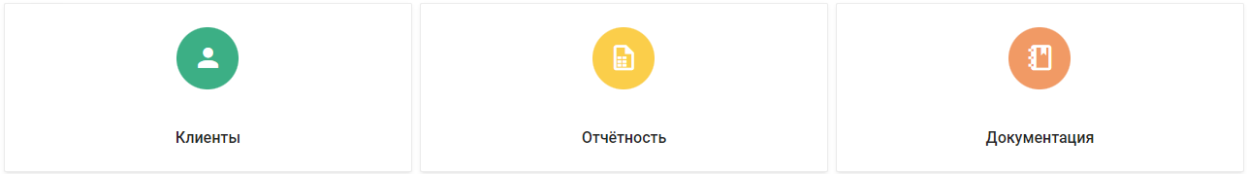


Рис. 6: Рис. 2 - Функциональная тема

Рис. 7: Рис. 3 - Базовая тема

Параметры региона

Отличительной особенностью региона типа “Карточки” является необходимость обязательного заполнения поля Источник данных - Список. Более подробно обо всех параметрах региона в п. *Параметры регионов*.

5.8 HTML

Регион для вывода HTML. Используется как для вывода текста, так и для произвольных элементов, вроде картинок, видео и т.п.

5.8.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

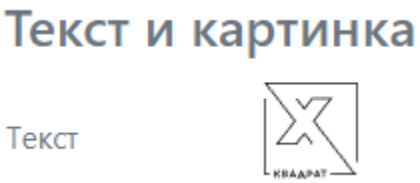


Рис. 8: Рис. 1 - Регион на странице

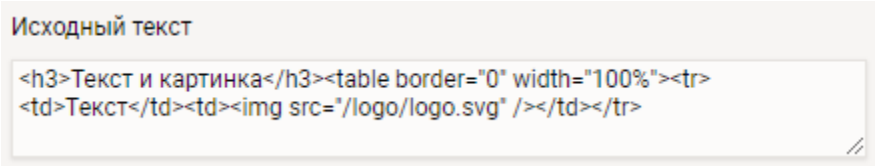


Рис. 9: Рис. 2 - Исходный текст

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новые регион в позиции “центр”.
7. Укажите тип региона - HTML.
8. Введите текст кода HTML в поле исходный текст, или запрос к БД в после источник данных SQL.

5.8.2 Настройка компонента

Помимо необходимости выбора источника, остальные параметры компонента стандартные для региона. Подробнее с полным списком параметров региона можно ознакомиться в п. *Параметры регионов*.

5.9 Контейнер для кнопок (BUTTONS)

Регион для размещения кнопок.

5.9.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите MODAL.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “низ”.
7. Укажите тип региона - BUTTONS.

5.9.2 Настройка компонента

Параметры региона стандартны, подробнее с полным списком параметров можно ознакомиться в п. *Параметры регионов*.

5.9.3 Кнопки

Кнопки могут размещаться в теле любых контейнеров, на форме, а также в заголовках любых регионов и служат для вызова процесса, динамического действия, или перехода на другую страницу по клику на них.

Параметры кнопок

Отображение кнопок

На рисунках ниже представлены варианты отображения кнопок в зависимости от выбранных местоположения и позиции, а также основных классов.

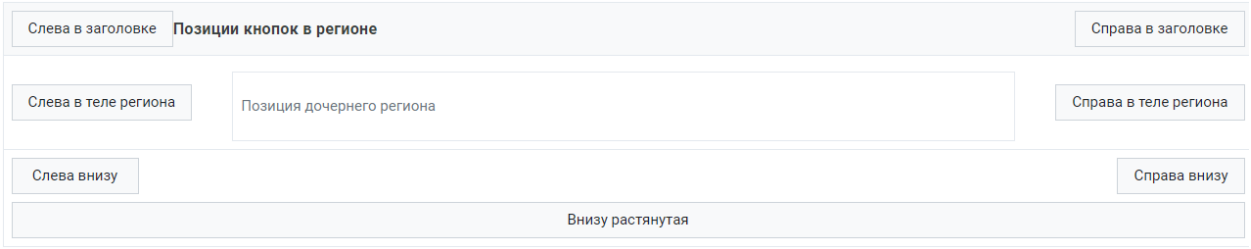


Рис. 10: Рис. 1 - Выбор позиции для местоположения REGION

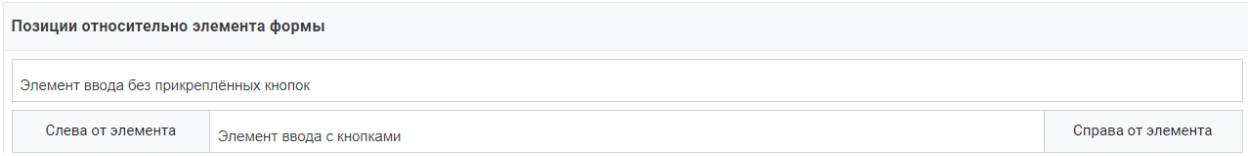


Рис. 11: Рис. 2 - Выбор позиции для местоположения ITEM

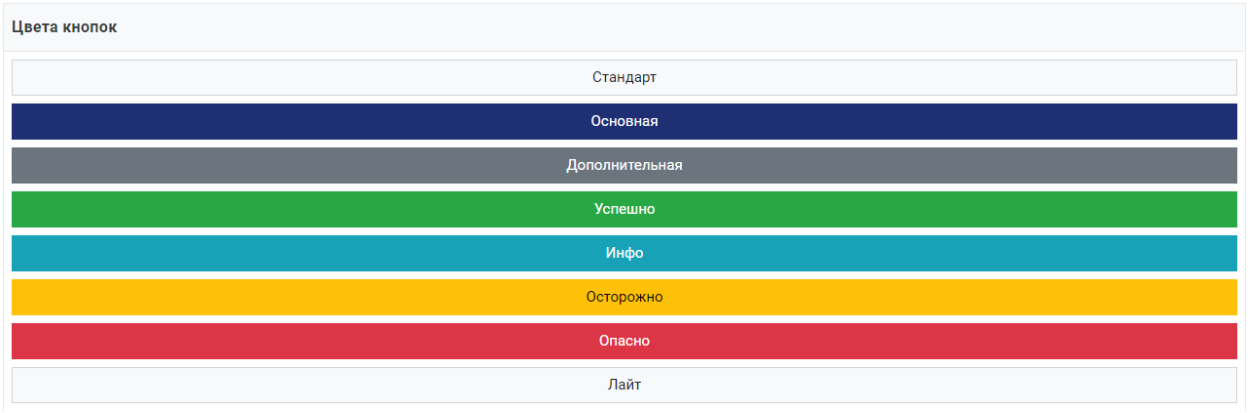


Рис. 12: Рис. 3 - Выбор цвета кнопки, местоположение FORM.

5.10 Навигационная панель (PAGE NAVIGATION)

Регион для размещения навигационной панели на странице на основе списка. Элементы списка отобразятся в виде области с иконкой, заголовком-ссылкой и описанием.

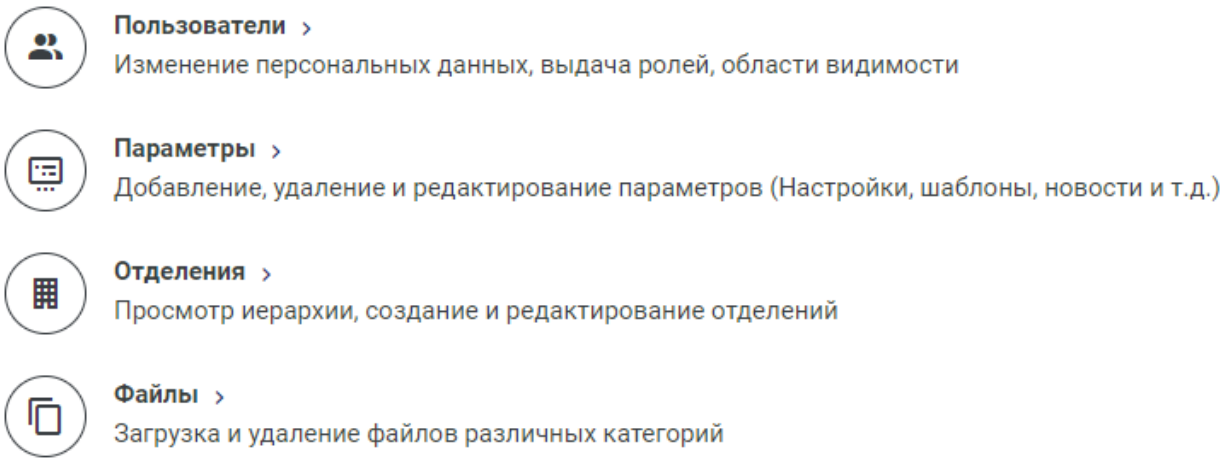


Рис. 13: Рис. 1 - Навигационная панель

5.10.1 Создание компонента на странице

Для создания навигационной панели проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - Page Navigation.
8. Укажите список в поле Источник - Список.

Создание элемента списка

Для вывода ссылки на компоненте необходимо добавить элемент в список, привязанный к региону.

1. В списке создайте новый элемент.
2. В настройках элемента введите:

- последовательность - численный порядок элемента, определяющий его позицию;
- CSS Класс иконки - иконка для карточки;
- лейбл - отображаемый заголовок;
- ссылка - ссылка на страницу или скрипт при клике на заголовок в навигационной панели;
- атрибут 1 - Описание элемента под заголовком.

Подробнее о работе со списками в п. *Управление списками*.

5.10.2 Настройка компонента

Помимо необходимости указания списка-источника, остальные параметры компонента стандартные для региона. Подробнее с полным списком параметров можно ознакомиться в п. *Параметры регионов*.

5.11 Вкладки (TABS)

Контейнер для группировки дочерних регионов на переключаемых вкладках.

5.11.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - TABS.
8. Создайте один, или несколько дочерних регионов, заполните им поле заголовок.

5.11.2 Настройка компонента

Параметры компонента стандартные для региона. Подробнее с полным списком параметров можно ознакомиться в п. *Параметры регионов*.

На вкладках отобразятся заголовки дочерних регионов, даже если для самих регионов заголовок отключен.

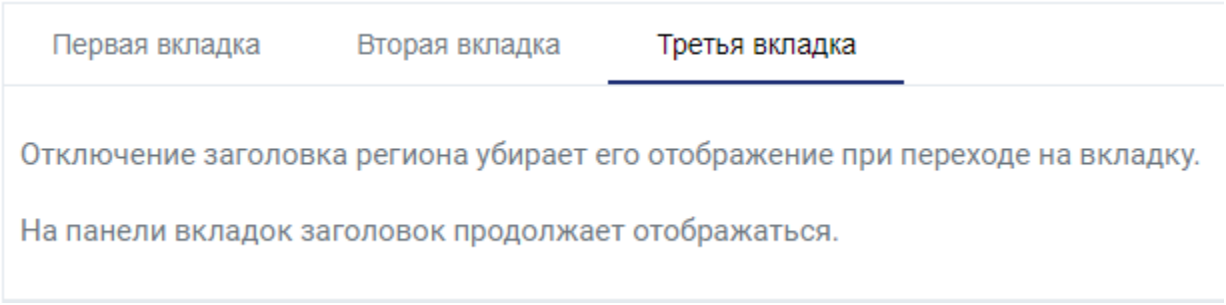


Рис. 14: Рис. 1 - Вкладки

5.12 Календарь (CALENDAR)

Регион для отображения данных с меткой времени в виде интерактивного календаря.

5.12.1 Создание компонента на странице

Для создания календаря проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - CALENDAR.
8. Введите запрос к БД в после источник данных SQL.

1. В случае наличия переменных подстановки в запросе, укажите входящие параметры.

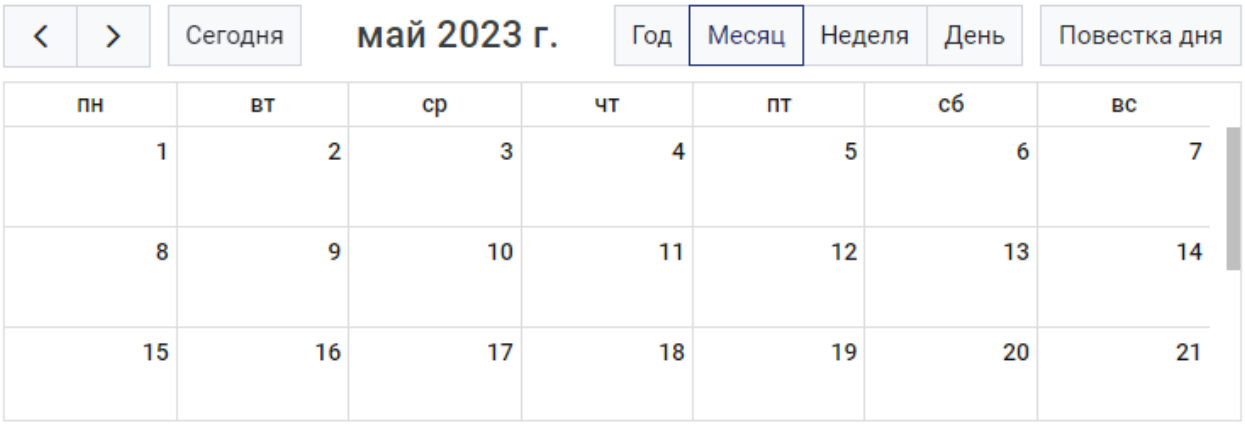


Рис. 15: Рис. 1 - Календарь

5.12.2 Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты календаря. Общие настройки параметров региона описаны в п. *Параметры регионов*

В данном разделе рассмотрим собственные атрибуты календаря.

5.13 Чат (CHAT)

Регион для вывода чатов.

5.13.1 Создание компонента на странице

Для создания чата проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - CHAT.
8. Введите запрос к БД в после источник данных SQL.

1. В случае наличия переменных подстановки в запросе, укажите входящие параметры.

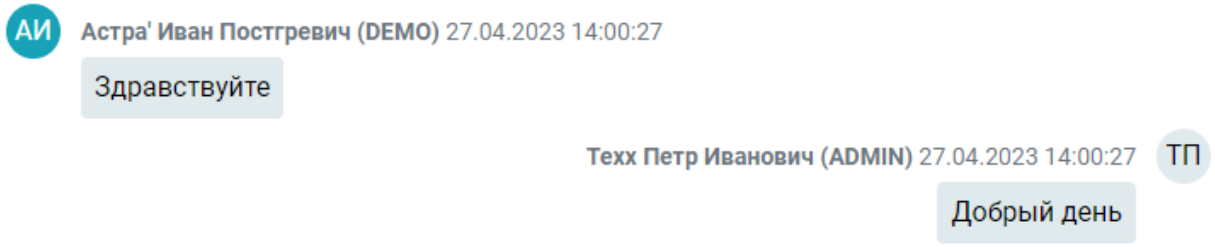


Рис. 16: Рис. 1 - Чат

5.13.2 Настройка компонента

Помимо необходимости выбора источника, остальные параметры компонента стандартные для региона. Подробнее с полным списком параметров региона можно ознакомиться в п. *Параметры регионов*.

5.14 Обертка (WRAPPER)

Основной контейнер для регионов. Используется для группировки и зонирования дочерних регионов.

5.14.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “центр”.
7. Укажите тип региона - WRAPPER.

5.14.2 Настройка компонента

Параметры компонента стандартные для региона. Подробнее с полным списком параметров можно ознакомиться в п. *Параметры регионов*.

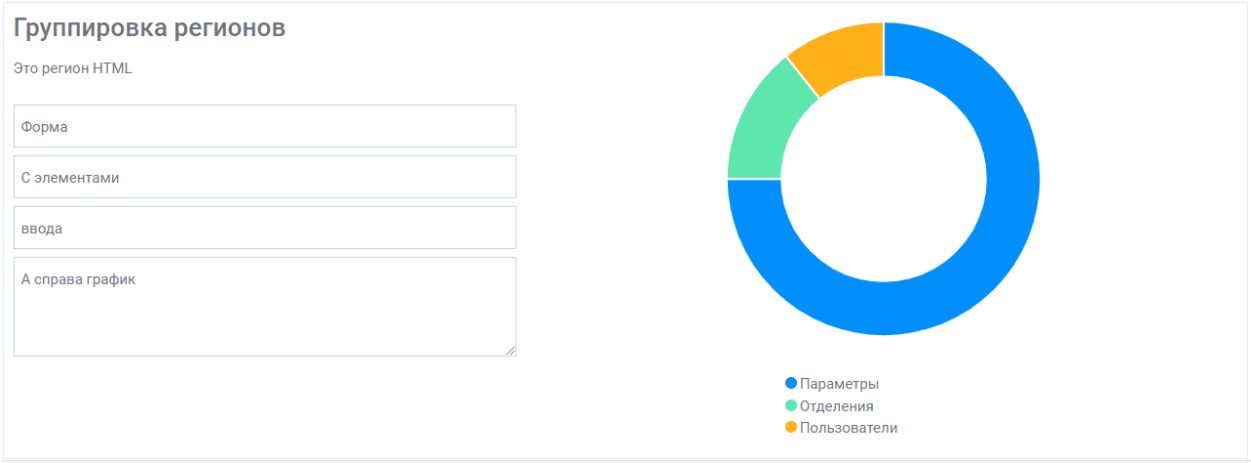


Рис. 17: Рис. 1 - Группировка разнотипных регионов

5.15 Визард (WIZARD)

Регион для графического представления последовательности действий на основе списка.

5.15.1 Создание компонента на странице

Для создания региона проделайте следующие шаги:

1. Войдите в среду разработки XRAD
2. Выберите пункт “Страницы” на верхней панели управления
3. Нажмите “Создать страницу”
4. Введите атрибуты страницы:

1. Номер страницы - целочисленное положительное число, которое идентифицирует страницу
2. Название - введите название страницы
3. Вид - шаблон представления страницы. Для примера укажите SIDEBAR.

5. После создания откроется редактор страниц.
6. Создайте новый регион в позиции “верх”.
7. Укажите тип региона - WIZARD.
8. Укажите список в поле Источник - Список.
9. Повторите действие для нескольких страниц, которым будут сопоставлены элементы из того же списка.

Создание элемента списка

Для прикрепления страницы необходимо добавить элемент в список, привязанный к региону.

1. В списке создайте новый элемент.
2. В настройках элемента введите:

- последовательность - численный порядок элемента, определяющий его относительную позицию на визарде;
- лейбл - отображаемый заголовок;
- ссылка - ссылка на страницу;
- условие для активного состояния - укажите здесь номер страницы, чтобы ползунок визарда передвинулся на соответствующую позицию при нахождении на ней.

Подробнее о работе со списками в п. *Управление списками*.

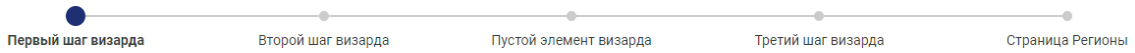


Рис. 18: Рис. 1 - Визард

5.15.2 Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты визарда. Общие настройки параметров региона описаны в п. *Параметры регионов*

В данном разделе рассмотрим собственные атрибуты визарда.

ГЛАВА 6

Валидация и процессинг страницы

Для формирования полноценного приложения требуется взаимодействие с пользователем. Всё, что делает пользователь, необходимо складывать в базу данных. Однако пользователь может испортить приложение введя не корректные данные – возложите сохранение и проверку данных на XRAD.

6.1 Валидация страницы

В любом приложении имеются формы взаимодействия с пользователями, но некорректные данные, введенные пользователем, могут навредить приложению. Поэтому перед отправкой данных используют валидацию данных.

6.1.1 Что такое валидация

Валидация – это проверка корректности введенных пользователем данных в поля ввода формы.

Для системы XRAD – это различные функции и процедуры, которые выполняются перед тем как данные будут отправлены на сервер для обработки. На случай возникновения исключения у разработчика имеется возможность задать текст сообщения, который будет отображен пользователю.

Разработчику доступно 11 типов валидаторов:

1. *Exists (SQL query returns at least one row)* – позволяет выполнить проверочный SQL-запрос. Проверка считается пройденной успешно, если запрос вернул хотя бы 1 строку;
2. *Value of Item / Column in Expression 1 Is NOT NULL* – встроенная проверка на заполненность поля или ячейки таблицы. Проверка считается пройденной успешно, если поле ввода или ячейка таблицы содержит какое-либо значение;
3. *Value of Item / Column in Expression 1 != Zero* – встроенная проверка на значение отличное от 0 (нуль) поля ввода или столбца таблицы. Проверка считается пройденной успешно, если объект проверки будет содержать значение отличное от 0 (нуль);
4. *Value of Item / Column in Expression 1 Is NULL* – встроенная проверка на значение *NULL* (пустое значение) поля ввода или столбца таблицы. Проверка считается пройденной успешно, если объект проверки содержит *NULL* (пустое значение);

5. *Value of Item / Column in Expression 1 Is NULL or Zero* – встроенная проверка на значение *NULL* (пустое значение) или 0 (нуль) поля ввода или столбца таблицы. Проверка считается пройденной успешно, если объект проверки содержит *NULL* (пустое значение) или равно 0 (нуль);
6. *Value of Item / Column in Expression 1 = Zero* – встроенная проверка на значение 0 (нуль) поля ввода или столбца таблицы. Проверка считается пройденной успешно, если объект проверки будет содержать значение 0 (нуль);
7. *Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero* – встроенная проверка на заполненность и значение отличное от 0 (нуль) поля ввода или столбца таблицы. Проверка считается пройденной успешно, если объект проверки не содержит *NULL* (пустое значение) и не равно 0 (нуль);
8. *NOT Exists (SQL query returns no rows)* – позволяет выполнить проверочный SQL-запрос. Проверка считается пройденной успешно, если запрос не вернул ни одной строки;
9. *SQL Expression* – позволяет выполнить SQL-запрос возвращающий *TRUE* или *FALSE*. В поле ввода запроса указывается только тело запроса, без ключевых слов *SELECT* и/или *FROM*. Если необходимо выполнить какой-то подзапрос, то его необходимо обернуть в "(" ")". Проверка считается пройденной успешно, если запрос вернул *TRUE*;
10. *Function returning Error Text* – позволяет выполнить SQL-запрос возвращающий текст ошибки. В поле запроса необходимо указать полноценный запрос возвращающий строку. Проверка считается пройденной успешно, если запрос вернул *NULL*. В противном случае будет отображен возвращенный запросом текст ошибки;
11. *Regular Expression* – позволяет проверить значение поля формы по регулярному выражению. В поле *Item* указывается проверяемое поле, а в поле *Value* регулярное выражение. Проверка считается пройденной успешно, если значение поля *Item* будет соответствовать регулярному выражению.

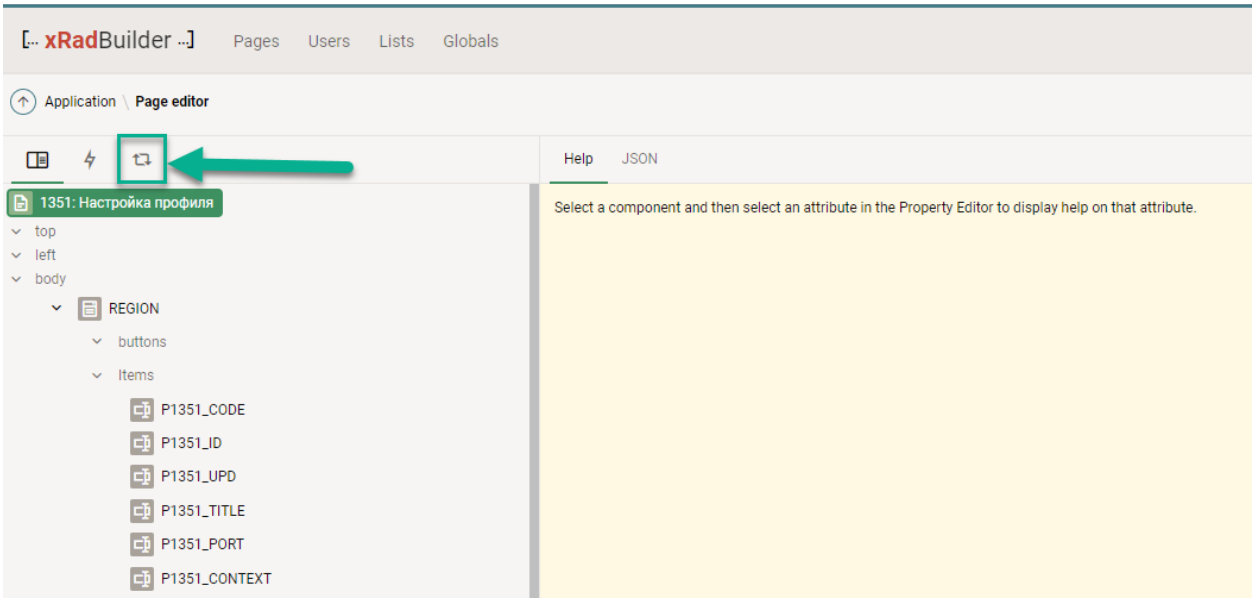
Однако имеется отдельный тип валидации – это флаг обязательности заполнения поля формы. Для любого поля ввода можно поставить флаг что поле обязательно для заполнения. В таком случае, даже если не сформировать иных валидаторов, при попытке отправить данные на сервер будет проведена проверка на заполненность поля.

6.1.2 Что происходит когда валидация не выполняется

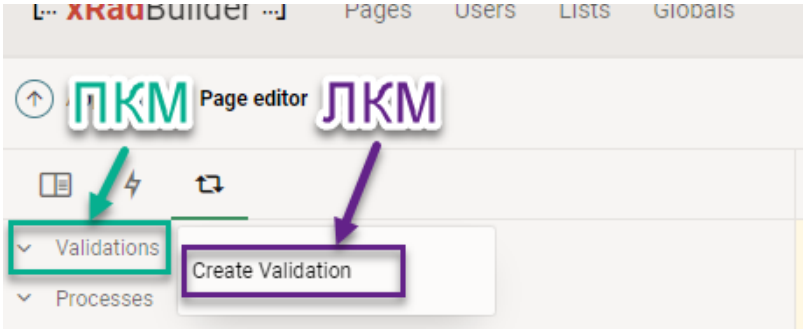
В случае возникновения исключения в любом, из присутствующих на форме, валидаторе отправка на сервер блокируется и пользователю выдается заданное разработчиком предупреждение. Если валидатор привязан к полю ввода, то так же будет подсвечено поле ввода, на котором произошло исключение.

6.1.3 Создание валидации

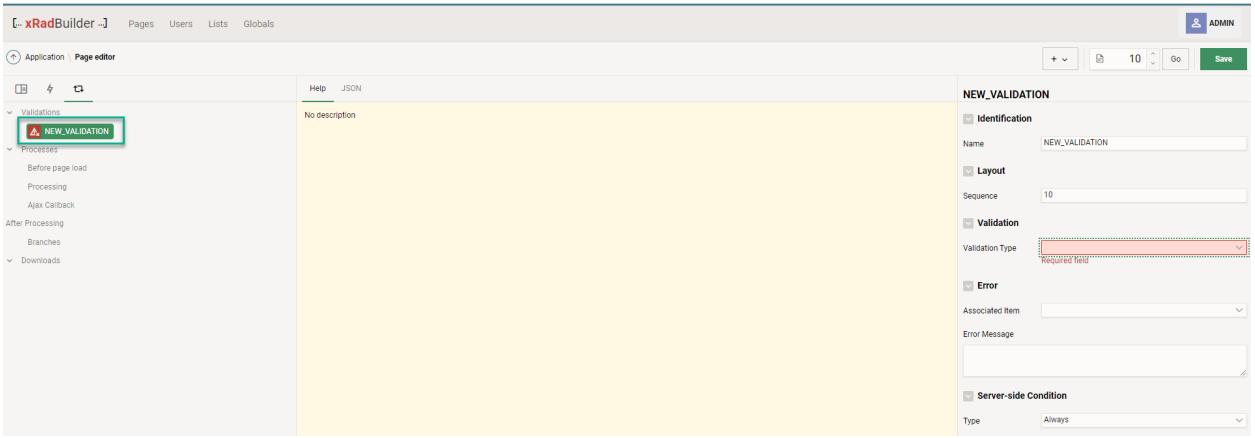
Для создания валидатора необходимо системе xRadBuilder открыть на редактирование страницу формы, на которой планируется произвести валидацию и в левой части перейти на вкладку (процессы).



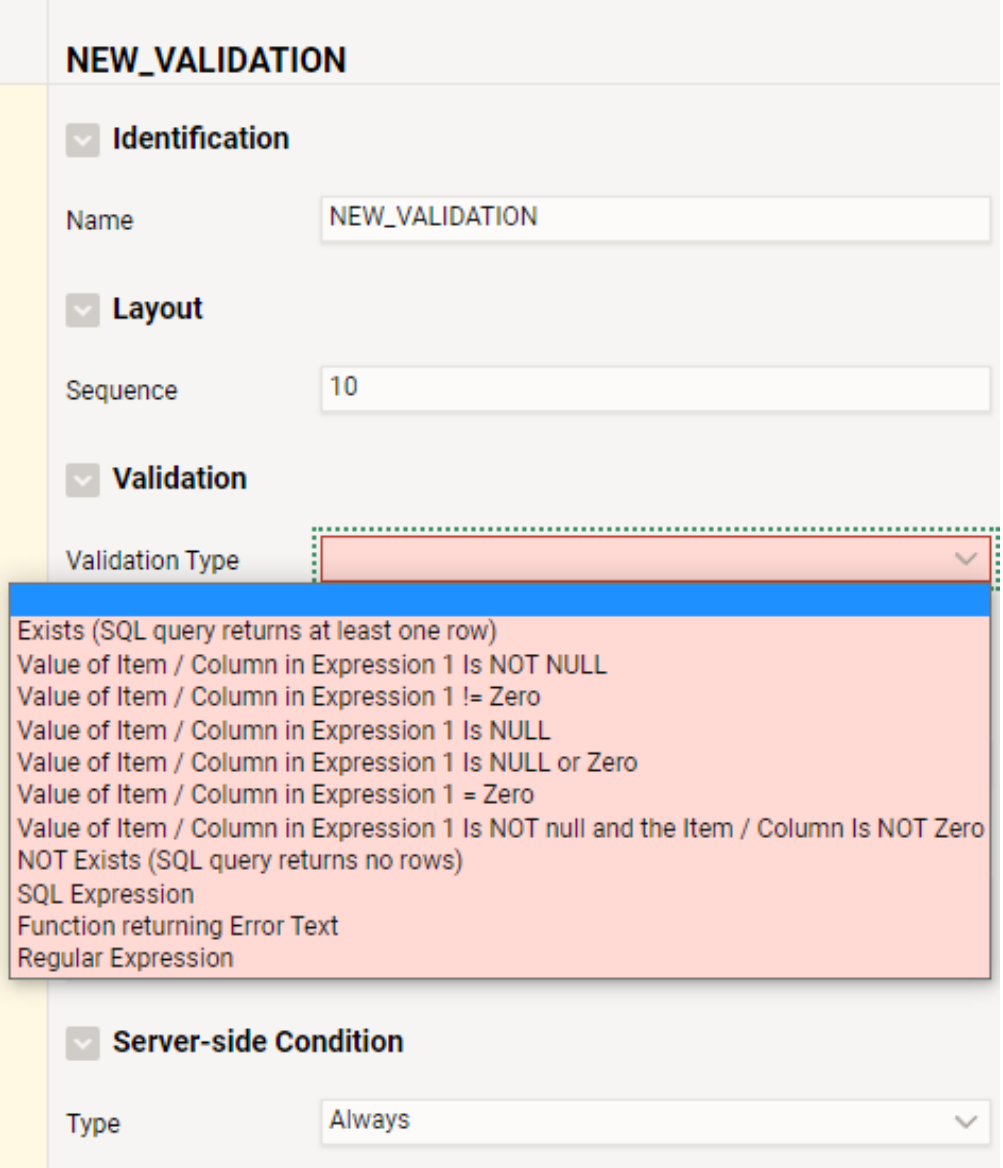
Далее щелкнуть ПКМ по раскрывающемуся списку Validations.



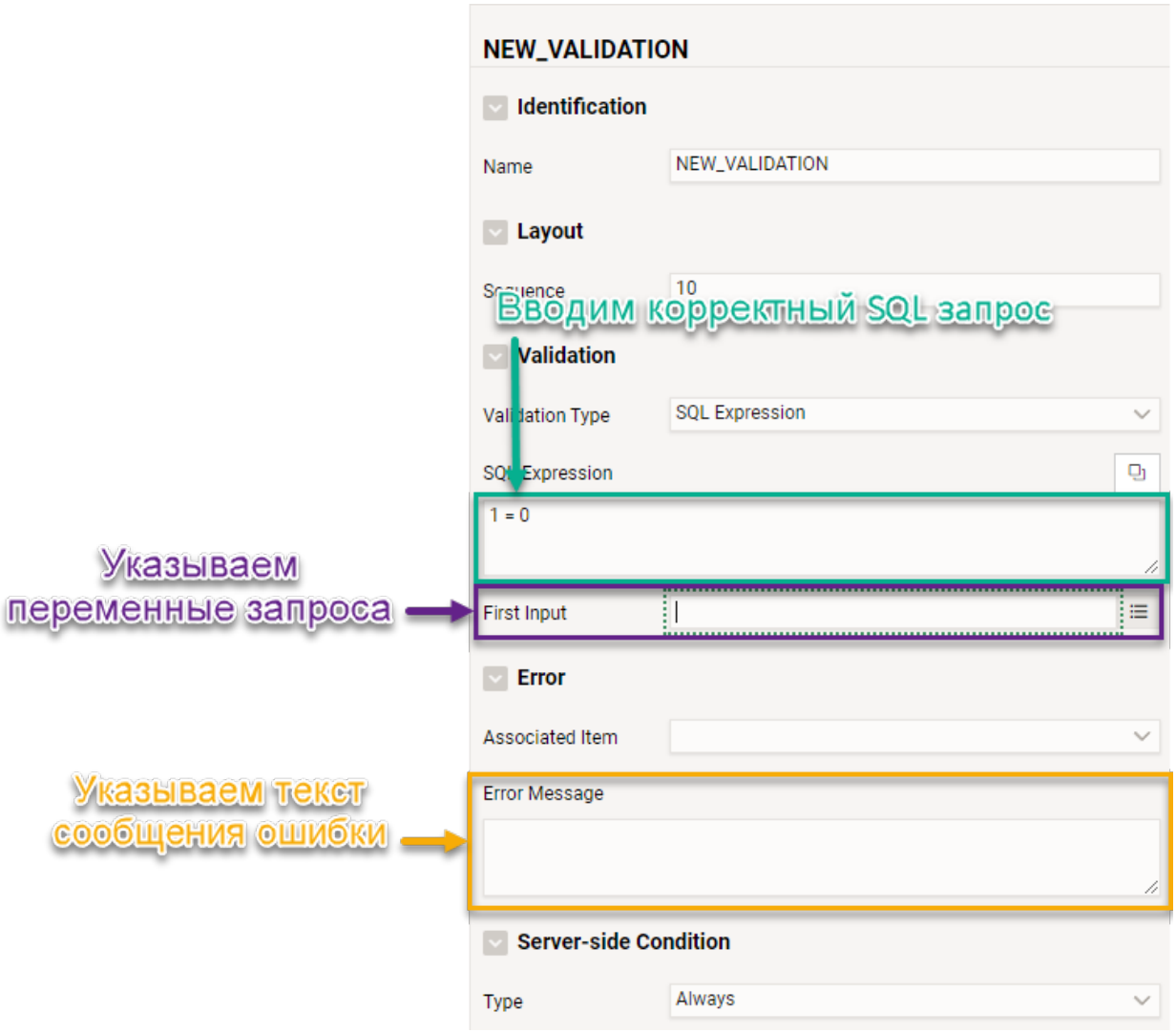
Далее выбрать Create Validation. Будет создан шаблон валидатора где автоматически будет сгенерировано имя валидатора и его порядковый номер.



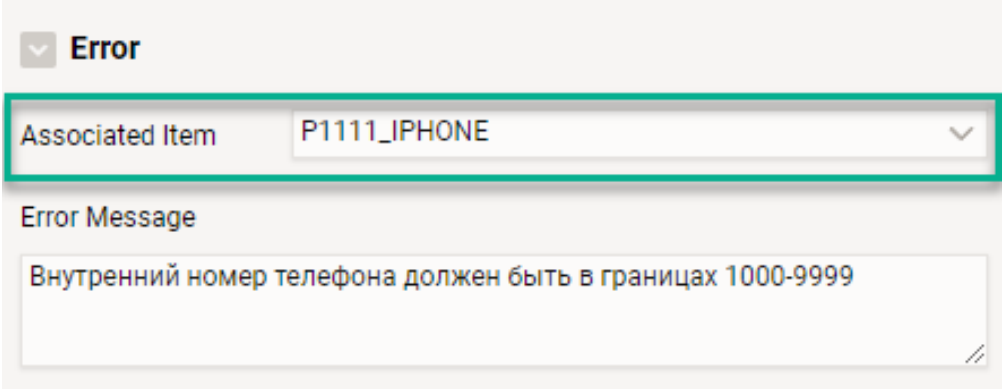
В шаблоне валидатора обязательные поля будут подсвечены красным. Сразу после создания обязательно требуется выбрать только тип проверки (о типах проверки будет рассказано в разделе *Про роли пользователей*).



В зависимости от выбранного типа проверки будет отображено дополнительное поле, которое требуется обязательно заполнить.



По желанию можно указать к какому полю ввода относится валидатор. В этом случае при возникновении исключения поле будет подсвечено как ошибочное и текст ошибки будет отображен под этим полем.



Так же, по желанию, в блоке Server-side Condition можно указать в зависимости от каких условий должна выполняться валидация.

6.1.4 Отображение ошибок валидации

Если поле отмечено как обязательное, но оно не заполнено, то можно валидаторы не создавать, а пользователь будет уведомлен о необходимости заполнить обязательные поля всплывающей подсказкой красного цвета, а так же красной подписью под полем ввода, на котором возникла ошибка:

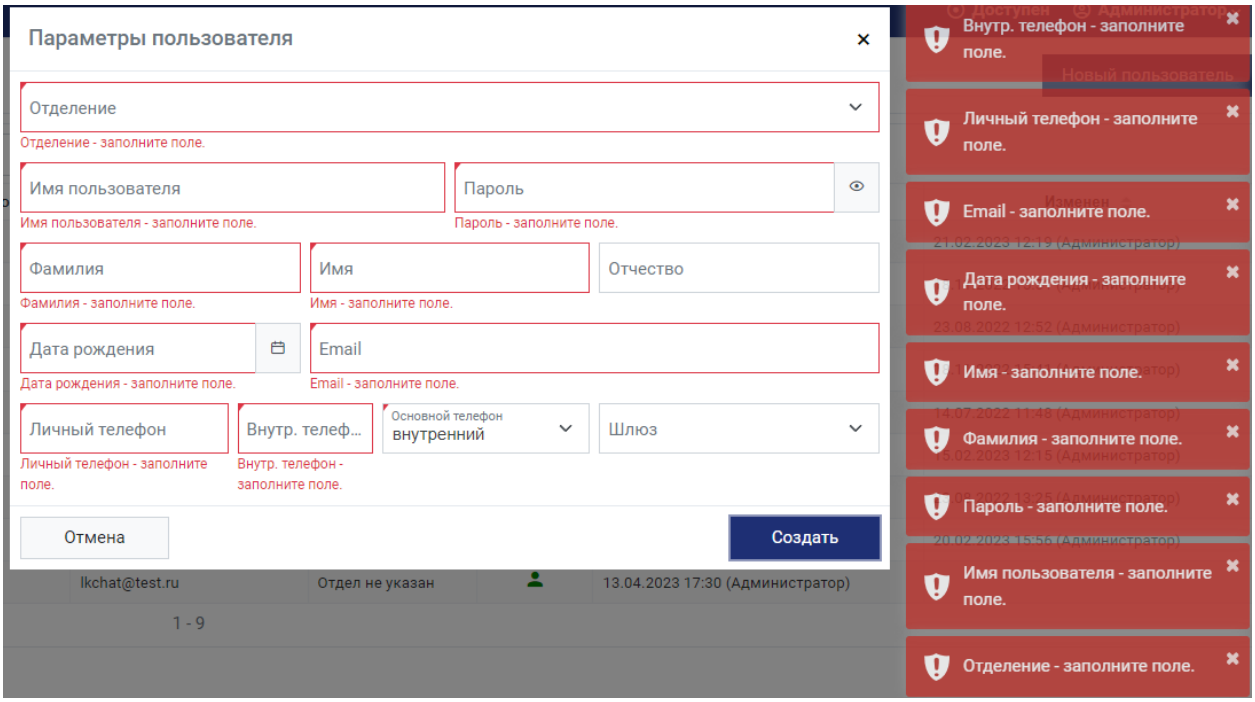
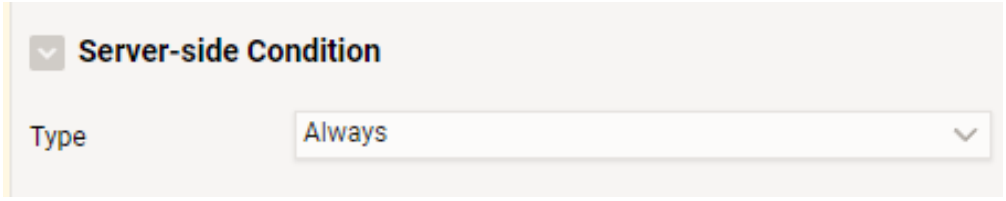


Рис. 1: Не заполнены обязательные к заполнению поля

Однако, если на поле требуется дополнительно добавить иную обработку, то необходимо создать валидатор. А для удобства пользователя привязать валидатор к полю ввода. В таком случае сначала будет выполнена проверка на заполненность, а затем будет запущен созданный валидатор и, в случае возникновения ошибки, она так же будет показана во всплывающей подсказке и текст будет продублирован под полем ввода:

6.1.5 Условия для выполнения валидации

Так как проверки выполняются автоматически при submit страницы, то часто возникают ситуации когда необходимо ограничить их выполнение. Для этого в системе XRAD предусмотрен блок условий, по которым выполняются проверки.



Разработчику доступно 11 типов условий выполнения валидатора:

1. *Always* – выполняется всегда;
2. *Exists* (SQL query returns at least one row) – позволяет выполнить проверочный SQL-запрос. Условие считается выполненным, если запрос вернул хотя бы 1 строку;

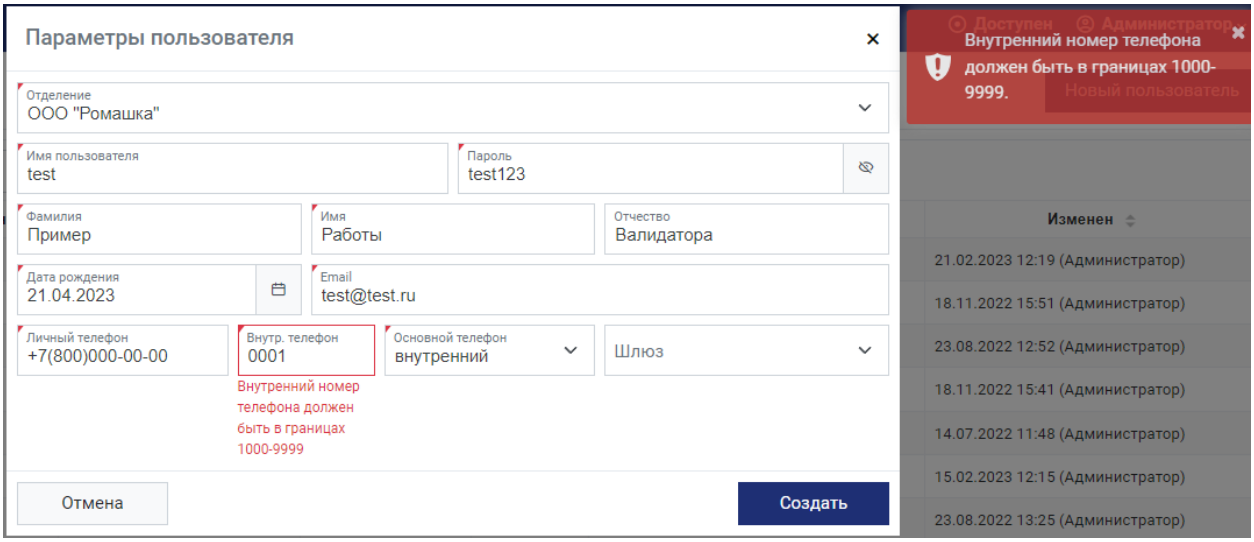


Рис. 2: Выполнение дополнительной проверки корректности введенных данных в поле

3. *Value of Item / Column in Expression 1 Is NOT NULL* – встроенное условие на заполненность поля или ячейки таблицы. Условие считается выполненным, если поле ввода или ячейка таблицы содержит какое-либо значение;
4. *Value of Item / Column in Expression 1 != Zero* – встроенное условие на значение отличное от 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки будет содержать значение отличное от 0 (нуль);
5. *Value of Item / Column in Expression 1 Is NULL* – встроенное условие на значение *NULL* (пустое значение) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки содержит NULL (пустое значение);
6. *Value of Item / Column in Expression 1 Is NULL or Zero* – встроенное условие на значение *NULL* (пустое значение) или 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки содержит *NULL* (пустое значение) или равно 0 (нуль);
7. *Value of Item / Column in Expression 1 = Zero* – встроенное условие на значение 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки будет содержать значение 0 (нуль);
8. *Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero* – встроенное условие на заполненность и значение отличное от 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки не содержит NULL (пустое значение) и не равно 0 (нуль);
9. *NOT Exists (SQL query returns no rows)* – позволяет выполнить проверочный SQL-запрос. Условие считается выполненным, если запрос не вернул ни одной строки;
10. *SQL Expression* – позволяет выполнить SQL-запрос возвращающий *TRUE* или *FALSE*. В поле ввода запроса указывается только тело запроса, без ключевых слов *SELECT* и/или *FROM*. Если необходимо выполнить какой-то подзапрос, то его необходимо обернуть в “(” “)”. Условие считается выполненным, если запрос вернул *TRUE*;
11. *Never* – не выполнять ни когда.

6.2 Процессинг страницы

При работе приложения часто требуется выполнять запросы к базе данных (далее БД). Для этих целей в системе XRAD предусмотрены процессы различных видов. Управление процессами происходит по генерации приложением различных запросов (*Requests*). При появлении запроса от приложения выполняются все процессы соответствующего запросу типа в порядке очередности (*Sequence*).

6.2.1 Что такое процесс

Процесс – это механизм взаимодействия приложения с базой данных.

В системе XRAD предусмотрены следующие виды процессов:

1. *Before page load* – процессы данного вида выполняются непосредственно перед загрузкой страницы. Их основное назначение подготовить окно для взаимодействия с пользователем (например заполнить поля формы значениями из БД);
2. *Process* – процессы данного вида выполняются при отправке формы на сервер;
3. *Ajax Callback* – процессы данного вида призваны взаимодействовать с БД по требованию от JS-скрипта.

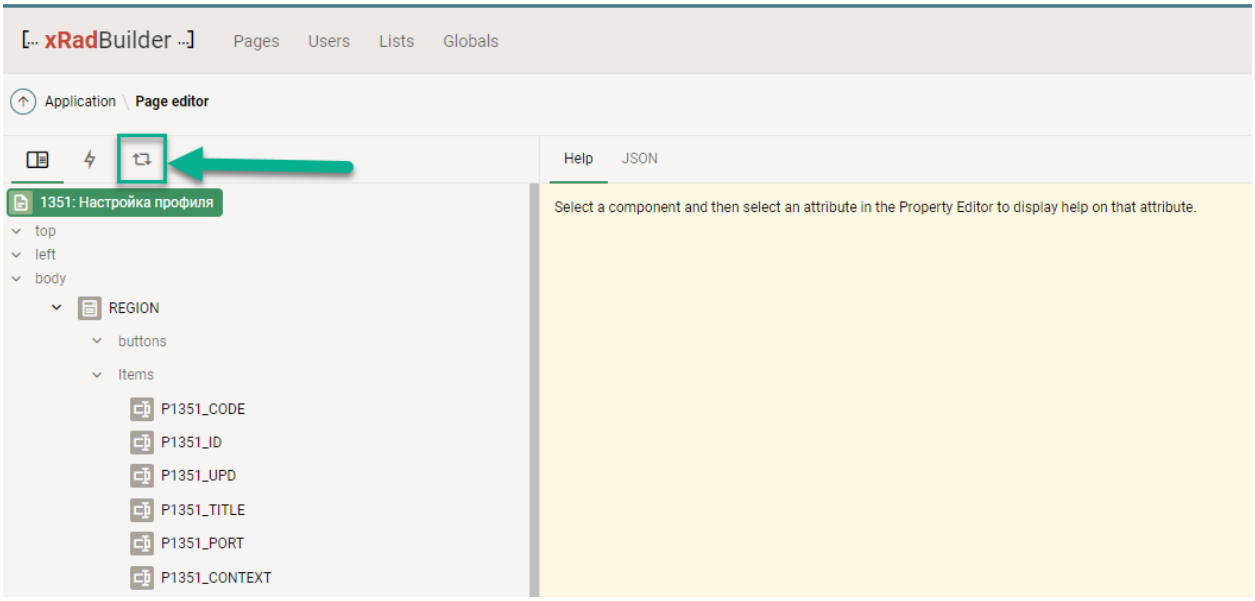
Процессы первых 2 видов так же могут самостоятельно взаимодействовать с пользователем сообщая об успешности выполнения запроса или о возникшей ошибке. Ответ от процессов третьего вида должен обрабатываться разработчиком в JS-скрипте.

По завершению работы процесса с типом Process автоматически возможно выполнить такие действия как:

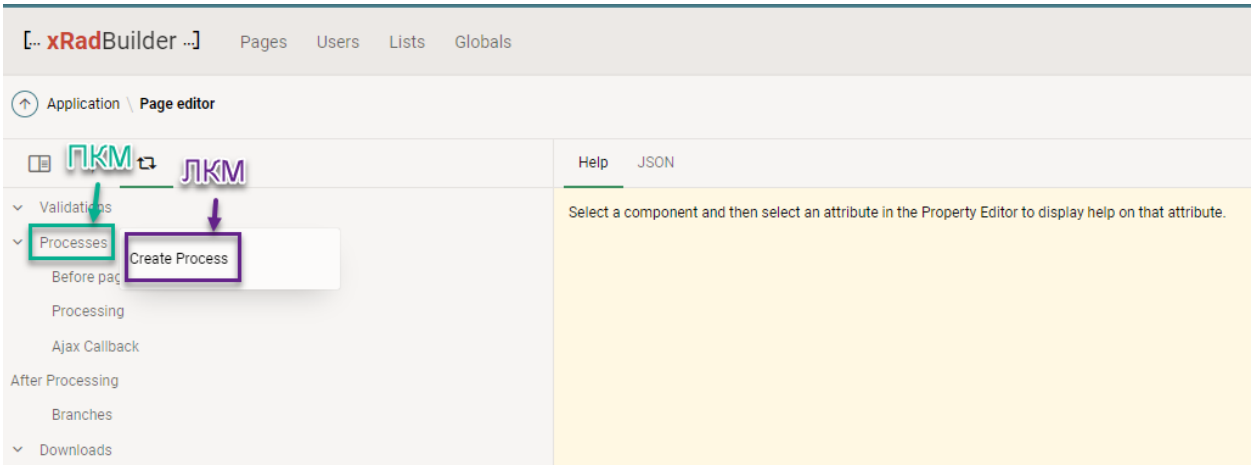
1. закрытие диалогового окна – *Accept Modal*
2. переход на другую страницу – *Branches*
3. загрузка файла – *Downloads*

6.2.2 Создание процесса

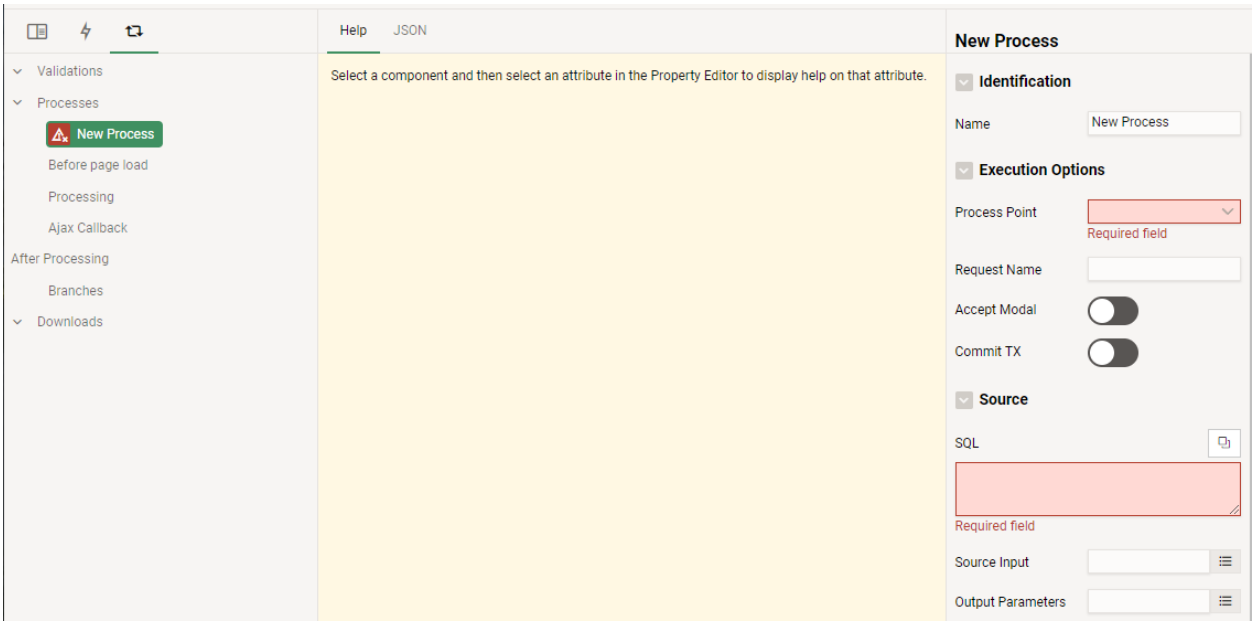
Для создания процесса необходимо в системе xRadBuilder открыть на редактирование страницу формы, для которой планируется сформировать процесс и в левой части перейти на вкладку (процессы).



Далее можно щелкнуть ПКМ по раскрывающемуся списку Process

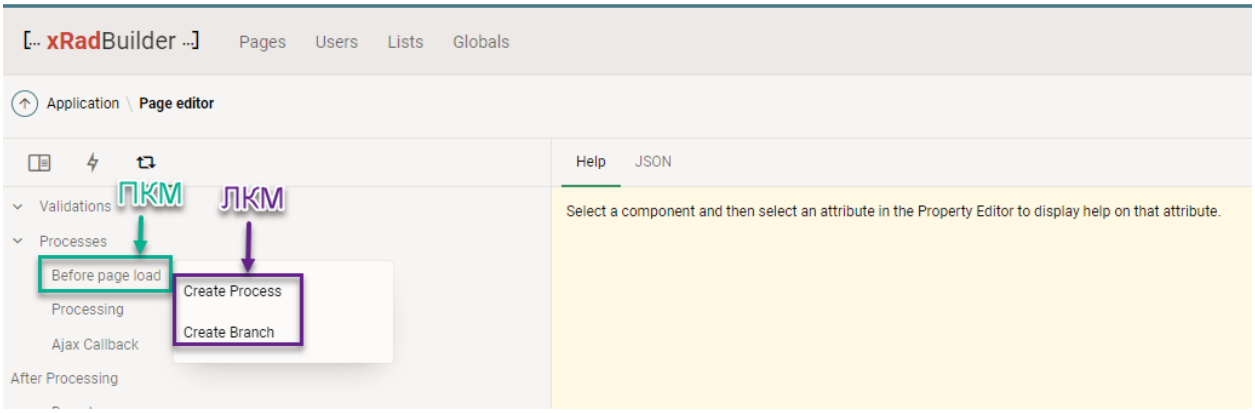


и выбрать Create Process. Будет создан шаблон процесса типа Process где автоматически будет сгенерировано имя процесса и его порядковый номер. При данном варианте потребуется указать тип процесса (Process Point) и SQL-запрос. Заполнение других полей будет зависеть от выбранного типа процесса и целесообразности

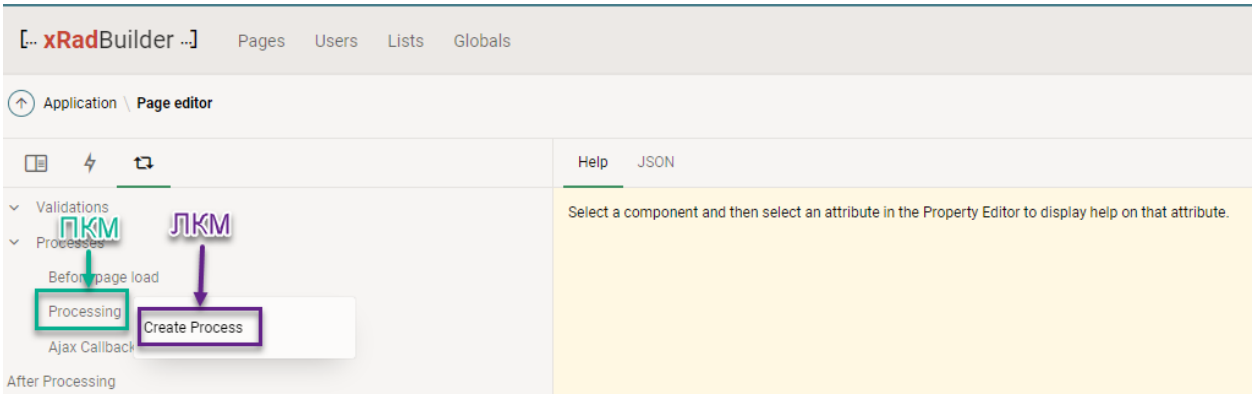


Так же можно развернуть блок Process – буду показаны типы процессов:

- *Before page load* – при щелчке на который ПКМ будет предложено 2 варианта:
  - *Create Process* – будет создан шаблон процесса выполняемого перед загрузкой страницы;
  - *Create Branch* – будет создан шаблон процесса, по выполнению условия (Server-side Condition) которого произойдет перенаправление на другую страницу.



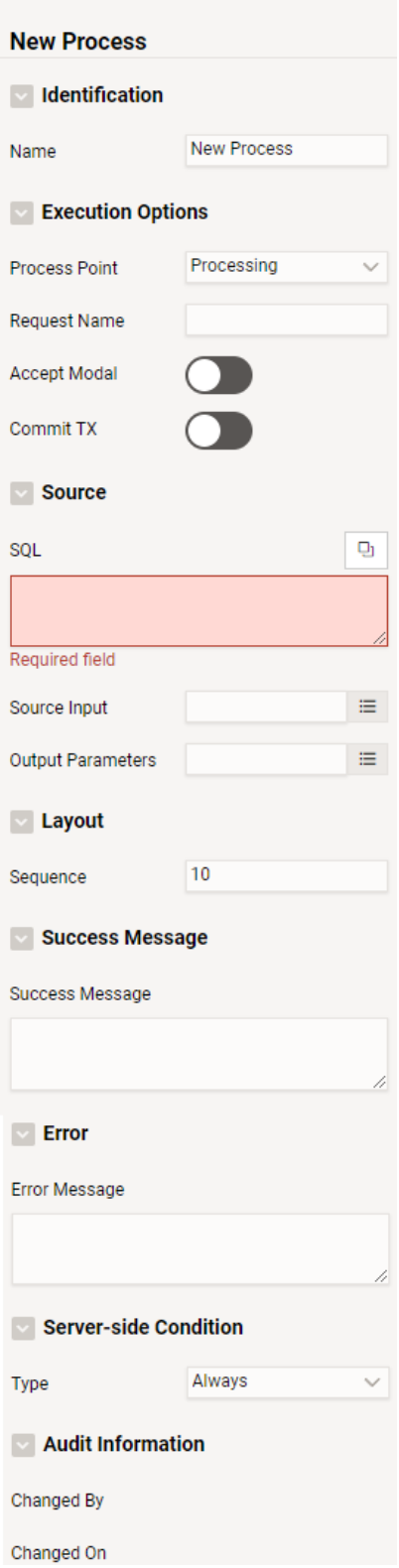
- ***Process* – при щелчке на который ПКМ будет предложен 1 вариант:**
  - *Create Process* – будет создан шаблон процесса



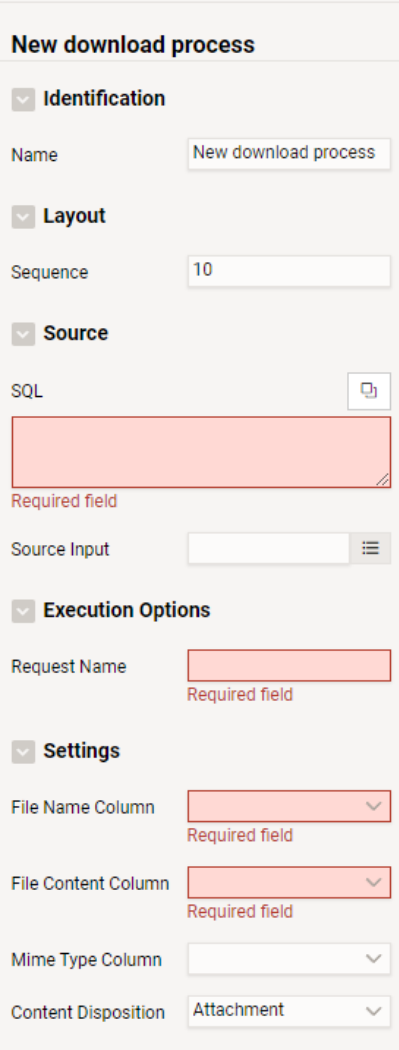
- *Ajax Callback* – при щелчке на который ПКМ будет предложен 1 вариант:
  - *Create Process* – будет создан шаблон AJAX процесса

При создании процесса через меню любого из типов поле типа процесса (Process Point) будет заполнено согласно выбранного типа.

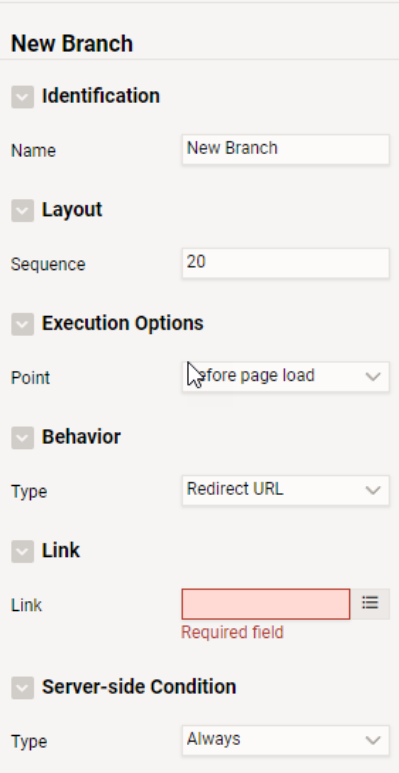
В шаблоне процесса обязательные поля будут подсвечены красным. Для большинства процессов сразу после создания обязательно требуется заполнить только выполняемый SQL-запрос.



Однако, для процессов типа Download так же обязательными являются поля Request Name, File Name Column и File Content Column



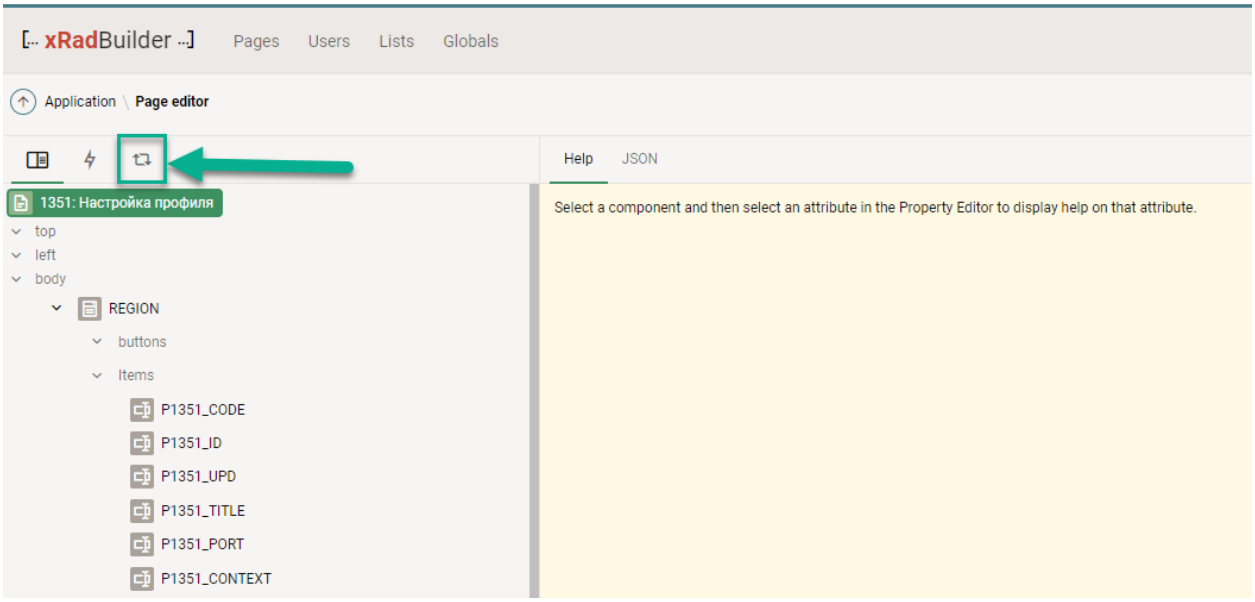
а для типа Branch обязательным является только поле Link



Для недопущения неконтролируемого выполнения процессов рекомендуется всем процессам указывать поле Request Name или выставить условие в блоке Server-side Condition.

6.2.3 Редактирование процесса

Для редактирования процесса необходимо в системе xRadBuilder открыть на редактирование страницу формы, на которой расположен процесс и в левой части перейти на вкладку (процессы).



Далее выбрать требуемый процесс и внести в его настройки требуемые изменения.

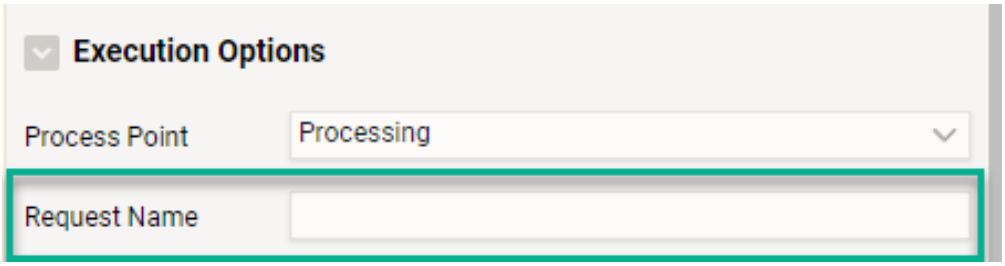
6.2.4 Условия выполнения процесса

Для выполнения процесса есть 2 вида условий:

1. выполнение по запросу (*Request*);
2. условие на стороне сервера (*Server-side condition*)

Выполнение по запросу (Request)

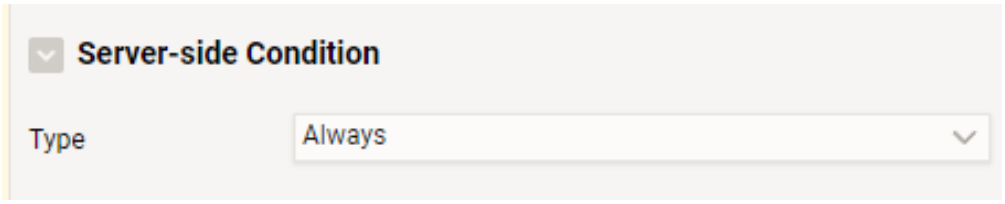
При работе приложение постоянно посылает различные запросы. Будь то *submit* страницы или запрос из JS-скрипта. В зависимости от источника запроса исполняются процессы различного типа, но их всех объединяет вид исполнения – выполняются все процессы одного типа в порядке очередности. Для того, чтобы на запрос не были выполнены не нужные процессы используется поле *Request Name*:



Если поле *Request Name* пустое, то процесс будет выполнен при любом запросе со стороны приложения, если тип процесса подходит под тип запроса. Однако, если указать конкретное имя запроса в поле *Request Name*, то процесс будет выполнен только в том случае, если имя запроса, посланного приложением, совпадет с именем в поле *Request Name*.

> Того же эффекта можно достичь используя условие на стороне сервера (*Server-side Condition*) с типом **SQL Expression**, а в качестве входного поля указать поле **REQUEST**

Условие на стороне сервера (Server-side condition)



Разработчику доступно 11 типов условий выполнения процесса:

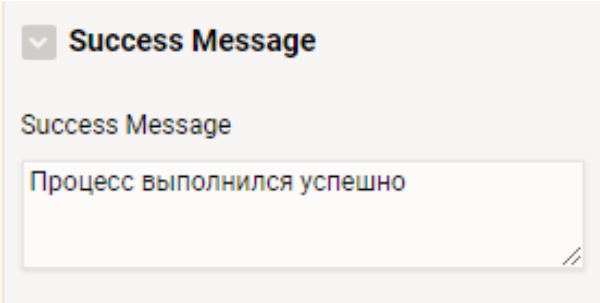
1. *Always* – выполняется всегда;
2. *Exists (SQL query returns at least one row)* – позволяет выполнить проверочный SQL-запрос. Условие считается выполненным, если запрос вернул хотя бы 1 строку;
3. *Value of Item / Column in Expression 1 Is NOT NULL* – встроенное условие на заполненность поля или ячейки таблицы. Условие считается выполненным, если поле ввода или ячейка таблицы содержит какое-либо значение;
4. *Value of Item / Column in Expression 1 != Zero* – встроенное условие на значение отличное от 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки будет содержать значение отличное от 0 (нуль);

5. *Value of Item / Column in Expression 1 Is NULL* – встроенное условие на значение *NULL* (пустое значение) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки содержит *NULL* (пустое значение);
6. *Value of Item / Column in Expression 1 Is NULL or Zero* – встроенное условие на значение *NULL* (пустое значение) или 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки содержит *NULL* (пустое значение) или равно 0 (нуль);
7. *Value of Item / Column in Expression 1 = Zero* – встроенное условие на значение 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки будет содержать значение 0 (нуль);
8. *Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero* – встроенное условие на заполненность и значение отличное от 0 (нуль) поля ввода или столбца таблицы. Условие считается выполненным, если объект проверки не содержит *NULL* (пустое значение) и не равно 0 (нуль);
9. *NOT Exists (SQL query returns no rows)* – позволяет выполнить проверочный SQL-запрос. Условие считается выполненным, если запрос не вернул ни одной строки;
10. *SQL Expression* – позволяет выполнить SQL-запрос возвращающий *TRUE* или *FALSE*. В поле ввода запроса указывается только тело запроса, без ключевых слов *SELECT* и/или *FROM*. Если необходимо выполнить какой-то подзапрос, то его необходимо обернуть в “(” “)”. Условие считается выполненным, если запрос вернул *TRUE*;
11. *Never* – не выполнять ни когда.

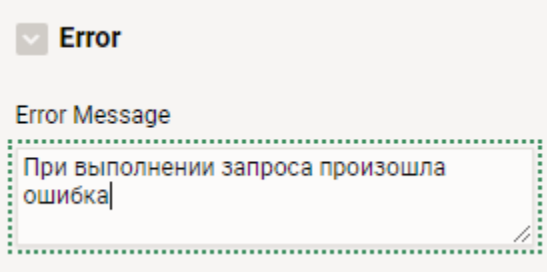
6.2.5 Сообщения успешного выполнения или ошибки

Процессы типа *Before page load* и *Process* могут самостоятельно взаимодействовать с пользователем сообщая об успешности выполнения или о возникновении ошибки.

Текст, который будет показан пользователю при успешном выполнении процесса указывается в разделе *Success Message*



Текст, который будет показан пользователю при возникновении ошибки указывается в разделе *Error Message*



При успешном выполнении на запрос нескольких процессов будет показано сообщение об успешности только последнего процесса из цепочки.

6.2.6 Управление транзакцией

При выполнении цепочки запросов к базе данных иногда возникает ситуация при которой возникновении ошибки в очередном шаге не должно приводить к отмене результатов предыдущих шагов. Эта ситуация разрешается завершением транзакции при успешном завершении процесса. Чтобы завершить транзакцию необходимо процессу выставить флаг завершения транзакции (*Commit TX*)

Если данный флаг не выставлен, то, в случае возникновения ошибки в любом из процессов, будут отменены все изменения внесенные другими процессами у которых так же не выставлен флаг завершения транзакции. Таким образом имеется возможность разбивать большое действие на группы более мелких действия ошибки в которых не будут влиять на действия выполненные в предыдущих группах.

6.2.7 Закрытие диалога

В работе приложений часто используются модальные окна для выполнения некоторых действий. Например: создание пользователя. При нажатии на кнопку “Создать” выполняется некоторый, привязанный к кнопке, процесс. В таких случаях задача модального окна выполнена и окно можно закрыть. Для облегчения задачи у процессов имеется специальный флаг

При выставленном флаге в случае успешного выполнения процесса модальное окно будет автоматически закрыто с результатом *Accept*.

> При успешном выполнении на запрос нескольких процессов модальное окно будет закрыто автоматически только в случае, если у последнего из цепочки процессов выставлен флаг *Accept Modal*.

6.3 Глобальные процессы

В главе *Процессинг страницы* были рассмотрены процессы привязанные к текущей странице. Однако, при работе приложения бывают ситуации когда необходимо выполнять один и тот же запрос к базе данных не зависимо от открытой страницы. Создавать на каждой странице приложения одинаковые процессы – не выход. Для этого предназначены глобальные процессы.

6.3.1 Что такое глобальный процесс

Глобальный процесс – это процесс XRAD не имеющий привязки к какой-либо странице. Такие процессы, так же как и процессы на страницах, выполняются по команде от приложения (*Request*).

6.3.2 Порядок выполнения глобального процесса

При появлении команды от приложения сначала выполняются все удовлетворяющие условиям команды глобальные процессы в порядке следования (*Sequence*), затем процессы текущей открытой страницы.

6.4 Асинхронные процессы

При рассмотрении процессов в разделе *Процессинг страницы* был упомянут такой тип процессов как *Ajax Callback*. Данный тип процессов позволяет выполнить запрос к базе данных из JS-скрипта приложения. А так же получить результат выполнения и обработать его в рамках скрипта. Отличие данного типа процессов от других состоит в том, что процесс не может взаимодействовать с пользователем через поля *Success Message* и *Error Message*. Для возврата результата выполнения запроса в JS-скрипт доступна глобальная переменная *RESPONSE*.

6.4.1 Создание асинхронного процесса

Создание AJAX-процесса ни чем не отличается от создания простого процесса описанного в разделе *Создание валидации*. Если процесс должен вернуть какие-либо данные для обработки в скрипт, то в поле *Output Parameters* необходимо указать глобальную переменную *RESPONSE*. Переменная *RESPONSE* принимает только текстовый результат запроса.

6.4.2 Вызов процесса

Для выполнения AJAX-процесса необходимо в JS-скрипте воспользоваться методом jsAPI.process:

```
jsAPI.process("MyRequest", {}, {
      onSuccess: function (resp) {
              // Действия после успешного выполнения процесса
              // Если процесс возвращает какие-то данные в глобальную переменную␣
↪RESPONSE,
              // то значение будет содержаться в параметре функции resp (тип JSON) в␣
↪блоке data (resp.data)
      },
      onError: function (e) {
              // Действия в случае возникновения ошибки при выполнении асинхронного␣
↪процесса
              // Информация об ошибке будет передана в парметр функции e
      }
})
```

6.5 Переходы между страницами (branches)

При работе приложения бывают ситуации при которых необходимо произвести автоматический переход на некоторую страницу при загрузке выбранной или после выполнения какого-либо процесса. Для этих целей используются переходы между страницами (Branch).

Если *Branch* расположен в блоке **Before page load**, то переход на указанную в поле Link страницу будет осуществлён при загрузке страницы, на которой расположен *Branch* при выполнении условия в блоке *Server-side Condition*. Для безусловного перенаправления в блоке *Server-side Condition* необходимо оставить значение *Always*, для отключения перенаправления – *Never*.

Если *Branch* расположен в блоке *After Processing*, то переход на указанную в поле *Link* страницу будет осуществлён при завершении всех процессовпри выполнении условия в блоке *Server-side Condition*. Для безусловного перенаправления в блоке *Server-side Condition* необходимо оставить значение *Always*, для отключения перенаправления – *Never*.

ГЛАВА 7

Управление списками

В Web-приложении используются такие объекты как меню, пути к текущей странице (breadcrumb), панели навигации и другое. Для вывода в них информации можно использовать как динамически формируемый набор значений (через SQL-запрос), так и статичный – строго определенный список не меняющийся сам по себе.

7.1 Что такое список

Список – это набор однотипных данных.

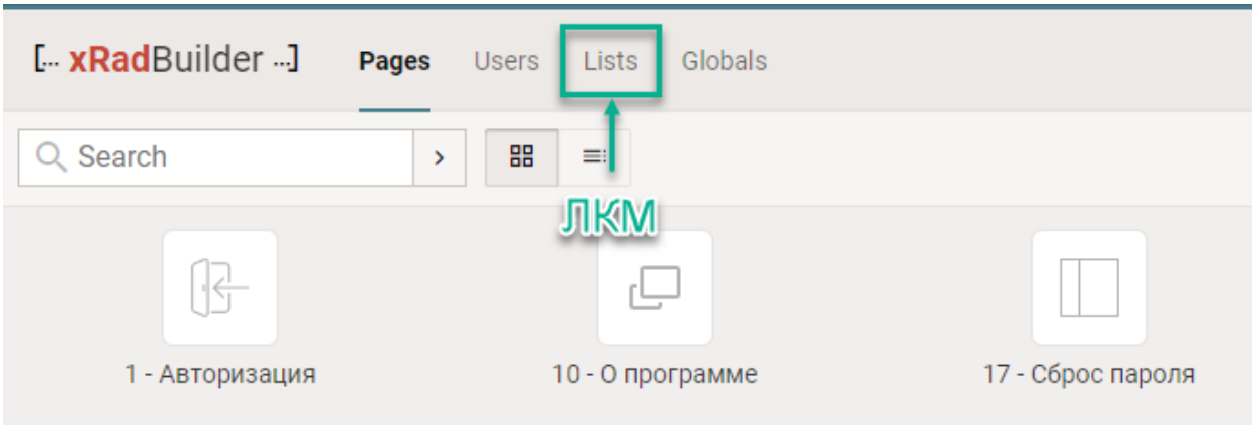
В системе XRAD существует 2 типа списков:

1. Статичный (*Static*) – все элементы списка формируются через интерфейс XRAD Builder и редактироваться может только разработчиком;
2. Динамический (*Based on Query*) – формируется на основе SQL-запроса. Может меняться пользователем через интерфейс приложения

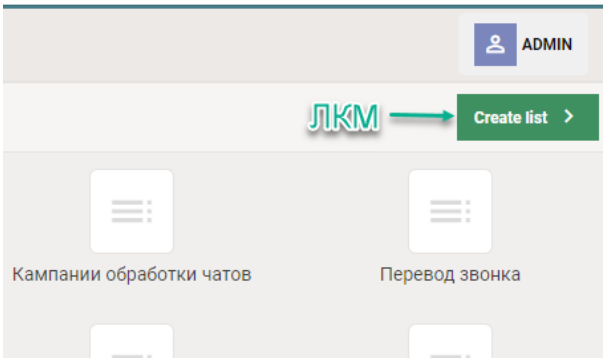
Отдельным видом списков стоит список типа *Breadcrumb* предназначенный для вывода пути страницы приложения.

7.2 Создание списка

Для создания списка необходимо перейти в раздел Списки (*Lists*):



Далее нажать кнопку Создать (Create list):



будет открыто окно создания списка:

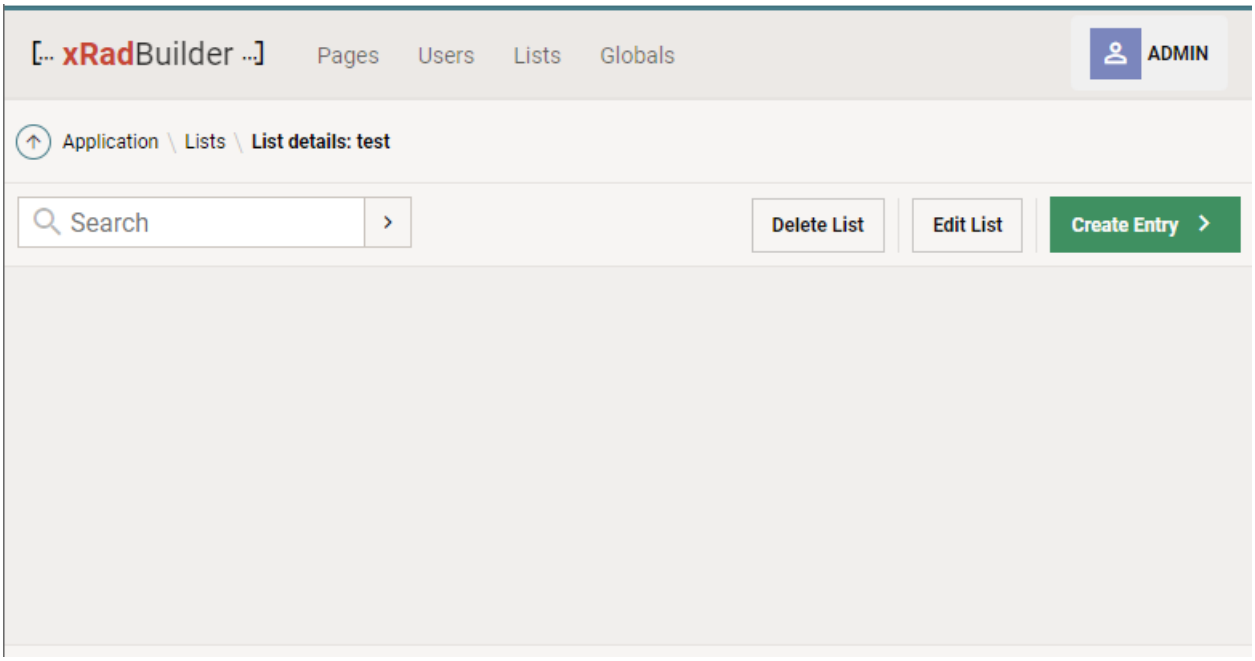
Рис. 1: Создание списка

В окне необходимо указать наименование списка и выбрать его тип. Если список планируется использовать как путь к странице, то необходимо выставить флаг *Breadcrumb*.

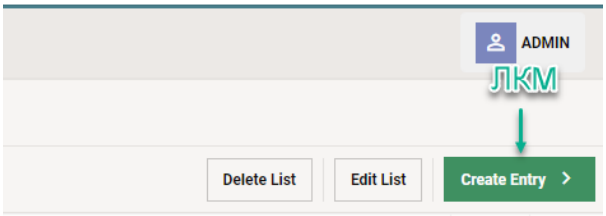
7.2.1 Создание статичного списка

Для создания статичного списка необходимо выбрать тип *Статичный* (*Static*) на этом базовые настройки списка будут завершены и можно нажать кнопку *Create*. Затем необходимо сформировать набор элементов списка.

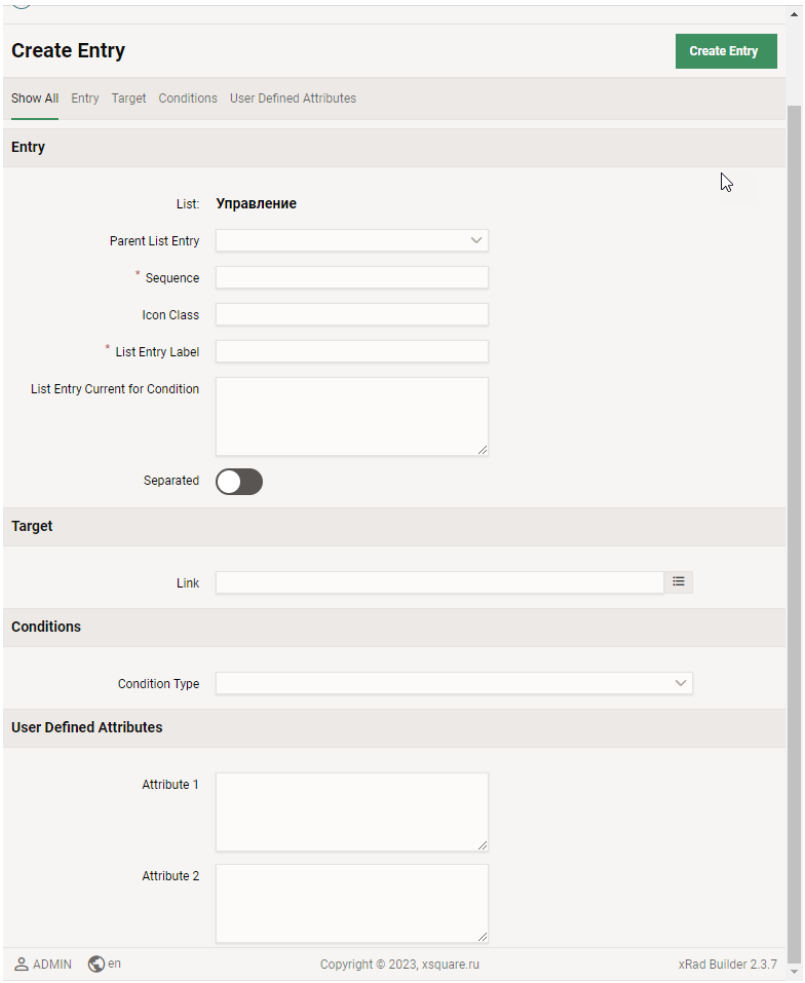
Для формирования набора элементов необходимо открыть созданный список (щелкнуть по спику ЛКМ) – будет открыто окно содержания списка:



Для создания нового элемента, требуется кликнуть на кнопку Создать элемент (*Create Entry*):



Будет открыто окно формирования параметров элемента:



Обязательные поля отмечены * (звездочкой) – таких поля 2: *Sequence* – управляет порядком отображения пунктов списка и *List Entry Label* – указывает выводимый текст.

Дополнительно можно указать:

- *Parent List Entry* – родительский пункт списка. Предназначен для формирования многоуровневого меню;
- *Icon Class* – выводимая иконка;
- *List Entry Current for Condition* – для меню указывает при нахождении на какой странице (страницах) пункт меню будет выделен как текущий;
- *Link* – действие, которое будет выполняться при нажатии на элемент списка;
- *Condition Type* – для элемента списка можно указать при каком условии пользователю будет отображен создаваемый/редактируемый элемент списка;
- *Attribute 1/2* – для некоторых вариантах вывода так же можно использовать дополнительные атрибуты списков. Например для компонента **Page Navigation** в *Attribute* 1 можно указать что будет отображаться под наименованием элемента, а в *Attribute 2* – что будет отображаться в правой части.

7.2.2 Создание динамического списка

Для создания динамического списка необходимо в качестве типа указать Динамический (*Based on Query*). Задать запрос, на основе которого будет строиться список и, по необходимости, переменные, которые используются в запросе:

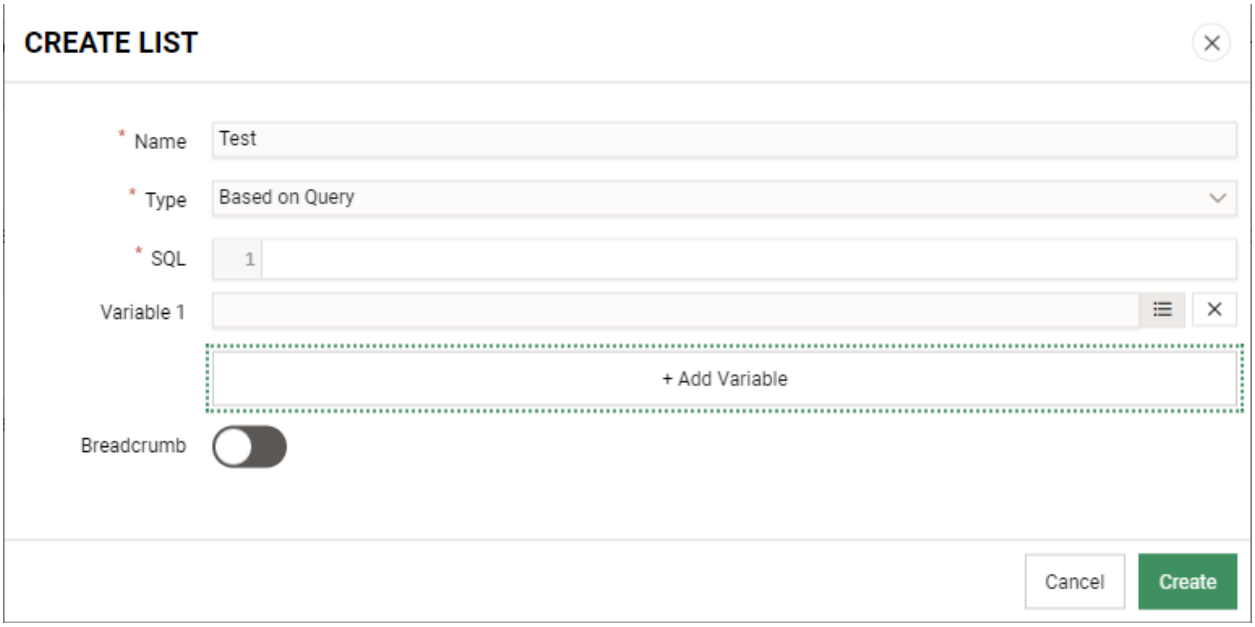


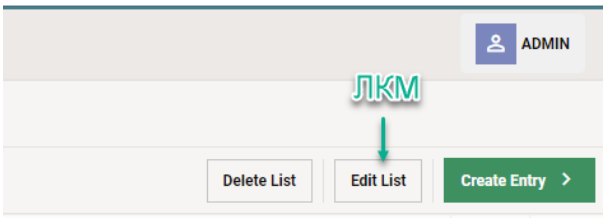
Рис. 2: Создание динамического списка

Результат запроса должен возвращать следующие поля:

- *name* – наименование элемента списка;
- *icon* – классы иконки;
- *attribute_01* – то же что и *Attribute 1* для статичного типа;
- *attribute_02* – то же что и *Attribute 2* для статичного типа;
- *target* – действие, которое будет выполняться при нажатии на элемент списка.

7.3 Редактирование списка

Для редактирования списка необходимо открыть список (щелкнуть по спику ЛКМ). Редактирование базовых настроек списка, а так же настроек динамического списка осуществляется нажатием на кнопку Редактировать список (*Edit List*):



Будет открыто окно редактирования списка:

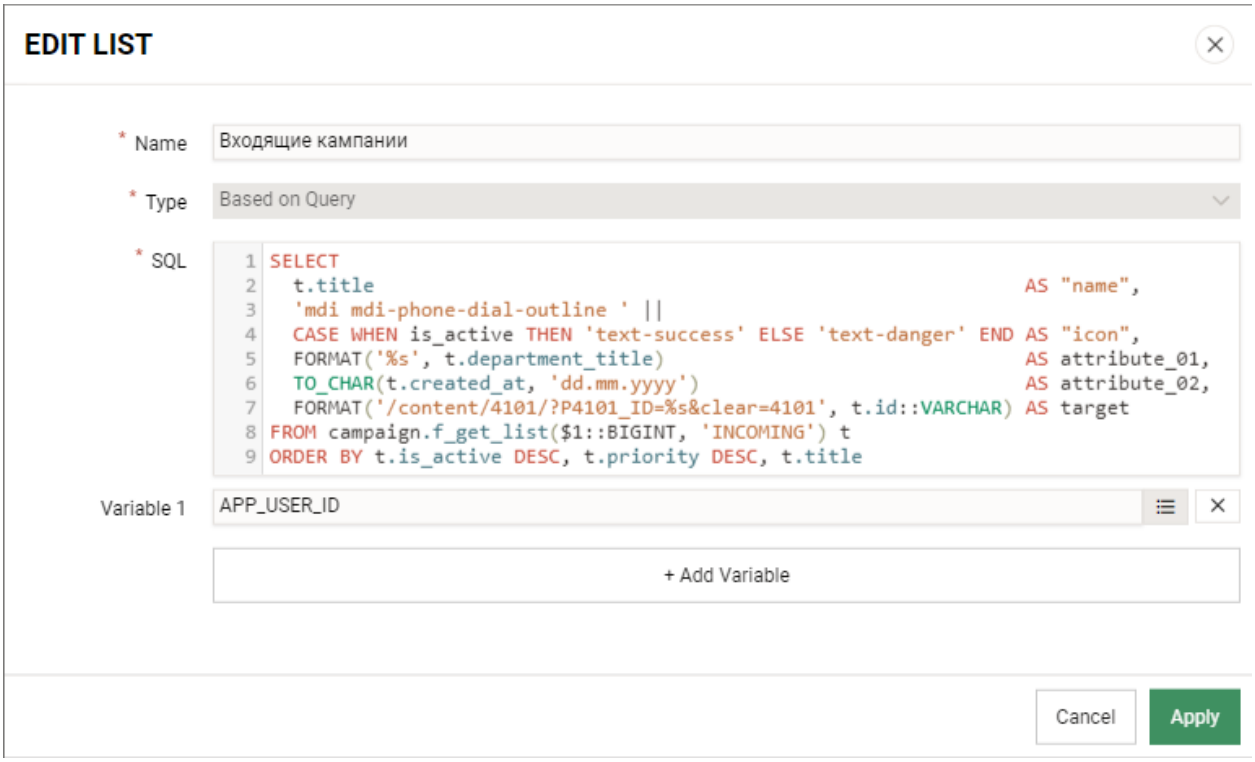


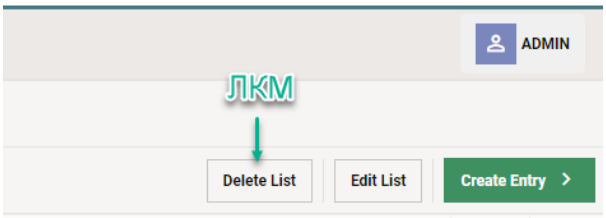
Рис. 3: Редактирование динамического списка

здесь необходимо внести требуемые изменения и нажать кнопку Применить (Apply).

Для редактирования элементов статичного списка необходимо щелкнуть ЛКМ на требуемом элементе и внести требуемые изменения. Для добавления элемента списка см. п. 7.2.1.

7.4 Удаление списка

Для удаления списка необходимо открыть список (щелкнуть по спику ЛКМ) и нажать кнопку Удалить список (*Delete List*):

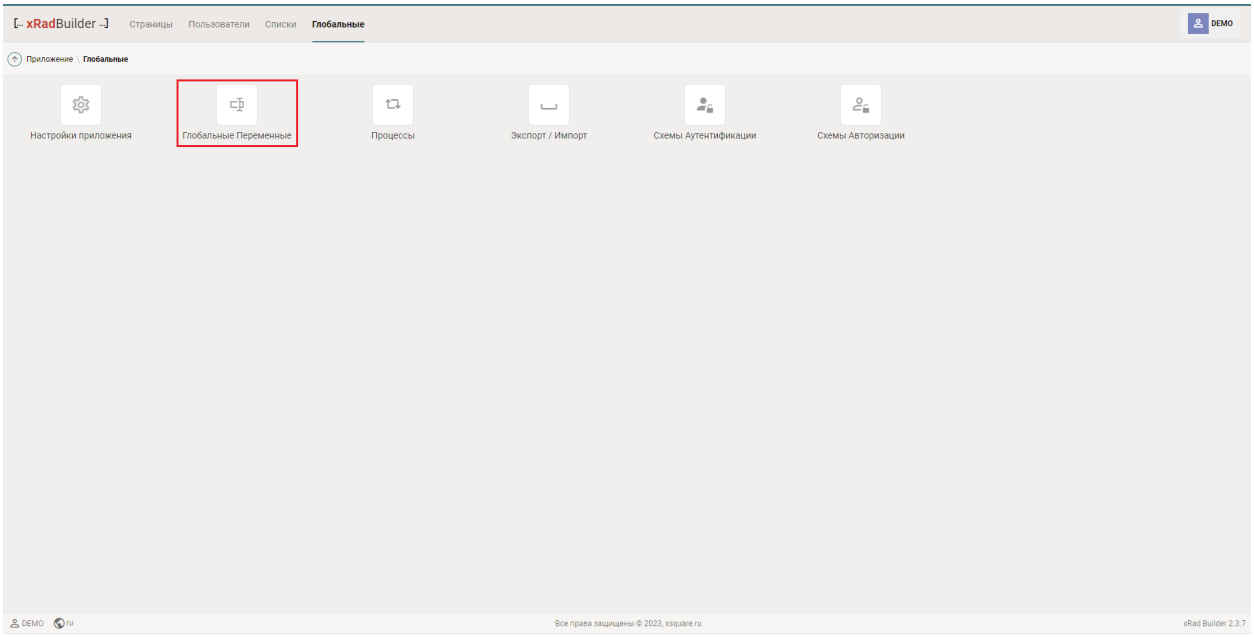


ГЛАВА 8

Глобальные переменные

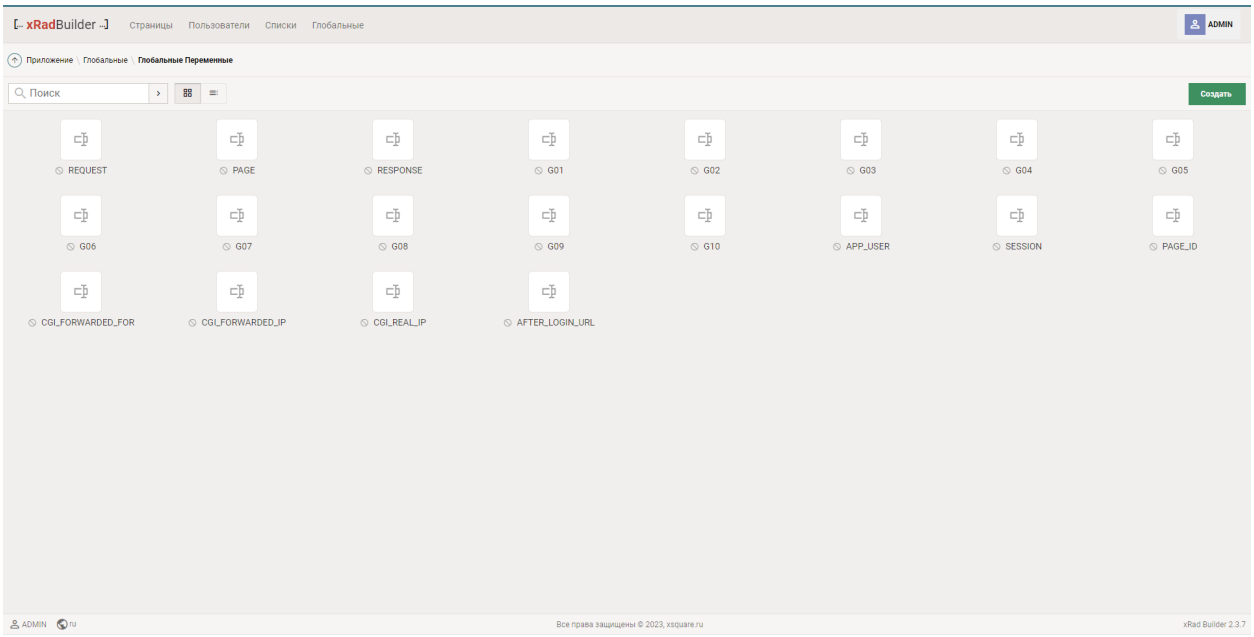
Глобальные переменные - это переменные, которые доступны и повсеместно используются в приложении для хранения какого-либо значения.

Страница с глобальными переменными располагается во вкладке “Глобальные”:



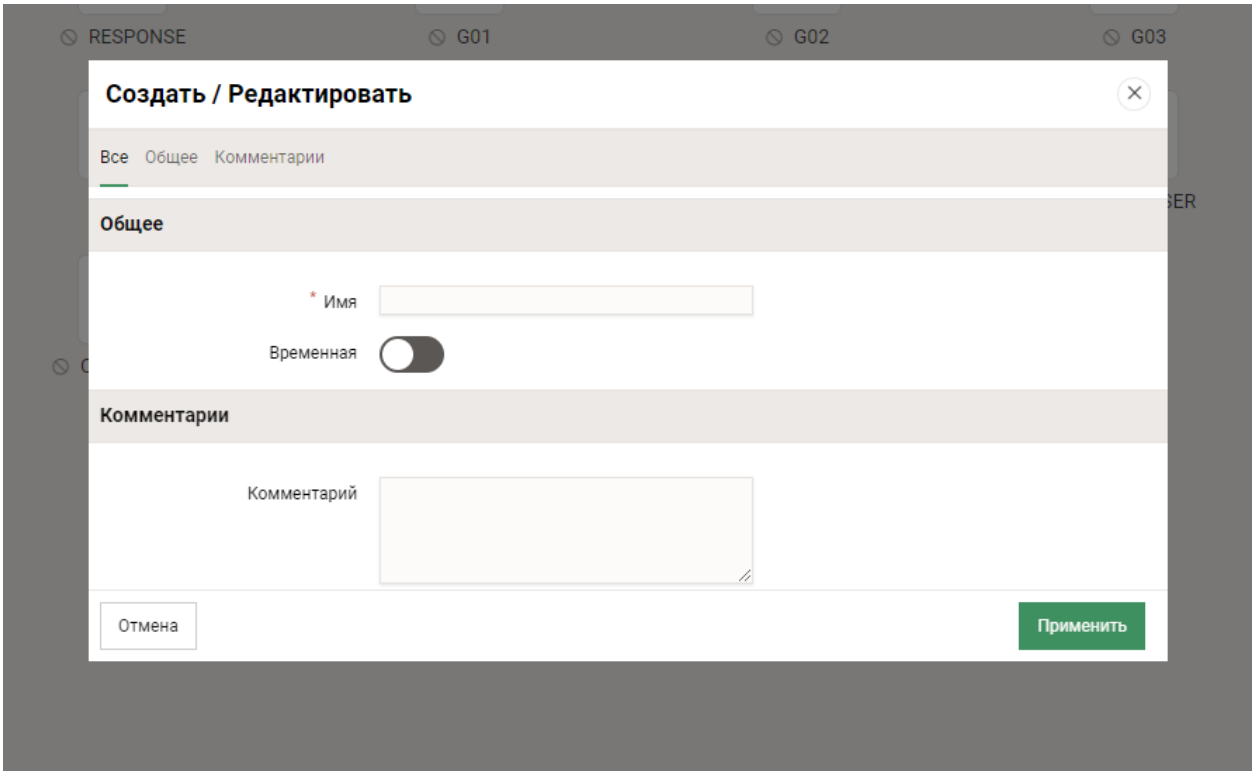
При переходе на страницу “Глобальные Переменные” будет отображен перечень всех глобальных переменных, объявленных в приложении, и кнопка для создания новой глобальной переменной:

Для перехода на страницу необходимо авторизоваться от лица аккаунта с ролью доступа выше VIEW.



8.1 Создание глобальных переменных

Объявление новой глобальной переменной осуществляется по нажатию кнопки “Создать”. Откроется форма создания глобальной переменной:



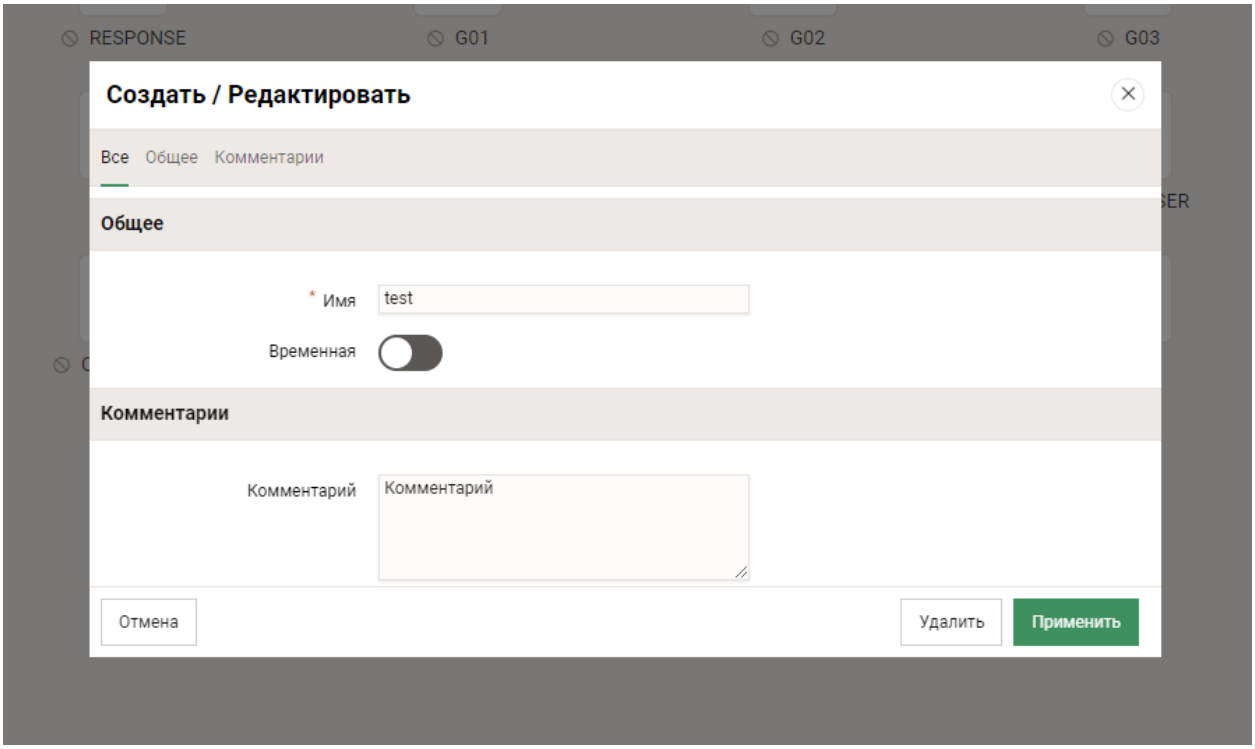
8.1.1 Атрибуты:

- Имя - единственный **обязательный** атрибут, определяющий имя создаваемой переменной;
- Временная - определяет, будет ли переменная временной: Временная переменная в процессе работы программы автоматически задает и очищает собственное значение после использования;
- Комментарий - текстовый комментарий.

После подтверждения введенных данных переменная будет занесена в базу данных XRAD и доступна для просмотра, редактирования и удаления.

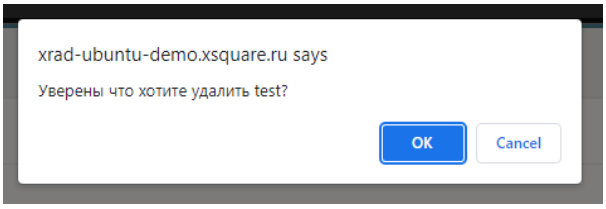
8.2 Редактирование глобальных переменных

Для редактирования переменной выберите нужную вам (не предустановленную) переменную. После нажатия на переменную откроется форма, аналогичная форме создания новой переменной (см. *Создание глобальных переменных*), с информацией о выбранной переменной:



Отредактируйте необходимые вам атрибуты и сохраните их новые значения, нажав кнопку “Применить”.

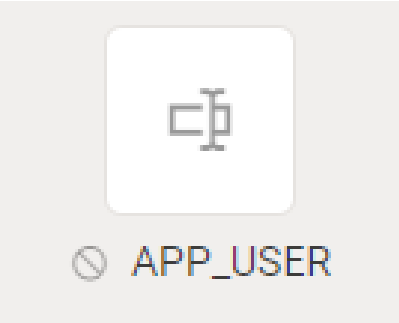
Для удаления переменной нажмите кнопку “Удалить” и подтвердите действие в появившемся диалоговом окне браузера:



8.3 Значения предустановленных переменных

Предустановленные переменные - это переменные, которые присутствуют в каждом приложении на базе платформы XRAD. Данные переменные не подлежат редактированию и удалению.

Понять, является ли какая-либо переменная предустановленной, помогает специальный значок с левой стороны от имени переменной:



Примечание: Наличие этого значка означает невозможность редактирования данной переменной. Обычно данным свойством обладают только предустановленные глобальные переменные.

8.3.1 Часто используемые предустановленные переменные:

Временные:

- **REQUEST** - код запроса для выполнения необходимого процесса;
- **PAGE** - номер страницы;
- **RESPONSE** - выходной параметр выполненного процесса Ajax;

Постоянные:

- **APP_USER** - id авторизованного на данный момент пользователя;
- **SESSION** - уникальный код сессии.

ГЛАВА 9

Безопасность

В данном разделе описаны следующие параметры безопасности приложения:

- *Схемы аутентификации*
- *Схемы авторизации*
- *Контрольные суммы*

9.1 Схемы аутентификации

9.1.1 Общее описание

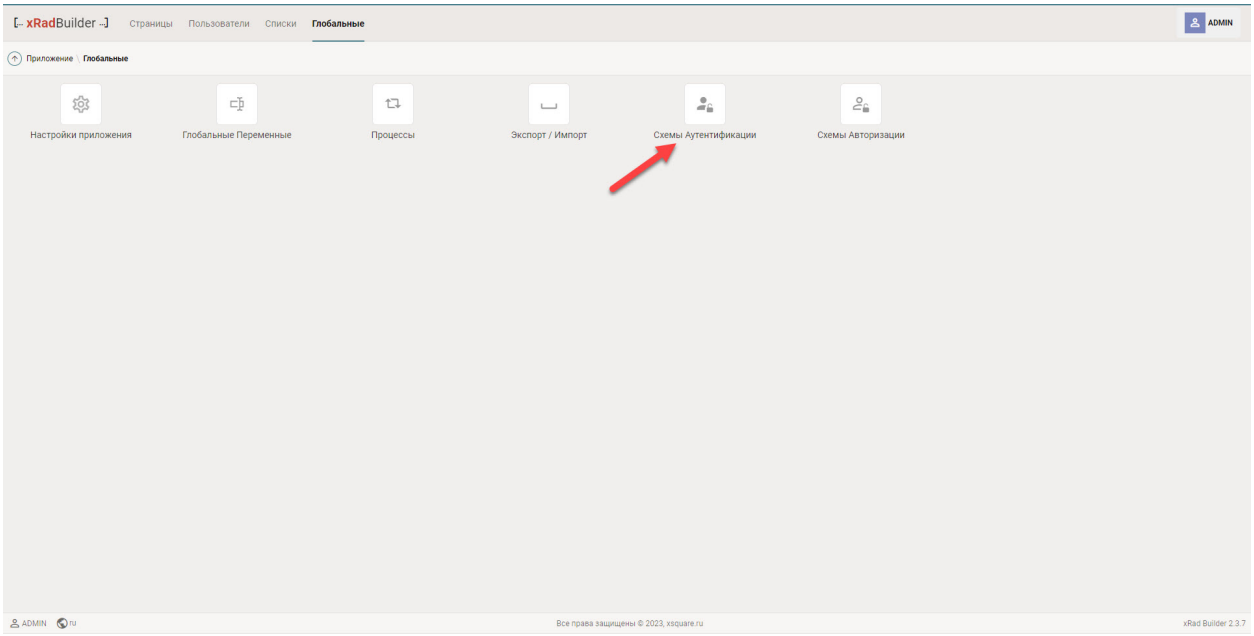
Аутентификация - это процесс проверки подлинности данных каждого пользователя, который получает доступ к приложению.

Процесс проверки подлинности требует, чтобы пользователь предоставил определенные учетные данные, например, имя пользователя и пароль. Если учетные данные проходят проверку, пользователь получает доступ к приложению, если нет, то доступ будет запрещен.

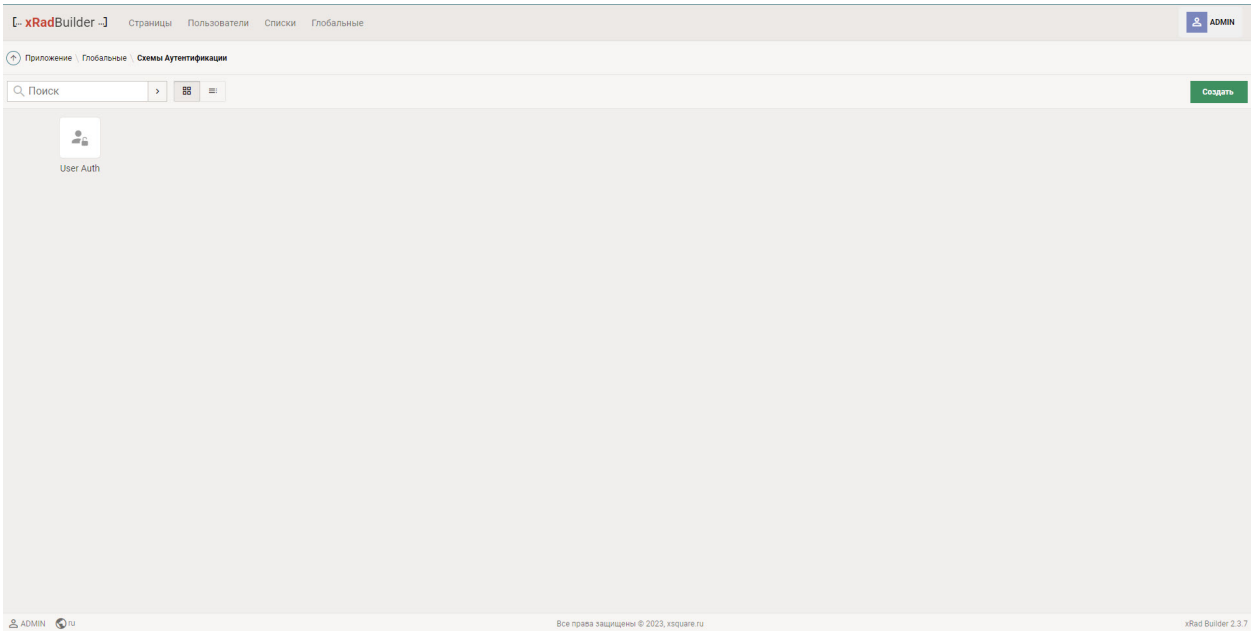
Как только пользователь идентифицирован, устанавливается значение глобальной переменной APP_USER (см. *Глобальные переменные*). Когда пользователь переходит со страницы на страницу, значение APP_USER используется для идентификации пользователя.

9.1.2 Создание схемы аутентификации

Страница со схемами аутентификации располагается во вкладке “Глобальные”:

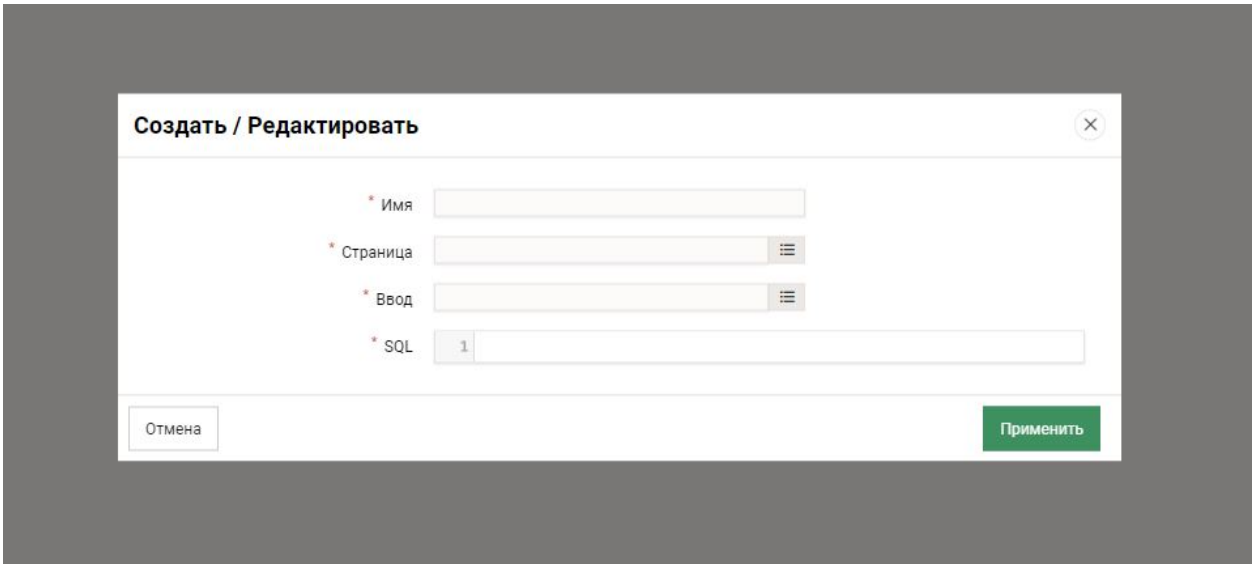


При переходе на страницу “Схемы Аутентификации” будет отображен перечень всех схем аутентификации, доступных в приложении, и кнопка для создания новой схемы:



Для перехода на страницу необходимо авторизоваться от лица аккаунта с ролью доступа выше VIEW.

Создание новой схемы аутентификации осуществляется по нажатию кнопки “Создать”. Откроется форма создания схемы аутентификации:



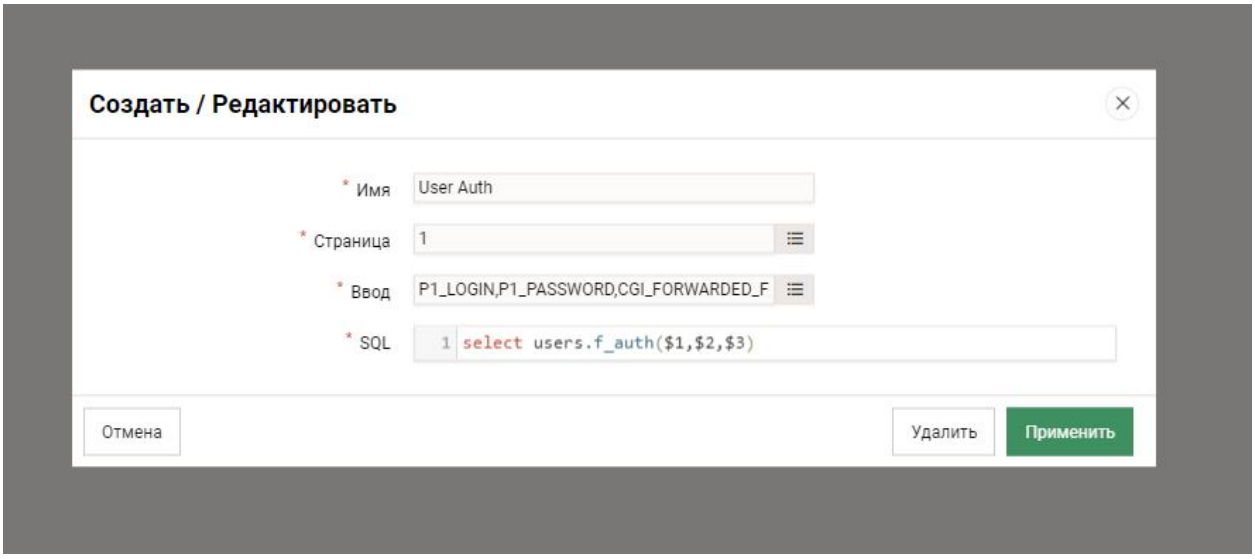
Атрибуты:

- Имя - **обязательный** атрибут, определяющий имя создаваемой схемы аутентификации;
- Страница - **обязательный** атрибут, определяющий номер страницы приложения, на которой происходит аутентификация (см. ниже пункт “Создание страницы для аутентификации”);
- Ввод - **обязательный атрибут**, определяющий список входных параметров для аутентификации;
- SQL - **обязательный атрибут**, определяющий SQL-запрос для аутентификации.

После подтверждения введенных данных схема аутентификации будет занесена в базу данных XRAD и доступна для просмотра, редактирования и удаления.

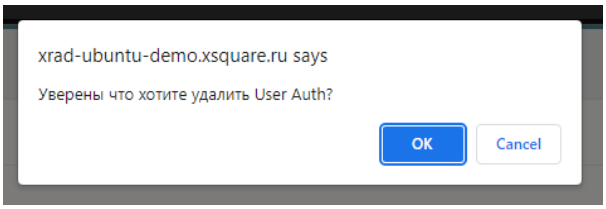
9.1.3 Редактирование схемы аутентификации

Для редактирования схемы аутентификации выберите нужную вам схему. После нажатия на схему откроется форма, аналогичная форме создания новой схемы аутентификации (см. выше “Создание схемы аутентификации”), с информацией о выбранной схеме:



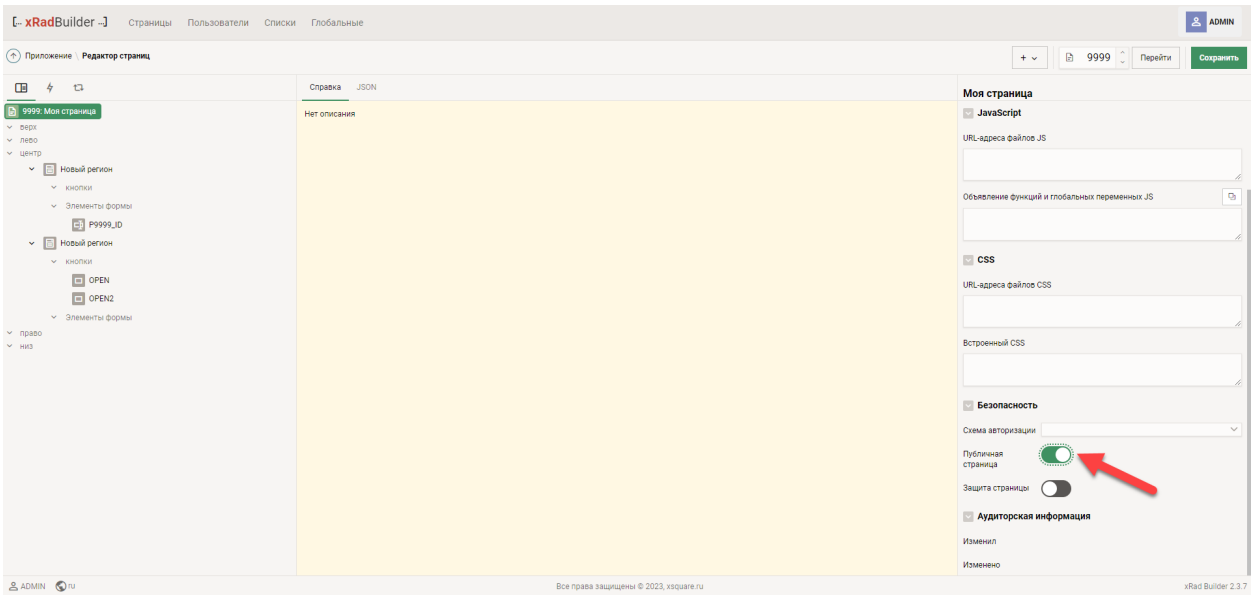
Отредактируйте необходимые вам атрибуты и сохраните их новые значения, нажав кнопку “Применить”.

Для удаления схемы нажмите кнопку “Удалить” и подтвердите действие в появившемся диалоговом окне браузера:



9.1.4 Публичная страница

Определить для страницы, будет ли она публичной или будет нужна аутентификация, можно в *редактор страниц* через свойство “Безопасность - Публичная страница”.



Если свойство “Публичная страница” не установлено, то необходима аутентификация пользователя для просмотра страницы.

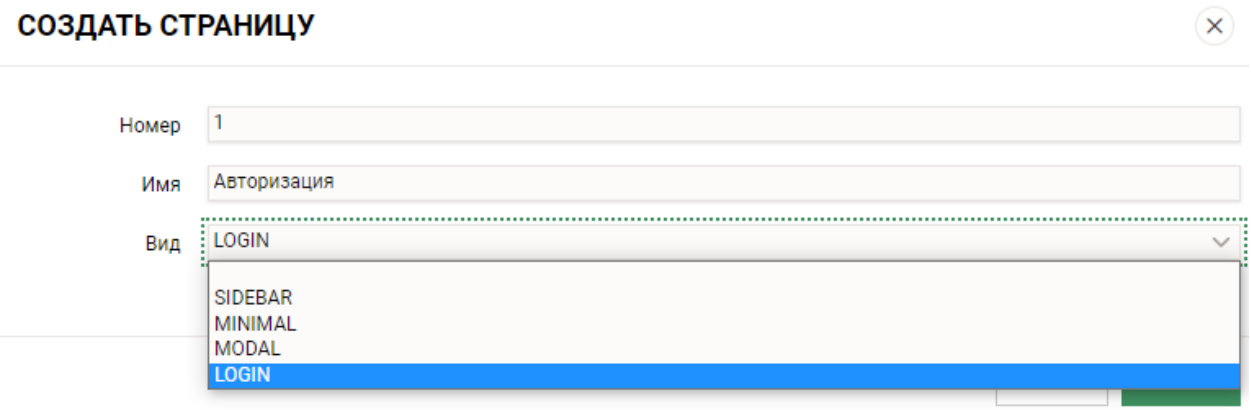
Если свойство “Публичная страница” установлено, то страница будет являться публичной и нет необходимости в аутентификации пользователя для просмотра страницы.

Данное свойство будет применено к странице после сохранения изменений через кнопку “Сохранить”.

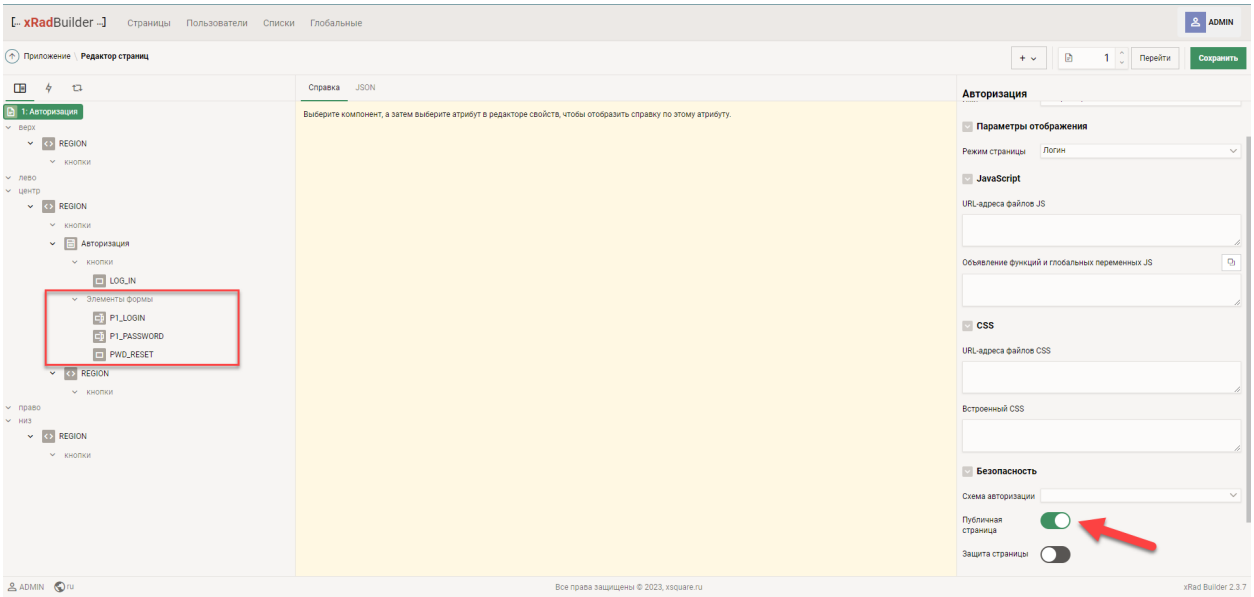
9.1.5 Создание страницы для аутентификации

Шаги по созданию страницы приложения описаны в разделе *Добавление новой страницы*

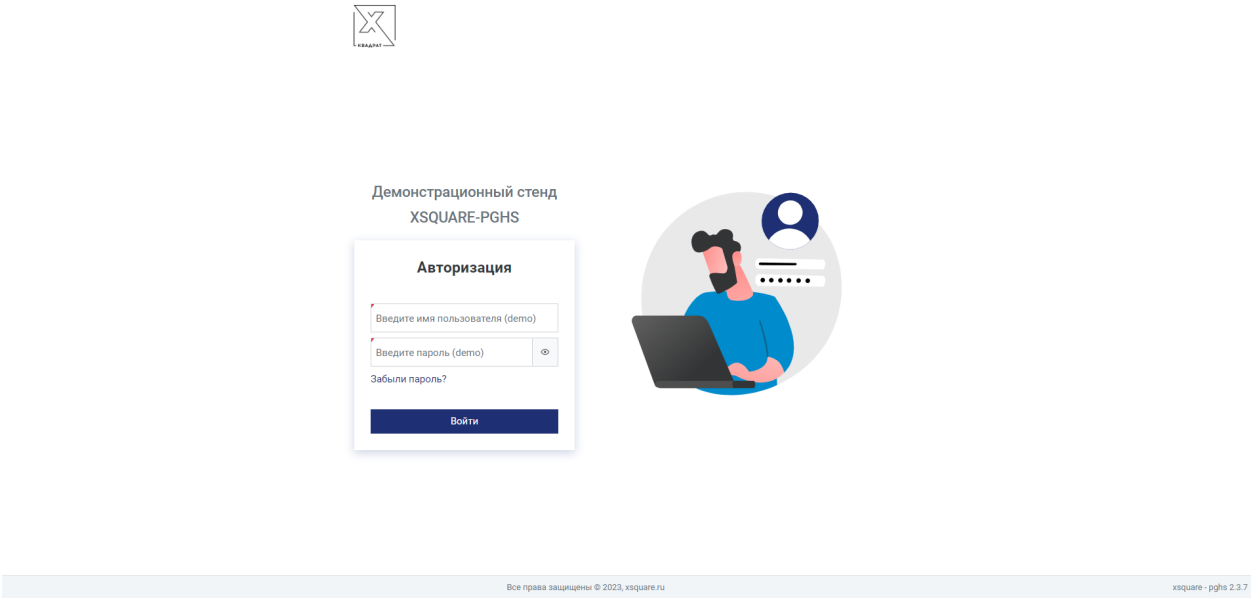
Для страницы аутентификации необходимо выбрать режим LOGIN.



Далее необходимо указать атрибуты страницы в редакторе страниц и установить свойство “Публичная страница”:



Страница для аутентификации в приложении будет выглядеть следующим образом:



9.2 Схемы авторизации

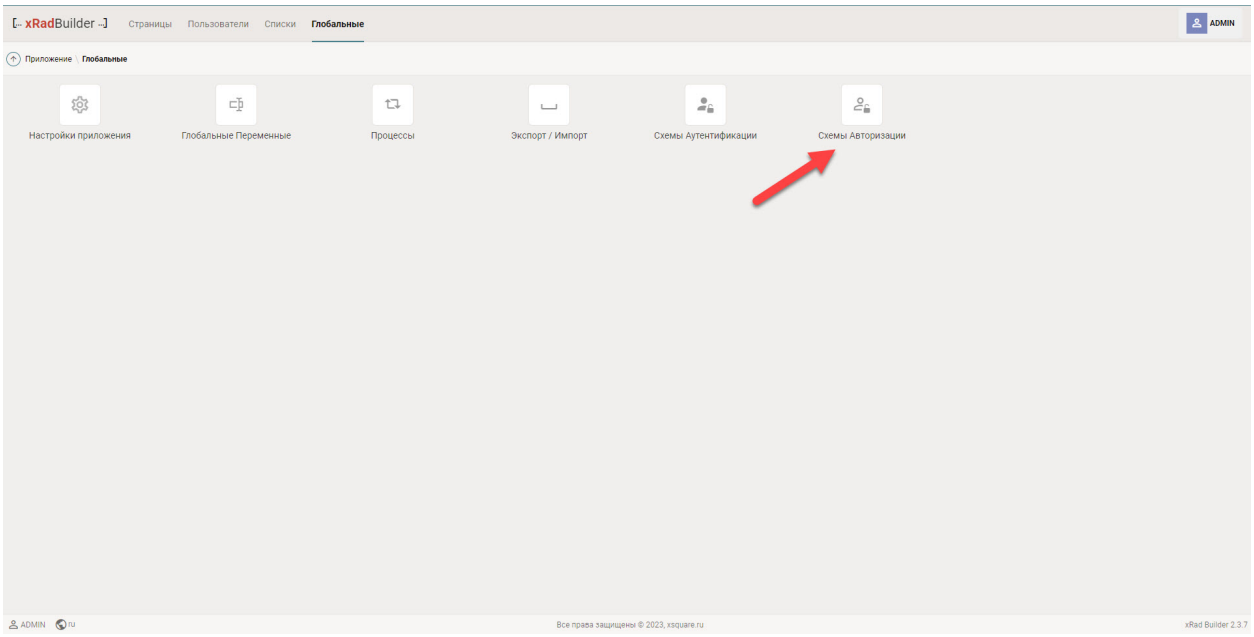
9.2.1 Общее описание

Схема авторизации определяет доступность страницы конкретному пользователю.

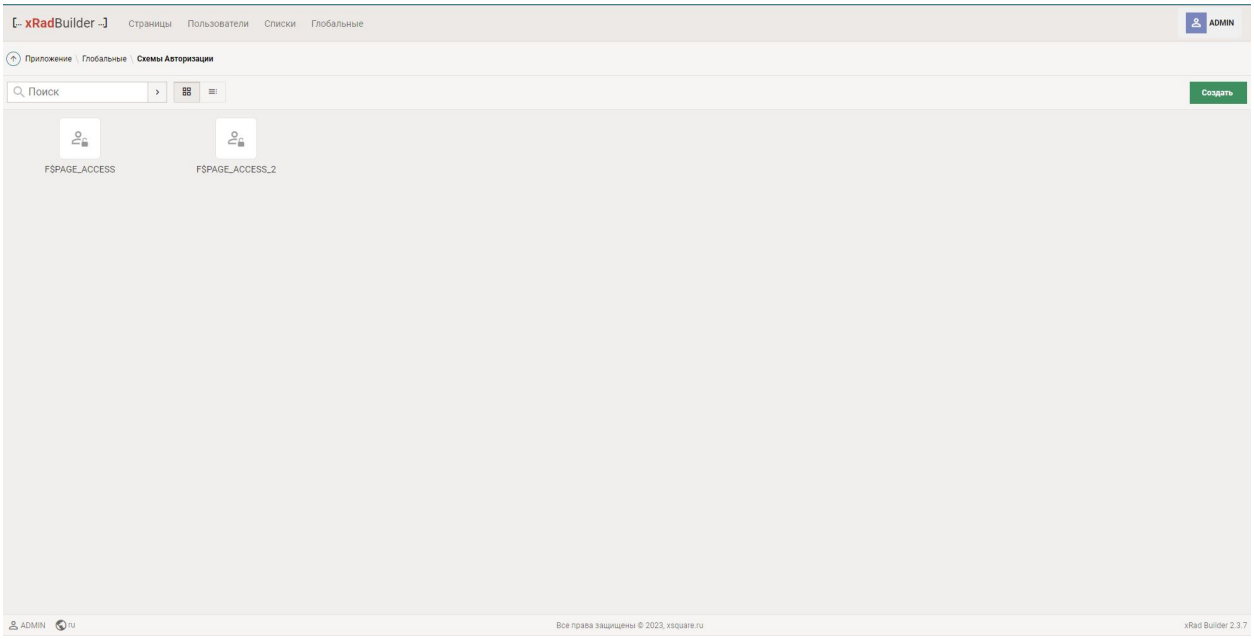
Если авторизация проходит успешно, то пользователю отображается нужная страница. Если при авторизации происходит ошибка, то на экран выводится сообщение об ошибке.

9.2.2 Создание схемы авторизации

Страница со схемами авторизации располагается во вкладке “Глобальные”:

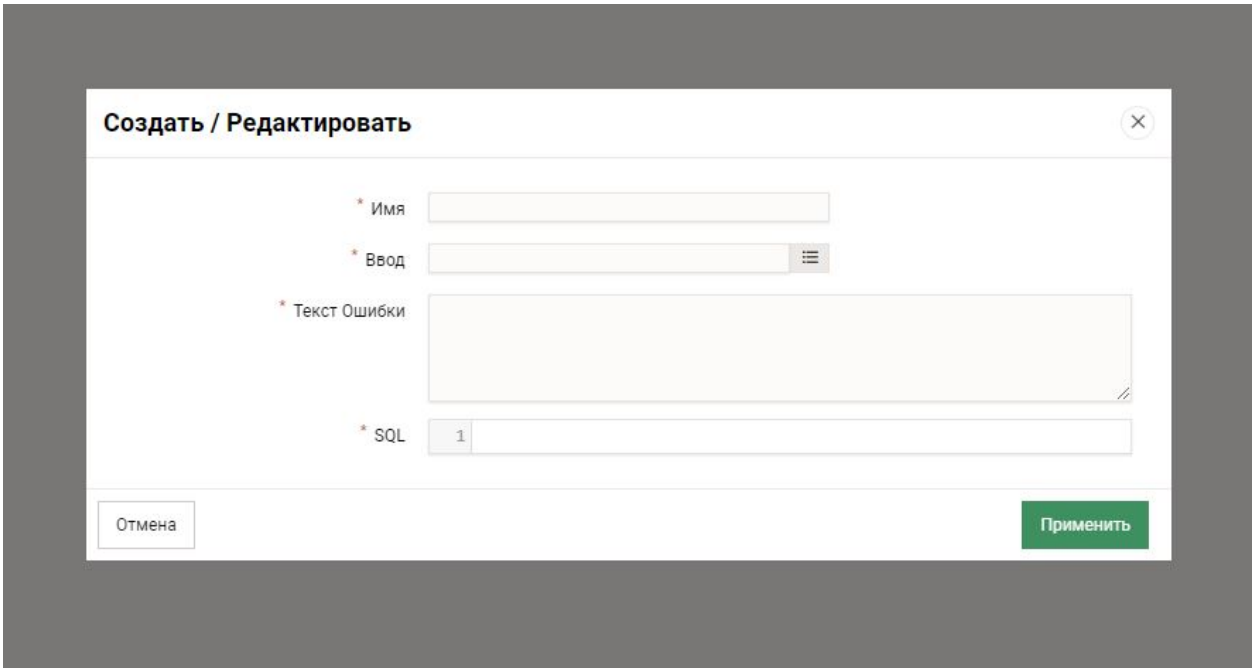


При переходе на страницу “Схемы Авторизации” будет отображен перечень всех схем авторизации, доступных в приложении, и кнопка для создания новой схемы:



Для перехода на страницу необходимо авторизоваться от лица аккаунта с ролью доступа выше VIEW.

Создание новой схемы авторизации осуществляется по нажатию кнопки “Создать”. Откроется форма создания схемы авторизации:



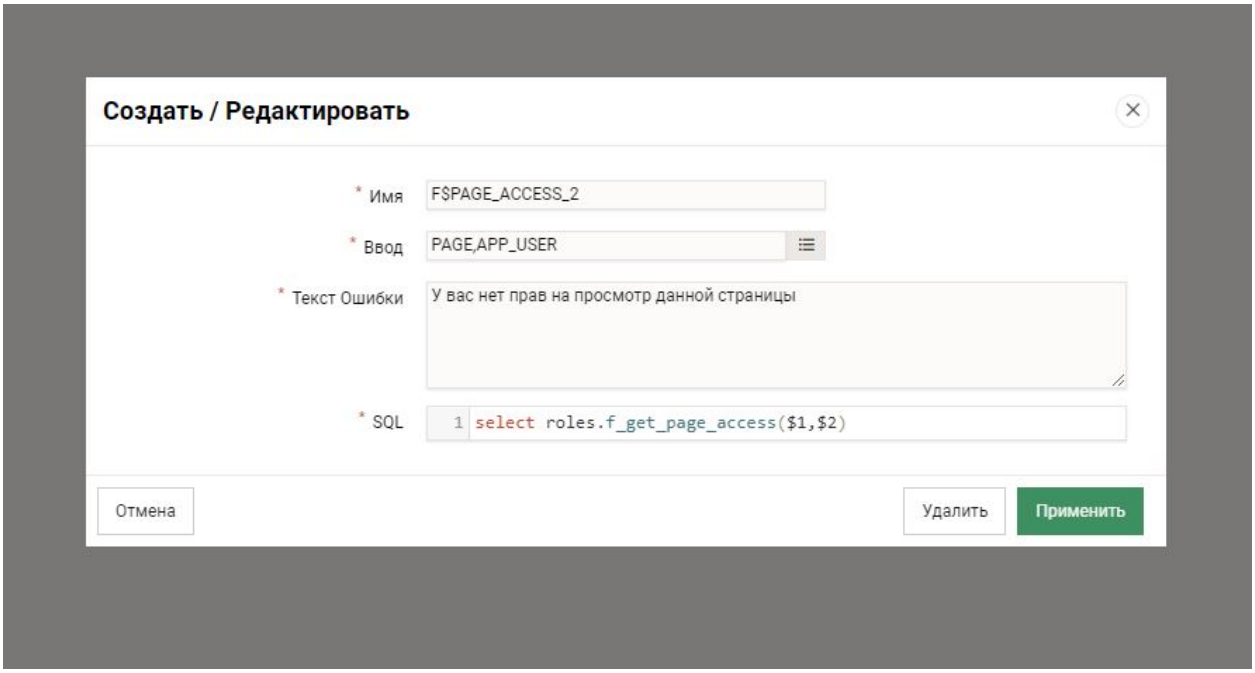
Атрибуты:

- Имя - **обязательный** атрибут, определяющий имя создаваемой схемы авторизации;
- Ввод - **обязательный** атрибут, определяющий список входных параметров для авторизации;
- Текст Ошибки - **обязательный** атрибут, определяющий текст ошибки авторизации;
- SQL - **обязательный** атрибут, определяющий SQL-запрос для авторизации.

После подтверждения введенных данных схема авторизации будет занесена в базу данных XRAD и доступна для просмотра, редактирования и удаления.

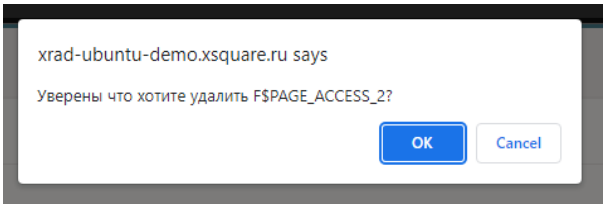
9.2.3 Редактирование схемы авторизации

Для редактирования схемы авторизации выберите нужную вам схему. После нажатия на схему откроется форма, аналогичная форме создания новой схемы авторизации (см. выше “Создание схемы авторизации”), с информацией о выбранной схеме:



Отредактируйте необходимые вам атрибуты и сохраните их новые значения, нажав кнопку “Применить”.

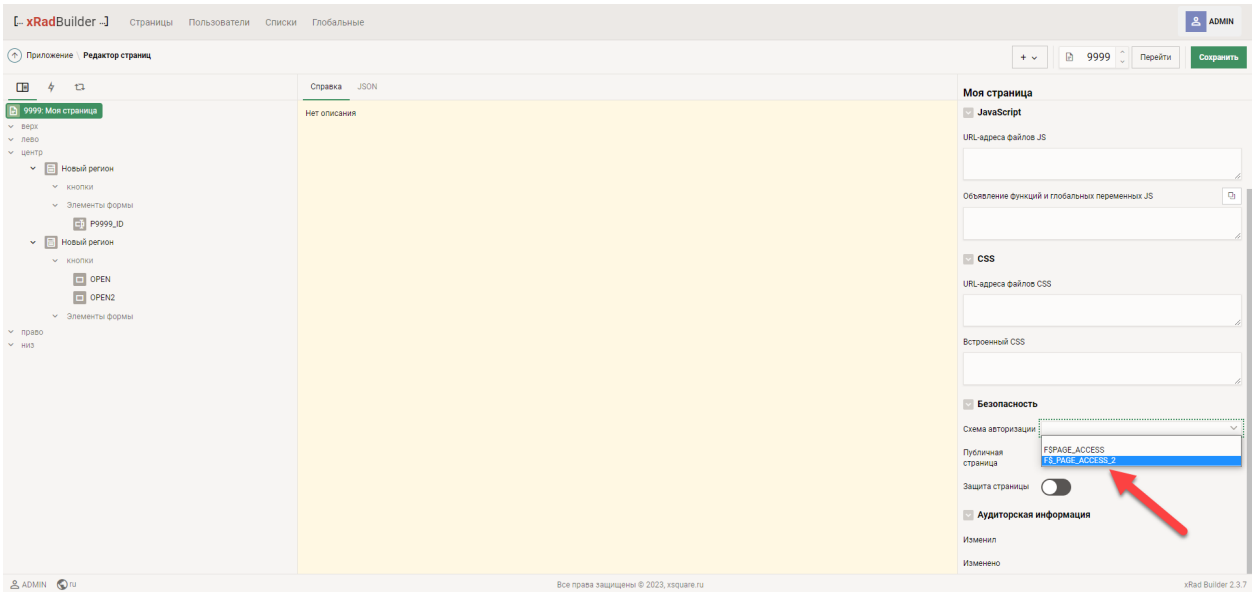
Для удаления схемы нажмите кнопку “Удалить” и подтвердите действие в появившемся диалоговом окне браузера:



9.2.4 Выбор схемы авторизации

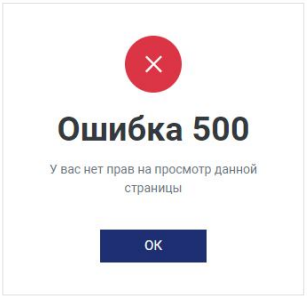
Определить доступность страницы конкретному пользователю можно в *редактор страниц* через свойство “Безопасность - Схема авторизации”.

В выпадающем списке можно увидеть все доступные схемы авторизации.



Выбранная схема авторизации будет применена к странице после сохранения изменений через кнопку “Сохранить”.

Сообщение при ошибке авторизации, если у пользователя нет прав на просмотр страницы, будет выглядеть следующим образом:



9.3 Контрольные суммы

9.3.1 Общее описание

Контрольная сумма используется в случае, когда необходимо обеспечить защиту от смены параметров страницы в URL.

Контрольная сумма формируется для каждой ссылки для конкретного пользователя. Пользователь не может поменять параметры в URL, не вызвав ошибки проверки контрольной суммы.

Если страница защищена с помощью контрольной суммы, в URL появляется дополнительный GET-параметр вида:

```
&cs=4825bc2e8de876595d5a36bccc89891b415e2575f45833cc5df786c000e24308
```

Контрольная сумма состоит из:

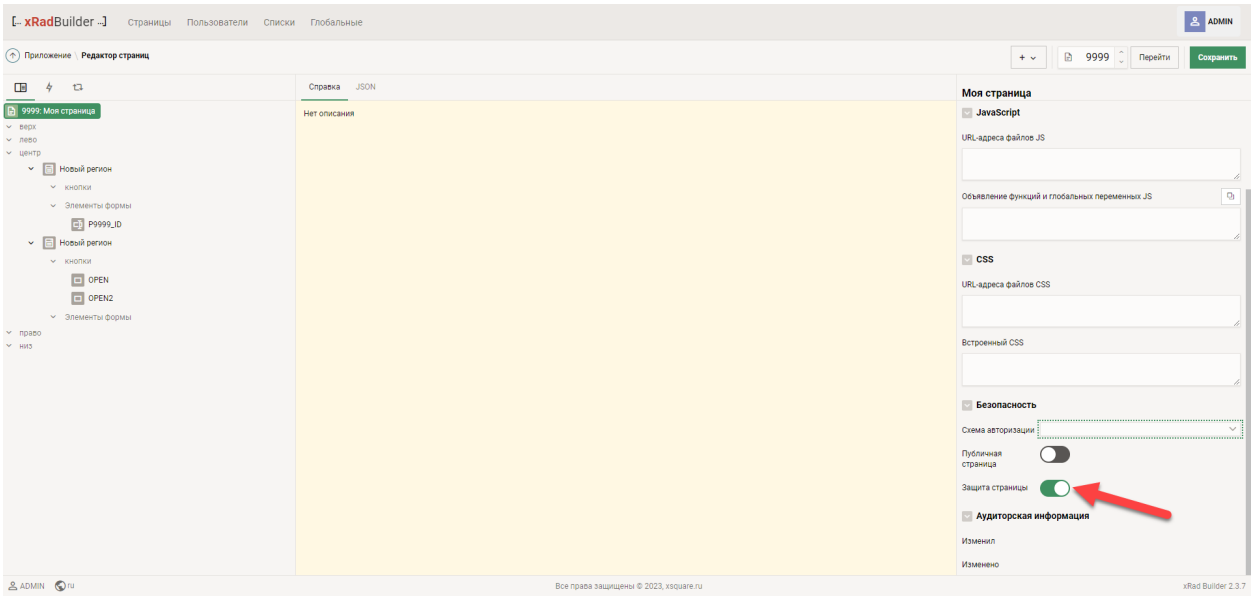
- набора параметров страницы;
- специальных наборов символов, которые называются salt (соль). Соль формируется для каждой сессии пользователя.

Если пользователь вручную укажет ссылку на страницу, которая защищена контрольной суммой, система распознает эту ссылку и добавит к ней контрольную сумму.

> Защита страницы через контрольную сумму настраивается отдельно для каждой страницы.

9.3.2 Настройка защиты страницы

Настроить защиту страницы с помощью контрольной суммы можно в *редактор страниц* через свойство “Безопасность - Защита страницы”.



Если свойство “Защита страницы” не установлено, то страница не будет защищена с помощью контрольной суммы.

Если свойство “Защита страницы” установлено, то будет включена защита страницы через контрольную сумму.

Данное свойство будет применено к странице после сохранения изменений через кнопку “Сохранить”.

ГЛАВА 10

Настройки приложения

Настройки уровня приложения являются обязательными для заполнения и делятся на три группы: общие, списки и сессия. Ниже будет рассмотрен каждый раздел.

Чтобы настроить приложение, в билдере перейдите в меню “Глобальные” - “Настройки приложения”.

10.1 Общее

Основные настройки приложения.

Параметр	Тип	Описание
Имя приложения	Текст	Название приложения. Выводится в заголовке страницы.
Домашняя страница	Текст / Всплывающее окно	Основная страница, на которую попадает авторизованный пользователь. Здесь указывается номер, страницу можно выбрать также во всплывающем окне. Подробнее о страницах в п. *управление страницами*
Страница входа	Текст / Всплывающее окно	Основная публичная страница, на которую попадает неавторизованный пользователь. Как правило, страница авторизации. Здесь указывается номер, страницу можно выбрать также во всплывающем окне. Подробнее о страницах в п. *управление страницами*

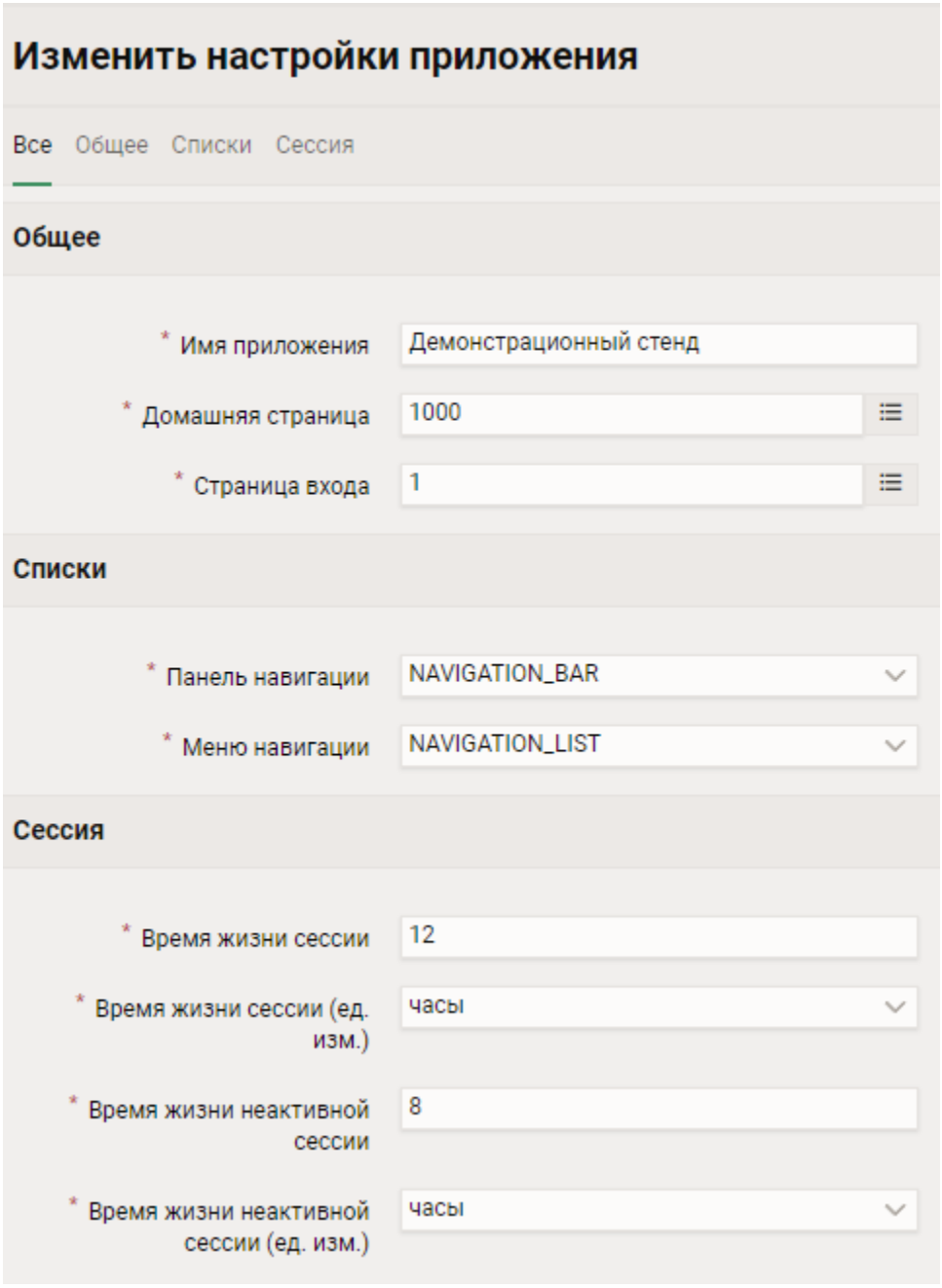


Рис. 1: Рис. 1 - настройки приложения

10.2 Списки

Здесь определяются списки уровня приложения.

Параметр	Тип	Описание
Панель навигации	Список	Определяет список для меню в заголовке страницы. Выберите из перечня заранее созданных списков в соответствующем разделе. Подробнее о списках см. п. *Управление списками*
Меню навигации	Список	Определяет список для основного меню в левом блоке страницы. Выберите из перечня заранее созданных списков в соответствующем разделе. Подробнее о списках см. п. *Управление списками*

10.3 Сессия

Параметры пользовательских сессий.

Параметр	Тип	Описание
Время жизни сессии	Число	Определяет время жизни сессии. Пройдёт именно столько единиц времени, прежде чем авторизованный пользователь будет принудительно отключен.
Время жизни сессии (ед. изм.)	Список	Определяет единицу измерения для указанного в параметре “Время жизни сессии” числа. На выбор 4 варианта: секунды, минуты, часы и дни.
Время жизни неактивной сессии	Число	Определяет время жизни сессии пользователя, не производящего никаких действий в приложении. Пройдёт именно столько единиц времени, прежде чем авторизованный пользователь будет принудительно отключен.
Время жизни неактивной сессии (ед. изм.)	Список	Определяет единицу измерения для указанного в параметре “Время жизни неактивной сессии” числа. На выбор 4 варианта: секунды, минуты, часы и дни.

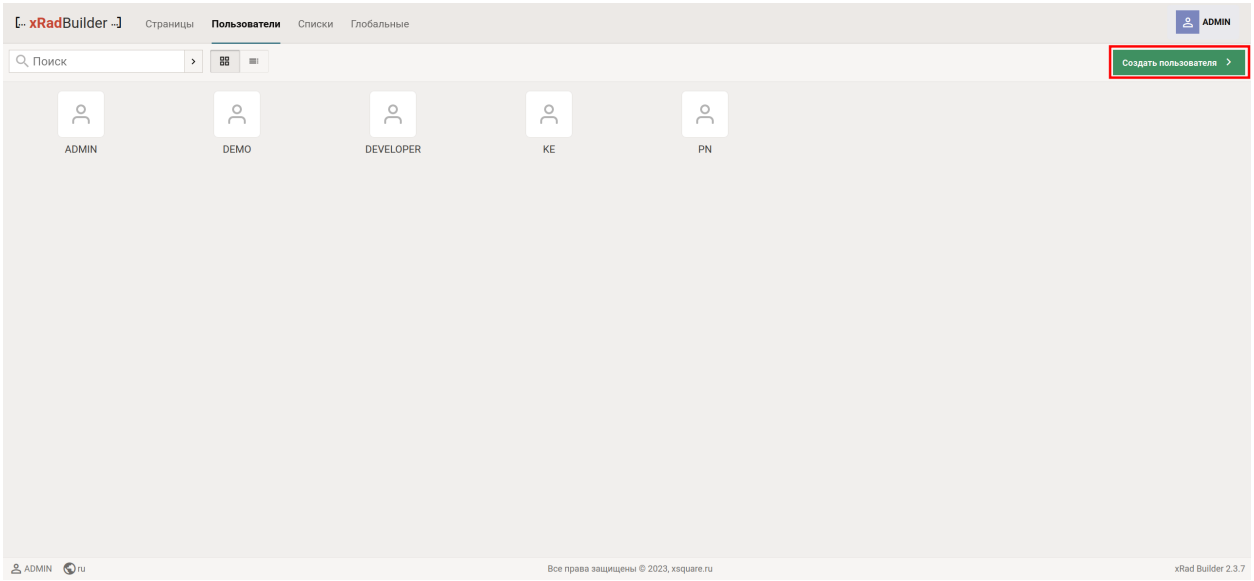
ГЛАВА 11

Управление пользователями

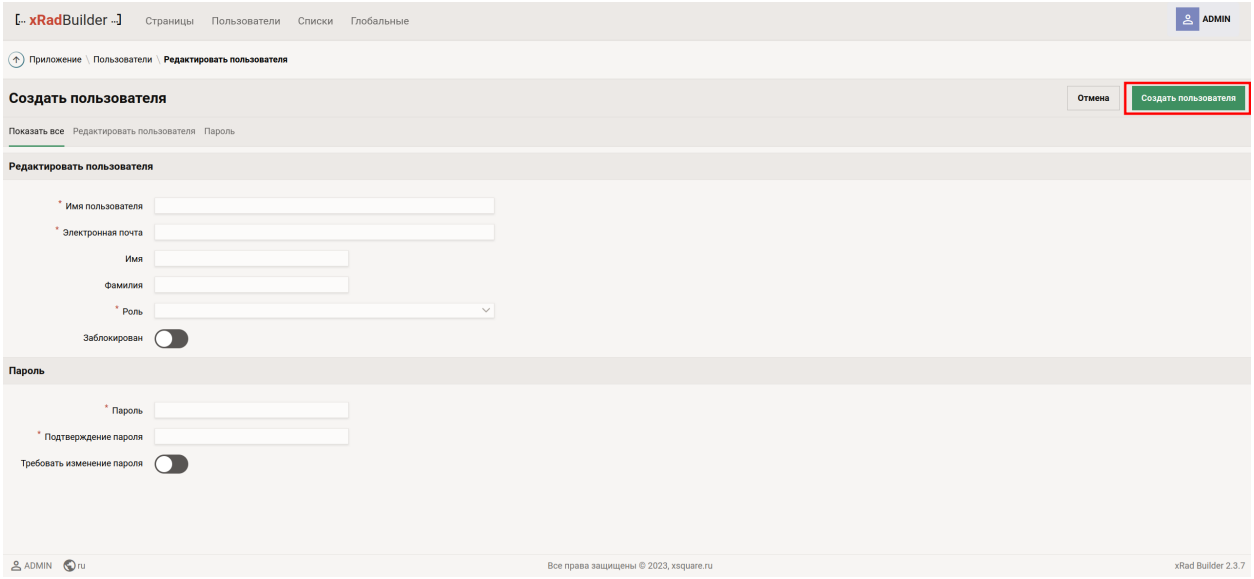
XRAD предоставляет возможность управлять пользователями напрямую из XRAD Builder. На данный момент существует три основные роли, с помощью которых можно разграничить зоны доступа для пользователей. Процессы создания и редактирования пользователей довольны минималистичны, но имеют весь необходимый функционал. В этом разделе будет подробно разобрано как Создавать и Редактировать пользователей, а также будет информация по возможным ролям пользователей.

11.1 Создание пользователя

Для того чтобы создать нового пользователя, необходимо войти в аккаунт пользователя с ролью ADMIN. Зайдите в Пользователи и нажмите Создать пользователя.



После этого открывается раздел в котором можно править параметры пользователя



11.1.1 Редактировать пользователя

- Имя пользователя - имя которое будет отображаться при выборе пользователя, используетя в качестве логина, обязательное поле
- Электронная почта - адрес электронной почты пользователя, обязательное поле
- Имя
- Фамилия
- Роль - одна из ролей, которая будет присвоена пользователю. *Про роли пользователей*
- Заблокирован - если включено, пользователь не сможет войти в свой аккаунт

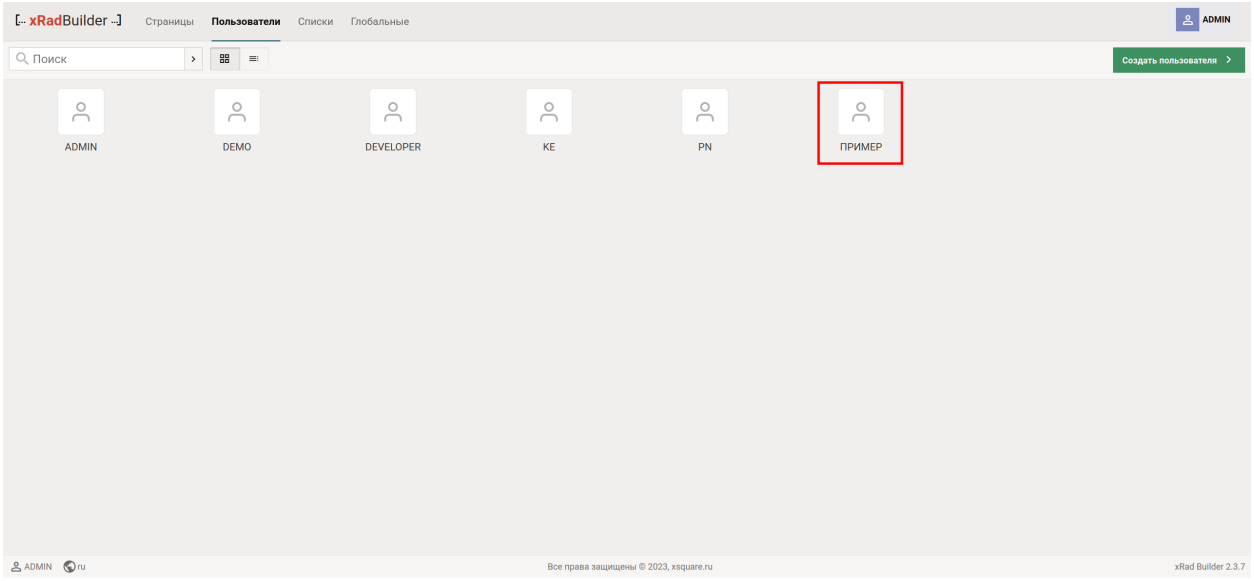
11.1.2 Пароль

- Пароль
- Подтверждение пароля - поле должно совпадать с полем **Пароль**
- Требовать изменение пароля (в работе)

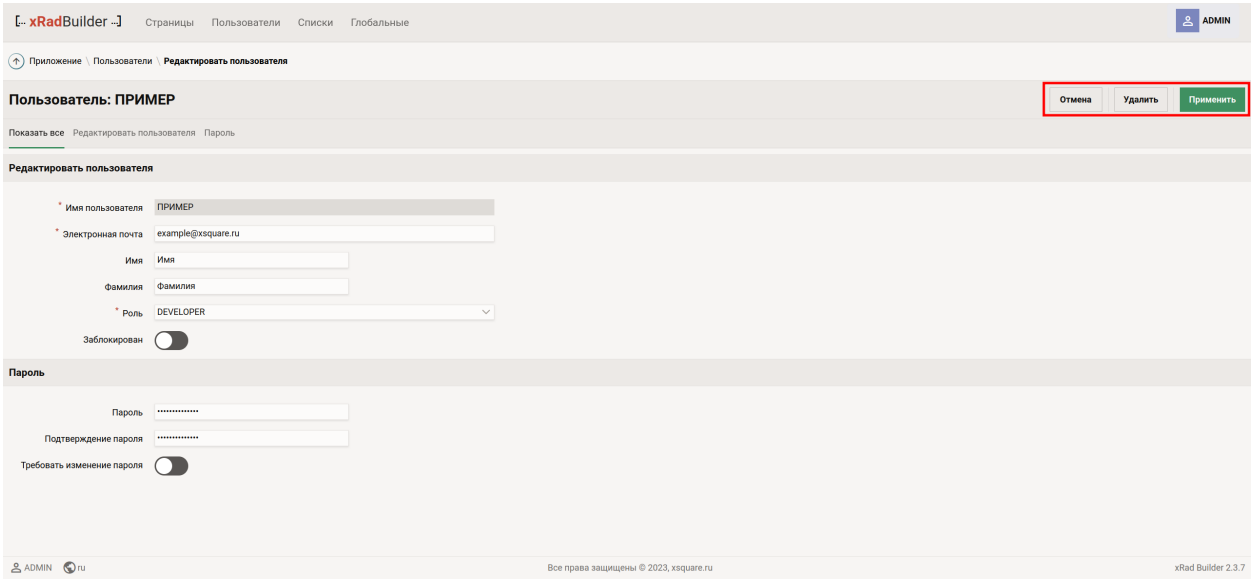
Для того чтобы изменения сохранились, необходимо нажать **Создать пользователя**

11.2 Редактирование пользователя

Редактирование пользователя очень похоже на его создание. Только пользователь с ролью `ADMIN` имеет право редактировать пользователей.



Допусти существует пользователь **ПРИМЕР**, чтобы его редактировать нужно просто нажать на соответсвующую иконку. Откроется данная вкладка



Поля для изменения полностью совпадают с полями в Создание пользователя, для дополнительной информации посетите соответсвующую страницу *Создание пользователя*

Единственные отличия: **Имя пользователя** поменять нельзя, для продолжения три опции:

- Отмена - отменить редактирование без сохранения изменений
- Удалить - удалить пользователя
- Применить - сохранить изменения для данного пользователя

11.3 Про роли пользователей

На данный момент существует 3 возможных роли: `ADMIN`, `DEVELOPER`, `VIEW`.

11.3.1 VIEW

Роль с наименьшим количеством прав. `VIEW` может просматривать **Страницы** и их содержимое, а также **Списки** и **Глобальные** (но не их содержимое!).

Не может просматривать **Пользователей**.Не имеет права создавать **Страницы**, не имеет права изменять настройки своего пользователя.

Данная роль отлично подходит для пользователей которые хотят познакомиться с **XRAD**.

11.3.2 DEVELOPER

Роль разработчика. `DEVELOPER` может просматривать и изменять **Страницы**, создавать и удалять их. Имеет право редактировать **Списки** и **Глобальные**. Имеет право просматривать **Пользователей**, но не имеет права изменять их (в том числе и своего пользователя).

Данная роль подходит для разработчиков, так как пользователем с данной ролью доступны все интсрументы кроме управления **Пользователями**.

11.3.3 ADMIN

Роль администратора. `ADMIN` имеет полные права, поэтому следует с особой осторожностью выдавать данную роль. Пользователь с данной ролью может редактировать, создавать и удалять других пользователей. Имеет право создать другого пользователя с ролью `ADMIN`.

Данную роль следует строго выдавать администратору, разработчикам должно хватать прав роли `DEVELOPER`.

ГЛАВА 12

Стили и темы

В PGHS предусмотрено управление стандартными стилями через кастомные CSS свойства. Для их использования достаточно создать css файл и поместить в `<head></head>` или разместить в `index.html` через тег `<style></style>`. Кастомные свойства PGHS применяются к псевдо-классу `:root`.

Перечень кастомных CSS свойств PGHS и пример их использования:

```
:root {
/* Лайаут */
--primary-color: #1e2f74; /* Основной цвет.  */
--header-color: #1e2f74; /* Цвет хедера. Наследуется от --primary-color */
--app-bg-color: white; /* Фоновый цвет страниц. */

/* Сайдбар (навигационное меню) */
--sidebar-bg: #f2f5f7; /* Фоновый цвет сайдбара. */
--sidebar-border: 1px solid #e5eaef; /* Бордер сайдбара. */
--nav-color: #393e42; /* Цвет текста элементов меню */
--nav-color-hover: #687179; /* Цвет текста элементов меню при наведении */
--nav-color-active: #008bcd; /* Цвет текста активных элементов меню */

/* Элементы формы*/
--disabled-items-color: #e9ecef; /* Фоновый цвет полей формы в состоянии :disabled. */

/* Кнопки */
--btn-color: #f8f9fa; /* Фоновый цвет стандартной кнопки. */
--btn-primary-color: #1e2f74; /* Фоновый цвет кнопки primary. Наследуется от --
↪primary-color */
--btn-secondary-color: #1e2f74; /* Фоновый цвет кнопки secondary. Наследуется от --
↪secondary-color */
--btn-success-color: #1e2f74; /* Фоновый цвет кнопки secondary. Наследуется от --
↪success-color */
--btn-danger-color: #1e2f74; /* Фоновый цвет кнопки secondary. Наследуется от --
↪danger-color */
--btn-warning-color: #1e2f74; /* Фоновый цвет кнопки warning. Наследуется от --
↪warning-color */
```

(continues on next page)

(продолжение с предыдущей страницы)

```
--btn-info-color: #1e2f74; /* Фоновый цвет кнопки info. Наследуется от --info-color */
--btn-hot-color: #1e2f74; /* Фоновый цвет кнопки hot. Наследуется от --hot-color */

/* Регионы */
--region-header-color: #f8f9fa; /* Фоновый цвет хедера региона. */
--region-border-color: #e5eaef; /* Цвет бордера региона. */

/* Ссылки */
--link-color: #1e2f74; /* Цвет ссылки. Наследуется от --primary-color */

/* Индикатор состояния загрузки */
--loader-color: #1e2f74; /* Цвет индикатора состояния загрузки. Наследуется от --
↪primary-color */

/* Таблица (Report) */
--table-color: #2e3031; /* Цвет текста таблицы */
--table-border-color: #f0f2f5; /* Цвет бордера таблицы */
--table-accent-bg: rgba(0,0,0,0.02); /* Фоновый цвет акцента строк (для всех нечетных␣
↪строк таблицы) */
--table-hover-color: #2e3031; /* Цвет текста при наведении. Наследуется от --table-
↪color */
--table-hover-bg: #d8e7ee; /* Фоновый цвет при наведении. */
}
```

ГЛАВА 13

Жизненный цикл

13.1 Общие сведения

Описание процессов, обеспечивающих поддержание жизненного цикла программного обеспечения XSQUARE-RAD, в том числе устранение неисправностей, выявленных в ходе эксплуатации программного обеспечения, совершенствование программного обеспечения, а также информация о персонале, необходимом для такой поддержки.

Таблица 1: Информация

Программа	Сервер приложений XSQUARE-RAD
Разработчик	ООО «Хи-Квадрат»
Пользователь	Юридическое лицо, использующее Программу согласно договора с Разработчиком
Сайт	https://lcdp.xsquare.ru

13.2 Поддержание жизненного цикла Программы

Жизненный цикл Программы включает в себя следующие этапы:

- Проектирование и разработка Программы, осуществляемые Разработчиком;
- Тестирование и выявление неисправностей в работе Программы Разработчиком;
- Установка, использование и обновление Программы Пользователем согласно лицензионному соглашению с Разработчиком;
- Модернизация программы Разработчиком согласно собственному плану доработок и улучшений, а также по заявкам Пользователя;
- Осуществление технической поддержки Пользователя Разработчиком по вопросам установки, интеграции и эксплуатации Программы;
- Выпуск Разработчиком обновленных сборок модернизированной Программы.

Разработчик регулирует проведение всех этапов жизненного цикла программы за исключением процессов установки, интеграции и использования Программы Пользователем.

13.3 Устранение неисправностей, выявленных в ходе эксплуатации Программы

Неисправности, выявленные в ходе эксплуатации Программы могут быть устранены следующими способами:

- Внесение исправлений в код Программы Разработчиком согласно своей дорожной карте разработки Программы;
- Внесение исправлений в код Программы на основе обращения Пользователя;

Пользователь может сформировать следующие запросы:

- Отчёт об инциденте с приложением информации об условиях возникновения бага с использованием графической информации, лог-файлов, информации о программном окружении и номерах версий используемого программного обеспечения, включая версию и редакцию Программы. Запрос также должен содержать информацию об ожидаемом и фактическом поведении Программы и любую другую информацию, которая поможет диагностировать и устранить неисправность Программы Разработчиком;
- Запрос на улучшение Программы в целях изменения её поведения для достижения нужных результатов в решениях Пользователя;
- Запрос на предоставление дополнительной информации о функционировании и возможностях Программы.

Запросы могут быть отправлены Пользователем только с помощью трекинговой системы Разработчика - https://tracker.xsquare.ru. Доступ к трекенговой системе предоставляется по факту приобретения Программы.

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваиваится уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Разработчик информирует Пользователя о новом функционале Программы, либо о добавлении задачи по развитию в план разработки.

Разработчик оставляет за собой право запросить дополнительную информацию от Пользователя, которая может быть полезна для устранения неисправностей в работе Программы.

При непредоставлении либо недостаточном предоставлении Пользователем информации, требуемой Разработчиком, последний вправе приостановить внесение требуемых изменений в код Программы.

13.4 Совершенствование Программы

Программа непрерывно улучшается и модернизируется, выпускаются регулярные обновления, публикуются информационные материалы на Сайте Программы, Пользователи информируются об изменениях в Программе.

Пользователь может инициировать запрос на изменение или улучшение работы Программы с помощью формирования запроса к Разработчику.

Запросы могут быть отправлены Пользователем с помощью трекинговой системы https://tracker.xsquare.ru

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваивается уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Разработчик информирует Пользователя о внесенных изменениях в код Программы, либо о добавлении задачи по модернизации в план разработки.

Разработчик оставляет за собой право запросить дополнительную информацию от Пользователя, которая может быть полезна для улучшения работы Программы.

При непредоставлении либо недостаточном предоставлении Пользователем информации, требуемой Разработчиком, последний вправе приостановить внесение требуемых изменений в код Программы

13.5 Техническая поддержка Программы

Техническая поддержка Программы осуществляется с помощью формирования запросов Разработчику.

Запросы могут быть отправлены Пользователем с помощью трекинговой системы https://tracker.xsquare.ru.

Разработчик принимает и фиксирует все запросы Пользователя. Каждому запросу присваивается уникальный номер, который позволяет отследить историю общения Пользователя и Разработчика в дальнейшем.

Техническая поддержка Пользователя включает в себя:

- Помощь в установке Программы;
- Помощь в установке базовых общесистемных компонентов (операционной системы, HTTP Сервер, сервер баз данных и)
- Помощь в интеграции Программы в существующие решения Пользователя;
- Помощь в устранении неисправностей, возникающих в работе Программы;
- Консультации по функционированию Программы;
- Сбор информации о некорректной работе Программы для последующего выпуска модернизации Программы согласно плану доработок;

Информирование Пользователя об обновлениях Программы

13.6 Информация о персонале, необходимом для обеспечения поддержки

Персонал, который будет осуществлять поддержку Программы со стороны Пользователя, должен обладать:

- Базовыми навыками администрирование операционных систем семейства Unix
- Базовыми навыками администрирование СУБД PostgresPro и/или PostgreSQL
- Пользователи Программы должны обладать навыками работы с персональным компьютером и веб браузером.

В случае возникновения вопросов у персонала, им следует обратиться к Разработчику за получением технической поддержки

ГЛАВА 14

Справочник jsAPI

14.1 jsAPI

Объект *jsAPI* служит инструментом взаимодействия клинтской части приложения с сервером и фронтендом *pghs*, направлен на упрощение разработки фронтенда, предоставляет готовые инструменты работы с *DOM*, обёртки для стандартных javascript-функций и многое другое.

jsAPI установлен по умолчанию и уже находится в глобальной области видимости *pghs* (*window*). Для просмотра всех методов и объектов достаточно вызвать объект *jsAPI* из отладчика (*jsAPI*, *console.log(jsAPI)*, *console.dir(jsAPI)*)

Обновлено: 28.04.2023

Страница

- submit()
- process()
- reload()
- reloadWindow()
- redirect()
- getCurrentPage()
- notification

Модальное окно

- modal
- confirmModal()

Регион

- refresh()
- changeTab()

- component()

Элемент формы (item)

- setItem()
- getItem()
- updateItem()
- hideItem()
- showItem()
- refreshList()

Работа с DOM

- dom.hide()
- dom.show()
- dom.focus()
- dom.blur()
- dom.setAttribute()

Специальные

- debug
- logout()
- clearLocalStorage()

14.1.1 jsAPI: changeTab()

Переключает активный таб в регионе с типом «Tabs»

Синтаксис

```
jsAPI.changeTab(id, tab);
```

Параметры

id String

id региона с типом «Tabs»

tab Integer

Порядковый номер вкладки (начиная с 1)

Примечание

Annotation

Пример

```
jsAPI.changeTab('MY_TABS', 2);
```

14.1.2 jsAPI: clearLocalStorage()

Удаляет все записи в localStorage приложения. Является обёрткой нативного метода *window.localStorage.clear();*

Синтаксис

```
jsAPI.clearLocalStorage();
```

Параметры

Данная функция не принимает аргументов.

Пример

```
jsAPI.clearLocalStorage();
```

14.1.3 jsAPI: component()

jsAPI.component - интерфейс для работы с данными регионов в рамках заданного контекста

Синтаксис

```
jsAPI.component(id).method();
```

Параметры

id [String] - id региона

Методы

refresh()

```
jsAPI.component(id).resfresh();
```

Пример:

```
jsAPI.component('REPORT').resfresh();
```

Реализован для следующих регионов:

- *Calendar*
- *Report*

Обновляет регион с учётом указанных параметров (items). Например, с указанием имени поля для типа региона «календарь»:

```
{
  "items": [
    "P50_FIO"
  ]
}
```

значение поля P50_FIO добавится в тело запроса /callAction помимо основных параметров

Тело запроса:

```
{
    "action": "CALENDAR",
    "data": {
        "page": 50,
        "id": "CLNDR",
        "items": {
            "G01": "20230428030000",
            "G02": "20230429030000",
            "P50_FIO": "Иванов Иван Иванович"
        }
    }
}
```

14.1.4 jsAPI: confirmModal()

Отображает окно для подтверждения какого-либо действия.

Синтаксис

```
jsAPI.confirmModal(options, callbacks);
```

Параметры

options Object

Содержит параметры:

- title String - Заголовок окна;
- text String - Текст, отобажаемый в диалоговом окне;

callbacks Принимает функции:

- onAccept Function - Выполняется при нажатии кнопки «Да»
- onDecline Function - Выполняется при нажатии кнопки «Нет» и стандартном закрытии

Примечание

Данный метод является частным случаем вызова *jsAPI.modal.open*:

```
jsAPI.modal.open({
    title: 'Заголовок',
    text: 'Текст',
    width: 420,
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    buttons: [{
        text: 'Да',
        action: 'accept',
        class: 'btn-primary'
    }, {
        text: 'Нет',
        action: 'decline',
        class: 'btn-secondary'
    }, ]
}, {
    onClose: function (e) {
        // Выполнится при закрытии
    },
    onDecline: function (e) {
        // Выполнится при нажатии кнопки "Нет" и стандартном закрытии модального окна
    },
    onAccept: function (e) {
        // Выполнится при нажатии кнопки "Да
    }
})
```

Пример

```
jsAPI.confirmModal({
    title: 'Вы уверены?',
    text: 'При закрытии данные не сохранятся',
}, {
    onDecline: function (e) {
        // Выполнится при нажатии кнопки "Нет" и стандартном закрытии модального окна
    },
    onAccept: function (e) {
        // Выполнится при нажатии кнопки "Да
    }
})
```

14.1.5 jsAPI: debug

Включение и отключение различных уровней отображения отладочной информации. Включает методы *enable* и *disable*.

Активировать debug

```
jsAPI.debug.enable(type);
```

Деактивировать debug

```
jsAPI.debug.disable(type);
```

Параметры

type [String]

Имя уровня отладки.

Доступные уровни отладки

- *CRITICAL*
- *ERROR*
- *WARNING*
- *INFO*
- *DEBUG*

Примеры

Активировать отладку с типом CRITICAL:

```
jsAPI.debug.enable("CRITICAL");
```

Деативировать отладку с типом CRITICAL:

```
jsAPI.debug.disable("CRITICAL");
```

14.1.6 jsAPI: dom.blur()

Удаляет фокус клавиатуры с текущего элемента. Обёртка для нативного метода *document.activeElement.blur()*

Синтаксис

```
jsAPI.dom.blur();
```

Параметры

Данная функция не принимает аргументов.

Примеры

```
jsAPI.dom.blur();
```

14.1.7 jsAPI: dom.focus()

Устанавливает фокус на указанный элемент, если он может быть сфокусирован. Обёртка для нативного метода *el.focus()*

Синтаксис

```
jsAPI.dom.focus(selector);
```

Параметры

selector [String]

Селектор элемента

Примеры

```
jsAPI.dom.focus('#P1000_MY_ITEM');
```

14.1.8 jsAPI: dom.hide()

Прячет элемент с указанным css селектором путём добавления css класса js-api-hidden

Синтаксис

```
jsAPI.dom.hide(selector);
```

Параметры

selector [String]

Селектор элемента

Примеры

Example 1:

```
jsAPI.hide('#P1000_MY_ITEM');
```

14.1.9 jsAPI: dom.setAttribute()

Добавляет и изменяет атрибуты у элемента с указанным селектором

Синтаксис

```
jsAPI.dom.setAttribute(selector, attrs);
```

Параметры

selector [String]

Селектор элемента

attrs [Object]

Атрибуты

Примеры

```
jsAPI.setAttribute('#P1000_MY_ITEM',{
    disabled: true,
    'data-custom-attr': 'My custom value'
});
```

14.1.10 jsAPI: dom.show()

Отображает элемент с указанным css селектором путём удаления css класса js-api-hidden

Синтаксис

```
jsAPI.dom.show(selector);
```

Параметры

selector [String]

Селектор элемента

Примеры

Example 1:

```
jsAPI.show('#P1000_MY_ITEM');
```

14.1.11 jsAPI: getCurrentPage()

Отдаёт информацию о текущей странице в виде:

```
{
    "pageId": "1000",
    "ELEMENT_STATES": {
        ...
    },
    "CLIENT_VALIDATION": {
        ...
    },
    "VALIDATION": {
        ...
    },
    "items": {
        ...
    },
    "page": {
        ...
    }
}
```

Синтаксис

```
jsAPI.getCurrentPage();
```

Параметры

Данная функция не принимает аргументов.

Примеры

Получить id страницы:

```
jsAPI.getCurrentPage().pageId;
```

Получить значения item-а:

```
jsAPI.getCurrentPage().items['P1000_MY_ITEM'];
```

14.1.12 jsAPI: getItem()

Возвращает значение элемента формы

Синтаксис

```
jsAPI.getItem(id);
```

Параметры

id [String] - id элемента формы. НЕ является селектором (указывать без „#“)

Пример

```
jsAPI.getItem('P1000_ITEM');
```

14.1.13 jsAPI: hideItem()

Удаляет элемент формы (item) из DOM и привязанную к нему колонку

Синтаксис

```
jsAPI.hideItem(id);
```

Параметры

id [String] - id элемента формы. НЕ является селектором (указывать без „#“)

Пример

```
jsAPI.hideItem('P1000_ITEM');
```

14.1.14 jsAPI: logout()

Совершает выход из приложения (логаут), вызывая метод {API_URL}/logout

Синтаксис

```
jsAPI.logout();
```

Параметры

Данная функция не принимает аргументов.

Пример

```
jsAPI.logout();
```

14.1.15 jsAPI: modal

jsAPI.modal - модуль для управления модальными окнами приложения.

Методы

```
jsAPI.modal.open(options, callbacks);
```

Параметры

options Object

Опции отображения диалогового окна

Содержит параметры:

- title String - Заголовок окна
- page String - URL - страницы, которая будет отображаться в диалоговом окне
- width Number - Ширина окна
- text String - Текст, отображаемый в контентной части диалогового окна
- buttons Array - Кнопки, отображаемые в нижней части диалогового окна
- selector String - Селектор элемента для привязки к динамическому действию (Dynamic Action, DA)

callbacks Object

Принимает функции:

- onClose Function - Выполняется при закрытии окна *jsAPI.modal.close();*, *jsAPI.modal.accept();*
- onAccept Function - Выполняется при закрытии окна с ипользовании метода *jsAPI.modal.accept();*
- onDecline Function - Выполняется при закрытии окна с ипользовании метода *jsAPI.modal.close();*

```
jsAPI.modal.close();
```

Закрывает модальное окно, вызывает *onClose*, *onDecline* в *jsAPI.modal.open*

```
jsAPI.modal.accept();
```

Закрывает модальное окно, вызывает *onAccept*, в *jsAPI.modal.open*

Примеры

Вызов confirm модала

```
jsAPI.modal.open({
    title: 'Заголовок',
    text: 'Текст',
    width: 420,
    buttons: [{
        text: 'Да',
        action: 'accept',
        class: 'btn-primary'
    }, {
        text: 'Нет',
        action: 'decline',
        class: 'btn-secondary'
    }, ]
}, {
    onClose: function (e) {
        // Выполнится при закрытии
    },
    onDecline: function (e) {
        // Выполнится при нажатии кнопки "Нет" и стандартном закрытии модального окна
    },
    onAccept: function (e) {
        // Выполнится при нажатии кнопки "Да
    }
})
```

Стандартный вызов страницы в модальном окне

```
jsAPI.modal.open({
    page: `/1000/`,
    title: 'Заголовок',
});
```

Открытие страницы в модале с указанием GET параметров и селектором для привязки к динамическому действию

```
jsAPI.modal.open({
    page: `/1000/?P2000_ITEM=${jsAPI.getItem("P2000_ITEM")}&clear=1000`,
    title: 'Заголовок',
    selector: '#DA_ELEMENT_ID'
});
```

Открытие страницы в модале с указанием GET параметров, селектором для привязки к динамическому действию и коллбэками

```
jsAPI.modal.open({
    page: `/1000/?P2000_ITEM=${jsAPI.getItem("P2000_ITEM")}&clear=1000`,
    title: 'Заголовок',
    selector: '#DA_ELEMENT_ID'
}, {
    onClose: function (e) {
      console.log(e);
      // Выполнится при jsAPI.modal.close(), jsAPI.modal.accept()
    },
    onAccept: function (e) {
      console.log(e);
      // Выполнится при jsAPI.modal.accept()
    },
    onDecline: function (e) {
      console.log(e);
      // Выполнится при jsAPI.modal.close()
    }
});
```

14.1.16 jsAPI: notification

Включает методы *success*, *error* и *warning* которые выводят стандартную нотификацию приложения.

Уведомление об успешном действии

```
jsAPI.notification.success(message);//Выводит уведомление об успешном действии
```

Уведомление об ошибке

```
jsAPI.notification.error(message);//Выводит уведомление об ошибке
```

Предупреждение (warning)

```
jsAPI.notification.warning(message);//Выводит предупреждение (warning)
```

Параметры

message [String], [Array]

Одно сообщение если аргумент указан в виде строки, или массив сообщений если аргумент указан в виде массива строк (выводятся последовательно друг за другом)

Примеры

Сообщение об успешном действии:

```
jsAPI.notification.success('Успешное действие');
```

Сообщения об успешных действиях:

```
jsAPI.notification.success(['Успешное действие', 'Успешное действие 2']);
```

Сообщение об ошибке:

```
jsAPI.notification.error('Ошибка');
```

Сообщения об ошибках:

```
jsAPI.notification.error(['Ошибка', 'Ошибка 2']);
```

Предупреждение (warning)

```
jsAPI.notification.warning('Внимание!');//Выводит предупреждение (warning)
```

Предупреждения (warnings)

```
jsAPI.notification.warning(['Предупреждение', 'Предупреждение 2']);//Выводит␣
↪предупреждения (warnings)
```

14.1.17 jsAPI: process()

Метод jsAPI.process выполняет xhr запрос {API_URL}/callAction со следующими параметрами:

```
{
  action: "PROCESS",
  data: {
    page: pageID,
    request: requestName,
    items: items
  }
}
```

pageID (*data.page*)

ID текущей страницы. Определяется автоматически.

requestName (*data.request*)

Имя запроса. Определяется аргументом *requestName*

items (*data.items*)

Значения *items*. Определяется аргументом *items*

Синтаксис

```
jsAPI.process(requestName, items, callbacks);
```

Параметры

requestName String

Имя запроса

items Object

Значения items, отправляемые в теле запроса {API_URL}/callAction. Допускается использовать пустой объект {} если не требутся отправки значений.

callbacks Object

Принимает функции: onSuccess Function, onError Function.

Примеры

Вызов без дополнительных параметров:

```
jsAPI.process("MY_PROCESS");
```

Вызов с items:

```
jsAPI.process("MY_PROCESS", {
    P1000_MY_ITEM_1: 'Value 1',
    P1000_MY_ITEM_2: 'Value 2',
});
```

Вызов с items и обработчиком успешного ответа:

```
jsAPI.process("MY_PROCESS", {
    P1000_MY_ITEM_1: 'Value 1',
    P1000_MY_ITEM_2: 'Value 2',
}, {
    onSuccess: function (res) {
        console.log(res);
    }
});
```

Вызов с items, обработчиками успешного ответа и ошибки:

```
jsAPI.process("MY_PROCESS", {
    P1000_MY_ITEM_1: 'Value 1',
    P1000_MY_ITEM_2: 'Value 2',
}, {
    onSuccess: function (res) {
        console.log(res);
    },
    onError: function (err) {
        console.error(err);
    }
});
```

14.1.18 jsAPI: redirect()

Осуществляет переход на указанную страницу приложения

Синтаксис

```
jsAPI.redirect(path, callbacks);
```

Параметры

path String

Относительный путь до страницы

callbacks

Принимает функцию: onSuccess Function

Примечание

Данный метод использует внутренний роутинг приложения и не является аналогом window.location.href. Изменение роута не предполагает перезагрузку страницы

Примеры

Стандартный вызов:

```
jsAPI.redirect('/content/5000?clear=5000');
```

Вызов с обработчиком:

```
jsAPI.redirect('/content/5000?clear=5000', {
    onSuccess: function (data) {
        console.log(data)
    }
});
```

14.1.19 jsAPI: refresh()

Метод jsAPI.refresh предназначен для обновления данных в регионах. Выполняет xhr запрос {API_URL}/callAction со следующими параметрами:

```
{
  action: "REFRESH",
  data: {
    data: data
  }
}
```

Синтаксис

```
jsAPI.refresh(data, callbacks);
```

Параметры

data Object

Данные для отправки в теле запроса

callbacks Object

Принимает функции: *onSuccess* Function, *onError* Function.

Примечание

Для обновления региона необходимо указать его id в параметре data:

```
data: {
  {
    id: 'MY_REGION_ID'
  }
}
```

Примеры

Стандартный вызов:

```
jsAPI.refresh({
    id: "PARAMS",
    page: "1000",
    items: {
        P1000_ID: '1'
    }
});
```

Стандартный вызов с обработчиками:

```
jsAPI.refresh({
    id: "PARAMS",
    page: "1000",
    items: {
        P1000_ID: '1'
    }
}, {
    onSuccess: function (data) {
        console.log(data);
    },
    onError: function (err) {
        console.error(err);
    }
});
```

14.1.20 jsAPI: refreshList()

Метод jsAPI.refreshList предназначен для обновления списков привязанных к элементам форм. Таких как Select List, Multiselect, Autocomplete. Выполняет xhr запрос {API_URL}/callAction со следующими параметрами:

```
{
  action: "REFRESH_LIST",
  data: {
    data: data
  }
}
```

Синтаксис

```
jsAPI.refreshList(data, callback);
```

Параметры

data Object

Данные для отправки в теле запроса

callbacks Object

Принимает функции: *onSuccess* Function, *onError* Function.

Примечание

Для обновления элемента формы необходимо указать его id в параметре data:

```
data: {
  {
    id: 'P1000_SELECT'
  }
}
```

Примеры

Стандартный вызов

```
jsAPI.refreshList({
    id: "P1000_SELECT",
    page: "1000",
    items: {
        "P1000_PARAM": 'New P1000_PARAM val'
    }
})
```

Стандартный вызов с обработчиками:

```
jsAPI.refreshList({
    id: "P1000_SELECT",
    page: "1000",
    items: {
```

(continues on next page)

(продолжение с предыдущей страницы)

```
        "P1000_PARAM": 'New P1000_PARAM val'
    }
}, {
    onSuccess: function (data) {
        console.log(data);
    },
    onError: function (err) {
        console.error(err);
    }
});
```

14.1.21 jsAPI: reload()

Осуществляет ререндеринг страницы после успешного ответа метода {API_URL}/showPage?page=id

Синтаксис

```
jsAPI.reload();
```

Параметры

Данная функция не принимает аргументов.

Примечание

не является аналогом window.loacation.reload(). Метод вызывает ререндеринг страницы после успешного ответа метода {API_URL}/showPage, как это делается при смене роута.

Примеры

```
jsAPI.reload();
```

14.1.22 jsAPI: reloadWindow()

Обёртка для нативного метода *window.location.reload()*

Перезагружает приложение на текущей странице.

Синтаксис

```
jsAPI.reloadWindow();
```

Параметры

Данная функция не принимает аргументов.

Примеры

```
jsAPI.reloadWindow();
```

14.1.23 jsAPI: setItem()

Изменяет значение элемента формы

Синтаксис

```
jsAPI.setItem(id, value);
```

Параметры

id [String] - id элемента формы. НЕ является селектором (указывать без „#“).

value [String] - Значение

Примеры

Присвоить новое значение

```
jsAPI.setItem('P1000_ITEM', 'Новое значение');
```

Очистить

```
jsAPI.setItem('P1000_ITEM', '');
```

14.1.24 jsAPI: showItem()

Возвращает удалённый при помощи *jsAPI.hideItem(id)* элемент формы (item) в DOM и привязанную к нему колонку

Синтаксис

```
jsAPI.showItem(id);
```

Параметры

id [String] - id элемента формы. НЕ является селектором (указывать без „#“)

Пример

```
jsAPI.showItem('P1000_ITEM');
```

14.1.25 jsAPI: submit()

Осуществляет сабмит страницы методом {API_URL}/processPage

Синтаксис

```
jsAPI.submit(items, callbacks);
```

Параметры

items [Object]

Дополнительные значения items, отправляемые в теле запроса {API_URL}/processPage, помимо основных items страницы. Допускается использовать пустой объект {} если не требутся отправки кастомных значений.

callbacks [Object]

Принимает функции *onSuccess* [Function], *onError* [Function].

Примечание

При кастомном обрабортчике onSuccess не будет запущена перезагрузка страницы (переопределение поведения по умолчанию), если требуется перезагрузка страницы и дополнительная логика в кастомном обработчике, используется метод jsAPI.reload() внутри onSuccess

Примеры

Стандартный вызов:

```
jsAPI.submit();
```

Вызов с дополнительными значениями items и обработчиком успешной отправки и ошибки:

```
jsAPI.submit(
  {
    MY_CUSTOM_ITEM: "CUSTOM VALUE",
  },
  {
    onSuccess: function (res) {
      console.log("При кастомном обрабортчике не будет перезагрузки страницы");
    },
    onError: function (err) {
      console.error(err); //Обработчик ошибки сервиса {API_URL}/processPage
```

(continues on next page)

(продолжение с предыдущей страницы)

```
    },
  }
);
```

Вызов без дополнительных значений items и с обработчиком успешной отправки:

```
jsAPI.submit(
  {},
  {
    onSuccess: function (res) {
      //Здесь может быть любой код
      jsAPI.reload(); //И после него выполнится перезагрузка страницы
    },
  }
);
```

14.1.26 jsAPI: updateItem()

Обновляет параметры элемента формы на уровне страницы ({API_URL}/showPage)

Синтаксис

```
jsAPI.updateItem(id, parameters);
```

Параметры

id String - id элемента формы. НЕ является селектором (указывать без „#“)

parameters Object

Содержит параметры:

label String - Плейсхолдер (лейбл элемента)

required Boolean - Обязательность заполнения

col Integer - Ширина колонки в форме (от 1 до 12)

mask String - Маска поля (работает только с TEXT)

maxlength Integer - Макс. кол-во символов (работает только с TEXT)

Пример

```
jsAPI.updateItem('P1_LOGIN', {
  "label": "Новый лейбл",
  "required": false,
  "col": 6,
  "mask": '999',
  "maxlength": 3
});
```

ГЛАВА 15

Функциональные характеристики

15.1 Архитектура

Архитектура приложения зависит от назначения и подразделяется:

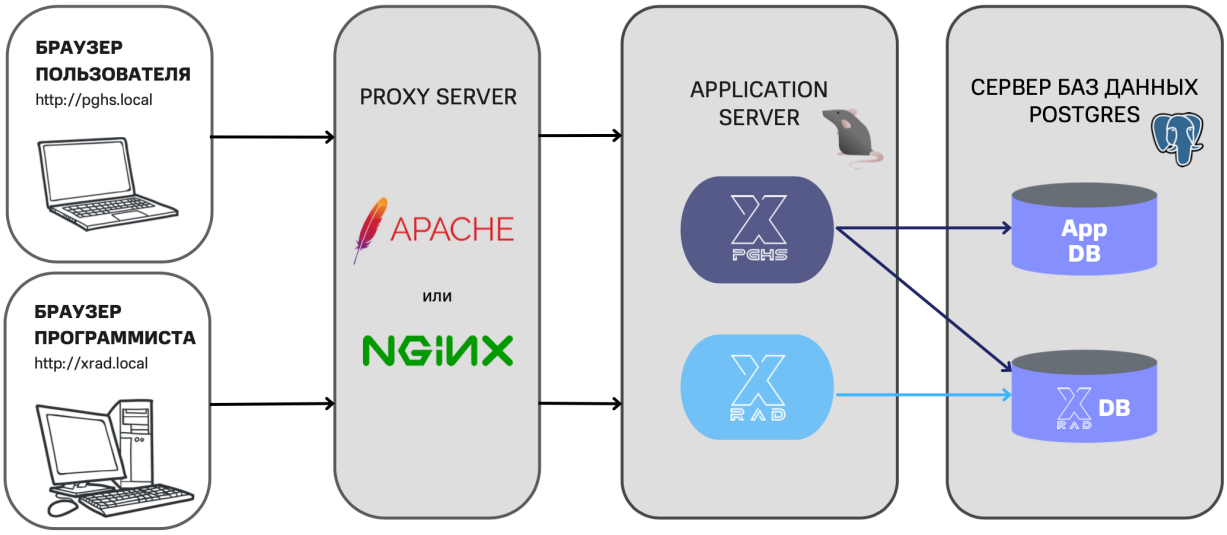
- Базовая
- Базовая плюс среда разработки (описывается в документации разработчика)
- Базовая для высоконагруженной среды

и состоит из четырех основных компонентов:

- HTTP Proxy Server
- PGHS - сервер приложений
- App DB - база данных приложения, в которой содержатся бизнес данные или бизнес DB.
- XRAD DB - мета база данных приложения, в которой храним описание самого приложения.

15.1.1 Базовая архитектура

Принципиальная или базовая архитектура приложения для **среды разработки** выглядит следующим образом:


БРАУЗЕР ПОЛЬЗОВАТЕЛЯ
http://pghs.local
БРАУЗЕР ПРОГРАММИСТА
http://xrad.local
PROXY SERVER
APACHE
или
NGINX
APPLICATION SERVER
PGHS
RAD
СЕРВЕР БАЗ ДАННЫХ POSTGRES
App DB
DB

ГЛАВА 16

Эксплуатация

16.1 Настройка Apache 2.4

Добавляем в файл /etc/hosts и прописываем в локальный DNS организации

```
127.0.0.1 xrad.xsquare
```

Создаем файл конфигурации *VirtualHost* для xrad-demo сервера

vi /etc/apache2/sites-available/xrad.xsquare.conf (vi /etc/httpd/conf.d/xrad-demo.xsquare.conf)

```
<VirtualHost *:80>
       ServerAdmin info@xsquare.ru
       ServerName  xrad.xsquare
       ServerAlias xrad.xsquare

       DocumentRoot /var/www/xrad.xsquare

       ProxyPass        /ds  http://127.0.0.1:8889/ds
       ProxyPassReverse /ds  http://127.0.0.1:8889/ds

       ErrorLog ${APACHE_LOG_DIR}/error.log
       CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Включаем сайты http://xrad.xsquare

```
a2ensite xrad.xsquare.conf
systemctl restart apache2
```

16.2 Настройка дистрибутива XSQUARE - XRAD

16.2.1 Настройка Web Resources XSQUARE - XRAD

Копируем файл Web приложения из дистрибутива

```
scp -r -P 22 xrad.xsquare root@IP_Server:/var/www/
```

16.2.2 Настройка сервера приложений XSQUARE - XRAD

Копируем дистрибутив сервера приложений

```
scp -r -P 22 /usr/local/xsquare.xrad  root@IP_Server:/usr/local
```

16.2.3 Настраиваем параметры соединения с PostgreSQL

vi /usr/local/xsquare.xrad/config.json

```
{
    "app": {
        "port": "8889"
    },
    "XRADDatabase": {
        "login": "xrad_user",
        "password": "xrad_user",
        "domain": "localhost",
        "port": "5432",
        "dbName":"xraddb",
        "runtimeOptions":{
            "LC_NUMERIC":"ru_RU.UTF-8"
        }
    },
    "AppDatabase": {
        "login": "app_user",
        "password": "app_user",
        "domain": "localhost",
        "port": "5432",
        "dbName":"appdb",
        "runtimeOptions":{
            "LC_NUMERIC":"ru_RU.UTF-8"
        }
    }
}
```

Создаем сервис

vi /etc/systemd/system/xsquare.xrad.service

```
[Unit]
Description=XRAD Services
After=syslog.target network.target
After=postgresql.service
```

(continues on next page)

(продолжение с предыдущей страницы)

```
[Service]
Type=simple
ExecStart=/usr/local/xsquare.xrad/xrad
WorkingDirectory=/usr/local/xsquare.xrad
Restart=on-failure
RestartSec=3

[Install]
WantedBy=default.target
```

Включаем для сервиса AvtoStart и запускаем и проверяем статус

```
systemctl enable xsquare.xrad.service
systemctl start  xsquare.xrad.service
systemctl status xsquare.xrad.service
```

16.2.4 Открываем и проверяем запуск

```
http://xrad.xsquare/
```