

---

# **Документация**

*Выпуск 5.1.2*

**Руководство разработчика XSQUARE - RAD 5.1.2**

**февр. 11, 2025**

<b>1</b>	<b>Общие сведения</b>	<b>1</b>
1.1	Для кого это руководство . . . . .	1
1.2	Требования к разработчику . . . . .	1
<b>2</b>	<b>Быстрый старт</b>	<b>2</b>
2.1	Знакомство с XRAD . . . . .	2
<b>3</b>	<b>Быстрая установка</b>	<b>4</b>
3.1	Быстрая установка на DEB-based ОС . . . . .	4
3.2	Быстрая установка на RPM-based ОС . . . . .	6
3.3	Введение в среду разработки . . . . .	8
<b>4</b>	<b>Архитектура и системные требования</b>	<b>11</b>
4.1	Архитектура . . . . .	11
4.2	Системные требования . . . . .	12
<b>5</b>	<b>Установка и настройка</b>	<b>14</b>
5.1	Установка XRAD . . . . .	14
5.2	Настройка NGINX . . . . .	15
5.3	Настройка Apache2 на DEB-based ОС . . . . .	16
5.4	Настройка Apache2 на RPM-based ОС . . . . .	16
5.5	Настройка XRAD . . . . .	17
<b>6</b>	<b>Конфигурационные файлы</b>	<b>19</b>
6.1	Файл конфигурации config.json . . . . .	19
6.2	Файл конфигурации схем аутентификации auth_config.json . . . . .	20
<b>7</b>	<b>Разработка приложений</b>	<b>25</b>
7.1	Основные концепции . . . . .	25
7.2	Настройки . . . . .	27
7.3	Работа со страницами . . . . .	38
7.4	Визуальные компоненты . . . . .	61
7.5	Регионы . . . . .	63
7.6	Компоненты . . . . .	67
7.7	Работа со списками . . . . .	97
7.8	Управление пользователями . . . . .	101
7.9	Стили и темы . . . . .	102

7.10 Справочник jsAPI . . . . . 109

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 - база данных) с помощью инструментов разработки XSQUARE RAD, далее XRAD. В руководстве описано как использовать инструменты для построения, отладки, управления и развертывания приложения.

### 1.2 Требования к разработчику

Для разработки приложений необходимы базовые знания концепции работы реляционных баз данных, SQL, основы HTML и JavaScript (js).

В данном разделе Вы познакомитесь с общей концепцией XRAD. Данный раздел описывает основные компоненты и возможности среды разработки/

## 2.1 Знакомство с XRAD

XRAD обеспечивает разработчика всеми инструментами для построения приложения в единой, расширяемой платформе на основе сервера баз данных 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.

Обычно веб-приложение разделяют на следующие компоненты:

- пользовательский интерфейс (frontend)
- компонент для взаимодействия с базой данных (backend)
- связующий компонент, который отвечает за различную 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 поддержание работоспособности, что требует наличия системного администратора.

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

Платформа XRAD предлагает совершенно другой подход к разработке.

XRAD - это платформа для быстрой разработки приложений, разработанная ООО «Хи-Квадрат» (Xsquare Rapid Application Development).

XRAD - является декларативной средой 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 значительно ускоряет процесс разработки приложений.

### 2.1.2 Как работает XRAD?

XRAD использует стандартную 3-уровневую архитектуру, в которой запросы отправляются из браузера через сервер приложений в базу данных. Вся обработка, манипулирование данными и бизнес-логика выполняются в базе данных. После того, как база данных обработает код, результаты будут переданы веб-сервером в виде структуры JSON, на основе которой веб-приложение отобразит страницу.

Эта архитектура гарантирует доступ к данным практически с нулевой задержкой, высочайшую производительность и горизонтальную масштабируемость «из коробки».

### 3.1 Быстрая установка на DEB-based ОС

Рассмотрим быструю установку XRAD + PGHS на примере ОС Debian. Все команды следует выполнять с правами суперпользователя (root)

#### 1. Создаем каталог для дистрибутива

```
mkdir /root/xsquare
```

переходим в каталог

```
cd /root/xsquare
```

#### 2. Скачиваем/получаем дистрибутив в созданный каталог

```
wget https://lcdp.xsquare.ru/files/pghs/xsquare.lcdp.v5/xsquare.lcdp.5.0_latest_
↪release.zip
```

#### 3. Распаковываем дистрибутив

```
apt -y install unzip
unzip xsquare.lcdp.5.0.0.0.0_release.zip
```

#### 4. Переходим в каталог с файлами дистрибутива

```
cd xsquare.lcdp.5.0.0.0.0_release
```

#### 5. Настраиваем часовой пояс и локализацию ОС

```
echo "Europe/Moscow" > /etc/timezone && \
dpkg-reconfigure -f noninteractive tzdata && \
sed -i -e 's/# en_US.UTF-8 UTF-8/en_US.UTF-8 UTF-8/' /etc/locale.gen && \
sed -i -e 's/# ru_RU.UTF-8 UTF-8/ru_RU.UTF-8 UTF-8/' /etc/locale.gen && \
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
echo 'LANG="ru_RU.UTF-8"'>/etc/default/locale && \  
dpkg-reconfigure --frontend=noninteractive locales && \  
  
export LANG=ru_RU.UTF-8
```

## 6. Устанавливаем PostgreSQL

```
apt -y install postgresql
```

## 7. Подготавливаем PostgreSQL

переключаемся на пользователя postgres

```
su - postgres
```

создаем пользователей БД xrad\_user и app\_user

```
psql -c "create user xrad_user with encrypted password 'xrad_user';"  
psql -c "create user app_user with encrypted password 'app_user';"
```

создаем базы appdb и xraddb

```
psql -c "CREATE DATABASE \"appdb\" WITH OWNER \"app_user\" ENCODING 'UTF8' LC_COLLATE_  
↪= 'ru_RU.UTF-8' LC_CTYPE = 'ru_RU.UTF-8';"  
psql -c "CREATE DATABASE \"xraddb\" WITH OWNER \"xrad_user\" ENCODING 'UTF8' LC_  
↪COLLATE = 'ru_RU.UTF-8' LC_CTYPE = 'ru_RU.UTF-8';"
```

назначаем пользователям xrad\_user и app\_user максимальные привилегии

```
psql -c "ALTER USER xrad_user WITH SUPERUSER;"  
psql -c "ALTER USER app_user WITH SUPERUSER;"
```

выходим из сеанса учетной записи postgres

```
exit
```

## 8. Импортируем базы данных

```
export PGPASSWORD='xrad_user';  
psql -U xrad_user -h 127.0.0.1 xraddb < db/xraddb.xsquare.pgsql  
  
export PGPASSWORD='app_user';  
psql -U app_user -h 127.0.0.1 appdb < db/appdb.xsquare.pgsql
```

## 9. Устанавливаем nginx

```
apt -y install nginx
```

отключаем сайт по умолчанию

```
rm -f /etc/nginx/sites-enabled/default
```

копируем из дистрибутива файлы веб-контроллера для PGHS и XRAD



```
cp -R ./var /
```

копируем из дистрибутива конфигурационные файлы nginx

```
cp -R ./etc/nginx /etc/
```

### 10. Перезапускаем nginx

```
systemctl restart nginx
```

проверяем его состояние

```
systemctl status nginx  
systemctl enable nginx
```

### 11. Копируем исполняемые и конфигурационные файлы XRAD, PGHS

```
cp -R ./etc/systemd /etc/  
cp -R ./usr /
```

### 12. Запускаем XRAD как службу и проверяем статус

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

### 13. Запускаем PGHS как службу и проверяем статус

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

### 14. Проверяем доступность дефолтного веб-приложения и конструктора XRAD в браузере

Примечание: в случае проблем с доступом по http необходимо проверить настройки nginx и разрешения в брандмауэре.

## 3.2 Быстрая установка на RPM-based ОС

Рассмотрим быструю установку XRAD + PGHS на примере ОС Fedora. Все команды следует выполнять с правами суперпользователя (root)

#### 1. Создаем каталог для дистрибутива

```
mkdir /root/xsquare
```

переходим в каталог

```
cd /root/xsquare
```

#### 2. Скачиваем дистрибутив в созданный каталог

```
wget https://lcdp.xsquare.ru/files/pghs/xsquare.lcdp.v5/xsquare.lcdp.5.0.0.0.0.0_  
↪release.zip
```

### 3. Распаковываем дистрибутив

```
dnf install -y unzip
unzip xsquare.lcdp.5.0.0.0.0.0_release.zip
```

### 4. Переходим в каталог с файлами дистрибутива

```
cd xsquare.lcdp.5.0.0.0.0.0_release
```

### 5. Настраиваем часовой пояс и локализацию ОС

```
timedatectl set-timezone Europe/Moscow
localectl set-locale LANG=ru_RU.UTF-8
export LANG=ru_RU.UTF-8
```

### 6. Устанавливаем и запускаем PostgreSQL

```
dnf install -y postgresql
postgresql-setup --initdb
systemctl start postgresql
systemctl enable postgresql
```

### 7. Подготавливаем PostgreSQL

переключаемся на пользователя postgres

```
su - postgres
```

создаем пользователей БД xrad\_user и app\_user

```
psql -c "create user xrad_user with encrypted password 'xrad_user';"
psql -c "create user app_user with encrypted password 'app_user';"
```

создаем базы appdb и xraddb

```
psql -c "CREATE DATABASE \"appdb\" WITH OWNER \"app_user\" ENCODING 'UTF8' LC_COLLATE_
↳= 'ru_RU.UTF-8' LC_CTYPE = 'ru_RU.UTF-8';"
psql -c "CREATE DATABASE \"xraddb\" WITH OWNER \"xrad_user\" ENCODING 'UTF8' LC_
↳COLLATE = 'ru_RU.UTF-8' LC_CTYPE = 'ru_RU.UTF-8';"
```

назначаем пользователям xrad\_user и app\_user максимальные привилегии

```
psql -c "ALTER USER xrad_user WITH SUPERUSER;"
psql -c "ALTER USER app_user WITH SUPERUSER;"
```

выходим из сеанса учетной записи postgres

```
exit
```

### 8. Импортируем базы данных

```
export PGPASSWORD='xrad_user';
psql -U xrad_user -h 127.0.0.1 xraddb < db/xraddb.xsquare.pgsql
```

```
export PGPASSWORD='app_user';
psql -U app_user -h 127.0.0.1 appdb < db/appdb.xsquare.pgsql
```

### 9. Устанавливаем nginx

```
dnf install -y nginx
```

отключаем сайт по умолчанию

```
rm -f /etc/nginx/sites-enabled/default
```

копируем из дистрибутива файлы веб-контроллера для PGHS и XRAD

```
cp -R ./var /
```

копируем из дистрибутива конфигурационные файлы nginx

```
cp -R ./etc/nginx /etc/
```

### 10. Перезапускаем nginx

```
systemctl restart nginx
```

проверяем его состояние

```
systemctl --no-pager status nginx
```

### 11. Копируем исполняемые и конфигурационные файлы XRAD, PGHS

```
cp -R ./etc/systemd /etc/  
cp -R ./usr /
```

### 12. Запускаем XRAD как службу и проверяем статус

```
systemctl start xsquare.xrad.service  
systemctl enable xsquare.xrad.service  
systemctl --no-pager status xsquare.xrad.service
```

### 13. Запускаем PGHS как службу и проверяем статус

```
systemctl start xsquare.pghs.service  
systemctl enable xsquare.pghs.service  
systemctl --no-pager status xsquare.pghs.service
```

### 14. Проверяем доступность дефолтного веб-приложения и конструктора XRAD в браузере

Примечание: в случае проблем с доступом по http необходимо проверить настройки nginx и разрешения в брандмауэре.

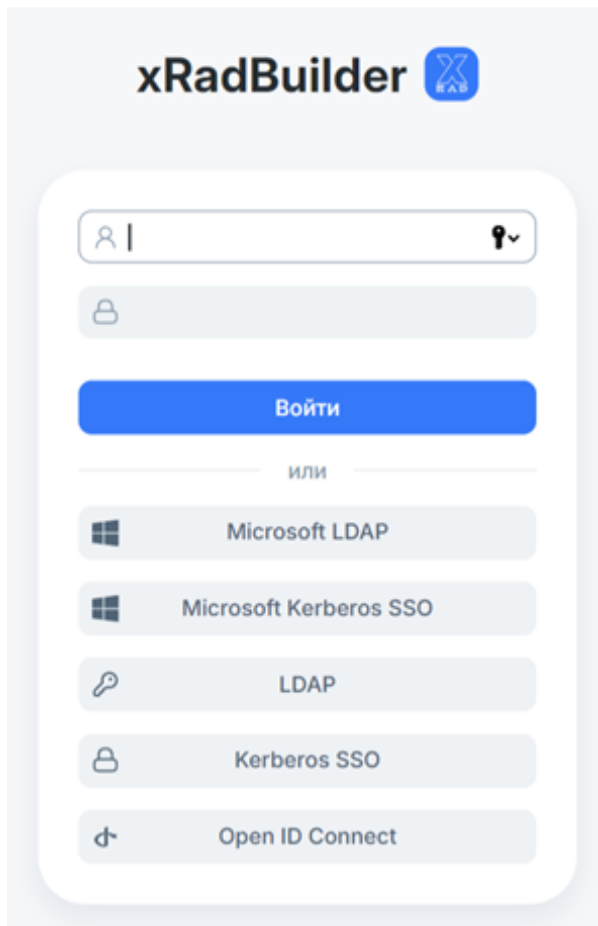
## 3.3 Введение в среду разработки

Среда разработки XRAD предоставляет разработчику гибкий и интуитивно понятный интерфейс для разработки и управления приложением.

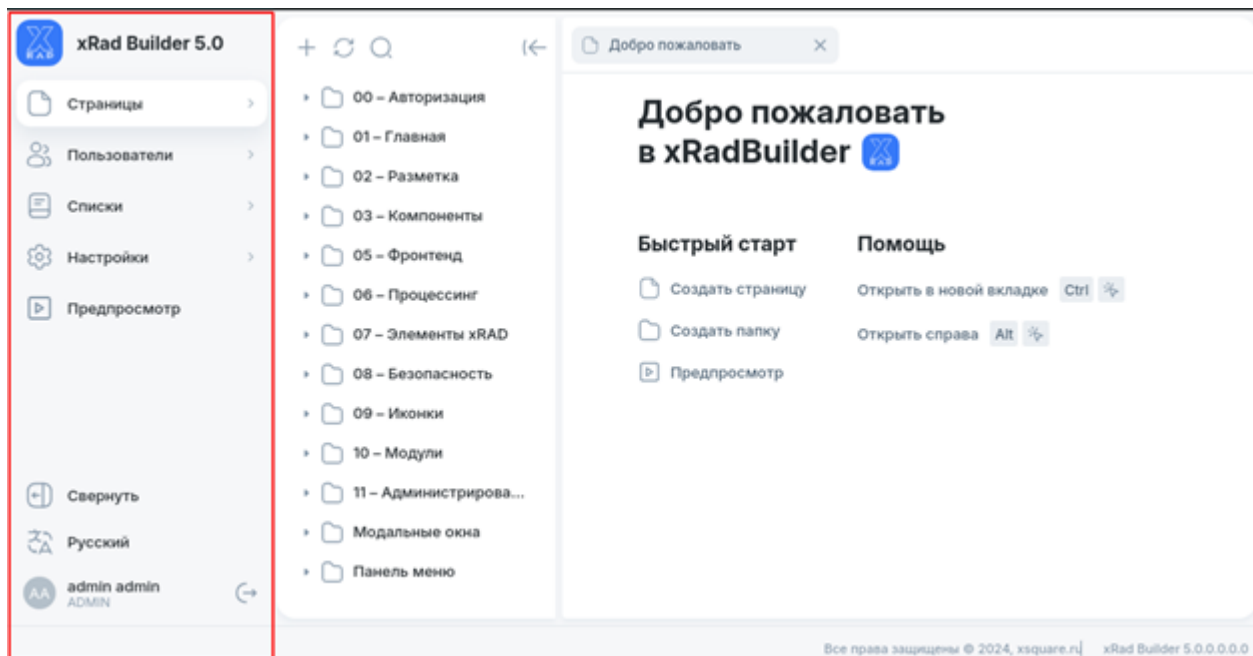
Начало работы со средой начинается с процесса аутентификации пользователя на странице:

<http://hostname:8080>

Где hostname – имя хоста с установленным XRAD.



После успешной аутентификации откроется главная страница среды разработки. Интерфейс среды разработки можно условно разделить на 3 области (слева-направо):



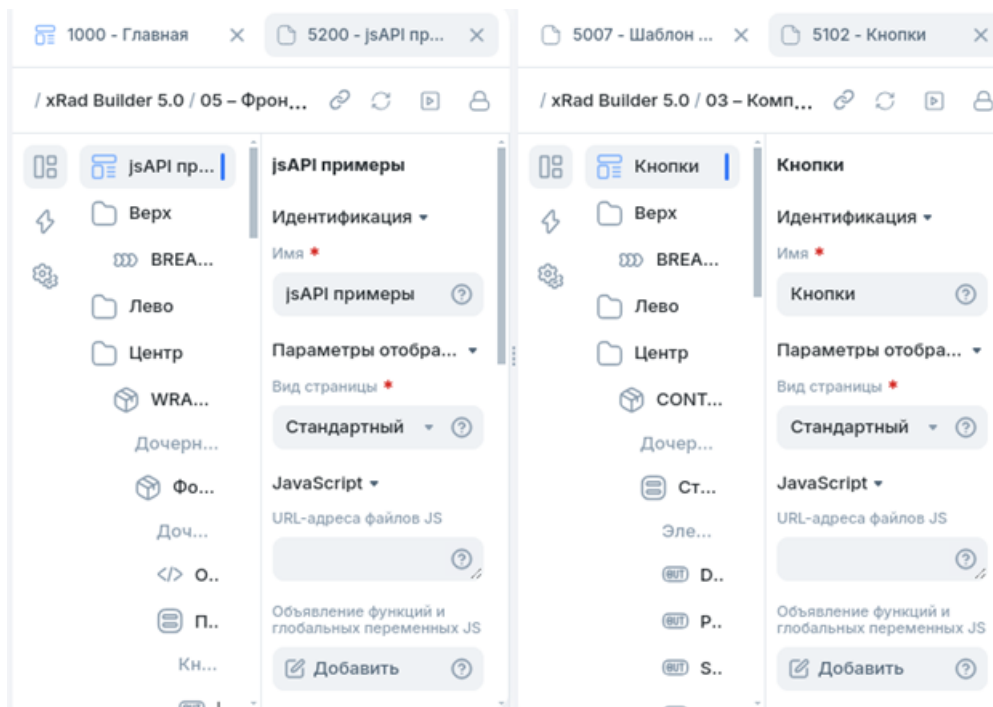
1. Главное меню – содержит ключевые элементы разрабатываемого приложения, а также информация о текущем пользователе и языке интерфейса.
2. Список элементов – здесь отображаются наборы данных для каждого пункта главного меню, а также кнопки для создания новых элементов, обновления набора данных и контекстного поиска по набору данных. Данная область поддерживает контекстное меню, доступное по нажатию ПКМ.
3. Редактор – здесь отображаются вкладки, в которых происходит создание и редактирование выбранных элементов.

Главное меню содержит следующие разделы:

- Страницы – список страниц веб-приложения.
- Пользователи – список пользователей среды разработки.
- Списки - перечень predefined списков, используемых для работы компонентов приложения и навигации.
- Настройки – настройки различных параметров приложения.
- Предпросмотр – кнопка перехода к просмотру приложения в новом окне.

Также в главном меню можно сменить язык интерфейса и завершить сессию текущего пользователя.

Среда разработки поддерживает механизм перетаскивания элементов - drag&drop, а редактор поддерживает режим разделения окна, что позволяет значительно ускорить разработку.



В следующих главах будет дано детальное описание среды разработки и ее компонентов.

---

## Архитектура и системные требования

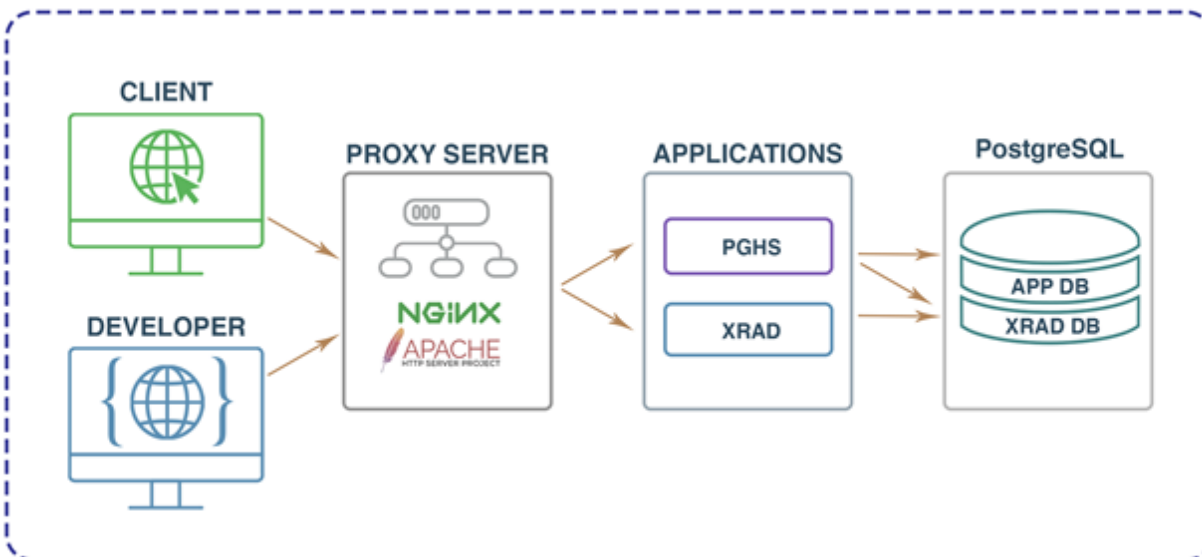
---

### 4.1 Архитектура

Базовая архитектура XRAD состоит из 4-х компонентов:

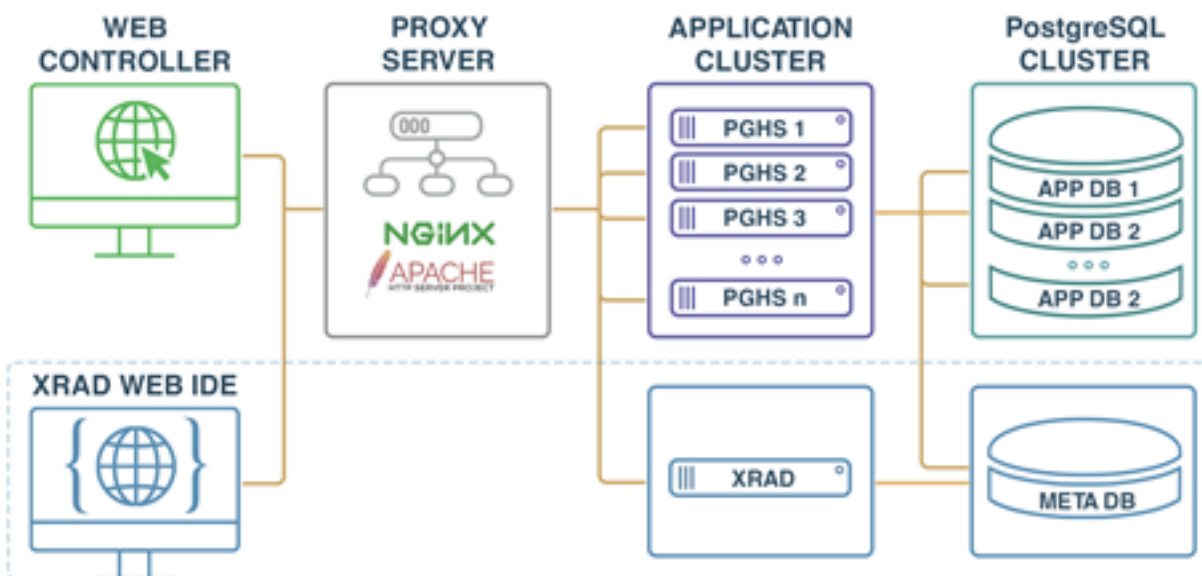
- База данных XRAD – база данных с метаданными приложения. В ней хранится информация о компонентах запрашиваемой станицы, процессах которые должны произойти во время обработки станицы.
- Сервер приложений XRAD, который отвечает за обработку действий разработчика и подготовку структуры станицы для рендеринга.
- Веб-контроллер XRAD, которое обрабатывает полученную от веб-сервера структуру станицы и отображает её.
- HTTP Proxy Server, который связывает веб-сервер и веб-контроллер XRAD.

Для отображения разрабатываемого веб-приложения необходим сервер приложений PGHS, поэтому XRAD и PGHS работают в связке и имеют одинаковую 4-х компонентную архитектуру. Независимо от того, запускаете ли вы среду разработки XRAD или приложение, созданное с использованием XRAD - процесс функционирования один и тот же. Ваш браузер отправляет запрос, который преобразуется в соответствующий вызов кода на стороне базы данных. После того, как база данных обработает код, результаты будут переданы обратно в ваш браузер в виде структуры JSON, на базе которой веб-приложение сформирует станицу. Этот цикл происходит каждый раз, когда вы запрашиваете или отправляете станицу.



Все компоненты могут быть развернуты как в рамках одного сервера, так и разнесены по разным физическим или виртуальным серверам.

Для высоконагруженных систем можно легко произвести горизонтальное масштабирование. Пример высоконагруженной архитектуры выглядит следующим образом:



## 4.2 Системные требования

### 4.2.1 Среда исполнения

Поддерживаемые архитектуры:

- x86-64

- ARM
- Loongson

Поддерживаемые ОС:

- **DEB**-based – любые
- **RPM**-based – любые
- Debian 12 – рекомендуемая

Базы данных:

- PostgreSQL 13+
- PostgreSQL 15 – рекомендуемая

HTTP/Proxy Server:

- Apache 2.4+
- NGINX 19+

## 4.2.2 Системные требования

XRAD - сервер:

- CPU - 1 Ядро
- RAM - 100 Мб
- HDD - 100 Мб + Логи

XRAD DB:

- CPU - 1 Ядро
- RAM - 50 Мб
- HDD - 10 Мб БД PostgreSQL

Установка системы виртуализации/контейнеризации, операционной системы, базы данных осуществляется на усмотрение Администратора исходя из потребностей.



---

## Установка и настройка

---

### 5.1 Установка XRAD

Подробная настройка ОС и установка всех компонентов кроме XRAD рассмотрены в документации PGHS. В данной документации рассмотрим непосредственно развертывание XRAD.

Примечание: примеры приводятся для случая, когда пользователь находится в каталоге с дистрибутивом, все действия выполняются с правами суперпользователя (root).

Для установки XRAD копируем исполняемые файлы из дистрибутива в каталог `/usr/local/xsquare.xrad`

Например:

1. Копируем все компоненты

```
cp -R ./usr /
```

или 2. Копируем непосредственно дистрибутив XRAD

```
cp -R ./usr/local/xsquare.xrad /usr/local/
```

Примечание: назначьте права на исполнение

```
chmod +x /usr/local/xsquare.xrad/xrad
```

Создаем сервис

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

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

[Service]
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

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

[Install]
WantedBy=default.target
```

Примечание: файл сервиса по умолчанию можно скопировать из дистрибутива. Например:

```
cp -R ./etc/systemd /etc/
```

Запускаем XRAD как службу и проверяем статус

```
systemctl start xsquare.xrad.service
systemctl enable xsquare.xrad.service
systemctl --no-pager status xsquare.xrad.service
```

Далее настроим HTTP проxy server на примере NGINX и Apache2

Копируем из дистрибутива файлы веб-контроллера XRAD

```
cp -R ./var/www/xrad.xsquare*
```

## 5.2 Настройка NGINX

Копируем из дистрибутива конфигурационные файлы nginx

```
cp -R ./etc/nginx /etc/
```

и редактируем файл `/etc/nginx/conf.d/xrad.xsquare.conf`, внося необходимые изменения `vi /etc/nginx/conf.d/xrad.xsquare.conf`

```
server {
    listen 8080;
    server_name xrad.xsquare;
    root /var/www/xrad.xsquare;
    index index.html;
    location /ds{
        proxy_pass http://127.0.0.1:8889/ds;
    }
    location / {
        try_files $uri $uri/ =404;
    }
}
```

Для применения новых настроек перезапускаем nginx

```
systemctl restart nginx
systemctl enable nginx
```

проверяем его состояние

```
systemctl --no-pager status nginx
```

## 5.3 Настройка Apache2 на DEB-based ОС

Копируем из дистрибутива конфигурационные файлы apache2

```
cp -R ./etc/apache2 /etc/
```

и редактируем файл конфигурации VirtualHost `/etc/apache2/sites-available/xrad.xsquare.conf`, внося необходимые изменения:

```
vi /etc/apache2/sites-available/xrad.xsquare.conf

<VirtualHost *:80>
ServerAdmin info@xsquare.ru
ServerName xrad.xsquare
ServerAlias xrad.xsquare
DocumentRoot /var/www/xrad.xsquare

Alias /files "/var/www/xrad.xsquare.files.local"
<Directory /var/www/xrad.xsquare.files.local>
    Options FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

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>
```

Применяем новую конфигурацию

```
a2ensite xrad.xsquare.conf
```

и перезапускаем apache2

```
systemctl restart apache2
```

## 5.4 Настройка Apache2 на RPM-based ОС

Копируем из дистрибутива конфигурационные файлы httpd

```
cp -R ./etc/httpd /etc/
```

и редактируем файл конфигурации VirtualHost `/etc/httpd/conf.d/xrad.xsquare.conf`, внося необходимые изменения:

```

vi /etc/httpd/conf.d/xrad.xsquare.conf
<VirtualHost *:80>
ServerAdmin info@xsquare.ru
ServerName xrad.xsquare
ServerAlias xrad.xsquare
DocumentRoot /var/www/xrad.xsquare

Alias /files "/var/www/xrad.xsquare.files.local"
<Directory /var/www/xrad.xsquare.files.local>
    Options FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>

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>

```

Отключаем Security-Enhanced Linux для HTTP запросов

```
setsebool -P httpd_can_network_connect 1
```

и перезапускаем apache

```
systemctl restart httpd
```

## 5.5 Настройка XRAD

Для настройки XRAD редактируем файл config.json:

```

vi /usr/local/xsquare.xrad/config.json
{
"app": {
    "port": "8889"
},
"XRAD": {
    "login": "xrad_user",
    "password": "xrad_user",
    "host": "localhost",
    "port": 5432,
    "minCons": 1,
    "maxCons": 15,
    "dbName": "xraddb",
    "runtimeOptions": {
        "LC_NUMERIC": "ru_RU.UTF-8"
    }
},
"datasources": {
    "login": "app_user",

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
"password": "app_user",
"host": "localhost",
"port": 5432,
"minCons": 1,
"maxCons": 15,
"dbName": "appdb",
"runtimeOptions": {
  "LC_NUMERIC": "ru_RU.UTF-8"
}
}
```

---

## Конфигурационные файлы

---

### 6.1 Файл конфигурации config.json

Для работы XRAD необходимо, чтобы в каталоге с исполняемым файлом присутствовал файл конфигурации config.json.

Файл конфигурации содержит 3 раздела: Описатель «app», где можно определить базовые настройки сервера:

```
{
  "app": {
    "port": "8889"
  },

```

«port» - строка. Определяет номер сетевого порта, на котором будет запущен сервер (по умолчанию - 8889)

Описатель «XRAD», где определяются настройки для работы с БД XRAD:

- «login» – строка. Имя пользователя для подключения к базе данных XRAD.
- «password» – строка. Пароль пользователя для подключения к базе данных XRAD.
- «host» – строка. IP-адрес сервера базы данных XRAD.
- «port» - число. Номер порта, на котором работает сервер базы данных XRAD.
- «dbName» – строка. Имя базы данных, к которой требуется подключиться XRAD.
- «minCons» – число. Минимальное количество одновременных соединений с базой данных.
- «maxCons» - число. Максимальное количество одновременных соединений с базой данных.

описатель runtimeOptions содержит локальные настройки:

- «LC\_NUMERIC» – строка. Локальные настройки числового формата, используемого для работы с базой данных.

Описатель «datasources» определяет массив источников данных, используемых сервером приложений. Блок описания источника данных содержит тот же набор полей, что и описание БД XRAD и дополнительное поле:

- «name» - строка. Имя источника данных.

Например, в следующем блоке определяется два источника данных с именами DEFAULT\_APP и DEFAULT\_APP\_TEST

```
"datasources": [
  {
    "login": "app_user",
    "password": "app_user",
    "host": "10.100.117.219",
    "name": "DEFAULT_APP",
    "port": 5432,
    "minCons": 1,
    "maxCons": 15,
    "dbName": "pghs",
    "runtimeOptions": {
      "LC_NUMERIC": "ru_RU.UTF-8"
    }
  },
  {
    "login": "app_user",
    "password": "app_user",
    "host": "10.100.117.219",
    "name": "DEFAULT_APP_TEST",
    "port": 5432,
    "minCons": 1,
    "maxCons": 15,
    "dbName": "pghs",
    "runtimeOptions": {
      "LC_NUMERIC": "ru_RU.UTF-8"
    }
  }
]
```

## 6.2 Файл конфигурации схем аутентификации auth\_config.json

XRAD поддерживает аутентификацию и авторизацию с использованием следующих схем:

- Microsoft LDAP,
- Microsoft Kerberos SSO
- LDAP
- Kerberos SSO
- Open ID Connect

При старте сервер разработки XRAD загружает схемы из файла auth\_config.json и использует их при аутентификации и авторизации разработчиков.

Файл auth\_config.json содержит массив описателей схем аутентификации следующего формата:

```
[
  {
    "name": "",
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
"label": "",  
"type": "",  
"enabled": ,  
"order": ,  
"options": {}  
}
```

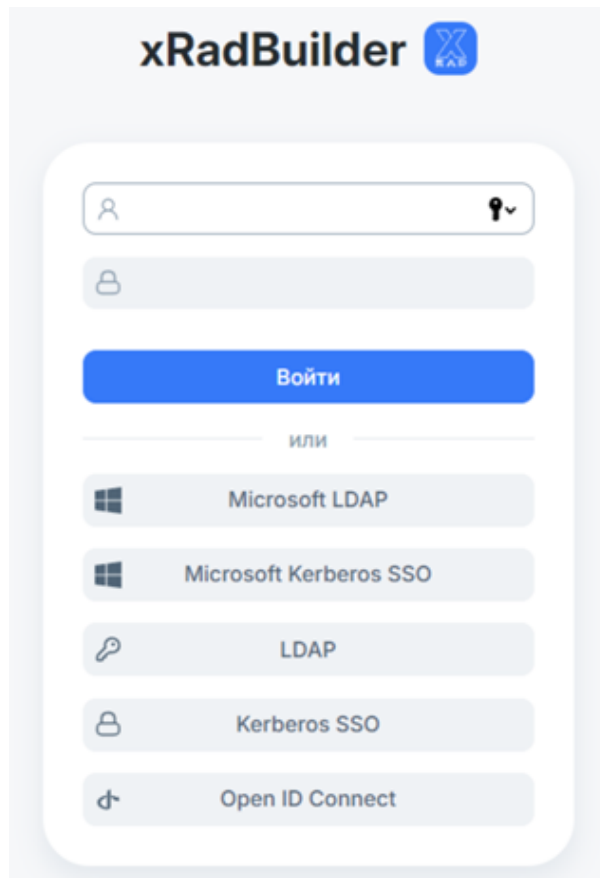
- «name» - строка. Имя схемы аутентификации.
- «label» - строка. Отображаемое на кнопке наименование схемы аутентификации.
- «type» – строка. Тип схемы аутентификации.

Поддерживаемые значения:

- ldap
- kerberos\_sso
- microsoft\_ldap
- microsoft\_kerberos\_sso
- oide
- «enabled» – логическое значение. Данный флаг определяет доступность схемы аутентификации для использования в приложении.
- «order» – число. Порядковый номер в общем списке схем при отображении на странице аутентификации.
- «options» – блок параметров схемы, который зависит от типа схемы.

Загруженные из конфигурационного файла схемы будут использоваться на странице авторизации XRAD в соответствии с флагом enabled и порядковым номером order.





Рассмотрим блок `options` для различных схем аутентификации.

### 6.2.1 LDAP, MICROSOFT\_LDAP

Для типа схемы `ldap` и `microsoft_ldap` блок «options» описывает параметры подключения к LDAP-серверу, которые используются для аутентификации и получения информации о пользователях.

- «host» – строка. IP-адрес LDAP-сервера.
- «port» – число. Порт, используемый для связи с LDAP-сервером.
- «base» – строка. Определяет базовую точку входа в LDAP-каталог.
- «encryption» – строка. Определяет тип шифрования, используемого для связи с LDAP-сервером. Поддерживаемые значения:
  - `start_tls` – устанавливает защищенное TLS-соединение после начальной аутентификации по незашифрованному каналу. Обычно используется на порту 389.
  - `simple_tls` – устанавливает полностью зашифрованное TLS-соединение с самого начала. Обычно используется на порту 636.
  - `plain` – не использует шифрование и работает по незащищенному каналу. Обычно используется на порту 389.
- «bind\_dn» – строка. Distinguished Name (DN) пользователя, используемого для аутентификации на LDAP-сервере.
- «password» – строка. Пароль, связанный с указанным `bind_dn`.

- «request\_user\_groups» – логическое значение. Определяет необходимость запроса групп, к которым принадлежит пользователь.

Например:

```
{
  "name": "User Auth LDAP",
  "type": "ldap",
  "options": {
    "host": "10.100.117.229",
    "port": 389,
    "base": "DC=ald,DC=dom",
    "encryption": "start_tls",
    "bind_dn": "uid=ldap,cn=users,cn=accounts,dc=ald,dc=dom",
    "password": "password",
    "request_user_groups": true
  }
}
```

## 6.2.2 KERBEROS\_SSO, MICROSOFT\_KERBEROS\_SSO

Для типа схемы `kerberos_sso` и `microsoft_kerberos_sso` блок «options» описывает параметры подключения к LDAP-серверу, которые используются для аутентификации и получения информации о пользователях.

- «keytab» – строка. Закодированное в BASE64 содержимое keytab файла сгенерированного для сквозной доменной аутентификации.
- «request\_user\_groups» – логическое значение. Определяет необходимость запроса групп, к которым принадлежит пользователь.

Например:

```
{
  "name": "User Auth Kerberos SSO",
  "type": "kerberos_sso",
  "options": {
    "keytab": "BQIAAABVAAIAB0FMRC5ET00ABEhUVFAAEXBnaHM0LWRldi5hbG",
    "request_user_groups": true
  }
},
```

## 6.2.3 OIDC

Для типа схемы `oidc` (Open Id Connect) блок «options» описывает конфигурацию аутентификации и авторизации пользователей с использованием OpenID Connect (OIDC).

- «scope»- строка. Список запрашиваемых областей, к которым требуется доступ при аутентификации.
- «issuer»- строка. URL-адрес сервера аутентификации (Issuer).
- «uid\_field» - строка. Определяет поле, содержащее уникальный идентификатор пользователя (UID).
- «pkce»- логическое значение. Флаг, указывающий на необходимость использования Proof Key for Code Exchange (PKCE) для повышения безопасности.
- «client\_options» - описатель блока параметров клиента, использующего OIDC-аутентификацию.
- «id»- строка. Идентификатор клиента(Client ID).

- «secret» - строка. Секретный ключ клиента(Client Secret).
- «redirect\_uri»- строка. URL-адрес, на который будет перенаправлен пользователь после успешной аутентификации.
- «request\_user\_groups» – логическое значение. Определяет необходимость запроса групп, к которым принадлежит пользователь.

### 7.1 Основные концепции

Для эффективной работы в XRAD разработчики должны понимать некоторые ключевые понятия:

- как происходит управление дизайном пользовательского интерфейса;
- как происходят процессы обработки и отображения страниц;
- что такое сессии, транзакции и как с ними работать.

#### 7.1.1 Концепция приложения

##### Что такое приложение?

Веб-приложение, которое разрабатывается с помощью XRAD - это HTML интерфейс, который существует поверх объектов базы данных: таблиц и процедур. Приложение логически и физически разделяется на две базы данных: база данных, отвечающая за бизнес-логику (APP\_DB) и база данных с набором метаданных интерфейса приложения (XRAD\_DB). Сервер приложений 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. Один экземпляр PGHS может обращаться к неограниченному количеству баз данных APP\_DB и только к одной базе данных XRAD\_DB.

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

##### Что такое страница?

Страница - это «строительный блок» приложения. Приложение может содержать всего одну, но обычно множество таких блоков. Каждая страница может содержать множество как простых элементов - кнопки и поля для ввода, так и более сложные - разнообразные отчеты, графики, таблицы. Элементы на странице группируются с помощью специального контейнера - региона. Страницы могут включать в себя логику обработки данных, называемую процессами. Связь страниц друг с другом обеспечивают ссылки или ветви.

## 7.1.2 Рендеринг страницы и обработка ввода

Для просмотра страниц разрабатываемого приложения необходимо вызвать страницу приложения XRAD по 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.

### Транзакции

При вызове процессов showPage или processPage инициируется транзакция в БД с бизнес-данными. Эта транзакция существует на протяжении всего времени выполнения процессов showPage или processPage. В случае успешного выполнения **всех** процессов транзакция завершается вызовом commit, за исключением случая, когда на уровне процесс установлен атрибут **Фиксация ТХ**. Если возникает ошибка в процессах обработки данных - транзакция будет завершена вызовом rollback, что отменит все изменения 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 уровня изолированности транзакции, настроенного в БД с бизнес-данными. 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.

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.

## 7.1.3 Сессии и их состояние

### Что такое сессия

Сессия — это период взаимодействия пользователя с приложением. Она начинается, когда пользователь обращается к приложению, и заканчивается, когда пользователь закрывает сессию, выходит из системы или проходит определённый период бездействия (таймаут). Во время сессии приложение хранит информацию о пользователе и его действиях, что позволяет поддерживать контекст взаимодействия и персонализировать отображаемую информацию.

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

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

Сессии логически и физически отличаются от сессий 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 сессии, может иницииро-

вать создание новой сессии базы данных или использовать повторно открытые сессии базы данных для доступа к ресурсам базы данных. Часто такие сессии базы данных длятся всего доли секунды.

### **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 состоянии сессии. XRAD предоставляет разработчикам возможность получать и устанавливать значения состояния сессии с любой страницы приложения.

## **7.1.4 Управление значениями сессии**

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

## **7.2 Настройки**

Раздел «Настройки» среды разработки содержит:

1. Основные – главные настройки разрабатываемого приложения.
2. Источники данных – список БД, которые могут использоваться в качестве источников данных для приложения.
3. Глобальные переменные – список глобальных переменных.
4. Глобальные процессы – список глобальных процессов приложения.
5. Схемы авторизации – список доступных при разработке схем для авторизации пользователей.
6. Схемы аутентификации – список доступных при разработке схем для аутентификации пользователей.

### **7.2.1 Основные**

Основные настройки уровня приложения являются обязательными для заполнения и делятся на следующие группы:

- Основные;
- Списки;
- Сессия;
- Статичные ресурсы.

Параметр	Описание
Наименование приложения	Название приложения. Выводится в заголовке страницы.
Короткое наименование	Отображаемое название приложения в среде разработки.
URL приложения	URL адрес приложения.
Домашняя страница	Основная страница, на которую попадает авторизованный пользователь. В поле ввода необходимо ввести, либо выбрать из списка номер страницы, которая будет использоваться в качестве домашней.
Страница входа	Основная публичная страница, на которую попадает неавторизованный пользователь. Как правило, страница авторизации. В поле ввода необходимо ввести, либо выбрать из списка номер страницы, которая будет использоваться в качестве страницы входа.

### Списки

В этом блоке параметров определяются списки уровня приложения.

Параметр	Описание
Панель навигации	Определяет список для меню в заголовке страницы. Выберите из перечня заранее созданных списков в соответствующем разделе.
Меню навигации	Определяет список для основного меню в левом блоке страницы. Выберите из перечня заранее созданных списков в соответствующем разделе.

### Сессия

В этом блоке определяются параметры пользовательских сессий.

Параметр	Описание
Время жизни сессии	Определяет время жизни пользовательской сессии, т.е. период в единицах времени, в течение которого активна пользовательская сессия взаимодействия с приложением.
Время жизни сессии (ед. изм.)	Определяет единицу измерения для указанного в параметре «Время жизни сессии» числа. На выбор 4 варианта: секунды, минуты, часы и дни.
Время жизни неактивной сессии	Определяет период, в единицах времени, по истечению которого будет удалена пользовательская сессия, если пользователь не производит никаких действий в приложении.
Время жизни неактивной сессии (ед. изм.)	Определяет единицу измерения для указанного в параметре «Время жизни неактивной сессии» числа. На выбор 4 варианта: секунды, минуты, часы и дни.

### Статические ресурсы

В этом блоке определяются внешние статические ресурсы, которые позволяют изменить отображение пользовательского интерфейса приложения.

Параметр	Описание
CSS файлы	Список файлов .css, содержащие таблицы стилей. Каждая строка определяет относительный либо абсолютный путь, на внутренний или внешний ресурс.
JS файлы	Список файлов .js, содержащие код JavaScript. Каждая строка определяет относительный либо абсолютный путь, на внутренний или внешний ресурс.

## 7.2.2 Источники данных

Источники данных – один из важнейших компонентов XRAD, предоставляющий возможность горизонтально масштабировать разрабатываемое приложение. Приложение позволяет создать неограниченное количество источников данных. Источником данных выступает БД, данные которой используются для отображения компонентов приложения. Указывая различные источники данных, разработчик может объединить вывод данных из разных баз на одной странице.

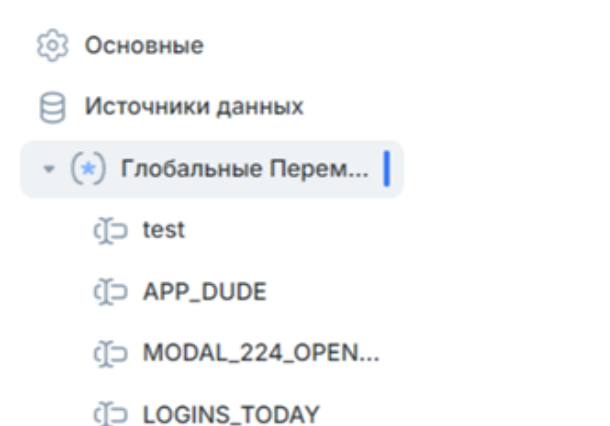
Для того чтобы добавить новый источник данных укажите его в настройках приложения. В настройках указывается только название источника, которое будет использовано для подстановки в соответствующие поля компонентов редактора страниц.

Настройки подключения источника данных указываются в конфигурационных файлах config.json PGHS и XRAD, при этом наименование источника в конфигурационном файле должно соответствовать введенному в настройках приложения.

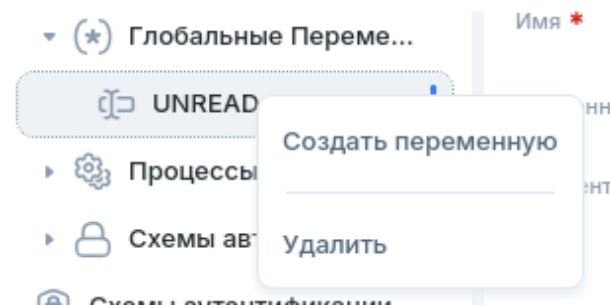
## 7.2.3 Глобальные переменные

В этом блоке настроек определяются глобальные переменные приложения.

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



Объявление новой глобальной переменной осуществляется по нажатию ПКМ по списку «Глобальные переменные», либо по любому элементу данного списка.



При создании глобальной переменной необходимо заполнить ее атрибуты:



Параметр	Описание
Имя	единственный обязательный атрибут, определяющий имя создаваемой переменной.
Временная	флаг, который определяет, будет ли переменная временной. Временная переменная не хранится в состоянии сессии и доступна только на время выполнения процессов.
Комментарий	текстовый комментарий.

### Глобальная переменная UNREAD\_NOTIFICATIONS

Имя *	UNREAD_NOTIFICATIONS
Временная	<input type="checkbox"/>
Комментарий	Непрочитанные уведомления (Глобальные процессы)

После нажатия кнопки «Сохранить» переменная будет занесена в базу данных XRAD и доступна для просмотра, редактирования и удаления. Аналогично созданию происходит редактирование существующих переменных.

Для удаления глобальной переменной необходимо нажать ПКМ по необходимой переменной в списке и выбрать из меню «Удалить».

### Предустановленные переменные

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

#### Зарезервированные глобальные переменные:

Временные:

- **REQUEST** - код запроса для выполнения необходимого процесса;
- **PAGE** - номер страницы;
- **RESPONSE** - выходной параметр выполненного процесса Ajax;
- **G01 - G10** – встроенные переменный для передачи значений в процессы. Можно использовать при программном вызове процессов с помощью js.
- **CGI\_FORWARDED\_FOR** – переменная, которая хранит значение заголовка X-Forwarded-For, переданного веб сервером.
- **CGI\_FORWARDED\_IP** - переменная, которая хранит значение заголовка X-Forwarded-IP, переданного веб сервером.
- **CGI\_REAL\_IP** - переменная, которая хранит значение заголовка X-Real-IP, переданного веб сервером.

Постоянные:

- **APP\_USER** - id авторизованного на данный момент пользователя;
- **SESSION** - уникальный код сессии;
- **APP\_USER\_GROUPS** – доменные группы, авторизованного на данный момент пользователя;

## 7.2.4 Глобальные процессы

В этом блоке настроек определяются глобальные процессы для всего приложения.

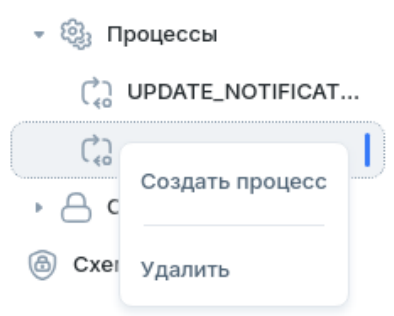
Процесс приложения – это процесс XRAD не имеющий привязки к какой-либо странице. Такие процессы, также как и процессы на страницах, выполняются по запросу от приложения (Request).

### Порядок выполнения процессов приложения

При появлении запроса от приложения система отбирает все удовлетворяющие условиям запроса процессы (глобальные и страничные), сортирует в соответствии с их порядковыми номерами и выполняет.

### Управление процессами

Для создания или удаления процесса необходимо вызвать контекстное меню, нажав ПКМ на списке процессов и выбрать соответствующее действие.



После заполнения или редактирования параметров процесса необходимо подтвердить внесенные изменения, нажав кнопку «Сохранить».

## Процесс UPDATE\_NOTIFICATIONS

## Основные ▾

Имя *	UPDATE_NOTIFICATIONS
Порядковый номер *	1
Источник данных *	DEFAULT_APP ▾
Тип *	SHOW ▾
Коммит после выполнения	<input type="checkbox"/>
Текст ошибки	<input type="text"/>
Sql код *	<pre> 1 SELECT 2   COUNT(1) 3 FROM 4   events.t_notifications </pre>

## Элементы ▾

Ввод	<input type="text"/>
Вывод	UNREAD_NOTIFICATIONS
Условия ▾	
Тип Условия	Always ▾ ×

Параметры процесса делятся на 3 группы:

## 1. Основные

Параметр	Описание
Имя	обязательный атрибут, определяющий имя процесса.
Порядковый номер	обязательный атрибут, определяющий очередность выполнения процесса.
Источник данных	обязательный атрибут, определяющий в каком источнике данных будет выполнен процесс.
Тип	обязательный атрибут, определяющий тип процесса: <ul style="list-style-type: none"> <li>• SHOW</li> <li>• PROCESS</li> <li>• AJAX</li> </ul>
Коммит после выполнения	флаг, определяющий необходимость завершения транзакции
Текст ошибки	Строка с текстом, которая будет отображена, если процесс завершится с ошибкой.
Sql код	Обязательный атрибут, определяющий код SQL-запроса.

## 2. Элементы

Параметр	Описание
Ввод	Наименования переменных через запятую, содержащих входные параметры
Вывод	Наименования переменных через запятую, куда будут записаны результаты

### 3. Условия

Тип условия – определяет условие при котором будет выполнен процесс, действие или событие.

#### Типы условий

Разработчику доступно 11 типов условий:

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 никогда.

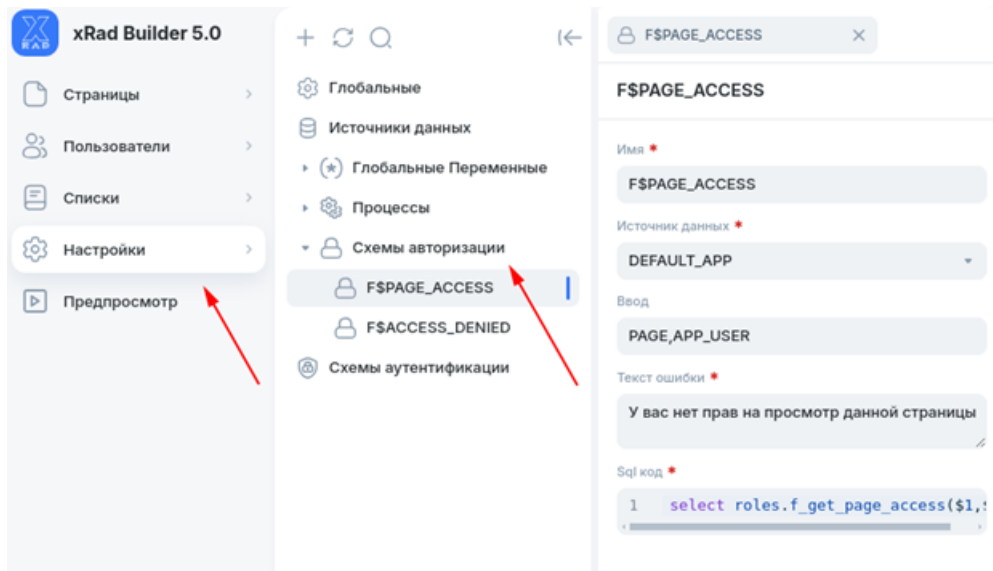
## 7.2.5 Схемы авторизации

### Общее описание

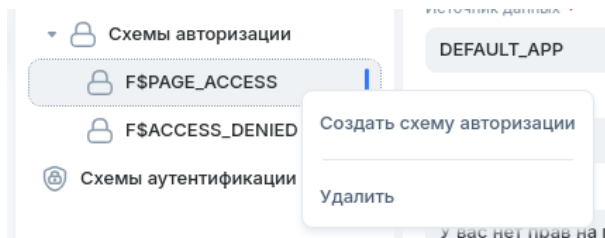
Схема авторизации определяет доступность страницы конкретному пользователю. Если авторизация проходит успешно, то пользователю отображается запрашиваемая страница. Если при авторизации происходит ошибка, то отображается сообщение об ошибке.

### Создание схемы авторизации

Список схем авторизации располагается в разделе «Настройки» -> «Схемы авторизации»:



При нажатии ПКМ на элементах списка «Схемы Авторизации» будет отображено контекстное меню с кнопками для создания новой схемы и удаления выделенной:



Примечание: 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:

Имя *	F\$PAGE_ACCES
Источник данных *	DEFAULT_APP
Ввод	PAGE,APP_USER
Текст ошибки *	У вас нет прав на просмотр данной страницы
Sql код *	1 select roles.f_get_page_access(\$

Атрибуты:

- Имя - обязательный атрибут, определяющий имя создаваемой схемы авторизации.
- Источник данных - обязательный атрибут, определяющий источник данных для 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 нажатия кнопки «Сохранить» и подтверждения введенных данных схема авторизации будет занесена в базу данных и доступна для просмотра, редактирования и удаления.

### 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 «Сохранить».

Для удаления схемы нажмите ПКМ по схеме, нажмите в меню кнопку «Удалить» и подтвердите действие.

## 7.2.6 Схемы аутентификации

### Общие сведения

Аутентификация - это процесс проверки подлинности 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 - доступ будет запрещен.

После успешной аутентификации пользователя устанавливается значение глобальной переменной APP\_USER (см. Глобальные переменные). Когда пользователь переходит со страницы на страницу, значение APP\_USER используется для идентификации пользователя.

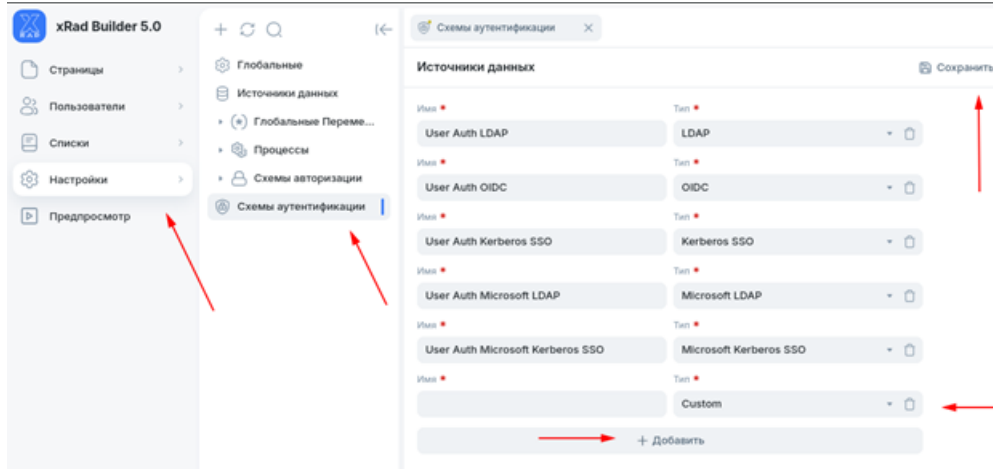
Сервер приложений PGHS поддерживает аутентификацию и авторизацию с использованием следующих схем:

- По логину и паролю - CUSTOM
- Microsoft LDAP
- Microsoft Kerberos SSO
- LDAP
- Kerberos SSO
- Open ID Connect

При разработке приложения в конструкторе XRAD разработчик определяет возможные схемы аутентификации посредством указания имени схемы и ее типа. Таким образом в приложении может быть множество схем одного типа, что позволяет гибко настраивать аутентификацию пользователей. Например, можно реализовать аутентификацию пользователей разных доменов Active Directory.

### Создание схемы аутентификации

Страница со схемами аутентификации располагается во вкладке «Настройки» -> «Схемы аутентификации», по нажатию кнопки «Добавить» можно добавить новую схему. Также любую схему можно удалить, нажав кнопку с иконкой корзины, для применения всех изменений необходимо нажать кнопку «Сохранить». Непосредственно параметры каждой схемы аутентификации задаются в файле auth\_config.json сервера приложений PGHS, с описанием данного файла и параметров можно ознакомиться в документации PGHS.



Примечание: для перехода на страницу необходимо авторизоваться от лица аккаунта с ролью доступа выше VIEW.

При старте сервер приложений PGHS загружает схемы из файла `auth_config.json` и сопоставляет их по имени и типу с параметрами в БД XRAD.

Для каждой страницы аутентификации разрабатываемого приложения можно создать свой набор способов аутентификации:

Пример 1:

/ xRad Builder 5.0 / 00 – Авторизация / 1 - Авторизация [↗](#)

- До загрузки страницы
- Обработка
- Валидации
  - login\_not\_empty
  - pass\_not\_empty
- После обработки
- Обратный вызов Ajax
- Загрузки
- Схемы аутентификации
  - BASE
  - AUTH**
  - AUTH
  - AUTH
  - AUTH
  - AUTH

### AUTH

Идентификация ▾

Тип аутентификации \*  
Microsoft LDAP ▾ ?

Имя \*  
User Auth Microsoft LDAP ▾ ?

Макет ▾

Последовательность \*  
20 ?

Параметры выполнения ▾

Источник данных \*  
DEFAULT\_APP ▾ ?

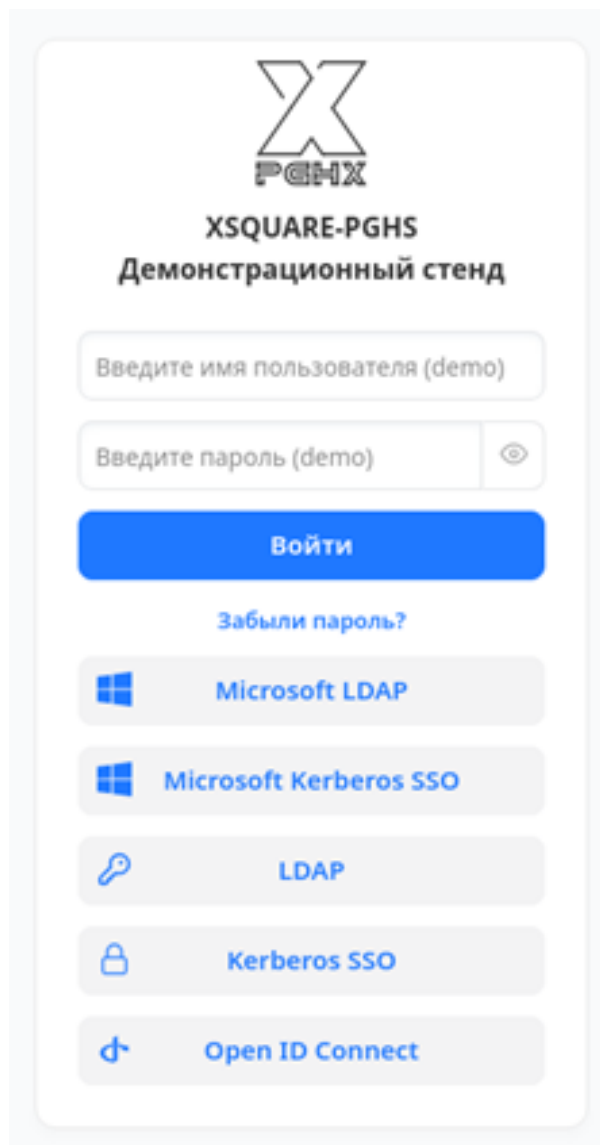
Источник ▾

Логин \*  
P7\_LOGIN ▾ ?

Пароль \*

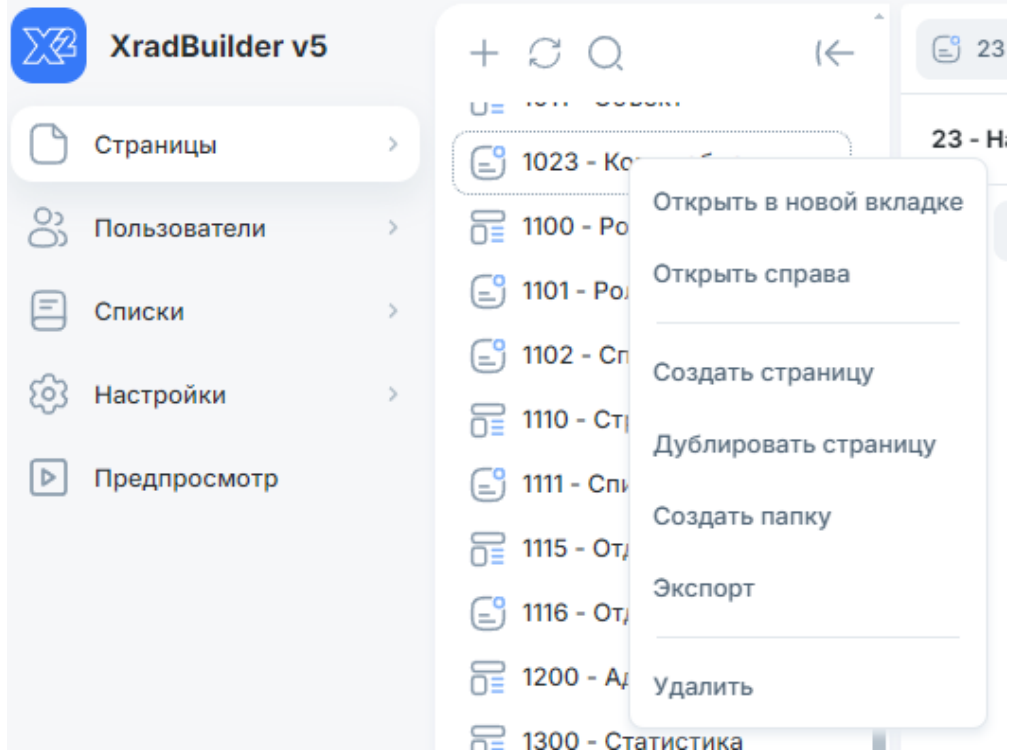
Пример 2:





### 7.3 Работа со страницами

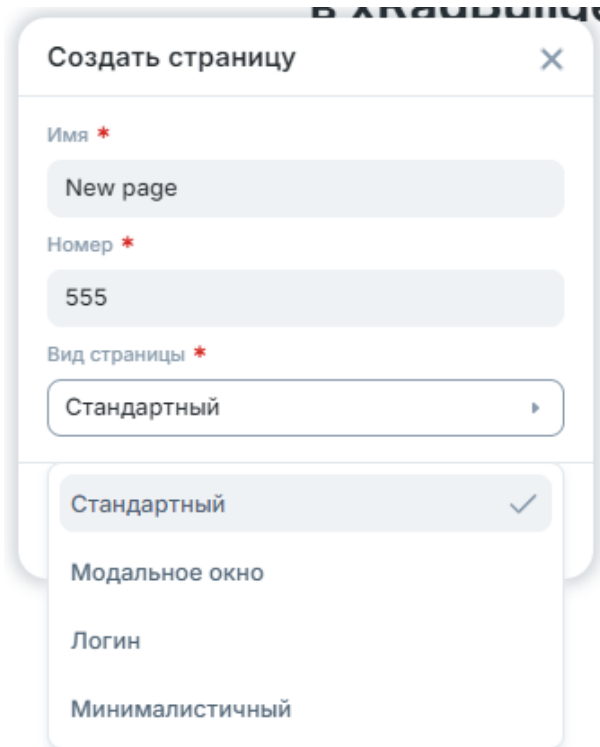
Страница — это основной блок разрабатываемого веб-приложения. Для того, чтобы отобразить список доступных страниц необходимо выбрать в главном меню раздел «Страницы». Все возможные действия со страницами доступны из контекстного меню, вызываемого по нажатию ПКМ. Разработчику доступно создание новой страницы, дублирование, экспорт и удаление текущей страницы, а также создание папок для группировки страниц.



### 7.3.1 Создание новой страницы

Создать новую страницу можно с помощью контекстного меню или с помощью кнопки «+». В окне создания страницы необходимо указать атрибуты новой страницы:

- Имя - строковое наименование страницы
- Номер – целочисленный идентификатор страницы
- Вид страницы - шаблон представления страницы.



Доступны следующие виды страниц:

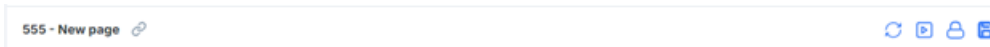
- Стандартный - страница с боковым меню и панелью навигации в заголовке страницы.
- Модальное окно – страница, которая отображается поверх родительской (вызывающей страницы).
- Логин - страница для аутентификации
- Минималистичный – простая страница без бокового меню и с заголовком без панели навигации.

После создания страницы - откроется «Редактор страниц».

### 7.3.2 Редактор страниц

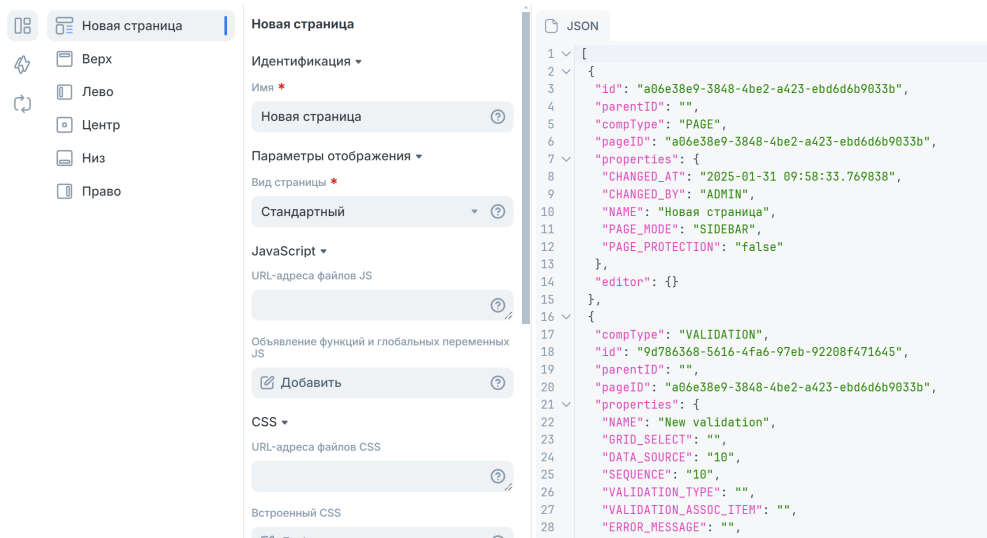
Вся работа по формированию новой страницы происходит в Редакторе страниц.

В верхней части Редактора страниц – расположена панель инструментов, где отображается идентификатор и имя страницы, кнопка копирования адреса страницы в буфер обмена, а также кнопки для обновления, предпросмотра, блокировки и сохранения изменений страницы.



Ниже панели инструментов располагаются три вертикальные области Редактора страниц, логически повторяющие концепцию всей среды разработки:

- Главное меню
- Список параметров
- Область редактирования кода



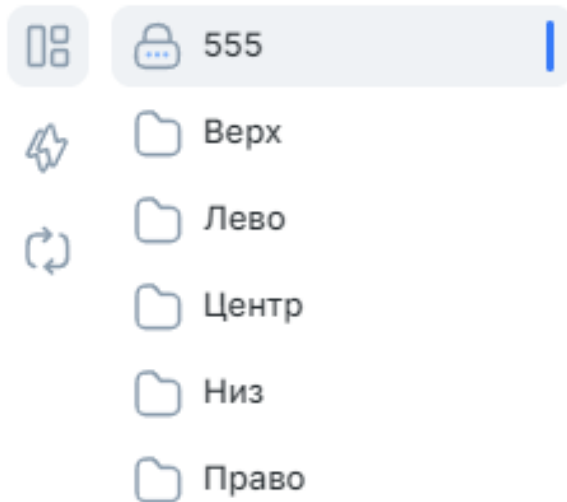
## Главное меню редактора страниц

Главное меню состоит из 3-х вкладок:

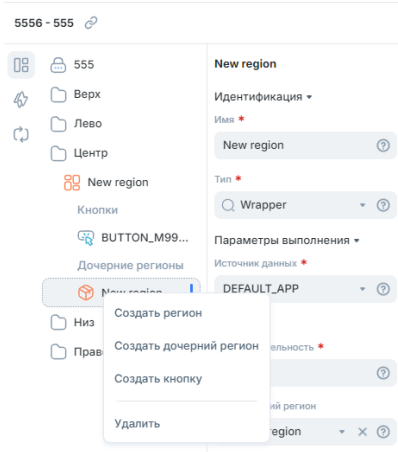
- Элементы
- События
- Процессинг

### Элементы

На вкладке Элементы отображаются компоненты, из которых состоит страница - регионы, элементы формы и кнопки.



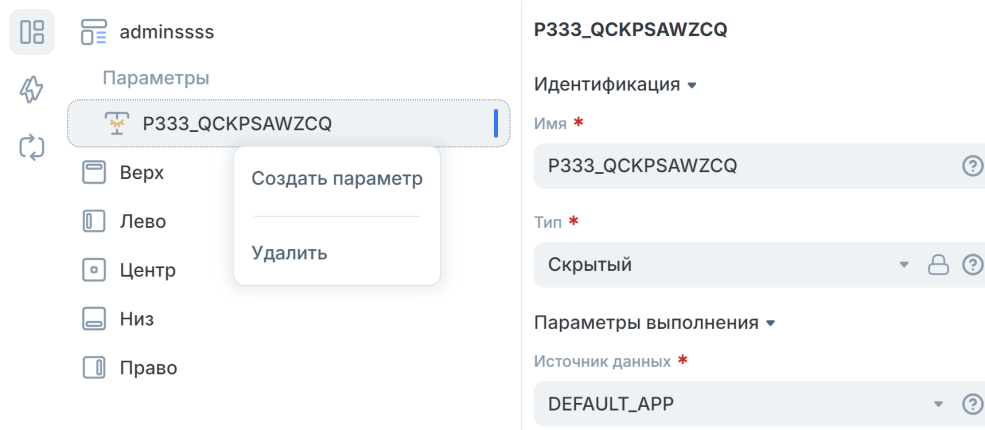
Компоненты могут быть размещены на уровне страницы в определенной области страницы (Верх, Лево, Центр, Низ, Право) или могут быть вложены в регионы в определенной позиции региона. Базовая конструкция, которая создается из контекстного меню – это регион. Регион может включать дочерние регионы, кнопки и другие визуальные компоненты, которые определяются типом региона.



Если не выбрана область страницы, то по умолчанию элементы создаются в центральной области (Центр).

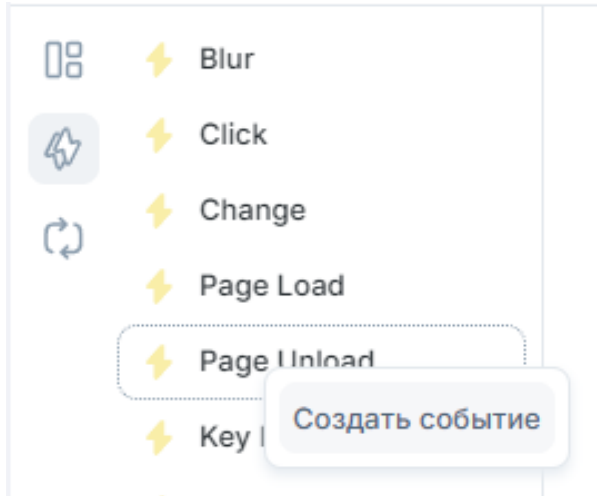
Примечание: кнопки могут быть созданы только внутри региона.

Также из контекстного меню имени страницы доступно создание параметров страницы. Данные параметры в отличие от глобальных параметров приложения доступны только в пределах конкретной страницы.



### 7.3.3 События

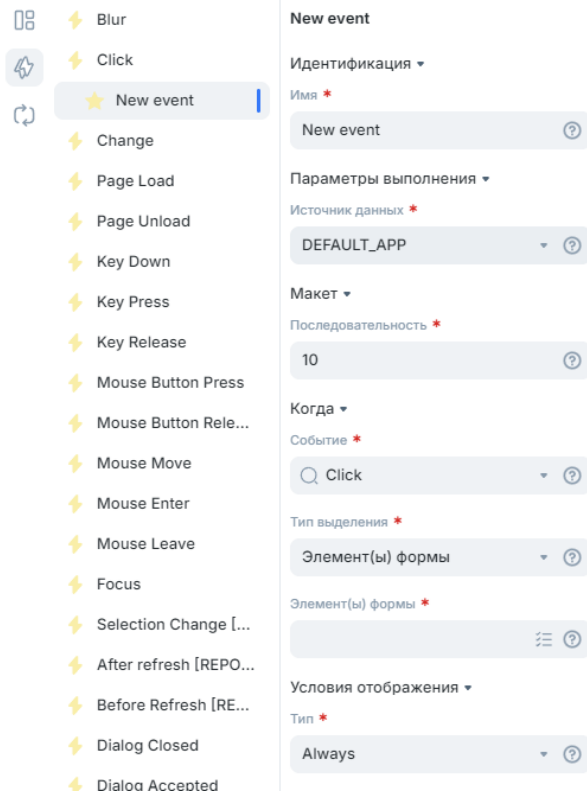
На вкладке **События** отображается список событий и действий, сгруппированных по типам возможных событий. Для создания нового события Разработчику необходимо вызвать контекстное меню с помощью ПКМ.



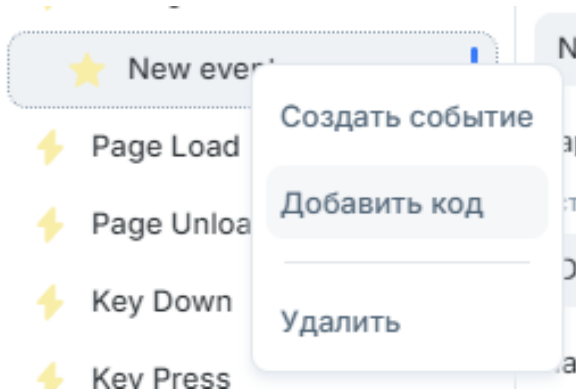
После создания события, необходимо настроить его параметры в области параметров:

- Имя – обязательный атрибут, определяющий наименование события.
- Источник данных – обязательный атрибут, определяющий источник данных для хранения элемента списка.
- Последовательность – обязательный атрибут, порядковый номер в соответствии с которым действие будет выполнено.
- Событие – обязательный атрибут, определяющий тип события.
- Тип выделения – обязательный атрибут, определяющий тип элемента страницы для срабатывания события.
- Тип условия отображения – обязательный атрибут, определяющий тип условия, которое должно быть выполнено для того, чтобы это событие могло быть вызвано.

Для применения изменений необходимо сохранить изменения через кнопку в панели инструментов.



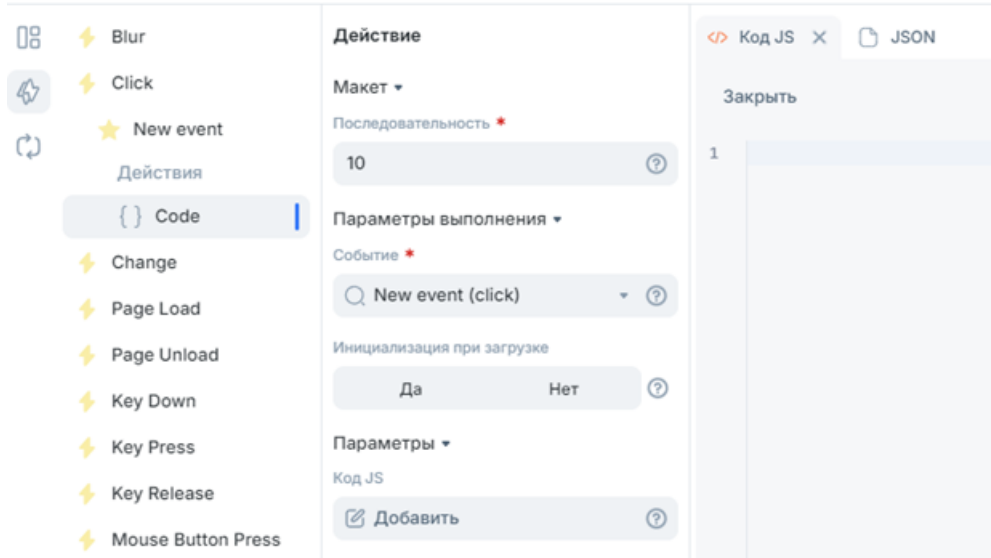
К событию можно привязать Действие в виде JavaScript кода. Создание действия осуществляется из контекстного меню по кнопке «Добавить код».



Для создаваемого действия необходимо также определить параметры:

- Последовательность – обязательный атрибут, порядковый номер в соответствии с которым действие будет выполнено.
- Событие – обязательный атрибут, определяет имя события, к которому привязывается код действия.
- Инициализация при загрузке – определяет выполнение действия при загрузке страницы.
- Код JS – обязательный атрибут, определяет JavaScript код, который будет выполнен при наступлении события.

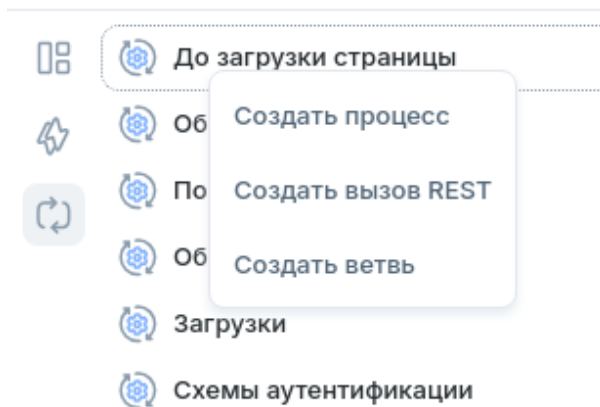
По нажатию кнопки «Добавить» в области вкладок редактора будет открыта вкладка «Код JS» с интерактивным редактором кода. Сохранение изменений JS кода производится в редакторе кода, а сохранение параметров с помощью кнопки в панели инструментов.



### 7.3.4 Процессинг

При работе приложения часто требуется выполнять запросы к базе данных. Для этих целей в XRAD предусмотрен механизм обработки запросов для каждой страницы - **процессинг**.

На вкладке **Процессинг** отображается список обработчиков запросов, сгруппированных по виду процессов или типу вызова обработчика. Состав контекстного меню доступного по нажатию ПКМ определяется типом группы:



Процессинг предоставляет следующие виды обработчиков запросов:

- Процесс - это основной механизм взаимодействия приложения с базой данных. Выполнение процессов происходит при поступлении от приложения различных запросов (Request). При появлении запроса от приложения все процессы соответствующего типа выполняются в порядке очереди (Sequence).
- Ветвь – процесс по выполнению условия (Server-side Condition) которого произойдет перенаправление на другую страницу приложения или ссылку.
- Валидация – процесс проверки корректности введенных пользователем данных.
- Вызов REST - процесс который позволяет вызвать сторонний REST API. Позволяет приложению обрабатывать и осуществлять запросы к сторонним сервисам. При этом вызов будет осуществлять сервер приложений PGHS.
- Процесс Data Grid - процесс для обработки измененных данных в DATAGRID.



- Загрузка – процесс, позволяющий загрузить файл из базы на клиент.
- Схема аутентификации – процесс аутентификации пользователя.

Процессы группируются также по типу процесса:

- До загрузки страницы – процессы, которые выполняются перед загрузкой страницы. Их основное предназначение – подготовить компоненты для взаимодействия с пользователем (например, заполнить поля формы значениями из БД). До загрузки страницы могут быть выполнены следующие виды обработчиков: процессы, вызовы REST и ветви.
- Обработка – процессы данного вида выполняются при отправке формы на сервер (submit).
- После обработки - процессы данного вида выполняются после обработки страницы. Например, переходы на другие страницы по нажатию кнопок.
- Обратный вызов Ajax - процессы данного вида позволяют взаимодействовать с БД по запросу от JS-скрипта.

Ответ от процессов Ajax должен обрабатываться разработчиком в JS-скрипте. Остальные виды процессов могут самостоятельно взаимодействовать с пользователем, сообщая об успешности выполнения запроса или о возникшей ошибке.

## Процессы

Для создания обработчика типа процесс необходимо выбрать соответствующий пункт контекстного меню по ПКМ и заполнить параметры создаваемого процесса.

The screenshot shows a configuration window titled "New process". On the left, a sidebar lists various process types, with "New process" selected. The main area contains the following fields and options:

- Идентификация** (dropdown): Имя \*
- Input field: New process
- Параметры выполнения** (dropdown): Тип процесса \*
- Dropdown menu: Перед загрузкой страницы
- Источник данных** (dropdown): Имя \*
- Dropdown menu: DEFAULT\_APP
- Input field: Имя запроса
- Принять модальный: Да Нет ?
- Фиксация TX \*: Да Нет ?

Параметр	Тип	Описание
Имя	Текст	Имя процесса.

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Тип процесса	Список	Тип процесса. Тип влияет на порядок вызова кода. В XRAD предусмотрены следующие типы процессов: <ul style="list-style-type: none"> <li>• До загрузки страницы</li> <li>• Обработка</li> <li>• Обратный вызов Ajax</li> </ul>
Источник данных	Список	Название источника данных для данного процесса.
Имя запроса	Текст	Определяет имя запроса. Если указано, то процесс будет выполняться только тогда, когда значение глобальной переменной REQUEST будет соответствовать введенному значению.
Принять модальный	Переключатель	Определяет необходимость закрытия модального окна после выполнения процесса.
Фиксация TX	Переключатель	Определяет необходимость вызывать COMMIT транзакции после успешного завершения процесса. После вызова COMMIT все изменения, внесенные в базу процессами, которые выполнялись до этого процесса, будут записаны в базу. После чего XRAD откроет новую транзакцию для выполнения последующих процессов.
Источник - SQL	Область текста	Определяет SQL-запрос, который необходимо выполнить.
Параметры запроса	Текст / Окно конструктора	Определяет элементы, которые необходимо передать в исполняемый запрос в качестве переменных.
Выходные параметры	Текст / Окно конструктора	Определяет выходные элементы. Если процессу необходимо установить значения сессионных переменных, необходимо указать их в данном поле.
Последовательность	Число	Определяет последовательность выполнения процесса в списке процессов одного типа.
Сообщение об успехе	Область текста	Определяет сообщение, которое будет отображено пользователю при успешном выполнении процесса. Если вызывается цепочка процессов - будет выведено последнее сообщение об успешном исполнении.
Сообщение об ошибке	Область текста	Определяет сообщение об ошибке.
Условия отображения - Тип	Список	Определяет тип условия отображения региона на странице. По умолчанию - Always. В зависимости от типа потребуются указать дополнительные параметры условия. Для Always и Never дополнительных условий не требуется.
Условия отображения - Первое условие	Область текста	Запрос в формате SQL. Параметр применим для следующих типов условий: <ul style="list-style-type: none"> <li>• Exists (SQL query returns at least one row). Если запрос вернул хотя бы одну строку, регион будет отображен на странице.</li> <li>• NOT Exists (SQL query returns no rows). Если запрос не вернул ни одной строки, регион будет отображен на странице.</li> </ul>
Условия отображения - Выражение SQL	Область текста	Логическое выражение на языке SQL. Если выражение вернёт значение истина, регион будет отображен на странице. Параметр применим для типа условия SQL Expression.

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Условия отображения - Первый вход	Текст / Окно кон- структора	<p>Задаёт список входных параметров для запроса SQL в поле “Первое условие”, либо “Выражение SQL”. Для каждой переменной подстановки в запросе должен быть определён входящий параметр. Может принимать значения глобальных переменных и элементов ввода и выбора страницы. Можно ввести как текстом, так и выбрать в конструкторе. Применим для типов условий</p> <ul style="list-style-type: none"> <li>• Exists (SQL query returns at least one row).</li> <li>• NOT Exists (SQL query returns no rows).</li> <li>• SQL Expression.</li> </ul>
Условия отображения - Элемент	Текст / Окно кон- структора	<p>Позволяет выбрать элемент, в зависимости от значения которого регион будет отображён или нет. Применим для следующих типов условий:</p> <ul style="list-style-type: none"> <li>• Value of Item / Column in Expression 1 Is NOT NULL. Регион будет отображён, если значение элемента не NULL.</li> <li>• Value of Item / Column in Expression 1 != Zero. Регион будет отображён, если значение элемента не равно 0.</li> <li>• Value of Item / Column in Expression 1 Is NULL. Регион будет отображён, если значение элемента NULL.</li> <li>• Value of Item / Column in Expression 1 Is NULL or Zero. Регион будет отображён, если значение элемента NULL, или равно нулю.</li> <li>• Value of Item / Column in Expression 1 = Zero. Регион будет отображён, если значение элемента равно 0.</li> <li>• Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero. Регион будет отображён, если значение элемента не NULL и не равно нулю.</li> </ul>

**Примечание:** при появлении запроса от приложения сначала выполняются все удовлетворяющие условиям запроса глобальные процессы в соответствии с их порядковыми номерами, а затем процессы текущей открытой страницы.

### 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 никогда.

## Управление транзакцией

При выполнении цепочки запросов к базе данных иногда возникает ситуация, когда возникновение 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 фиксации 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 действий, ошибки в которых не будут влиять на действия, выполненные в предыдущих группах.

## 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 Ассерта.

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

## Ветви

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 ветви - переходы между страницами (Branch).

Если ветвь расположена в блоке, 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 процессов при выполнении условия в блоке Server-side Condition. Для безусловного перенаправления в блоке Server-side Condition необходимо оставить значение Always, для отключения перенаправления – Never.

**New branch**

Идентификация ▾

Имя \*

New branch ?

Параметры выполнения ▾

Источник данных \*

DEFAULT\_APP ▾ ?

Место выполнения \*

Перед загрузкой стра... ▾ ?

Макет ▾

Последовательность \*

10 ?

Поведение ▾

Тип \*

URL-адрес перенапра... ▾ ?

Ссылка ▾

Ссылка \*

☰ ?

Рассмотрим уникальные параметры для процессов типа ветвь:

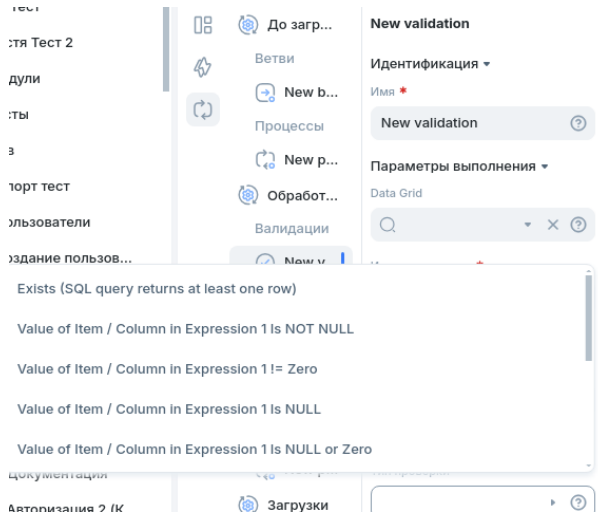
Параметр	Тип	Описание
Имя	Текст	Имя ветви.
Источник данных	Список	Определяет источник данных для данной ветви.
Место выполнения	Список	Определяет место выполнения перехода. <ul style="list-style-type: none"> <li>• Перед загрузкой страницы - переход будет выполнен перед загрузкой страницы. Данный пункт удобно использовать в случае, если необходимо реализовать перенаправления пользователя в зависимости от каких-либо условий.</li> <li>• После обработки - перенаправление будет вызвано после завершения выполнения процессов.</li> </ul>
Последовательность	Число	Определяет последовательность выполнения процесса в списке процессов одного типа.
Поведение - Тип	Список	Определяет тип перехода страницы для выполнения: <ul style="list-style-type: none"> <li>• URL-адрес перенаправления</li> <li>• Функция, возвращающая URL</li> </ul>
Ссылка	Текст / Окно конструктора	Определяет ссылку на страницу, если задан тип перехода «URL-адрес перенаправления»
Источник - SQL	Область текста	Определяет SQL-запрос, который возвращает URL
Параметры запроса	Текст / Окно конструктора	Определяет элементы, которые необходимо передать в исполняемый запрос в качестве переменных.
Выходные параметры	Текст / Окно конструктора	Определяет выходные элементы. Если процессу необходимо установить значения сессионных переменных, необходимо указать их в данном поле.
Условия отображения - Тип	Список	Определяет тип условия отображения региона на странице. По умолчанию - Always. В зависимости от типа потребуется указать дополнительные параметры условия. Для Always и Never дополнительных условий не требуется.

## Валидация

Для создания полноценного приложения требуется взаимодействие с пользователем. Все действия и данные пользователя необходимо сохранять в базу данных. Однако, пользователь может умышленно или непреднамеренно вводить некорректные данные. Для того чтобы этого избежать, в XRAD предусмотрена проверка данных – валидация. Валидация – это проверка корректности введенных пользователем данных в поля ввода формы.

## Типы валидаторов

В XRAD процесс валидации – это вызов различных функций и процедур (далее - валидаторы), которые выполняются перед тем как данные будут отправлены на сервер для обработки.



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 круглые скобки «(» «)». 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;

- 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.

Также имеется отдельный тип валидации – это флаг, который определяет необходимость 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 данными.

### Ошибка валидации

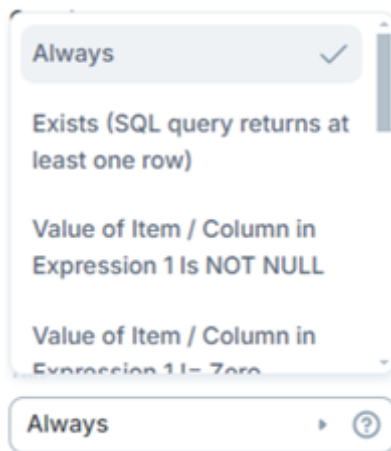
В случае возникновения исключения в любом из присутствующих на форме валидаторе отправка на сервер блокируется и пользователю выдается заданное разработчиком предупреждение. Если валидатор привязан к полю ввода, то дополнительно будет подсвечено поле ввода, на котором произошло исключение.

### Системные валидаторы

Если поле отмечено как обязательное, но оно не заполнено, то в этом случае сработает системный валидатор и пользователь будет уведомлен о необходимости заполнить обязательные поля всплывающей подсказкой. Если на поле ввода требуется добавить дополнительную проверку, то необходимо создать валидатор для данного поля. В этом случае сначала будут вызваны системные валидаторы, а затем пользовательские.

### Условия для выполнения валидации

Так как валидации 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 валидации.



Разработчику доступно 11 типов условий выполнения валидатора:

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

## Вызовы REST

Вызов REST - процесс, который позволяет вызвать внешний REST API.

Рассмотрим уникальные для данного типа процесса параметры:

Параметр	Тип	Описание
Источник - URL	Текст	Определяет URL-адрес для выполнения запроса. URL необходимо указывать полностью включая протокол.
HTTP-метод	Список	Определяет HTTP-метод для выполнения запроса: <ul style="list-style-type: none"> <li>• GET</li> <li>• POST</li> <li>• PUT</li> <li>• DELETE</li> <li>• PATCH</li> </ul>
HTTP-заголовки	Окно конструктора	Определяет заголовки HTTP. Заголовки могут включать в себя статические значения, а также принимать значения элементов формы.
Параметр запроса URL	Окно конструктора	Определяет параметры запроса, которые будут добавлены в URL. Значения параметров могут быть как статическими, так и взятыми из элемента ввода.
Тип тела запроса	Список	Определяет тип тела запроса. Возможные варианты: <ul style="list-style-type: none"> <li>• Отсутствует</li> <li>• Статичное значение</li> <li>• Form-encoded</li> <li>• SQL</li> </ul>
Обработка ответа	Список	Определяет тип обработчика ответа сервиса: <ul style="list-style-type: none"> <li>• Отсутствует</li> <li>• Назначить переменные</li> <li>• SQL-обработчик</li> </ul>
Назначение переменных	Окно конструктора	Определяет соответствие элементов формы и элементов ответа в формате JSON. В качестве аргумента обработчика ответа можно указать путь JMES. В случае если путь вернет просто значение JSON (строка, число, булево значение) элемент будет содержать это значение. Если JMES путь возвращает составное значение (объект или массив) в элемент будет передано вернувшееся значение в формате JSON.
Обработка SQL	Окно конструктора	Определяет SQL-обработчик.
Входные параметры	Текст / Окно конструктора	Определяет элементы, которые необходимо передать в SQL-запрос в качестве переменных.

## Загрузки

Процесс загрузки файла от веб-приложения. При создании процесса загрузки необходимо задать параметры:

Параметр	Тип	Описание
Имя	Текст	Имя процесса загрузки.
Источник данных	Список	Определяет источник данных для данного процесса.
Имя запроса	Текст	Определяет имя запроса. Если указано, то процесс будет выполняться только тогда, когда значение глобальной переменной REQUEST будет соответствовать введенному значению.
Последовательность	Число	Определяет последовательность выполнения процесса в списке процессов одного типа.
Источник - SQL	Область текста	Определяет SQL-запрос, который возвращает URL.
Колонка с наименованием	Список	Определяет колонку, содержащую название файла.
Колонка с файлом	Список	Определяет колонку, содержащую данные файла.
Mime Type Колонка	Список	Определяет колонку содержащую mime тип файла.
Content Disposition	Список	Определяет содержание заголовка Content-Disposition: <ul style="list-style-type: none"> <li>• Attachment</li> <li>• Inline</li> </ul> Заголовок отвечает за то как необходимо отобразить содержимое файла в браузере. Если указано attachment - файл будет загружен на компьютер пользователя. Если указано inline – содержимое файла будет отображено в браузере.

### Схемы аутентификации

Для ограничения доступа к непубличным страницам приложения используются страницы аутентификации. Способы аутентификации определяются процессами аутентификации, которые в свою очередь ссылаются на глобальные схемы аутентификации.

Рассмотрим уникальные параметры процесса аутентификации:

Параметр	Тип	Описание
Тип аутентификации	Список	Определяет тип аутентификации. Поддерживаемые типы аутентификации: <ul style="list-style-type: none"> <li>• Custom</li> <li>• Ldap</li> <li>• Microsoft Ldap</li> <li>• Kerberos SSO</li> <li>• Microsoft Kerberos SSO</li> <li>• OIDC</li> </ul>
Имя	Список	Определяет имя процесса аутентификации из списка глобальных схем аутентификации в соответствии с выбранным типом.
Источник - Логин	Список	Определяет элемент формы, который будет использоваться для ввода логина. Атрибут доступен для схем Ldap и Microsoft Ldap.
Функция пост-входа	Область текста	Определяет SQL-запрос, который будет выполнен после успешной аутентификации.

## Параметры страницы

Для каждой создаваемой страницы доступен определенный набор параметров:

- Имя – обязательный атрибут, определяющий наименование страницы.
- Вид страницы – обязательный атрибут, определяющий шаблон представления страницы. Доступны следующие варианты шаблонов: стандартный, модальное окно, логин, минималистичный (описание в разделе создание страницы)
- URL-адреса файлов JS – список URL-адресов файлов JavaScript с кодом, который будет загружен для данной страницы.

Объявление функций и глобальных переменных JS – используется для определения JS переменных и функций на уровне страницы.

- URL-адреса файлов CSS – список URL-адресов файлов CSS, который будет загружен для данной страницы.
- Встроенный CSS - используется для определения стилей на уровне страницы.
- Ширина - Определяет ширину модального окна. Доступно только для режима страницы «Модальное окно».
- Схема авторизации - определяет доступность страницы конкретному пользователю (См. раздел «Схемы авторизации»).
- Публичная страница - определяет необходимость аутентификации пользователя (См. раздел «Схемы аутентификации»).
- Защита страницы – определяет необходимость защиты параметров в запросе страницы с помощью контрольной суммы.

**New page**

Идентификация ▾

Имя \*

?

Параметры отображения ▾

Вид страницы \*

?

JavaScript ▾

URL-адреса файлов JS

?

Объявление функций и глобальных переменных JS

?

CSS ▾

URL-адреса файлов CSS

?

Встроенный CSS

?

Параметры окна ▾

Ширина

?

Безопасность ▾

Схема авторизации

?

Публичная страница

?

Защита страницы \*

?

### 7.3.5 Модальная страница

Модальная страница представляет собой дочернее окно поверх родительского, расположенное в том же окне браузера. Модальная страница остается активной до тех пор, пока пользователь не завершит работу с ней и не закроет ее. Пользователь не может взаимодействовать с родительской частью страницы, до тех пор, пока модальная страница не будет закрыта.

#### Вызов модальной страницы

Вызвать модальную страницу возможно 2 способами:

- По ссылке - данный способ не предусматривает обработку результата выполнения модальной страницы.
- Из кода Javascript

Пример вызова модальной страницы с помощью JavaScript:

```
jsAPI.modal.open({
  page: {
    src: '1010',
    query: {
      P1010_ID: jsAPI.getItem('P1000_ID'),
      clear: '1010'
    }
  },
  title: 'Моя модальная страница'
});
```

JavaScript-код вызова можно использовать как значение ссылки через действие Перенаправление на страницу, так и через событие типа Click на выбранную кнопку.

### Заккрытие модальной страницы (событие «Отмена»)

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

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

### Обновление данных на родительской форме

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

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

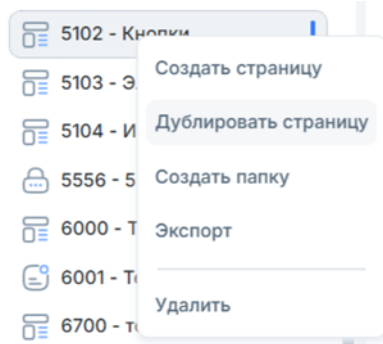
Обработку результата выполнения модальной страницы можно также определить при вызове модальной страницы в блоках onClose, onAccept или onDecline:

```
jsAPI.modal.open({
  page: {
    src: '1010',
    query: {
      P1010_ID: jsAPI.getItem('P1000_ID'),
      clear: '1010'
    }
  },
  title: 'Моя модальная страница'
}, {
  onAccept: function(e, i) {
    jsAPI.setItem('P1000_ID', i.P1010_ID);
  },
  onClose: function(e, i) {
    alert(3);
  }
});
```

### Дублирование, удаление, группировка и экспорт страницы

Помимо создания страницы из контекстного меню доступного по ПКМ в списке страниц доступны следующие действия:

- Дублировать страницу – создает копию выбранной страницы. Для новой страницы необходимо указать новое имя, порядковый номер и вид.
- Создать папку – создает папку с заданным именем, которая используется для группировки страниц.
- Экспорт – экспортирует страницу в виде файла \*.sql с SQL-кодом.
- Удалить – удаляет выбранную страницу.



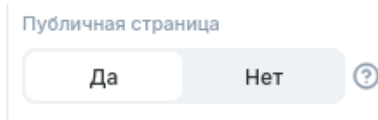
### 7.3.6 Безопасность страниц

Далее будут рассмотрены механизмы для обеспечения безопасности и защиты страниц.

#### Публичная страница

Публичная страница – это страница, для которой не требуется аутентификация и авторизация пользователя.

Установить атрибут публичности можно в настройках страницы через свойство «Публичная страница».



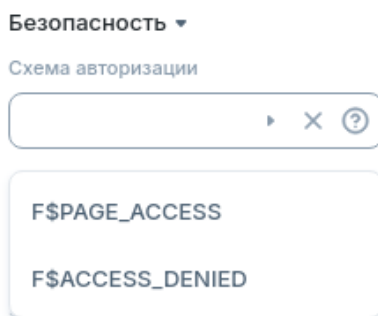
Если свойство «Публичная страница» не установлено или установлено в состояние «Нет», то при запросе страницы отобразится страница для аутентификации пользователя.

Данное свойство будет применено к странице после сохранения изменений через кнопку «Сохранить».

#### Схемы авторизации

Определить доступность страницы конкретному пользователю можно в редакторе страниц через свойство «Безопасность - Схема авторизации».

В выпадающем списке можно увидеть доступные схемы авторизации.



Выбранная схема авторизации будет применена к странице после сохранения изменений через кнопку «Сохранить».

Если у пользователя нет прав на просмотр страницы, то будет отображена сообщение об ошибке, указанное в схеме авторизации.

### Контрольные суммы

Контрольная сумма используется в случае, когда необходимо обеспечить защиту параметров страницы в URL от изменения. Контрольная сумма формируется для каждой ссылки для конкретного пользователя. Изменение параметров в URL вызовет ошибку проверки контрольной суммы.

Если страница защищена с помощью контрольной суммы, в URL появляется дополнительный GET-параметр «cs» 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.

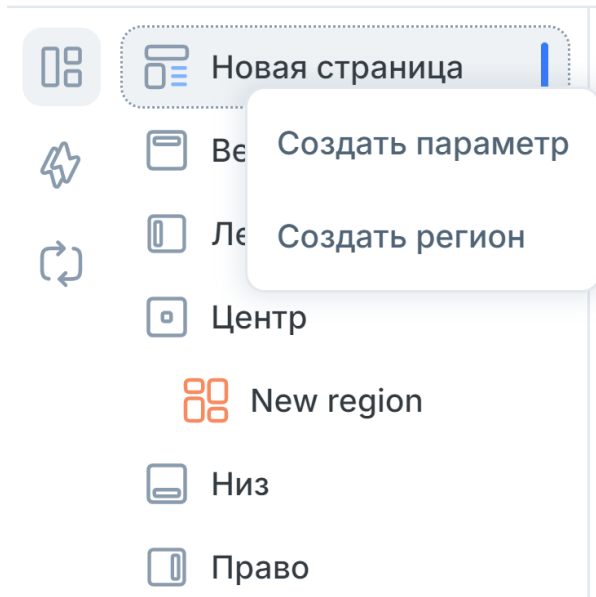
Защита страницы \*

Да  Нет

## 7.4 Визуальные компоненты

Визуальные компоненты - ключевые элементы пользовательского интерфейса PGHS. Большинство страниц веб-приложений состоит из множества компонентов, которые разработчик группирует в контейнеры. В XRAD таким универсальным контейнером является **регион**. Регионы могут содержать в себе поля для ввода, кнопки и другие регионы, либо непосредственно выводить данные в виде отчётов, графиков и других способов предоставления информации. Регионы по умолчанию соответствуют типу «обертка» (WRAPPER) и являются базовыми компонентами страницы. Преобразовать регион в другой визуальный компонент можно изменив его тип. Новые регионы создаются через контекстное меню по ПКМ.



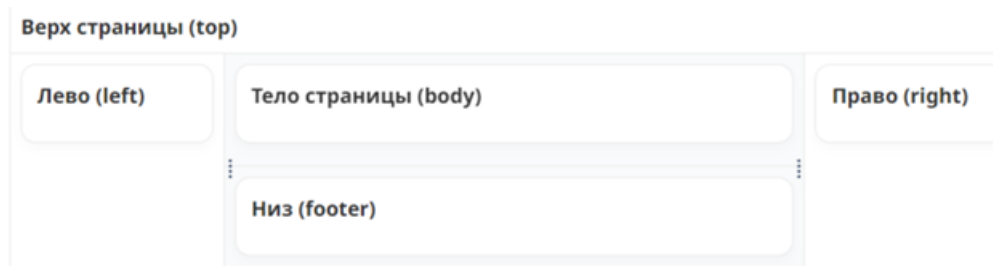


### 7.4.1 Расположение компонентов на странице

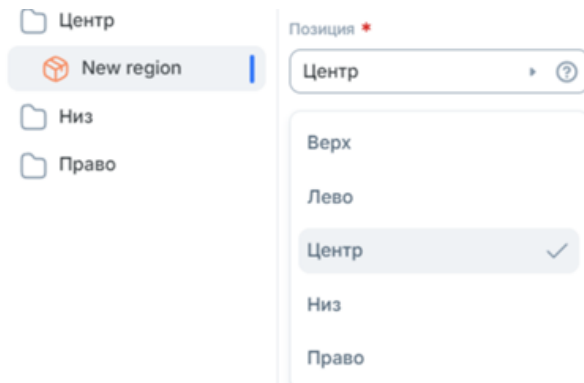
Для региона обязательным параметром является положение на странице.

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). Область для размещения общей информации. На страницах вида модальное окно здесь располагаются кнопки.

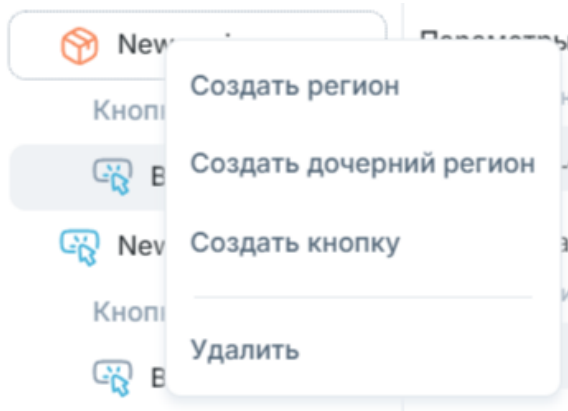


После создания, регион можно переместить в другую область страницы, поменяв ему в параметрах свойство “Позиция”(Position).



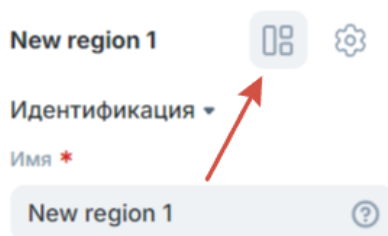
## 7.5 Регионы

Регионы могут быть родительскими, дочерними или независимыми по отношению друг к другу. Внутри региона может быть создано множество других регионов – дочерних, с вложенностью без ограничения, а также кнопки.



### 7.5.1 Параметры регионов

После создания региона необходимо настроить его параметры.



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

Параметр	Тип	Описание
Имя	Текст	Имя региона. Определяет отображение региона в дереве объектов и настройках других компонентов, а также выводится в заголовке региона на странице приложения.

продолжается на следующей странице

Таблица 2 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Тип	Список	Тип региона. Определяет набор параметров и наличие особых атрибутов региона, отображение региона на странице приложения.
Источник данных	Список	Название источника данных для данного компонента.
Последовательность	Число	Порядковый номер региона. Определяет положение региона на сетке внутри области расположения / родительского региона.
Родительский регион	Список	В списке на выбор - страница и регионы, не являющиеся дочерними для данного. В случае выбора страницы, регион будет отображен внутри области расположения, иначе внутри выбранного региона.
Позиция	Список	Выбор области расположения региона.
Количество столбцов	Число	Выбор Automatic, либо число 1-12. Указывает количество ячеек сетки, занимаемых регионом. По умолчанию 12.
Начать новую строку	Переключатель	Определяет положение на сетке по вертикали.
Настройки темы	Окно настроек	Настройка темы региона.
Классы компонентов	Текст	Классы дочерних компонентов внутри региона.
Классы css	Текст	Классы css региона.
Показать заголовок	Переключатель	Определяет, будет ли отображено имя региона в заголовке и визуальная область заголовка (важно для позиции кнопок - "в заголовке").
Складной	Переключатель	Определяет, будет ли регион сворачиваемым.
Развернут	Переключатель	Определяет состояние региона при открытии страницы. Если выбрано, то регион будет отображаться в развернутом состоянии. Параметр доступен только если "Складной" = "да".
Запомнить состояние	Переключатель	Определяет сохранение выбранной пользователем вкладки. При входе на страницу будет отображаться последняя выбранная пользователем вкладка. Параметр доступен только если "Складной" = "да".
Статический идентификатор	Текст	Задаёт идентификатор для региона для дальнейшего обращения к нему из jsAPI.
Источник - Тип	Список	Определяет тип источника данных. Применим для регионов типа HTML. На выбор два варианта: <ul style="list-style-type: none"> <li>• Статический. В поле исходный текст потребуется ввести текст HTML.</li> <li>• SQL. В поле SQL потребуется ввести запрос к БД.</li> </ul>
Источник HTML код	Область текста	HTML-код. Применим для регионов типа HTML с выбранным типом источника - Статический.
Источник - SQL	Область текста	Запрос к БД. Формат вывода определяется типом региона и его атрибутами. Параметр применим для следующих видов регионов: HTML (с выбранным типом источника SQL) <ul style="list-style-type: none"> <li>• REPORT</li> <li>• TREE</li> <li>• CHAT</li> <li>• CHART</li> <li>• CALENDAR</li> <li>• DATAGRID</li> </ul>

продолжается на следующей странице

Таблица 2 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Источник - Исходный вход	Текст / Окно конструктора	Определяет список входных параметров для запроса SQL в поле источник - SQL. Применим для видов регионов, для которых существует параметр Источник - SQL (см. строку выше). Для каждой переменной подстановки в запросе должен быть определён входящий параметр. Может принимать значения глобальных переменных и элементов ввода и выбора страницы. Можно ввести как текстом, так и выбрать в конструкторе.
Источник - список	Список	Привязывает список к региону. На выбор будут представлены все созданные в приложении списки. Параметр применим для следующих видов регионов: <ul style="list-style-type: none"> <li>• BREADCRUMB</li> <li>• PAGE NAVIGATION</li> <li>• CARDS</li> <li>• WIZARD</li> <li>• TILES</li> </ul>
Условия отображения - Тип	Список	Определяет тип условия отображения региона на странице. По умолчанию - Always. В зависимости от типа потребуются указать дополнительные параметры условия. Для Always и Never дополнительных условий не потребуется.
Условия отображения - Первое условие	Область текста	Запрос в формате SQL. Параметр применим для следующих типов условий: <ul style="list-style-type: none"> <li>• Exists (SQL query returns at least one row). Если запрос вернул хотя бы одну строку, регион будет отображен на странице.</li> <li>• NOT Exists (SQL query returns no rows). Если запрос не вернул ни одной строки, регион будет отображён на странице.</li> </ul>
Условия отображения - Выражение SQL	Область текста	Логическое выражение на языке SQL. Если выражение вернёт значение истина, регион будет отображён на странице. Параметр применим для типа условия SQL Expression.
Условия отображения - Первый вход	Текст / Окно конструктора	Задаёт список входных параметров для запроса SQL в поле “Первое условие”, либо “Выражение SQL”. Для каждой переменной подстановки в запросе должен быть определён входящий параметр. Может принимать значения глобальных переменных и элементов ввода и выбора страницы. Можно ввести как текстом, так и выбрать в конструкторе. Применим для типов условий: <ul style="list-style-type: none"> <li>• Exists (SQL query returns at least one row).</li> <li>• NOT Exists (SQL query returns no rows).</li> <li>• SQL Expression.</li> </ul>

продолжается на следующей странице

Таблица 2 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Условия отображения - Элемент	Текст / Окно кон- структора	<p>Позволяет выбрать элемент, в зависимости от значения которого регион будет отображён или нет. Применим для следующих типов условий:</p> <ul style="list-style-type: none"> <li>• Value of Item / Column in Expression 1 Is NOT NULL. Регион будет отображён, если значение элемента не NULL.</li> <li>• Value of Item / Column in Expression 1 != Zero. Регион будет отображён, если значение элемента не равно 0.</li> <li>• Value of Item / Column in Expression 1 Is NULL. Регион будет отображён, если значение элемента NULL.</li> <li>• Value of Item / Column in Expression 1 Is NULL or Zero. Регион будет отображён, если значение элемента NULL, или равно нулю.</li> <li>• Value of Item / Column in Expression 1 = Zero. Регион будет отображён, если значение элемента равно 0.</li> <li>• Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero. Регион будет отображён, если значение элемента не NULL и не равно нулю.</li> </ul>

## 7.5.2 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.

Стандартная страница представляет собой сетку из 12 колонок:



В XRAD положение компонента на сетке регулируется следующими параметрами:

- Количество 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, при попытке открыть страницу отобразится ошибка рендеринга.

В случае, если сумма компонентов одной строки составит менее 12, оставшиеся ячейки заполнятся, если хотя

бы у одного региона в строку указано значение Automatic в количестве столбцов, иначе останется свободное пространство.


### 7.5.3 Атрибуты регионов

Для некоторых типов регионов также необходимо указать специфичные атрибуты компонента:


**New region**  

Параметры отображения ▾

Тема \*

Базовая ▾ 

Колонки \*

2 ▾ 

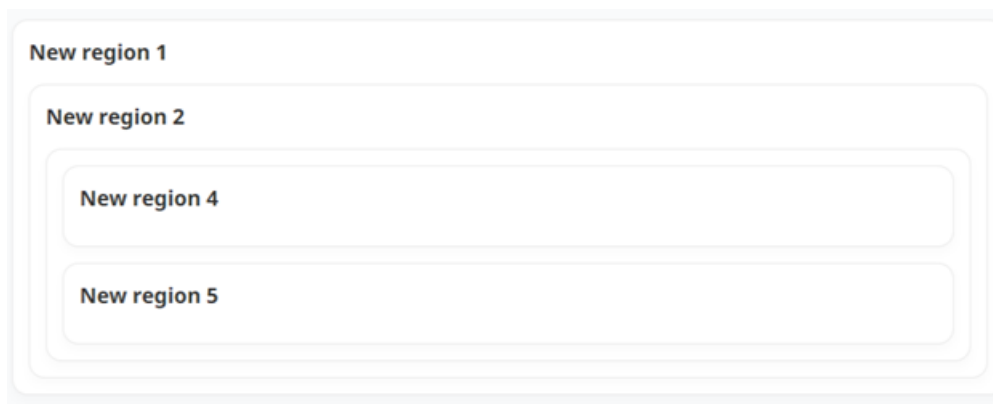
## 7.6 Компоненты

### 7.6.1 Обертка (WRAPPER)

WRAPPER - основной контейнер для регионов. Используется для группировки и зонирования компонентов или дочерних регионов.

Основными параметрами регионов типа WRAPPER является позиция на странице и отображение имени региона.

Из регионов данного типа можно составлять блоки любой сложности и вложенности.



#### Настройка компонента

Параметры компонента – стандартные для региона.

### 7.6.2 Хлебные крошки (BREADCRUMB)

BREADCRUMB - компонент для вывода навигационной цепочки и заголовка страницы на основе списка.

Для создания компонента проделайте следующие шаги:

1. Выберите страницу.
2. Создайте новый регион в позиции “верх”.

3. Укажите тип региона - BREADCRUMB.
4. Создайте список с типом BREADCRUMB в разделе «Списки».
5. Укажите созданный 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

Главная > Дополнительно

## Отчёты

В списке типа BREADCRUMB создайте новый элемент.

В настройках элемента введите:

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

### Настройка компонента

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

## 7.6.3 Контейнер для кнопок (BUTTONS)

BUTTONS - компонент для размещения кнопок.

### Настройка компонента

Для данного компонента все параметры – стандартные для региона.

Рассмотрим подробнее кнопки.

Кнопки могут размещаться в теле любых контейнеров, на форме, а также в заголовках и подвалах любых регионов и служат для вызова процесса, события или перехода на другую страницу, а также для отображения выпадающих списков.

### Параметры кнопок

Параметр	Тип	Описание
Имя	Текст	Имя кнопки
Источник	дан- ных Список	Название источника данных для данного компонента
Наименование	Текст	Текст на кнопке
Последователь- ность	Число	Определяет порядок расположения кнопки на родительском регионе в рамках местоположения и позиции.

продолжается на следующей странице

Таблица 3 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Родительский регион	Список	Определяет регион, на котором кнопка будет отображена
Местоположение	Список	<p>Способ определения позиции кнопки. Для всех родительских регионов, кроме формы имеет одно значение на выбор - REGION. Всего вариантов три:</p> <ul style="list-style-type: none"> <li>• REGION. Вариант по умолчанию. Кнопка располагается в рамках региона на выбранной позиции по заданному порядку относительно других кнопок позиции.</li> <li>• ITEM. Применимо при родительском регионе типа FORM. Позиция кнопки определяется относительно привязанного элемента по заданному порядку среди других кнопок той же позиции того же связанного элемента.</li> <li>• FORM. Применимо при родительском регионе типа FORM. Кнопка располагается на форме, порядок определяется порядком среди других элементов формы</li> </ul>
Позиция	Список	<p>Определяет относительное расположение кнопки. Применимо для вариантов местоположения REGION и ITEM. Для местоположения REGION доступны следующие варианты:</p> <ul style="list-style-type: none"> <li>• Bottom Fluid. Кнопка растянется на всю ширину вдоль нижней границы региона.</li> <li>• Bottom Left. Кнопка расположится под основным содержимым региона в левой части.</li> <li>• Bottom Right. Кнопка расположится под основным содержимым региона в правой части.</li> <li>• Left Body. Кнопка расположится слева от основного содержимого в теле региона.</li> <li>• Right Body. Кнопка расположится справа от основного содержимого в теле региона.</li> <li>• Top Left. Кнопка расположится в левой части заголовка региона перед его названием. Отобразится только если включен параметр региона “Показать заголовок”.</li> <li>• Top Right. Кнопка расположится в правой части заголовка региона. Отобразится только если включен параметр региона “Показать заголовок”.</li> </ul> <p>Для местоположения ITEM варианта два:</p> <ul style="list-style-type: none"> <li>• Left of Item. Кнопка прикрепится к левой границе элемента ввода.</li> <li>• Right of Item. Кнопка прикрепится к правой границе элемента ввода.</li> </ul>
Действие	Список	<p>Поведение при нажатии на кнопку. На выбор четыре варианта:</p> <ul style="list-style-type: none"> <li>• Отправить страницу. Отправляет форму на сервер, передаёт в процесс равный параметру статический идентификатор кнопки.</li> <li>• Перенаправление на Страницу. направляет на страницу, указанную в поле связь.</li> <li>• Вызывает событие Click.</li> <li>• Открыть раскрывающийся список. Нажатие на кнопку вызовет контекстное меню, содержание которого определяется привязанным списком.</li> </ul>

продолжается на следующей странице



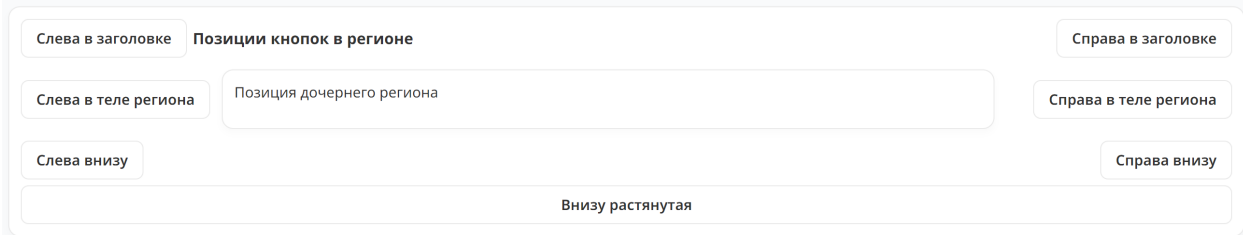
Таблица 3 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Ссылка	Текст / Окно кон- структора	Позволяет задать ссылку на другую страницу, вызвать динамический скрипт, либо настроить в конструкторе ссылку на страницу внутри приложения. Применимо для действия “Перенаправление на Страницу”.
Источник - Список	Список	Определяет список, отображаемый в контекстном меню. Применимо для действия “Открыть раскрывающийся список”.
Классы	Текст	Перечень CSS-классов кнопки. Здесь определяется цвет кнопки, иконка и др.
Статический идентификатор	Текст	Идентификатор кнопки, передаваемый в переменную REQUEST при нажатии на неё.
Условия отображения - Тип	Список	Определяет тип условия отображения кнопки. По умолчанию - Always. В зависимости от типа потребуется указать дополнительные параметры условия. Для Always и Never дополнительных условий не потребуется.
Условия отображения - Первое условие	Область текста	Запрос в формате SQL. Параметр применим для следующих типов условий: <ul style="list-style-type: none"> <li>Exists (SQL query returns at least one row). Если запрос вернул хотя бы одну строку, кнопка будет отображена.</li> <li>NOT Exists (SQL query returns no rows). Если запрос не вернул ни одной строки, кнопка будет отображена.</li> </ul>
Условия отображения - Выражение SQL	Область текста	Логическое выражение на языке SQL. Если выражение вернёт значение истина, кнопка будет отображена. Параметр применим для типа условия SQL Expression.
Условия отображения - Первый вход	Текст / Окно кон- структора	Задаёт список входных параметров для запроса SQL в поле “Первое условие”, либо “Выражение SQL”. Для каждой переменной подстановки в запросе должен быть определён входящий параметр. Может принимать значения глобальных переменных, элементов ввода и выбора страницы. Можно ввести как текстом, так и выбрать в конструкторе. Применим для типов условий: <ul style="list-style-type: none"> <li>Exists (SQL query returns at least one row).</li> <li>NOT Exists (SQL query returns no rows).</li> <li>SQL Expression</li> </ul>
Условия отображения - Элемент	Текст / Окно кон- структора	Позволяет выбрать элемент, в зависимости от значения которого столбец будет отображён или нет. Применим для следующих типов условий: <ul style="list-style-type: none"> <li>Value of Item / Column in Expression 1 Is NOT NULL. Кнопка будет отображена, если значение элемента не NULL.</li> <li>Value of Item / Column in Expression 1 != Zero. Кнопка будет отображена, если значение элемента не равно 0.</li> <li>Value of Item / Column in Expression 1 Is NULL. Кнопка будет отображена, если значение элемента NULL.</li> <li>Value of Item / Column in Expression 1 Is NULL or Zero. Кнопка будет отображена, если значение элемента NULL, или равно нулю.</li> <li>Value of Item / Column in Expression 1 = Zero. Кнопка будет отображена, если значение элемента равно 0.</li> <li>Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero. Кнопка будет отображена, если значение элемента не NULL и не равно нулю.</li> </ul>

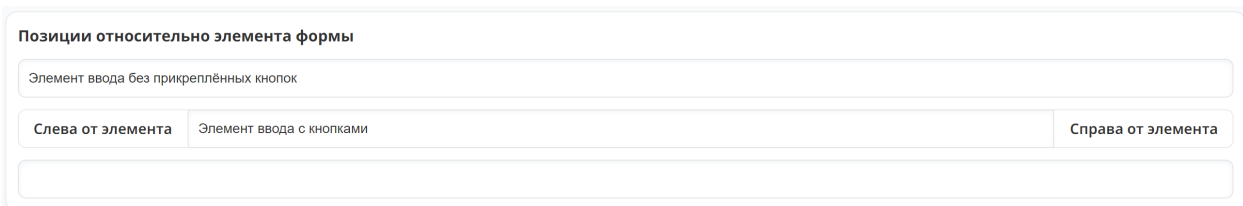
## Отображение кнопок

В зависимости от типа местоположения кнопки могут отображаться по-разному. Для наглядности возможные варианты – изображены на скриншотах.

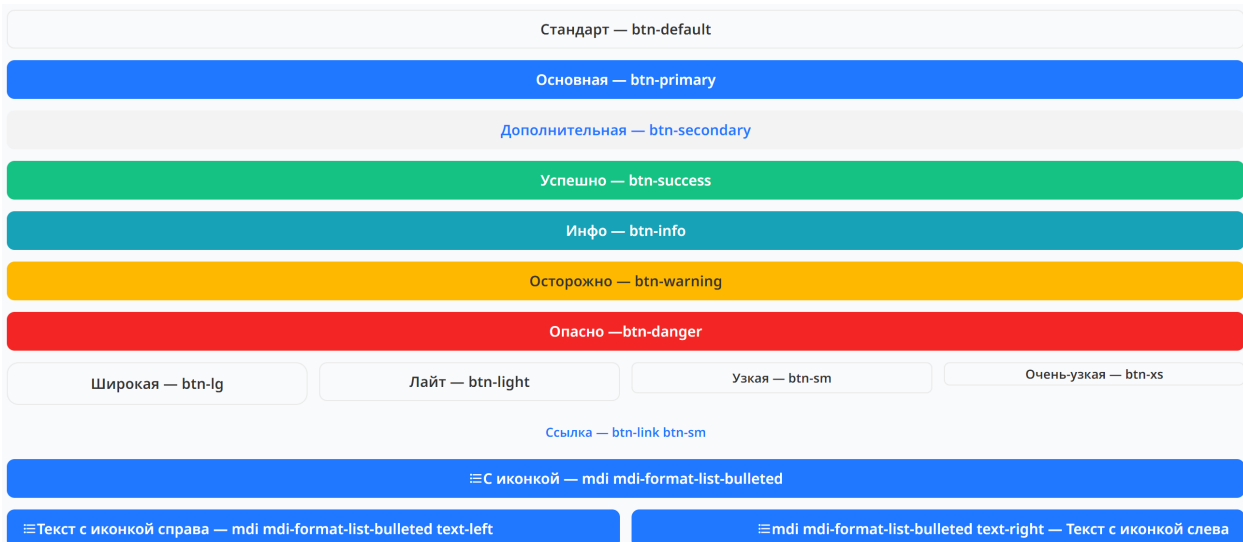
Для местоположения REGION:



Для местоположения ITEM:

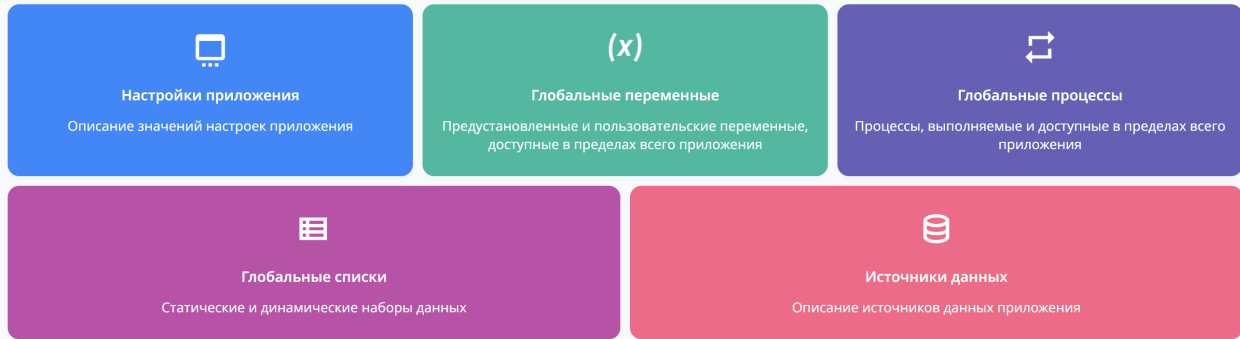


Для местоположения FORM с демонстрацией различных классов кнопок:



## 7.6.4 Карточки (CARDS)

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



Источником данных для компонента CARDS является – список.

### Создание элемента списка

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

В списке создайте новый элемент и настройте параметры:

- Имя – наименование элемента и отображаемый заголовок;
- Последовательность - численный порядок элемента, определяющий его позицию;
- CSS класс иконки - иконка для карточки;
- Ссылка - ссылка на страницу или скрипт при клике на карточку;
- Атрибут 1 – вспомогательная надпись;
- Атрибут 2 - RGB-цвет карточки в формате #ffffff.

Пример SQL-запроса для создания динамического списка:

```
SELECT
  'Регионы' title,
  'dripicons dripicons-browser' css_icon,
  '/content/5020' target,
  'Демонстрация типов регионов на примерах использования' attribute_01,
  '#FFC62D' attribute_02
```

**Примечание:** При создании динамического списка для использования его в качестве источника данных региона Карточки требуется соблюдение строгого именования столбцов как в вышеуказанном примере!

### Настройка компонента

Для компонента CARDS доступны настройки двух уровней: параметры региона и атрибуты.

#### Параметры региона

Для данного компонента все параметры – стандартные для региона, кроме того, что источником данных для компонента является список, который нужно указать в поле Источник данных – Список.

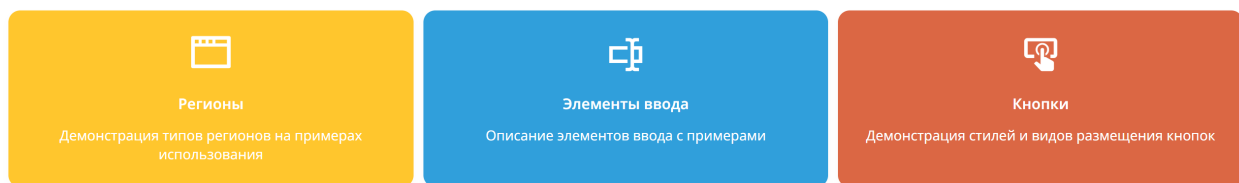
#### Атрибуты региона

Компонент CARDS имеет два специальных атрибута, доступных в настройках компонента:

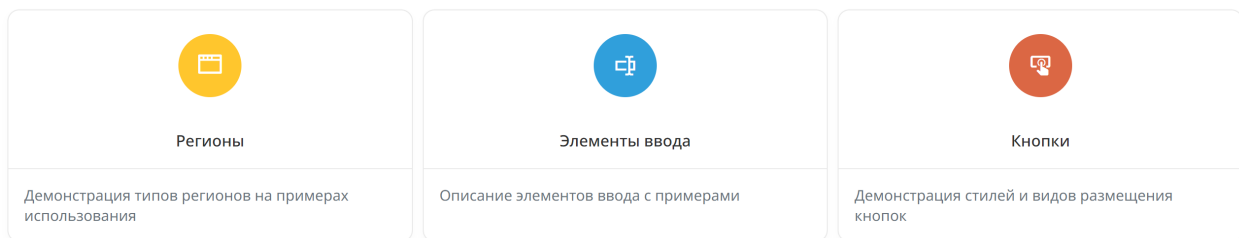
Атрибут	Тип	Описание
Тема	Список	Шаблон, определяющий отображение карточек. На выбор три варианта: <ul style="list-style-type: none"> <li>• Блочная.</li> <li>• Функциональная.</li> <li>• Базовая.</li> </ul>
Колонки	Список	Количество карточек в ряд. В случае если элементов в списке больше, чем указанное здесь число, карточки расположатся в несколько рядов. Принимает значение от одного до двенадцати.

Для наглядности виды шаблонов изображены на скриншотах:

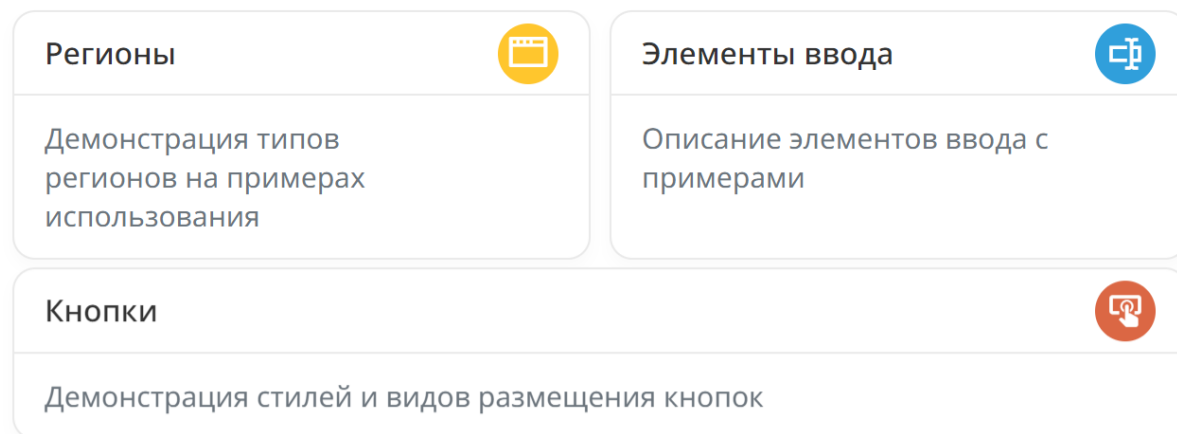
### Блочная



### Функциональная



### Базовая



## 7.6.5 График (CHART)

CHART – служит для графического представления числовых данных в виде диаграмм. Компонент поддерживает следующие типы диаграмм:

- Столбчатая – вертикальная

- Столбчатая - горизонтальная
- Кольцевая диаграмма
- Линейная диаграмма
- Диаграмма с областями

Источником данных для компонента служат результаты SQL – запроса.

Пример SQL-запроса:

```
SELECT
  'Рабочие дни' day_kind,      -- Подпись
  247::numeric days          -- Значение
UNION ALL
SELECT
  'Выходные дни',
  118::numeric
```



- Выходные дни
- Рабочие дни

### Настройка компонента

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

### Параметры региона

Отличительной особенностью региона типа “График” является необходимость обязательного заполнения поля Источник данных - SQL.

### Атрибуты региона

Группа специальных атрибутов, характерных для данного вида регионов.

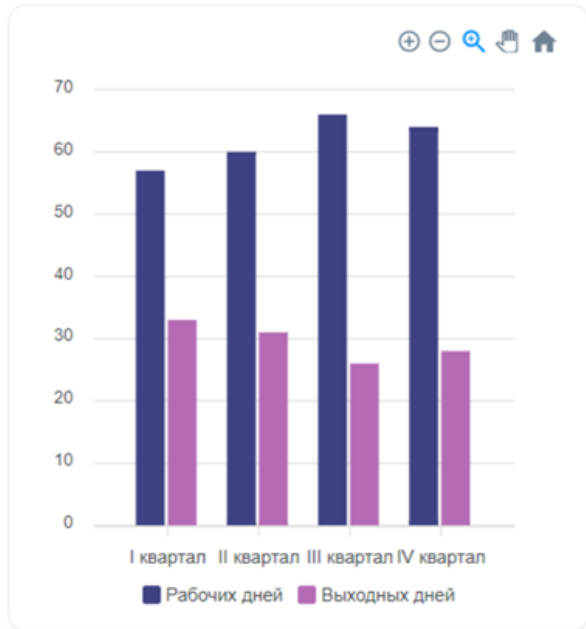
Атрибут	Тип	Описание
Диаграмма - Тип	Список	Определяет внешний вид диаграммы. Доступны варианты: <ul style="list-style-type: none"><li>• Столбчатая – вертикальная</li><li>• Столбчатая - горизонтальная</li><li>• Кольцевая диаграмма</li><li>• Линейная диаграмма</li><li>• Диаграмма с областями</li></ul>
Колонка категории	Список	Определяет какой из столбцов запроса к БД содержит значение для категории диаграммы. Значения серий должны быть сгруппированы по категориям.
Динамическая загрузка	Переключатель	Если установлен, то диаграмма не будет строиться без явного вызова из динамического события. Иначе загружается при открытии страницы.
Ширина	Число	Определяет ширину для компонента.
Высота	Число	Определяет высоту для компонента.

## Параметры серий

Параметр	Тип	Описание
Имя	Текст	Определяет отображаемое имя серии. Для дерева объектов и для отображения на диаграмме в приложении. Поскольку в графиках типа кольцевая диаграмма серия одна, для этого типа имя серии отобразится только в дереве объектов.
Последовательность	Число	Определяет порядок серий при выводе в интерфейс. Поскольку в графиках типа кольцевая диаграмма серия одна, отобразится в приложении только первая из них.
Колонка значений	Список	Из выпадающего списка необходимо выбрать один из столбцов запроса, содержащий значение типа numeric, для вывода значения в серию.
Условия отображения - Тип	Список	Определяет тип условия отображения серии. По умолчанию - Always. В зависимости от типа потребуется указать дополнительные параметры условия. Для Always и Never дополнительных условий не потребуется.
Условия отображения - Первое условие	Область текста	Запрос в формате SQL. Параметр применим для следующих типов условий: <ul style="list-style-type: none"> <li>Exists (SQL query returns at least one row). Если запрос вернул хотя бы одну строку, серия будет отображена на графике.</li> <li>NOT Exists (SQL query returns no rows). Если запрос не вернул ни одной строки, серия будет отображена на графике.</li> </ul>
Условия отображения - Выражение SQL	Область текста	Логическое выражение на языке SQL. Если выражение вернёт значение истина, серия будет отображена на графике. Параметр применим для типа условия SQL Expression.
Условия отображения - Первый вход	Текст / Окно конструктора	Задаёт список входных параметров для запроса SQL в поле “Первое условие”, либо “Выражение SQL”. Для каждой переменной подстановки в запросе должен быть определён входящий параметр. Может принимать значения глобальных переменных, элементов ввода и выбора страницы. Можно ввести как текстом, так и выбрать в конструкторе. Применим для типов условий: <ul style="list-style-type: none"> <li>Exists (SQL query returns at least one row).</li> <li>NOT Exists (SQL query returns no rows).</li> <li>SQL Expression</li> </ul>
Условия отображения - Элемент	Текст / Окно конструктора	Позволяет выбрать элемент, в зависимости от значения которого столбец будет отображён или нет. Применим для следующих типов условий: <ul style="list-style-type: none"> <li>Value of Item / Column in Expression 1 Is NOT NULL. Серия будет отображена, если значение элемента не NULL.</li> <li>Value of Item / Column in Expression 1 != Zero. Серия будет отображена, если значение элемента не равно 0.</li> <li>Value of Item / Column in Expression 1 Is NULL. Серия будет отображена, если значение элемента NULL.</li> <li>Value of Item / Column in Expression 1 Is NULL or Zero. Серия будет отображена, если значение элемента NULL, или равно нулю.</li> <li>Value of Item / Column in Expression 1 = Zero. Серия будет отображена, если значение элемента равно 0.</li> <li>Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero. Серия будет отображена, если значение элемента не NULL и не равно нулю.</li> </ul>

Для наглядности виды диаграмм изображены на скриншотах:

### Столбчатая



**Кольцевая**



- Выходные дни
- Рабочие дни

**Линейная**



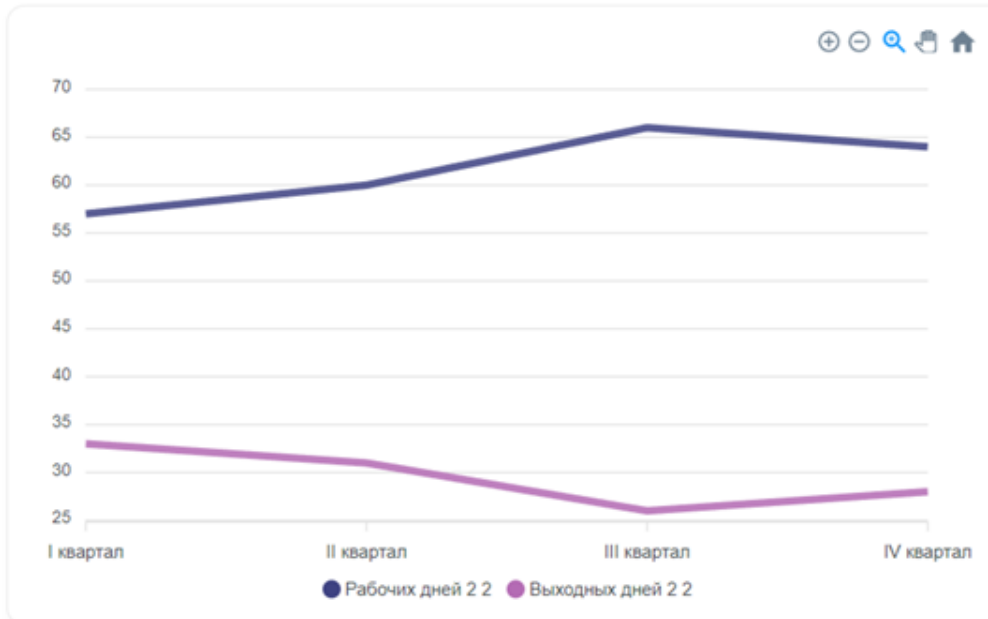
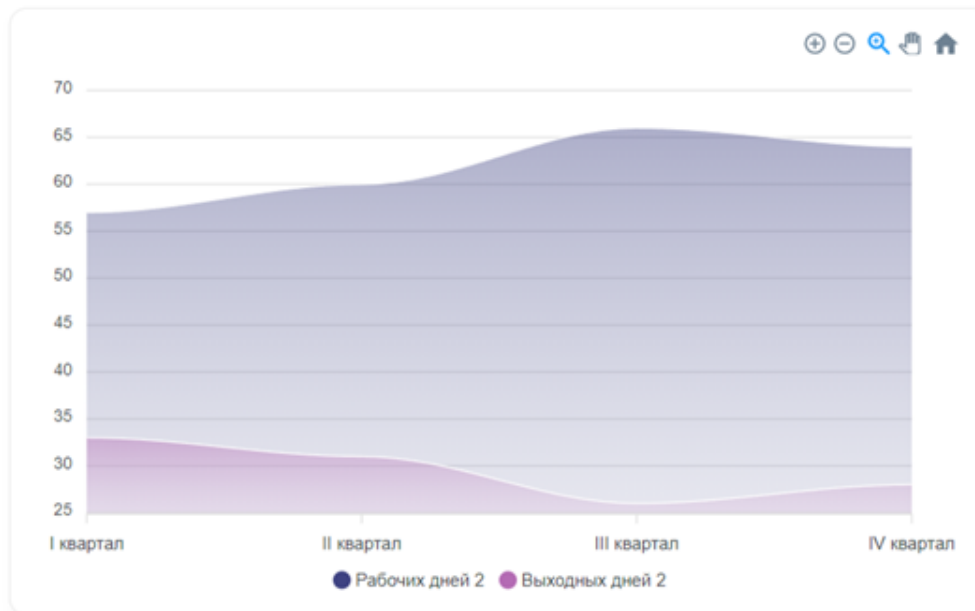


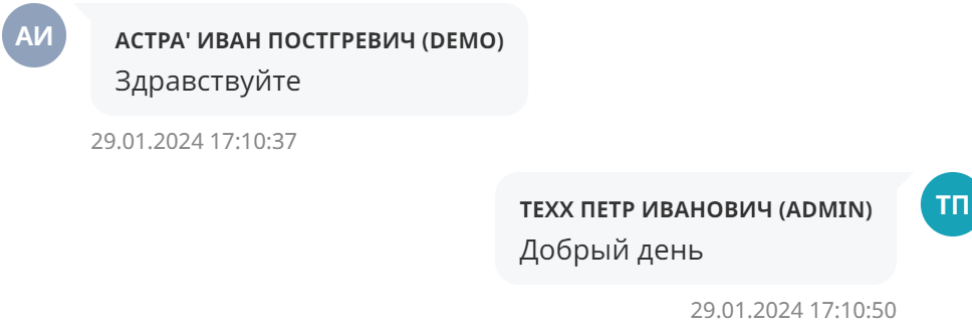
Диаграмма с областями



Графику можно установить пользовательские фильтры данных по элементам формы. Для этого необходимо прописать правильное условие в запросе выборки данных чарта и передать необходимые параметры, которые будут ограничивать выборку.

### 7.6.6 Чат (CHAT)

CHAT – регион для вывода чатов. В качестве источника данных для компонента является результат SQL-запроса. Таким образом компонент отображает полученный в результате SQL-запроса набор данных в виде пользовательского чата.



В запросе помимо текста сообщения, даты и пользователя, также указывается положение текста и стиль оформления иконки.

Пример SQL- запроса:

```
SELECT
  cm.message_text comment_text, -- Текст сообщения
  TRIM(CONCAT(ul.last_name, ' ',
              ul.first_name, ' ',
              ul.second_name)) || ' (' || ul.username || ')' user_name, -- Имя пользователя
CASE
  WHEN ul.username = UPPER($1) THEN 'right'
  ELSE 'left'
END align, -- Выравнивание сообщения (left, right)
TO_CHAR(cm.message_date, 'DD.MM.YYYY HH24:MI:SS') comment_date, -- Дата отправки
SUBSTR(ul.last_name,1,1) || SUBSTR(ul.first_name,1,1) user_icon, -- Текст в иконке
CASE
  WHEN ul.username = UPPER($1) THEN 'bg-info text-white'
  ELSE NULL
END icon_modifier -- CSS классы иконки
FROM
  chat.t_chat_messages cm
JOIN
  users.t_user_list ul ON cm.id_user = ul.id
WHERE
  cm.id_chat = 1
ORDER BY
  cm.message_date
```

## Настройка компонента

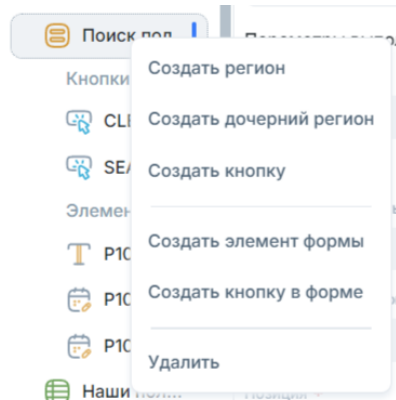
Помимо необходимости выбора и настройки источника данных, остальные параметры компонента стандартные для региона.

## 7.6.7 Форма (FORM)

FORM – компонент для создания разнообразных элементов ввода пользовательской информации.

### Элементы формы

После создания на странице региона с типом FORM можно добавлять элементы формы. Добавление элементов осуществляется через контекстное меню по ПКМ.



### Основные атрибуты элемента

Рассмотрим основные атрибуты, которые присутствуют вне зависимости от типа элемента ввода.

Атрибут	Описание
Имя	Для идентификации элемента необходимо указать его имя. Имя должно быть уникальным в разрезе заданной страницы. Для корректной работы приложения настоятельно рекомендуется указывать префикс в формате: PXXX. Где XXX - номер текущей страницы. При создании нового элемента ввода на форме, имя будет заполнено автоматически, с правильно сформированным префиксом. Имя также будет выступать в роле идентификатора элемента в HTML.
Тип	Необходимо указать тип элемента. По умолчанию проставляется в элемент типа “Текст”. См. далее для описания типов элементов.
Источник данных	Источник данных для данного компонента.
Последовательность	Поле определяет последовательность отображения элемента ввода в данном регионе. Необходимо указать целочисленное значение. При создании нового элемента поле заполняется автоматически - увеличивая значение на 10 от предыдущего элемента.
Тип	Определяет регион, которому принадлежит элемент ввода, можно выбрать регион только с типом FORM.
Начать новую строку	Указывает необходимость перенести элемент на новую строку внутри сетки региона.
Количество столбцов	Указывает количество занимаемых столбцов внутри сетки региона.
Наименование	Подпись элемента на странице. Отображается только, если отображается элемент.
Не пустой	Определяет необходимость наличия значения внутри элемента при обработке результата ввода. Если значение не было указано пользователю будет отправлено сообщение об ошибке, с требованием заполнить значение для заданного элемента.
Максимальная длина	Определяет максимальное возможное значение длины элемента в символах. Действует для типов элементов, где ожидается ввод с клавиатуры.
Подсказка	Значение для подсказки, которая показывает ожидаемые значения.

продолжается на следующей странице

Таблица 4 – продолжение с предыдущей страницы

Атрибут	Описание
Маска ввода	<p>Определяет формат ввода информации в поле. Для форматирования пользовательского ввода используйте следующие идентификаторы:</p> <ul style="list-style-type: none"> <li>• 9 — цифровой символ</li> <li>• a — символ буквы</li> <li>• * — буквенно-цифровой символ</li> </ul>
Настройки JS	Определяет дополнительные настройки элемента ввода. См. подробности далее.
Тип настроек по умолчанию	<p>Выберите значение по умолчанию. Тип определяет откуда брать значение. Доступные значения:</p> <ul style="list-style-type: none"> <li>• Статический - введите значение по умолчанию, которое будет отображаться, если значение сессии для данного элемента не задано</li> <li>• SQL - введите sql выражение, которое вернет значение для элемента</li> <li>• Элемент - значение будет взять из значения элемента приложения, записанного в сессию.</li> </ul>
Условия отображения	Определяет условия вывода элемента на форму.
Только чтение	Определяет условия вывода элемента, как доступного только для чтения.

### Типы элементов

Для компонента FORM доступны следующие типы элементов:

Атрибут	Описание
Текст	Элемент для текстового ввода пользовательской информации. Дополнительные настройки (Настройки JS) включают в себе возможность отправки формы на сервер по нажатию на Enter.
Текстовая область	Элемент для текстового ввода многострочного значения. Дополнительных настроек не имеет.
Список	Элемент выбора из списка элементов.
Числовой	<p>Элемент для числового ввода. Дополнительные настройки включают в себя настройку следующих параметров:</p> <ul style="list-style-type: none"> <li>• Знак разделителя десятичной части.</li> <li>• Знак разделителя разрядов числа.</li> <li>• Минимальное и максимальное значение числа</li> <li>• Количество знаков после запятой</li> <li>• Отправку формы по нажатию на Enter</li> </ul> <p>При обработке значения, введенного пользователем в данное поле необходимо приводить указанное значение к числовому, используя функцию <code>to_number</code>.</p>
Скрытый	Элемент, который не будет отображаться на форме. Используется для передачи параметров на форму.

продолжается на следующей странице

Таблица 5 – продолжение с предыдущей страницы

Атрибут	Описание
Дата	<p>Элемент для выбора значения типа дата и время. Дополнительные настройки включают в себя:</p> <ul style="list-style-type: none"> <li>• Возможность выбора даты</li> <li>• Возможность выбора времени</li> <li>• Минимальная и максимальная дата для выбора</li> <li>• Отправку формы по нажатию на Enter</li> </ul> <p>При обработке значения, введенного пользователем в данное поле необходимо приводить указанное значение к типу даты, используя функцию <code>to_date/ to_timestamp</code>.</p>
Радиокнопки	<p>Элемент для отображения радиокнопок. Для задания списка значений используйте выражения типа SQL, которое вернет два столбца. Первый столбец будет содержать отображаемое на форме значение радиокнопки, второй столбец будет содержать значение, передаваемое обработчику форму.</p>
Флажки	<p>Элемент для выбора значений по флажкам. Также как для радиокнопок необходимо задать список значений используя SQL выражение. Дополнительные настройки:</p> <ul style="list-style-type: none"> <li>• Разделитель - определяет формат разделителя значений, которые будут переданы в базу в виде строки.</li> <li>• Колонки - количество колонок, на которое будут разделены значения флажков.</li> </ul>
Переключатель	<p>Элемент для отображения переключателя значения. Работает также как радиокнопки, отличием является визуальное оформление. Также как для радиокнопок необходимо задать список значений используя SQL выражение. Дополнительные настройки отсутствуют.</p>
Автозаполнение	<p>Элемент типа список с возможностью автозаполнения. При вводе значения будет производиться поиск по доступным значениям списка, отфильтровывая результат возможных значений. Для заполнения элемента конечному пользователю необходимо выбрать значение из отфильтрованного списка. Также как для радиокнопок необходимо задать список значений используя SQL выражение. Дополнительные настройки отсутствуют.</p>
Пароль	<p>Элемент для ввода пароля. Введенные символы автоматически заменяются на знак * , по требованию пользователя информация может быть отображена по нажатию на кнопку рядом с элементом.</p>
WYSIWYG	<p>Элемент отображается в виде редактора WYSIWYG</p>
Ввод файла	<p>Элемент для загрузки пользовательских файлов. Настройки элемента задают:</p> <ul style="list-style-type: none"> <li>• Высота блока - определяет высоту области drag and drop.</li> <li>• Mime типы - задаёт ограничение по загружаемым типам файлов. Типы файлов должны быть указаны как mime-типе через запятую. При указании значения "any" - снимает ограничение по типам.</li> <li>• Текст подсказки - задаёт текст подсказки для пользователя</li> <li>• Макс. размер файла - определяет максимально допустимый размер файла</li> <li>• Макс. кол-во файлов - определяет максимальное количество файлов.</li> </ul> <p>Для обработки загруженных файлов XRAD создаёт временную таблицу в схеме <code>pg_temp</code>, используя следующее выражение: <code>CREATE TEMP TABLE IF NOT EXISTS pg_temp.xrad_files (name text, file_name text, file_mime text, file_content bytea)</code>. Колонка <code>name</code> будет содержать значение из элемента ввода файла, переданное на сервер. При загрузке нескольких файлов значение элемента будет передано в виде строки, в которой значения будут разделены знаком двоеточия (:).</p>

продолжается на следующей странице

Таблица 5 – продолжение с предыдущей страницы

Атрибут	Описание
Множественный выбор	Список элементов с возможностью выбора нескольких значений.

## 7.6.8 HTML-код (HTML)

HTML – регион для отображения HTML-кода. Данный тип региона используется для вывода текста, картинок, видео и любых других элементов HTML-кода.

### Текст и картинка

Текст



Компонент поддерживает два типа источника данных:

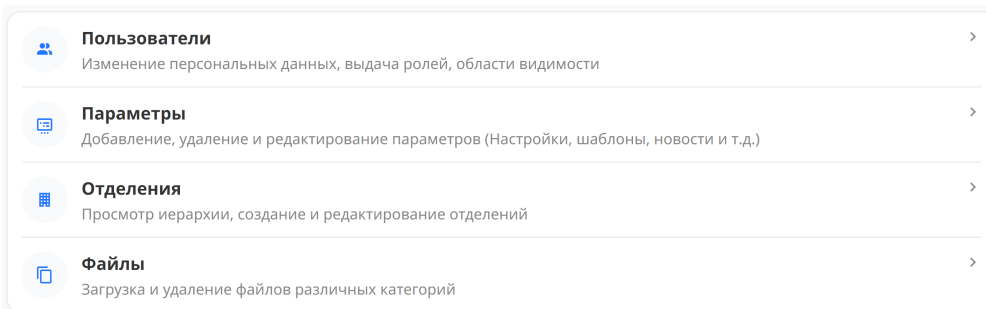
- Статический – HTML-код указывается напрямую в параметре HTML код.
- SQL – HTML-код будет получен в результате выполнения SQL-запроса.

### Настройка компонента

Помимо необходимости выбора источника, остальные параметры компонента стандартные для региона.

## 7.6.9 Навигационная панель (PAGE NAVIGATION)

PAGE NAVIGATION – компонент для размещения навигационной панели на странице. Источником данных служит список, элементы которого отобразятся с иконкой, заголовком-ссылкой и описанием.



Пример SQL-кода для создания динамического списка:

```
SELECT
  n.title name,
  'uil uil-envelope' icon,
  n.message_short attribute_01,
  '/content/9201?P9201_ID='||n.id target,
  ns.status_name attribute_02
FROM orgn.t_notifications n
ORDER BY n.date_created DESC;
```

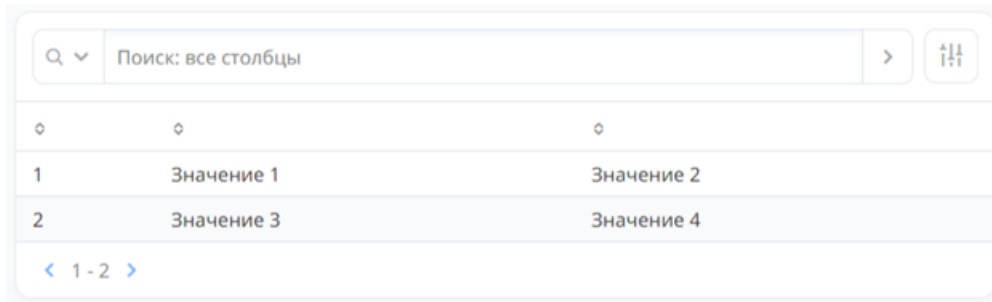
**Примечание:** При создании динамического списка для использования его в качестве источника данных региона Панель навигации требуется соблюдение строгого именования столбцов как в вышеуказанном примере!

### Настройка компонента

Помимо необходимости указания списка-источника, остальные параметры компонента стандартные для региона.

#### 7.6.10 Отчет (REPORT)

REPORT – компонент для вывода данных в виде таблицы. Данные для вывода формируются в результате SQL-запроса.



The screenshot shows a user interface for a REPORT component. At the top, there is a search bar with a magnifying glass icon, a dropdown arrow, and the text "Поиск: все столбцы". To the right of the search bar are two icons: a right-pointing arrow and a list icon. Below the search bar is a table with three columns. The first column contains the numbers 1 and 2. The second column contains "Значение 1" and "Значение 3". The third column contains "Значение 2" and "Значение 4". At the bottom of the table, there is a pagination control showing "< 1 - 2 >".

1	Значение 1	Значение 2
2	Значение 3	Значение 4

Параметры компонента REPORT позволяют:

- установить количество отображаемых строк
- добавить пагинации
- отобразить/скрыть заголовки столбцов
- добавить поле поиска по набору данных
- задать сообщение, выводимое при отсутствии строк в выборке.

Настройки столбцов отчёта позволяют:

- изменить тип с текста на ссылку, установить для ссылки иконку
- добавить возможность сортировки
- включить столбец в список доступных для поиска
- задать маску вывода
- установить ширину, выравнивание и другие настройки отображения.

Отчету можно установить пользовательские фильтры данных по элементам формы. Для этого необходимо прописать правильное условие в запросе выборки данных отчета и передать необходимые параметры, которые будут ограничивать выборку.

### Настройка компонента

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

#### Параметры региона

Отличительной особенностью компонента типа “Отчёт” является необходимость обязательного заполнения поля Источник данных - SQL. Необходимо указать SQL-запрос, на основе которого формируется список колонок отчета и выбираются данные для вывода на форму.

## Атрибуты региона

Группа специальных атрибутов, характерных для данного вида регионов.

Атрибут	Тип	Описание
Включить поиск	Переключатель	Добавляет элемент поиска по отчёту. Текстовое поле ввода и выпадающий список с возможностью выбрать столбцы репорта для поиска по ним.
Включить разбиение на страницы	Переключатель	Добавляет пагинации - возможность просматривать страницы далее первой.
Включение заголовков	Переключатель	Добавляет заголовки столбцам отчёта.
Шаблон	Список	Задаёт базовый шаблон отчёта. На текущий момент существует два варианта шаблонов: <ul style="list-style-type: none"> <li>• Стандарт</li> <li>• Столбец - значение</li> </ul>
Ограничение строк	Число	Устанавливает количество строк отчёта на одну страницу пагинации.
Сообщение об отсутствии данных	Область текста	Позволяет указать текст, который будет выведен на экран в случае, если запрос к БД не вернул данных.
Шаблоны отчета	Окно настроек	Позволяет выбрать шаблон отчета, как шаблон по умолчанию. По нажатию открывается окно со списком шаблонов отчета. Шаблон формируется каждый раз, когда пользователь задаёт новый фильтр отбора, сортировку или подсветку. Разработчик может выбрать один из таких шаблонов как шаблон по умолчанию, после чего данный вид отчета будет открываться всем пользователям.

## 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.

Параметр	Тип	Описание
Имя столбца	Текст (только чтение)	Наименование столбца. Определяется алиасом столбца запроса источника данных. Поле не редактируется в настройках параметров - оно всегда соответствует наименованию столбца в запросе. Ссылка на значение столбца происходит по этому полю.

продолжается на следующей странице



Таблица 6 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Тип	Список	<p>Определяет набор параметров столбца и его поведение при выводе на страницу. Может принимать следующие значения:</p> <ul style="list-style-type: none"> <li>• Текст. Простой вывод значения информации из БД.</li> <li>• Ссылка. Позволяет нажатием на элемент внутри ячейки столбца перенаправить пользователя на другую страницу, либо вызвать динамический скрипт.</li> <li>• Скрытый. Столбец репорта будет скрыт в пользовательском интерфейсе. Его значение (например, для передачи в параметры ссылки) будет всё ещё доступно</li> <li>• Флажок. Столбец представляет собой группу чекбоксов для множественного выбора значений отчёта.</li> </ul>
Связанный элемент	Текст / Окно конструктора	Позволяет выбрать элемент ввода, либо ввести глобальную переменную, куда будут переданы значения из ячеек отмеченных чекбоксом строк. Параметр применим для типа столбца - Флажок.
Заголовок	Текст	Заголовок столбца для вывода в пользовательском интерфейсе на странице приложения.
Последовательность	Число	Определяет порядок столбцов при выводе в интерфейс. По умолчанию равен порядку столбцов в запросе при первой генерации. При добавлении столбца в запрос в дальнейшем, новому будет присвоен последний номер, вне зависимости от позиции в запросе.
Выравнивание	Переключатель	Выравнивание текста-содержимого столбца. По левому краю, по центру, или по правому краю.
Ссылка	Текст / Окно конструктора	Позволяет задать ссылку на другую страницу, вызвать динамический скрипт, либо настроить в конструкторе ссылку на страницу внутри приложения. В последнем случае можно выбирать значения не только полей ввода, но и столбцов отчёта. Вместо текста ссылки можно также передать значение строки репорта - такой подход позволяет настраивать ссылку внутри sql-запроса. Параметр применим для Тип столбца - Ссылка.
Значок ссылки	Текст	Позволяет выбрать иконку для вывода в ячейке столбца. Вместо текста ссылки можно также передать значение строки репорта - такой подход позволяет выбрать картинку внутри sql-запроса. Параметр применим для Тип столбца - Ссылка.
Ширина	Текст	Фиксирует ширину столбца.
Формат маски	Текст	Форматирует вывод наложением маски по правилам форматирования SQL-функции to_char.
Включить сортировку	Переключатель	Разрешает сортировать строки по значению столбца.
Поиск по столбцам	Переключатель	Разрешает осуществлять поиск по значению столбца, если поиск включен в атрибутах отчёта.
Выражение HTML	Область текста	Позволяет определить формат вывода в виде HTML. Можно сослаться на значение столбца репорта.
Условия отображения - Тип	Список	Определяет тип условия отображения столбца отчёта. По умолчанию - Always. В зависимости от типа потребуется указать дополнительные параметры условия. Для Always и Never дополнительных условий не потребуется.

продолжается на следующей странице

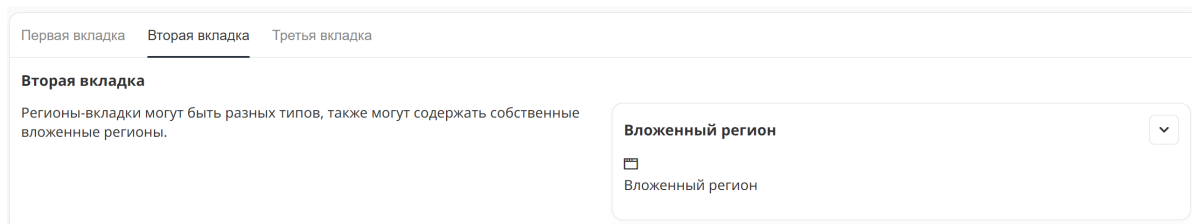
Таблица 6 – продолжение с предыдущей страницы

Параметр	Тип	Описание
Условия отображения - Первое условие	Область текста	Запрос в формате SQL. Параметр применим для следующих типов условий: <ul style="list-style-type: none"> <li>Exists (SQL query returns at least one row). Если запрос вернул хотя бы одну строку, столбец будет отображен в отчёте.</li> <li>NOT Exists (SQL query returns no rows). Если запрос не вернул ни одной строки, столбец будет отображен в отчёте.</li> </ul>
Условия отображения - Выражение SQL	Область текста	Логическое выражение на языке SQL. Если выражение вернёт значение истина, столбец будет отображен в отчёте. Параметр применим для типа условия SQL Expression.
Условия отображения - Первый вход	Текст / Окно конструктора	Задаёт список входных параметров для запроса SQL в поле “Первое условие”, либо “Выражение SQL”. Для каждой переменной подстановки в запросе должен быть определён входящий параметр. Может принимать значения глобальных переменных, элементов ввода и выбора страницы. Можно ввести как текстом, так и выбрать в конструкторе. Применим для типов условий: <ul style="list-style-type: none"> <li>Exists (SQL query returns at least one row).</li> <li>NOT Exists (SQL query returns no rows).</li> <li>SQL Expression.</li> </ul>
Условия отображения - Элемент	Текст / Окно конструктора	Позволяет выбрать элемент, в зависимости от значения которого столбец будет отображён или нет. Применим для следующих типов условий: <ul style="list-style-type: none"> <li>Value of Item / Column in Expression 1 Is NOT NULL. Столбец будет отображён, если значение элемента не NULL.</li> <li>Value of Item / Column in Expression 1 != Zero. Столбец будет отображён, если значение элемента не равно 0.</li> <li>Value of Item / Column in Expression 1 Is NULL. Столбец будет отображён, если значение элемента NULL.</li> <li>Value of Item / Column in Expression 1 Is NULL or Zero. Столбец будет отображён, если значение элемента NULL, или равно нулю.</li> <li>Value of Item / Column in Expression 1 = Zero. Столбец будет отображён, если значение элемента равно 0.</li> <li>Value of Item / Column in Expression 1 Is NOT null and the Item / Column Is NOT Zero. Столбец будет отображён, если значение элемента не NULL и не равно нулю.</li> </ul>

### 7.6.11 Вкладки (TABS)

TABS – контейнер для группировки дочерних регионов на переключаемых вкладках. После создания на странице компонента с типом TABS необходимо добавить дочерние регионы с помощью контекстного меню. Каждый дочерний регион отображается в виде вкладки.

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

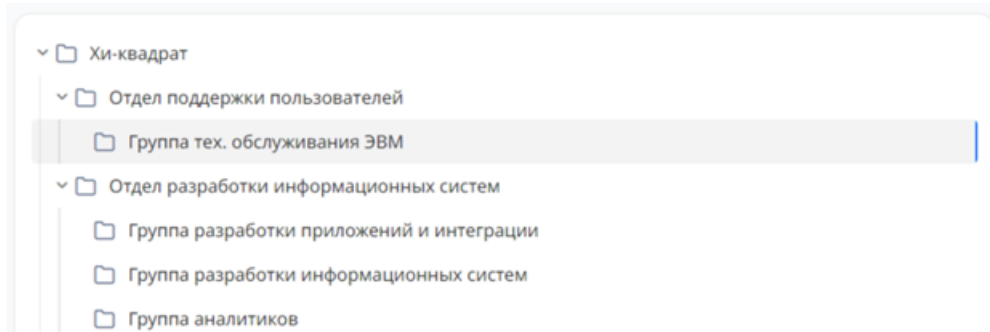


## Настройки компонента

Параметры компонента стандартные для региона.

### 7.6.12 Дерево (TREE)

TREE – регион для вывода древовидной структуры данных. В качестве источника данных используется SQL-запрос.



Пример SQL-запроса для компонента TREE:

```
SELECT
  t.id,      -- ID элемента
  t.id_fk,   -- ID родительского элемента
  t.title,   -- Подпись элемента
  t.link     -- Ссылка, по которой будет осуществлен переход после нажатия на элемент
FROM
  dep.f_get_tree_with_scope(1) t;
```

## Настройки компонента

Доступны настройки двух уровней: параметры региона и атрибуты дерева.

### Параметры региона

Параметры компонента стандартные для региона.

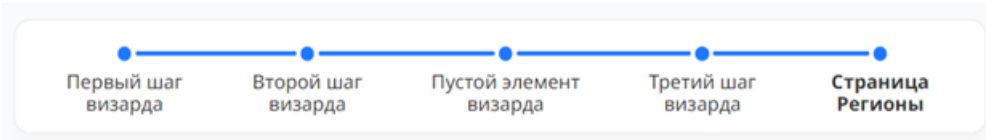
### Атрибуты региона

Атрибут	Тип	Описание
Столбец РК	Список	Идентификатор узла. Из выпадающего списка необходимо выбрать один из столбцов запроса.
Столбец FK	Список	Идентификатор родительского узла. Из выпадающего списка необходимо выбрать один из столбцов запроса.
Столбец заголовка	Список	Отображаемый заголовок узла. Из выпадающего списка необходимо выбрать один из столбцов запроса.
Ссылка	Текст / Окно конструктора	Ссылка куда будет произведено перенаправление после нажатия на узел дерева.
Столбец с ссылкой	Список	Определяет столбец, содержащий ссылку для перехода.

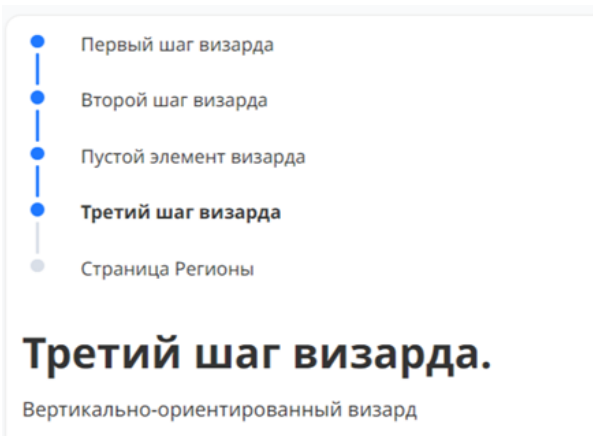
### 7.6.13 Визард (WIZARD)

WIZARD – компонент для графического представления последовательности действий на основе списка. Компонент поддерживает два вида направленности:

#### Горизонтальная



#### Вертикальная



#### Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты визарда.

#### Параметры региона

Параметры компонента стандартные для региона.

#### Атрибуты региона

Атрибут	Тип	Описание
Разрешить переход по нажатию	Переключатель	Определяет, будет ли осуществляться переход на связанную страницу при клике на узле визарда.
Отображать шаги	Список	Доступны несколько вариантов отображения заголовков шагов визарда: <ul style="list-style-type: none"> <li>• Все. Заголовки каждого элемента списка будут отображаться на визарде</li> <li>• Текущий. На визард будет выведен только заголовок текущего шага, остальные будут безымянными узлами.</li> <li>• Никогда. На визард не будут выводиться заголовки узлов.</li> </ul>
Направление	Список	Пространственная направленность визарда. На выбор два варианта: <ul style="list-style-type: none"> <li>• Горизонтально</li> <li>• Вертикально</li> </ul>

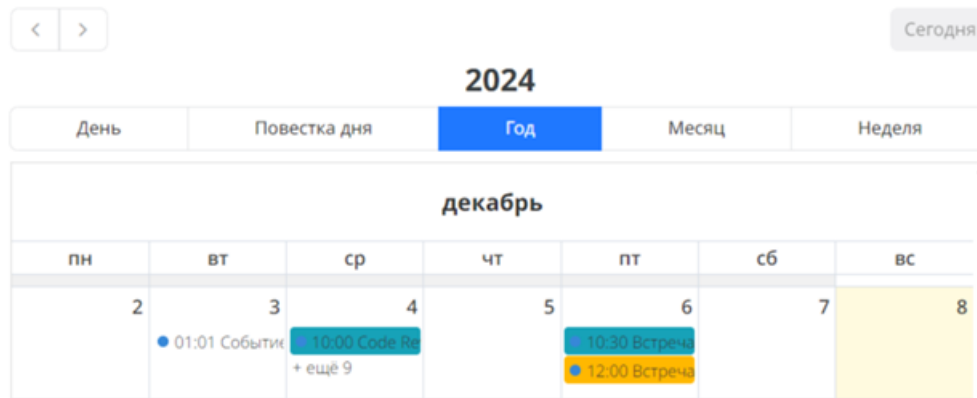
## 7.6.14 Календарь (CALENDAR)

CALENDAR – компонент для отображения данных с меткой времени начала и конца события в виде интерактивного календаря. Источником данных для компонента является результат SQL-запроса.

Пример SQL-запроса

```
SELECT
  c.id,           -- ID строки для добавления в ссылку
  c.title,       -- Заголовок события
  c.start_ev,    -- Начало периода события
  c.end_ev,      -- Окончание периода события
  c.color_class  -- HEX-код цвета
FROM
  events.t_calendar c;
```

Сам календарь можно настроить, изменив отображение выходных, выбрав необходимые для показа представления (День, Неделя, Месяц, Год, Повестка дня), а также высоту календаря и отображение подсказки о событии при наведении курсора. Событию календаря можно установить цвет и ссылку, по которой будет произведен переход после нажатия (в ссылку можно добавить id записи).



### Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты календаря.

### Параметры региона

Параметры компонента стандартные для региона.

## Атрибуты региона

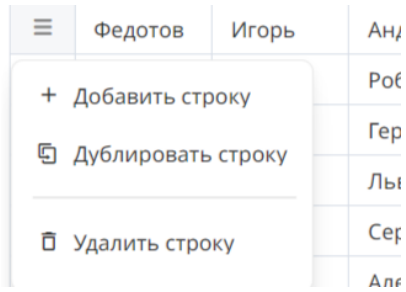
Атрибут	Тип	Описание
Колонка с заголовком	Список	Один из столбцов источника данных SQL можно назначить в качестве заголовка отображаемого на календаре события.
Колонка с началом периода	Список	Один из столбцов источника данных SQL можно назначить в качестве времени начала отображаемого на календаре события.
Колонка с окончанием периода	Список	Один из столбцов источника данных SQL можно назначить в качестве времени окончания отображаемого на календаре события.
CSS колонка	Список	Один из столбцов источника данных SQL можно назначить для определения стиля отображаемого на календаре события.
Высота	Число	Размер компонента по вертикали.
Представления	Множественный выбор	Перечень доступных уровней представления календаря. Выбранные варианты будут предложены пользователю в интерфейсе компонента. Список представлений: <ul style="list-style-type: none"> <li>• Год</li> <li>• Месяц</li> <li>• Неделя</li> <li>• День</li> <li>• Повестка дня</li> </ul>
Отображать время	Переключатель	Если установлен, отображаться будет не только дата, но и время события.
Показывать подсказку	Переключатель	Если установлен, наименование события будет отображаться на всплывающей подсказке, при наведении на ячейку в календаре.
Отображать выходные	Переключатель	Если установлен, будут отображаться все дни недели.
Ссылка на просмотр	Текст / Окно конструктора	Определяет ссылку, на которую будет перенаправлен пользователь при клике на событие в календаре.

## 7.6.15 Таблица (DATAGRID)

DATAGRID – таблица с данными аналогичная REPORT, но с возможностью редактирования данных. Записи в DATAGRID можно добавлять, изменять и удалять.

☰	Фамилия	Имя	Отчество	Телефон	Дата ро...	Электр...	Тип кли...	<input type="checkbox"/>
☰	Горелов	Виктор	Лукич	899912...	07.08.1...	info@xs...	Физ. ли...	<input checked="" type="checkbox"/>
☰	Романов	Кирилл	Фёдоро...	899912...	23.07.1...	info@xs...	Физ. ли...	<input type="checkbox"/>
☰	Калмык...	Илья	Роберт...	899912...	14.12.1...	info@xs...	Физ. ли...	<input type="checkbox"/>

Помимо этого, в DATAGRID можно менять последовательность колонок простым перетаскиванием заголовка колонки в необходимое место. Также, существующие строки можно дублировать.



The image shows a table with three columns. The first column contains a menu icon (three horizontal lines). The second column contains the name 'Федотов', and the third column contains the name 'Игорь'. A context menu is open over the first row, showing three options: '+ Добавить строку', 'Дублировать строку', and 'Удалить строку'. The table has several rows, with the first row highlighted. The other rows contain names: 'Ан', 'Ро', 'Гер', 'Ль', 'Се', and 'Ал'.

	Федотов	Игорь	Ан
+ Добавить строку			Ро
Дублировать строку			Гер
Удалить строку			Ль
			Се
			Ал

### Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты таблицы.

### Параметры региона

Параметры компонента стандартные для региона.

## Атрибуты региона

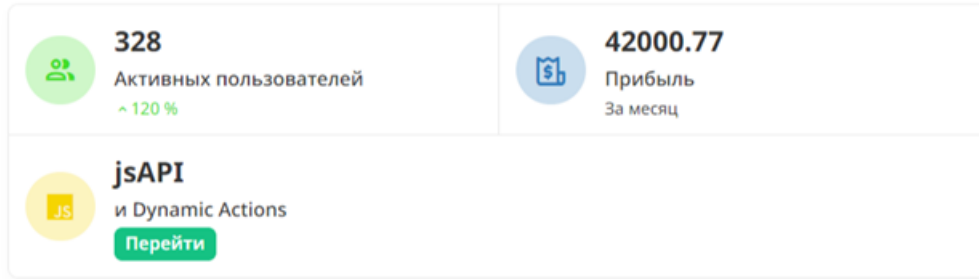
Атрибут	Тип	Описание
Редактирование - разрешено	Переключатель	Определяет возможность редактирования содержимого таблицы.
Разрешенные операции	Множественный выбор	<p>Список доступных операции в таблице:</p> <ul style="list-style-type: none"> <li>• Вставка</li> <li>• Обновление</li> <li>• Удаление</li> </ul> <p>Настройка разрешенных операций доступна если разрешено редактирование.</p>
Потерянные обновления	Список	<p>Атрибут определяет способ, которым предотвращается потеря данных при совместной работе нескольких пользователей с одним и тем же массивом данных. Доступные значения:</p> <ul style="list-style-type: none"> <li>• Значение строки.</li> </ul> <p>Перед отправкой изменений система вычисляет хэш измененной строки и сравнивает с хэшем, полученном при загрузке данных. Если хэши расходятся, система не позволит произвести обновление (кто-то другой уже поменял строку).</p>
Авторизация на редактирование	Список	Определяет какая схема авторизации пользователей используется для допуска к выбранной операции в компоненте.
Отобразить NULL как	Текст	Определяет, что будет выведено в ячейке таблицы в случае отсутствия значения (Null).
Панель инструментов – Показать панель	Переключатель	Определяет отображение панели инструментов компонента.
Панель инструментов – Кнопки панели	Множественный выбор	<p>Определяет набор кнопок на панели инструментов, если включено ее отображение. Доступны следующие компоненты панели инструментов:</p> <ul style="list-style-type: none"> <li>• Столбцы поиска</li> <li>• Поле поиска</li> <li>• Действия</li> <li>• Сохранить</li> </ul>
Шаблоны	Текст / Окно конструктора	Позволяет указать шаблон по умолчанию.

### 7.6.16 Плитки (TILES)

TILES – служит для отображения статистической информации. Источником данных для компонента TILES является результат SQL-запроса. Для настройки внешнего вида плиток используются определенные параметры SQL-запроса, например:

```
SELECT
  '42000.77' title,      -- Главная подпись (заголовок)
  'Прибыль' text,      -- Подпись, описывающая заголовок
  '#2C7ABV' color,     -- HEX-код цвета оформления плитки
  'За месяц' descr,    -- Вспомогательная подпись под основным текстом
  'uil uil-bill' icon  -- CSS класс иконки
```





### Настройка компонента

Доступны настройки двух уровней: параметры региона и атрибуты таблицы.

### Параметры региона

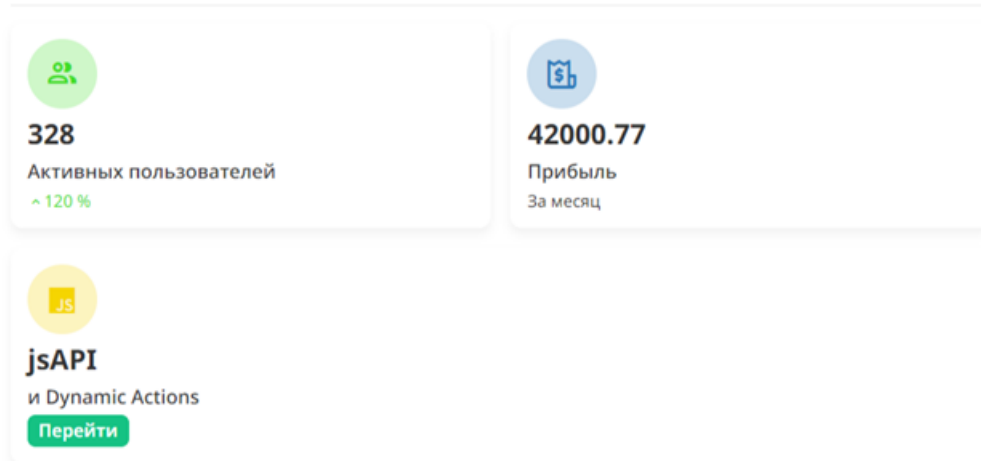
Отличительной особенностью компонента типа “Отчёт” является необходимость обязательного заполнения поля Источник - SQL. Необходимо указать SQL-запрос, на основе которого формируется список и внешний вид плиток.

## Атрибуты региона

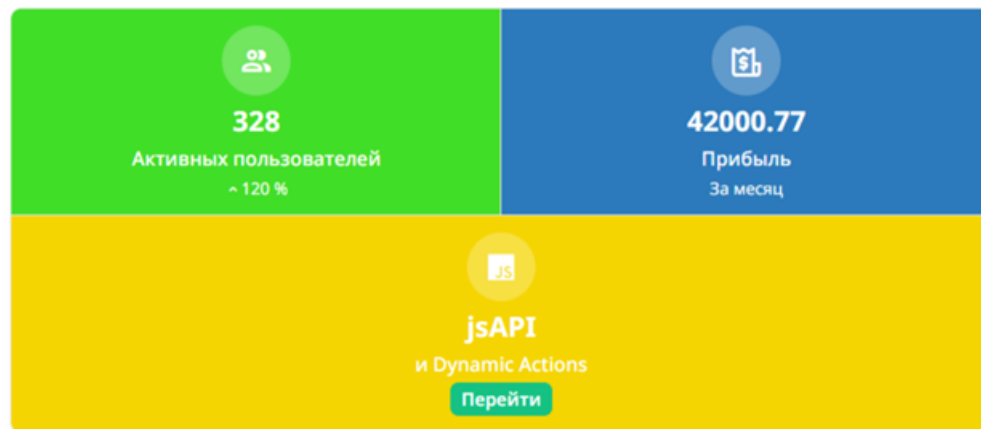
Атрибут	Тип	Описание
Тема	Список	<p>Определяет предопределенную тему отображения плиток:</p> <ul style="list-style-type: none"> <li>• Классическая - отображает плитки с иконкой сверху. Цветовая настройка в данном варианте определяет цвет иконки.</li> <li>• Контрастная – аналогична классической, но цветовая настройка определяет цвет фона плитки.</li> <li>• Горизонтальная - компактное отображение с иконкой слева. Цветовая настройка определяет цвет иконки.</li> <li>• Горизонтально-контрастная – компактное отображение, где цветовая настройка определяет фон плитки</li> </ul>
Колонки	Список	<p>Определяет настройки сетки для отображения компонента. При указании параметра «auto» компонент самостоятельно распределит все элементы по строкам. Ширина каждого элемента в строке будет одинакова. Параметр «auto-float» работает также как и «auto», но ширина каждого элемента будет соответствовать содержанию компонента.</p>
Вид	Список	<p>Определяет внешний вид компонента.</p> <ul style="list-style-type: none"> <li>• Сетка – плоское отображение без промежутков между плитками.</li> <li>• Раздельный – отображение плиток с промежутками и тенями</li> </ul>
Выравнивание	Переключатель	<p>Определяет тип выравнивания содержимого в плитке:</p> <ul style="list-style-type: none"> <li>• По левому краю</li> <li>• По центру</li> <li>• По правому краю</li> </ul>
Контрастные иконки	Переключатель	<p>Определяет контрастный режим отображения. В контрастном режиме иконки отображаются белым цветом на цветном фоне.</p>
Колонка с заголовком	Список	<p>Определяет наименование параметра для отображения заголовка плитки.</p>
Колонка с текстом	Список	<p>Определяет наименование параметра для отображения основного текста плитки.</p>
Колонка с описанием	Список	<p>Определяет наименование параметра для отображения дополнительного текста плитки.</p>
Колонка с иконкой	Список	<p>Определяет наименование параметра для отображения иконки плитки.</p>
Колонка с цветом	Список	<p>Определяет наименование параметра для управления цветом плитки. Необходимо указывать цветовое значение в формате hex.</p>

Для плиток реализовано несколько разных тем, по аналогии с компонентом CARDS. Для текста внутри плитки также можно выбрать разные варианты выравнивания: слева, справа и по центру. Помимо тем отображения плиток есть также два вида расположения плиток: в виде цельной сетки элементов и в виде отдельных элементов. Для наглядности возможные варианты – изображены на скриншотах:

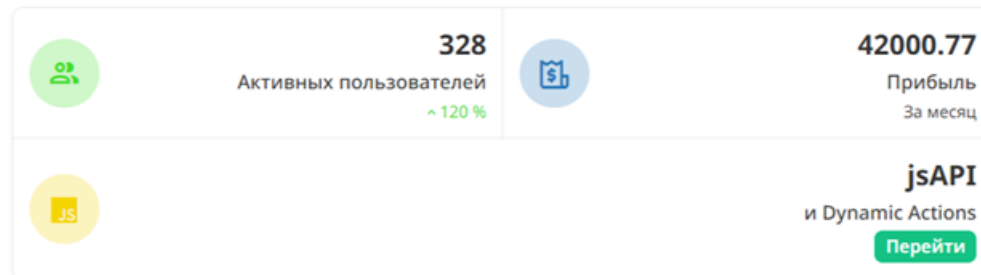
Классическая тема в 3 колонки с раздельными плитками



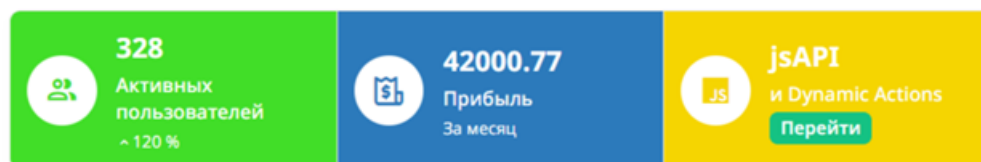
Контрастная тема в 2 колонки с выравниванием по центру



Горизонтальная тема в 1 колонку с выравниванием справа



Горизонтально-контрастная тема с контрастными иконками



## 7.7 Работа со списками

В разрабатываемых приложениях часто используются такие элементы как меню, путь к текущей странице (breadcrumb), панели навигации и другое. Для вывода в них информации можно использовать как динамически формируемый набор значений (с помощью SQL-запроса), так и статичный – строго определенный список.

### 7.7.1 Что такое список

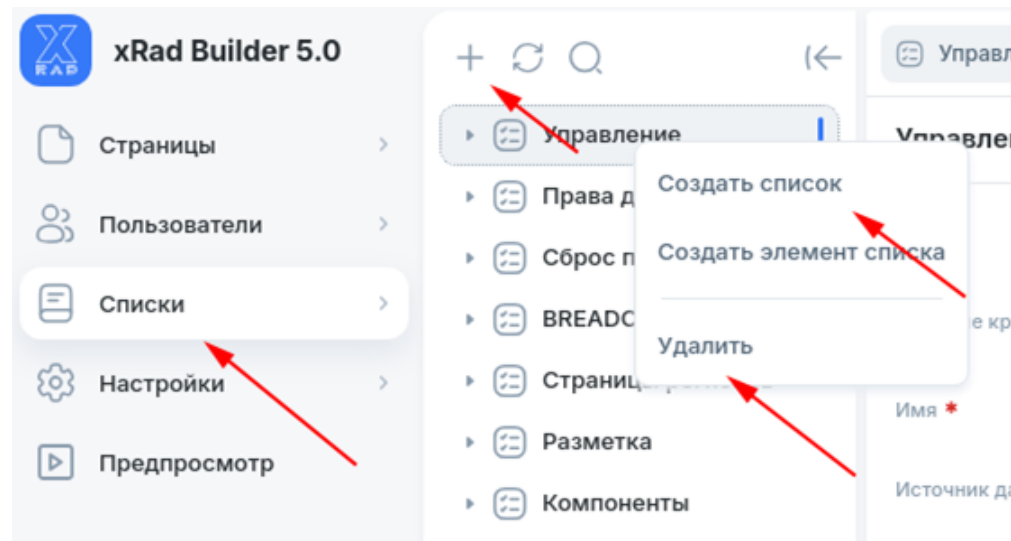
Список – это набор однотипных данных.

В XRAD существует 3 типа списков:

1. Статичный (Static) – список создается и редактируется разработчиком в среде XRAD.
2. Динамический (Based on Query) – список создается на основе SQL-запроса. Может меняться пользователем через интерфейс веб-приложения.
3. Breadcrumb – специальный тип, предназначенный для вывода пути до страницы приложения.

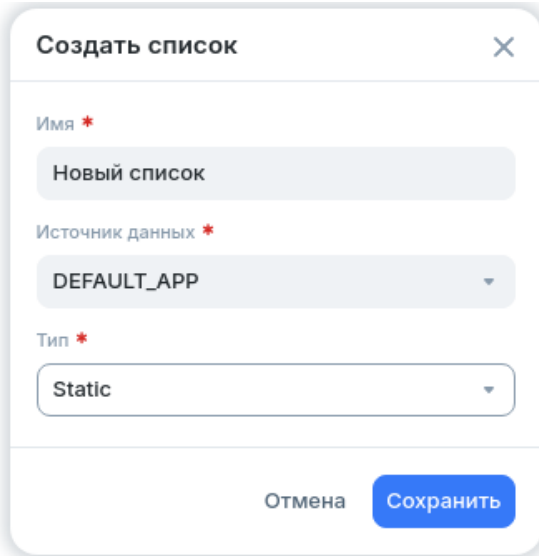
### 7.7.2 Создание, редактирование, удаление списков

Для создания списка необходимо выбрать в главном меню раздел «Списки» и нажать кнопку «+» в подменю. Также для создания или удаления списка можно использовать контекстное меню, нажав ПКМ на области, где перечислены списки и выбрать соответствующее действие.



Создание нового списка происходит во всплывающем окне, где необходимо указать:

- Имя – обязательный атрибут, определяющий имя списка.
- Источник данных – обязательный атрибут, определяющий БД для хранения списка.
- Тип списка – Static, Based on Query, Breadcrumb



Создать список

Имя \*

Новый список

Источник данных \*

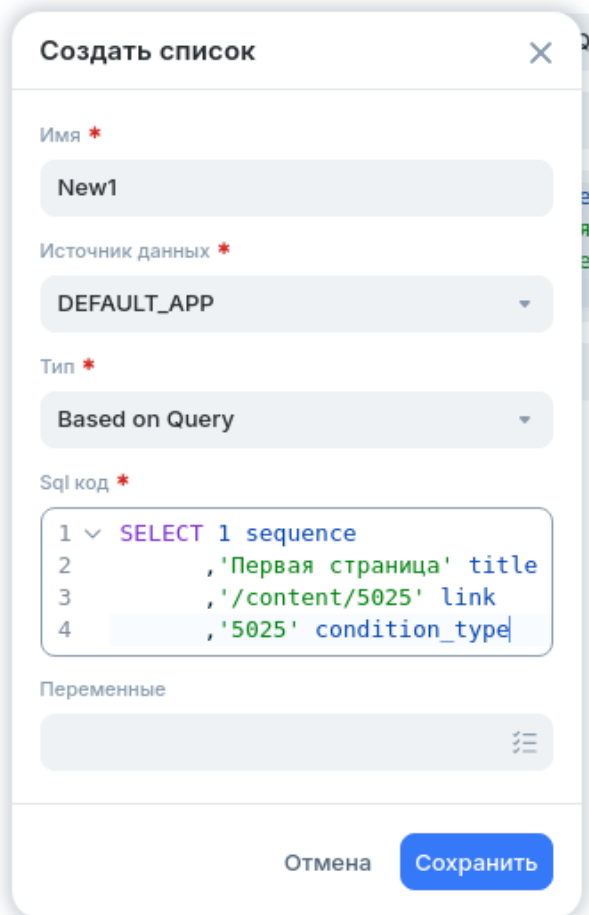
DEFAULT\_APP

Тип \*

Static

Отмена Сохранить

При создании динамического списка (Based on Query) необходимо указать SQL-запрос, на основе которого будет строиться список и, по необходимости, переменные, которые используются в запросе.



Создать список

Имя \*

New1

Источник данных \*

DEFAULT\_APP

Тип \*

Based on Query

Sql код \*

```
1 SELECT 1 sequence
2     , 'Первая страница' title
3     , '/content/5025' link
4     , '5025' condition_type
```

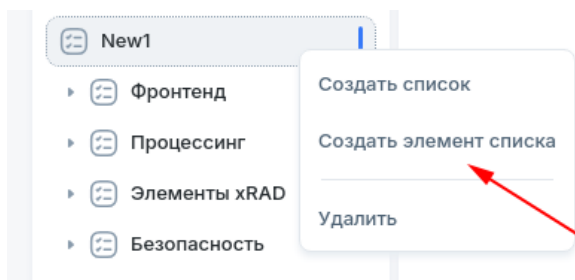
Переменные

Отмена Сохранить

После нажатия кнопки «Сохранить» будет создан новый список, к которому можно добавить элементы списка.

### 7.7.3 Создание элементов списка

Для создания элементов списка необходимо с помощью ПКМ вызвать контекстное меню и выбрать «Создать элемент списка».



В новой вкладке заполнить атрибуты нового элемента списка.

Новый элемент списка New1
Сохранить

---

**Элемент списка** ▾

Родительский элемент

Имя \*

Источник данных \*

Порядковый номер \*

Css класс иконки

Подсказка

Условие для активного состояния

Отделён

**Назначение** ▾

Ссылка

**Условия** ▾

Тип Условия

**Пользовательские атрибуты** ▾

Атрибут 1

Атрибут 2

Родительский элемент - родительский пункт списка. Предназначен для формирования многоуровневого меню.

- Имя – обязательный атрибут, определяющий имя элемента списка.
- Источник данных – обязательный атрибут, определяющий источник данных.
- Порядковый номер - обязательный атрибут, определяющий порядок отображения элемента в списке.
- Css класс иконки – определяет отображаемую иконку.
- Подсказка – определяет текстовую подсказку элемента.
- Активен для страниц – список номеров страниц, на которых данный элемент списка будет выделен как активный.
- Отделен – Проставляет разделитель между пунктами, если список используется для формирования нави-

гационного меню, находящегося сверху страницы.

- Ссылка – действия, которое будет выполняться при выборе элемента списка
- Тип условия – определяет тип условия, которое должно быть выполнено для перехода по ссылке.

Атрибуты 1, 2 - для некоторых вариантов 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

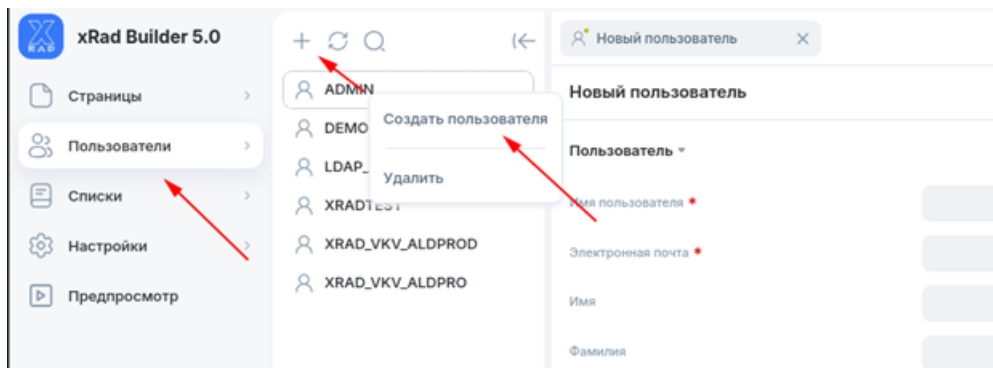
## 7.8 Управление пользователями

XRAD предоставляет возможность управлять пользователями напрямую в среде разработки. Внутренние пользователи среды разработки проходят аутентификацию по логину и паролю (схема CUSTOM), либо сопоставляются по имени с внешними пользователями, которые проходят аутентификацию в соответствии с выбранной схемой. Например, пользователь с именем DEVELOPER может войти в систему по логину и паролю, а может быть сопоставлен с пользователем DEVELOPER из Active Directory или OIDC. Внутренние пользователи XRAD имеют роль, которая определяет права доступа. Процессы создания и редактирования пользователей довольно минималистичны, но имеют весь необходимый функционал. Определены три основные роли, с помощью которых можно разграничить зоны доступа для пользователей:

- **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 назначать строго администратору, для разработчиков достаточно прав роли DEVELOPER.
- **DEVELOPER** - роль разработчика. DEVELOPER может просматривать и изменять Страницы, создавать и удалять их. Имеет право редактировать Списки и Настройки. 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 инструменты кроме управления Пользователями.
- **VIEW** - роль с минимальным набором привилегий. VIEW может просматривать Страницы и их содержимое, а также Списки и Настройки (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.

### 7.8.1 Создание, редактирование, удаление пользователей

Для создания пользователя необходимо выбрать в главном меню раздел «Пользователи» и нажать кнопку «+» в подменю. Также для создания или удаления пользователя можно вызвать контекстное меню, нажав ПКМ на области, где перечислены пользователи и выбрать соответствующее действие.



После заполнения или редактирования параметров выбранной учетной записи необходимо подтвердить внесенные изменения, нажав кнопку «Сохранить».



Параметр	Описание
Имя пользователя	обязательный атрибут, определяющий имя создаваемого пользователя, используется в качестве логина.
Электронная почта	обязательный атрибут, определяющий адрес электронной почты пользователя.
Имя, Фамилия	атрибуты для внесения имени и фамилий человека, закрепленного за данным пользователем.
Роль	обязательный атрибут, определяющий одну из возможных ролей пользователя.
Заблокирован	флаг, определяющий блокировку пользователя в системе.

Обязательно необходимо указать и подтвердить пароль пользователя, а также можно выставить флаг «Требовать изменение пароля» при необходимости смены пароля после первичного входа. Примечание: при редактировании параметров пользователя изменение поля «Имя пользователя» – недоступно.

## 7.9 Стили и темы

Разработчик может управлять стандартными стилями через кастомные CSS-свойства.

Подключить свои CSS-свойства можно двумя способами:

1. В среде разработки XRAD в разделе настроек страницы добавить CSS-свойства в поле «Встроенный CSS» (как локальные свойства).
2. Добавить css между тегами <head></head> в файле index.html веб-контроллера PGHS (как глобальные свойства).

Глобальные кастомные свойства PGHS применяются к псевдо-классу: root, локальные (в рамках страницы) применяются к классу: page-{ID}, где {ID} - идентификатор страницы.

### Внимание

*Указывайте только те переменные, которые нужно переопределить. Значения по умолчанию уже определены. Нет необходимости копировать весь пример на страницу*

Если нужно переопределить только основной цвет, достаточно указать значение `--color-primary`.

```
:root {
  --color-primary: #1f78ff;
}
```

Перечень кастомных CSS свойств PGHS и пример их использования:

```
:root {
  --color-primary: #1f78ff; /* Основной цвет*/

  --color-danger: #f42525; /* Цвет состояния "ошибка" (подсветка полей, всплывающие
  ↳подсказки) */
  --color-warning: #ffb800; /* Цвет состояния "предупреждение" (всплывающие
  ↳подсказки) */
  --color-success: #15c283; /* Цвет состояния "успех" (всплывающие подсказки) */
  --color-info: #17a2b8; /* Цвет состояния "информирование" (всплывающие подсказки)
  ↳*/
  --color-link: #1f78ff; /*Цвет ссылки, наследуется от --color-primary */

  --global-bg-color: #f9fafb; /* Цвет фона страницы */
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

/* Кнопки */
--btn-radius: 8px; /* Закругления кнопок */
--btn-lg-radius: 12px; /* Закругления кнопок с модификатором btn-lg (large) */
--btn-sm-radius: 6px; /* Закругления кнопок с модификатором btn-sm (small) */
--btn-xs-radius: 6px; /* Закругления кнопок с модификатором btn-xs (extra small) */
/* .btn */
--btn-color: transparent; /* Цвет фона кнопки .btn (стандарт - без модификатора ↪
↪цвета) */
--btn-border-color: #eaeaea; /* Цвет фона кнопки .btn (стандарт - без модификатора ↪
↪цвета) */
--btn-text-color: #333; /* Цвет текста кнопки .btn (стандарт - без модификатора ↪
↪цвета) */
/* .btn-primary */
--btn-primary-color: #1f78ff; /* Цвет фона кнопки .btn-primary, наследуется от --
↪color-primary */
--btn-primary-text-color: #fff; /* Цвет текста кнопки .btn-primary */
/* .btn-secondary */
--btn-secondary-color: #f3f3f4; /* Цвет фона кнопки .btn-secondary */
--btn-secondary-text-color: #1f78ff; /* Цвет текста кнопки .btn-secondary, ↪
↪наследуется от --color-primary */
/* .btn-success */
--btn-success-color: #15c283; /* Цвет фона кнопки .btn-success, наследуется от --
↪color-success */
--btn-success-text-color: #fff; /* Цвет текста кнопки .btn-success */
/* .btn-danger */
--btn-danger-color: #f42525; /* Цвет фона кнопки .btn-danger, наследуется от --
↪color-danger */
--btn-danger-text-color: #fff; /* Цвет текста кнопки .btn-danger */
/* .btn-warning */
--btn-warning-color: #ffb800; /* Цвет фона кнопки .btn-warning, наследуется от --
↪color-warning */
--btn-warning-text-color: #333; /* Цвет текста кнопки .btn-warning */
/* .btn-info */
--btn-info-color: #17a2b8; /* Цвет фона кнопки .btn-info, наследуется от --color-
↪info */
--btn-info-text-color: #fff; /* Цвет текста кнопки .btn-info */

/*Индикатор загрузки*/
--loader-color: #1f78ff; /* Цвет индикатора загрузки --color-info, наследуется от -
↪color-primary */
--loader-width: 80px;
--loader-width-sm: 1rem; /* Размер индикатора загрузки .spinner-border-sm */

/* Шрифты */
--font-family: "Noto Sans", sans-serif; /* Название, семейство шрифта */

/* Типографика */
--text-title-color: #333; /* Стандартный цвет заголовков */
--text-color: #333; /* Стандартный цвет текста */
--text-font-size: 15px; /* Стандартный размер текста */
--text-font-weight: 400; /* Стандартная жирность текста */

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

/* Прокрутка страницы */
--scroll-color: #bfbfbf; /* Цвет индикатора прокрутки */
--scroll-bg-color: transparent; /* Фоновый цвет индикатора прокрутки */

/* Сетка */
--region-grid_col_gap: 12px; /* Отступы между колонками (col) */
--regions-grid_row_gap: 12px; /* Отступы между рядами (row) */

/* Страница */
--page-position-border: 1px solid #eaeaea; /* border (линия-разделитель) между
↪ позициями страницы: sidebar (навигационное меню), top (верх), left (лево), right
↪ (право), body (центр), footer (подвал) */
--resizer-hover-color: #1f78ff; /* Цвет resizer (контроля управления шириной
↪ блока) при наведении */

/* sidebar (навигационное меню) */
--sidebar-width: 250px; /* Ширина sidebar (навигационного меню) */
--sidebar-border: 1px solid #eaeaea; /* border (линия-разделитель) sidebar
↪ (навигационного меню), наследуется от --page-position-border */
--sidebar-bg: #fff; /* Фоновый цвет sidebar (навигационного меню) */
--sidebar-text-color: #4b5667; /* Цвет текста sidebar (навигационного меню) */
--sidebar-font-size: 12px; /* Размер шрифта sidebar (навигационного меню) */
--sidebar-active-bg: #f6f7f9; /* Фоновый цвет активного пункта меню в sidebar
↪ (навигационного меню) */
--sidebar-active-border-color: #1f78ff; /* Фоновый цвет границы справа активного
↪ пункта меню в sidebar (навигационного меню), наследуется от --color-primary*/
--sidebar-active-bg-radius: 4px; /* Закругление активного пункта меню в sidebar
↪ (навигационного меню) */
--sidebar-icon-color: #4b5667; /* Цвет иконок пункта меню в sidebar
↪ (навигационного меню), наследуется от --sidebar-text-color */
--sidebar-drp-icon-color: #888888; /* Цвет иконки открытия/закрытия дочерних
↪ элементов пункта меню в sidebar (навигационного меню) */

/* left (лево) */
--page-left-position-width: 250px; /* Ширина позиции left (лево), наследуется от --
↪ sidebar-width */
--page-left-position-padding: 8px; /* Внутренний отступ позиции left (лево) */
--page-left-position-border: 1px solid #eaeaea; /* border (линия-разделитель)
↪ позиции left (лево), наследуется от --page-position-border */
--page-left-position-bg-color: #fff; /* Фоновый цвет позиции left (лево) */

/* body (центр) */
--page-body-position-padding: 8px; /* Внутренний отступ позиции body (центр) */

/* right (право) */
--page-right-position-width: 250px; /* Ширина позиции right (право), наследуется
↪ от --sidebar-width */
--page-right-position-padding: 8px; /* Внутренний отступ позиции right (право) */
--page-right-position-border: 1px solid #eaeaea; /* border (линия-разделитель)
↪ позиции right (право), наследуется от --page-position-border */
--page-right-position-bg-color: #fff; /* Фоновый цвет позиции right (право) */

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

/* top (верх) */
--page-top-position-padding: 8px; /* Внутренний отступ позиции top (верх) */
--page-top-position-border: 1px solid #eaeaea; /* border (линия-разделитель) ↵
↪позиции top (верх), наследуется от --page-position-border */
--page-top-position-bg-color: #fff; /* Фоновый цвет позиции top (верх) */

/* footer (низ) */
--page-footer-position-padding: 8px; /* Внутренний отступ позиции footer (низ) */
--page-footer-position-border: 1px solid #eaeaea; /* border (линия-разделитель) ↵
↪позиции footer (низ), наследуется от --page-position-border */
--page-footer-position-bg-color: transparent; /* Фоновый цвет позиции footer (низ) ↵
↪*/

/* header (шапка) для видов страниц: Стандартный, минималистичный */
--header-bg-color: #1f78ff; /* Фоновый цвет header (шапки), наследуется от --color-
↪primary */
--header-text-color: #fff; /* Цвет текста в header (шапке) */
--header-height: 48px; /* Высота header (шапки) */
--header-menu-btn-border-color: transparent; /* Цвет обводки кнопки навигационного ↵
↪меню */
--header-controls-border-width: 1px; /* Размер обводки кнопки навигационного меню ↵
↪*/
--header-menu-btn-bg-color: rgba(
    0,
    0,
    0,
    0.1
); /* Фоновый цвет кнопки навигационного меню */
--header-controls-radius: 6px; /* Закругления кнопки навигационного меню */
--app-logo-text-font-size: 15px; /* Размер шрифта названия приложения */
--app-logo-text-font-weight: 600; /* Жирность шрифта названия приложения */
--app-logo-text-line-height: 20px; /* Высота строки названия приложения */
--app-logo-display: none; /* (none / block) Отображение логотипа (справа от кнопки ↵
↪навигационного меню) */
--app-logo-url: none; /* (none / url("/files/images/my_logo.svg")) Картинка ↵
↪логотипа (справа от кнопки навигационного меню), применяется как background-image */
--app-logo-size: 32px; /* Размер логотипа (справа от кнопки навигационного меню), ↵
↪применяется как background-size */
--app-logo-position-top: 0; /* Отступ логотипа (справа от кнопки навигационного ↵
↪меню) сверху, применяется как background-position (top) */
--app-logo-position-left: 0; /* Отступ логотипа (справа от кнопки навигационного ↵
↪меню) слева, применяется как background-position (left) */
--navbar-icon-radius: 6px; /* Закругления блока иконки навигационного меню */
--navbar-icon-bg-color: rgba(
    0,
    0,
    0,
    0.1
); /* Фоновый цвет иконки навигационного меню */
--navbar-icon-color: #fff; /* Цвет иконки навигационного меню, наследуется от --
↪header-text-color */

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

--navbar-icon-order: 3; /* Порядок отображения (flex order) для иконки
↳навигационного меню */
--navbar-icon-margin: 0 0 0 8px; /* Внешний отступ иконки навигационного меню */
--navbar-text-order: 2; /* Порядок отображения (flex order) для текста пункта
↳навигационного меню */
--navbar-arrow-order: 1; /* Порядок отображения (flex order) для иконки пункта
↳выпадающего списка навигационного меню (стрелка вниз) */

/* Регионы */
--region-bg-color: #fff; /* Фоновый цвет региона */
--region-border-color: #eaeaea; /* Цвет обводки региона */
--region-border-width: 1px; /* Ширина обводки региона */
--region-box-shadow: 0px 4px 7px 0px rgba(0, 0, 0, 0.02); /* Тень региона */
--region-padding: 12px; /* Внутренние отступы региона */
--region-radius: 12px; /* Закругления региона */
--region-head-separated-gap: 12px; /* Отступ снизу от отделённого заголовка
↳региона (в настройках региона), наследуется от --region-padding */

/* Модальные окна */
--modal-box-shadow: 0px 18px 30px 0px rgba(51, 51, 51, 0.64); /* Тень модального
↳окна */
--modal-radius: 20px; /* Закругления модального окна */
--modal-controls-btn-icon-color: #888888; /* Цвет иконок в хедере модала, справа
↳от заголовка (кнопка закрыть) */

/* Элементы формы (ITEMS) */
--item-placeholder-color: #888; /* Цвет лейбла */
--item-text-color: #333; /* Цвет текста, наследуется от --text-color */
--item-border-radius: 8px; /* Закругления */
--item-box-shadow: 0px 2px 5px 0px rgba(85, 114, 157, 0.11) inset; /* Тень */
--item-border-color: #e4e5e7; /* Цвет обводки */
--item-focus-color: #1f78ff; /* Цвет обводки в состоянии :focus, наследуется от --
↳color-primary */
--item-error-color: #f42525; /* Цвет обводки в состоянии ошибки и цвет текста
↳ошибки под полем, наследуется от --color-danger */
--disabled-items-bg-color: #f9fafb; /* Фоновый цвет в состоянии :disabled,
↳:readonly */
--disabled-items-text-color: #aaa; /* Цвет текста в состоянии :disabled, :readonly
↳*/
--item-control-inside-icon-color: #aaaaaa; /* Цвет иконки внутри элемента (тултип,
↳стрелка селекта и пр.)*/

/* Чекбоксы */
--checkbox-border-color: #eaeaea; /* Цвет обводки чекбокса */
--checkbox-checked-bg-color: #1f78ff; /* Цвет фона выбранного чекбокса */
--checkbox-icon-color: #fff; /* Цвет иконки выбранного чекбокса */
--checkbox-radius: 4px; /* Закругления чекбокса */

/* Радиокнопки */
--radio-border-color: #eaeaea; /* Цвет обводки радиокнопки */
--radio-checked-bg-color: #1f78ff; /* Цвет фона выбранного радиокнопки */
--radio-icon-color: #fff; /* Цвет иконки выбранного радиокнопки */

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

--radio-radius: 100%; /* Закругления радиокнопки */

/* Переключатель (switcher) */
--switcher-bg-color: #f3f3f4; /* Фоновый цвет переключателя*/
--switcher-separator-color: #d9dfe8; /* Фоновый цвет разделителя (когда не выбран
↳ ни один элемент) */
--switcher-radius: 6px; /* Закругления переключателя */
--switcher-btn-radius: 4px; /* Закругления кнопки переключателя */
--switcher-btn-active-bg-color: #fff; /* Фоновый цвет */
--switcher-btn-active-box-shadow: 0px 2px 8px 0px rgba(0, 0, 0, 0.08); /* */
--switcher-btn-text-color: #333; /* Цвет текста кнопки переключателя, наследуется
↳ от --text-color */
--switcher-active-btn-text-color: #333; /* Цвет текста кнопки переключателя в
↳ активном состоянии, наследуется от --text-color */
--switcher-lg-radius: 8px; /* Закругления переключателя (при отсутствии лейбла) */
--switcher-lg-btn-radius: 6px; /* Закругления кнопки переключателя (при отсутстви
↳ лейбла) */

/* Dropdown (выпадающий список, элемент управления) */
--dropdown-bg-color: #fff; /* Фоновый цвет */
--dropdown-border-color: #eaeaea; /* Цвет обводки */
--dropdown-radius: 8px; /* Закругление */
--dropdown-box-shadow: 0px 3px 6px 0px rgba(0, 0, 0, 0.05), 0px 11px 11px 0px
    rgba(0, 0, 0, 0.04), 0px 25px 15px 0px rgba(0, 0, 0, 0.03); /* Тень */
--dropdown-active-bg: #f6f7f9; /* Фоновый цвет активного пункта меню */
--dropdown-active-border-color: #1f78ff; /* Фоновый цвет границы справа активного
↳ пункта меню, наследуется от --color-primary */
--dropdown-active-bg-radius: 4px; /* Закругление активного пункта меню */

/* Календарь (ITEM type DATE) */
--datepicker-days-text-color: #aaa; /* */
--datepicker-days-text-color: #333; /* , наследуется от --text-color */
--datepicker-active-bg-color: #1f78ff; /* Фоновый цвет активной даты, наследуется
↳ от --color-primary */
--datepicker-active-text-color: #fff; /* Цвет текста активной даты */
--datepicker-hover-bg-color: rgba(
    45,
    52,
    62,
    0.06
); /* Фоновый цвет даты наведения */

/* Поле Файл */
--item-file-radius: 10px; /* Закругления */
--item-file-border-color: #888; /* Цвет обводки */
--item-file-placeholder-color: #888; /* Цвет текста в зоне перетаскивания
↳ (dropzone) */
--item-file-msg-color: #888; /* Цвет текста поясняющего сообщения под полем */

/* Вкладки (TABS) */
--tabs-padding: 0 12px 0 12px; /* Внутренний отступ блока вкладок */
--tab-padding: 12px 0 12px 0; /* Внутренний отступ кнопки-переключателя вкладок */

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

--tabs-gap: 24px; /* Расстояние между кнопками-переключателями */
--tabs-border-width: 1px; /* Ширина границы блока вкладок, наследуется от --region-
↪border-width */
--tabs-border-color: #eaeaea; /* Цвет границы блока вкладок, наследуется от --
↪region-border-color */
--tabs-color-border-active: #2d343e; /* Цвет границы активной вкладки */
--tabs-color-text: #888; /* Цвет текста кнопок-переключателей */
--tabs-color-text-active: #333; /* Цвет текста активной кнопки-переключателя */

/* Карточки (CARDS) */
--cards-radius: 12px; /* Закругления, наследуется от --region-radius */
--cards-border-color: #eaeaea; /* Цвет границы, наследуется от --region-border-
↪color */
--cards-border-width: 1px; /* Ширина границы, наследуется от --region-border-
↪width*/
--cards-box-shadow: 0px 4px 7px 0px rgba(0, 0, 0, 0.02); /* Тень, наследуется от --
↪region-box-shadow*/
--cards-bg-color: #fff; /* Фоновый цвет */

/* Отчёт (REPORT) */
--report-th-text-color: #888; /* Цвет текста ячеек <th> в <thead> */
--report-th-font-size: 14px; /* Размер шрифта ячеек <th> в <thead> */
--report-th-line-height: 16px; /* Высота строки ячеек <th> в <thead> */
--report-th-font-weight: 400; /* Жирность шрифта ячеек <th> в <thead> */
--report-td-text-color: #333; /* Цвет текста ячеек <td> в <tbody> , наследуется от
↪--text-color */
--report-td-font-size: 14px; /* Размер шрифта ячеек <td> в <tbody> */
--report-td-line-height: 16px; /* Высота строки ячеек <td> в <tbody> */
--report-td-font-weight: 400; /* Жирность шрифта ячеек <td> в <tbody> */
--report-cellpadding: 8px 12px; /* Внутренние отступы в ячейках */
--report-border-width: 1px; /* Ширина границы */
--report-border-color: #d9dfe8; /* Цвет границы */
--report-stripe-bg-color: #f9fafb; /* Фоновый цвет чётных строк */
--report-acs-desc-active-color: #333; /* Цвет иконки активной сортировки,
↪наследуется от --report-td-text-color */
--report-hover-bg: #d8e7ee; /* Фоновый цвет строки при наведении */
--report-pagination-text-color: #888; /* Цвет текста пагинации */
--report-pagination-button-color: #1f78ff; /* Цвет кнопок пагинации, наследуется
↪от --color-primary */
--report-pagination-font-size: 14px; /* Размер шрифта пагинации*/
--report-pagination-line-height: 16px; /* Высота строки пагинации */
--report-footer-padding: 8px 12px; /* Внутренний отступ блока пагинации,
↪наследуется от --report-cellpadding */
--report-header-padding: 12px; /* Внутренний отступ верхнего блока с поиском и
↪фильтрами, наследуется от --region-padding */

/* Пошаговая навигация (WIZARD) */
--wizard-unactive-bg-color: #d9dfe8; /* Фоновый цвет неактивного/незавершенного
↪состояния */
--wizard-active-bg-color: #1f78ff; /* Фоновый цвет активного/завершенного
↪состояния, наследуется от --color-primary */
--wizard-step-label-color: #333; /* Цвет текста лейблов, наследуется от --text-

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

↪color */
  --wizard-step-counter-color: #888888; /* Цвет текста внутри точки неактивного/
↪незавершенного состояния */
  --wizard-step-active-counter-color: #fff; /* Цвет текста внутри точки активного/
↪завершенного состояния */
}

```

## 7.10 Справочник jsAPI

Объект jsAPI служит инструментом взаимодействия клиентской части приложения с сервером и фронтендом PGHS, направлен на упрощение разработки визуального интерфейса приложения, 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))

### 7.10.1 jsAPI: submit()

Осуществляет обработку (сабмит) страницы методом {API\_URL}/processPage

#### Синтаксис

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

#### Параметры

items Object Значения items, отправляемые в теле запроса {APIURL}/processPage, помимо основных items страницы. Допускается использовать пустой объект {}, если не требуется отправки кастомных значений.

callbacks Object Принимает функции onSuccess Function, onError Function.

Примечание: При кастомном обработчике 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 обработчике не будет перезагрузки страницы");
    },
    onError: function (err) {
      console.error(err); //Обработчик ошибки сервиса {API_URL}/processPage
    }
  }
);

```



**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
    }
  }
);
```

**7.10.2 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 требуется отправки значений.
- callbacks `Object` - Принимает функции: onSuccess Function, onError Function.
- onSuccess `Function` - коллбэк, содержит ответ сервера (res)
- onError `Function` - коллбэк, содержит ошибку сервера (err)

**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); // Ответ сервера
    },
    onError: function (err) {
      console.error(err); // Ошибка сервера
    }
  }
);
```

### 7.10.3 jsAPI: reload()

Осуществляет рендеринг страницы после успешного ответа метода {API\_URL}/showPage?page=id

#### Синтаксис

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

#### Параметры

Данная функция не принимает аргументов.

*Примечание* не является аналогом window.location.reload(). Метод вызывает ререндеринг страницы после успешного ответа метода {API\_URL}/showPage, как это делается при смене роута.

#### Примеры

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

### 7.10.4 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();
```

## 7.10.5 jsAPI: redirect()

Осуществляет переход на указанную страницу приложения.

### Синтаксис

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

### Параметры

- path <sup>String</sup> - Относительный путь до страницы
- callbacks - принимает функцию: onSuccess <sup>Function</sup>

*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);
  }
});
```

## 7.10.6 jsAPI: getCurrentPage()

Возвращает информацию о текущей странице в виде:

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

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

    ...
  },
  "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"];
```

**7.10.7 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 <sup>String, Array</sup> - 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)
```

## 7.10.8 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)
- minHeight `Number` - Минимальная высота окна, по умолчанию 240 v4+
- centered `Boolean` - Вертикальное выравнивание по центру страницы, по умолчанию false v4+
- callbacks `Object` Принимает функции:
- onClose `Function` - Выполняется при закрытии окна с использованием методов jsAPI.modal.close();, jsAPI.modal.accept();
- onAccept `Function` - Выполняется при закрытии окна с использованием метода jsAPI.modal.accept();
- onDecline `Function` - Выполняется при закрытии окна с использованием 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**

**скому 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()
    }
  }
);

```

**7.10.9 jsAPI: confirmModal()**

Отображает окно для подтверждения какого-либо действия.

**Синтаксис**

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

**Параметры**

- options <sup>Object</sup> - Содержит параметры:
- title <sup>String</sup> - Заголовок окна;
- text <sup>String</sup> - Текст, отображаемый в диалоговом окне;
- callbacks Принимает функции:
- onAccept <sup>Function</sup> - Выполняется при нажатии кнопки «Да»
- onDecline <sup>Function</sup> - 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,
    buttons: [ {
      text: "Да",
      action: "accept",
      class: "btn-primary"
    }
  ]
);

```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```

    },
    {
      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"
    }
  }
);

```

**7.10.10 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 <sup>Object</sup> - Данные для отправки в теле запроса
- callbacks <sup>Object</sup> - 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",
  items: {
    P1000_ID: "1"
  }
});
```

#### Стандартный вызов с обработчиками:

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

### 7.10.11 jsAPI: changeTab()

Переключает активный таб в регионе с типом «Tabs».

#### Синтаксис

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

### Параметры

- id <sup>String</sup> - id региона с типом «Tabs»

- `tab` <sup>Integer</sup> - Порядковый номер вкладки (начиная с 1)

**Пример**

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

**7.10.12 jsAPI: component()**

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

**Синтаксис**

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

**Параметры**

- `id` <sup>String</sup> - id региона

**Методы****refresh()**

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

**Пример**

```
jsAPI.component("REPORT").refresh();
```

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"
    }
  }
}
```

### 7.10.13 jsAPI: setItem()

Изменяет значение элемента формы

#### Синтаксис

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

#### Параметры

- id <sup>String</sup> - id элемента формы. НЕ является селектором (указывать без '#').
- value <sup>String</sup> - Значение

#### Примеры

##### Присвоить новое значение

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

##### Очистить

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

### 7.10.14 jsAPI: getItem()

Возвращает значение элемента формы

#### Синтаксис

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

#### Параметры

- id <sup>String</sup> - id элемента формы. НЕ является селектором (указывать без '#')

#### Пример

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

### 7.10.15 jsAPI: updateItem()

Обновляет параметры элемента формы на уровне страницы ({API\_URL}/showPage)

#### Синтаксис

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

#### Параметры

- id <sup>String</sup> - id элемента формы. НЕ является селектором (указывать без '#')
- parameters <sup>Object</sup>

Содержит параметры:

- label <sup>String</sup> - Плейсхолдер (лейбл элемента)
- required <sup>Boolean</sup> - Обязательность заполнения
- col <sup>Integer</sup> - Ширина колонки в форме (от 1 до 12)
- mask <sup>String</sup> - Маска поля (работает только с TEXT)

- `maxlength` <sup>Integer</sup> - Макс. кол-во символов (работает только с TEXT)

**Пример**

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

**7.10.16 jsAPI: hideItem()**

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

**Синтаксис**

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

**Параметры**

- `id` <sup>String</sup> - id элемента формы. НЕ является селектором (указывать без '#')

**Пример**

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

**7.10.17 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` <sup>String</sup> - id элемента формы. НЕ является селектором (указывать без '#')

**Пример**

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

**7.10.18 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 <sup>Object</sup> Данные для отправки в теле запроса
- callbacks <sup>Object</sup>

Принимает функции: onSuccess <sup>Function</sup>, onError <sup>Function</sup>.

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

```
data: {
  {
    id: "P1000_SELECT";
  }
}
```

## Примеры

### Стандартный вызов

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

### Стандартный вызов с обработчиками:

```
jsAPI.refreshList (
  {
    id: "P1000_SELECT",
    items: {
      P1000_PARAM: "New P1000_PARAM val"
    }
  },
  {
    onSuccess: function (data) {
      console.log(data);
    },
    onError: function (err) {
      console.error(err);
    }
  }
);
```

## 7.10.19 jsAPI: dom.hide()

Скрывает элемент с указанным CSS селектором путём добавления CSS класса js-api-hidden

### Синтаксис

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

#### Параметры

- selector <sup>String</sup> - Селектор элемента

#### Примеры

##### Example 1:

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

### 7.10.20 jsAPI: dom.show()

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

#### Синтаксис

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

#### Параметры

- selector <sup>String</sup> - Селектор элемента

#### Примеры

##### Example 1:

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

### 7.10.21 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 <sup>String</sup> - Селектор элемента

#### Примеры

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

### 7.10.22 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();
```

### 7.10.23 jsAPI: dom.setAttribute()

Добавляет и изменяет атрибуты у элемента с указанным селектором

#### Синтаксис

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

#### Параметры

- selector <sup>String</sup> - Селектор элемента
- attrs <sup>Object</sup> - Атрибуты

#### Примеры

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

### 7.10.24 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 <sup>String</sup> - Имя уровня отладки.

#### Доступные уровни отладки

- CRITICAL
- ERROR
- WARNING
- INFO
- DEBUG

#### Примеры

##### Активировать отладку с типом CRITICAL:

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

##### Деактивировать отладку с типом CRITICAL:

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

### 7.10.25 jsAPI: logout()

Совершает выход из приложения (логаут), вызывая метод {API\_URL}/logout

#### Синтаксис

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

#### Параметры

Данная функция не принимает аргументов.

#### Пример

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

### 7.10.26 jsAPI: clearLocalStorage()

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

#### Синтаксис

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

#### Параметры

Данная функция не принимает аргументов.

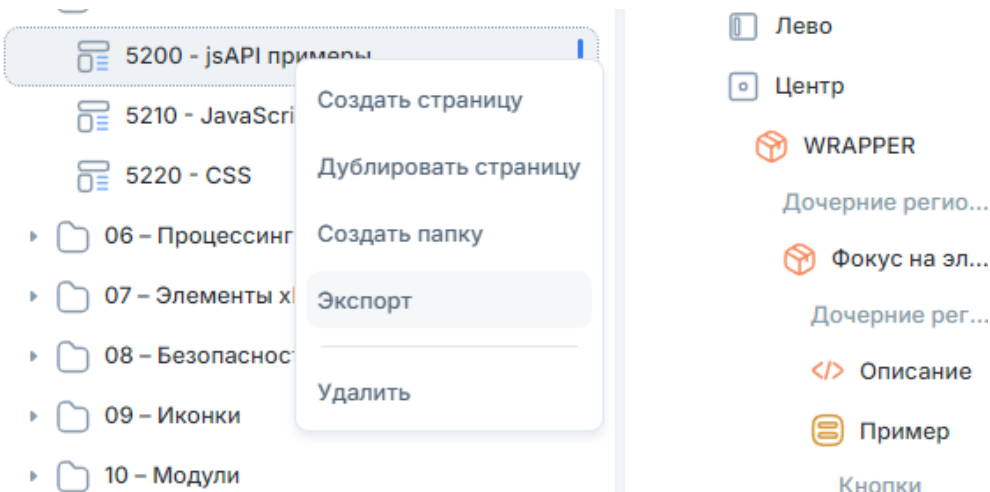
#### Пример

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

## Экспорт/импорт

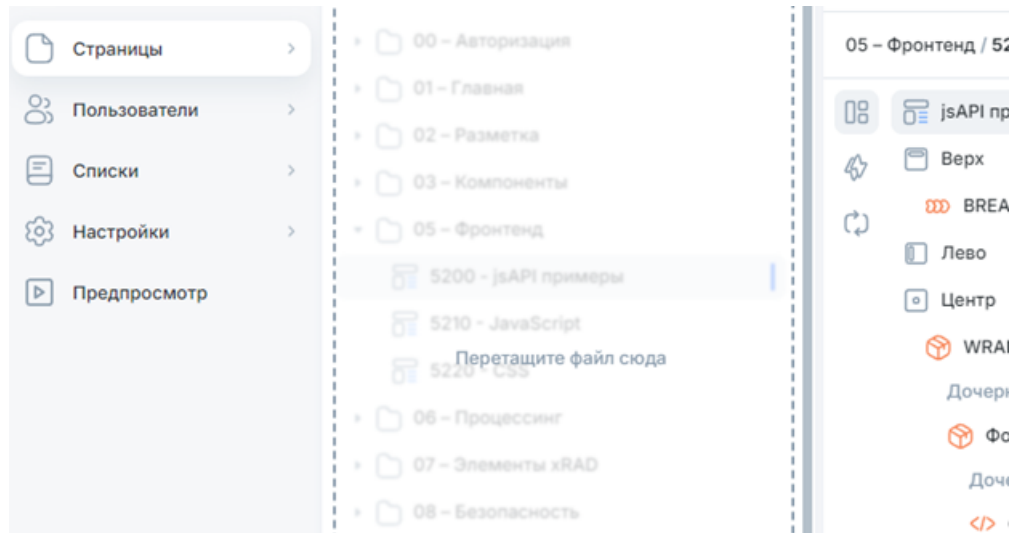
XRAD позволяет выполнить экспорт и импорт страницы или целого приложения. Данный функционал полезен, когда необходимо перенести проект на другой стенд или использовать наработки одного из проектов в другом.

Экспорт страницы доступен в контекстном меню страницы в списке страниц (ПКМ - Экспорт). Система сформирует скрипт с расширением .sql для создания страницы.



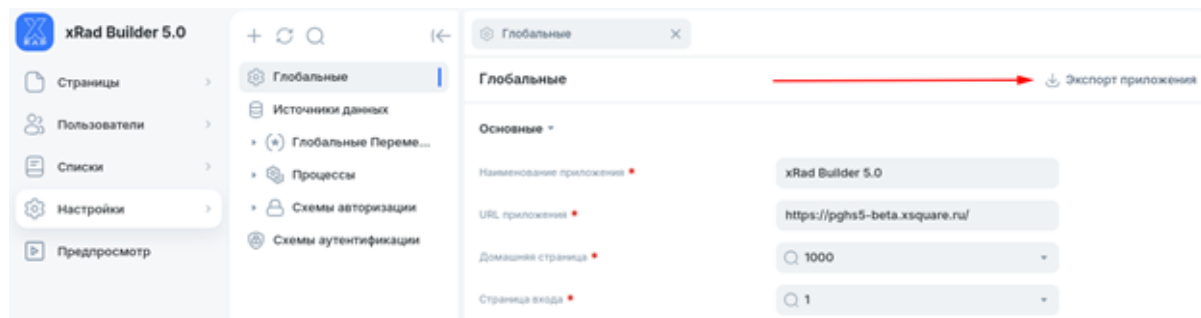


Импорт страницы осуществляется путём перетаскивания файла скрипата .sql в область со списком страниц.

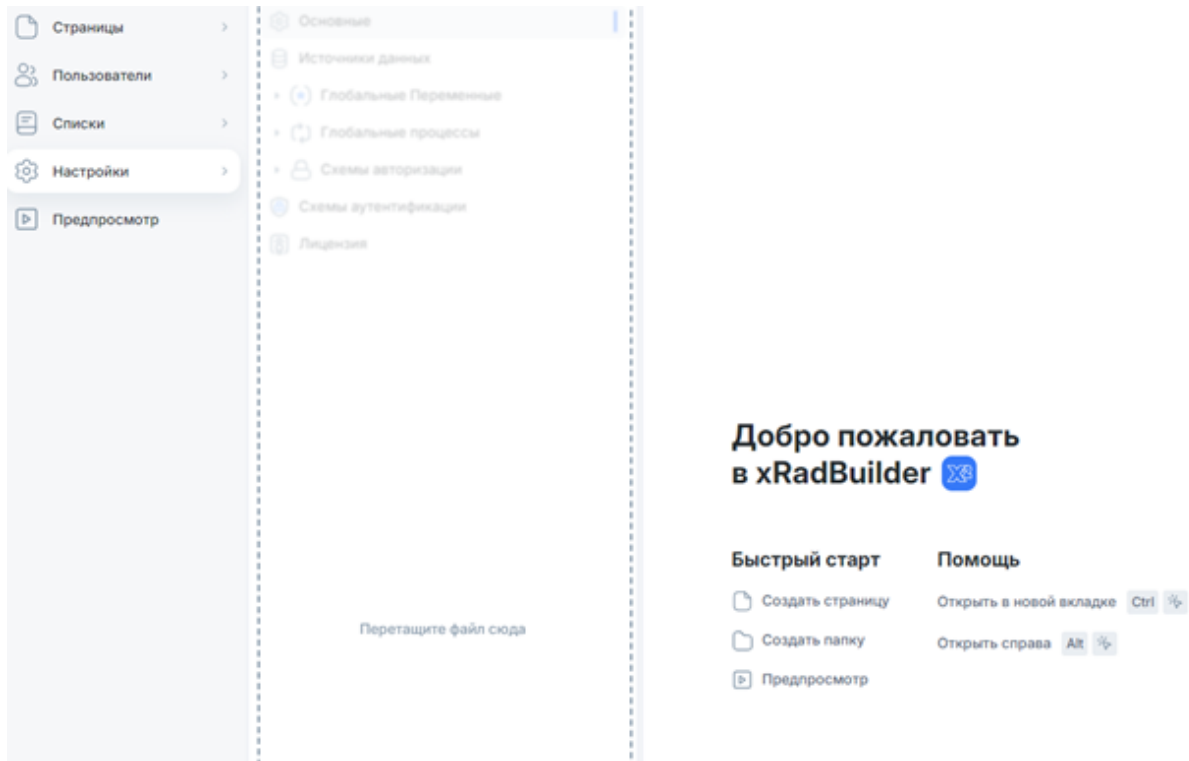


**Внимание!** Скрипт содержит вызов процедуры, которая удаляет страницу перед импортом. Также скрипт содержит все идентификаторы и ссылки внутри компонентов в том виде, в котором они находятся в базе.

Экспорт приложения доступен в разделе Настройки – Глобальные. При этом система сформирует скрипт с расширением .sql для импорта всего приложения.



Импорт приложения осуществляется путём перетаскивания файла .sql в область списка настроек приложения.



**Внимание!** При импорте скрипт полностью перезапишет приложения на целевом стенде. Скрипт выполняется в одну транзакцию и в случае сбоя целевое приложение останется в исходном виде.

### 7.10.27 jsAPI: setLoading()

Устанавливает состояние загрузки для текущей страницы или модального окна

#### Синтаксис

```
jsAPI.setLoading(value: boolean) : void
```

#### Параметры

*value: boolean* - **true** - отобразить состояние загрузки. **false** - скрыть состояние загрузки.

#### Пример

```
jsAPI.setLoading(true);
jsAPI.setLoading(false);
```

### 7.10.28 Свойство loading

Установить состояние загрузки для кнопки/региона.

Поддерживаемые регионы:

- FORM;
- WRAPPER;
- HTML\_TEXT;
- TABS;

- TREE;
- CARDS;
- TILES;
- REPORT;
- DATA\_GRID;
- CHAT;
- CHART;
- CALENDAR.

### Синтаксис

```
jsAPI.component("REGION_ID").properties({ loading: boolean }) : void
```

### Параметры

*loading: boolean* - **true** - отобразить состояние загрузки. **false** - скрыть состояние загрузки.

### Пример

```
jsAPI.component("FORM_SAMPLE").properties({ loading: true })
jsAPI.component("FORM_SAMPLE").properties({ loading: false })
```

## 7.10.29 jsAPI: download()

Скачать файл через JavaScript, без перенаправления или обновления страницы.

Этот метод позволяет отслеживать прогресс скачивания файла и обрабатывать ошибки возникшие во время генерации.

Вы можете заблокировать кнопку генерации файла или вывести информацию о прогрессе пользователю.

Это может быть особенно полезно если генерация файла занимает длительное время.

### Синтаксис

```
jsAPI.download({
  process?: string
  url?: string
  method?: "GET" | "POST"
  data?: Record<string, any> | FormData
  headers?: Record<string, any>
  onProgress?: (progress: ProgressEvent) => void
  onSuccess?: (result: {
    response: any
  }) => void
  onError?: (error: { title: string; message: string, code: number }, raw: any) =>
  void
}) : void
```

### Параметры

*process?: string* - Имя процесса, который необходимо вызвать для скачивания файла;

*url?: string* - URL для скачивания файла (файл или метод API);

*method?: «GET» | «POST»* - метод запроса. Поддерживаемые значения *GET* | *POST*;

*data?: Record<string, any> | FormData* - данные, которые необходимо отправить. Для метода POST реализована поддержка формата данных *FormData*;

*headers?: Record<string, any>* - заголовки запроса;

*onProgress?: (progress: ProgressEvent) => void* - Коллбэк, вызываемый при обновлении прогресса скачивания файла. Вы можете использовать его для отображения состояния загрузки. Коллбэк принимает аргумент типа `[ ProgressEvent` (документация: <https://developer.mozilla.org/en-US/docs/Web/API/ProgressEvent>)

```
onSuccess?: (result: {
  success: boolean
  response: any
}) => void
```

Коллбэк, вызываемый когда скачивание файла завершено. Принимает объект - *result*, который содержит: - *response: any* - ответ от сервера;

*onError?: (error: { title: string; message: string; code: number }, raw: any) => void* - Коллбэк, вызываемый когда во время скачивания файла произошла ошибка: - *error.title* - краткий текст ошибки; - *error.message* - сообщение об ошибке; - *error.code* - код ошибки.<br>

#### Значения:

- -2 - не удалось декодировать ответ от сервера;
- 0 - Ошибка интернет-соединения;
- 4xx,5xx - сервер вернул ошибку.

#### Примеры

Скачать файл сгенерированный процессом *fooProcess* с данными из полей *USERNAME* и *AGE*

```
jsAPI.download({
  process: `fooProcess`,
  data: {
    username: jsAPI.getItem('USERNAME'),
    age: jsAPI.getItem('AGE'),
  }
})
```

Скачать файл `/test.txt` и отобразить прогресс

```
jsAPI.download({
  url: "/test.txt",
  method: "GET",
  onSuccess() {
    jsAPI.notification.success("Скачивание завершено!");
  },
  onError(error) {
    console.error(error)
    jsAPI.notification.error("Ошибка #" + error.code + " <br>" + error.message);
  },
  onProgress(progress) {
    let percent = progress.loaded / progress.total * 100;
    jsAPI.notification.warning("Скачено " + percent + "%");
  }
});
```

Отправить данные методом POST в объекте *FormData* на адрес */api/download\_file* с заголовком *token* и заблокировать кнопку на время скачивания:

```
let payload = new FormData();
payload.append('username', jsAPI.getItem('USERNAME'));
payload.append('age', jsAPI.getItem('AGE'));

jsAPI.component("BUTTON").properties({ loading: true }); // Блокируем кнопку

jsAPI.download({
  url: "/api/download_file",
  method: "POST",
  data: payload,
  headers: { token: '@JKnfsdlk@E$@!BNKOGDSbjO#$BKJO' },
  onSuccess() {
    jsAPI.component("BUTTON").properties({ loading: false }); // Разблокируем кнопку
    jsAPI.notification.success("Скачивание завершено!");
  },
  onError(error) {
    jsAPI.component("BUTTON").properties({ loading: false }); // Разблокируем кнопку
    console.error(error)
    jsAPI.notification.error("Ошибка #" + error.code + " <br>" + error.message);
  },
  onProgress(progress) {
    let percent = progress.loaded / progress.total * 100;
    jsAPI.notification.warning("Скачано " + percent + "%");
  }
});
```