Общество с ограниченной ответственностью
“ГЕНОМ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ”
ОГРН 1227700857548
ИНН 7714499420/КПП 773101001
Адрес: 121205, Город Москва, вн.тер. г. Муниципальный Округ Можайский, тер Инновационного Центра Сколково, б-р Большой, дом 42, строение 1

Системный Геном

Описание функциональных характеристик
Руководство пользователя и администратора

Версия Системного Генома	1.0.0
Версия инструкции	от 19.12.2023

Оглавление

Введение

Цели документа

1. **Руководство по установке и использованию** – содержит пошаговые инструкции по установке и использованию программного обеспечения.
2. **Информация об особенностях и функциях** – содержит подробное описание функциональных характеристик.
3. **Помощь в оценке и принятии решений** – для потенциальных пользователей содержит исчерпывающую информацию, помогающую оценить пригодность программного обеспечения для их конкретных потребностей и задач.

Целевая аудитория

1. **Конечные пользователи** – физические или юридические лица, которые будут напрямую взаимодействовать с внедрёнными прикладными программными продуктами (пользователи с разным уровнем технических знаний, от новичков до продвинутых).
2. **ИТ-специалисты** – системные администраторы, разработчики и сотрудники ИТ-поддержки, которые отвечают за установку, настройку, использование и обслуживание программного обеспечения в организации.
3. **Лица, принимающие бизнес-решения** – менеджеры и руководители, рассматривающие возможность внедрения программного обеспечения в свои бизнес-процессы. Им требуется понимание возможностей программного обеспечения и потенциального влияния на их работу.

Краткий обзор Системного Генома

Системный Геном – low-code платформа, предназначенная для создания корпоративных и государственных информационных систем (прежде всего, класса ERP). В основе платформы – среда визуального программирования, основанная на онтологическом графе.

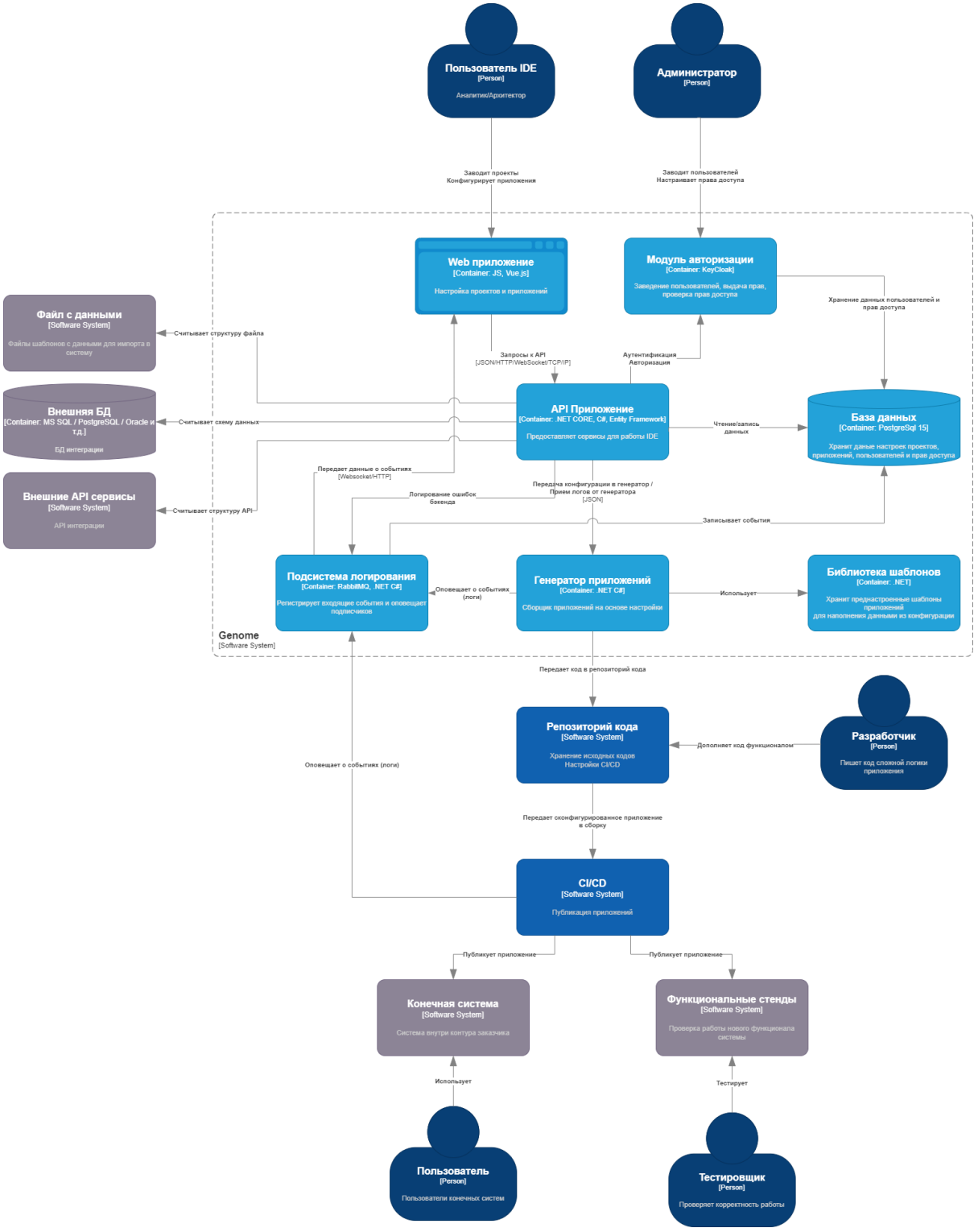
Основная концепция

Основной элемент платформы «Системный Геном» - среда визуального программирования, представленная в виде онтологического графа. Графическое представление структур данных и элементов информационных систем в одном окне позволяет системным аналитикам и архитекторам самостоятельно разрабатывать большую часть функциональности сложных информационных без написания кода. За программистами остается дополнение кодом тел сложных методов. (подробнее)

Стратегия интеграции

Внедрение платформы предполагает «лоскутный» подход к интеграции, решающий общие проблемы в крупных организациях, такие как изолированные системы, дублирующиеся функциональные возможности и разрозненные рабочие процессы. Эта стратегия предполагает создание небольших веб-приложений («лоскутков») для автоматизации определенных сегментов процессов и их интеграцию с основными системами. (подробнее)

Архитектура Системного Генома



Архитектура Системного Генома в нотации C4

L1 Контекстная диаграмма системы

Внешние системы

1. **Файл с данными** - файлы шаблонов с данными для импорта в систему
2. **Внешняя БД** - БД интеграции MS SQL / PostgreSQL / Oracle и т.д.
3. **Внешние API сервисы** - интеграции с внешними системами заказчика
4. **Конечная система** - система внутри контура заказчика, на которую будет разворачиваться продуктовые версии разрабатываемых приложений
5. **Функциональные стенды** - проверка работы нового функционала системы

Внутренние системы

1. **Конструктор приложений** – ядро Системного Генома, среда визуального программирования для проектирования и разработки прикладных приложений
2. **Репозиторий кода** – хранилище исходных кодов генерируемых приложений
3. **CI/CD** – система автоматического развёртывания конечных приложений, разработанных и сгенерированных в Конструкторе приложений

L2 Схема контейнера

Конструктор приложений

1. **Web-приложение** – создание и настройка проектов и приложений
2. **Модуль авторизации** – заведение пользователей и ролей, выдача прав, аутентификация и авторизация (проверка прав доступа)
3. **API Приложение** – предоставляет сервисы для работы IDE
4. **База данных** – хранит данные настроек проектов, приложений, пользователей и прав доступа
5. **Подсистема логирования** – регистрирует входящие события и оповещает подписчиков
6. **Генератор приложений** – выполняет проверку настроек и исходных кодов приложений и генерацию кодов
7. **Библиотека шаблонов** – хранит шаблоны приложений для наполнения данными из конфигурации

Пользователи платформы и конечных приложений

Заказчик

Описание: физическое или юридическое лицо, инициирующее проект разработки программного обеспечения для удовлетворения нужд и требований конечных пользователей.
Задачи: формулирование общих требований и ожиданий от программного обеспечения. Предоставление обратной связи и валидации во время и после процесса разработки.

Системный аналитик

Описание: специалист, ответственный за устранение разрыва между требованиями заказчика и технической реализацией.
Задачи: анализ и документирование требований заказчика, создание структуры приложения (включая структуру базы данных, серверной части и внешнего интерфейса) и выполнение задач разработки для упрощения логики и вычислений с использованием платформы.

Архитектор программного обеспечения

Описание: специалист старшего уровня, ответственный за принятие архитектурных решений и высокоуровневое проектирование, определение технических стандартов, инструментов и платформ.
Задачи: проверка структуры приложения, разработанной аналитиком, на предмет ее соответствия техническим стандартам и целям проекта; предоставление подробных требований для разработки методов классов и объектов.

Разработчик (SQL/C#/JS)

Описание: программист, специализирующийся на определенных языках, таких как SQL, C# или JavaScript.
Задачи: выполнение сложных задач разработки (включая разработку исходного кода), требующих специальных навыков программирования, особенно в областях, выходящих за рамки возможностей системного аналитика.

Тестировщик

Описание: специалист, ответственный за тестирование и проверку программного обеспечения на соответствие требованиям.
Задачи: совместная работа с аналитиком над тестированием функциональности Web-приложения, выявлением и документированием дефектов, а также проверкой конечного продукта на соответствие требованиям.

Этапы жизненного цикла ПО с Системным Геномом

Процесс разработки на платформе “Системный Геном” включает следующие этапы:

1. **Формулирование требований**: заказчик формулирует общие требования к проекту программного обеспечения.
2. **Анализ требований и документация**: системный аналитик документирует эти требования, определяя масштаб и основные потребности проекта.
3. **Разработка структуры приложения**: используя платформу “Системный Геном”, аналитик создает структуру приложения, включая детализацию базы данных, серверной части, внешнего интерфейса и их взаимосвязей.
4. **Проверка структуры**: архитектор проводит тщательную проверку среды приложения, чтобы убедиться, что она соответствует изложенным требованиям и стандартам.
5. **Детализация требований к разработке**: архитектор и аналитик формируют подробные требования к разработке методов классов и объектов (в форме аннотаций или комментариев внутри кода).
6. **Выполнение разработки**: в зависимости от сложности требований задачи разработки выполняются либо аналитиком, либо специализированными разработчиками (для более сложной логики и вычислений) на SQL, C# или JS. Этому способствует встроенный редактор методов платформы.
7. **Сборка и публикация веб-страницы**. Готовая веб-страница затем компилируется и развертывается путем выбора соответствующего сервера.
8. **Тестирование и проверка**. На заключительном этапе тестировщик и аналитик совместно проверяют функциональность и правильность веб-приложения. В случае возникновения каких-либо проблем процесс может вернуться к этапу выполнения разработки для внесения необходимых корректировок.

Эти этапы представляют собой комплексный и итеративный подход, гарантирующий, что каждый этап разработки тесно согласуется с требованиями заказчика и общими целями проекта.

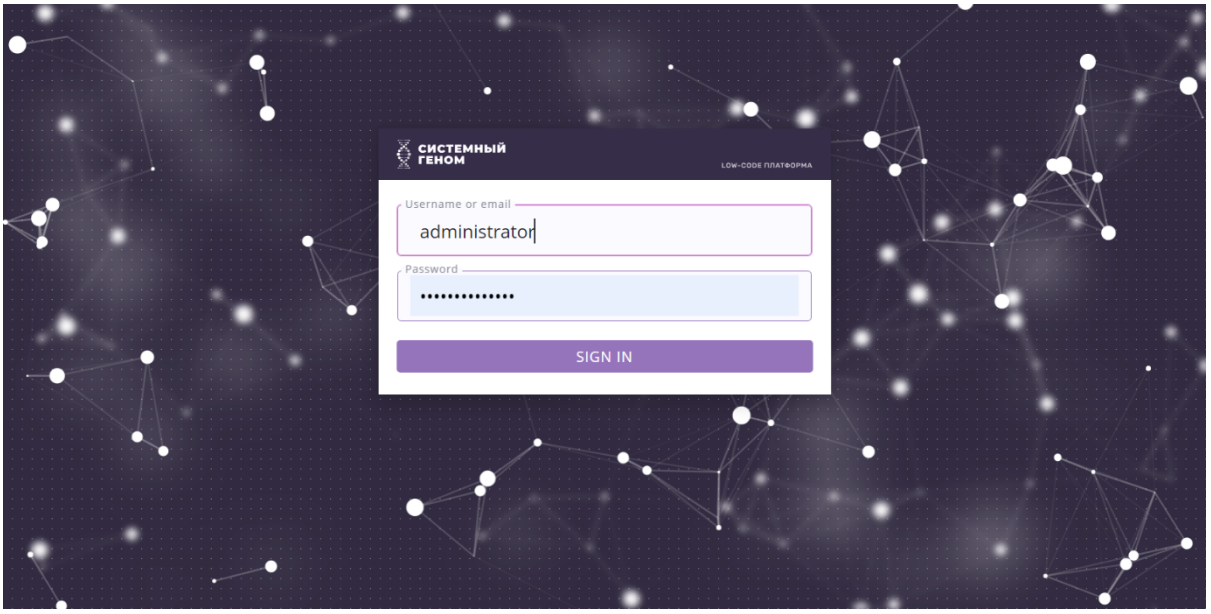
Установка

Пошаговая инструкция по установке экземпляра ПО приведена в документе «Инструкция по установке экземпляра ПО».

Пользовательский интерфейс IDE

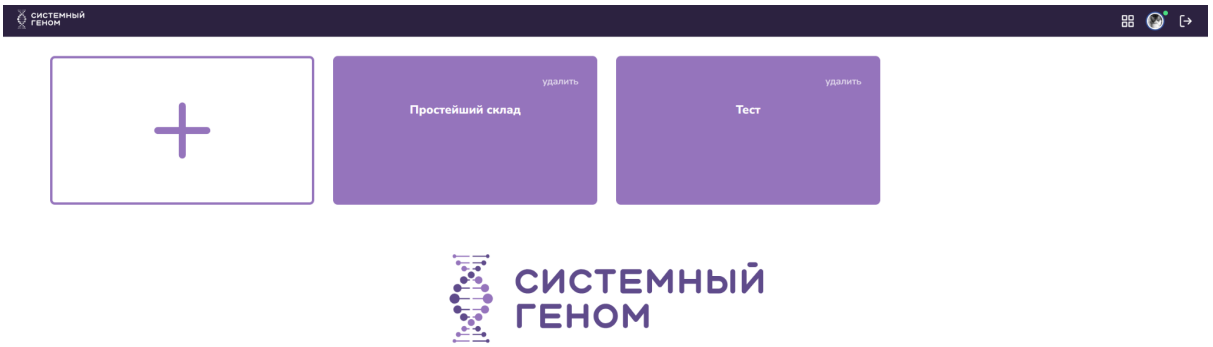
Авторизация

Вводим логин, пароль и нажимаем “Sign In” для авторизации в Системном Геноме.



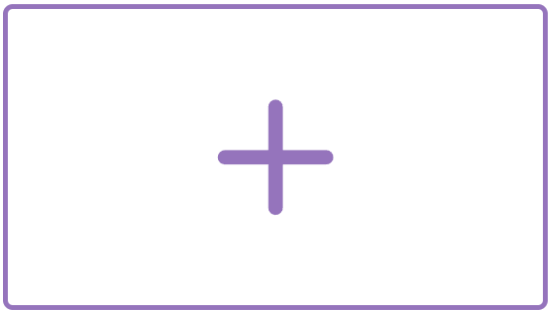
Список проектов

Здесь представлены все проекты, к которым есть доступ у пользователя. Открыть проект можно, просто кликнув по нему. Также можно удалить проект (кнопка в правом верхнем углу карточки проекта), если у пользователя на это есть разрешение.

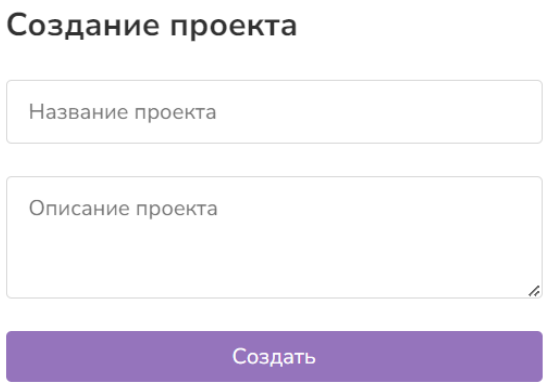


Создание проекта

1) Можно создать проект, кликнув по карточке с большим символом “+”.



2) Откроется модальное окно создания проекта



3) Указываем название и описание проекта и кликаем кнопку «Создать».

Проект

Это основная рабочая область IDE

Верхняя панель (шапка)

Содержит (слева направо):
– логотип «СИСТЕМНЫЙ ГЕНОМ», нажатие на который возвращает к списку проектов;
– название открытого проекта (двойной клик по нему позволяет его переименовать);
– техническую информацию о выбранном узле (если узел не выбран – пусто);
– кнопку возврата к списку проектов ;
– кнопку выхода из системы (завершения сессии пользователя) .

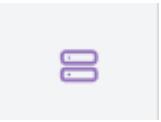
Панель инструментов

Панель для выполнения глобальных операций над проектом или рабочим пространством

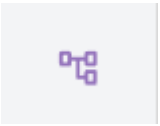
- создание приложения

- открытие модального окна “Сборки и релизы приложения”

- удалить выделенный узел из дерева



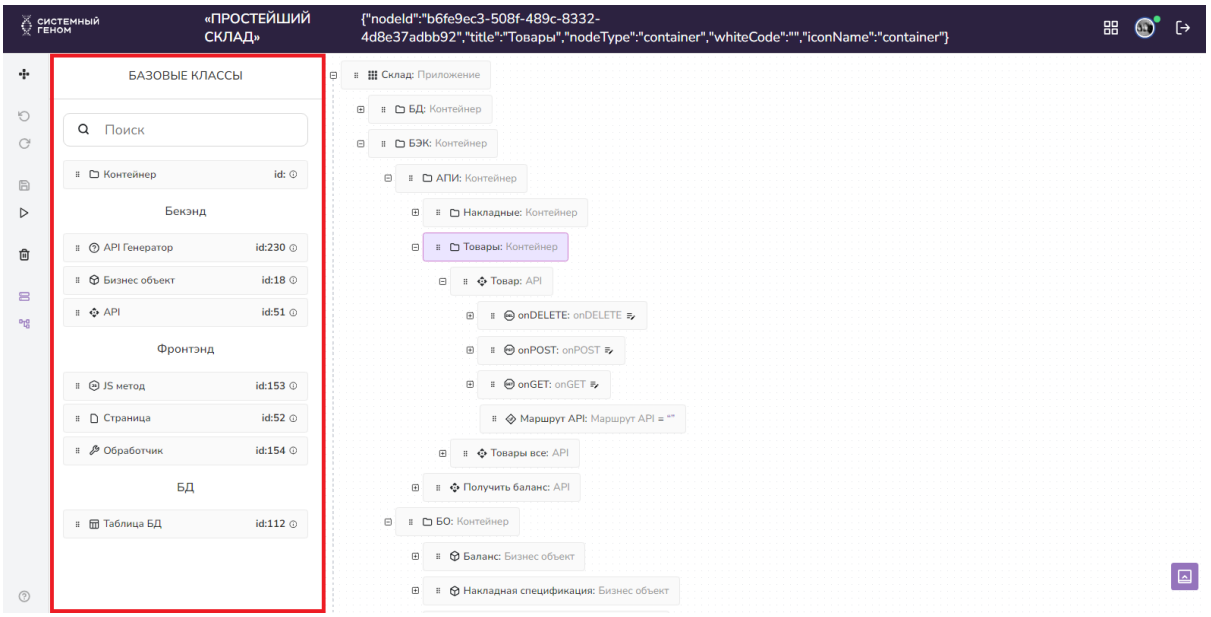
- скрыть/отобразить окно базовых классов



- скрыть/отобразить окно дерева классов и объектов

Панель Базовых классов

Эта панель служит центральным хабом для добавления узлов в дерево классов и объектов. Она динамически отображает коллекцию узлов, которые можно создать, и которая варьируется в зависимости от выбранного в данный момент узла в дереве.



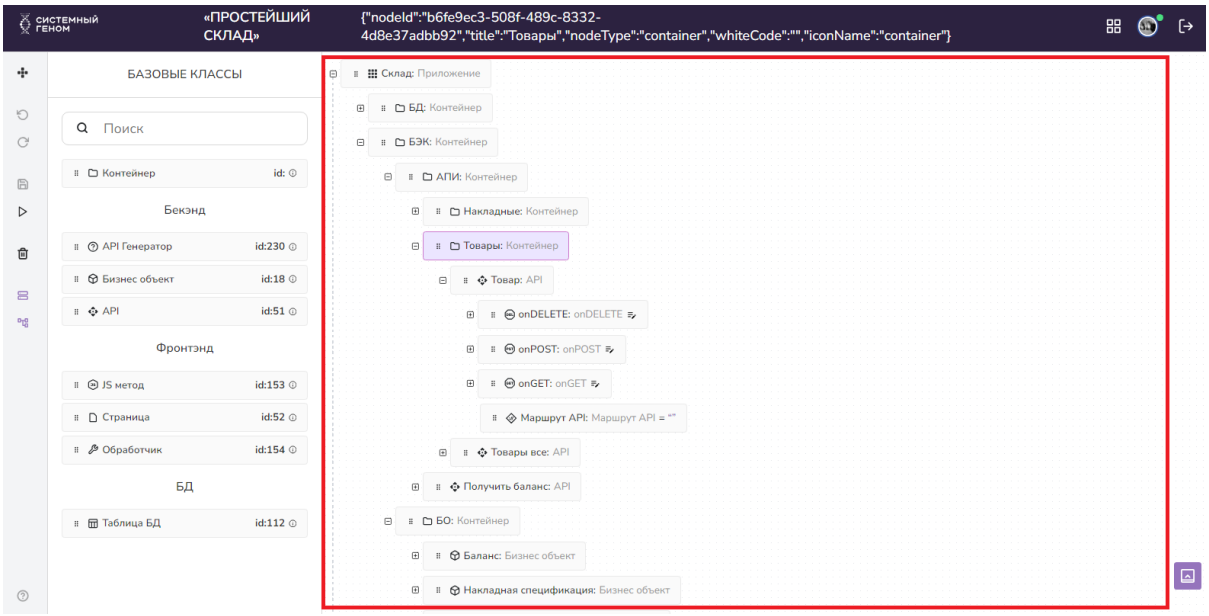
Панель Базовых классов

Ключевые особенности:

- **Динамическое отображение узлов**: панель настраивает отображаемые узлы в зависимости от выбора в дереве классов и объектов. Если ещё ни один узел не создан или не выбран, панель содержит ссылку для создания приложения.
- **Создание узла, в корне приложения**. После создания и выбора приложения в дереве на панели отображается ряд узлов, которые можно создать в корневом уровне приложения.
- **Функциональность перетаскивания**: экземпляры базовых классов можно легко добавить в дерево приложение с помощью механизма перетаскивания. Во время процесса перетаскивания система отображает ячейки-заполнители, подсвечивая узлы, в которые можно добавить выбранный базовый класс.
- **Контекстные предложения узлов**: после создания и выбора узла панель обновляется, предлагая потенциальные дочерние узлы, адаптируя свои предложения к конкретному выбранному родительскому узлу.
    - Для получения подробной информации о возможных родительско-дочерних отношениях в дереве можно обратиться к разделу, содержащему полный список базовых классов с описанием.

Дерево Классов и объектов

Дерево классов и объектов является краеугольным компонентом, занимающим центральное место в архитектуре системы и разработке приложений. Оно воплощает суть концепции онтологии, представляя ее в виде иерархического дерева (технически являющегося графом), узлы которого содержат разные элементы.



Дерево классов и объектов

Ключевые особенности:

- **Иерархическая структура**. По своей сути дерево образует иерархическую структуру, в которой каждый узел представляет определенный тип. Эти типы могут включать приложения, объекты, классы или ссылки, каждый из которых играет определенную роль в структуре дерева.
- **Свертываемые узлы**: узлы в дереве могут сворачиваться и разворачиваться нажатием на значок плюса или минуса слева от названия узла.
- **Доступ к контекстному меню**. Дополнительный уровень интерактивности предлагается через контекстное меню для каждого узла; содержимое меню (пункты и операции) зависит от типа и содержимого узла.
- **Типология узлов и взаимосвязь**. Кроме иерархии узлы, классифицированные как ссылки, могут содержать ссылки на другие узлы внутри дерева.
- **Ссылки зависят от базового класса**. Существуют системные ограничения значений ссылок узлов в зависимости от базового класса. Это обеспечивает контролируемую и логичную структуру дерева.
- **Значения для классов и объектов**. Узлы, идентифицированные как классы или объекты, могут иметь дополнительные поля для значений. Тип данных этих полей зависит как от его базового класса, так и от его родительского узла. Эта двойная зависимость адаптирует тип поля в соответствии с контекстом узла. Например, дочерний узел (поле) «значение по умолчанию» для целочисленного узла-родителя будет иметь типом «Целое число», а для строкового узла-родителя будет иметь тип «Строка».
- **Установка значений для узлов ссылок**. Присвоение значений для узлов ссылочного типа выполняется посредством простого перетаскивания (узла, на

который пользователь хочет сделать ссылку) или вставки ранее скопированного значения ссылки с помощью контекстного меню.

Консоль

Это удобная система журналирования, интегрированная в IDE, предназначенная для комплексного и эффективного ведения журналов действий и сообщений.

Ключевые особенности:

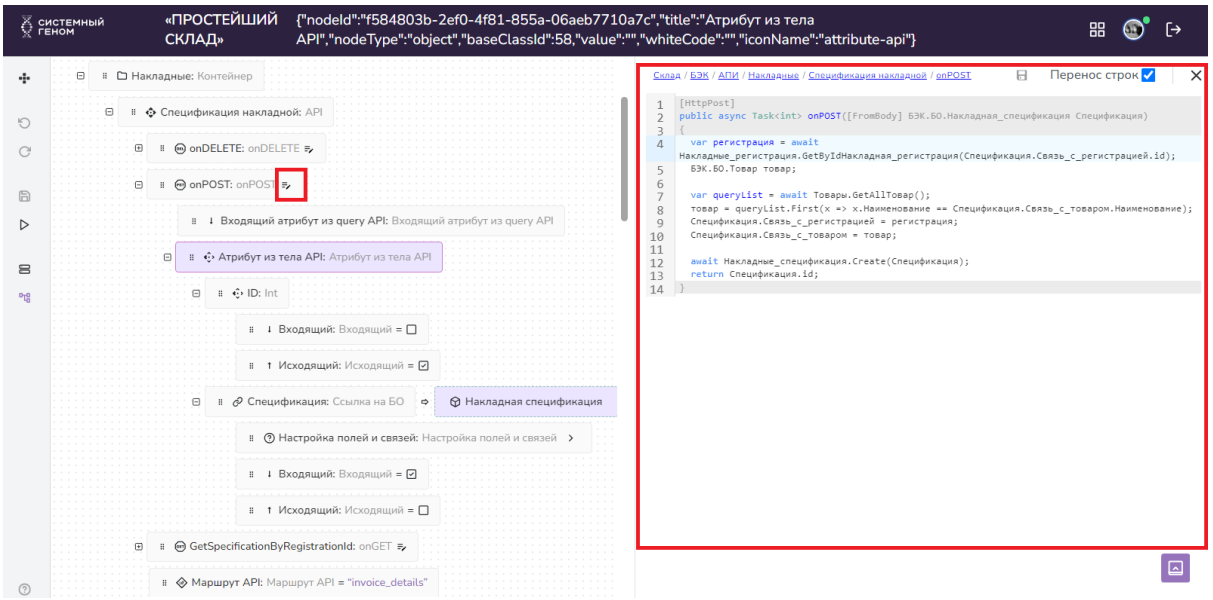
- Состав панели консоли. Панель консоли имеет информативную структуру и отображает различные важные детали:
    - Уровень сообщения: каждая запись журнала классифицируется по своему типу: информационная, предупреждающая или об ошибке.
    - Временная метка: все записи журнала имеют точную временную метку, указывающую дату и время возникновения сообщения.
    - Идентификация подсистемы: включена информация о конкретной подсистеме, к которой относятся записи журнала.
    - Сообщение журнала: основное содержимое сообщений, отображается в каждой строчке (записи).
    - Расширяемые сведения: пользователи могут просмотреть более подробную информацию в каждом сообщении журнала, щелкнув значок стрелки рядом с каждой записью.
- Функциональность поиска: консоль оснащена инструментом поиска, позволяющим пользователям эффективно находить определенные журналы.
- Расширенные параметры фильтрации. Чтобы упростить процесс просмотра журналов, консоль предлагает:
    - Фильтрация на основе компонентов: пользователи могут фильтровать журналы на основе их исходного компонента.
    - Фильтрация по типу: для более целенаправленного изучения доступна фильтрация по типу записей (информация, предупреждение, ошибка).
    - Очистка консоли. Для удобства использования в консоли предусмотрена функция очистки текущего журнала.
    - Возможность экспорта. Пользователи могут экспортировать содержимое консоли в форматы JSON или CSV, что облегчает анализ данных и составление отчетов.
- Гибкость пользовательского интерфейса. Консоль разработана с учетом удобства пользователя:
    - Переключить видимость: консоль можно легко открыть или закрыть по мере необходимости.
    - Специальные возможности: для быстрого доступа к консоли доступна плавающая кнопка, а закрытие консоли можно просто выполнить с помощью значка крестика в интерфейсе консоли.

Консоль System Genome представляет собой сочетание возможностей подробного журналирования с ориентированным на пользователя дизайном, что делает ее бесценным инструментом для разработчиков, работающих в среде IDE. Сочетание

подробной информации журнала, функций поиска и фильтрации, а также гибкости пользовательского интерфейса делает его важным компонентом для эффективного мониторинга и отладки системы.

Редактор кода

Доступ к редактору кода можно получить, нажав кнопку  справа от названия узла.



Редактор кода

Понимание отображения кода:

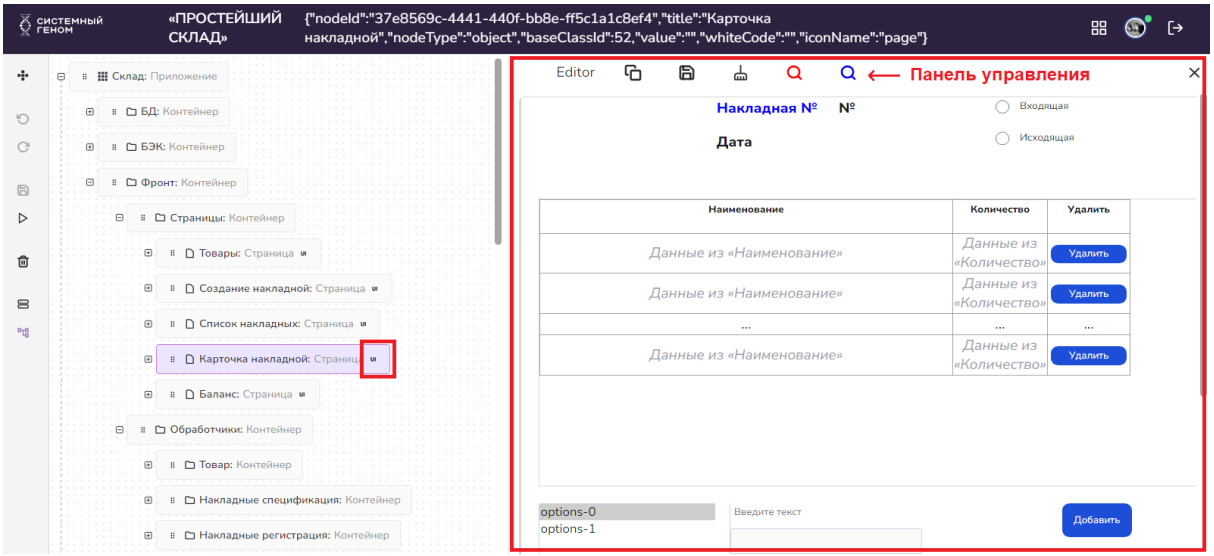
- **Серый код**: автоматически генерируется системой на основе узла и его дочерних узлов и недоступен для ручного редактирования пользователем. Обычно он включает в себя такие элементы, как тип метода, типы и названия входящих и исходящих параметров и др.
- **Белый код**: это редактируемая область, где пользователи могут писать или изменять код. Может содержать сгенерированный текст.

Функциональность:

- **Подсветка синтаксиса**. Редактор предлагает подсветку синтаксиса для всех языков программирования, поддерживаемых в системе (C#, JavaScript и SQL).
- **Функция автозаполнения**: интегрирована интеллектуальная система автозаполнения, предоставляющая предложения по написанию кода на основе древовидной структуры онтологии.
- **Регулировка методов, API**. Редактор автоматически пишет код методов и методов API в соответствии с их входящими и исходящими параметрами / атрибутами, указанными в дереве.
- **Интеграция с кодом проекта**. Любой код, разработанный в редакторе узла, будет автоматически включен в исходный код приложения при генерации.

Редактор интерфейса

Редактор интерфейса, необходимый для создания пользовательских интерфейсов в клиентских приложениях, открывается при нажатии на кнопку **UI** на поддерживаемых узлах, таких как страницы или портлеты.



Редактор интерфейса

1. **Работа со страницами и портлетами**:
    a. **Страницы**: отображают содержимое всех портлетов внутри них. Пользователи могут настраивать расположение и макет страницы и каждого портлета в отдельности.
    b. **Портлеты**: позволяют группировать компоненты и настраивать их размещение для создания удобных пользовательских интерфейсов.
2. **Возможности редактора**:
    a. **Панель управления**: Расположена вверху и включает в себя несколько функциональных кнопок:
        i. **Режим редактирования**: позволяет редактировать макет выбранного узла (страницы или портлета), а также некоторые свойства компонентов (размеры, текст).
        ii. **Кнопка «Сохранить»**: сохраняет изменения, внесенные в режиме редактирования. Он также используется для сохранения прямого ввода текста в компоненты вне режима редактирования.
        iii. **Кнопка «Отмена»**: отменяет изменения, внесенные с момента последнего сохранения.
        iv. **Увеличение/уменьшение масштаба**: настраивает вид редактора для лучшей видимости.
3. **Реактивность и обновления в реальном времени**:
    a. Изменения, внесенные в редакторе или в дереве классов и объектов, мгновенно отражаются в обеих областях. Такая реактивность гарантирует синхронизацию обновлений в режиме реального времени.
    b. Стоит отметить, что изменения, внесенные напрямую в дереве, сразу будут сохранены независимо от сохранения в редакторе интерфейсов.

Структура онтологии

Системный Геном использует структуру, основанную на онтологиях, предлагая уникальный, эффективный и универсальный подход к проектированию и разработке информационных систем.

Проект

В рамках экземпляра платформы проект представляет собой отдельную БД (либо кластер), и, как правило, приравнивается к организации. При необходимости масштабирования, крупная организация может быть представлена в нескольких проектах.

Приложение

Проекты содержат приложения. Приложение соответствует функциональному модулю и представлено в виде отдельных бэкенда и фронтенда на веб-сервере. При необходимости масштабирования, веб-сервер может быть заведен за балансировщик и размножен. Приложения также могут быть спроектированы как микросервисы.

Узлы

Узлы в онтологии могут быть нескольких типов: приложения, классы, объекты, ссылки. Есть строгий список ограничений того, кем может выступать наследник базового класса в дереве при создании.

Классы

Являются типом узла, который можно создавать в дереве, но классы не участвуют напрямую в генерации исходного кода, и необходимы для создания узлов типа объектов, называемых «узлами-наследниками».

Объекты

Объекты могут быть созданы как из базовых классов, так и из ранее созданных узлов классов в дереве. В процессе генерации эти объекты преобразуются в полноценные структуры кода.

Ссылки

Ссылки отличаются тем, что они реализуют логические связи в структуре онтологии между узлами. Например, ссылка обработчика на метод API демонстрирует эту функциональность. Код обработчика не только будет вызывать метод API в сгенерированном коде, но также на уровне дерева приложения создает себе дочерние узлы в соответствии со структурой атрибутов метода API.

Базовые классы

Название	Описание
Классы для описания Серверной части приложения	
Бизнес объект (БО)	Класс, описывающий структуру (поля и их начальные состояния) и определяющий алгоритмы (функции и методы) манипулирования объектами. Этот класс должен инкапсулировать бизнес-логику и представление данных для конкретного бизнес-объекта.
Связь многие к одному	Тип связи между бизнес-объектами, при котором несколько экземпляров одного типа объекта связаны с одним экземпляром объекта другого типа. В реляционных моделях данных крайне важно устанавливать иерархические структуры.
Связь многие ко многим	Тип связи между бизнес-объектами, при котором несколько экземпляров одного типа объекта могут относиться к нескольким экземплярам другого. Обычно это реализуется через промежуточную таблицу или объект для управления ассоциациями.
Связь один к одному	Тип связи между бизнес-объектами, при котором один экземпляр типа объекта связан ровно с одним экземпляром другого типа. Это часто используется для расширения или разделения данных одного объекта.
Строковое значение	Поле бизнес-объекта, в котором хранятся текстовые данные.
Идентификатор GUID	Поле уникального идентификатора бизнес-объекта. GUID обеспечивают глобальную уникальность, что крайне важно для распределенных систем.

Значение двойной точности	Поле бизнес-объекта для хранения чисел с плавающей запятой двойной точности, используемый для вычислений высокой точности.
Целочисленное значение	Поле бизнес-объекта для хранения целых чисел, используется для подсчета, идентификаторов и многого другого.
Дата и время	Тип поля в бизнес-объекте для хранения информации о дате и времени, что имеет решающее значение для функций отметки времени и планирования.
Логическое значение	Поле бизнес-объекта логического типа, используется для хранения признаков типа «истина/ложь» или «да/нет», флагов и двоичных вариантов выбора.
Метод CSharp	Включает описание бизнес-логики на языке программирования C#. Этот класс должен позволяет встроить фрагменты кода C# для реализации пользовательской логики.
Значение по умолчанию	Позволяет задать значение по умолчанию для поля бизнес-объекта.
Проверка на наличие значения	Тип проверки, используемый чтобы указать, что поле бизнес-объекта является обязательным (не может иметь пустое значение).
Ограничение длины строки	Устанавливает максимальную длину строкового поля для обеспечения целостности и согласованности данных.
Проверка подстроки	Устанавливает ограничение на значение строкового поля по определённому шаблону.
Входящий	Используется для настройки атрибута из тела API, определяющего способ получения данных из запросов API.

Исходящий	Настраивает атрибут для исходящих ответов API, определяя, что данные отправляются обратно клиентам.
API	Узел, где можно создавать методы API (end-point). Используется для объединения нескольких методов API с общим URL-маршрутом (например, для операций CRUD API).
Маршрут API	Общий URL-маршрут для методов API, который действует как общая часть адреса конечных точек для клиентского доступа.
onGET	Метод получения данных, обычно связанный с методом HTTP GET в RESTful API.
onPOST	Метод создания данных, обычно связанный с методом HTTP POST в RESTful API.
onPUT	Метод обновления данных, обычно соответствующий методу HTTP PUT в RESTful API.
onDELETE	Метод удаления данных, соответствующий методу HTTP DELETE в RESTful API.
Атрибут из тела API	Узел для создания контракта данных метода в дереве API. Здесь должна быть подробно описана структура данных, которыми обмениваются через API.
Входящий атрибут из query API	Узел для построения параметров запроса в дереве контракта данных API, определяющий, как данные фильтруются или запрашиваются.
Ссылка БО на таблицу	Узел, позволяющий связать бизнес-объект с таблицей базы данных, устанавливая механизмы получения и сохранения данных.
Параметры функции	Узел позволяет осуществлять параметризацию методов.

String (attribute-string)	Атрибут тела API, используемый для строковых типов данных.
String (attribute-string-input-only)	Атрибут параметра запроса API, предназначенный специально для ввода строковых данных.
Double (attribute-double)	Атрибут тела API для числовых данных двойной точности.
Double (attribute-double-input-only)	Атрибут параметра запроса API для числовых данных двойной точности.
Ссылка на БО	Атрибут тела API, который создает ссылку на бизнес-объект, обеспечивая настройку реляционных структур в данных API.
Boolean	Атрибут тела API для логических типов данных, используемый для принятия решений «истина/ложь» или «да/нет».
Ссылка на таблицу для М М	Ссылка на таблицу базы данных, используемая в отношении «многие ко многим» двух бизнес- объектов.
Main	Метод, являющийся точкой входа в консольное приложение
Список String	Атрибут тела API для обработки коллекций строк, позволяющий управлять несколькими текстовыми значениями в одном вызове API.
Список Double	Атрибут тела API для управления коллекциями чисел двойной точности, полезный для математических и статистических операций.
Список ссылок на БО	Атрибут тела API, который создает ссылку на бизнес-объекты, обеспечивая настройку реляционных структур в данных API.
Список Boolean	Атрибут тела API для обработки коллекций логических значений, полезный для пакетной обработки двоичных решений.

GUID (attribute-guid)	Атрибут тела API для обработки типов данных GUID, обеспечивающий уникальную идентификацию в системе.
GUID (attribute-guid-input-only)	Атрибут Query API для обработки типов данных GUID, обеспечивающий уникальную идентификацию в системе
Int (attribute-int)	Атрибут тела API для обработки целочисленных типов данных, необходимый для числовых операций и индексации.
Int (attribute-int-input-only)	Атрибут Query API для обработки целочисленных типов данных, необходимый для числовых операций и индексации.
Список GUID	Атрибут тела API для управления коллекциями GUID, полезный для обработки нескольких уникальных идентификаторов в одном вызове API.
Список Int	Атрибут тела API для управления коллекциями целых чисел, полезный для числовых операций и пакетной обработки.
API Генератор	Узел, позволяющий сгенерировать полнофункциональный API (CRUD + GetAll) для указанного бизнес-объекта, упрощая создание API.
Настройка полей и связей (api-attribute-relation)	Узел для настройки глубины реляций и данных на каждом уровне реляции, которые будут отправлены или приняты по API для бизнес-объекта.

Классы для описания Базы данных приложения	
Таблица БД	Таблица для хранения данных. Каждая таблица должна быть спроектирована так, чтобы хранить данные для конкретного бизнес-объекта или связанного набора данных
Столбик JSON	Столбец в таблице для хранения данных в формате JSON, позволяющий создавать гибкие иерархические структуры данных
Столбик timeStamp	Столбец в таблице для хранения данных временных меток, необходимый для отслеживания изменений и анализа исторических данных
Столбик text	Столбец в таблице для хранения текстовых данных, используемых для описаний, имен и другой строковой информации
Столбик uuid	Столбец в таблице для хранения универсальных уникальных идентификаторов (UUID), обеспечивающий глобальную уникальность в базе данных
Столбик real	Столбец в таблице для хранения действительных чисел (с плавающей запятой), используемый для точных числовых данных
Столбик int	Столбец в таблице для хранения целочисленных значений, который имеет решающее значение для счетчиков, идентификаторов и многого другого
Столбик boolean	Столбец в таблице для хранения логических значений (истина/ложь), используемых для двоичного выбора и флагов

primary	Указывает, что столбец является первичным ключом таблицы, однозначно идентифицирующий каждую запись в таблице
unique	Указывает, что на столбец таблицы наложено ограничение уникальности, гарантирующее, что все значения в этом столбце различны
nullable	Признак, являются ли NULL для столбца таблицы допустимыми или недопустимыми
foreign uuid	Столбец внешнего ключа, связанный со столбцом первичного ключа в другой таблице с использованием UUID, что имеет решающее значение для установления реляционных связей между таблицами
foreign int	Столбец внешнего ключа, связывающийся со столбцом первичного ключа в другой таблице с использованием целых чисел, что упрощает связи и ссылки между таблицами
Классы для описания Клиентской части приложения	
Страница	Центральный интерфейсный класс, представляющий Web-страницу в приложении. Каждая страница должна быть спроектирована так, чтобы обеспечить определенный пользовательский интерфейс и функциональность
Портлет	Компонент, который объединяет различные другие компоненты страницы в рамках единой бизнес-логики. Этот модульный подход помогает организовать макет страницы и интерактивность

Текст	Компонент для отображения произвольного текста, используемого для меток, описаний и другого информационного контента
Изображение	Компонент, отвечающий за рендеринг изображений, повышение их визуальной привлекательности и предоставление визуального контента
Ссылка	Гиперссылка на странице, направляющая пользователей на другие страницы или внешние ресурсы
Поле ввода	Поле для ввода текстовых данных пользователем, необходимое для ввода данных и взаимодействия с пользователем
Кнопка	Может инициировать открытие другой страницы или вызвать метод API, является важным элементом взаимодействия с пользователем
Чекбокс	Компонент, позволяющий дать пользователям возможность выбора между двумя вариантами
Радио группа	Набор переключателей, позволяющий выбрать один из нескольких вариантов
Радио кнопка	Часть радиогруппы, предлагающая единственный выбор из набора опций
Радиокнопка по умолчанию (default-value-radio-btn)	определяет, какой переключатель в группе выбран по умолчанию, помогая управлять выбором пользователя и предварительно настраивать значения
Таблица	Компонент для отображения данных в табличном формате, необходимый для представления больших объемов связанных данных

Отображаемый текст	Компонент для отображения текста на экране, используемый для передачи информации и инструкций пользователю
Скролл	Признак, позволяющая регулировать возможность полосу прокрутки для текстового компонента для навигации по большим объемам текста
Способ заполнения (image-fill-params)	Настройки для управления тем, как изображение заполняет свое пространство, обеспечивая правильное визуальное представление
Тип данных поля ввода (input-value-type)	Определяет тип данных, которые принимает поле ввода (например, текст, число, дата), обеспечивая правильное форматирование и проверку данных
Placeholder	Подсказка или инструкция, отображаемая в поле ввода перед тем, как пользователь вводит значение, что повышает удобство использования и направляет ввод данных пользователем
Раскрывающийся список (dropdown)	Компонент, позволяющий выбрать один элемент из выпадающего списка
Множественный выбор (multiselect-mode)	Позволяет выбирать несколько элементов в компоненте ввода, (например, в выпадающем списке)
Ширина в колонках (width-ui)	Определяет ширину компонента пользовательского интерфейса в виде столбцов сетки, что позволяет создавать макеты с выравниванием по сетке
Высота в строках (height-ui)	Определяет высоту компонента пользовательского интерфейса в виде строк сетки, помогая создать единообразный и организованный макет

Месторасположение по X (position-x)	Определяет горизонтальное положение компонента на экране (относительно родителя) для точного управления макетом
Месторасположение по Y (position-y)	Устанавливает вертикальное положение компонента на экране (относительно родителя) для точного управления макетом
Вид фона (background-type)	Определяет стиль фона компонента, параметры соответствуют https://developer.mozilla.org/en-US/docs/Web/CSS/background
Вид рамки (border-type)	Определяет стиль границ компонента, в соответствии с https://developer.mozilla.org/en-US/docs/Web/CSS/border-style
Размер текста в пикселях (text-size)	Размер текста внутри компонента, измеряемый в пикселях
Цвет текста (text-color)	Цвет текста
Подробности (details)	Используется для сворачивания и разворачивания больших объёмов текстовой информации
Размер ячеек (cell-size)	Определяет размер ячеек для сетки страницы
Слой (z-index)	Определяет порядок стека компонентов, необходимый для обработки перекрывающихся элементов и создания глубины интерфейса
JS метод (js-method)	Позволяет управлять поведением страницы с помощью кода
On value changed (event-on-value-changed)	Событие, срабатывающее при изменении значения компонента
Кликнули (event-clicked)	Срабатывает при щелчке клике на компоненте (с помощью ЛКМ или клавиатуры), что необходимо для обработки действий пользователя

OnLoad (event-on-load)	Вызывается, когда страница полностью загружена, используется для выполнения инициализации и др. Позволяет связать с об
Отработало (event-has-worked)	Срабатывает после того, как метод выполнил свою функцию, что полезно для объединения действий или при создании цепочек вызовов
Обработчик (handler)	Инструмент для вызова методов API
Ссылка на метод API (handler-link-to-api-method)	Связывает обработчик с методом API, обеспечивая взаимодействие с процессами и данными на стороне сервера
Входящие атрибуты API (handler-input-attribute)	Определяет атрибуты, которые компонент отправляет в API
Исходящие атрибуты API (handler-output-attribute)	Определяет атрибуты, которые компонент получает от API (важно для отображения динамических данных, например, таблиц)
Number (handler-number)	Тип атрибута обработчика для работы с числовыми данными
Number array (handler-number-array)	Тип атрибута обработчика для работы с массивом числовых данных
String (handler-string)	Тип атрибут обработчика для работы со строковыми данными
String array (handler-string-array)	Тип атрибут обработчика для работы с массивом строковых данных
Bool (handler-bool)	Тип атрибута обработчика для работы с данными логического типа
Bool array (handler-bool-array)	Тип атрибута обработчика для работы с массивом данных логического типа
Object (handler-object)	Тип атрибута обработчика данных объекта для манипулирования сложными структурами данных

Object array (handler-object-array)	Тип атрибута обработчика данных объекта для работы с массивом структурированных данных (например, для получения таблиц)
Данные приложения (global-store)	Хранилище данных всего клиентского приложения (далее – Global Store)
Данные страницы (page-store)	Хранилище данных страницы (далее – Page Store)
Number (store-number)	Целочисленные значения или значения с плавающей запятой, хранящиеся в Page / Global Store
Number array (store-number-array)	Массив целочисленных значений или значений с плавающей запятой, хранящийся в Page / Global Store
String (store-string)	Строковые значения, хранящиеся в Page / Global Store
String array (store-string-array)	Массив строк, хранящийся в Page / Global Store
Bool (store-bool)	Значения логического типа, хранящиеся в Page / Global Store
Bool array (store-bool-array)	Массив значений логического типа, хранящийся в Page / Global Store
Object (store-object)	Структурированные значения, хранящиеся в Page / Global Store
Object array (store-object-array)	Массив структурированных значений, хранящиеся в Page / Global Store
Колонка (grid-table-column)	Колонка (столбец) таблицы
Название колонки (grid-table-column-name)	Отображаемое название для колонки в таблице
Высота заголовка в строках (header-table-front-height)	Высота заголовка колонки таблицы в ячейках сетки страницы
Ширина колонки (grid-table-column-width)	Ширина колонки таблицы в ячейках сетки страницы
Толщина текста (font-weight)	Толщина текста в пикселях

Значения колонки (grid-table-column-values)	Ссылка для привязки значений в колонке к переменной
Value отображаемого текста (fe-runtime-value-text)	Позволяет привязать значение компонента Текст к переменной
Value поля ввода (fe-runtime-value-input)	Позволяет привязать значение компонента Поле ввода к переменной
Value ссылки (fe-runtime-value-link)	Позволяет привязать значение компонента Ссылка к переменной
Value чекбокса (fe-runtime-value-check-box)	Позволяет привязать значение компонента Checkbox к переменной
Value радиогруппы (fe-runtime-value-radio-group)	Позволяет привязать значение компонента RadioGroup к переменной
Value списка (fe-runtime-value-dropdown)	Позволяет привязать значение компонента Dropdown к переменной
Value значений списка (fe-runtime-values-dropdown)	Позволяет привязать массив выборов компонента Dropdown к переменной
Value радиокнопки (fe-runtime-value-radio-button)	Позволяет привязать значение компонента RadioButton к переменной
Value URL ссылки (fe-ahref-url)	Позволяет привязать значение компонента Ссылка к переменной
Value id выделенной строки таблицы (fe-runtime-table-front-selected-line-id)	Позволяет привязать значение поля id выделенной строки таблицы к переменной
Value массива объектов таблицы (fe-runtime-table-front-objects-array)	Позволяет указать переменную массива, из которой будут браться значения колонок таблицы
Адрес страницы (page-route)	Маршрут страницы (URL)
Query параметр страницы (page-route-query-parameter)	Позволяет создать Query параметр страницы
Value для query параметра страницы (fe-runtime-value-page-query)	Позволяет указать на переменную, с которой будет связано значение Query параметра страницы

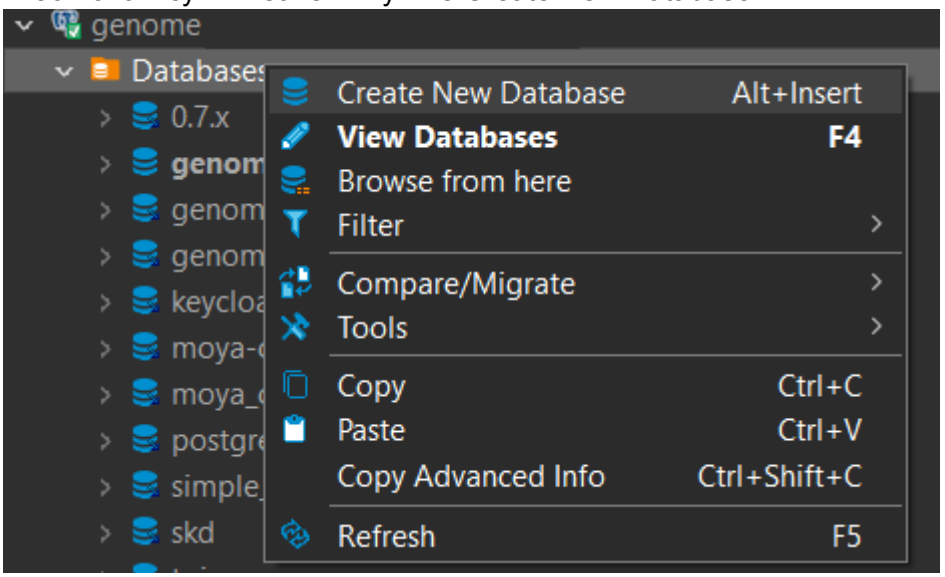
Колонка с кнопкой (grid-table-column-with-button)	Колонка с кнопкой, по клику на которую можно вызвать событие для конкретной записи в таблице
Кнопка колонки (grid-table-columns-button)	Компонент для настройки кнопки для каждой записи
Входящие параметры (js-method-input-params)	Позволяет задать входящие параметры JS-метода
Исходящие параметры (js-method-output-params)	Позволяет задать исходящие параметры JS-метода

Развертывание приложения

Развертывание приложения осуществляется с помощью модального окна «Сборки и релизы приложения», которое открывается через панель инструментов. Окно является контекстно-зависимым и открывается для приложения, выбранного в данный момент в дереве проекта (его название будет отображаться в заголовке окна).

Подготовительные действия

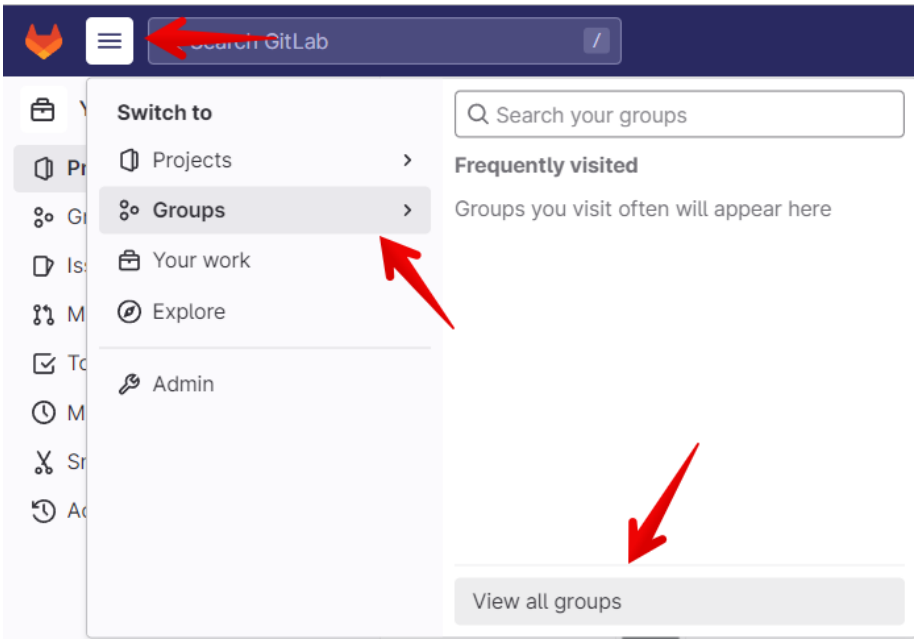
Для генерации и развертывания проекта необходимо совершить первоначально ряд подготовительных действий в СУБД и системе контроля версий СКВ GitLab, а именно:

1. Создать новую БД в СУБД PostgreSql путем выполнения команды на сервере
   psql -u genome -d postgres
   CREATE DATABASE myapp WITH TEMPLATE = template0 ENCODING = 'UTF8' LC_COLLATE = 'en_US.UTF-8';
   либо используя DBeaver в пункте Create New Database
   

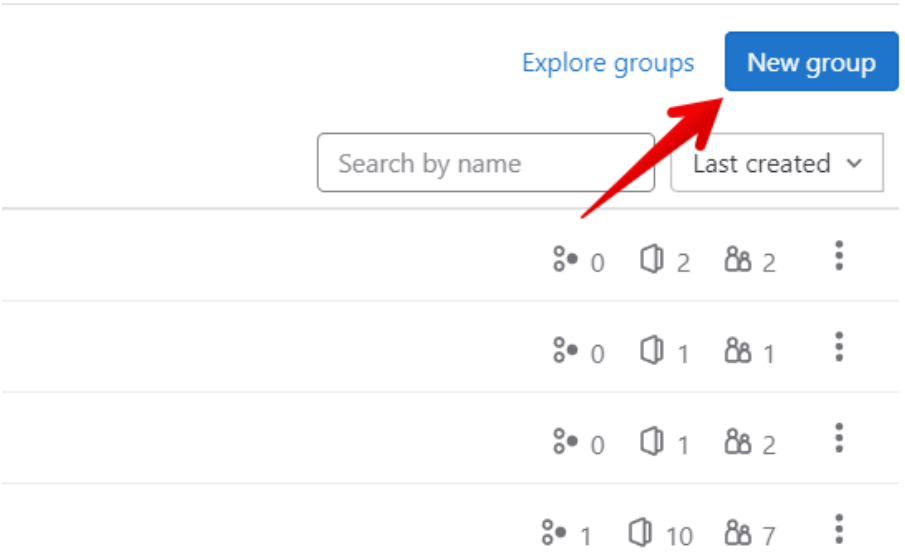
2. (Опционально) Создать пользователя СУБД
   psql -u genome -d myapp
   CREATE ROLE UserName login;
   \password UserName

3. Назначить владельца БД командой
   psql -u genome -d myapp
   ALTER DATABASE myapp OWNER TO UserName;

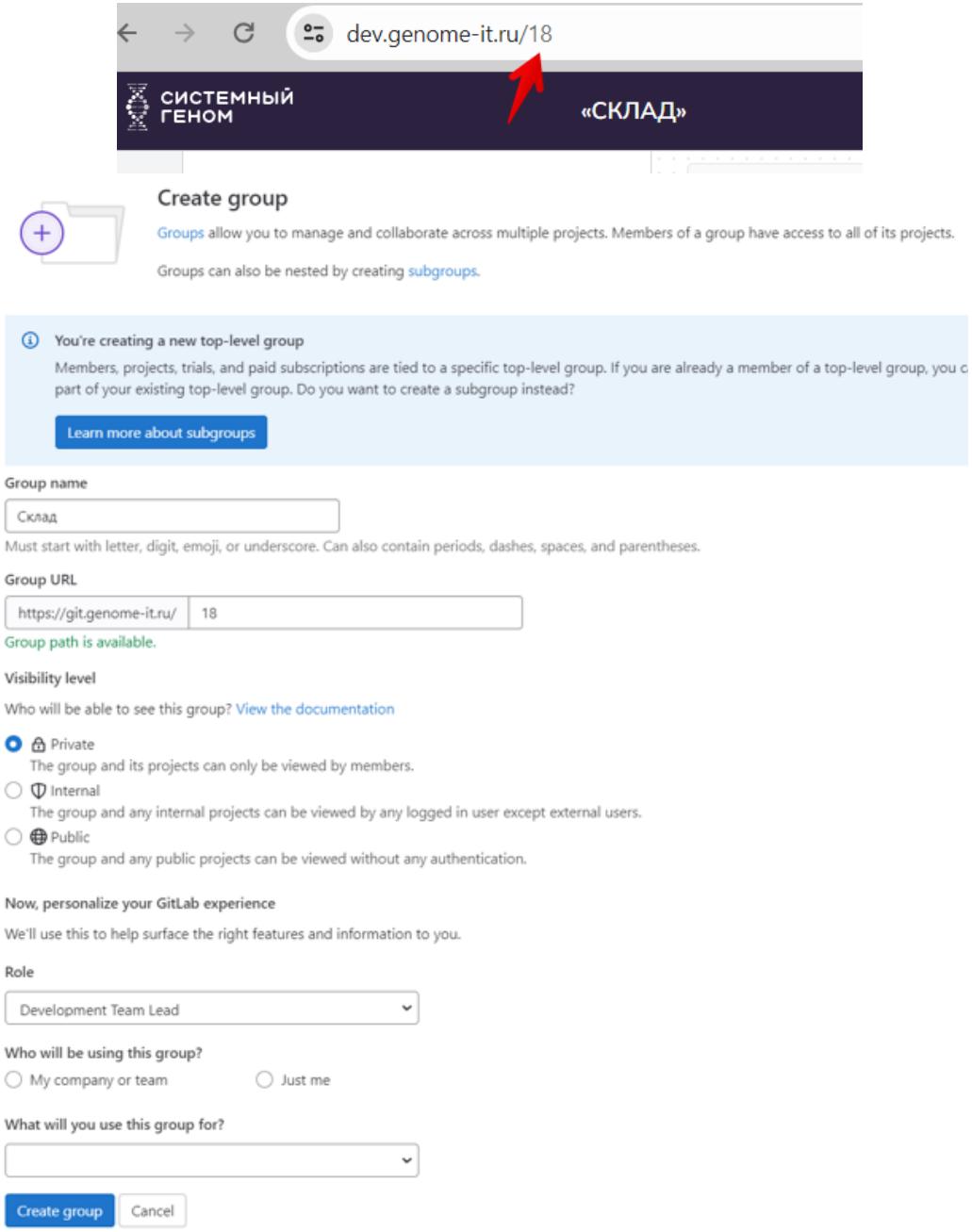
4. Создать группу в GitLab с именем проекта в Genome
   a. Открыть пункт меню, выбрать View all groups



b. Нажать справа на кнопку New group



c. Выбрать пункт Create group
d. Заполнить данные о группе
    i. Group name = Имя проекта
    ii. Group URL = ввести уникальный идентификатор проекта, из Системного Генома. Уникальный идентификатор проекта можно взять из URL при входе в проект



iii. Нажать кнопку Create group

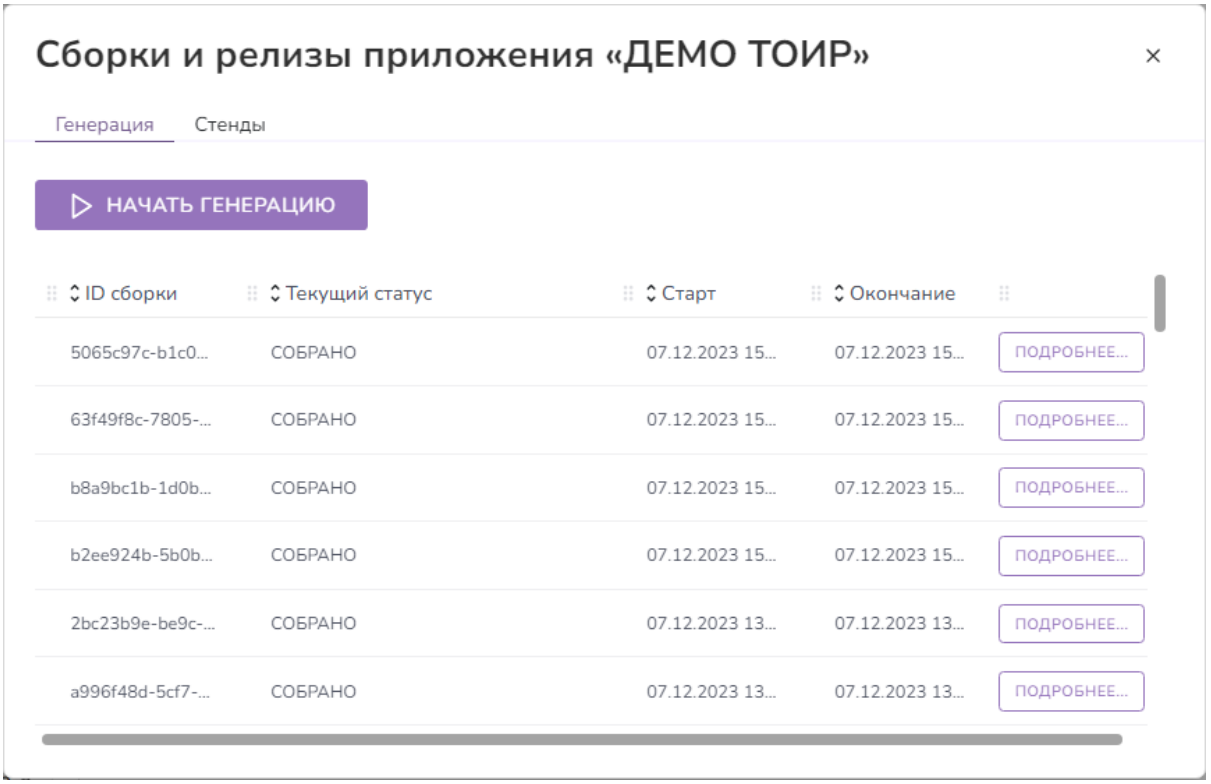
Вкладка «Генерация»

Эта вкладка позволяет запустить генерацию и компиляцию кода приложения, а также содержит информацию о ранее запущенных генерациях приложения.

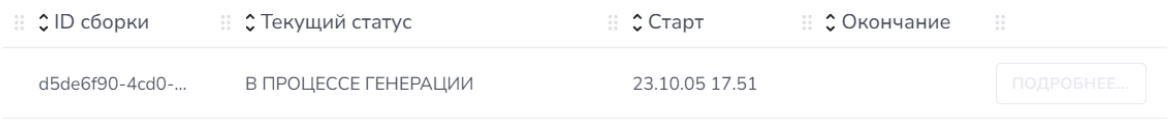
- **Процесс генерации** – при нажатии кнопки «Начать генерацию» сервер копирует информацию о дереве приложения и запускает процесс генерации всех частей приложения, включая серверную часть, пользовательский интерфейс и базу данных.
    - Процесс включает предварительную сборку для проверки сгенерированного исходного кода перед отправкой в репозиторий;

- Сгенерированный код затем фиксируется в ~~стандартной~~ ветке репозитория.

- **Таблица информации о сборке** – отображает сведения о сборке, включая идентификатор сборки, её текущий статус, дата и время начала и окончания, а также кнопку для открытия протокола сборки и полного журнала генерации



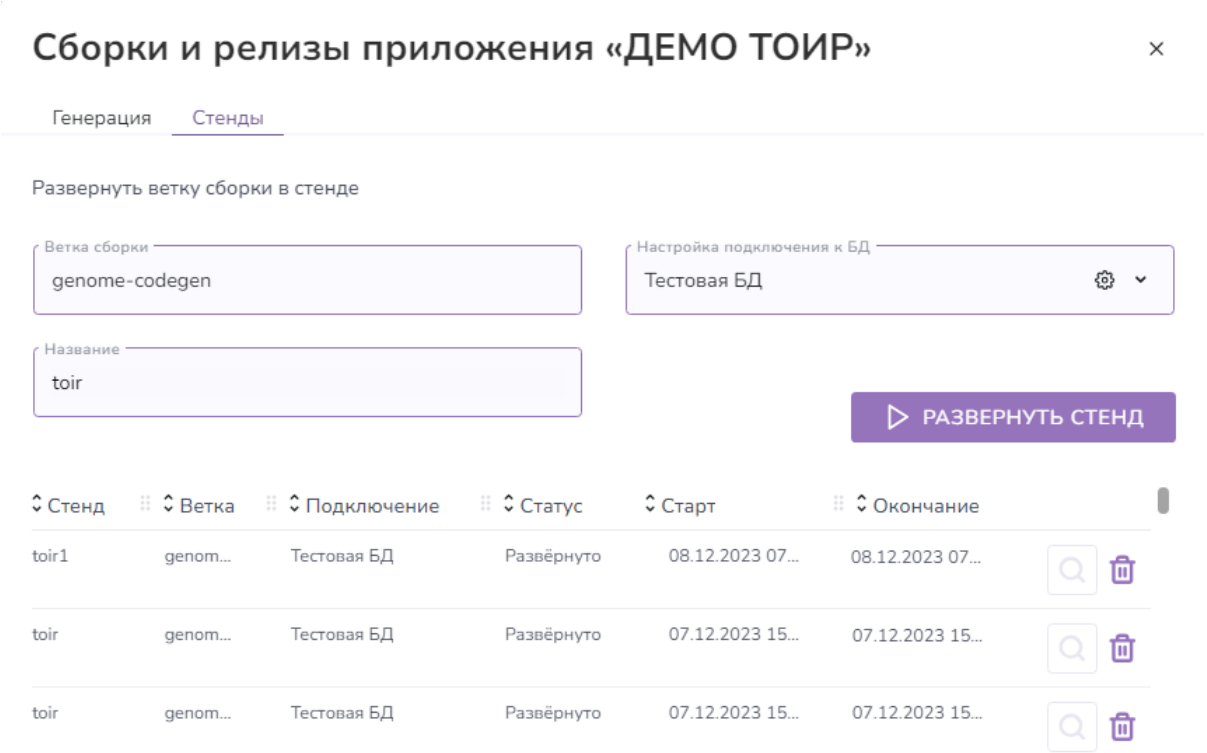
Окно сборки и релиза приложений, вкладка «Генерация»



Пример запущенной генерации в таблице списка генераций

Вкладка «Стенды»

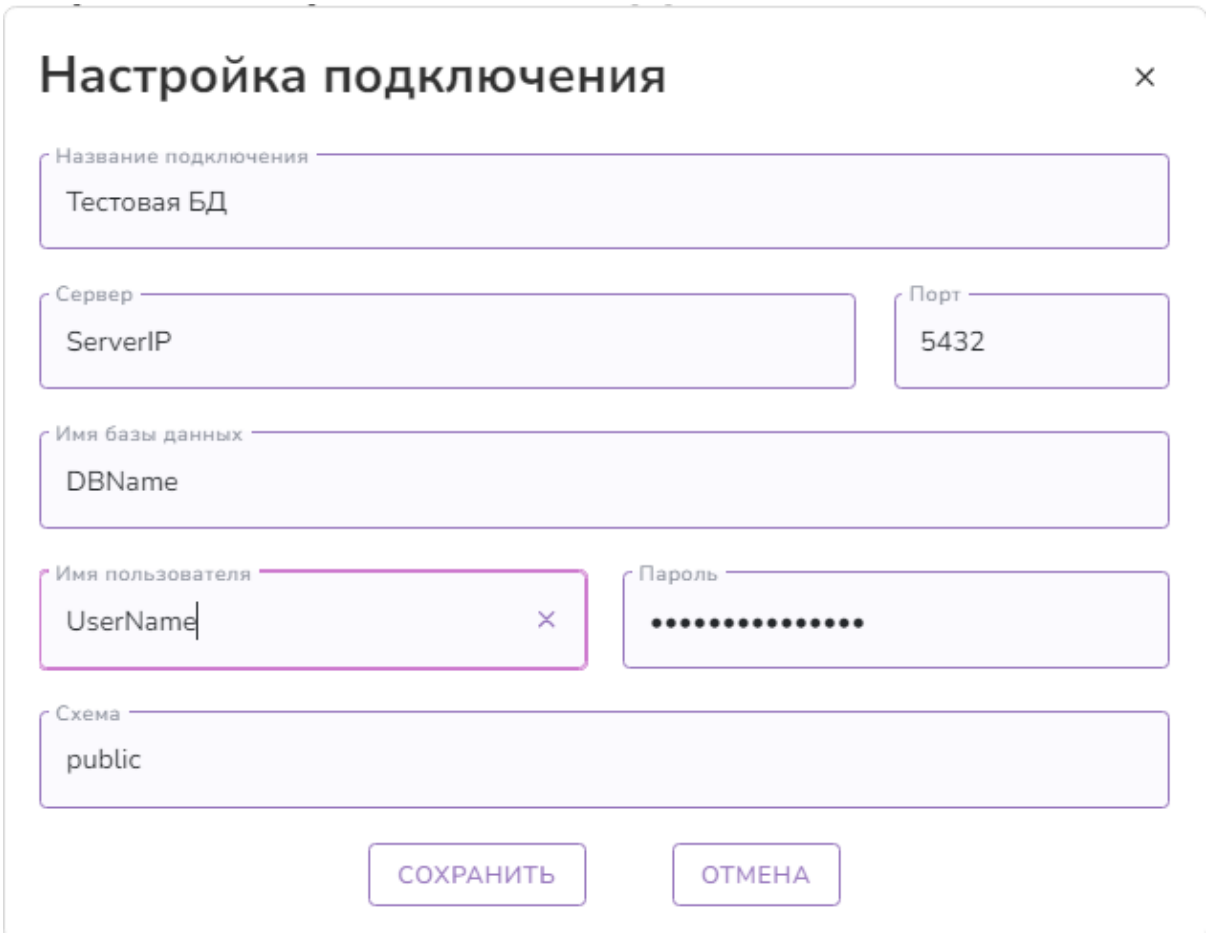
Эта вкладка позволяет развернуть ранее сгенерированную ветку приложения, а также содержит информацию о ранее развернутых стендах приложения.



Окно сборки и релиза приложений, вкладка «Стенды»

- **Выбор ветки** – позволяет выбрать конкретную ветку, что полезно для организаций, вносящих изменения в сгенерированный код и работающих для этого в своих настраиваемых ветках репозитория.
- **Название стенда** – позволяет ввести название стенда
- **Настройки подключения к базе данных**
    - Шаблон подключения можно выбрать из выпадающего списка
- **Развернуть стенд** – при нажатии кнопки «Развернуть стенд» сервер запускает процесс развертывания приложения на стенде с заданными параметрами.
- **Таблица информации о стендах** – отображает список ранее развернутых стендов приложения, включая название стенда, ветку сборки, текущий статус стенда, дата и время начала и окончания развёртывания, а также кнопку для открытия диалога с подробностями развёртывания стенда (если развёртывание закончилось с ошибкой) и кнопку для удаления записи из списка.

- **Дополнительные** конфигурации настроек подключения доступны, если щелкнуть значок шестеренки рядом с выбранным шаблоном. Открывается дополнительное модальное окно для детальной настройки шаблона подключения к базе данных



Окно «Настройки подключения»