Настройки приложения

Система настроек

В системе, помимо кода приложения, присутствуют настройки, которые отвечают за бизнес-логику приложения и его визуальную составляющую. Эти настройки позволяют гибко управлять различными модулями системы.

Основные возможности настроек:

Регулирование элементов системы:

- Отчеты
- Формы
- График
- Правила

Технология настроек:

- JSONata: Настройки реализованы с использованием JSONata — языка для обработки и трансформации данных в формате JSON
- Rudderstack: Также используются настройки на RudderStack, который представляет собой язык программирования для трансформации JSON-данных. RudderStack включает в себя JSON Template Engine, который упрощает преобразование данных из одного формата в другой, минимизируя накладные расходы на выполнение и улучшая производительность. Он генерирует оптимизированный JavaScript-код из шаблонов трансформации, что позволяет легко добавлять новые шаблоны и поддерживать их.

Доступ к настройкам:

Доступ к настройкам осуществляется через базу данных PostgreSQL с использованием DBeaver.

Настройки для различных модулей системы организованы в отдельные таблицы базы данных, что обеспечивает удобство управления и гибкость в настройках. Ниже представлены основные таблицы, используемые для хранения настроек:

- Таблица forms - Используется для настройки форм в системе.
- Таблица flex_settings - Применяется для гибких настроек, позволяя адаптировать систему под специфические требования.
- Таблица mail_reports - Служит для настройки рассылаемых актов, обеспечивая автоматизацию отчетности.
- Таблица reports - Используется для настройки печатных отчетов, позволяя формировать документы в нужном формате.
- Таблица settings - Применяется для настройки различных модулей системы, обеспечивая их корректную работу.
- Таблица settingTypes - Используется для хранения настроек по умолчанию, определяющих базовые параметры системы.

Использование настроек:

Настройки в системе могут быть заданы на нескольких уровнях, что обеспечивает гибкость и индивидуальный подход к каждому пользователю.

- Настройки для ролей:
  - Каждая роль в системе может иметь свои собственные настройки, которые применяются ко всем пользователям, работающим под данной ролью. Это позволяет управлять доступом и функциональностью на уровне группы пользователей.

- Персональные настройки для пользователей:
  - Кроме настроек, связанных с ролями, каждая учетная запись пользователя может иметь свои персональные настройки. Эти настройки имеют приоритет над настройками, заданными для роли.

- Настройки по умолчанию:
  - Если для пользователя не заданы персональные настройки и для его роли отсутствуют соответствующие настройки, система применяет настройки по умолчанию. Эти настройки определяют базовые параметры и функциональность.

Логика применения настроек:

При определении активных настроек система следует следующему порядку приоритета:

1. Персональные настройки пользователя:
   - Если у пользователя есть индивидуальные настройки, они применяются в первую очередь.

2. Настройки роли:
    - Если персональные настройки отсутствуют, система проверяет настройки, связанные с ролью пользователя.
3. Настройки по умолчанию:
    - Если ни персональные настройки, ни настройки роли не заданы, система использует настройки по умолчанию.

Таким образом, данная иерархия позволяет обеспечить максимальную гибкость и адаптивность системы, удовлетворяя потребности как отдельных пользователей, так и групп пользователей с одинаковыми ролями.

Описание и работа с настройками

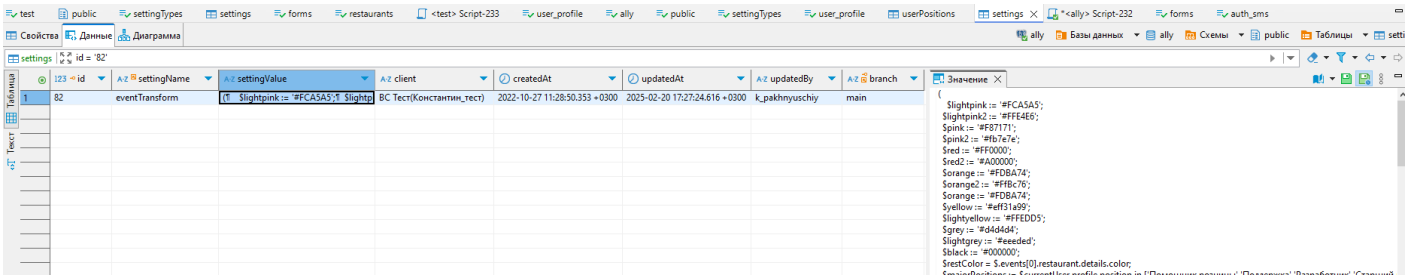
Структура таблицы settings

Таблица settings предназначена для хранения настроек, которые могут быть применены как к ролям, так и к пользователям. Она имеет следующую структуру:

Поля таблицы:

- settingName - Название настройки, которое описывает ее назначение.
- settingValue - Код самой настройки.
- client - Поле, описывающее, для кого предназначена данная настройка (например, для конкретного клиента или группы пользователей).
- branch - Указывает, в какой ветке находится данная настройка, используется для управления версиями или различными конфигурациями.

Пример записи из таблицы settings:

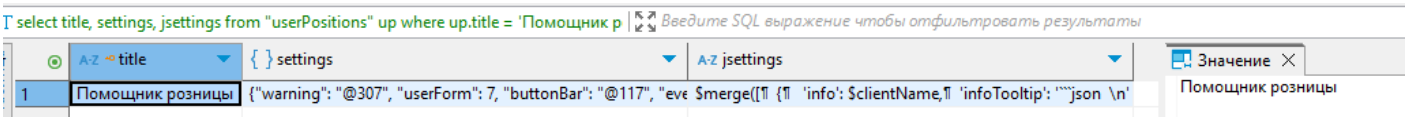


Настройки для ролей

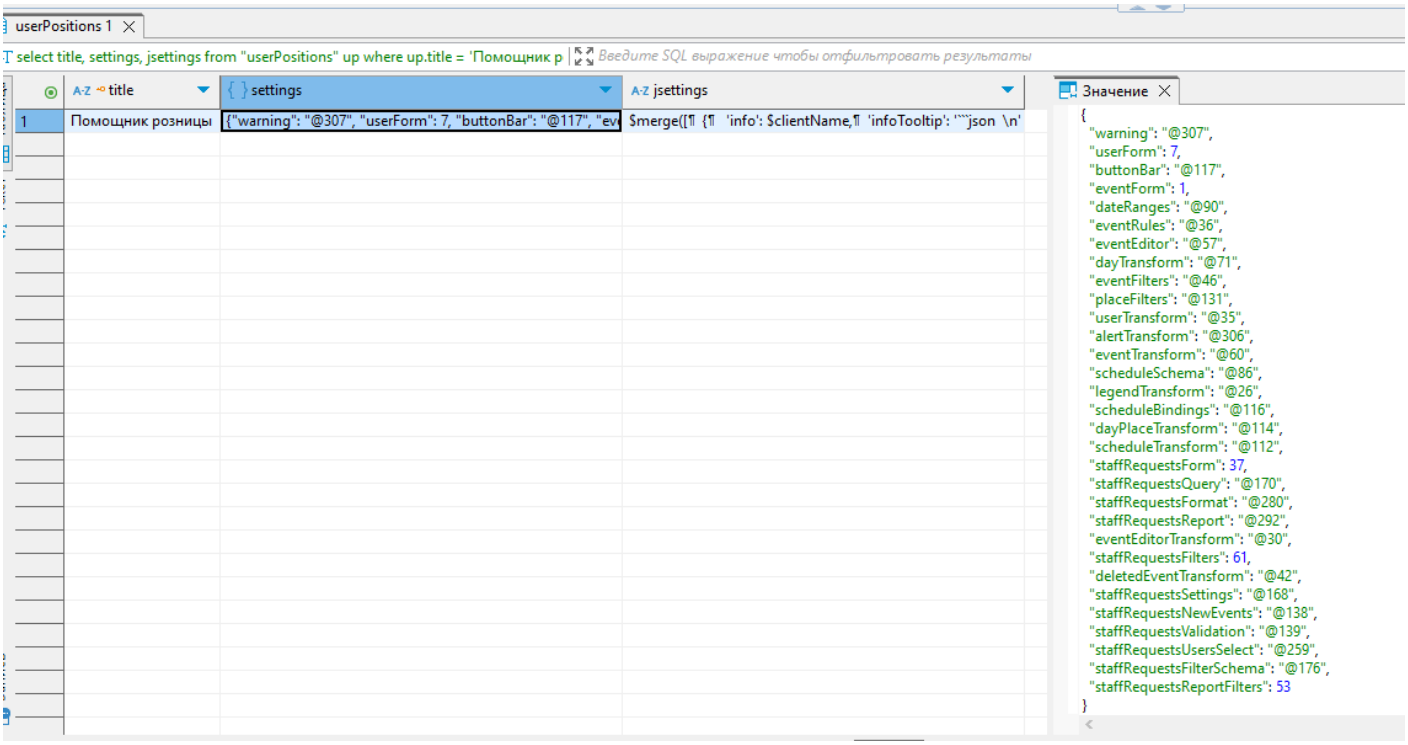
В таблице userPositions настройки для ролей задаются в полях settings и jsettings:

- settings - Используется для хранения настроек в формате JSON.
- jsettings - Позволяет использовать JSONata для задания условий, например, для применения настроек только для определенной версии продукта.

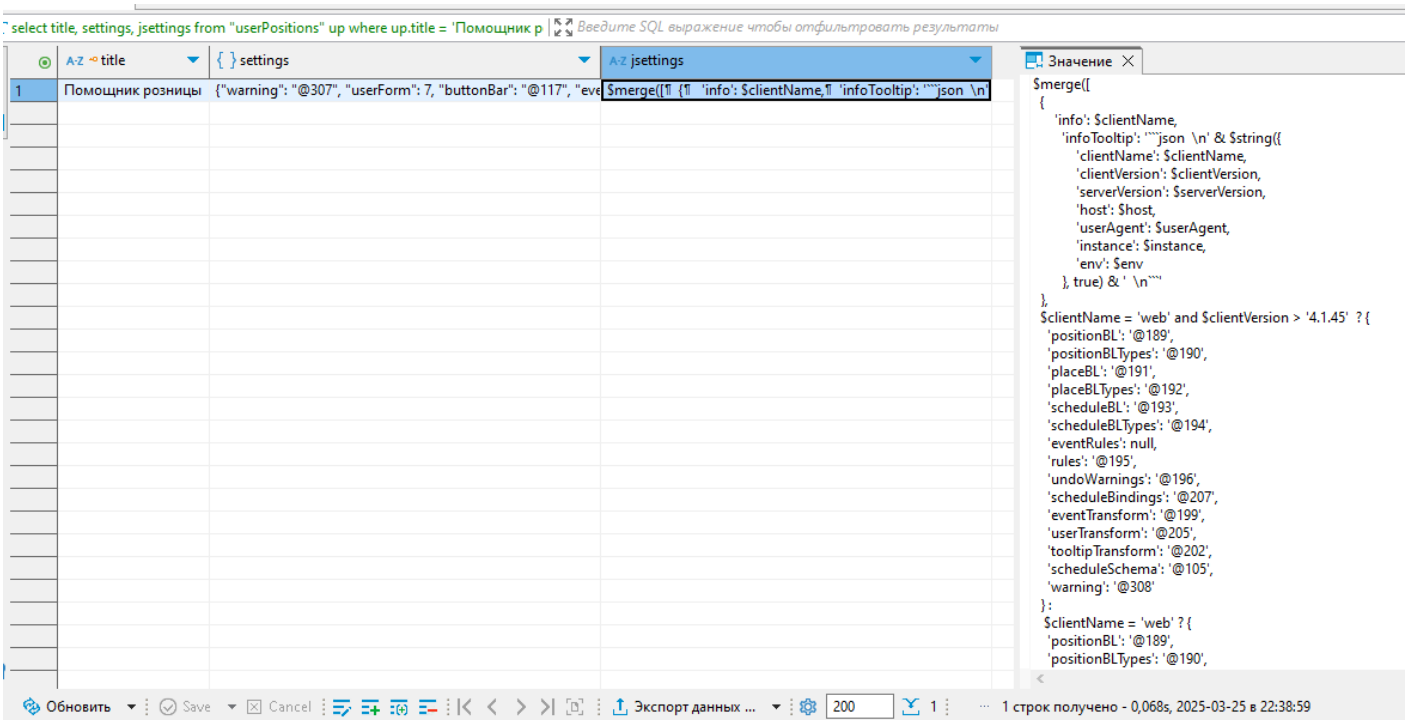
Пример записи из таблицы userPositions:



Пример поля settings из таблицы userPositions:



Пример поля jsettingsиз таблицы userPositions:



Приоритет настроек:

Настройки в jsettings имеют более высокий приоритет по сравнению с настройками в settings. Если для одной и той же настройки заданы разные идентификаторы в обеих таблицах, будет применена настройка из jsettings.

Комбинирование настроек:

- Если в jsettings заданы настройки, они используются в первую очередь. Затем, если в settings есть другие настройки, они комбинируются с теми, которые уже определены в jsettings.
- Аналогично, если у роли установлены определенные настройки, а у пользователя есть дополнительные настройки, которые отсутствуют у роли, то применяются настройки роли плюс индивидуальные настройки пользователя. Тоже самое работает и наоборот
- Если у пользователя и у роли задана настройка с одинаковым названием, но с разными идентификаторами, то будет использована настройка пользователя, и идентификатор роли не будет добавлен. Таким образом, приоритет остается за настройками пользователя.

Правила добавления настроек:

При добавлении настроек в таблицу settings необходимо следовать определенным правилам форматирования:

- Если настройка не является настройкой формы или отчета:

Идентификатор настройки указывается в кавычках и предшествует знаком собачки (@), например:

```
"название_настройки": "@идентификатор"
```

Пример:

```
"warning": "@307"
```

- Если настройка является настройкой формы или отчета:

Идентификатор указывается сразу после двоеточия без кавычек и знака собачки, например:

```
"название_настройки": идентификатор
```

Пример:

```
"userForm": 7
```

Ветки настроек

Механизм создания веток настроек:

1. Создание веток:
   - Вводится механизм создания отдельных веток настроек для конкретных задач.
   - Ветки можно мержить с основными настройками по завершении задач (аналогично git).
2. Основные настройки:
   - Все базовые настройки относятся к ветке main.
3. Процесс работы:
   - При начале новой задачи создается новая ветка настроек с заданным именем на основе родительской ветви.
   - По умолчанию новая ветка наследует все настройки из родительской.
4. Изменения настроек:
   - При изменении настройки и ее сохранении формируется новая настройка с тем же ID в новой ветке.
   - Это исключает перезатирание одной настройки другой.
5. Тестирование:
   - В любой момент можно указать тестировщику ветку с новыми настройками для тестирования.
   - Тестировщик может протестировать настройки, просто переключив ветку.
6. Завершение работы:
   - По завершении задачи все изменения мержатся с родительской веткой.
   - Созданная ветка удаляется.

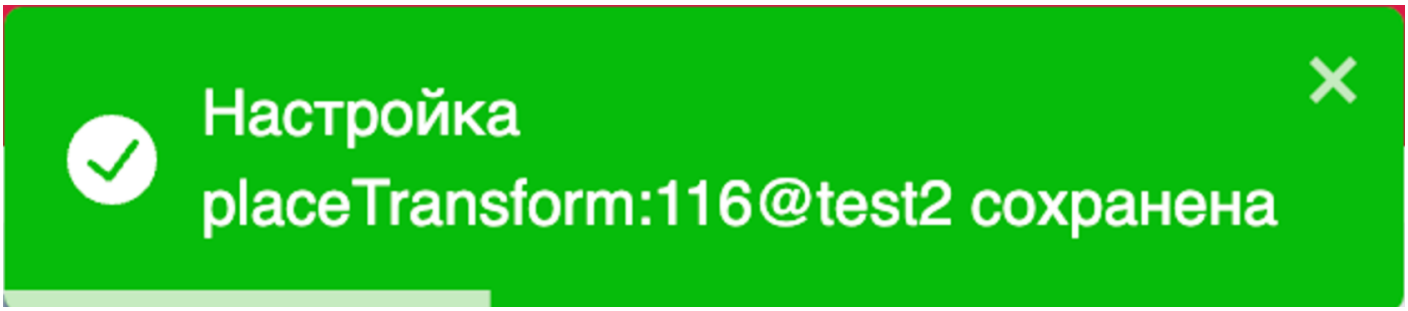
Базовая ветка всегда остается чистой, не содержит лишних настроек и готова к синхронизации с продакшеном.

Техническая реализация:

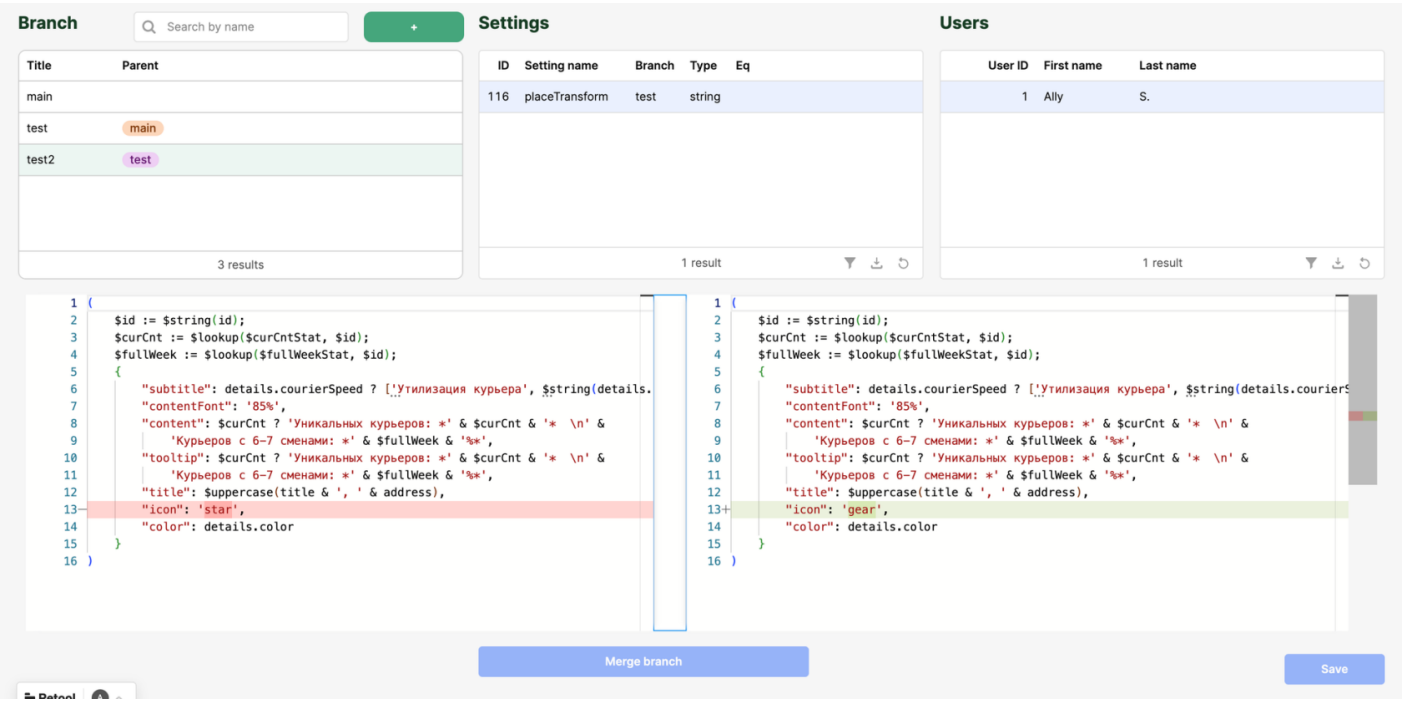
1. Структура таблицы branches:
   - title: имя ветви.
   - parent: родительская ветвь.
2. Доступ к данным:
   Данные из таблицы доступны через запрос branches с привилегиями:
   - app:branches:view
   - web:branches:view
3. Связь с другими таблицами:

- В таблицы user_profile и settings добавлено поле branch, ссылающееся на таблицу branches.

4. Получение настроек:
   - При получении настроек:
     - Сначала берутся настройки из текущей ветки пользователя.
     - Затем из родительской ветки.
     - Далее из родительской ветки по отношению к родительской и так далее, вплоть до ветки main или до settingTypes, если настройки отсутствуют и там.
5. Редактирование настроек:
   - При редактировании настройки через левую панель:
     - Формируется новая настройка с заданным ID в текущей ветке пользователя, если она не существует.
     - Если настройка уже создана, она обновляется.
     - Если настройки с таким ID нет и в родительских ветвях, создается новая настройка с новым ID, который записывается в settings текущего пользователя.
6. Уведомление:
   - Информация о сохраненной настройке отображается во всплывающем сообщении.



В Retool добавлена задача Settings позволяющая формировать новые ветви и мержить их с родительскими: http://10.1.241.240:3000/apps/9fd9f242-3dd2-11ef-a965-8b7ae0e8ee4e/Settings

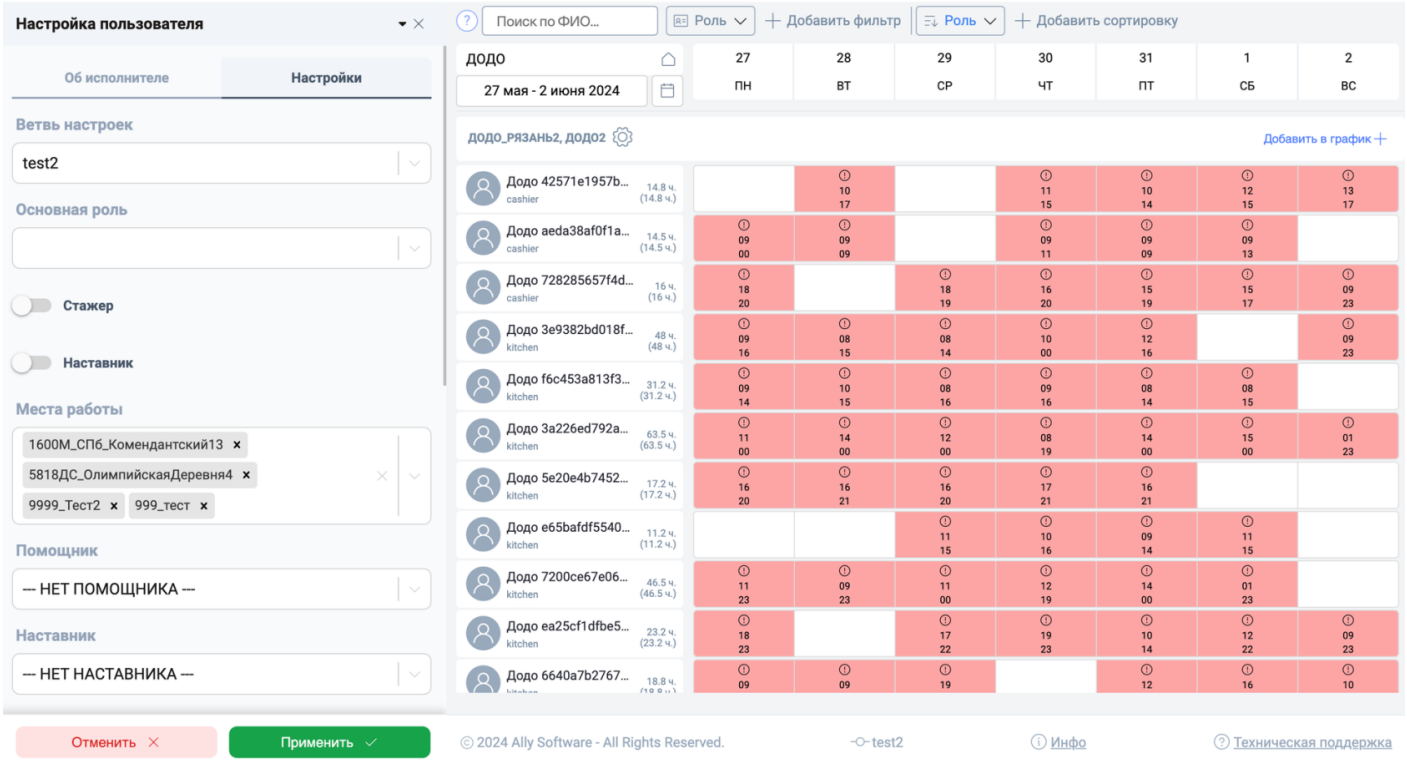


Панели управления ветвями и настройками:

1. Панель Branch:
   - Позволяет:
     - Выбрать ветвь.
     - Изменить родительскую ветвь.
     - Создать новую ветвь при необходимости.
2. Панель Settings:
   - Отображает настройки, созданные в данной ветви.
   - Поле Branch:
     - Отражает ветвь с родительской настройкой.
     - Если поле пусто, значит базовой настройки нет, и используется settingTypes.defaultValue.
   - Поле eq:
     - Отражает равенство базовой и модифицированной настроек.
3. Панель Users:
   - Отображает пользователей, использующих данную ветвь настроек.
4. Панель редактора:
   - Отображает:
     - Исходный вид выбранной настройки (слева).
     - Модифицированную настройку в выбранной ветви (справа).
5. Слияние ветвей:
   - Перед слиянием необходимо устранить все отличия в настройках между ветвями:
     - В панели Settings в поле Eq для каждой настройки должна стоять галочка.

- Устранить различия можно:
  - Приведя родительскую настройку к состоянию дочерней (левая панель редактора).
  - Или дочернюю к состоянию родительской (правая панель редактора).
- Кнопка Save сохраняет изменения в настройках.
  - Если у настройки нет родительской, изменения сохраняются в settingTypes.defaultValue.
- После устранения всех изменений становится доступной кнопка Merge branch, которая:
  - Удаляет все настройки из данной ветви.
  - Переключает всех пользователей на родительскую ветвь.
  - Удаляет ветвь из таблицы branches.

6. Дополнительные предложения:
   - Предлагается переключать текущую ветвь пользователя через форму редактирования.
   - Информация о текущей ветви может отображаться в футере.



Редактор настроек

Для редактирования настроек в системе используется редактор настроек, который доступен в левой панели интерфейса.

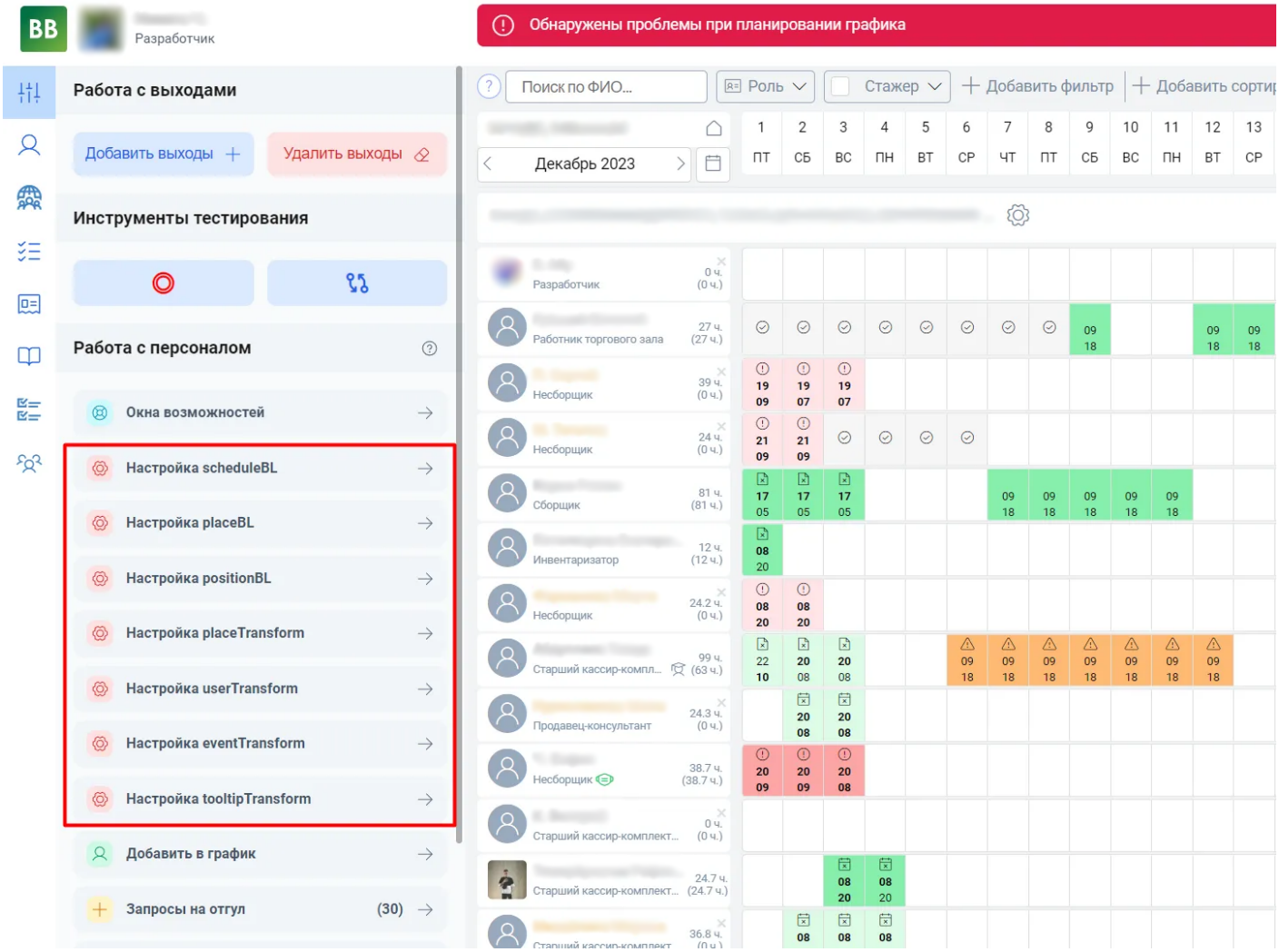
Требования для работы редактора:

1. Скоуп доступа:
   - Для использования редактора настроек необходим скоуп web:setting:update, который обеспечивает права на редактирование настроек.
2. Настройка editableSettings:
   - В настройке editableSettings указываются конкретные настройки, которые будут доступны для редактирования. Это позволяет ограничить доступ к редактированию только определенных параметров, обеспечивая безопасность и контроль.

Функциональные возможности редактора:

1. Подсветка синтаксиса:
   - Редактор поддерживает подсветку синтаксиса, что облегчает процесс редактирования и позволяет пользователям легче ориентироваться в коде.
2. Вывод ошибок:
   - В случае возникновения ошибок при редактировании, редактор предоставляет вывод ошибок, что помогает пользователям быстро идентифицировать и исправлять проблемы.
3. Мгновенное отображение изменений:
   - Результаты отредактированных настроек отображаются в графике немедленно, без необходимости перезагружать страницу. Это позволяет пользователям видеть изменения в реальном времени и улучшает взаимодействие с системой.
4. Описание переменных
   - В редакторе настроек при наведении на переменные отображается информация, которая помогает пользователям понять их назначение и использование.

Скриншот редактора настроек:


Разработчик
Обнаружены проблемы при планировании графика
Работа с выходами
Добавить выходы
Удалить выходы
Инструменты тестирования
Работа с персоналом
Окна возможностей
Настройка scheduleBL
Настройка placeBL
Настройка positionBL
Настройка placeTransform
Настройка userTransform
Настройка eventTransform
Настройка tooltipTransform
Добавить в график
Запросы на отгул
(30)
Поиск по ФИО...
Роль
Стажер
Добавить фильтр
Декабрь 2023
1 ПТ
2 СБ
3 ВС
4 ПН
5 ВТ
6 СР
7 ЧТ
8 ПТ
9 СБ
10 ВС
11 ПН
12 ВТ
13 СР
Разработчик
Работник торгового зала
Несборщик
Несборщик
Сборщик
Инвентаризатор
Несборщик
Старший кассир-компл...
Продавец-консультант
Несборщик
Старший кассир-комплект...
Старший кассир-комплект...

Эталонные настройки

Для обеспечения корректного тестирования в тестовой среде созданы эталонные настройки, которые представляют собой копию актуальных настроек с продакшена. Эти настройки добавлены к ролям, которые настроены в тестовой среде аналогично продакшену.

Формат эталонных настроек

Эталонные настройки имеют следующий вид:

- Эталонная настройка: "название" "айди" с продакшена

Логика разработки и переноса настроек через эталонные

Процесс разработки и переноса настроек включает несколько этапов:

- Создание тестовой настройки:
  - Изначально создается тестовая настройка, которая служит основой для дальнейших проверок и доработок.
- Проверка тестовой настройки:
  - После создания тестовая настройка проходит проверку, чтобы убедиться в ее корректности и соответствии требованиям.
- Слияние с эталонной настройкой:
  - После успешной проверки тестовая настройка мерджится в эталонную настройку на тесте, что позволяет обновить эталонные параметры.
- Выкладка на продакшен:
  - После завершения всех проверок и слияний обновленные эталонные настройки выкладываются на продакшен, обеспечивая актуальность и корректность работы системы.

Пример эталонной настройки в таблице settings: