Задача:

Реализовать универсальную основу для отрисовки объектов (блоков). Используем технологию canvas и paper.js и современный JS

На данном этапе необходимо реализовать только графику и некоторые взаимодействия с пользователем (hover, zoom, клики, перемещение блоков). То есть, нарисовать все нужные варианты отображения объектов, реализовать перемешение блоков, реализовать зуммирование, реализовать "интерактив" в виде изменения цвета текста, фона, бордюра, тени и т.д. для объектов, которые имеют обработчик клика (имеют ховер, когда объект кликабельный)

На выходе мы должны получить библиотеку, рисующую на canvas необходимые элементы с настраиваемыми параметрами.

При выполнении задачи учитываем следующие требования:

1. Умный зум - центрируем и увеличиваем ту область, над которой находится курсор (посмотреть в качестве примера сервисы draw.io, карты гугл, яндекс и т.д.). При самом минимальном зуме на экране должна помещаться вся схема - последние шаги "отзумливания" должны рассчитываться с этим учетом. Когда объекты слишком мелкие, то должна отключаться некоторый функционал работы с блоками. Как частный случай - отключение возможности работы с "коннекторами" (о них ниже) .
2. Область отображения объектов должна свободно двигаться мышкой (скролинг) - при перемещении курсора с одновременной нажатой правой или левой кнопки мыши на свободном пространстве.
3. Блоки можно свободно перетаскивать по рабочему полю, это необходимо для логической организации блоков (чтобы понятно и красиво было).
4. Выделение, то есть подсветка активного (текущего выбранного элемента), выделено может быть сам блок, и поле внутри блока - при этом блок тоже выделен.
5. Выделение блока над которым находится курсор мыши. Выделение конкретного поля при нахождении над ним курсора, если выбран блок.
6. Между блоками могут быть связи в виде стрелок. Стрелка должна быть нарисована между полем и связанным с полем блоком и выходит всегда с правой стороны блока от конкретного поля. Стрелки лучше делать красивыми сплайнами немного изогнутыми чтобы визуально было красиво. Стрелки должны изменяться в соответствии изменением положения блоков.

Блок

По сути это основной графический объект который будет использован в дальнейшем в различных редакторах. Блок состоит из строк. Первая строка это заголовок блока. Блока можно считать таблицой с одной колонкой. Объект "блок" автоматически высчитывает максимальную строку которую содержит и принимает её размер. То есть все строки меньшего размера будут выровнены по максимальной. Для этого у нужены две функции одна расчет размеров, и вторая сам render.

Свойства:

1. Фон. Или один цвет или 2 цвета (указан второй параметр отдельной переменной). Если 2 цвета то отображаем линейный градиент сверху вниз.
2. Цвет рамки
3. Скругление рамки
4. Толщина линий рамки
5. Цвет разделителей строк (если не указан нету разделителей)
6. Толщина линий разделителей
7. Отступы откраев рамки и до контента берем из настроек (описано ниже)
8. + может еще что-то добавиться по смыслу

Строка блока

Блок состоит из внутренних объектов - строк, каждая из которых имеет следующие параметры:

1. Может обладать иконкой. Вывод иконки слева внутри строки
2. Основной контент строка. Может быть позиционирован внутри строки, слева, по центру, справа. Если есть иконка то иконка всегда левее. Строка может содержать символы перевода строки их нужно адекватно обрабатывать т.е. Делать многострочное отображение.
3. Доп строка. Позиция справа. Если основной контент позиционирован справа то отображение последовательно: [основной] [доп строка]
4. Высота строки (если не указана то вычислять автоматом согласно размерам строк, иконки и доп)
5. Отступы сверху/снизу слева справа берутся из объекта списка.
6. Цвет текста (если не указано то берем из списка, если не указано в списке берем глобально, ниже опишу)
7. Цвет текста в режиме hover (см. настройки)
8. Цвет тени в режиме hover (размытие)
9. Цвета для режима selected (когда строка выбрана)
10. Крайне желательно сделать возможность отображения фона у строки, это может вызвать трудности для первой и последней строки в случаи когда рамка всего списка скруглена (по этому отображение фона строки можно сделать в конце работы и продумать варианты)
11. Настройка шрифта
12. Возможность добавить визуальный "коннектор" к полю - это активное поле в виде "кружка", которое отображается с правой стороны поля. Наличие "коннектора" у поля позволяет пользователю добавить связь в виде стрелки к другому блоку, потянув мышкой с зажатым курсором от "коннектора" к соответствующему блоку. Цвет коннектора можно устанавливать.
13. + могут быть добавлены еще доп настройки. Изначально не строит делать все возможные варианты настроек, в реальной жизки использовано не так уж много я почти все описал. По ходу работы добавим нужные чтобы не тратить много времени.

Блок всегда имеет первую строку являющуюся заголовоком, а так же опционально - иконку. Стиль заголовка и строк настраиваются отдельно.

Объект стрелки
Стрелка существует как отдельный графический объект. По сути содержит 2 координаты начало и конец стрелки. Объект стрелки рендерится при формировании блока, если есть связь. Стрелки рендерятся после расчета размеров всех блоков на неком втором этапе расчета специального для стрелок.

Объект "стока блока" хранит ссылки на стрелки, чтобы обновлять координаты в случаи изменения позиции или размеров блока. Из поля могут выходить более одной стрелки. *Настройки*

- Цвет
- Толщина линии

Настройки

Есть несколько уровней настроек, если какой-то настройки нету на нижнем уровни пробуем взять ее на более верхнем и так далее (наследование). В связи с этим предлагаю ввести уникальные текстовые названия для настроек, чтобы удобно было их определять, тем более что настроек не так уж и много.

1. Глобальный уровень настроек. На нем предпологаю наличие всех настроек, размеров, цветов, отступов и так далее.
2. Уровень объекта списка, находиться внутри самого объекта списка
3. Уровень строка списка. В связи с этим логично сделать объект настроек который может хранрить в себе настройки и быть связанным с другими объектами родителями.

```
//настройки
{
    "global":{
        "block.frame.border":"#D6B656",
        "block.frame.border_radius":4,
        "block.frame.background":"#FFF2CC",
        "block.frame.background2":"#FFD966",

        "block.hover":"#3527e8",  //цвет выделения блока когда на вели мышку
        //при наведении мышки я делал так:
        //   frame.strokeColor = style.block.hover;  <--- тут тот самый цвет
        //   frame.shadowColor = style.block.hover;
        //   frame.shadowBlur =  8;

        "block.select":{       //вариант выделения блока это нарисовать отдельно
//пунктирную рамку вокруг самого блока
                "strokeColor": "#2486ff",
                "dashArray": [4, 10],
        },

        "title.font_family": "Helvetica",
        "title.color": "#000000",
        "title.font_size": "14",

        "row.padding_top":5,
        "row.padding_bottom":5,
        "row.padding_left":10,
        "row.padding_right":10,

        "row.font_family": "Helvetica",
        "row.color": "#000000",
        "row.font_size": "11",
    },
    //настройки уровня блока загружаются из json описания сущности
}
```

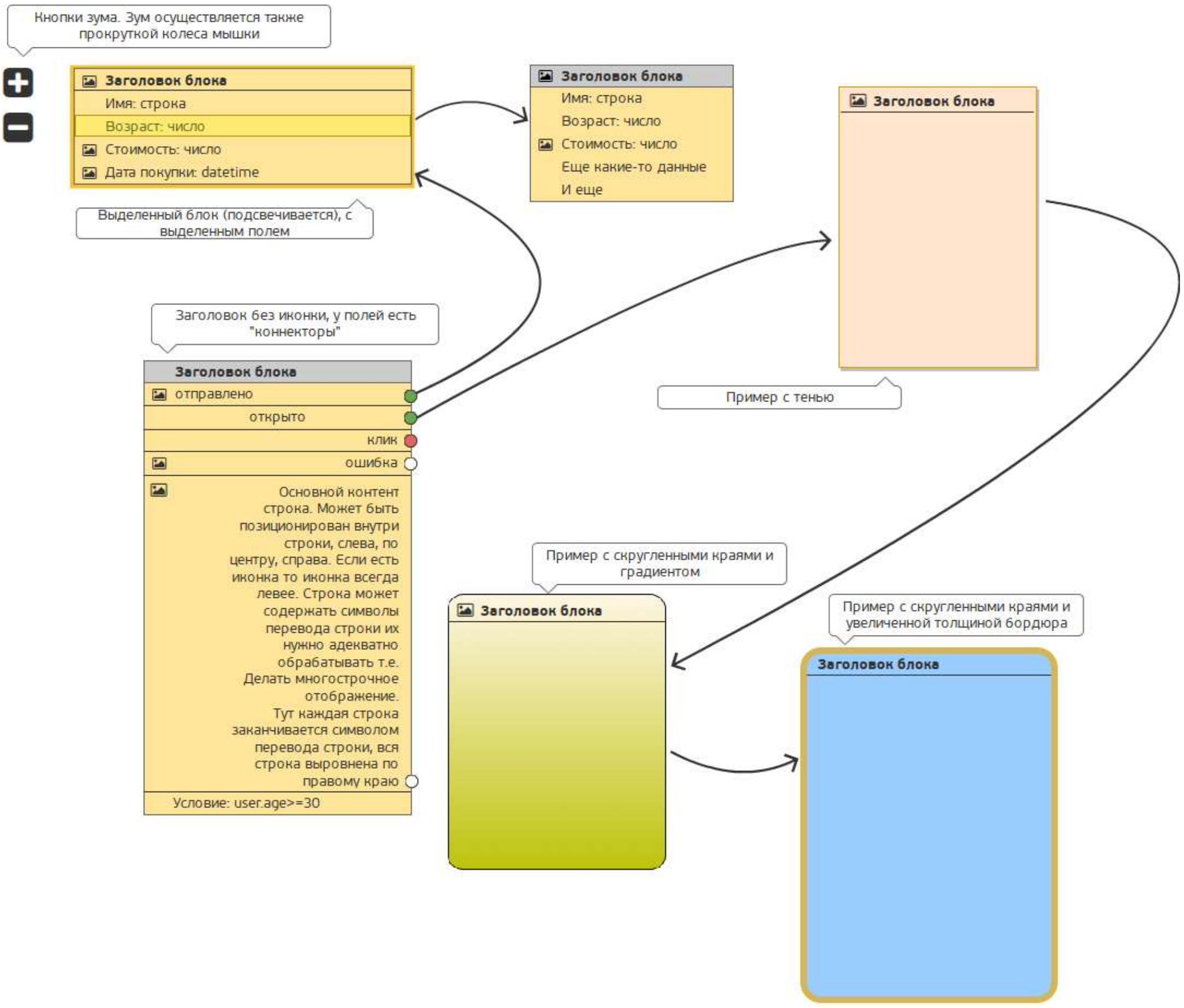
Идея, как можно использовать настройки стилей в коде

```
....
var all_blocks = new Style({
        "block.frame.border":"#D6B656",
        "block.frame.border_radius":4,
        "block.frame.background":"#FFF2CC",
});
var green_blocks = new Style({"block.frame.background":"#00FF00"}, all_blocks);
var red_blocks = new Style({"block.frame.background":"#FF0000"}, all_blocks);
var blue_blocks = new Style({"block.frame.background":"#0000FF"}, all_blocks);
//может быть любая вложенность
var super_blocks = new Style({"row.color": "#FF0000",}, red_blocks);
....
var block1 = new Block("Заголовок блока", red_blocks);
var row1=block1.add_row("Тестовая строка1");  //должны быть доступны параметра строки.
block1.add_row("Тестовая строка2").set_style(super_blocks);
....
```

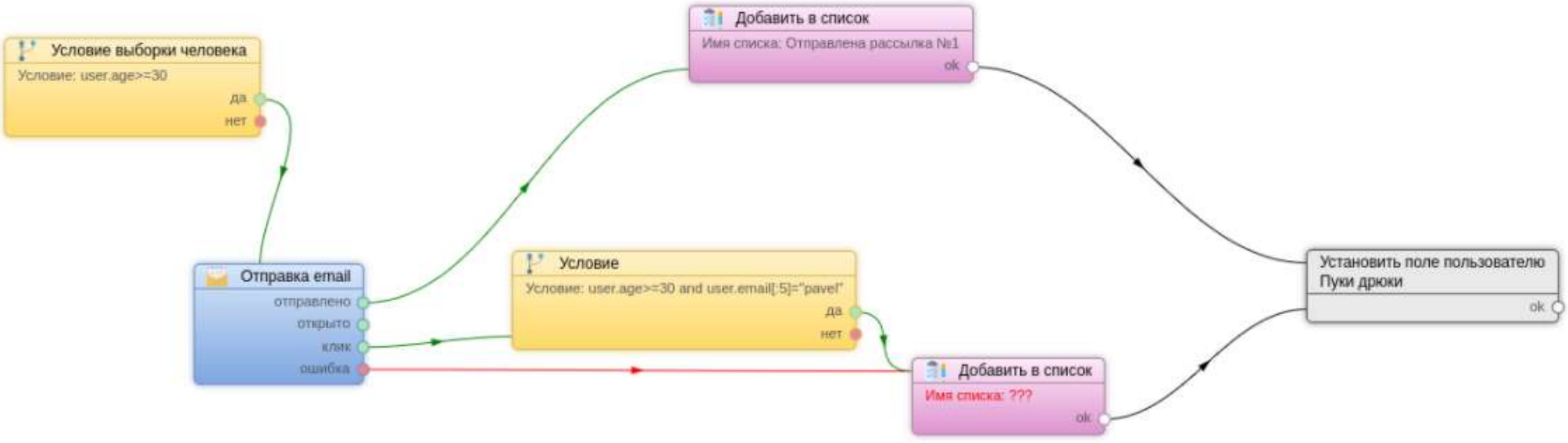
Также нужно понимать что принимаются предложения и пожелания, этот документ определяет вектор движения, а не жесткое требование.

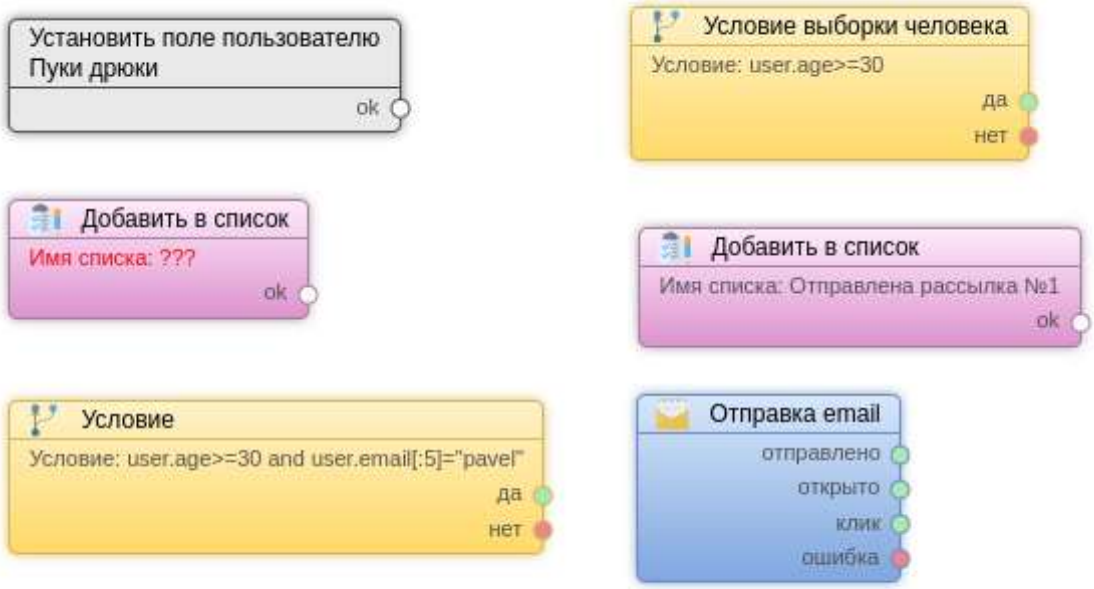
Графическое дополнение к заданию:

Потенциальная идея отображения блоков:



Пример реализации подобной идеи в другом проекте:





Результат:

Код в git репозитории (данные для доступа будут предоставлены), красиво оформленный, снабженный достаточными комментариями на русском языке (кол-во комментариев регулируется здравым смыслом).

Также должны быть предоставлены все необходимые демонстрации.
Для сборки демонстрационного проекта необходимо предоставить все скрипты и документацию по запуску сборки. Документация уровня для умелого админа.
Для сборки используем npm.
Красивое оформление кода и демонстраций приветствуется.