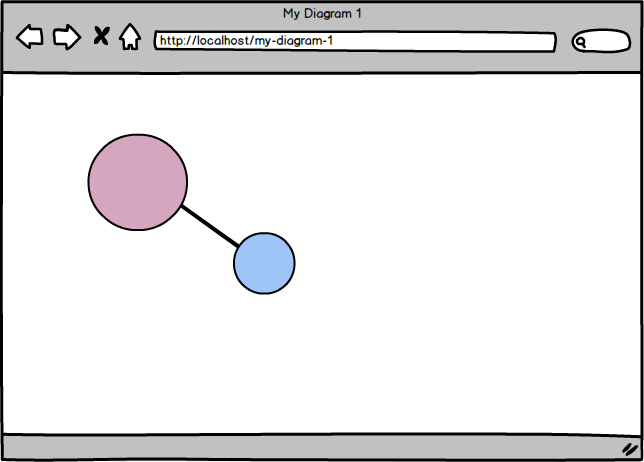
Идея проекта – создание небольшого редактора диаграмм.

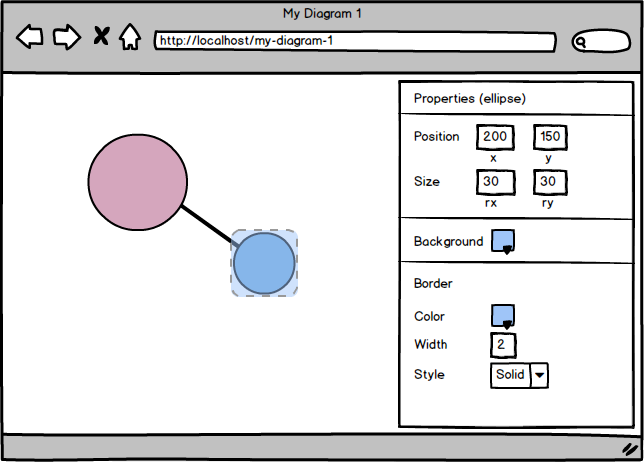
1. Открытие и показ диаграммы

* Приложение должно уметь показывать диаграммы
* Диаграммы состоят из компонентов и хранятся в формате JSON (пример ниже)
* Диаграммы хранятся на сервере (каким-то образом); приложение запрашивает диаграмму у сервера по id, указанному в url (e.g. http://localhost#my-diagram-1)
* Начальный набор компонентов, который поддерживает редактор: линия и эллипс
* Свойства, которые можно сконфигурировать у каждого из компонентов:
  + Линия: начальная и конечная точки, цвет, толщина, стиль (solid/dashed/dotted)
  + Эллипс: позиция, горизонтальный и вертикальный радиус, цвет фона, внешний вид границы (цвет, толщина, стиль)



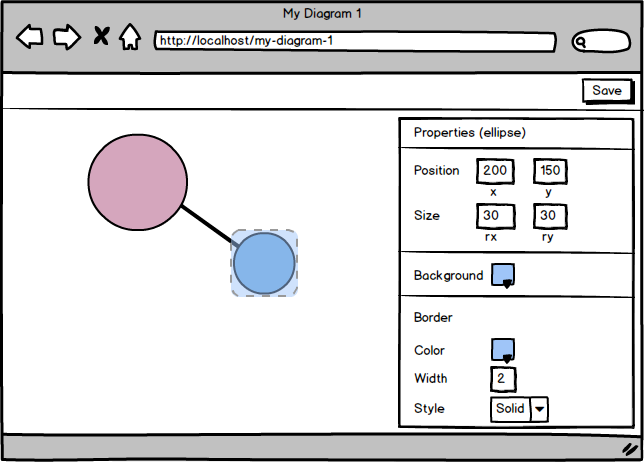
2. Редактирование свойств компонентов

* Приложение должно позволять редактировать свойства компонентов
* Возможноть выбрать компонент на диаграмме; выбранный компонент должен визуально выделяться
* Редактор свойств (отдельное представление) позволяет редактировать свойства выбранного компонента



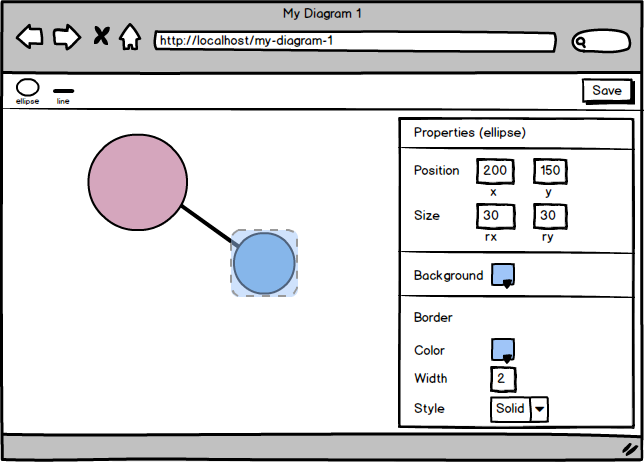
3. Сохранение диаграммы

* Приложение должно позволять сохранить отредактированную диаграмму
* Кнопка "Save" в тулбаре
* По нажатию кнопки приложение просит сервер сохранить новое состояние диаграммы



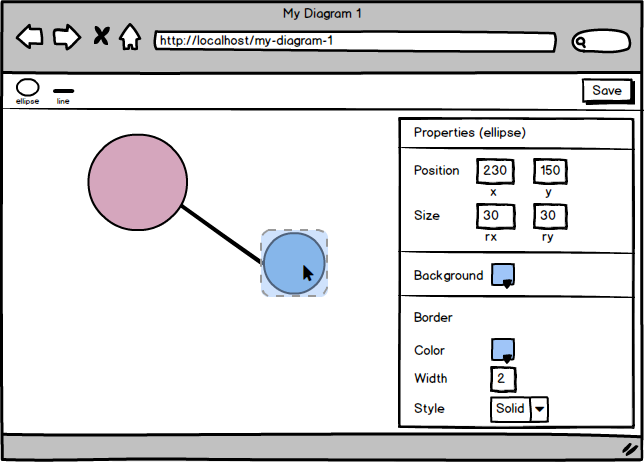
4. Добавление новых компонентов

* Приложение должно позволять добавлять новые компоненты на диаграмму
* Библиотека компонентов (отдельное представление, показывать в тулбаре)
* Возможность добавить компонент из библиотеки на диаграмму (либо drag-and-drop, либо выбрать компонент в библиотеке и кликнуть в нужное место на диаграмме)
* Новый компонент создаётся с дефолтными свойствами



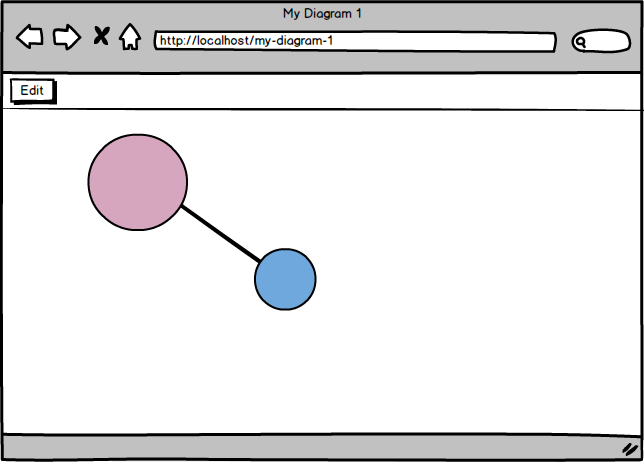
5. Перемещение компонентов с помощью мыши

* Приложение должно позволять перемещать компоненты на диаграмме с помощью мыши
* Изменение положения компонента должно сразу же применяться к соответствующим свойствам ("x" и "y" у эллипса, "start" и "end" у линии)



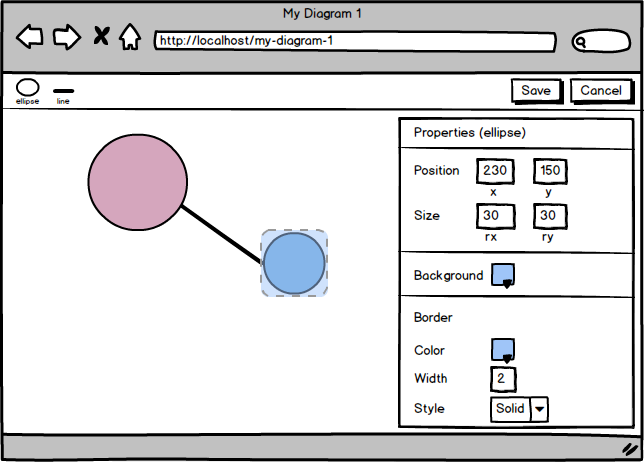
6. Режим просмотра

* Приложение должно открывать диаграммы в режиме просмотра
* В режиме просмотра приложение не позволяет изменить диаграмму
* Кнопка "Edit" в тулбаре переводит приложение в режим редактирования
* В режиме редактирования приложение позволяет изменять диаграмму, в тулбаре появляется кнопка "Save"
* Успешное сохранение диаграммы переводит приложение в режим просмотра



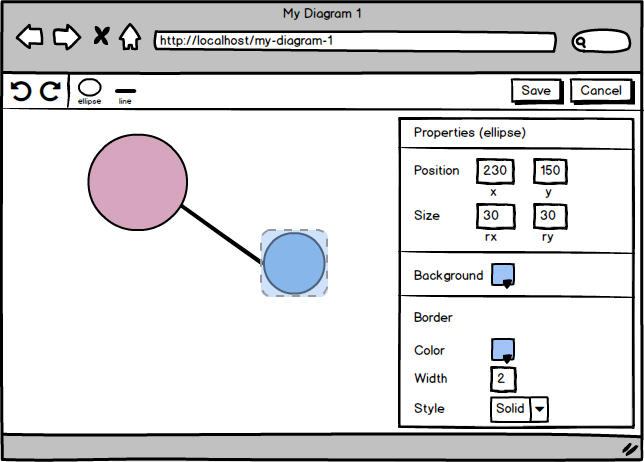
7. Отмена изменений

* Приложение должно позволять отказаться от сделанных изменений
* Кнопка "Cancel" в тулбаре
* По нажатию кнопки приложение "забывает" все сделанные пользователем изменения с момента последнего сохранения / открытия и переходит в режим просмотра



8. Undo / Redo

* Приложение должно позволять отменить недавние изменения или заново применить их
* Кноки undo / redo неактивны, если соответствующее действие невозможно
* Можно отменять произвольное число изменений до последнего сохранения или открытия диаграммы



9. Компонент Rectangle

* Приложение должно уметь рисовать прямоугольники
* Новый компонент rectangle в библиотеке
* Свойства компонента: позиция (x, y), размер (height, width), цвет фона, внешний вид границы (цвет, толщина, стиль)
* В идеале добавление (и удаление) компонента в библиотеку не должно затрагивать код остального приложения, т.е. компонент отвечает за специфичные для него вещи (внешний вид и поведение, набор свойств)

Загрузка и сохранение диаграмм должны происходить без перезагрузки страницы, приложение отправляет асинхронные запросы на сервер и обновляет своё состояние в соответствии с ответом сервера.

Сервер предоставляет api для работы с ресурасами, например:

GET /api/diagrams/1

- получение диаграммы, в ответ сервер отдает диаграмму с идентификатором "1"

PUT /api/diagrams/1

- сохранение диаграммы с идентификатором "1", в теле запроса передаётся новое состояние диаграммы

В коде состояние диаграммы должно быть отделено от представления диаграммы.

Рекомендуемые инструменты:

* git: управление исходным кодом
* node.js, express.js: сервер
* npm: менеджер пакетов node
* bower: менеджер фронт-энд пакетов
* require.js: загрузчик фронт-энд модулей
* backbone+marionette
* promises (e.g. bluebird): обработка асинхронных операций (например, обработка ответов сервера)
* svg, d3.js: рисование компонентов

Пример описания диаграммы:

{

    "title": "My diagram 1",

    "components": [

        {

            "type": "ellipse",

            "properties": {

                "x": 50,

                "y": 40,

                "rx": "50",

                "ry": "50",

                "fill": "#BDBBB8",

                "stroke-color": "#0000ff",

                "stroke-width": "2",

                "stroke-style": "solid"

            }

        },

        {

            "type": "line",

            "properties": {

                "start": {

                    "x": "100",

                    "y": "90"

                },

                "end": {

                    "x": "200",

                    "y": 160

                },

                "stroke-color": "#000000",

                "stroke-width": "2",

                "stroke-style": "solid"

            }

        },

        {

            "type": "ellipse",

            "properties": {

                "x": 200,

                "y": 150,

                "rx": 30,

                "ry": 30,

                "fill": "#ff0000",

                "stroke-color": "#0000ff",

                "stroke-width": "2",

                "stroke-style": "solid"

            }

        }

    ]

}