**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

**по дисциплине «Введение в нереляционные базы данных»**

**Тема: БД вакансий**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 8383 |  | Ишанина Л.Н. |
| Студент гр. 8383 |  | Ларин А. |
| Студентка гр. 8383 |  | Сырцова Е.А. |
| Преподаватель |  | Заславский М.М. |

Санкт-Петербург

2021

**ЗАДАНИЕ**

**НА ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ**

|  |  |  |
| --- | --- | --- |
| Студенты Ишанина Л.Н., Ларин А., Сырцова Е.А. | | |
| Группа 8383 | | |
| Тема проекта: БД вакансий | | |
| Исходные данные:  Веб-инструмент для CRUD вакансий, а также продвинутой аналитики (включая графовое представление). | | |
| Содержание пояснительной записки:  «Содержание», «Введение», «Качественные требования», «Сценарий использования», «Модель данных», «Разработанное приложение», «Заключение» | | |
| Предполагаемый объем пояснительной записки:  Не менее 10 страниц. | | |
| Дата выдачи задания: 13.09.2021 | | |
| Дата сдачи реферата: 28.12.2021 | | |
| Дата защиты реферата: 28.12.2021 | | |
| Студентка гр. 8383 |  | Ишанина Л.Н. |
| Студент гр. 8383 |  | Ларин А. |
| Студентка гр. 8383 |  | Сырцова Е.А. |
| Преподаватель |  | Заславский М.М. |

**АННОТАЦИЯ**

В рамках данного курса предполагалось разработать какое-либо приложение в команде на одну из поставленных тем. Была выбрана тема создания БД вакансий hh.ru для СУБД Neo4j. Во внимание будут приниматься такие аспекты как производительность и удобство разработки. Найти исходный код и всю дополнительную информацию можно по ссылке: https://github.com/moevm/nosql2h21-jobs

**SUMMARY**

As part of this course, it was supposed to develop an application in a team on one of the set topics. The topic of creating a database of vacancies was chosen hh.ru for the Neo4j DBMS. Aspects such as performance and ease of development will be taken into account. You can find the source code and all additional information at the link: https://github.com/moevm/nosql2h21-jobs

**содержание**

|  |  |  |
| --- | --- | --- |
|  | Введение | 5 |
| 1. | Качественные требования к решению | 6 |
| 2. | Сценарии использования | 7 |
| 2.1. | Макет пользовательского интерфейса | 7 |
| 2.2. | Описание сценариев использования | 7 |
| 3. | Модель данных | 15 |
| 3.1. | Список сущностей | 15 |
| 3.2.  3.3.  3.4.  3.5.  4.  4.1.  4.1.1.  4.1.2.  4.2.  4.3.  4.4. | Оценка удельного объема информации, хранимой в модели  Запросы к модели, с помощью которых реализуются сценарии использования  Аналоги модели данных для SQL СУБД  Модель данных для MongoDB  Разработанное приложение  Краткое описание  Архитектура  Docker  Схема экранов приложения  Использованные технологии  Ссылки на приложение | 17  19  20  23  26  26  26  26  26  28  28 |
|  | Заключение | 29 |

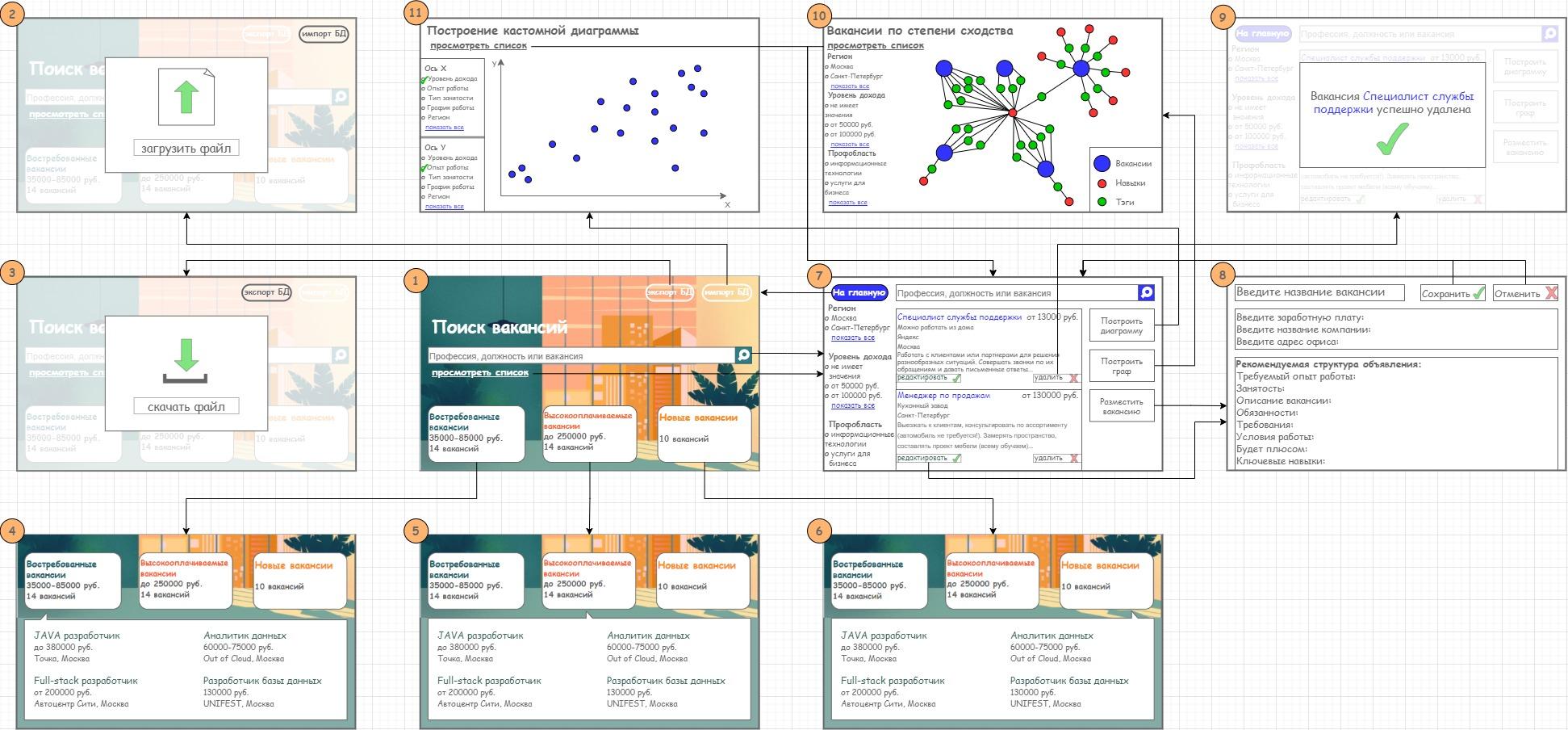
**ВВЕДЕНИЕ**

Цель работы – создать веб-инструмент для CRUD вакансий, а также продвинутой аналитики, включая графовое представление.

1. **Качественные требования к решению**

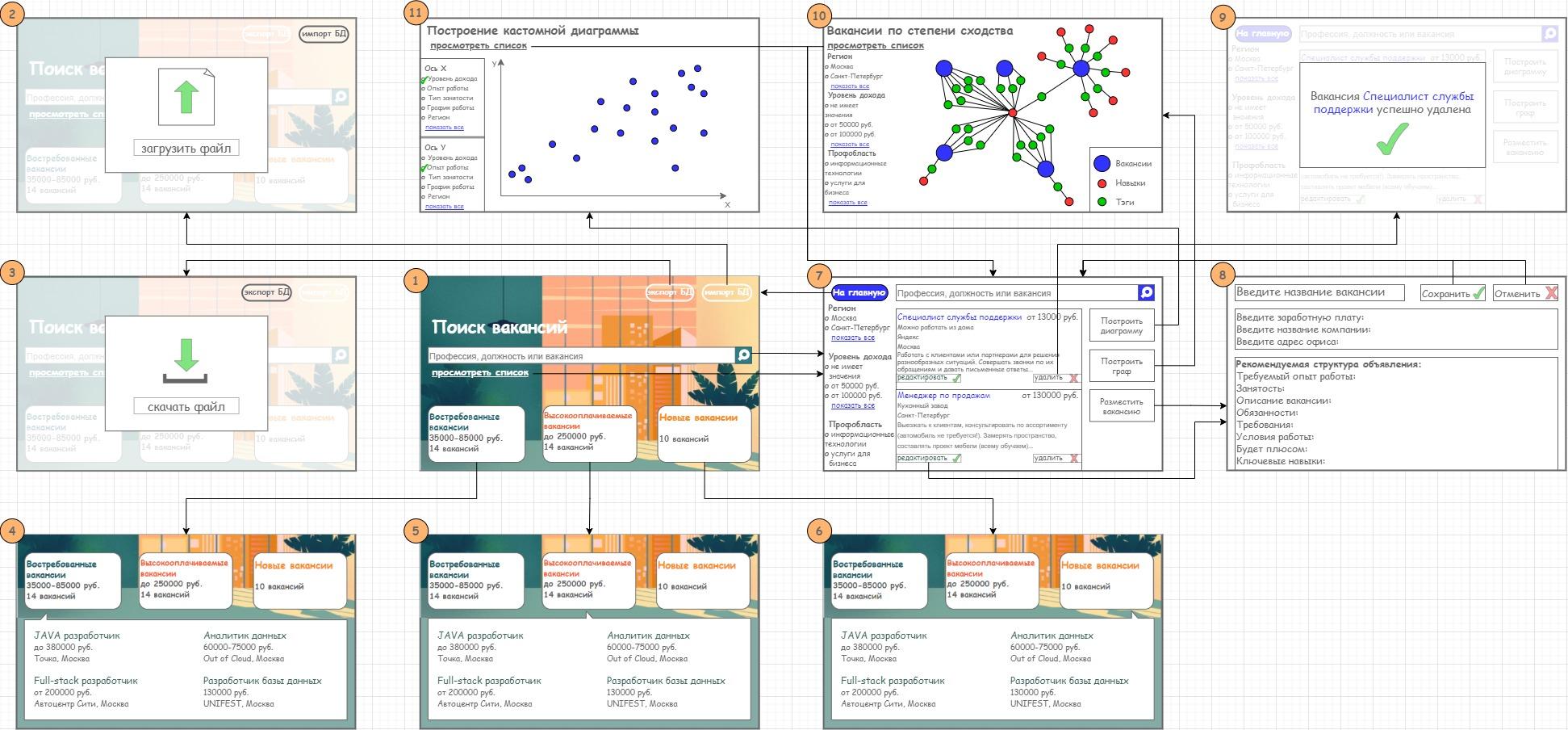
Требуется разработать веб-инструмент для CRUD вакансий, а также продвинутой аналитики, включая графовое представление. Приложение содержит страницу с табличным представлением списка вакансий и поиском, страницу для визуализации содержимого БД кастомной диаграммой, страницу массового импорта/экспорта всей БД, страницу графов по степени сходства вакансий. Возможная аналитика – самые востребованные/оплачиваемые/новые вакансии.

1. **Сценарий использования**
   1. **Макет пользовательского интерфейса**

****

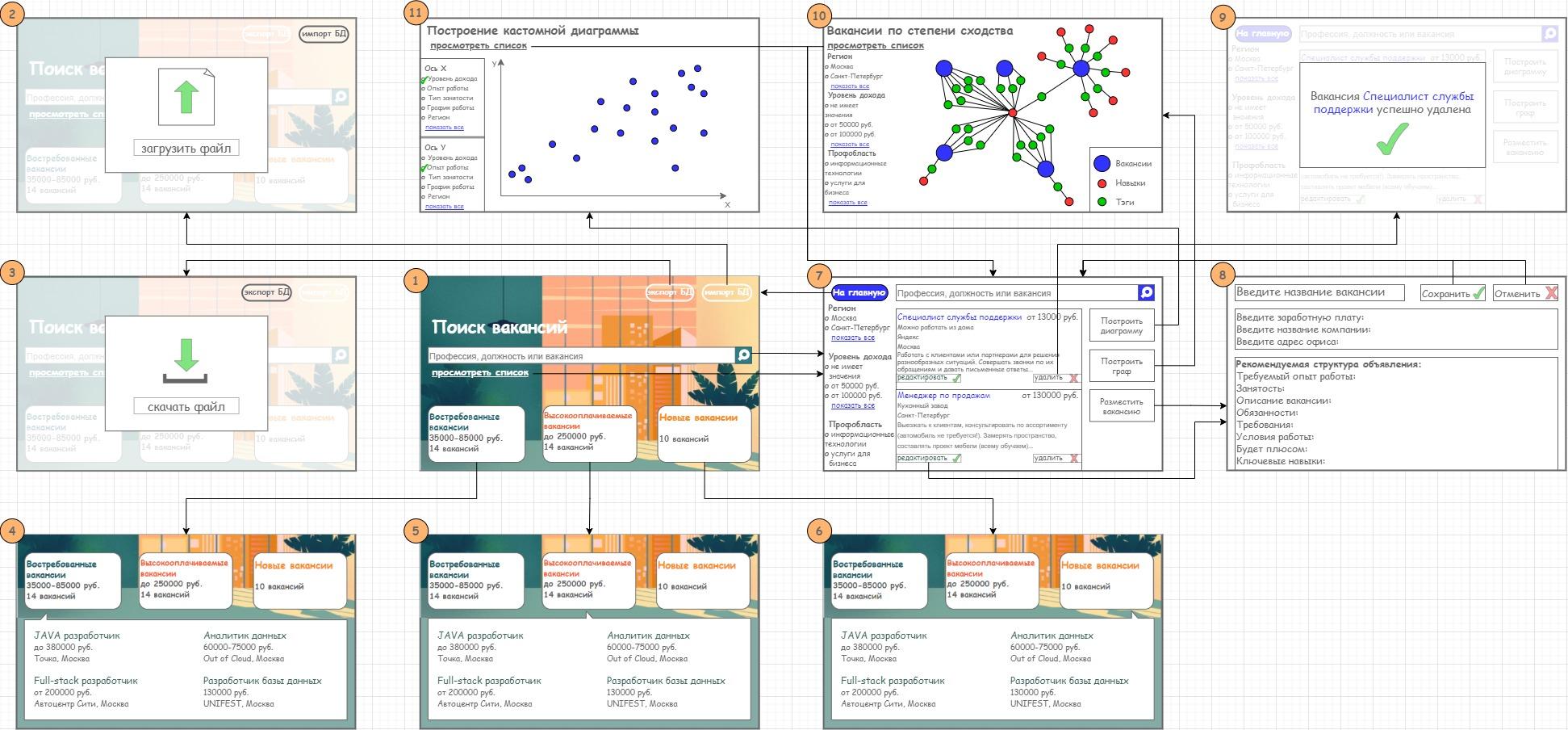
* 1. **Описание сценариев использования**

Единственная роль в системе - администратор. Пользователь входит на страницу (1). Он видит страницу поиска вакансий, в правом верхнем углу кнопки массового экспорта и импорта БД, внизу страницы некоторые элементы аналитики.

****

Для того чтобы **загрузить** базу данных, пользователь должен:

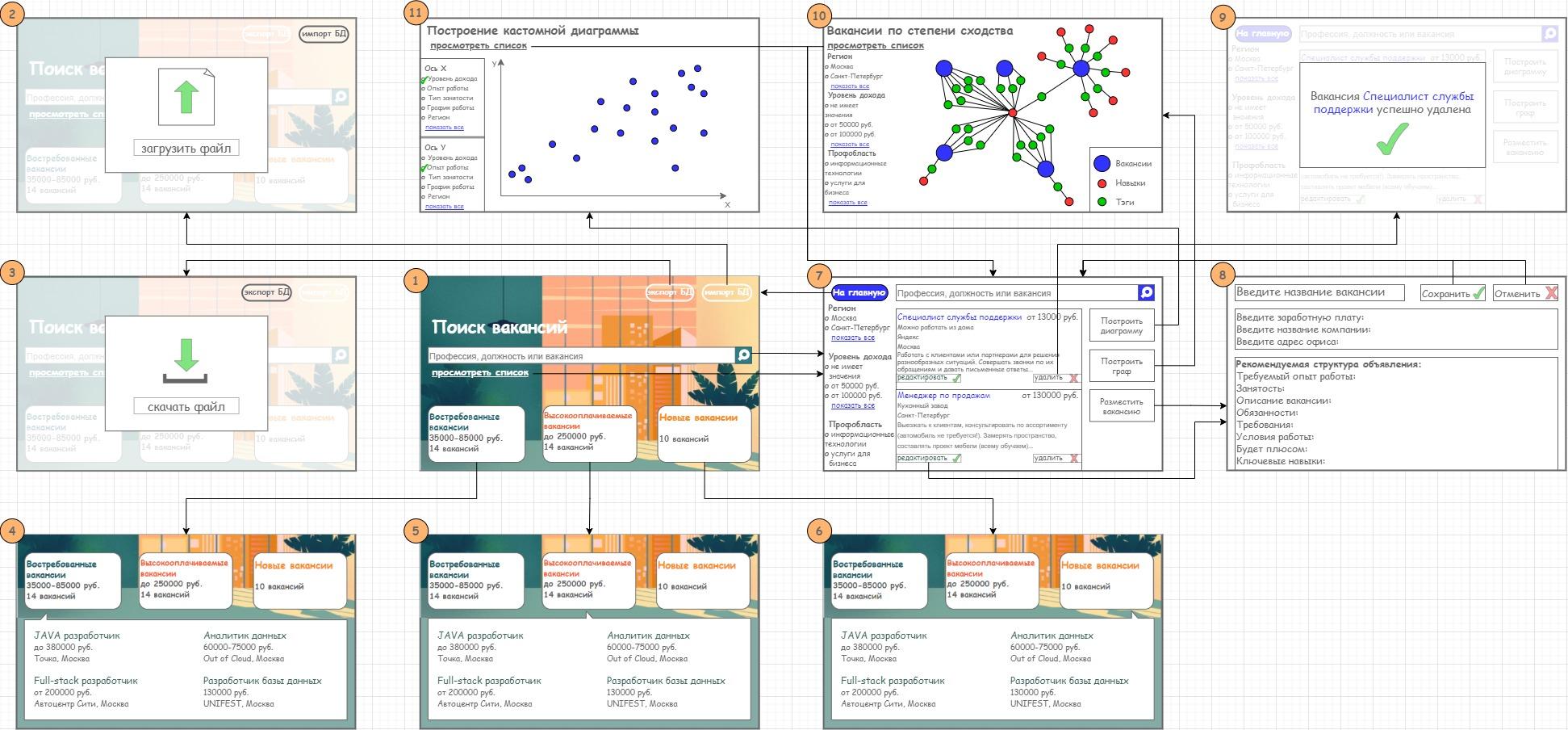
1. нажать на кнопку “импорт БД”
2. в открывшемся окне (2) нажать на кнопку “загрузить файл”

****

1. выбрать файл на компьютере и нажать на кнопку “загрузить”
2. подождать, пока “загрузка файла” не сменится на “база данных успешно загружена”
3. если “загрузка файла” сменилось на “ошибка загрузки базы данных”, то попробовать загрузку в другое время.

Для того чтобы **скачать** базу данных, пользователь должен:

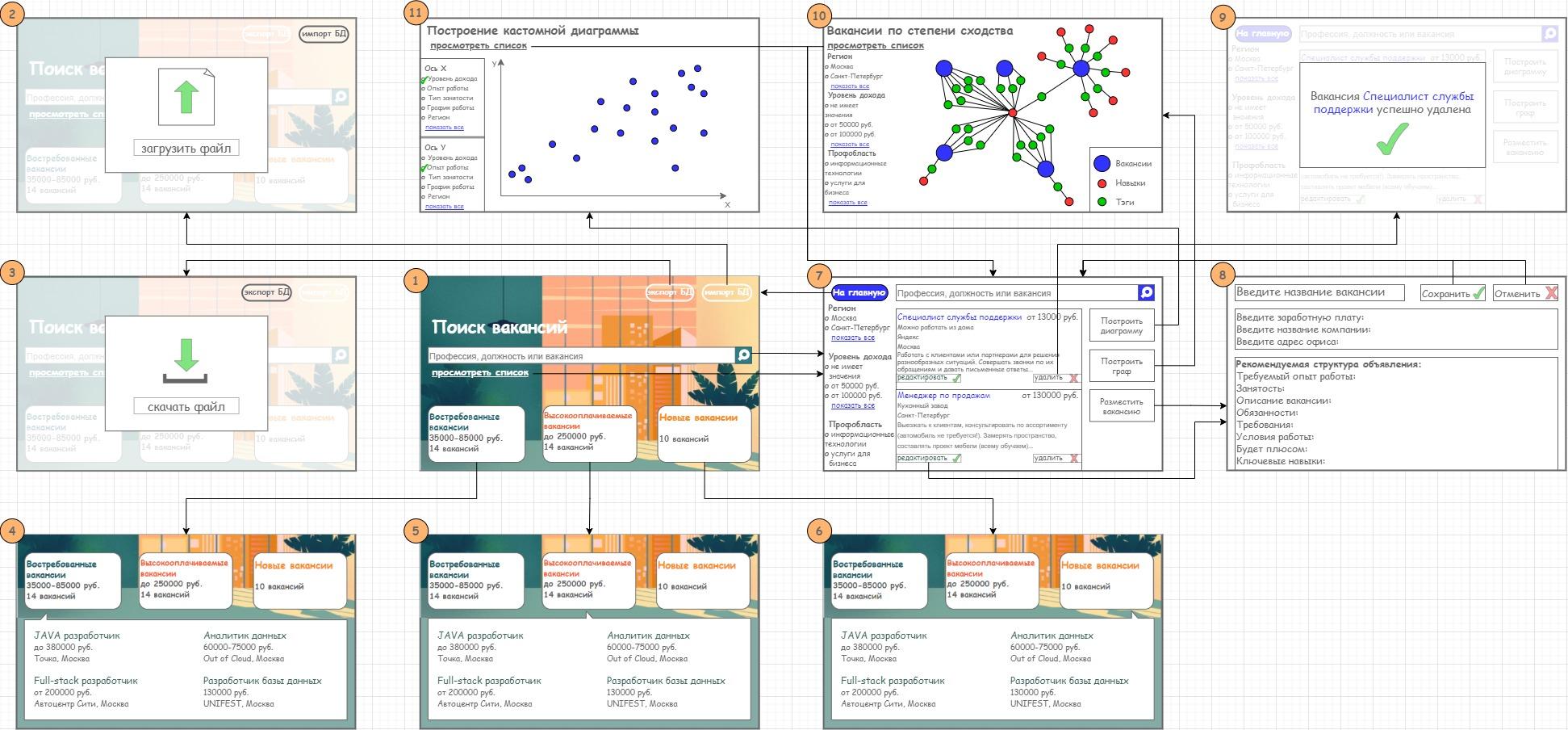
1. нажать на кнопку “экспорт БД”
2. в открывшемся окне (3) нажать на кнопку “скачать файл”

****

1. выбрать место скачивания на компьютере и нажать на кнопку “скачать”
2. подождать, пока “скачивание файла” не сменится на “база данных успешно скачана”
3. если “скачивание файла” сменилось на “ошибка скачивания базы данных”, то попробовать экспорт БД в другое время.

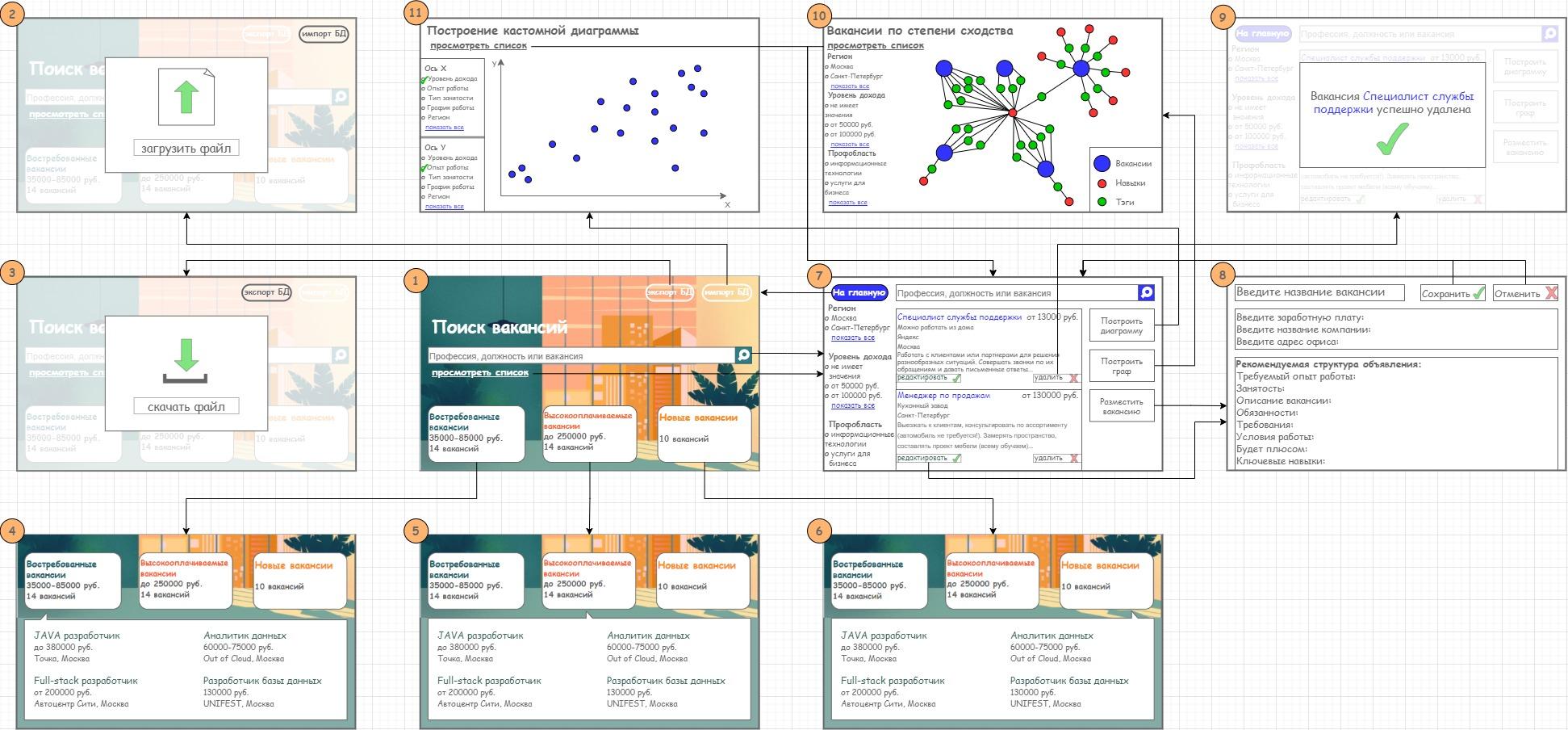
Для того чтобы **просмотреть список самых востребованных вакансий**, пользователь должен:

1. нажать на баннер “Востребованные вакансии”
2. в открывшейся таблице (4) представлен список самых востребованных вакансий

****

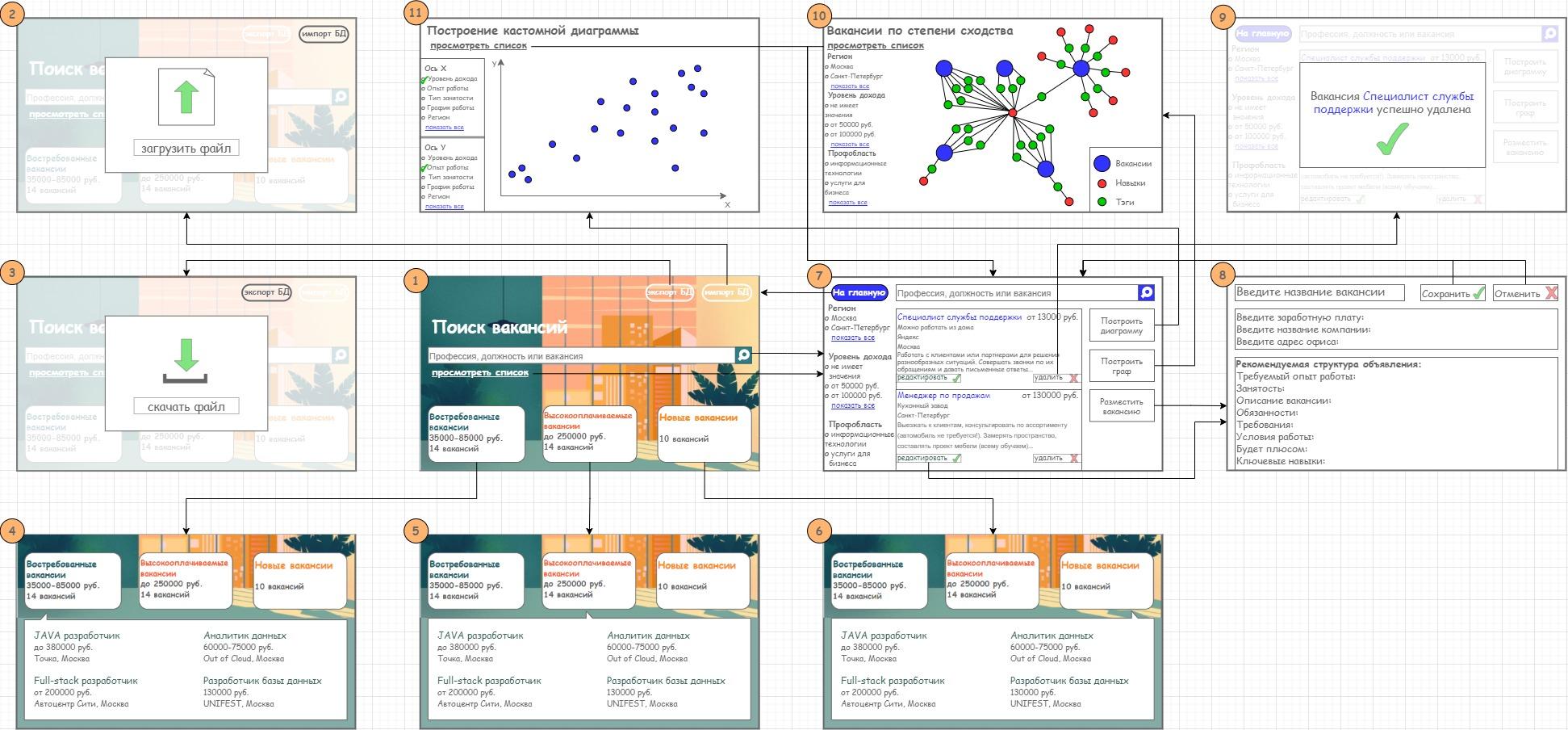
Для того чтобы **просмотреть список самых высокооплачиваемых вакансий**, пользователь должен:

1. нажать на баннер “Высокооплачиваемые вакансии”
2. в открывшейся таблице (5) представлен список самых высокооплачиваемых вакансий

****

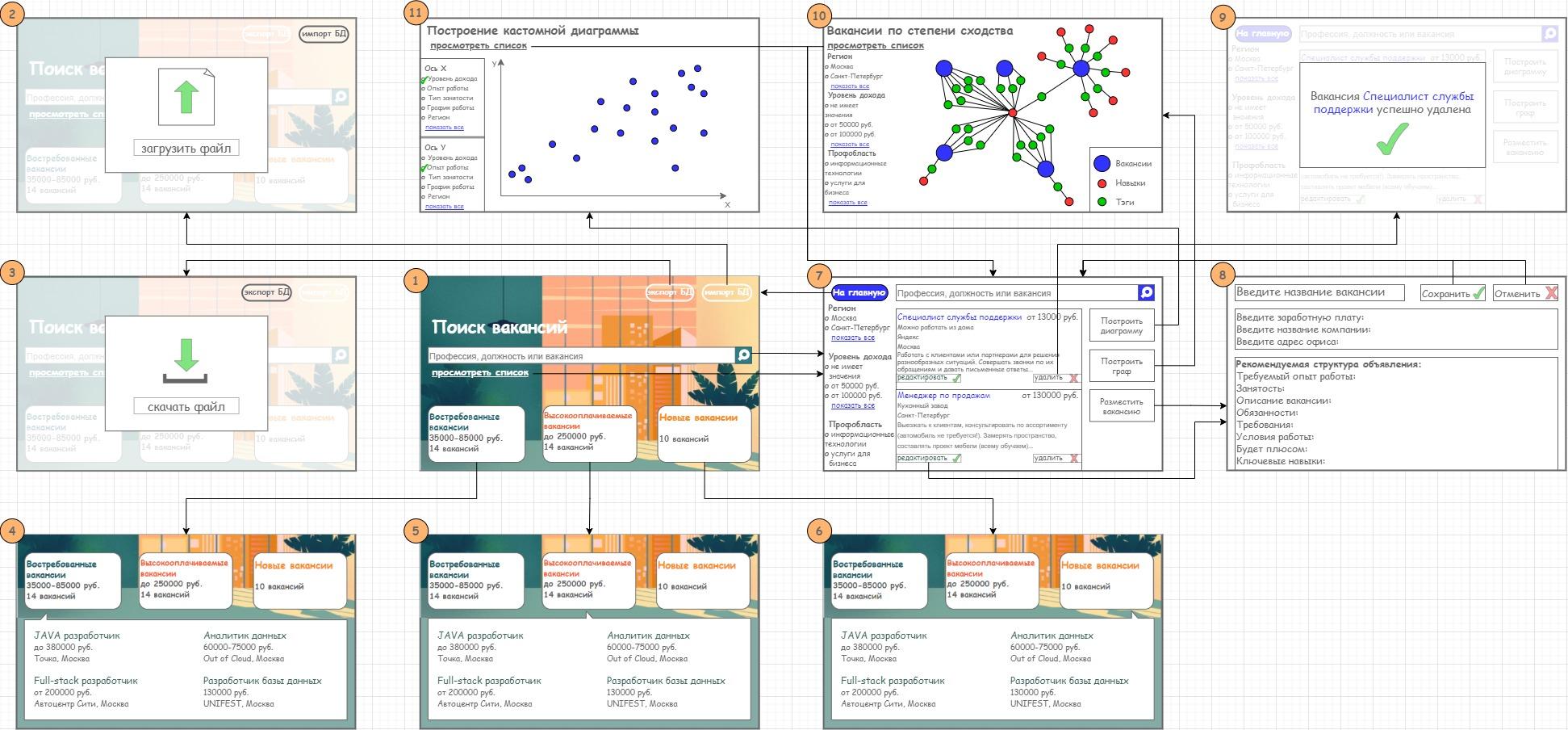
Для того чтобы **просмотреть список новых вакансий**, пользователь должен:

1. нажать на баннер “Новые вакансии”
2. в открывшейся таблице (6) представлен список самых новых вакансий

****

Для того чтобы **просмотреть список вакансий**, пользователь должен:

1. нажать на кнопку “просмотреть список”
2. в открывшемся окне (7) представлен список вакансий БД

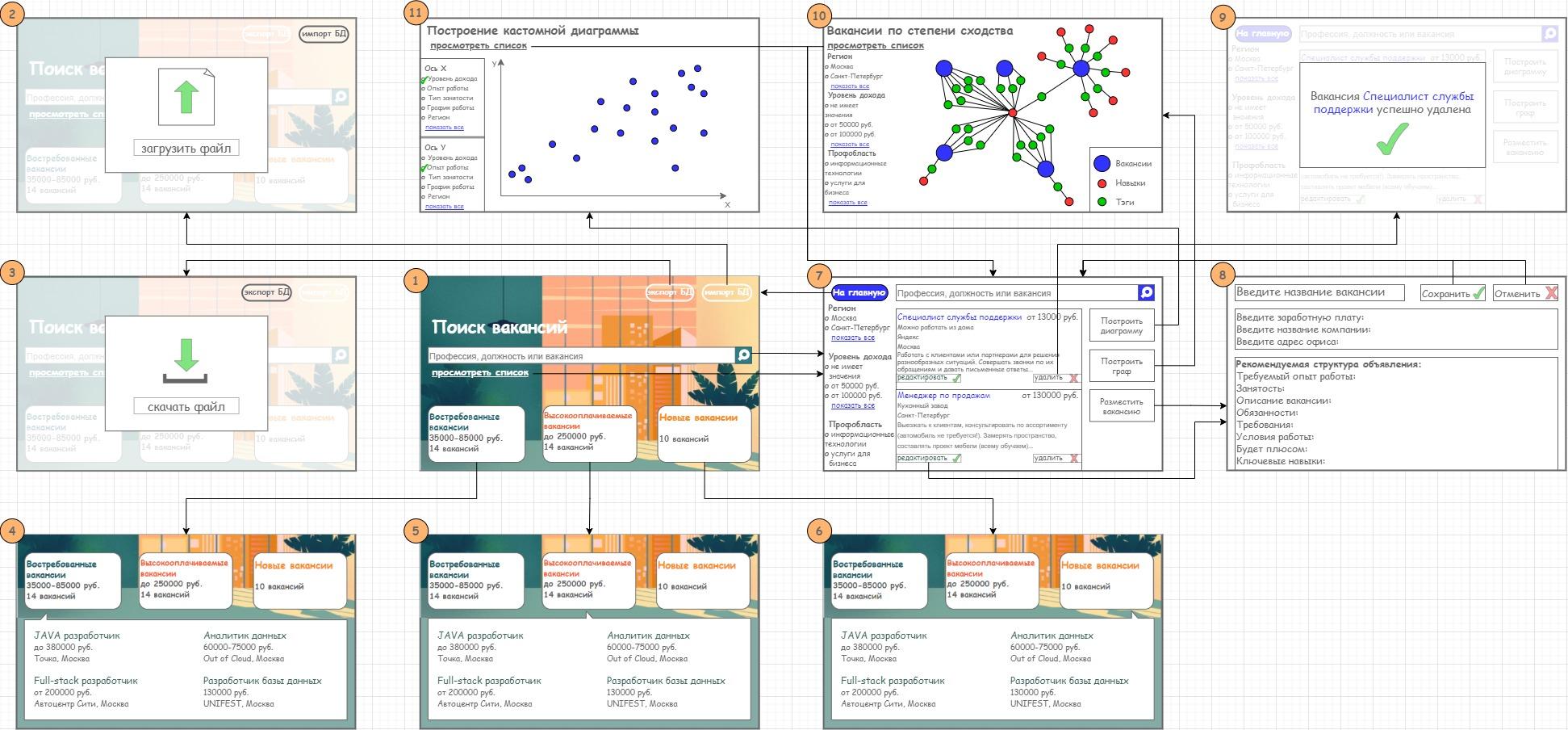
****

Для того чтобы осуществить **поиск вакансий**, пользователь должен:

1. ввести в строке поиска (в окне 1 или окне 7) запрос (указать профессию, должность или вакансию)
2. нажать на кнопку “лупа”
3. в открывшемся окне (7) представлен список найденных вакансий
4. дополнительно в окне 7 слева можно выбрать кластеры, список автоматически обновиться

Для того чтобы **отредактировать/разместить вакансию**, пользователь должен:

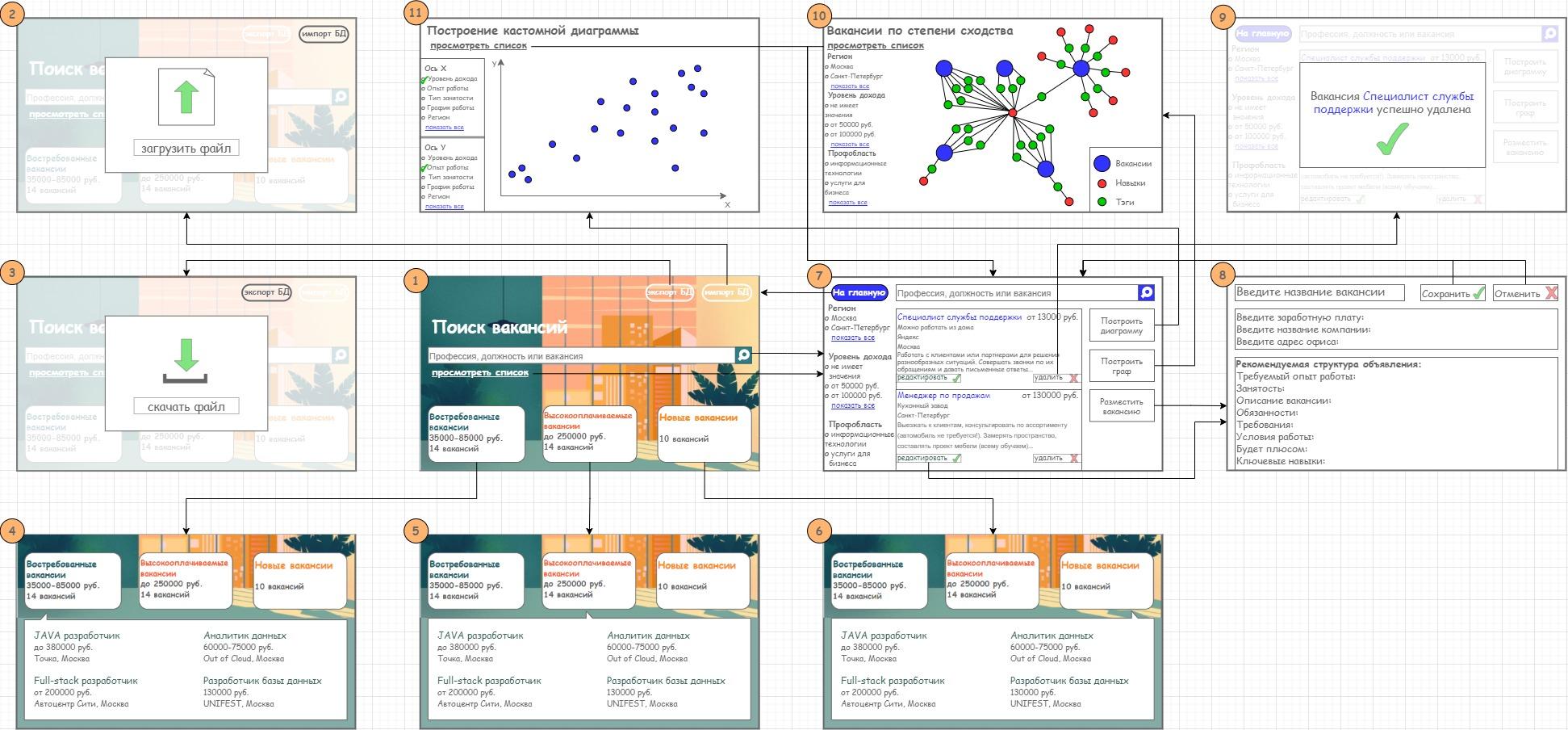
1. нажать на кнопку “редактировать” для выбранной вакансии/”Разместить вакансию”
2. в открывшемся окне (8) изменить нужные поля (Название вакансии, Заработная плата, Название компании, Адрес офиса, Требуемый опыт работы, Занятость, Описание вакансии, Обязанности, Требования, Условия работы, Будет плюсом, Ключевые навыки)

****

1. нажать на кнопку “Сохранить”, пользователь вернется к окну 7 с обновленным списком вакансий

Для того чтобы **удалить вакансию**, пользователь должен:

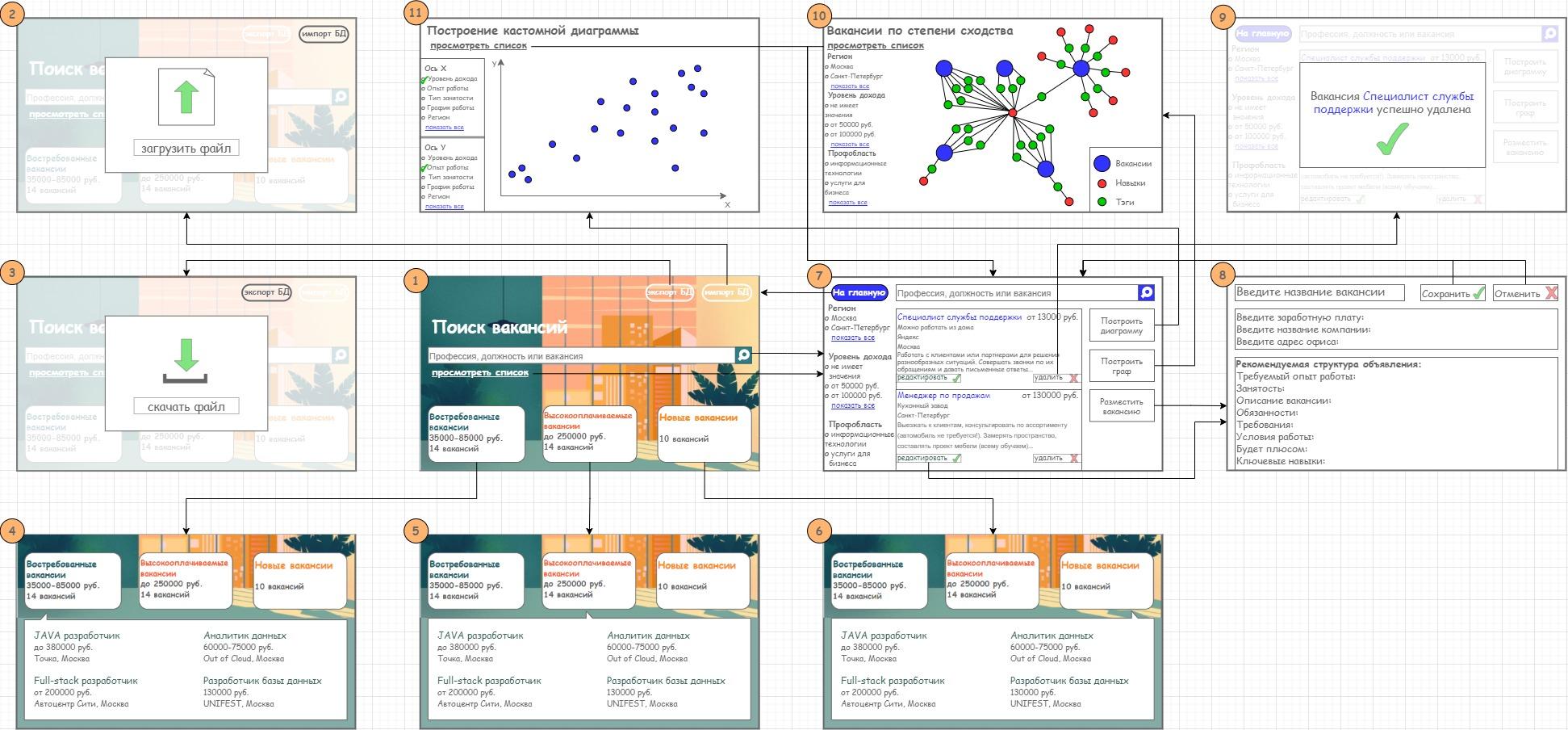
1. нажать на кнопку “удалить” для выбранной вакансии
2. в открывшемся окне 9 подождать пока “удаление вакансии” не сменится на “Вакансия успешно удалена”

****

1. если “удаление вакансии” сменилось на “ошибка удаления вакансии” то попробовать удаление в другое время

Для того чтобы **построить граф вакансий по степени сходства**, пользователь должен:

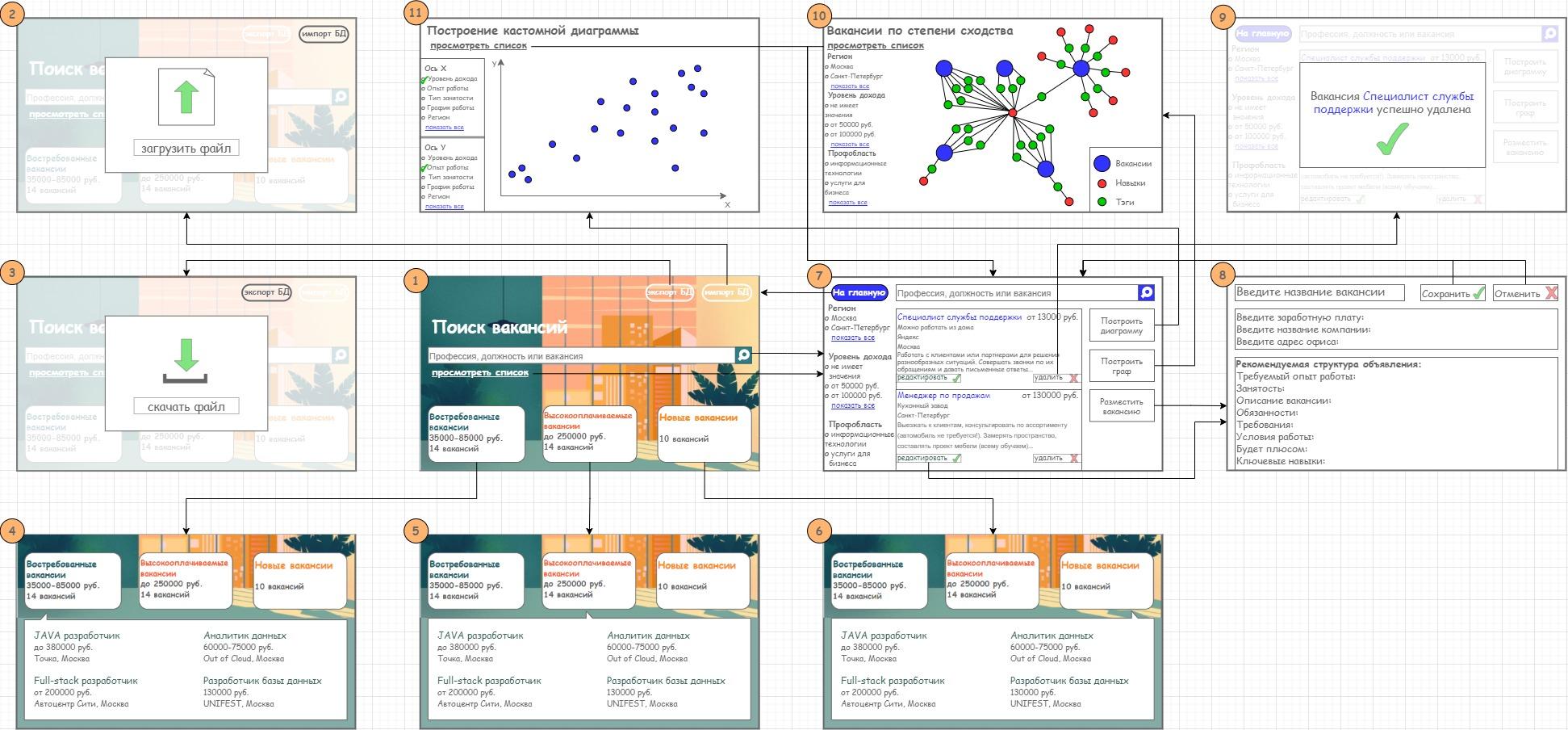
1. нажать на кнопку “Построить граф” в окне 7
2. в открывшемся окне 10 можно дополнительно изменить кластеры, определяющие вакансии в построении графа

****

1. граф отображается автоматически
2. для того, чтобы вернуться к списку вакансий, пользователь должен нажать на кнопку “просмотреть список”

Для того чтобы **построить кастомную диаграмму**, пользователь должен:

1. нажать на кнопку “Построить диаграмму” в окне 7
2. в открывшемся окне 11 выбрать поля, задающие оси X и Y в списке слева

****

1. диаграмма отображается автоматически
2. для того, чтобы вернуться к списку вакансий, пользователь должен нажать на кнопку “просмотреть список”

**3. Модель данных**

* 1. **Список сущностей**

В терминологии neo4j данные сущности представляют из себя вершины, а их названия - метки(lable)

**Area** - зона, в которой располагается вакансия.

{

"id": "integer",

"name": "String"

}

**Country** - подмножество Area: зоны, являющиеся странами  
Зоны и страны образуют древовидный граф

**Currency** - валюта

{

"name": "String"

}

Например: RUB, EUR

**Employer** - работодатель, указывается для всех вакансий, кроме анонимных

{

"id": "integer",

"name": "String"

}

**Schedule** - график работы

{

"id": "String",

"name": "String"

}

Гибкий, полный, и т.д.

**Vacancy\_type** - тип вакансии

{

"id": "String",

"name": "String"

}

Открытая, закрытая, анонимная

**Vacancy** - нода вакансии

{

"id": "integer", // Название

"name": "String", // Название

"has\_test": "boolean", // Есть ли тестовое

"salary\_from": "integer", // Нижняя граница зарплаты

"salary\_to": "integer", // Верхняя граница зарплаты

"published\_at": "String(DateTime)", // Время публикации вакансии

"created\_at": "String(DateTime)", // Время создания вакансии

"requirement": "String", // Требования

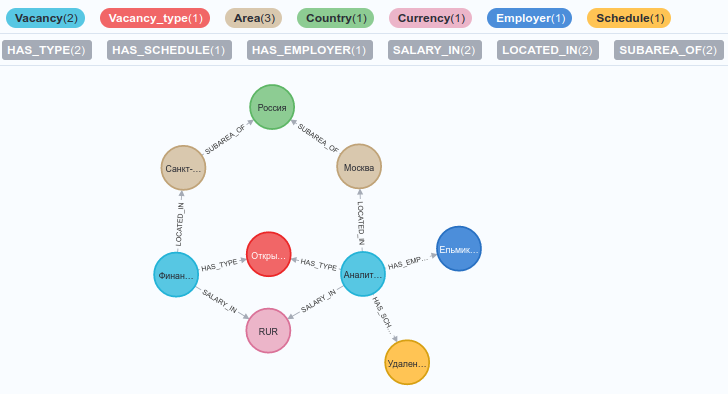
"responsibility": "String" // Ответственность

}

Список связей (дуг/рёбер)

|  |  |  |
| --- | --- | --- |
| Метка | Из | В |
| **LOCATED\_IN** | Vacancy | Area/Country |
| **SALARY\_IN** | Vacancy | Currency |
| **HAS\_EMPLOYER** | Vacancy | Employer |
| **HAS\_SCHEDULE** | Vacancy | Schedule |
| **HAS\_TYPE** | Vacancy | Vacancy\_type |
| **SUBAREA\_OF** | Area | Area/Country |

Графическое представление, здесь для примера представлены две вакансии – в Москве (Аналитик) и СПб (Финансист). У вакансии из СПб не указан работодатель и график работы.



* 1. **Оценка удельного объема информации, хранимой в модели**

Размеры различных типов данных представлены в документации. Размеры строки приняты за *S* и оценены по верхней границе в 255 байт (размер строк фиксированной длины по умолчанию во многих БД). У всех объектов есть внутренний *ID* размером 8 байт, который имеет смысл учитывать в "чистом" объеме для характеристики ребер. Имеется 7 + 6 различных меток. Положим 1 байт на задание метки объекта.

* Связи  
  Связи (рёбра) характеризуются *ID* и двумя вершинами, а так же меткой.  
  3\*8 + 1 = 25 байт
* Area/Country  
  8 + 4 + *S* + 1 = 368 байт
* Currency  
  Валюта всегда занимает три символа  
  8 + 3 + 1 = 12 байт
* Employer  
  8 + 4 + *S* + 1 = 368 байт
* Schedule  
  8 + 2\**S* + 1 = 519 байт
* Vacancy\_type  
  8 + 2\**S* + 1 = 519 байт
* Vacancy  
  8 + 4 + *S* + 1 + 4 + 4 + 4\**S* = 1296 байт

Т.е. для БД представленной на рис. в п. "Графическое представление" выше чистый объем составит: 3 \* Area + Currency + Employer + Schedule + Vacancy\_type + 2 \* Vacancy + 10 \* Edge = 5364 Байт

Рост объёма от числа вакансий

В худшем случае *N* = #Area = #Currency = #Employer = #Schedule = #Vacancy\_type

#Area = #edges = 2*N*

На практике с ростом значений *N*, количество этих объектов будет стремиться к константе

Объем информации в худшем случае:

*V* = *N* \* Area + *N* \* Currency + *N* \* Employer + *N* \* Schedule + *N* \* Vacancy\_type + *N* \* Vacancy + 2*N* \* Edge

Для *N* = 2 оценка *V* = 5600

Избыточность модели

Чистый объем, посчитан в пред. пункте: 5364 байт = 5.24 kB Практический, получен: 1MB.

Избыточности в модели нет.

Направление роста модели при увеличении количества объектов каждой сущности.

Количество всех объектов зависит от числа вакансий, т.е. объект работодателя появляется в базе только если появляется вакансия с новым работодателем.

* Area, Country - Количество объектов этого типа не превосходит число вакансий и растет в зависимости от числа вакансий не быстрее чем линейно. Так же общее количество регионов ограничено (~7000). При малом количестве данных рост можно оценить как линейный, а при больших - как константный.
* Currency - Количество Валют не превосходит 10, потребление памяти можно оценить как константу.
* Employer - При увеличении количество вакансий растет линейно с большим коэффициентом, т.к. у работодателя может быть несколько вакансий. Рост можно оценить как *O*(*n*\*(*m*\*2)), где *n* - число работодателей, *m* - число вакансий, коэффициент излишен, но он показывает затраты на объекты рёбер от вакансий к работодателям.
* Schedule - Количество разновидностей графиков работы константно.
* Vacancy\_type - Количество разновидностей типов вакансий константно.
* Vacancy - При росте числа вакансий.
  1. **Запросы к модели, с помощью которых реализуются сценарии использования**

Создание вакансии:

CREATE (a:Vacancy)

SET a.id = '48621910'

SET a.name = 'Финансовый аналитик'

SET a.has\_test = False

SET a.published\_at = '2021-11-04T14:52:20+0300'

SET a.created\_at = '2021-11-04T14:52:20+0300'

SET a.requirement = 'Высшее профессиональное образование (финансы, экономика, математика). Уверенный пользователь ПК (MSOffice), отличное владение пакетом Excel. Уверенный/продвинутый пользователь 1С, БИТ.Финанс...'

SET a.responsibility = 'Формирование управленческой отчетности, сбор и аналитика данных. Настраивание процесса бюджетирования, бюджетного контроля и управления. Построение управленческой отчетности для пользователей внутри...'

Связь вакансии с регионом:

MATCH (a:Vacancy {id:'48621910'}),

(b:Area {id:'1')

MERGE (a)-[r:LOCATED\_IN]->(b)

Получить все вакансии в Москве c зарплатой от 50 тыс.

MATCH (v:Vacancy)-[e:LOCATED\_IN]->(r:Area{name:'Москва'})

WHERE (v.salary\_from>=50000)

RETURN v

Оценка числа операций для юзкейсов и CRUD

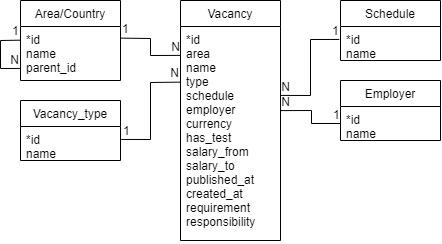
Благодаря богатому функционалу языка запросов Cypher, юзкейсы реализовываются в один запрос вне зависимости от числа объектов.  
Создание вакансии в худшем случае потребует создание ещё 6-ти сущностей (см. оценку роста модели), т.е. потребуется от 1 до 7 запросов в зависимости от ситуации и способа написания запроса (можно создавать несколько объектов в одном запросе, но получится громоздко и не обязательно быстрее).  
Для удаления вакансии потребуется два запроса - для удаления всех связей, а затем самой ноды.

Изменение параметров сущностей происходит за один запрос.

* 1. **Аналог модели данных для SQL СУБД**

Описанную модель данных можно представить в виде реляционной модели с помощью таблиц.

* Area/Country - множество всех зон расположения вакансии (страна/город) с идентификатором, названием и индексом зоны, подмножеством которой он является.
* Vacancy\_type - множество типов вакансий (открытая/закрытая/анонимная) с идентификатором и названием.
* Schedule - множество видов графика работы (гибкий/полный и т.д.) с идентификатором и названием.
* Employer - множество всех работодателей с идентификатором и названием.
* Vacancy - множество всех вакансий с идентификатором, названием, зоной расположения, типом вакансии, графиком работы, работодателем, валютой, верхней и нижней границей заработной платы, тестовым заданием, временем создания и публикации вакансии, требованиями и ответственностью.

****

Запросы к модели

Создание вакансии:

INSERT INTO Vacancy

VALUES (area, name, type, schedule, employer, currency, has\_test, salary\_from, salary\_to, published\_at, created\_at, requirement, responsibility);

Все вакансии в Москве c зарплатой от 50 тыс.

SELECT \* FROM Vacancy

WHERE Vacancy.salary\_from >= 50000 AND Vacancy.area = 'Москва';

Оценка числа операций для юзкейсов и CRUD

Удаление связано со сложностями выбора стратегии (setnull/cascade).

Добавление сущностей происходит за один запрос, но при добавлении вакансии может так же понадобится добавить до 6-ти других сущностей, соответственно ищет за 6 запросов.

Изменение информации происходит в один запрос.

Наибольшие трудности связаны с выборкой информации по многим фильтрам, т.к. для фильтрации по параметрам связных сущностей потребуется JOIN большого числа таблиц, соответственно породит большие запросы, которые трудно оптимизировать.

Оценка удельного объема информации, хранимой в модели

Положим размер *ID* 8 байт, размер строки *S* = 255 байт

* Area/Country  
  8 + *S* + 8 = 271
* Employer  
  8 + *S* = 263
* Schedule  
  2\**S* = 510
* Vacancy\_type  
  2\**S* = 510
* Vacancy  
  8 + 8 + *S* + 8 + 8 + 8 + 3 + 1 + 4 + 4 + 4\**S* = 1327

Т.е. для БД представленной на рис. в п. "Графическое представление" выше чистый объем составит: 3 \* Area + Employer + Schedule + Vacancy\_type + 2 \* Vacancy = 4750

Рост объёма от числа вакансий

В худшем случае *N* = #Area = #Employer = #Schedule = #Vacancy\_type

#Area = 2*N*

Объем информации в худшем случае: *V* = 2*N* \* Area + *N* \* Employer + *N* \* Schedule + *N* \* Vacancy\_type + *N* \* Vacancy

Для *N* = 2 оценка *V* = 6304

Избыточность модели

Чистый объем, посчитан в пред. пункте: 6304 байт = 6.3 kB

Практический, получен: 7MB

Избыточности в реляционной модели нет

Сравнение моделей

По памяти

В графовой модели чистый объем для заданных данных составил 5364 байт.

В реляционной для такой же конфигурации 4750. Экономия достигается в основном из-за отсутствия необходимости хранить рёбра, т.к. связь происходит через ключи.

По запросам

По запросам для создания информации графовая БД имеет теоретическое преимущество, т.к. может создавать цепочки сущностей за раз, что позволит делать меньше запросов при одновременном добавлении нескольких вакансий и вспомогательных объектов к ним.

По запросам для изменения информации большого различия нет

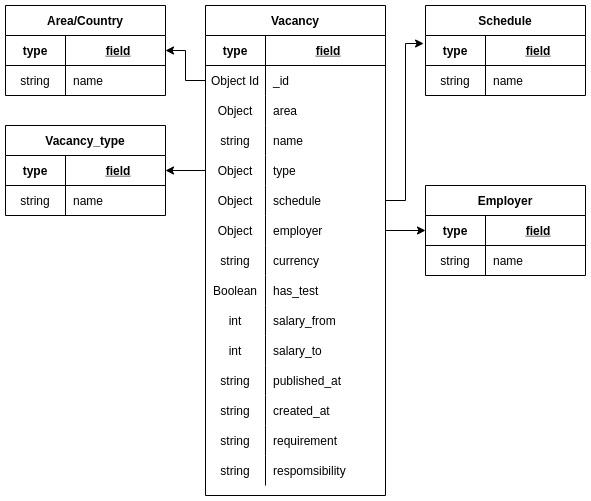
Удаление информации доставляет сложности в обоих случаях, т.к. надо решить, что делать со связями/ссылками на удаляемый объект, однако в графовой БД нет фиксированной схемы, по этому процесс удаления проще.

Больше всего преимущества графовая БД имеет в решении юзкейсов приложения, т.к. позволяет делать запросы со сложными связями быстрее.  
Например запрос всех вакансий в СПб с заданной зарплатой, от заданного работодателя с удаленным форматом: В графовой модели потребуется сделать MATCH по связям с тремя вершинами (работодатель, график, регион) и фильтрации по параметрам. В случае с реляционной моделью это потребует написания JOIN запроса по трем таблицам и фильтрации по результату, что сложнее.

Можно сделать вывод, что для задач аналитики графовая БД имеет преимущество, не смотря на проигрыш по дисковому пространству.

* 1. **Модель данных для MongoDB**

Графическое представление:

****

Оценка объема

* Объем для одного объекта Area/Country

name - string - 255 символов, 255 байт V(Area/Country) = 255 байта

* Объем для одного объекта Vacancy\_type

name - string - 255 символов, 255 байт V(Vacancy\_type) = 255 байта

* Объем для одного объекта Schedule

name - string - 255 символов, 255 байт V(Schedule) = 255 байта

* Объем для одного объекта Employer

name - string - 255 символов, 255 байт V(Employer) = 255 байта

* Объем для одного объекта Vacancy

\_id - ObjectId - 12 байт

name - string - 255 символов, 255 байт

currency - string - 5 символов, 5 байт

has\_test - boolean - 2 байт

salary\_from - int - 4 байт

salary\_to - int - 4 байт

published\_at - string - 255 символов, 255 байт

created\_at - string - 255 символов, 255 байт

requirement - string - 255 символов, 255 байт

responsibility - string - 255 символов, 255 байт

Area/Country - Object - V(Area/Country) = 255 байт

Vacancy\_type - Object - V(Vacancy\_type) = 255 байт

Schedule - Object - V(Schedule) = 255 байт

Employer - Object - V(Employer) = 255 байт

V(Vacancy) = 2322 байт Для N = 2 оценка V = 6304

Рост объёма от числа вакансий

V(объём данных в коллекции Vacancy) = N \* V(объект коллекции Vacancy) = 2 \* 2322 байт = 4644 байт, при N = 2 Данные растут линейно.

Избыточность модели

Избыточность модели низка, повторяющиеся фрагменты несущественны по объему.

Запросы к модели

Создание вакансии:

db.vacancies.insertOne(

{

"name":"Финансовый аналитик",

"has\_test":false,

"published\_at":"2021-11-04T14:52:20+0300",

"created\_at":"2021-11-04T14:52:20+0300",

"requirement":"Высшее профессиональное образование (финансы, экономика, математика). Уверенный пользователь ПК (MSOffice), отличное владение пакетом Excel. Уверенный/продвинутый пользователь 1С, БИТ.Финанс...",

"responsibility":"Формирование управленческой отчетности, сбор и аналитика данных. Настраивание процесса бюджетирования, бюджетного контроля и управления. Построение управленческой отчетности для пользователей внутри...",

"area":{

"name":"Москва"

},

"type":{

"name":"Открытая"

},

"schedule":{

"name":"Full time job"

},

"employer":{

"name":"Яндекс"

}

)

Все вакансии в Москве c зарплатой от 50 тыс.

db.vacancies.find( { area: "Москва" }, : salary\_from: { $gt: 50000 } )

Оценка числа операций для юзкейсов и CRUD

Удаление не имеет сложностей, при удалении документа коллекции (Vacancy) также удаляются все вложенные документы.

Добавление и изменение также происходит за один запрос. Существенных трудностей тривиальные операции CRUD не имеют.

Сравнение моделей

По памяти

MongoDB занимает схожий объем данных по сравнению с SQL (MongoDB 4644 байт vs SQL DB 4750 байт) и существенно меньше чем графовая БД Neo4j (5600 байт). Избыточность модели низка, повторяющиеся фрагменты несущественны по объему.

По запросам

В случае документо-ориентированной БД рассматриваемый выше запрос поиска (Зарплата в Москве > 50 000) должен выполняться быстрее по сравнению с SQL БД (отсутствует ресурсно-затратная операция JOIN, информация о локации хранится во вложенном документе) и с Neo4j (операция MATCH).

1. **Разработанное приложение**
   1. **Краткое описание**
      1. **Архитектура**

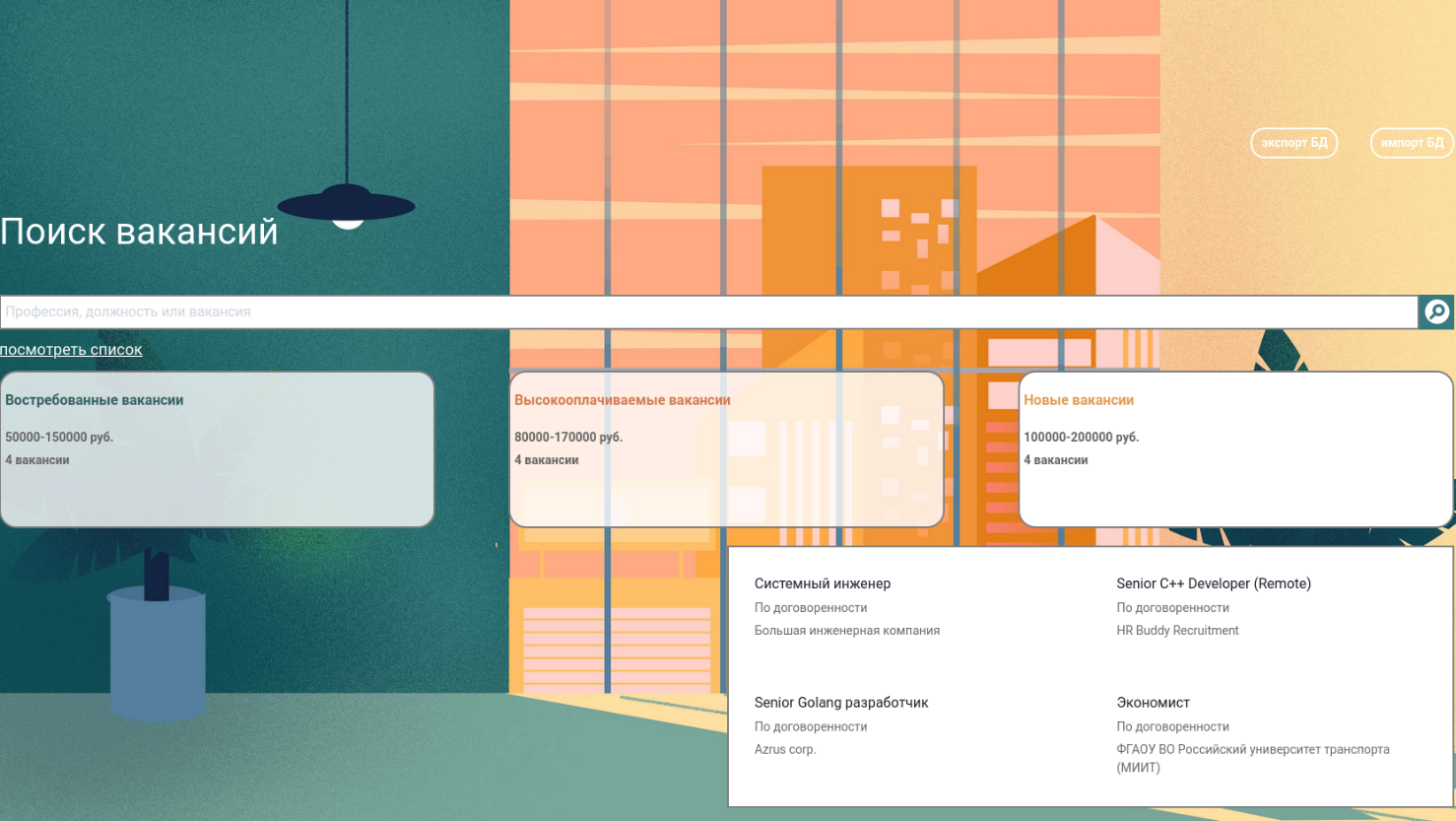
Приложение состоит из трёх сервисов, взаимодействующих между собой

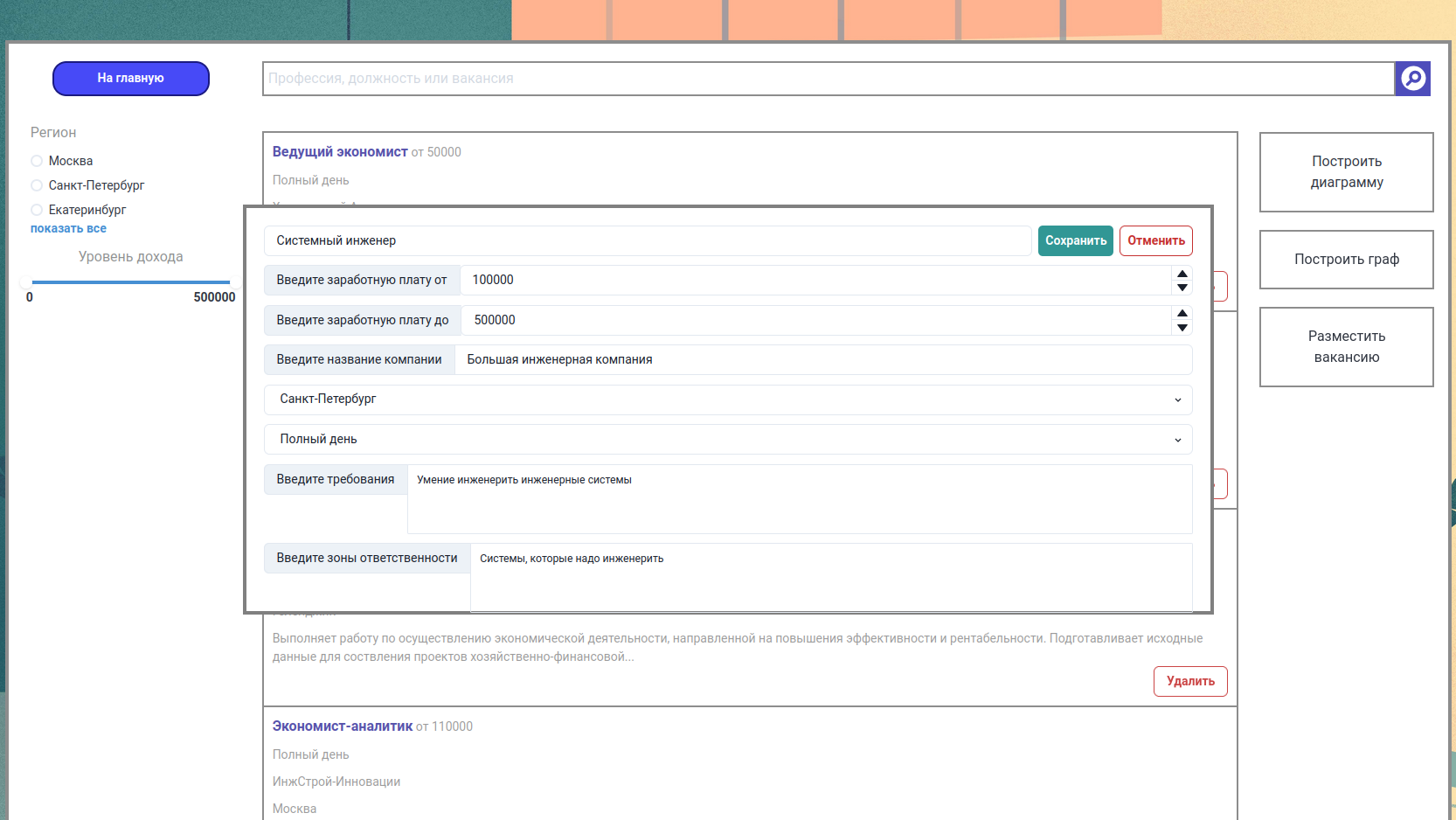
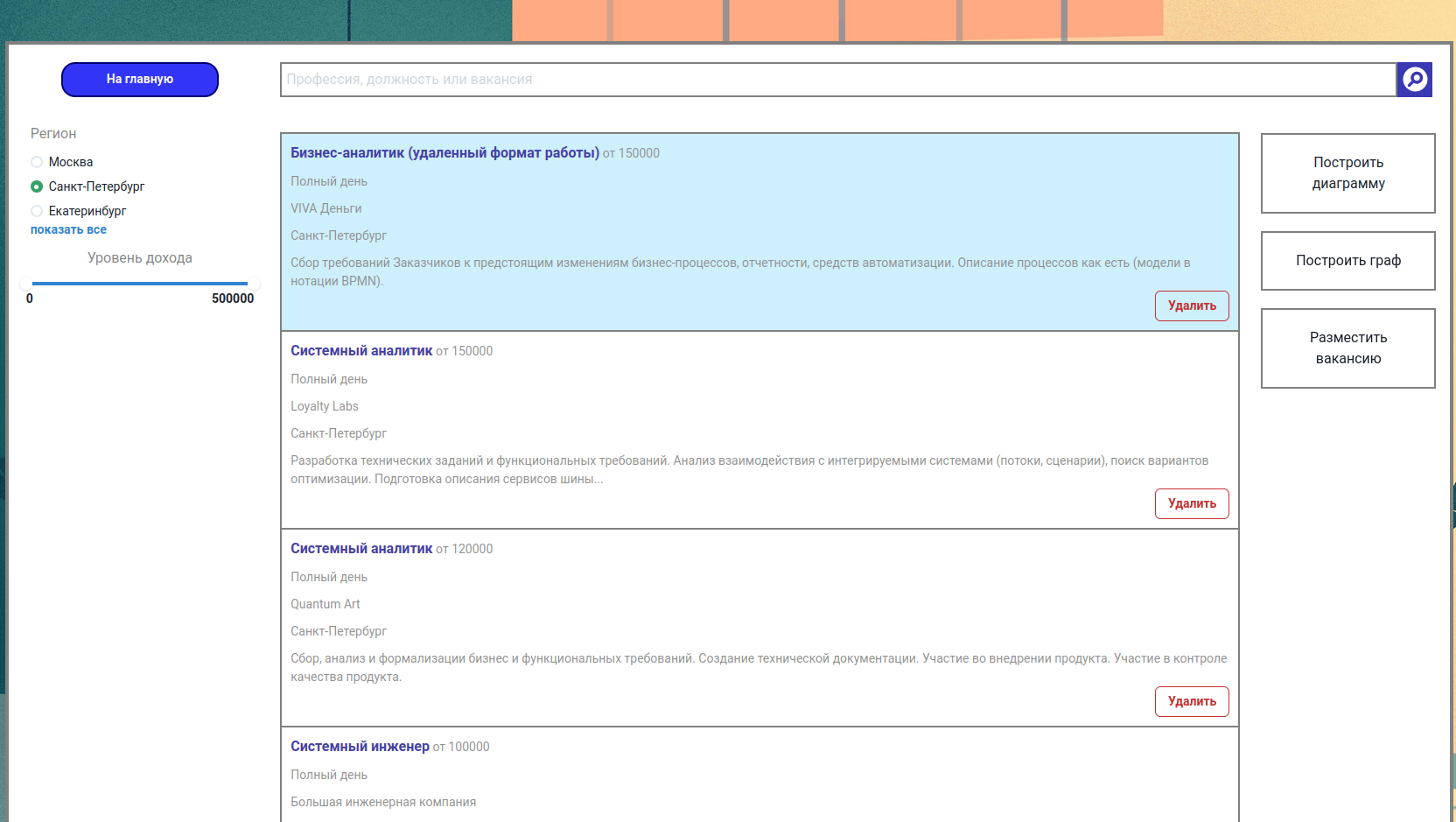
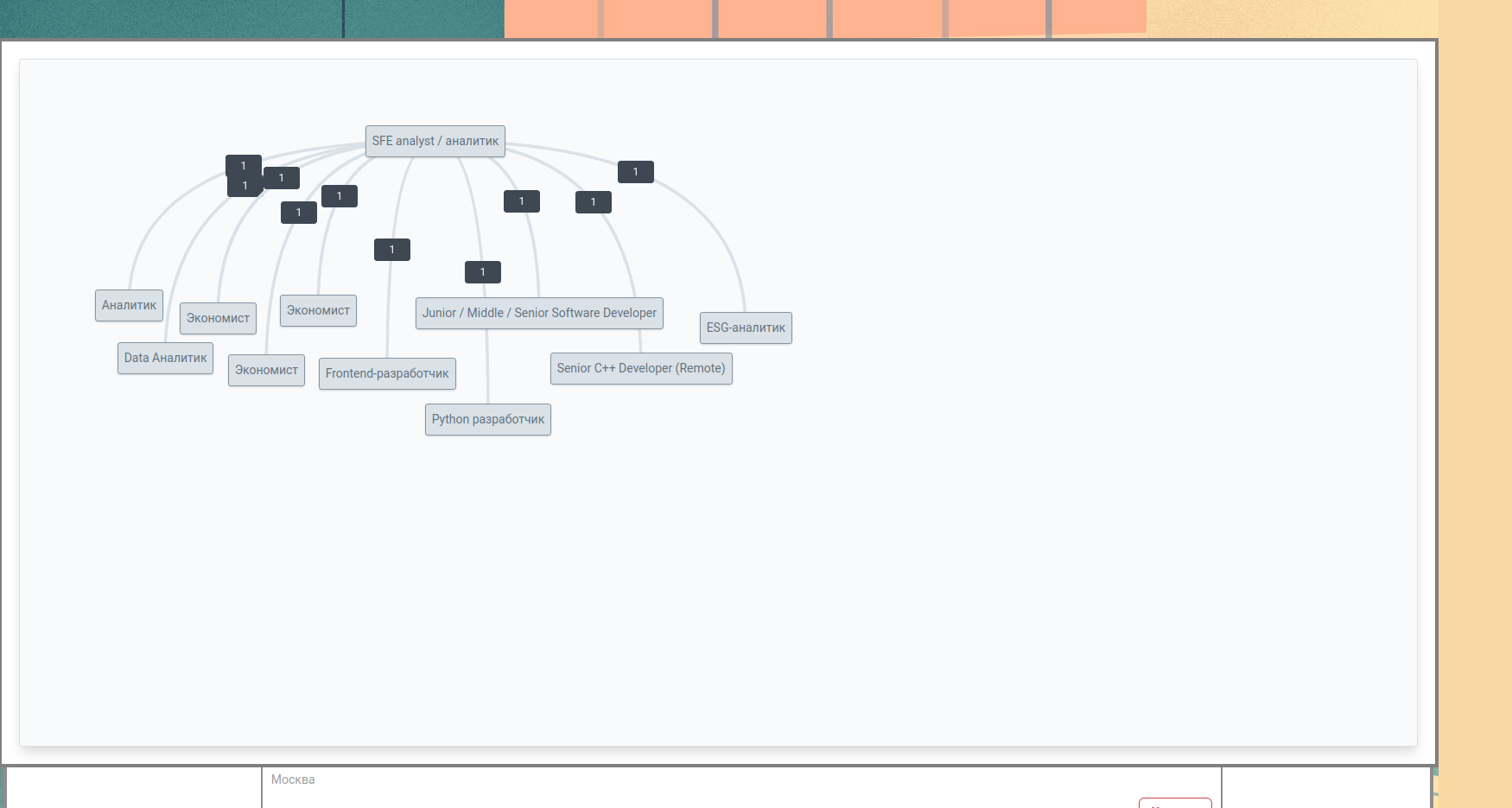
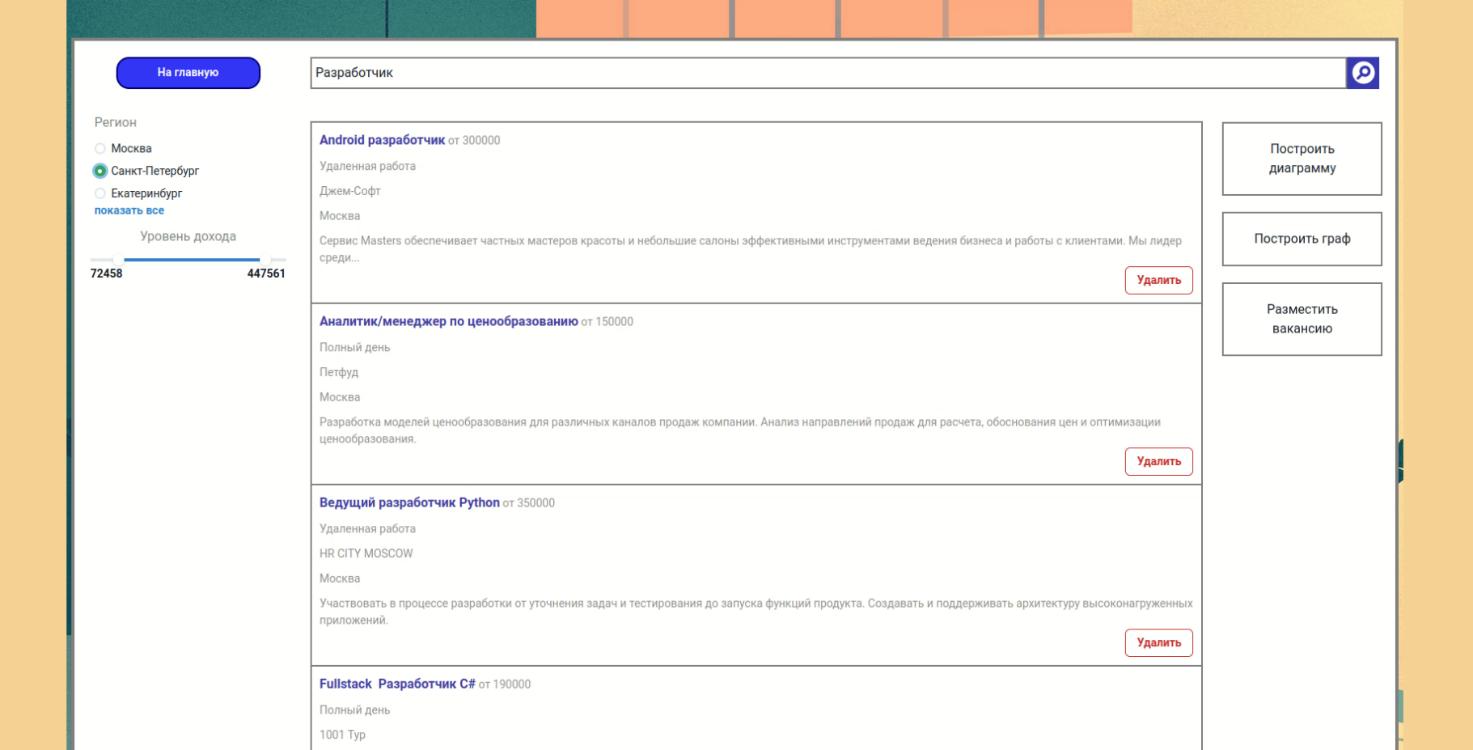
* Neo4j - база данных
* Rest - бэкэнд. Содержит бизнес-логику приложения. Общается с базой данных посредством запросов на языке запросов Cypher. Предоставляет REST-api для фронтенда. Rest-api генерируется на основе документации Openapi 3, при помощи connexion
* ui - фронтэнд. Обращается к REST-api и реализует взаимодействие с пользователем
  + 1. **Docker**

Каждый сервис контейнирезирован. Он исполняется в изолированной среде со всеми необходимыми ему зависимостями. Файлы конфигураций пробрасываются в контейнер при сборке

Для управления несколькими контейнерами используется docker-compose

* 1. **Схема экранов приложения**





* 1. **Использованные технологии**

БД: Neo4j

Back-end: Flask, Swagger (OpenAPIv3), connexion

Front-end: React

* 1. **Ссылки на приложение**

https://github.com/moevm/nosql2h21-jobs

**ЗАКЛЮЧЕНИЕ**

В процессе работы был разработан веб-инструмент для БД вакансий, который позволяет пользователю удобно и быстро добавлять и удалять вакансии, просматривать БД на странице табличного представления и строить граф по степени сходства вакансий. На стартовой странице есть аналитика самых востребованных, оплачиваемых и новых вакансий.