Санкт-Петербургский Государственный Университет
Кафедра технологии программирования

Пугачев Максим Сергеевич

Выпускная квалификационная работа бакалавра

Разработка ГИС для поиска территорий с учетом требований бизнеса

Направление 010400
Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,
ст. преподаватель,
Севрюков С. Ю.

Санкт-Петербург
2016

Оглавление

ВВЕДЕНИЕ

Всем известна ситуация, сложившаяся в российских деревнях. Большое количество хозяйств исчезли после развала системы государственного планирования, произошла массовая урбанизация, массовый отток населения из сельских районов. Но есть решение для этой ситуации - создание эффективного самоуправления в провинции.

Основополагающий фактор развития самоуправления составляет ответственное отношение самих жителей к своим природным, техническим и, что самое главное, человеческим ресурсам.

Те активисты и энтузиасты, которые решили взяться за возрождение сельских общин, скоро поняли, что русская деревня имеет немалый потенциал развития.

В 1997 году был организован Институт общественных и гуманитарных инициатив (ИОГИ). Цель института - возрождение сельских районов Архангельской области. ИОГИ имеет большой опыт работы по восстановлению местного самоуправления в деревнях и сельской местности северных регионов Европейской части России. Преодолевая непонимание и иногда сопротивление селян, Тюрин Глеб Владимирович и его единомышленники начали процесс по восстановлению самоуправления во

многих деревнях и сёлах. Сотрудники ИОГИ смогли воодушевить и убедить жителей даже самых дальних и неблагополучных районов в том, что развитие провинции зависит в основном от их собственных действий и усилий.

Тюрин Г.В. убежден, что нужно не спорить о теориях, а думать о реалиях жизни. Поэтому он попробовал воспроизвести традиции российского земства в условиях современного мира:

— Мы стали ездить по деревням, сёлам и малым населенным пунктам и организовывать встречи, клубы, семинары, деловые игры. Старались расшевелить людей, которые сникли и ни во что не верили. У нас есть наработанные технологии, которые позволяют людям посмотреть по–иному на себя, на свою ситуацию.

Население, которое начинает процесс самоорганизации, создает внутри себя орган территориального общественного самоуправления, сокращенно — ТОС. После создания ТОСа, ему вручают мандат доверия. На самом деле, это и есть земство, только современное, новая версия земства XIX века. Тогда земство было кастовым — купечество, разночинцы, и т.п. Но смысл остается тем же: самоорганизующаяся система, которая привязана к территории и отвечает за ее развитие.

Люди начинают осознавать, что они не просто решают проблему водо-, газо-, теплообеспечения, дорог, освещения, ресурсов и продуктов: они создают будущее провинции, свое будущее. Главный продукт их деятельности — новый тип общества и новые отношения в нем. ТОСы в своей деревне создают и стараются расширить зону большего благополучия. Некоторое количество проектов, которые удалось успешно реализовать в одном населенном пункте, наращивает критическую массу, которая меняет всю обстановку в районе в целом, мотивирует все большие массы населения заниматься аналогичным делом.

Всевозможные документы для или от ТОСа формируются и передаются либо в формате *.doc либо *.xls. Собирать все эти документы, приводить к

одному виду, анализировать и вести статистику крайне неудобно, что тормозит развитие процесса самоуправления. Также, если допускать вариант, что группа людей, будь то население деревни или просто молодые активисты захотят организовать подобный процесс, то в первую очередь, им необходимо будет определиться со некоторыми вещами, такими как: регион, тип производства, реализация которого возможно в этом регионе, бюджет или количество человек, необходимых для организации. Все это есть в форме вышеописанного документа, но нет никакой системы поиска или электронного помощника в этом непростом выборе.

Немаловажную роль играет создание, заполнение и организация базы данных, хранящей в себе успешные истории организации самоуправления.

ПОСТАНОВКА ЗАДАЧИ

Целью данной работы является повышение эффективности процесса обработки и анализа данных в рамках деятельности по привлечению населения. Для достижения поставленной цели предполагается решить **задачу разработки прототипа** web-приложения для обеспечения сбора информации об успешных реализациях самоуправления в деревнях, гибкого поиска по этой информации и её удобного визуального представления.

Решение данной задачи требует **реализации следующих этапов**:

1. Исследование предметной области:
    - Анализ бизнес процессов и определение структуры документа;
    - Анализ сервисов, реализующих поиск различного типа;
    - Исследование возможностей пространственного анализа;
2. Формирование требований к программному продукту;
3. На основе пунктов 1,2 проектирование архитектуры предполагаемого решения;
4. Разработка модулей поиска и добавления документов:
    - Выбор паттерна проектирования приложения;
    - Создание и заполнение индекса в БД;
    - Реализация необходимых модулей;
5. Разработка модуля поиска подходящих регионов, основанного на пространственном анализе данных;
6. Нагрузочное тестирование прототипа.

Сформулированная выше задача будет решаться для Тюрина Глеба Владимировича, который обладает необходимыми данными для заполнения

БД, организационной позицией, а также потребностью в обозначенном исследовании.

ГЛАВА 1. Определение требований и выбор оптимального решения

В данной главе будет исследована предметная область и проведен анализ бизнес процессов, используемых при организации ТОСов и начале самоуправления в сельском районе. Будут сформулированы требования к разрабатываемому решению и выбран оптимальный подход, который позволит покрыть весь необходимый функционал, а именно – поиск и добавление документов, и, в то же время, минимизировать затраты (время, ресурсы и т.п.)

§1. Исследование бизнес процессов

Основным объектом, который описывает организацию процесса самоуправления является документ, содержащий следующую информацию:

1. Имя организатора (директора, руководителя);
2. Контакты организаторы – его телефон и электронная почта;
3. Название организации (далее – фирмы);
4. Адрес расположения фирмы (улица, номер дома, населенный пункт, регион, индекс);
5. Тип производства, которым занимается фирма;
6. Дополнительное ответственное лицо и его контакты;
7. Сайт фирмы;
8. Описание фирмы.

Такая информация формирует документ–историю создания процесса самоуправления и от результата деятельности фирмы зависит, попадет ли документ в базу, то есть будет ли история успешной.

Обмен информацией может происходить как в одностороннем режиме, так и в режиме диалога.

Первый случай описывает ситуацию, когда энтузиасты, которые решили взяться за развитие территории, передают документ, описывающий их деятельность, организаторам этой идеи – таким людям, как Тюрин Г.В.

Второй - если Тюрин Глеб Владимирович и единомышленники, занимающиеся сбором и анализом вышеописанных документов, на основе этого консультируют обращающихся к ним организаторов фирм.

В рамках работы были исследованы основные критерии и задачи при организации процессов по созданию современного земства. Ниже они представлены:

- Организацию должен характеризовать документ, речь о котором шла выше;
- Организация должна составлять полноценный отчет, где описываются все экономически важные факторы развивающегося предприятия
- Тюрин Г.В. отвечает за консалтинг по развитию малонаселенных территорий, включая сбор и аналитику данных

Обозначенные критерии и задачи дают основания к созданию базы данных, содержащей документы с необходимыми параметрами для описания процесса самоуправления. Нельзя забывать, что хранимые в базе данные будут увеличиваться и, следовательно, выбранная архитектура базы должна обеспечивать быстрое выполнение запросов, скорость работы и корректность выводимых данных. При этом, надо иметь ввиду, что мощности, предоставляемые базе данных, ограничены, поэтому она не может быть масштабируема вертикально.

§2. Формирование требований

В ходе анализа бизнес процессов были выделены базовые сценарии – последовательности действий, производимых пользователем, то есть группой единомышленников Тюрина Г.В.

1. Поиск необходимых документов по тем или иным параметрам (месторасположению фирмы, типу производства и т.п.);
2. Общий анализ документов, включая пространственный анализ, то есть определение регионов, наиболее подходящих под определенный тип производства;
3. Добавление новых документов к базе.

Таким образом, разрабатываемый программный продукт должен в полной мере обеспечивать выполнение этих базовых операций. Причем, поиск необходимых документов должен быть как можно более гибким для того, чтобы дать пользователю возможность наискорейшего получения результата.

Под **гибким поиском** следует понимать, что должны присутствовать возможности поиска по основным параметрам документа. В нашем случае этими параметрами могут быть:

- Название фирмы
- Имя руководителя
- Тип производства
- Описание фирмы

Причем, также должно присутствовать разделение всех документов по группам – типам производства для более удобного дальнейшего анализа. В качестве групп были выбраны именно типы производства, так как их количество ограничено и они могут объединять в себе наибольшее количество документов, в то время как, допустим, одному названию фирмы соответствует один документ и т.п.

Так же, для более удобного визуального восприятия пользователями описания фирмы, было решено использовать виджет, то есть некий независимый программный модуль, который представляет собой карту и выполняет функцию визуализации месторасположения фирмы, описанной в документе.

Требования к виджету:

- Доступное бесплатное API
- Возможность основываться на адресе, представляющем строку, а не численных координатах
- Удобная интеграция с разрабатываемым прототипом

Пространственный анализ также составляет часть поиска и подразумевает под собой наличие некого рода соответствий «Тип производства – подходящий регион», где определяется, насколько подходит регион с помощью анализа имеющихся в нем ресурсов, которые требуются для конкретного производства. Для составления же такого соответствия необходимо воспользоваться инструментами для решения задач анализа и обработки пространственно-связанных данных

Под **добавлением нового документа** надо понимать, что должна присутствовать возможность «регистрировать» новую успешную фирму в базе данных со всеми её параметрами и описанием.

Обобщая все вышесказанное, можно сформулировать **требования**, которым должен удовлетворять разрабатываемый программный комплекс:

1. Поиск по названию фирмы;
2. Поиск по имени руководителя;
3. Возможность обобщения документов по группам – типам производства;
4. Поиск по ключевым словам в описании;

5. Поиск по любым словам в найденных результатах предыдущего запроса (для более гибкого механизма поиска);
6. Возможность добавления нового документа;
7. Просмотр подробного описания документа, т.е. всех его параметров, один из которых – адрес фирмы – будет визуализирован с помощью виджета;
8. Поиск подходящих регионов при выборе типа производства посредством пространственного анализа;
9. Масштабируемость.

§3. Выводы

Анализ функциональных требований, предъявляемых к разрабатываемому программному обеспечению позволил сделать следующие выводы:

- Было решено использовать технологию фасетного поиска. Фасетный поиск основан на выдаче результата поиска в виде маленькой части найденных данных, а также набора всех значений всех атрибутов искомых документов, которые встречаются в найденном множестве документов. Поиск состоит из многоих этапов, на каждом этапе пользователь уточняет значения различных параметров, сужая таким образом размер найденных документов. Т.е. выбрав сперва тип производства, далее пользователь может выбирать слова, которые входят в описание фирмы, или, например, в её название, сужая тем самым результаты поиска. Яркий пример технологии фасетного поиска – Яндекс.Маркет (https://market.yandex.ru/);
- В выборе архитектурного решения для реализации базы данных было принято применить NoSQL-решение, учитывая, что в дальнейшем будет необходимость масштабироваться и со временем иметь дело с большими объемами данных. В рамках текущей задачи, количество

операций на чтение значительно больше, чем на изменение, к тому же изменения краткосрочны, следовательно, можно отказаться от некоторых свойств ACID и получить большую производительность [1]. Помимо этого, NoSQL-решение имеет свойство, которое является решающим в контексте текущей задачи – это документоориентированность.

- Простота разрабатываемого решения, необходимость использования многокретириального поиска и то, что решение является именно веб приложением приводят нас к выбору инструмента, который будет выступать не только поисковым механизмом, но и документоориенторованной базой данных.
- В качестве виджета для визуализации месторасположения фирмы были анализированы наиболее популярные директивы для AngularJS, а именно 2GIS, Google Maps и Яндекс.Карты. Решение использовать именно Google Maps директивы было принято исходя из следующих соображений:
    - При проведении тестирования (см. Гл.4) было проанализировано, что используемый API виджета для корректной работы приложения требует получения ключа разработчика;
    - У всех трех кандидатов для доступа к API необходимо получить ключ разработчика. Google и Яндекс предоставляют ключ платно, но у Google есть пробный период сроком в 30 дней;
    - В API, предоставляемом 2GIS нет методов, позволяющих использовать в качестве параметра строчный адрес – необходимо использовать только координаты, что для нашего решения не подходит;
    - Google сообщество одно из самых крупных, что, соответственно, ускоряет разработку.

- Для пространственного анализа было решено использовать стороннюю систему, следовательно, выбор типа слоев – растровые или векторные - остается за такой системой. Ожидаемый результат пространственного анализа на клиенте – это либо набор подходящих регионов, либо тепловая карта с отображением различных областей для последующей оценки, насколько они подходят заданным пользователем критериям (см. Гл.3). В качестве основной источника информации о пространственном анализе, а также принципах веб-ГИС, были использованы выпускные работы и статьи конференции CPS («Процессы управления и устойчивость») предыдущих лет. В качестве основного инструмента для решения текущей задачи было решено использовать открытую географическую информационную систему O-GIS [2] (см. Гл.2 пар.3).

ГЛАВА 2. Разработка прототипа web-приложения

На основании выводов, сформулированных в предыдущей главе, перед составлением подробных спецификаций было решено разработать прототип будущей системы. Прототип подразумевает полностью работающую систему, удовлетворяющую предъявляемым базовым функциональным требованиям.

В этой главе рассматриваются вопросы архитектуры, технологических особенностей и реализации разрабатываемого прототипа.

§1. Основная концепция и выбор технологий

Можно разделить поиск, предоставляемый пользователю приложения на две составляющие. Первая – поиск документов в базе посредством инструментов ElasticSearch, вторая составляющая – это поиск подходящих регионов посредством анализа пространственных данных и операций растровой алгебры и реклассификации.

Ключевых особенностью реализации первой составляющей поиска является фасетная классификация. Именно парадигма фасетного поиска была выбрана для обеспечения удобства работы с большим количеством материалов. В качестве основного труда для изучения парадигмы фасетного поиска использовалась работа Tunkelang D. Faceted Search [3].

В качестве NoSQL решения, подходящего под заданные функциональные требования можно выбрать поисковый движок ElasticSearch [4], так как предоставляет не только функции поиска, но и является документоориентированной базой данных. Основных поисковых движков на данный момент существует несколько - Sphinx, Solr, Xapian [5, 6, 7] и др. Главное отличие ElasticSearch от похожих инструментов это гибкость и простота в использовании. Хотя на самом деле, инструмент поиска ElasticSearch не вполне самостоятельный продукт, а обертка над библиотекой Apache Lucene [8]. Но именно эта обертка предоставляет API для ввода данных, поисковых запросов, кластеризации и прочий полезный функционал.

Преимущества использования ElasticSearch в контексте текущей задачи:

- **Горизонтальная масштабируемость и отказоустойчивость**. Elasticsearch позволяет легко применить горизонтальную масштабируемость. К уже имеющейся системе можно на ходу добавлять новые серверы и поисковый движок сможет сам распределить на них нагрузку. При этом данные будут распределены таким образом, что при отказе какой-то из нод (задач), данные не будут утеряны и сама поисковая система продолжит работу без сбоев. При выпадении каких-либо серверов из кластера, если реплики данных были правильно распределены, корректно настроенное приложение продолжит поиск, как будто ничего и не произошло. После того, как сервер поднимется, он сам вернется в кластер и подтянет последние изменения в данных. В рамках нашей задачи, при возможном росте базы данных в будущем,

горизонтальная масштабируемость и отказоустойчивость является преимуществом.

- **Мультиарендность** - возможность организовать несколько различных поисковых систем в рамках одного объекта Elasticsearch. Причем организовать их можно абсолютно динамически. Очень интересная особенность, которая в отдельных случаях становится определяющей при выборе поисковой системы. На первый взгляд может показаться, что необходимости в этой особенности нет. Классические системы поиска типа Sphinx обычно индексируют какую-то одну базу с определенным кругом данных. Это форумы, интернет-магазины, чаты, различные каталоги. Все те места, где поиск для всех посетителей должен быть идентичным. Но на самом деле довольно часто возникают ситуации, когда систем поиска должно быть больше одной. Это либо мультиязычные системы, либо системы, где есть определенное количество пользователей, которым нужно предоставлять возможность поиска по их персональным данным. В случае нашего прототипа это не самая приоритетная вещь, но, смотря наперед, при дальнейшем развитии приложения, она может оказаться очень полезной. В контексте задачи, если предполагать, что инициатива Тюрина Глеба Владимировича будет охватывать не только Российскую Федерацию, то этот является существенным плюсом.
- **Отсутствие схемы** — Elasticsearch позволяет загружать в него обычный JSON-объект, а далее он уже сам все проиндексирует, добавит в базу поиска. Позволяет не тратить много времени и ресурсов над настройкой структуры данных при быстром прототипировании. Это выгодно для решения текущей задачи, так как на клиентской стороне будут формироваться именно JSON-объекты для работы с базой (см Гл.3).

- **RESTful api** — ElasticSearch почти полностью управляется по HHTP, т.е., например, добавление информации в индекс и поиск по индексу производятся с помощью HTTP-запросов [9]. Обмен информацией идет в формате JSON.

Безусловно, на этапе прототипирования, разумно применять механизмы, позволяющие максимально снизить время реализации программного комплекса. Для создания web-приложения был выбран open-source фреймворк AngularJS исходя из опыта автора работы с ним [10].

Некоторые плюсы фреймворка AngularJS:

- **Декларативный стиль кода** - при создании шаблонов в Angular.js применяется декларативная парадигма программирования [11]. Это делает код более легковесным, облегчает его чтение и поддержку, так как описывается необходимый конечный результат, а не все шаги по его достижению.
- **Использование директив** - В качестве языка шаблонов в Angular.js используется HTML. Он расширяется с помощью директив, которые добавляют в код сведения о требуемом поведении (например, о необходимости загрузить определенный модуль сразу после загрузки страницы). Директивы позволяют вам сконцентрироваться на проработке логики и работать более продуктивно. Их можно использовать повторно, что также повышает читабельность кода. Применимо к текущей задаче используются сторонние директивы для интеграции клиентской части с серверной, где хранится база данных, а также разработаны собственные директивы для валидации форм.
- **Двустороннее связывание данных** - любые изменения в пользовательском интерфейсе сразу же отражаются на объектах приложения и наоборот. Фреймворк сам следит за событиями

браузера, изменениями модели и действиями пользователя на странице, чтобы сразу обновлять нужные шаблоны. При этом в коде JavaScript не требуется хранить ссылки на DOM-элементы и явно ими манипулировать. Требуется только описание необходимого результата в терминах состояния модели, и тогда не нужно использовать низкоуровневые конструкции. Двустороннее связывание данных напрямую зайдествовано в используемом паттерне, который подробно описан в текущей главе, параграфе 3.

В качестве программной платформы была использована Node.js, которая будет создавать и запускать HTTP-сервер для клиентской части [12]. В качестве менеджера зависимостей используется Bower [13]. Этот пакетный менеджер для клиентской части JavaScript помогает подключать другие необходимы внешние библиотеки, обеспечивая вызовы к ним из JavaScript кода. Несмотря на то, что Node.js сам имеет возможность управлять зависимостями (npm), его обычно используют для серверного JavaScript. Главное отличие npm и Bower — подход к установке зависимостей пакетов. Npm устанавливает зависимости для каждого пакета отдельно, в папку этого пакета, потом так же ставит зависимости зависимостей и так далее. В клиентском JavaScript это недопустимо: нельзя подключить на страницу две версии jQuery или любой другой библиотеки. В Bower каждый пакет устанавливается один раз, и в случае конфликта зависимостей Bower просто не станет устанавливать пакет, несовместимый с уже установленными. Таким образом, Node.js выполняет роль веб-сервера клиентской части, а Bower – роль менеджера пакетов.

§2. Создание и заполнение индекса БД

С помощью предоставляемого ElastiSearch API был создан индекс в NoSQL базе данных, отвечающий требованиям поставленной задачи. Индекс — это просто материализованная структура данных, такая как B-дерево или хэш [14]. Индекс используется для быстрого доступа к определенному

содержимому таблицы реляционной базы данных. Индекс представляет собой структуру, в которой упорядочены значения одного или нескольких столбцов таблицы базы данных, таких как столбец фамилии в таблице работников. Если искать какого-либо сотрудника по его фамилии, использование индекса позволит получить нужную информацию быстрее, чем при поиске с просмотром всех строк таблицы. В контексте NoSQL решения индекс – такое же описание содержимого базы данных, только с помощью типов ключ-значение (Рисунок 1).

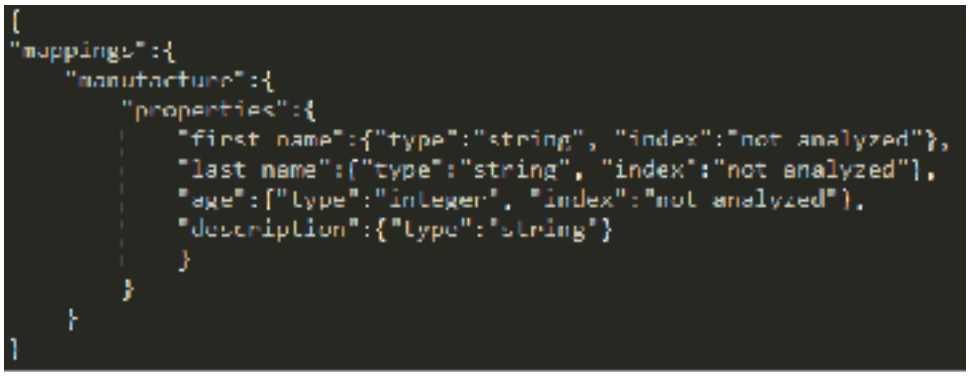


Рисунок 1. Пример индекса NoSQL

В API ElasticSearch существуют удобные методы для заполнения данными индекса. Через HTTP-запрос типа PUT можно импортировать как один объект, так и множество, содержащееся в JSON-файле. Чтобы сгенерировать необходимые данные и заполнить ими индекс базы использовался скрипт на языке программирования Python – elasticsearch-test-data [15].

§3. Архитектура разрабатываемого решения

После выбора основных технологий, необходимо определиться с архитектурой разрабатываемого комплекса. Проанализировав требования, предъявляемые к программному продукту, было принято решение об использовании модульной системы и шаблона проектирования MVC [16].

Для поставленной задачи необходимо реализовать несколько модулей:

- Модуль поиска по документам;
- Модуль ввода и регистрации документов (в будущем можно реализовать модуль CRUD [17], т.е. добавить еще возможности обновления и удаления существующего документа);
- Модуль интеграции клиентской части с сервером системы, предоставляющей функционал для пространственного анализа;
- Модуль редактирования описания производств.

Основная цель применения концепции MVC состоит в отделении бизнес-логики (*модели*) от её визуализации (*представления*, *вида*). За счет такого разделения повышается возможность повторного использования. Концепция MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента:

- *Модель* (англ. *Model*). Модель предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
- *Представление*, *вид* (англ. *View*). Отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами.
- *Контроллер* (англ. *Controller*). Обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

На рисунке 2 представлена диаграмма взаимодействия между компонентами MVC и пользователем.

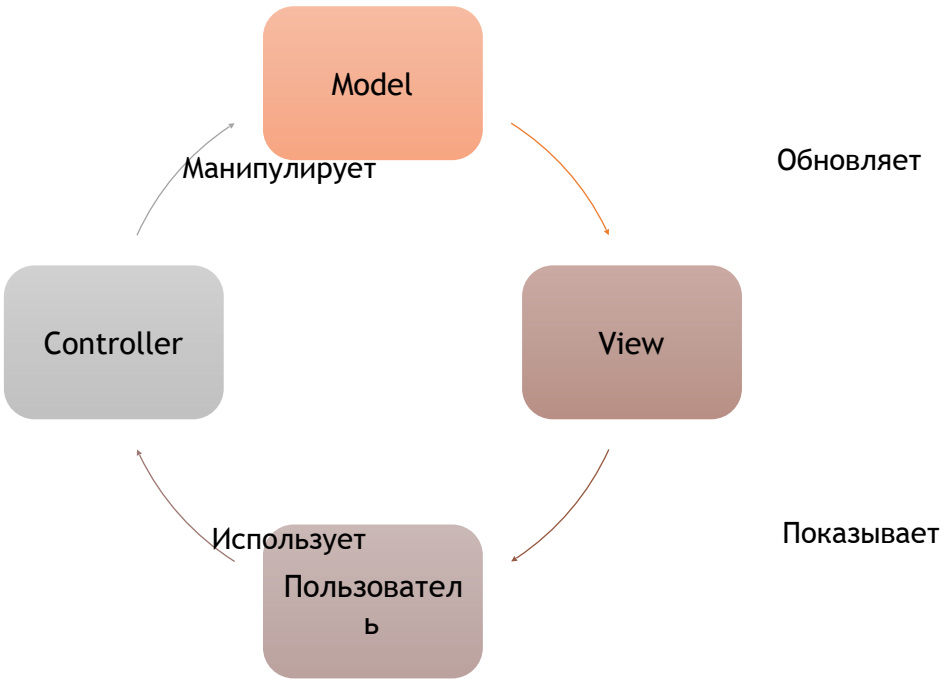


Рисунок 2. Диаграмма взаимодействия MVC и пользователя

В контексте нашей задачи реализованы несколько контроллеров:

1. Контроллер, отвечающий за список результатов поиска
2. Контроллер, отвечающий за подробное описание документа
3. Контроллер, отвечающий за интеграцию между клиентом и сервером ElasticSearch
4. Контроллер, отвечающий за интеграцию между клиентом и сервером O-GIS

Основные методы контроллеров, соответственно:

1. Получение и вывод в список результатов поиска документов
2. Получение и вывод данных для подробного описания документа
3. Проверка формы на валидность, отправка данных на сервер ElasticSearch, получение и проверка ответа с сервера
4. Проверка формы на валидность, отправка данных на сервер O-GIS, получение и проверка ответа с сервера, вывод результатов пользователю

Основной моделью в контексте задачи является модель документа со всеми соответствующими полями, такими как название фирмы, адрес фирмы и т.д.

Представления подробно описаны в следующем параграфе.

Базовые операции растрового анализа довольно ресурсоемки, следовательно, проведение их на описываемом клиенте существенно снизит производительность приложения. Одна из таких операций ключевая – это операция нахождения пересечения растровых слоев. Нас интересует классификация участков слоев, которая, по нашему мнению, будет отображать наиболее подходящие и неподходящие под определенный вид деятельности регионы. Классификация участков получается не просто нахождением пересечения нескольких слоев, а также дальнейшим эколого-географическим анализом значений этих пересечений. Необходимо изучить какие именно нижний и верхний пределы в каждом растровом слое образуют необходимый диапазон для конкретного типа производства и задать несколько таких диапазонов для более полного анализа регионов. Выделение таких диапазонов – базовая задача растровой реклассификации.

Исходя из вышеописанных условий, можно составить следующие правила:

1. Необходимо наличие словаря соответствий «Тип производства – набор слоев»;
2. Также у каждого слоя должен быть указан диапазон в соответствие текущему типу производства;
3. Необходимо использование удаленного сервера – системы с возможностью выполнения базового растрового анализа;
4. В качестве предобработки должны быть выполнены операции растровой реклассификации исходя из имеющихся диапазонов для каждого слоя из составленного набора;

5. Операция нахождения пересечений проводится уже на слоях, прошедших предобработку;
6. Результирующие участки должны быть получены на клиент и представлены пользователю.

В качестве удаленного сервера – системы с возможностью выполнения операций растровой алгебры и растрового анализа была использована система O-GIS, так как она полностью подходит под вышеописанные условия. API системы O-GIS предоставляет возможности выбора слоев, отправки их на сервер и выполнение операций над этими слоями, основными из которых для нас являются нахождение пересечений и растровая реклассификация.

На рисунке 3 приведена схема, отображающая архитектуру разрабатываемого комплекса и используемые ключевые технологии.

Рисунок 3. Схема архитектуры программного комплекса

§4. Разработка модулей прототипа приложения

4.1. Модуль поиска документов

Разработка модуля поиска требует реализации фасетной классификации. Так как в API ElasticSearch существуют методы для быстрой и легкой реализации фасетной классификации, остается задача обеспечения взаимодействия фреймворка AngularJS с поисковым движком ElasticSearch.

По большому счет AngularJS является основой, которая связывает HTML-код с JavaScript объектами-моделями. Когда изменяется один из объектов, автоматически обновляется и генерируемая страница. Обратное тоже верно – модели связаны с текстовым полем, то есть контентом страницы. Если изменяется контент, это вызывает изменение и в коде приложения.

AngularJS включает в себя множество директив, которые позволяют связать HTML-элементы моделей, добавлять им новое поведение, создавать компоненты. Именно директивы позволят облегчить процесс настройки взаимодействия web-приложения с серверной частью, где находится поисковый движок ElasticSearch.

AngularJS хорошо известный и популярный фреймворк, поэтому для более эффективной разработки было решено искать готовые директивы для работы с ElasticSearch, а не писать свои самостоятельно.

В качестве готового решения успешно подошел набор директив ElasticUI, разработанный Yousef El-Dardiry [18]. Концепт ElasticUI таков, что создается отдельное представление (View) под имеющийся индекс в ElasticSearch и после можем к нему добавлять агрегирование данных, сортировку, пейджинг (подкачку страниц), фильтры и т.п. путем добавления директивы в HTML-код. Основными директивами для реализации взаимодействия AngularJS с ElasticSearch и фасетной классификации были:

- eui-index – для указания названия индекса в ElasticSearch;

- eui-host – для указания адреса, на котором находится поисковый сервер;
- eui-searchbox – для создания элемента input с поиском по одному из полей индекса;
- eui-singleselect – для создания фасетов, т.е. возможности выбора интересующего значения какого-либо одного поля индекса;
- eui-checklist – для создания мульти-селектного фасета, т.е. возможности выбрать несколько значений у одного поля-индекса, тем самым сужая область поиска;
- eui-simple-paging – для реализации пейджинга.

На рисунке 4 изображен базовый пользовательский интерфейс над реализованным модулем поиска, без стилизации.

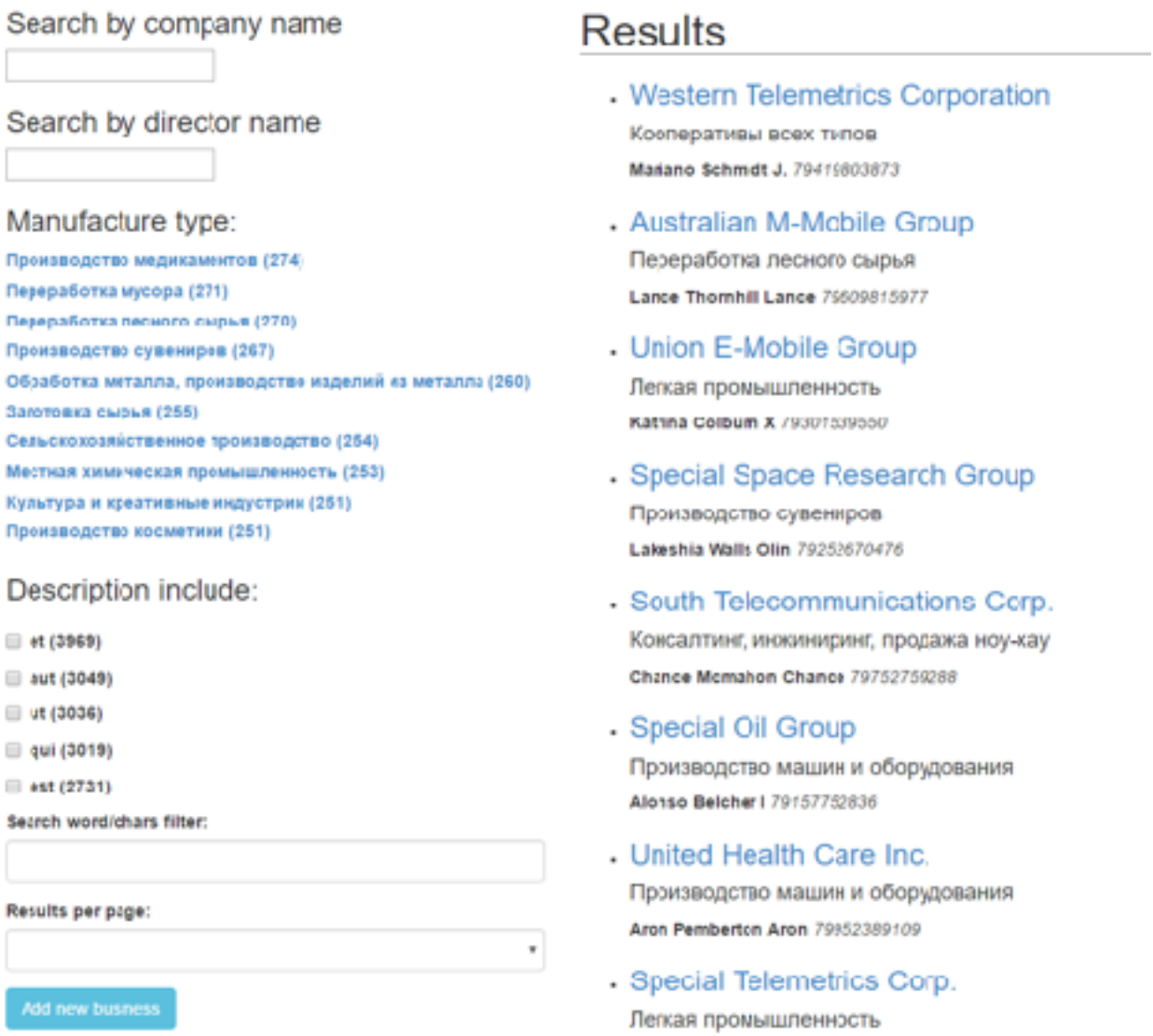


Рисунок 4. Пользовательский интерфейс модуля поиска

Здесь первые два поля слева реализуют поиск по соответствующим их названию полям, то есть пользователь может искать документы по названию фирмы или имени директора. Блок “Manufacture type” содержит фасеты по полю «Тип производства», что позволяет пользователю сразу видеть какое-либо количество сформированных фасетов (на рисунке их 10), и, выбирая один из них получать все значения этого фасета, то есть те документы, где поле «Тип производства» будет такое же, как выберет пользователь. Блок “Description include” является мульти-селектным фасетом. Он предоставляет пользователю выбирать слова, которые есть в описании документов. Ниже, на рисунке 5 вы можете посмотреть результат действия вышеописанных блоков и полей.

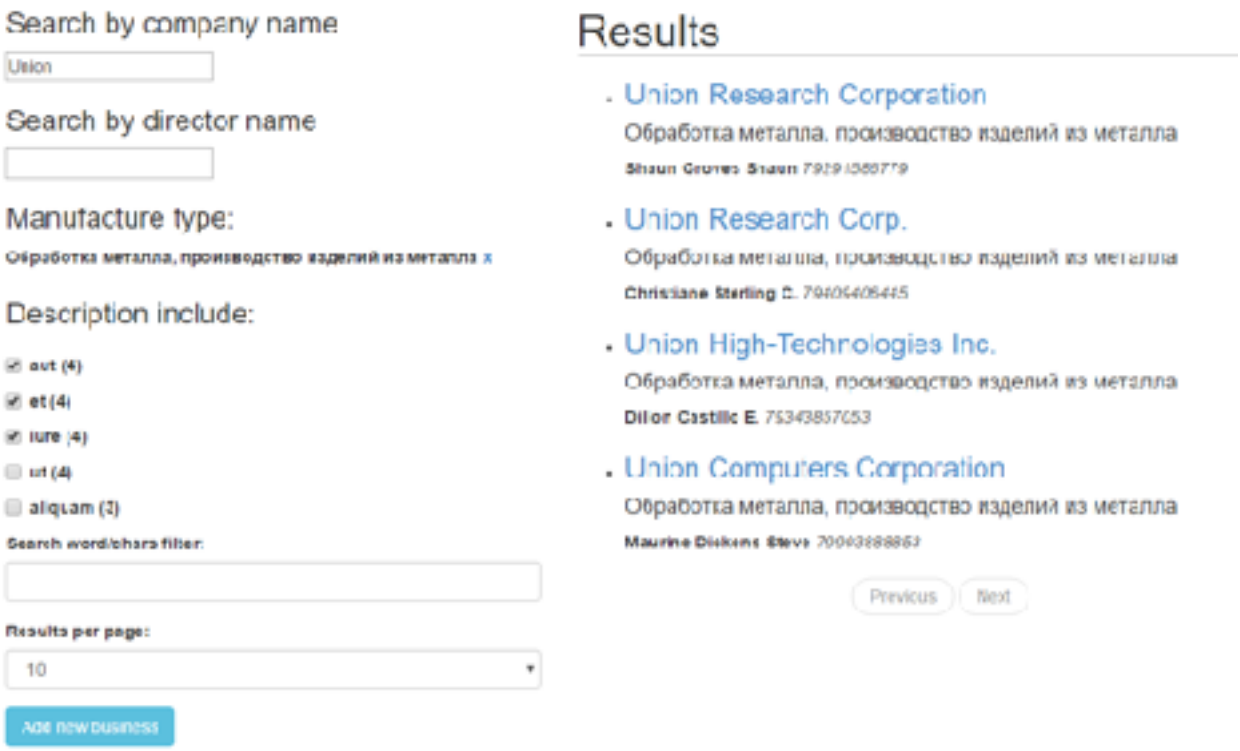


Рисунок 5. Использование поиска

4.2. Модуль ввода и регистрации документов

Разработка модуля добавления относительно проста и требует только создание HTML-формы и её правильной валидации при добавлении нового документы. Были имплементированы директивы, отвечающие за обеспечение корректной валидации и вывод ошибок при попытке отправить на сервер ElasticSearch невалидную форму. Таким образом, если пользователь не заполняет поле формы или заполняет его неправильно, то срабатывает директива, отвечающая конкретно за это поле и фиксирующая ошибку, которая впоследствии отображается пользователю. Если будет попытка обойти форму и прислать на сервер ElasticSearch некорректный запрос, то сработает проверка приходящего ответа, данные добавлены не будут и пользователь будет оповещен об этом.

На рисунке 6 представлен GUI при попытке отправить форму с неправильно заполненными полями (телефона и электронной почты).

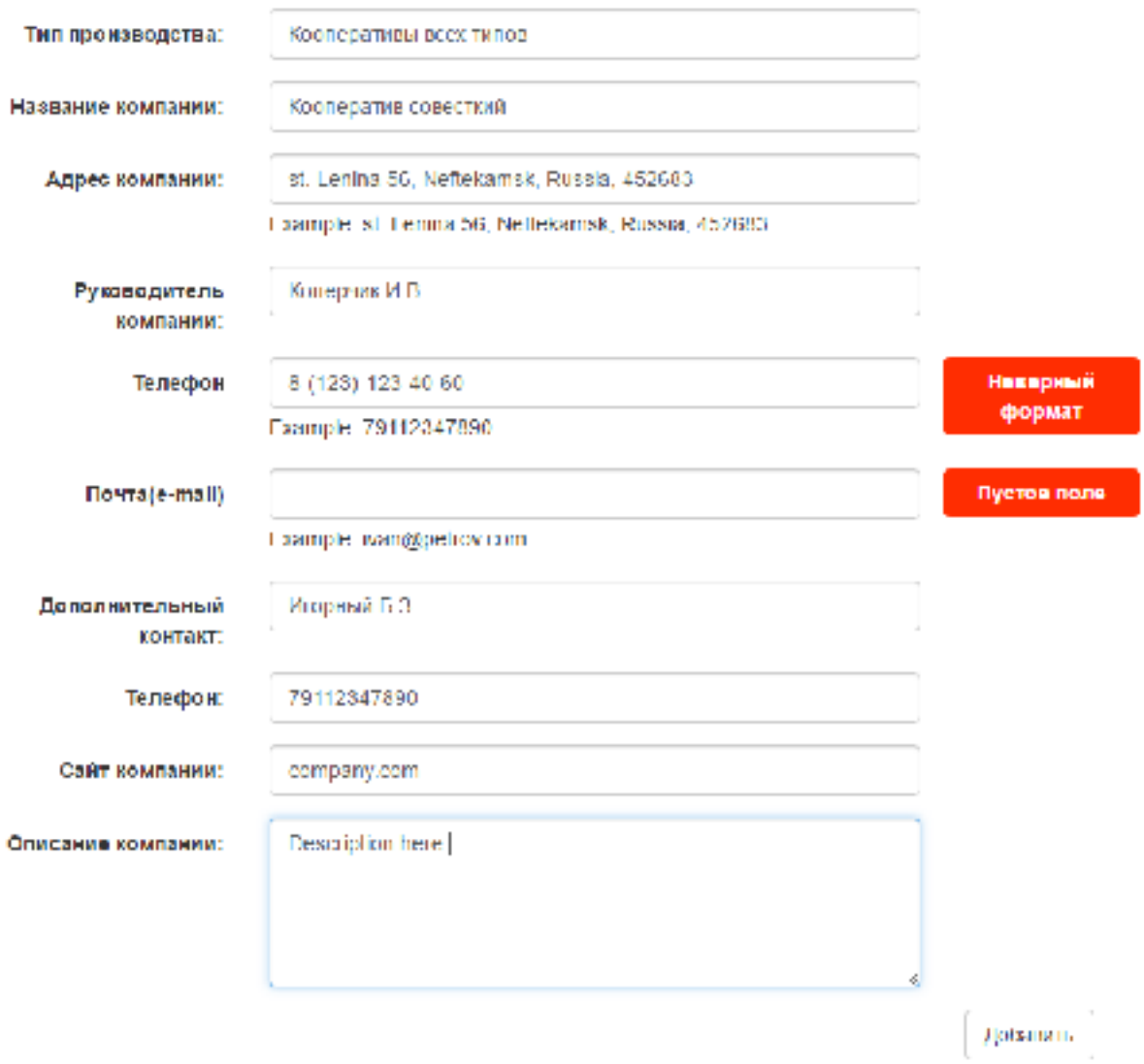


Рисунок 6. Отправка невалидной формы

4.3. Модуль интеграции

Разработка модуля интеграции основывается на взаимодействии клиентской части с сервером O-GIS. При выборе пользователем определенного типа производства ему должна быть представлена карта с классификацией регионов. Ссылка на эту карту находится в индексе ElasticSearch в виде строчных данных, а получение этой ссылки происходит посредством интеграции данных, получаемых из модуля редактирования описания производств и системы O-GIS. Эти данные формируются в JSON-пакет, после чего отсылаются на сервер O-GIS, и в ответ на клиентскую часть приходит вышеуказанная ссылка на готовую карту, которая заносится в индекс ElasticSearch.

4.4. Модуль редактирования описания производств

Разработка модуля редактирования описания производств необходима для того, чтобы пользователь мог сформировать все требуемые пространственные данные для типов производств, такие как набор слоев и соответствующие диапазоны значений.

Модуль включает в себя форму для заполнения, редактирования и/или обновления данных, необходимых для последующего пространственного анализа, таких как словарь соответствий «Тип производства – набор слоев» и набор диапазонов для каждого слоя. Этот модуль будет задействоваться при развертывании приложения для начальной инициализации-и по желанию пользователя для немедленного редактирования и/или обновления данных.

Структура пакета, протокол, способ и процедура его передачи подробно описаны в Гл.3. пар.3. Ниже, на рисунке 7 представлен пользовательский интерфейс модуля редактирования описания производств.

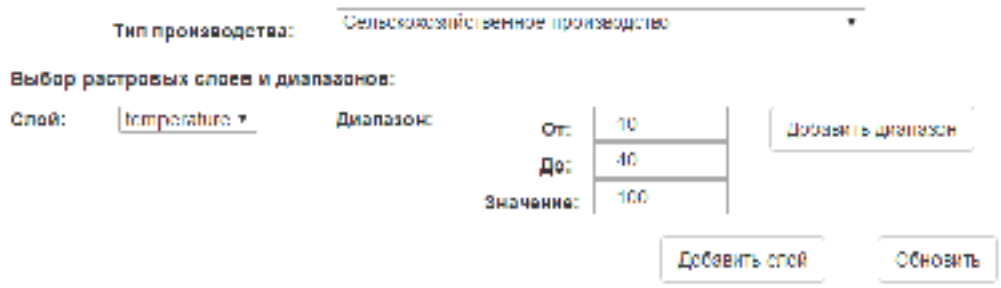


Рисунок 7. Пользовательский интерфейс модуля редактирования описания производств

ГЛАВА 3. Использование пространственного анализа для улучшения качества поиска

§1. Существующие технологии и методы

Технологии растрового геоинформационного анализа и моделирования могут быть использованы для решения разнообразных научных задач. Широкое применение эти технологии нашли в области эколого-географического моделирования (environmental niching).

Создание первых общедоступных систем растрового моделирования для решения научных задач в области эколого-географического анализа и моделирования (niching analysis) относится к 80-м годам XX века. Именно этот период характеризуется началом широкого распространения персональной компьютерной техники.

Метод эколого-географического моделирования на базе ГИС впервые широко использован в работах австралийских биологов [19, 20, 21]. Технология позволяет путем анализа распространения вида в пространстве экологических факторов среды (для проведения такого анализа необходимо иметь карты ареала биообъекта и экологических факторов среды) выявить факторы, лимитирующие распространение объекта и оценить экологические амплитуды объекта по отношению к лимитирующим факторам среды. На практике это позволяет прогнозировать потенциал распространения вредных и полезных объектов на новые для них территории.

§2. Применение методов пространственного анализа в текущей задаче

В рамках текущей задачи применяется вышеописанный метод. Для этого поставим в соответствие каждому типу производства определенный набор растров и выявим области их пересечения. Допустим, что тип производства «Переработка лесного сырья» требует наличие леса и воды. Ставим в соответствие этому типу два растра (карту плотности леса и карту пресных водоемов), находим их пересечение и выявляем регионы, наиболее благоприятные для данного типа производства. Таким образом пользователь сразу будет видеть, какие регионы ему наиболее интересны, более того, при этом он видит какие именно фирмы из тех, что занимаются выбранным производством, располагаются в подходящих регионах. Это может быть полезно при анализе и консалтинге этих фирм.

Встает вопрос о существовании необходимых баз данных и ГИС-слоев, так как слои являются основными объектами, с которыми работают

геоинформационные системы. Все необходимые данные выставлены в интернете и доступны для скачивания (например, worldclim.org, agroatlas.ru).

§3. Формирование правил пространственного анализа

Связь «Тип производства – набор слоев – набор диапазонов» является основным правилом пространственного анализа, которое формируется в формате JSON на клиенте и передается на сервер O-GIS для последующих операций. Формирование таких правил пользователем происходит при использовании модуля поиска подходящего региона. Пользователь указывает конкретные слои (будут показаны варианты, доступные из библиотеки O-GIS), их количество и диапазон значений, подходящий под вид деятельности. Эти данные посылаются на сервер O-GIS по протоколу HTTP с помощью технологии JSONP (JSON with padding) для последующей обработки и получения назад на клиент необходимых результатов. Технология JSONP – это дополнение к базовому формату JSON, которое предоставляет способ запросить данные с сервера, находящегося в другом домене. Обработка ответа в JSONP происходит следующим образом: вместе с запросом клиент в специальном параметре передает название функции, которая и будет выполнена после получения данных с сервера. Сервер кодирует данные в JSON формат и оборачивает их в вызов функции, название которой и получает из вышеуказанного параметра. В контексте нашей задачи имеем:

1. Пользователь, выполняя запрос JSONP, отправляет данные, связанные с пространственным анализом на сервер O-GIS;
2. На сервере происходит получение данных, их обработка, конкретно - растровая реклассификация каждого слоя в соответствии с указанными диапазонами и нахождение пересечения обработанных слоев. Полученный результат в формате JSON отправляется назад на клиент;
3. На клиенте выполняется заранее указанная в запросе функция-callback, которая проверяет ответ сервера и, в случае успеха, отправляет

полученные с сервера данные, а именно ссылку на готовую карту с классификацией регионов в индекс ElasticSearch.

Для описания формата исходящего пакета разработана следующая JSON схема:

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "layers": {
      "type": "object",
      "properties": {
        "0": {"type": "string"},
        "1": {"type": "string"}
      },
      "required": ["0","1"]
    },
    "operation": {"type": "string"},
    "parametres": {
      "type": "object",
      "properties": {
        "0": {
          "type": "object",
          "properties": {
            "100": {"type": "string"},
            "200": {"type": "string"}
          }
        },
        "1": {
          "type": "object",
          "properties": {
            "100": {"type": "string"},
            "200": {"type": "string"}
          }
        }
      },
      "required": ["0","1"]
    }
  },
  "required": ["layers","operation","parametres"]
```

```
}
```

Так как JSONP в силу своей природы выполняется асинхронно, то есть пользователь не видит сразу, выполнен ли его запрос корректно, то будет работать дополнительная функция, которая не даст пользователю продолжить работу, пока он не получит результат с сервера O-GIS. Только в случае успешного выполнения запроса к серверу O-GIS данные будут передаваться в индекс ElasticSearch. Иначе – выдавать пользователю ошибку.

ГЛАВА 4. Тестирование прототипа web-приложения

§1. Тестирование

Для тестирования прототипа на основе собранных требований была произведена генерация тестовых данных (см. гл.2 пар.2). Всего в индекс было добавлено 1000 документов.

Для сбора показателей и определения производительности и времени отклика разрабатываемой системы в ответ на внешний запрос с целью установления соответствия требованиям, предъявляемым к данной системе было решено провести нагрузочное тестирование (load testing).

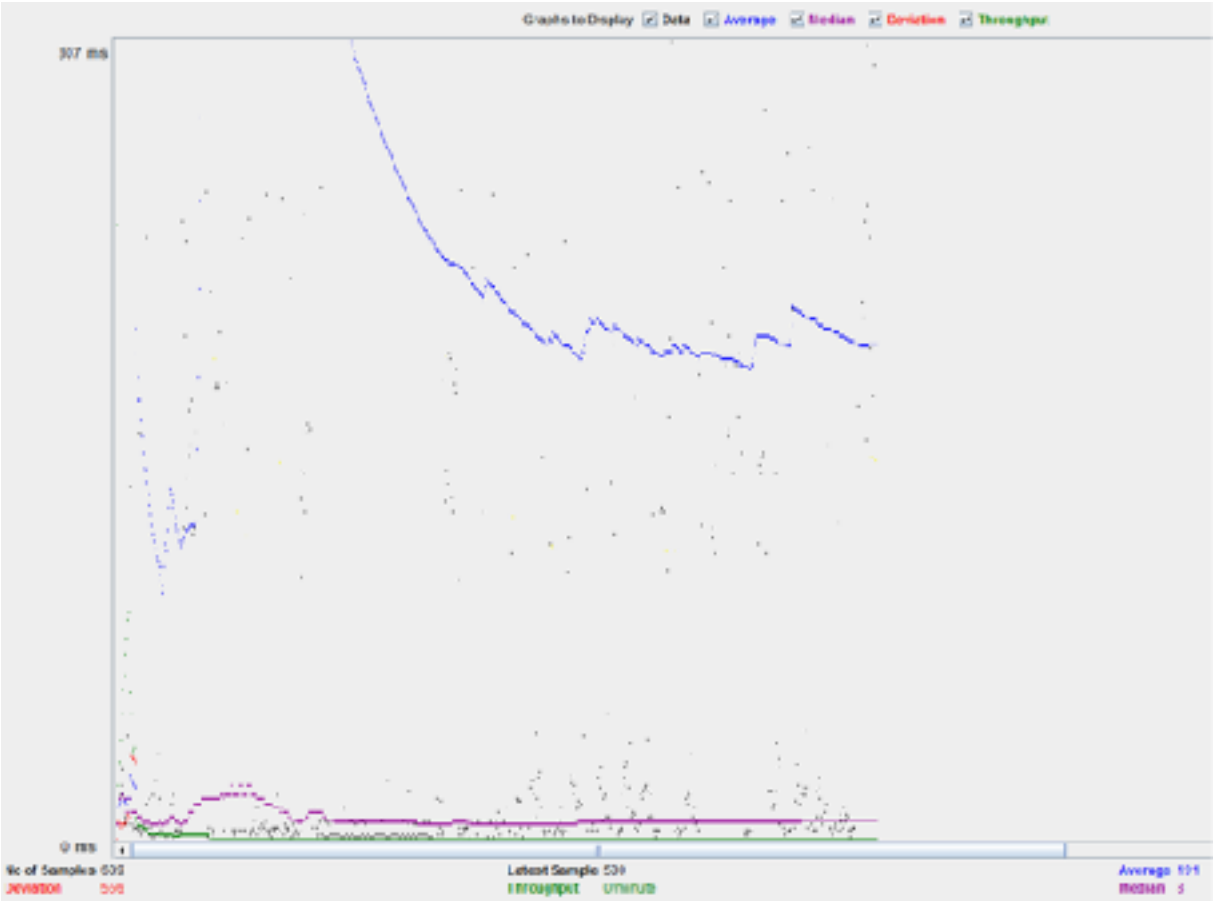
Для проведения нагрузочного тестирования есть много различных инструментов. Было принято решение использовать open source инструмент JMeter от Apache, так как у него есть ряд преимуществ:

- Open source
- JMeter является Java-приложением. Соответственно получаем независимость от операционной системы
- Широкая распространенность и наличие в интернете большого количества ресурсов по JMeter, как теоретических, так и практических примеров
- Удобное масштабирование

- Возможность работы как через GUI, так и через консоль
- Расширяемость

Так же JMeter имеет удобные плагины, которые расширяют его базовые функции. Например, дополняет Listeners, такие как PerfMon – для анализа загруженности сервера и/или Java-машины.

JMeter позволяет создавать Thread group, то есть группу потоков, в которых можно задавать необходимое количество предполагаемых пользователей разрабатываемого прототипа приложения. Так как разрабатываемая система не предполагает разделения ролей, то настройку авторизации пользователей можно пропустить. В рамках текущей задачи для тестирования было создано 100 пользователей с Ram-Up периодом в 10 секунд. Это означает что пользователи не все разом начнут использование приложения, а будут заходить группами по 10 человек.



После настройки конфигурации JMeter с веб-сервером, на котором запущена клиентская часть приложения и проведения самого тестирования были получены представленные на рисунке 7 результаты, описывающие работу прототипа.

Анализ графика говорит о том, что среднее время отклика (Average) двигается скачками, но находится примерно на одном уровне – примерно 200ms. Значение параметра скорости обработки запроса (Througput) является 0 запросов в минуту, так как отправляется только один запрос на получение данных в самом начале использования приложения и один запрос, когда хотим добавить новый документ в индекс, а во все остальное время никаких запросов не следует. Average и Throughtput являются основными показателями нагрузочного тестирования.

§2. Выводы

В ходе работы было проведено минимальное тестирование разработанного прототипа. Имеющееся решение полностью удовлетворяет предъявляемым функциональным требованиям и обладает необходимым потенциалом гибкости и масштабируемости. При необходимости, возможна независимая модификация каждого модуля системы.

Нельзя не заметить, что прототип не является полноценной системой, так как он не оснащен надлежащей системой безопасности и имеет ряд уязвимых мест, например, кросс-доменные запросы: в следствие того, что в клиентской части приложения используется стандарт XMLHttpRequest, который позволяет делать запрос только в рамках текущего сайта, а при использовании другого домена, порта или протокола браузер будет выдавать ошибку, а запросы с клиентской части идут как раз по другому порту (если система развернута на одной машине).

Чтобы исправить эту проблему и валидно отправлять запросы было решено использовать плагин для браузера Google Chrome под названием

CORS (Cross-Origin Resource Sharing) [22], который позволяет отсылать запросы между двумя ресурсами. Но здесь и рождается проблема безопасности. Так как запросы не фильтруются, пользователь будет иметь возможность, допустим, удалить весь индекс из базы или манипулировать данными и т.п.

Принимая во внимание, что планируется развитие прототипа до полноценной эксплуатационной системы, то всех аспектов безопасности веб приложения стоит придерживаться и исправить потенциально уязвимые места. Эта работа может быть закончена в магистратуре.

Список литературы и источников

1. ACID - http://www.softpoint.ru/archive/article_id368.php
2. O-GIS - http://app.o-gis.org/o-gis/web/app.php/
3. Tunkelang D. Faceted Search / ed. By G. Marchionini. University of North Carolina. Chapel Hill. 2009. 96p.
4. ElasticSearch – https://www.elastic.co/guide/index.html
5. Open source search server Sphinx – http://www.sphinxsearch.com
6. Open source enterprise search platform Solr – http://lucene.apache.org/solr
7. Open source search engine library Xapian – https://xapian.org
8. High-perfomance, full-featured text search engine library Apache Lucene – https://lucene.apache.org/core
9. Архитектура REST – https://habrahabr.ru/post/38730/
10. AngularJS Framework – https://angularjs.org
11. Императивные и декларативные языки программирования – http://progopedia.ru/paradigm/declarative
12. Платформа Node.js – http://nodejs.ru
13. Package manager Bower – http://bower.io
14. Базы данных и NoSQL – http://squadette.ru/blog/2014/08/12/bazy-dannykh-i-nosql/
15. Elasticsearch-test-data generator - https://github.com/oliver006/elasticsearch-test-data
16. Model-View-Controller design pattern – https://msdn.microsoft.com/en-us/library/ff649643.aspx
17. CRUD операции – http://bourabai.kz/php/crud.htm
18. ElasticUI – http://www.elasticui.com
19. Nix H.A. A biogeographic analysis of Australian Elapid Snakes. In. Atlas of Elapid Snakes of Australia. (ed.) R. Longmore pp. 415. Australian Flora and Fauna Series Number 7. Australian Government Publishing Service: Canberra. 1986.

20. Booth, T.H. (1990) Mapping regions climatically suitable for particular tree species at the global scale. Forest Ecology and Management 36. C. 47-60.
21. Busby, J.R. BIOCLIM - a bioclimatic analysis and prediction system // Margules, C.R. and Austin, M.P. (eds) Nature Conservation: Cost Effective Biological Surveys and data Analysis. Melbourne, 1991. C. 64-68.
22. CORS specification – http://www.w3.org/TR/cors
23. AngularJS directives for ElasticSearch - http://www.elasticui.com
24. REST API tutorial – http://www.restapitutorial.com/lessons/whatisrest.html
25. AngularJS Tutorial by w3schools – http://www.w3schools.com/angular
26. Перевод AngularJS Tutorial на русский язык – http://angular-doc.herokuapp.com
27. Индексы реляционных БД – http://www.sql.ru/articles/mssql/03013101indexes.shtml
28. Г.В. Тюрин. Опыт возрождения русских деревень. М.: Поколение, 2007. 240 с.
29. Описание nosql баз данных и движков - http://nosql-database.org
30. JSONP - https://learn.javascript.ru/ajax-jsonp
31. MV Шаблоны проектирования веб приложений – https://habrahabr.ru/post/151219/

Приложение 1. Рекомендуемые энциклопедические статьи

1. http://ru.wikipedia.org/wiki/ACID
2. http://ru.wikipedia.org/wiki/REST
3. http://ru.wikipedia.org/wiki/Индекс_(базы_данных)
4. http://ru.wikipedia.org/wiki/Декларативное_программирование
5. http://ru.wikipedia.org/wiki/Шаблон_проектирования
6. http://ru.wikipedia.org/wiki/Model-View-Controller
7. http://ru.wikipedia.org/wiki/NoSQL
8. http://en.wikipedia.org/wiki/Faceted_classification
9. http://ru.wikipedia.org/wiki/Тестирование_программного_обеспечения
10. http://ru.wikipedia.org/wiki/Нагрузочное_тестирование
11. http://ru.wikipedia.org/wiki/Стресс-тестирование