Правительство Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
**«Национальный исследовательский университет
«Высшая школа экономики»**

Факультет информатиционных технологий и вычислительной техники

Вычислительные машины, комплексы, системы и сети

кафедра Информационно-коммуникационных технологий

ДИПЛОМНЫЙ ПРОЕКТ

Разработка модуля управления географическими объектами для веб-системы в ЗАО «Флант»

Выполнил
Студент группы С-104
Гнусков Сергей Вадимович

Научный руководитель
Старший преподаватель
Курилов Игорь Дмитриевич

Консультант
Генеральный директор ЗАО «Флант»
Столяров Дмитрий Олегович

Москва, 2013

Аннотация

Данный дипломный проект посвящён разработке модуля для веб-системы, позволяющего структурировать её работу по географическим зонам, а так же предоставляющий максимальное упрощение и автоматизацию поиска адреса в данной географической зоне.

Оглавление

Введение

Актуальность. Автоматизация такси это шаг, на который сейчас идут многие службы такси. Объясняется это тем, что рынок такси постоянно пополняется организациями, оказывающими услуги по перевозке пассажиров, а значит, чтобы занять на нем своё место, необходимо максимально полно отвечать его требованиям и оперативно реагировать на происходящие изменения.

Автоматизация такси даёт следующие преимущества:

- возможность использования в работе стороннего оборудования (АТС, мобильные телефоны, электронные карты и др.);
- повышение производительности труда (распределение обязанностей персонала, увеличение скорости и количества обработки заказов, эффективное использование транспортных средств);
- оперативность работы (нет каждодневного подсчета заявок и процентов в журналах, информация о водителях в любое время);
- отслеживание в реальном времени за работой персонала и службы в целом;
- отчетность за любой период времени по заданным параметрам;
- распределение заказов согласно очередности водителей на стоянках (решение разногласий между водителями);
- сохранность базы данных в течение длительного промежутка времени;
- экономия денежных средств на канцтоварах;
- отсутствие беспорядка в диспетчерской (нет разбросанных листочков бумаги, многотомных журналов).

Автоматизация диспетчерской службы такси позволит вывести бизнес на новую ступень развития, сэкономить средства, структурировать статистические данные и максимально продуктивно использовать рабочее время сотрудников.

Целью данного дипломного проекта является повышение эффективности работы фирмы, предоставляющей услуги такси, путём снижения временных затрат на составление заказа, а также путём структурирования её деятельности по географическим зонам.

Для достижения этой цели требуется решить следующие **задачи**:

- Выявить требования к модулю управления географическими объектами на основе общих задач к системе;
- Провести обзор существующих решений;
- Разработать модуль, удовлетворяющий выявленным требованиям;
- Внедрить разработанный модуль в общую систему.

Практическая значимость данного проекта подтверждена успешным тестированием функций модуля операторами и диспетчерами такси, отметившими значительный прирост эффективности и удобства их работы по сравнению с ранее использовавшимися средствами.

Апробация работы. Разработанный модуль внедрён и используется в системе, находящейся на тестировании в ООО «Сити-Мобил» (Акт о внедрении).

1. Постановка задачи

1.1. Основные задачи системы

Разрабатываемая система предназначается для максимальной автоматизации работы фирмы, предоставляющей услуги такси:

- управление сотрудниками, водителями;
- создание, управление, трекинг заказа;
- создание и редактирование тарифов, услуг и сервисов;
- биллинг;
- управление отношениями с партнёрами.

1.2. Задачи разрабатываемого модуля

Для компании, предоставляющей такси в различных городах, необходимо структурировать всю свою деятельность относительно местности, в которой происходит предоставление услуг. Например, в зависимости от города могут отличаться тарифы, услуги.

Для упрощения и ускорения процесса создания заказа необходимо максимально быстро и точно вводить адреса, называемые клиентом, в систему.

Исходя из данных требований, можно выделить основные задачи разрабатываемого модуля:

- структурировать местность по географическим зонам;
- предоставить удобный поиск адресов назначения внутри географической зоны с автодополнением;
- предоставить водителям ориентиры для быстрого понимания местонахождения адреса подачи и назначения и максимально автоматизировать их поиск.

2. Обзорно-аналитическая часть

2.1. Анализ существующих аналогов

В связи с популярностью такого рода деятельности, как предоставление транспортных услуг, существует множество систем, так или иначе её упрощающих.

Для анализа было выбрано несколько таких программных решений, реализующих полный спектр необходимых функций:

- создание и управление заказами;
- управление клиентами;
- управление водителями (экипажами);
- управление тарифами и др.

В них сравнивалась реализация выявленных требований к разрабатываемому модулю.

2.1.1. Такси Мастер

Сайт: http://www.taximaster.ru/

Программа Такси Мастер предназначена для автоматизации работы служб такси и таксопарков. Она имеет демо-версию, что облегчило анализ её функций.

Управление географическими объектами в данной программе ведётся с помощью справочников: городов, районов, улиц, пунктов и стоянок. В предоставляемой с демо-версией базе данных доступна только Москва, все её улицы, а также несколько стоянок (рис. 2.1).

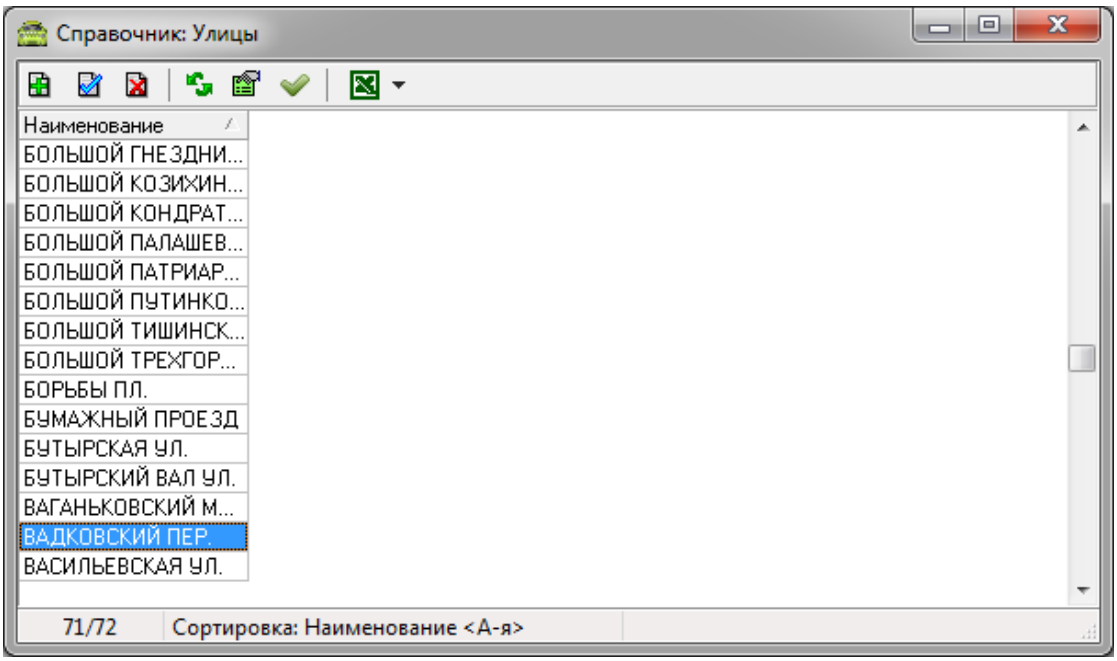

Рисунок 2.1 – Справочник улиц в «Такси Мастер»

База улиц заполняется автоматически при загрузке карты.

Районы задаются точками и могут быть произвольной формы (рис. 2.2).

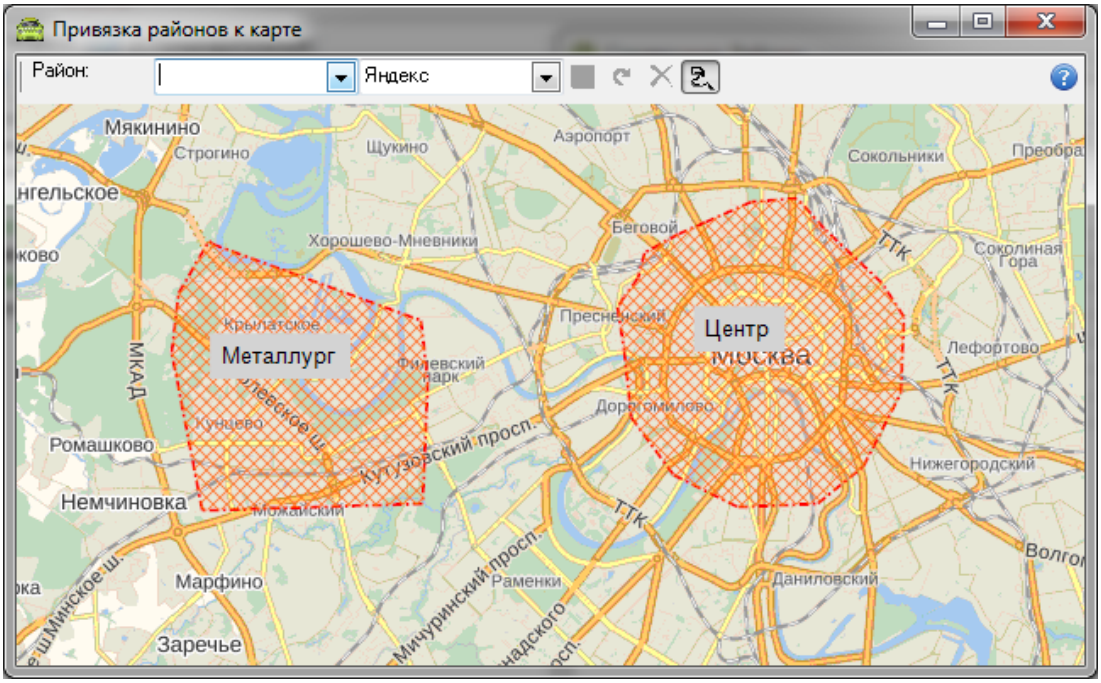

Рисунок 2.2 – Привязка районов к карте в «Такси Мастер»

Поддерживается работа с несколькими видами карт: загруженные, Яндекс.Карты, Google.Maps, карты 2GIS. У района есть несколько полей настроек (рис. 2.3).

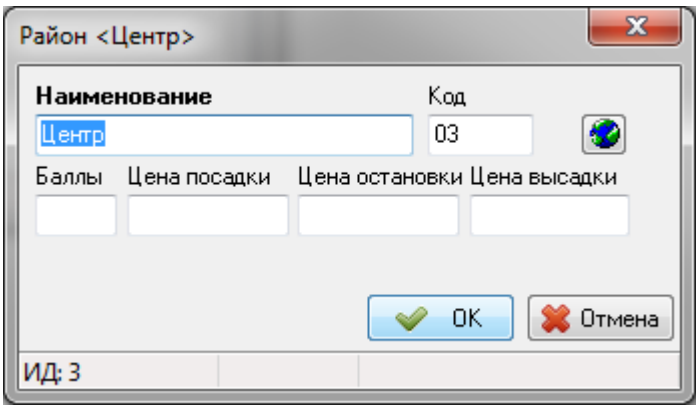

Рисунок 2.3 – Редактирование района в «Такси Мастер»

Более очевидного способа разделения на ценовые зоны обнаружено не было.

При вводе адреса присутствует автодополнение по справочнику улиц (рис. 2.4).

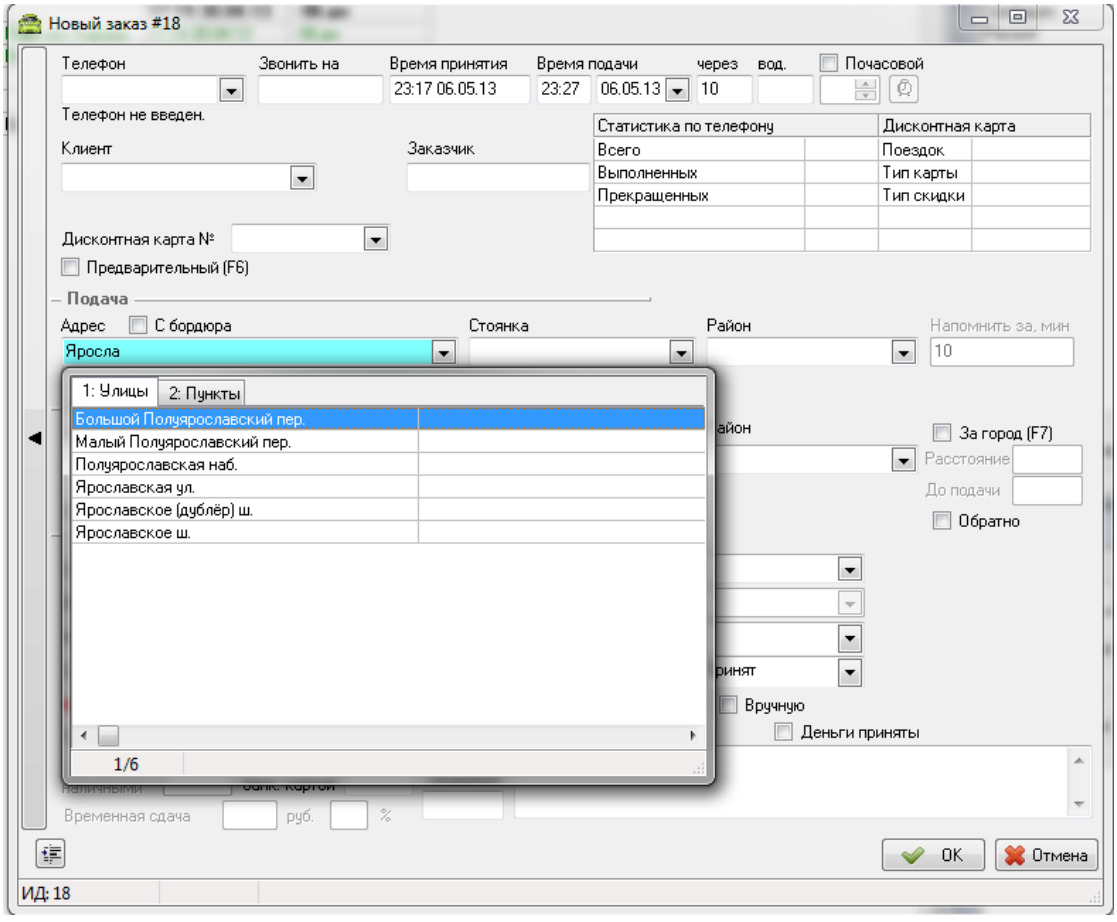

Рисунок 2.4 – Автодополнение в форме заказа в «Такси Мастер»

После ввода адреса происходит геокодирование, после чего, если заданный адрес попадает в один из известных районов, автоматически

заполняется поле «Район». Адреса можно вводить, указав точку на карте. После задания точек строится маршрут (рис. 2.5).

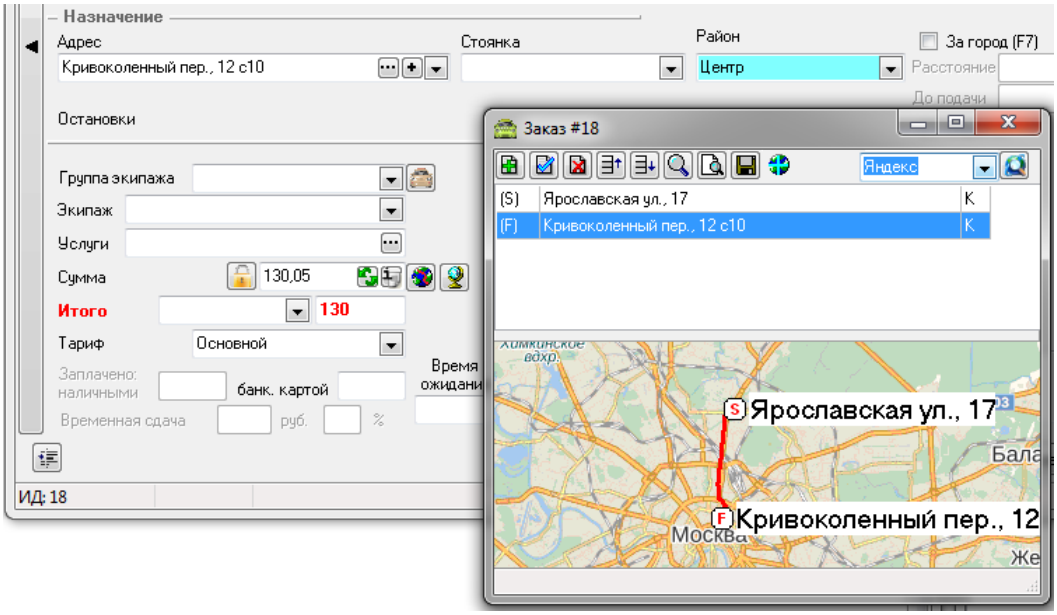

Рисунок 2.5 – Построенный маршрут в «Такси Мастер»

Как видно, Такси Мастер реализует большую часть требований, предъявляемых к разрабатываемому модулю, однако не всё является очевидным, и есть претензии к удобству данной реализации.

2.1.2. Infinity Taxi

Сайт: http://www.taxi-infinity.ru/

Демо-версия данной программы не предоставляется, в связи с чем трудно оценить необходимые возможности. Однако заявляется разделение местности на ценовые зоны с помощью графического редактора (рис. 2.6).

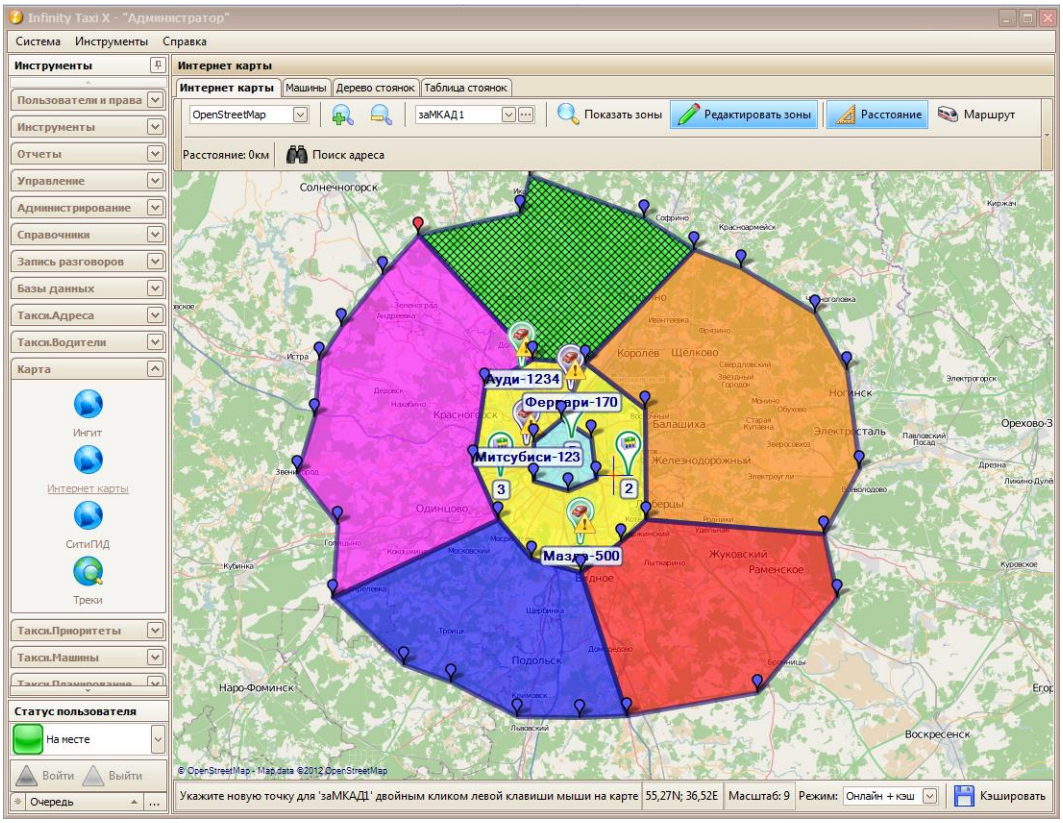

Рисунок 2.6 – Графический редактор ценовых зон в «Infinity Taxi»

2.1.3. Мадив «Интерактивное такси»

Сайт: http://www.madiv.ru/

Аналогично предыдущему случаю, демо-версии данной системы не существует, но так же сказано про возможность разбиения местности на зоны с помощью графического редактора (рис. 2.7).

Рисунок 2.7 – Редактор зон в «Интерактивном такси»

В дальнейшем зоны можно использовать для построения фиксированных тарифов (поездки в аэропорт, вокзал или из одного района в

другой), а так же для настройки области применения тарифа (в городе, за городом и т.д.).

2.1.4. Выводы

В связи с закрытостью большинства из имеющихся решений, не удаётся оценить их возможности в полной мере. Тем не менее понятно, что все они так или иначе реализуют большинство необходимых требований. Это говорит о необходимости данных функций для конкуренции с существующими решениями. Так же нужны и вещи, отличающие систему от остальных. Например, нигде не было найдено такой возможности, как нахождение адресов с опечатками.

2.2. Выбор программных компонентов

2.2.1. Программные компоненты системы

Для разработки общей системы был выбран веб-фреймворк Ruby On Rails, являющийся на данный момент одним из самых популярных средств для разработки крупных веб-проектов.

В качестве СУБД используется MySQL как самая популярная и простая в использовании, при этом хорошо показывающая себя при серьёзных нагрузках. Так же она отлично поддерживается в Ruby On Rails, используя самый популярный ORM - ActiveRecord.

Для обеспечения удобства разработки используется система контроля версий GIT.

Для ускорения реализации веб-интерфейса в системе использован css-фреймворк Twitter Bootstrap.

2.2.2. Программные компоненты разрабатываемого модуля

В процессе поиска готовых компонентов, реализующих задачи модуля, были найдены решения, которые можно было бы использовать в процессе разработки.

2.2.2.1. Geokit + geokit-rails3

Гем Ruby on Rails, предоставляющий приложению функции для работы с геоданными:

- геокодирование (нахождение координат по адресу), поддержка основных провайдеров (Google, Yandex, Yahoo);
- расчёт расстояния по имеющимся координатам;
- интеграция в ORM (ActiveRecord) для обеспечения поиска по базе данных с учётом расстояния;
- получение местоположения по IP-адресу клиента.

Недостатками данного гема является его слабая поддержка и недостаточный функционал. Первоначально он был написан для более старой версии Ruby On Rails, а его портирование на новую версию не было завершено.

2.2.2.2. Geocoder

Во многом схожий с предыдущим, но предоставляющий больше возможностей:

- обратное геокодирование (получения адреса по координатам);
- поддержка большего количества ORM;
- более гибкая настройка;
- поддержка большего количества провайдеров геокодирования.

2.2.2.3 Яндекс.Карты

Самый известный провайдер, предоставляющий геокодирование в России. Позволяет получать координаты по адресу, адрес по координатам, производить поиск ближайших объектов определённого типа (метро, населённый пункт и т.д.).

Яндекс.Карты предоставляют функциональное API, позволяющее рисовать карты на странице, осуществлять прямое и обратное геокодирование, строить маршруты с учётом пробок и без. Очень удобным является автодополнение адресов, реализованное на http://maps.yandex.ru (рис. 2.8).

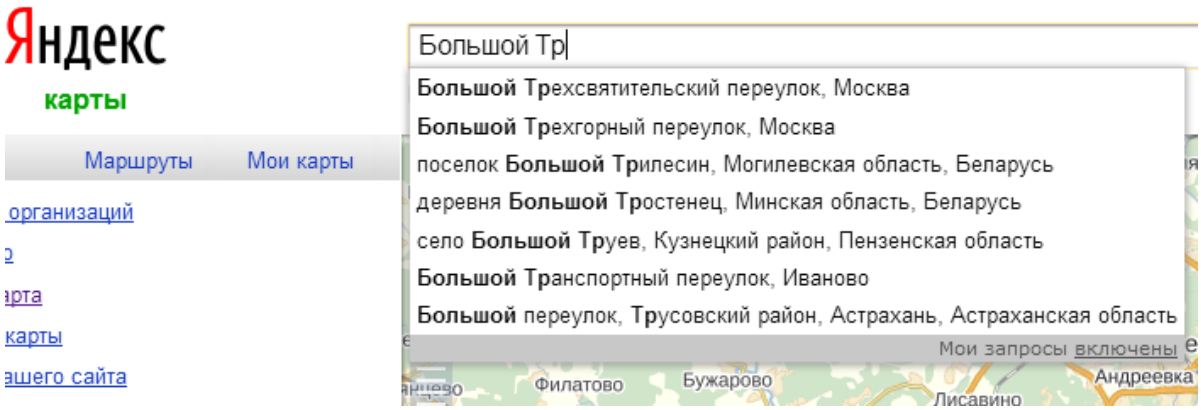

Рисунок 2.8 – Автодополнение на Яндекс.Картах.

Но, к сожалению, эта функция на данный момент не является общедоступной.

2.2.2.4. База данных Федеральной Информационной Адресной Системы

ФИАС является систематизированным сводом актуальных адресных сведений, истории их изменения. Она предоставляет для свободного скачивания полную базу адресов России в форматах DBF и XML.

База данных ФИАС хорошо структурирована, является иерархической: от крупных областей вплоть до домов.

2.2.2.5. Sphinx geo search

Sphinx (англ. SQL Phrase Index) — система полнотекстового поиска, разработанная Андреем Аксеновым и распространяемая по лицензии GNU GPL. Отличительной особенностью является высокая скорость индексации и поиска, а также интеграция с существующими СУБД (MySQL, PostgreSQL) и API для распространённых языков веб-программирования

(официально поддерживаются PHP, Python, Java; существуют реализованные сообществом API для Perl,Ruby,.NET и C++).

Это одна из самых популярных систем поиска, главными особенностями которой можно назвать простоту использования и высокую скорость индексации и поиска, даже при больших объемах данных.

Одной из возможностей данной системы является гео-поиск: поиск элементов с учётом расстояния по координатам.

2.2.2.6. Вывод

Таблица 1

Сравнение найденных решений

	geokit	geokoder	Яндекс.Карты	ФИАС + Sphinx
полнотекстовый поиск адреса	-	-	±	+
геокодирование	+	+	+	-
поиск ближайшего	+	+	+	+
отсутствие зависимости от стороннего сервиса	-	-	-	+

Для реализации разрабатываемого модуля необходимо мощное решение, имеющее такие возможности, как:

- быстрый полнотекстовый поиск адреса (с автодополнением и исправлением опечаток);
- геокодирование;
- поиск объекта, ближайшего к данным координатам.

Как видно из таблицы 1, из найденных решений ни одно в полной мере не предоставляет необходимые возможности. В связи с этим было решено объединить возможности нескольких из них и решить задачи, используя ФИАС+Sphinx для реализации полнотекстового поиска адреса и поиска ближайших объектов и Яндекс.Карты для геокодирования. Такое решение позволило создать свою систему, имеющую собственную

базу адресов и использующую максимум возможностей Sphinx по скорости и точности поиска с минимальной зависимостью от стороннего сервиса, который, в случае необходимости, не составит труда поменять.

3. Технологическая часть

3.1. Обзор паттерна MVC

Фреймворк Ruby On Rails, выбранный для реализации поставленной задачи, построен по паттерну MVC. MVC (Model-View-Controller) является паттерном написания приложений и состоит из 3 основных компонентов (рис. 3.1):

- Model (Модель) – это непосредственно чистые данные и методы их обработки. Данные изменяют свое состояние под действием другого компонента – контроллера.
- View (Представление, вид) – этот компонент отвечает за визуальное отображение данных. Реагирует на изменение модели, изменяя отображение.
- Controller (Контроллер) – это компонент, который взаимодействует с пользователем и как-то реагирует на них, используя представление и модель.

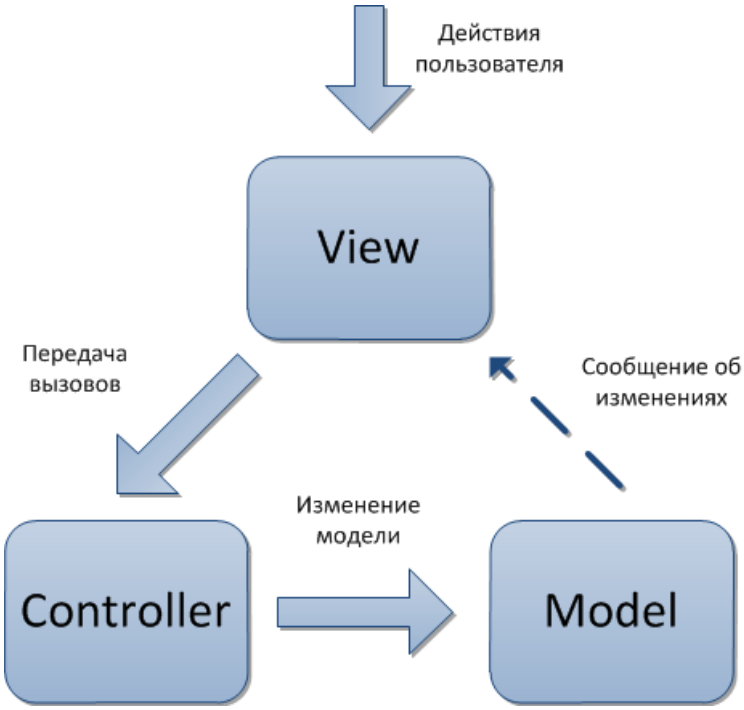

Рисунок 3.1 –Структура MVC

Каждый компонент является модульным, так что модификация каждого компонента может осуществляться независимо.

3.2. Обзор фреймворка Ruby On Rails

3.2.1. Обзор

Ruby on Rails — это среда выполнения, облегчающая разработку, внедрение и обслуживание веб-приложений. За месяцы, прошедшие с ее начального выпуска, Rails прошла путь от малоизвестной забавной технологии до феномена мирового масштаба. Она удостоилась наград, но, что более важно, стала именно той средой выполнения, которую стали выбирать для создания широкого круга называемых *приложений Веб 2.0*. И это не только модное веяние среди утонченных знатоков: Rails используется для создания веб-приложений многими интернациональными компаниями.

И этому есть множество причин.

В первую очередь, создается впечатление, что многие разработчики разочаровались в тех технологиях, которые применяются ими для создания веб-приложений. И похоже, что дело совсем не в том, что именно они используют — Java, PHP или .NET, — у них накапливается ощущение излишней трудоемкости их работы. А затем в один прекрасный момент пришла Rails, с которой работать стало намного проще.

Но сама по себе простота не означает упрощенность. Речь идет о профессиональных разработчиках, создающих реально востребованные веб-сайты. Им хочется видеть созданные ими приложения выдержавшими испытание временем — спроектированными и разработанными с использованием современных, профессиональных технологий. Поэтому разработчики занялись Rails всерьез и обнаружили, что она пригодна не только для разработки веб-сайтов.

К примеру, все Rails-приложения выполняются с использованием архитектуры Модель-Представление-Контроллер (Model-View-Controller, MVC). Привычная Java-разработчикам среда выполнения, к примеру Tapestry или Struts, тоже основана на MVC. Но Rails идет в использовании MVC еще дальше: при разработке Rails-приложений в каждом фрагменте кода остается возможность для полного взаимодействия. Похоже на то, что вы начинаете работу, имея уже готовую структуру приложения.

Rails-приложения пишутся на Ruby – современном интерпретируемом объектно-ориентированном языке. Краткость кода Ruby не влияет на его понятность - свои идеи на этом языке можно выражать вполне четко и естественно. В результате чего программы легко пишутся и (что не менее важно) по прошествии нескольких месяцев вполне легко читаются.

Rails использует все возможности Ruby, являясь его оригинальным расширением, облегчающим жизнь программистов. Программы становятся более короткими и читаемыми.

Разработчики, перешедшие на использование Rails, находят в ней еще и мощную философскую подоплёку. Замыслом Rails преследовалось соблюдение двух ключевых концепций: DRY и соглашения по конфигурации. DRY означает: «Don't Repeat Yourself», то есть «Не допускайте повторений» – любая порция сведений в системе должна быть отображена лишь в одном месте. Для воплощения этой концепции в жизнь Rails использует ту мощь, которая предоставляется ей языком Ruby. В Rails-приложениях дубликаты крайне редки; необходимые сведения приводятся в одном месте, которое часто предлагается в соответствии с соглашениями, существующими в архитектуре MVC, и далее об этом уже можно не волноваться. Для программистов, привыкших к работе с другой средой выполнения веб-приложений, в которой простые поправки в

схеме влекут за собой полудюжину или около того изменений в программном коде, это было настоящим открытием.

Весьма важными являются и соглашения по конфигурации. Их суть заключается в том, что у Rails есть установки по умолчанию, имеющие значения почти для всех аспектов, обеспечивающих целостность приложения. Если следовать соглашениям, можно написать Rails-приложение, используя меньше программных строк, чем имеется в обычном веб-приложении на Java, использующем XML-конфигурацию.

Разработчики озабочены также внедрением своих продуктов. И тут обнаруживается, что с Rails можно распространять удачную версию приложения на любое число серверов всего лишь одной командой (и также легко вернуть все назад, если версия окажется не вполне удачной).

Rails была выделена из реального коммерческого приложения. Оказалось, что лучшим способом создания среды выполнения является определение основных составляющих конкретного приложения, а затем занесение их в общий фонд кода. При разработке Rails-приложения с самого начала уже имеется половина по-настоящему хорошего приложения.

3.2.2. ActiveRecord

ActiveRecord — это надстройка, предоставляемая средой Rails для обеспечения объектно-реляционного отображения (ORM). ActiveRecord строго придерживается стандартной ORM-модели: таблицы отображаются в классы, строки — в объекты, а столбцы — в свойства этих объектов. Она отличается от множества других ORM-библиотек способом конфигурирования. Использование в ActiveRecord оптимальных установок по умолчанию позволяет свести к минимуму количество настроек, используемых разработчиками.

3.2.3. Unicorn

Unicorn – это HTTP-сервер для Rack-приложений. Основные возможности:

- разработан для Rack, Unix, лишён всего, что уже реализовано там;
- управление процессами: unicorn автоматически использует несколько процессов для обработки запросов, при необходимости перезагружая и убивая ненужные;
- распределение нагрузки происходит посредством ядра операционной системы, запросы никогда не накапливаются на загруженном процессе;
- не требуется строгой потокозащищённости, процессы запускаются в изолированном адресном пространстве.

Unicorn позволяет использовать быстрый многопоточный сервер, не требующий тонкой настройки.

3.3. EventMachine

EventMachine — быстрый и легкий фрэймворк для сетевого взаимодействия в Ruby. В EventMachine используется событийно-ориентированный (асинхронный) механизм обработки сетевых соединений. Она разработана для одновременной реализации двух нужд:

- высокая масштабируемость, производительность и стабильность для самых требовательных систем;
- API, который устраняет сложности высокопроизводительного многопоточного программирования сетевого взаимодействия, позволяя сконцентрироваться на логике приложения.

Сочетание данных возможностей делает EventMachine отличным выбором для реализации сложной сетевой системы.

3.4. Обзор возможностей Sphinx

3.4.1. Основные преимущества

- система для организации полнотекстового поиска;
- бесплатность;
- открытость;
- изначально разработана для интеграции с БД;
- документы Sphinx аналогичны записям в базе данных:
    - документ это набор текстовых полей и численных атрибутов плюс уникальный ID – аналог строки в таблице БД;
    - набор полей и атрибутов не меняется в пределах индекса – аналог таблицы в БД;
    - по полям можно вести полнотекстовый поиск;
    - по атрибутам можно дополнительно фильтровать, сортировать, группировать результаты поиска;
- множество видов источников данных:
    - встроенные «драйвера» для MySQL, Postgres, XML файлов специального формата;
    - источники разного типа можно комбинировать в одном индексе;
    - данные для индексации поступают не из таблицы, а из выборки, т.е. можно делать дополнительную обработку в момент выборки – JOIN-ы, выбирать любой поднабор полей, и т.д.

3.4.2. Дополнительные возможности

- поиск и ранжирование с учетом позиций слов и близости фраз;

- произвольные атрибуты, в том числе MVA (multi-value attribute, атрибут с несколькими значениями);
- сортировка, фильтрация, группировка;
- выдержки с подсветкой ключевых слов;
- распределенный поиск;
- оптимизация «близких» запросов;
- географическое расстояние (geo-distance).

3.5. Фонетические алгоритмы

Для реализации поиска с опечатками по текстовым полям, чаще всего используют фонетические алгоритмы. Они сопоставляют двум словам со схожим произношением одинаковые коды, что позволяет осуществлять сравнение и индексацию множества таких слов на основе их фонетического сходства.

Рассмотрим основные фонетические алгоритмы.

3.5.1. SoundEx и улучшенный SoundEx

Одним из первых фонетических алгоритмов является SoundEx, придуманный ещё в начале прошлого века Робертом Расселом. В версии этого алгоритма для английского языка производится сопоставление слова с кодом вида K321. В основе его работы стоит разбиение согласных букв на группы, из которых затем и составляется результирующее значение. Позднее был предложен улучшенный вариант алгоритма.

Таблица 2

Таблица преобразований SoundEx

Символ слова	**Код символа**
B P F V	1
C S K G J Q X Z	2
D T	3
L	4
M N	5

R	6

Таблица 3

Таблица преобразований улучшенного SoundEx

Символ слова	Код символа
B P	1
F V	2
C K S	3
G J	4
Q X Z	5
D T	6
L	7
M N	8
R	9

Первым символом кода является первый символ слова, последующие буквы слова сопоставляются цифрам по таблице (таблица 2 для SoundEx и таблица 3 для улучшенного SoundEx соответственно). Все символы, которых нет в таблице, игнорируются. Если соседние символы или символы, разделённые буквами H или W, попадают в одну группу, они записываются как один. Если результат больше 4 символов, он обрезается, если меньше — дополняется нулями так, чтобы в итоге получилось 4 символа. Таким образом, после всех преобразований остаётся около 7000 различных вариантов такого кода, в связи с чем существует множество непохожих слов, имеющих один код. Получается, результат очень часто включает в себя большое количество «ложноположительных» значений.

В улучшенной версии, как можно заметить из таблицы 3, буквы разбиты на большее количество групп. Помимо этого, игнорируются буквы H и W и нет ограничений по длине.

Примеры

Оригинальный Soundex:

- picture → P236 (pastern, Pasteur, pastier, pastors)
- ball → B400 (Baal, bail, bale, Bali)
- guitar → G360 (gadder, gaiter, gather, getter)

3.5.2. Daitch-Mokotoff Soundex

Этот алгоритм в 1985 году был разработан двумя генеалогами — Гарри Мокотофф и Рэнди Дэйч, стремясь улучшить оригинальный SoundEx для более точной работы с восточно-европейскими фамилиями.

Этот алгоритм серьёзно отличается от оригинального SoundEx, его результатом является последовательность цифр.

Он имеет значительно более сложные правила сопоставления — теперь в формировании результирующего кода участвуют не только одиночные символы, но и последовательности из нескольких символов. Результат работы алгоритма обеспечивает более 100 тысяч различных вариаций кода. Это, а так же усложнённые правила алгоритма, уменьшают количество «ложноположительных» слов, приходящихся на конкретный код.

Преобразования осуществляются по таблице 4 (порядок преобразований соответствует порядку буквосочетаний в таблице).

Таблица 4

Таблица преобразований Daitch-Mokotoff Soundex

Исходные буквосочетания	В начале	За гласной	Остальное
AI, AJ, AY, EI, EY, EJ, OI, OJ, OY, UI, UJ, UY	0	1	
AU	0	7	
IA, IE, IO, IU	1		
EU	1	1	
A, UE, E, I, O, U, Y	0		
J	1	1	1
SCHTSCH, SCHTSH, SCHTCH, SHTCH, SHCH, SHTSH, STCH,	2	4	4

STSCH, STRZ, STRS, STSH, SZCZ, SZCS			
SHT, SCHT, SCHD, ST, SZT, SHD, SZD, SD	2	43	43
CSZ, CZS, CS, CZ, DRZ, DRS, DSH, DS, DZH, DZS, DZ, TRZ, TRS, TRCH, TSH, TTSZ, TTZ, TZS, TSZ, SZ, TTCH, TCH, TTSCH, ZSCH, ZHSH, SCH, SH, TTS, TC, TS, TZ, ZH, ZS	4	4	4
SC	2	4	4
DT, D, TH, T	3	3	3
CHS, KS, X	5	54	54
S, Z	4	4	4
CH, CK, C, G, KH, K, Q	5	5	5
MN, NM		66	66
M, N	6	6	6
FB, B, PH, PF, F, P, V, W	7	7	7
H	5	5	
L	8	8	8
R	9	9	9

Возможны альтернативные варианты буквосочетаний (это позволяет генерировать несколько кодов):

CH → TCH

CK → TSK

C → TZ

J → DZH

Примеры

095747 → Архипцев, Архипцов, Архипычев, Арчибасов

095757 → Архипков, Архипцев, Архипцов, Архипычев

584360 → Галстян, Галустян, Гильштейн, Глистин, Гольдштейн, Калустьян, Хлистун, Хлыстун, Хлюстин.

В среднем, один и тот же код может отображать 5 фамилий.

3.5.3. Metaphone

Несколько лучшими характеристиками обладает алгоритм Metaphone, разработанный в 1990 году, отличающийся от предыдущих алгоритмов иным подходом к процессу кодирования: преобразование слова происходит с учетом правил английского языка, используя заметно более сложные правила, так что при этом гораздо меньше информации оказывается потерянной, так как не происходит разбиения букв на группы. Полученный код является набором символов из множества 0BFHJKLMNPRSTWXY, в начале слова могут быть гласные из множества AEIOU.

Алгоритм вычисления кода Metaphone

1. Удаляются все повторяющиеся соседние буквы, за исключением буквы C.
2. Начало слова преобразовывается по правилам:

   KN на N

   GN на N

   PN на N

   AE на E

   WR на R
3. Буква B в конце удаляется, если она идет после M.
4. C заменяется по правилам:

   На X в сочетаниях CIA, SCH, CH;

   На S в сочетаниях CI, CE , CY;

   На K в остальных случаях.
5. D заменяется по правилам:

   На J в сочетаниях DGE, DGY, DGI;

   На T в остальных случаях.
6. Производится замена GH на H, если это буквосочетание стоит не в конце и не перед гласной.

7. Заменяются GN на N и GNED на NED, если эти буквосочетания стоят в конце.
8. G заменяется по правилам:

   На J в сочетаниях GI, GE, GY;

   На K в остальных случаях.
9. Все H, идущие после гласных, но не перед гласными, удаляются.
10. Выполняются следующие преобразования:

    CK на K

    PH на F

    Q на K

    V на F

    Z на S
11. S заменяется на X в сочетаниях SH, SIO, SIA.
12. T заменяется по правилам:

    На X в сочетаниях TIA, TIO;

    На 0 в сочетании TH;

    Удаляется в сочетании TCH.
13. В начале слова сочетание WH переходит в W. Если после W нет гласной, она удаляется.
14. X в начале слова заменяется на S, иначе — на KS.
15. Удаляются все Y, которые находятся не перед гласными.
16. Удаляются все гласные, кроме начальной.

Примеры

AKXN → Агашин, Акаченок, Акишин, Аксионенко, Аксионов, Акчунаев, Акшанов, Акшенцев, Акшинский, Акшинцев, Акшонов.

FSLX → Василишин, Васильчак, Васильченко, Васильчик, Васильчиков, Васильченко, Васильчук, Василющенко.

SRFM → Серафимов, Серафимский, Серафимчук, Церейфман.

Одно и то же значение кода Metaphone имеют в среднем 6 фамилий.

3.5.4. Русский metaphone

В 2002 году в журнале «Программист» Пётр Каньковски опубликовал статью, которая рассказывала о его адаптации английской версии алгоритма Metaphone к русскому языку. Этот алгоритм производит преобразования слова в соответствии с правилами и нормами русского языка, учитывая фонетическое звучание безударных гласных и возможные «слияния» согласных при произношении. Несмотря на том, что он основывается на простых правилах, на практике он показывает очень хорошие результаты. Все буквы разбиты на группы по звучанию — гласные и согласные, глухие и звонкие. Звонкие согласные преобразуются в соответствующие им парные глухие, объединяются «сливающиеся» при произношении последовательности букв, и проводятся некоторые другие манипуляции.

Алгоритм вычисления кода русского Metaphone

1. Для всех гласных букв проделываются следующие замены:

   ЙО, ИО, ЙЕ, ИЕ на И;

   О, Ы, Я на А;

   Е, Ё, Э на И;

   Ю на У;

2. Для всех согласных букв, за которыми следует любая согласная, кроме Л, М, Н или Р, либо для согласных на конце слова, производится оглушение:

   Б заменяется на П;

   З на С;

   Д на Т;

   В на Ф;

   Г на К;

3. ТС и ДС заменяются на Ц.

В итоге, алгоритм очень хорошо справляется со своей задачей — в результирующем наборе содержатся действительно фонетически схожие слова. И при этом остается довольно мало лишних слов, в основном благодаря тому, что гласные не игнорируются, а преобразуются и используются в итоговом коде. Однако же, есть некоторые слова, которые, несмотря на свою фонетическую схожесть, не попадают в результирующий набор из-за слишком «строгих» правил алгоритма.

Примеры

праспикт → проспект, праспект;

парахат → пароход, параход, парохот;

падест → подъезд, падъезд, подъест.

Как видно, этот алгоритм слова, написанные с ошибками, которые можно допустить, воспринимая их на слух, приводит к одному коду, что позволяет использовать его для исправления опечаток.

3.5.5. Выводы

Из рассмотренных алгоритмов русский metaphone наилучшим образом подходит для разработки модуля. При своей простоте он показывает низкий уровень «ложноположительных» значений для русского языка.

3.6. GIT

Git – распределенная система управления версиями файлов с акцентом на простоту и скорость. Изначально создавался Линусом Торвальдсом для использования в процессе разработки ядра Linux. С тех пор он получил серьёзное развитие и используется во многих проектах.

Архитектура Git основана на опыте Линуса в разработке огромного распределённого проекта – ядра Linux, на его знаниях о производительности файловых систем и срочной необходимости реализовать работа-

ющую систему в короткие сроки. Эти влияния привели к следующим решениям:

- **Сильная поддержка нелинейной разработки:** Git поддерживает быстрое создание и слияние веток, имеет специализированные утилиты для просмотра истории;
- **Распределённая разработка:** каждый разработчик имеет полную локальную копию репозитория, где производит изменения и периодически синхронизируется с удалённым;
- **Совместимость с существующими системами и протоколами:** поддерживается публикация репозиториев через HTTP, FTP, rsync или собственный протокол git, работающий поверх ssh или HTTP. Так же есть совместимость с CVS и Subversion;
- **Эффективное обслуживание крупных проектов:** многочисленные тесты показали, что Git на порядок быстрее большинства других распределённых систем управления;
- **Криптографическая авторизация истории:** каждая ревизия имеет уникальный идентификатор и зависит от истории всех предыдущих изменений;
- **Дизайн, основанный на множестве утилит:** Git предоставляет множество программ, написанных на языке Си, и скрипты командной оболочки, являющиеся обёртками к этим программам;
- **Гибкие стратегии слияния:** Git предоставляет возможность автоматического слияния веток, а так же есть множество возможностей решения конфликтов при этом;
- **Периодическая явная упаковка объектов:** каждый новый объект в Git хранится отдельным файлом. Однако, для эффективного использования дискового пространства, периодически происходит так называемая упаковка объектов, когда

множество объектов упаковываются в один файл с помощью дельта-кодирования.

4. Разработка

4.1. Проектирование архитектуры

4.1.1. Архитектура модуля

При анализе задач были выявлены три типа объектов модуля:

- **ориентир** — точка или часть локации. Используется как ближайшее известное/заметное место по отношению к конкретному адресу для ориентирования водителями. Примеры: метро Китай-Город, населённый пункт Домодедово;
- **локация** — географический объект и прилегающие к нему регионы: города, пригородные области, поселки и т.п. Содержит в себе ориентиры. Примеры: Москва, Московская область;
- **географические зона** — географический регион, используемый для определения тарифных зон. Содержит в себе локации. Например, если фирма предоставляет единые услуги такси в Москве и Московской области, то её тарифная зона будет привязана к географической зоне «Москва», содержащей в себе локации «Москва» и «Московская область».

4.1.2. Архитектура ФИАС

Федеральная информационная адресная система (ФИАС) содержит достоверную единообразную и структурированную адресную информацию по территории Российской Федерации, доступную для использования органами государственной власти, органами местного самоуправления, физическими и юридическими лицами.

Данная информация предоставляется в двух форматах: DBF и XML. После изучения структуры файлов выгрузки в каждом из форматов, а также инструментов работы с каждым из них, было принято решение для интеграции в систему использовать формат XML.

Структура файла XML

Таким образом, XML-файл с объектами адресов хранит иерархию со связью с родительским элементом через *AOGUID* и всей необходимой информацией (имя, статус, коды, тип и т.д.) (таблица 5).

Таблица 5

Классификатор адресообразующих элементов (Object)

Наименование элемента	**Сокращенное наименование (код) элемента**	**Признак типа элемента**	**Формат элемента**	**Признак обязательности элемента**
Глобальный уникальный идентификатор адресного объекта	AOGUID	A	T(=36)	O
Формализованное наименование	FORMALNAME	A	T(1-120)	O
Код региона	REGIONCODE	A	T(=2)	O
Код автономии	AUTOCODE	A	T(=1)	O
Код района	AREACODE	A	T(=3)	O
Код города	CITYCODE	A	T(=3)	O
Код внутригородского района	CTARCODE	A	T(=3)	O
Код населенного пункта	PLACECODE	A	T(=3)	O
Код улицы	STREETCODE	A	T(=4)	O
Код дополнительного адресообразующего элемента	EXTRCODE	A	T(=4)	O
Код подчиненного дополнительного адресообразующего элемента	SEXTCODE	A	T(=3)	O
Официальное наименование	OFFNAME	A	T(1-120)	H
Почтовый индекс	POSTALCODE	A	T(=6)	H

Код ИФНС ФЛ	IFNSFL	A	T(=4)	H
Код территориального участка ИФНС ФЛ	TERRIFNSFL	A	T(=4)	H
Код ИФНС ЮЛ	IFNSUL	A	T(=4)	H
Код территориального участка ИФНС ЮЛ	TERRIFNSUL	A	T(=4)	H
ОКАТО	OKATO	A	T(=11)	H
ОКТМО	OKTMO	A	T(=8)	H
Дата внесения записи	UPDATEDATE	A		O
Краткое наименование типа объекта	SHORTNAME	A	T(1-10)	O
Уровень адресного объекта	AOLEVEL	A	N(10)	O
Идентификатор объекта родительского объекта	PARENTGUID	A	T(=36)	H
Уникальный идентификатор записи. Ключевое поле.	AOID	A	T(=36)	O
Идентификатор записи связывания с предыдушей исторической записью	PREVID	A	T(=36)	H
Идентификатор записи связывания с последующей исторической записью	NEXTID	A	T(=36)	H
Код адресного объекта одной строкой с признаком актуальности из КЛАДР 4.0.	CODE	A	T(0-17)	H
Код адресного объекта из КЛАДР 4.0 одной строкой без признака актуальности (последних двух цифр)	PLAINCODE	A	T(0-15)	H
Статус актуальности адресного объекта ФИАС. Акту-	ACTSTATUS	A	N(10)	O

альный адрес на текущую дату. Обычно последняя запись об адресном объекте				
Статус центра	CENTSTATUS	A	N(10)	O
Статус действия над записью – причина появления записи	OPERSTATUS	A	N(10)	O
Статус актуальности КЛАДР 4 (последние две цифры в коде)	CURRSTATUS	A	N(10)	O
Начало действия записи	STARTDATE	A		O
Окончание действия записи	ENDDATE	A		O
Внешний ключ на нормативный документ	NORMDOC	A	T(=36)	H
Признак действующего адресного объекта	LIVESTATUS	A	N(=1)	O

XML-файл с объектами домов хранит их со связью с объектом адреса через *AOGUID* и всей необходимой информацией (номер, статус, индекс, тип и т.д.) (таблица 6).

Таблица 6

Сведения по номерам домов улиц городов и населенных пунктов, номера земельных участков и т.п (House)

Наименование элемента	**Сокращенное наименование (код) элемента**	**Признак типа элемента**	**Формат элемента**	**Признак обязательности элемента**
Почтовый индекс	POSTALCODE	A	T(=6)	H
Код ИФНС ФЛ	IFNSFL	A	T(=4)	H
Код территориального участка ИФНС ФЛ	TERRIFNSFL	A	T(=4)	H
Код ИФНС ЮЛ	IFNSUL	A	T(=4)	H

Код территориального участка ИФНС ЮЛ	TERRIFNSUL	A	T(=4)	H
ОКАТО	OKATO	A	T(=11)	H
ОКТМО	OKTMO	A	T(=8)	H
Дата время внесения записи	UPDATEDATE	A		O
Номер дома	HOUSENUM	A	T(1-20)	H
Признак владения	ESTSTATUS	A	N(1)	O
Номер корпуса	BUILDNUM	A	T(0-10)	H
Номер строения	STRUCNUM	A	T(0-10)	H
Признак строения	STRSTATUS	A	N(10)	O
Уникальный идентификатор записи дома	HOUSEID	A	T(=36)	O
Глобальный уникальный идентификатор дома	HOUSEGUID	A	T(=36)	O
Guid записи родительского объекта (улицы, города, населенного пункта и т.п.)	AOGUID	A	T(=36)	O
Начало действия записи	STARTDATE	A		O
Окончание действия записи	ENDDATE	A		O
Состояние дома	STATSTATUS	A	N(10)	O
Внешний ключ на нормативный документ	NORMDOC	A	T(=36)	H
Счетчик записей домов для КЛАДР 4	COUNTER	A	N(10)	O

4.1.3. Результат

Для реализации такой задачи модуля, как поиск адреса с автодополнением в данной локации, необходимо связать элементы модуля и элементы ФИАС. С точки зрения объектов модуля, наилучшим выбором

будет связь между локацией и объектом адреса ФИАС (например, локация «Москва» свяжется с записью, представляющей объект адреса типа «город» и именем «Москва»). Тогда при поиске адресов можно будет искать только среди тех, в которых родительским элементом является выбранный в локации, а так же ниже по иерархии.

Тогда общая структура будет выглядеть так, как показано на рисунке 4.1.

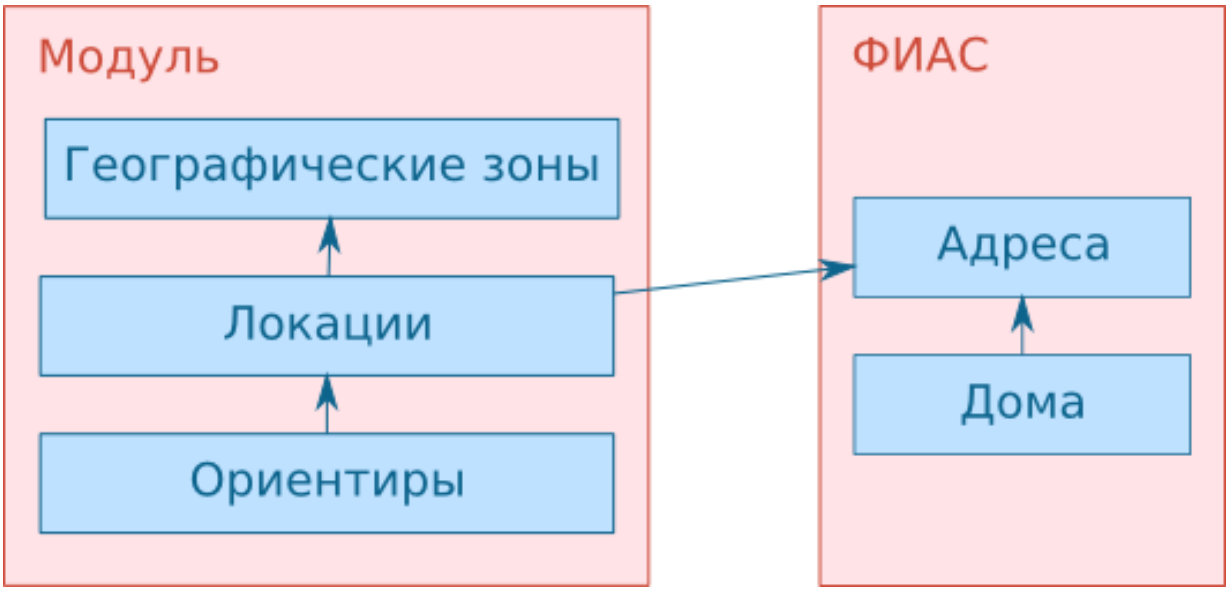

Рисунок 4.1 – Общая структура модуля

4.2. Проектирование базы данных

4.2.1. Общее

Полученная выше структура полностью соответствует принципам реляционной базы данных и может быть использована в качестве структуры БД. Так как данные ФИАС являются вспомогательными для работы модуля, целесообразно вынести их в отдельную базу, не смешивая с данными системы. Таким образом, модуль внесёт три таблицы в БД системы, а также вспомогательную базу данных с адресами и домами ФИАС.

4.2.2. Географическая зона

- *id*: идентификатор географической зоны;

- *name*: имя географической зоны;
- *bounds*: GPS-координаты прямоугольника, заключающего в себя географическую зону (для центрирования карты при работе в ней).

4.2.3. Локация

- *id*: идентификатор локации;
- *geographical_zone_id*: идентификатор географической зоны, к которой принадлежит локация;
- *name*: имя локации;
- *geo_name*: полное географическое название локации;
- *time_zone*: часовой пояс для правильной работы со временем в разных локациях.

4.2.4. Ориентир

- *id*: идентификатор ориентира;
- *location_id*: идентификатор локации, которой принадлежит ориентир;
- *l_type*: тип ориентира (метро, район, населённый пункт, аэропорт и т.д.)
- *name*: имя ориентира;
- *longitude, latitude*: GPS-координаты (долгота, широта) ориентира;
- *parent_id*: идентификатор родительского ориентира (для создания иерархии);
- *is_address*: флаг, является ли ориентир адресом, на который можно заказывать машину (например, станция метро или терминал аэропорта);

- *is_meta*: флаг, что это не ориентир как таковой, а «каталог» для группировки других ориентиров (например, «Аэропорты» или «ЦАО»);
- *not_closest*: флаг, что ориентир не участвует в автоматическом поиске ближайших ориентиров для адреса (например, неправильно, если для посёлка вблизи «Шереметьево» аэропорт будет показываться в качестве ориентира, так как к ним ведут разные дороги).

4.2.5. Адрес ФИАС

- *id*: идентификатор адреса;
- *guid*: глобальный идентификатор адреса в базе данных ФИАС;
- *parent_guid*: глобальный идентификатор родительского адреса;
- *addrtype*: тип адреса в базе данных ФИАС (например, «ул» или «пер»)
- *shortname*: короткое имя (например, «Владимирская улица»);
- *fullname*: полное имя (например, «Ивановская область, Южский район, город Южа, Владимирская улица»);
- *aliases*: нормализованный формат имени для фонетического поиска (например, если допущена опечатка);
- *level*: уровень вложенности;
- *order*: приоритет для поиска.

4.2.6. Дом ФИАС

- *id*: идентификатор дома;
- *guid*: глобальный идентификатор дома в базе данных ФИАС;
- *parent_guid*: глобальный идентификатор родительского адреса;

- *house*: номер дома (с учётом строения, корпуса и т.д.);
- *order*: приоритет для поиска.

4.2.7. Результат

Для связи локация и адресов ФИАС типа «много-ко-многим» была создана дополнительная таблица *location_fias_maps*. В итоге получаем базу данных, представленную на рисунке 4.2.

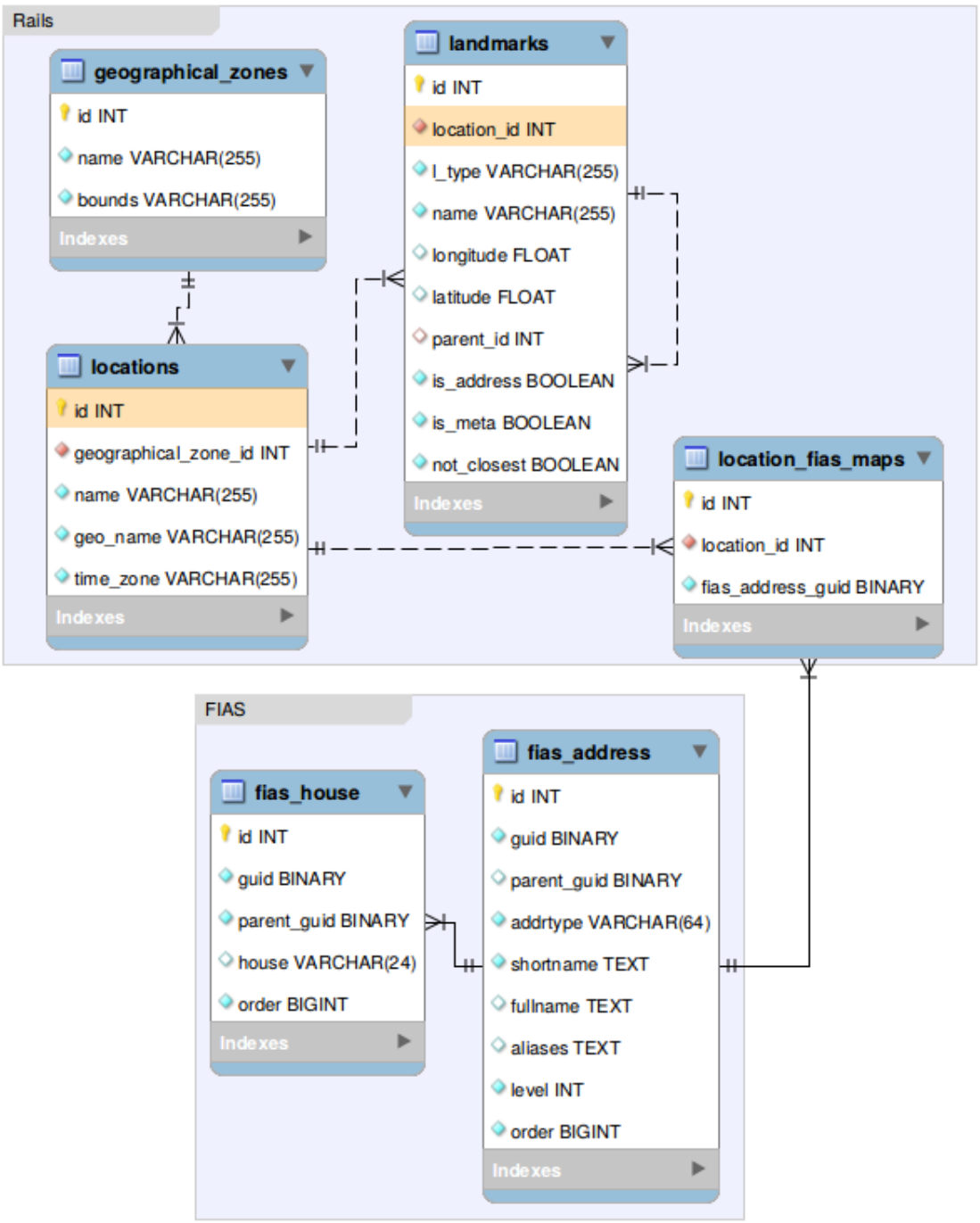

Рисунок 4.2 – Общая структура базы данных модуля

4.3. Разработка компонентов

4.3.1. Конвертация базы данных ФИАС из XML в MySQL

Как сказано выше, было решено использовать базу данных ФИАС в формате XML. А так как в работе модуля используется СУБД MySQL, необходимо перенести необходимые данные из файла XML в базу данных MySQL. Для этого был написан специальный конвертер на языке Python.

В связи с тем, что файлы адресов и домов имеют большой размер, загружать их полностью для конвертации не получится, в связи с чем для разбора XML было решено использовать популярную библиотеку libxml2, предоставляющую возможность постепенной загрузки и разбора данных, что позволило значительно снизить нагрузку.

В процессе конвертации осуществляется выбор только необходимых (описанных в пунктах 4.2.5. и 4.2.6.) данных. Полный адрес получается из сложения имен всех родителей данного адреса. Одновременно с этим производится нормализация адресов.

4.3.2. Нормализация адресов для фонетического поиска

Для нормализации адресов был написан специальный модуль на Python, реализующий рассмотренный в пункте 3.5.4. алгоритм представления текста для фонетического поиска.

4.3.3. Реализация поиска с помощью Sphinx

Было настроено три индекса Sphinx: для поиска по полному адресу дома, для поиска адреса и для поиска адреса по данному глобальному идентификатору (так как guid является текстовым полем, и для поиска по нему необходим индекс).

Для первых двух использован так называемый «инфиксный» (infix) поиск. Это означает, что искомая фраза может находиться внутри текстового поля, и всё равно такая запись будет найдена. Это позволяет

сильно ускорить поиск с помощью отображения в автодополнении необходимых результатов, найденных по неполному значению.

4.3.4. Реализация основной части модуля

Был реализован так называемый CRUD объектов, типичный для Ruby On Rails.

4.3.4.1. Географические зоны

На рисунках 4.3 и 4.4 показано, как выглядит добавление и просмотр географической зоны соответственно.

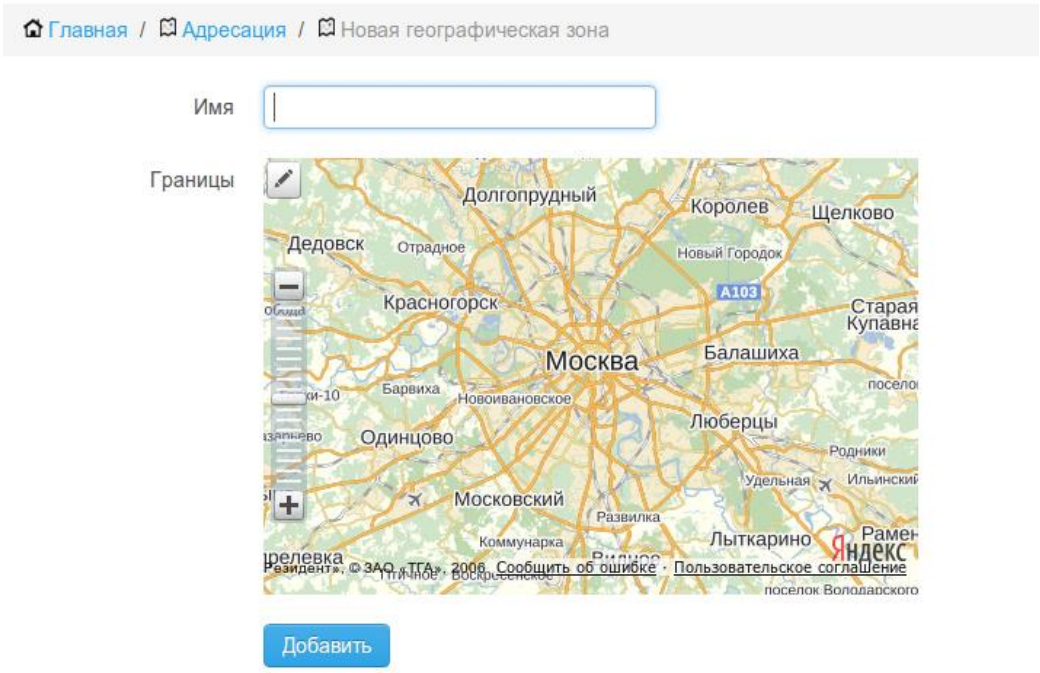

Рисунок 4.3 – Добавление географической зоны

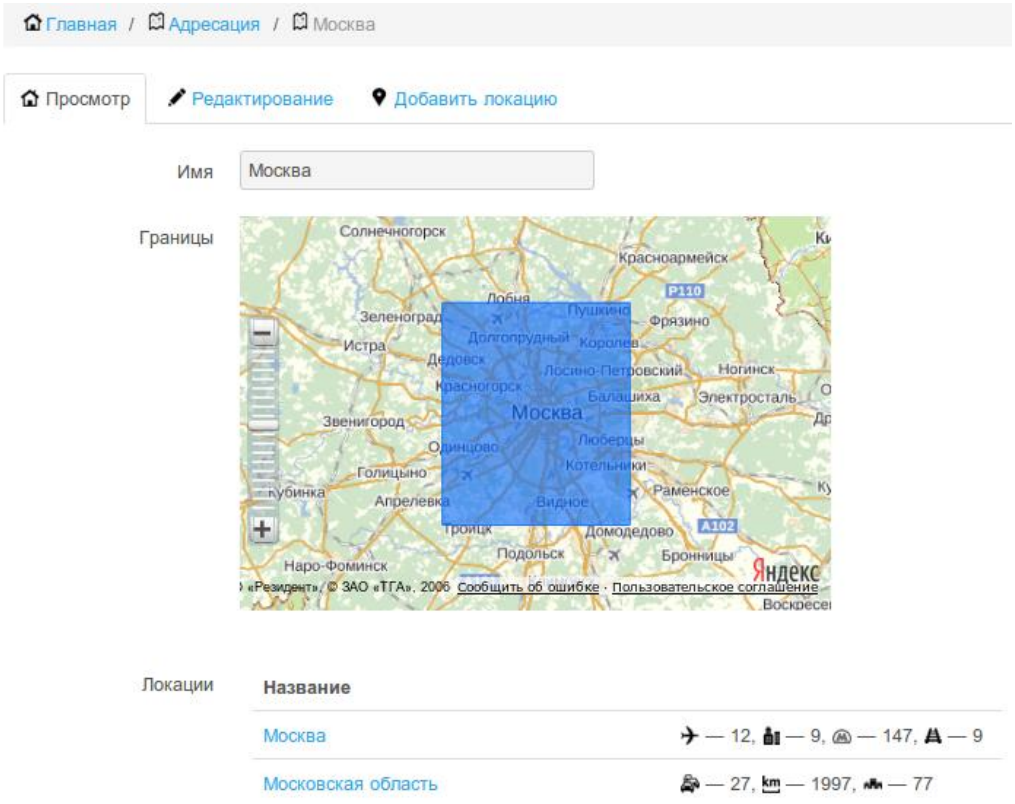

Рисунок 4.4 – Просмотр географической зоны «Москва»

На рисунке 4.5 представлен список географических зон. Для удобства навигации, в нем представлены и локации с помощью таблицы в виде дерева.

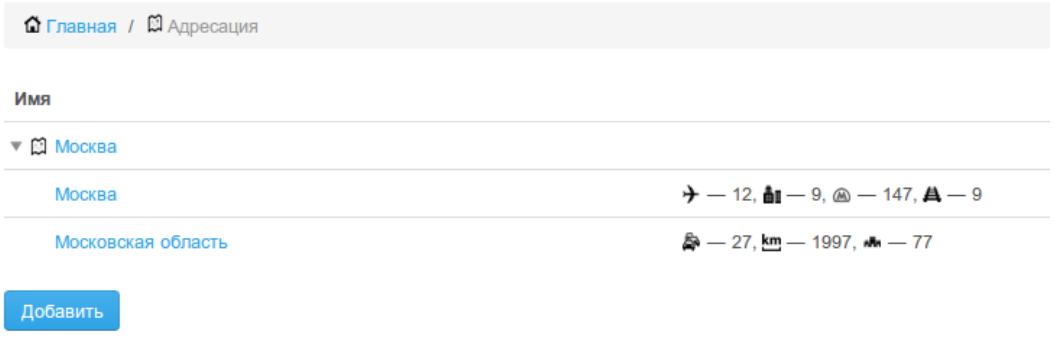

Рисунок 4.5 – Просмотр списка географических зон

4.3.4.2. Локации

На рисунках 4.6 и 4.7 представлены добавление и просмотр локации соответственно.

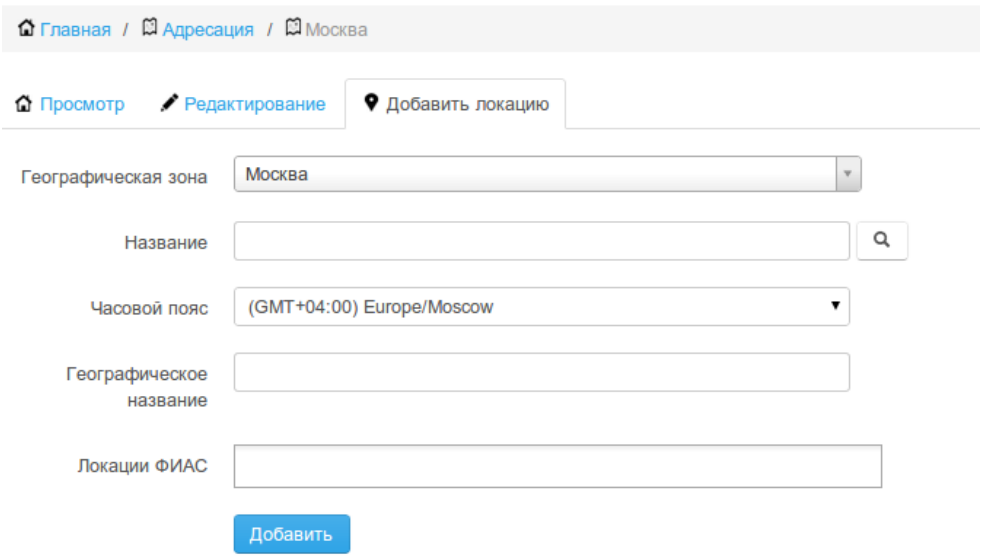

Рисунок 4.6 – Добавление локации

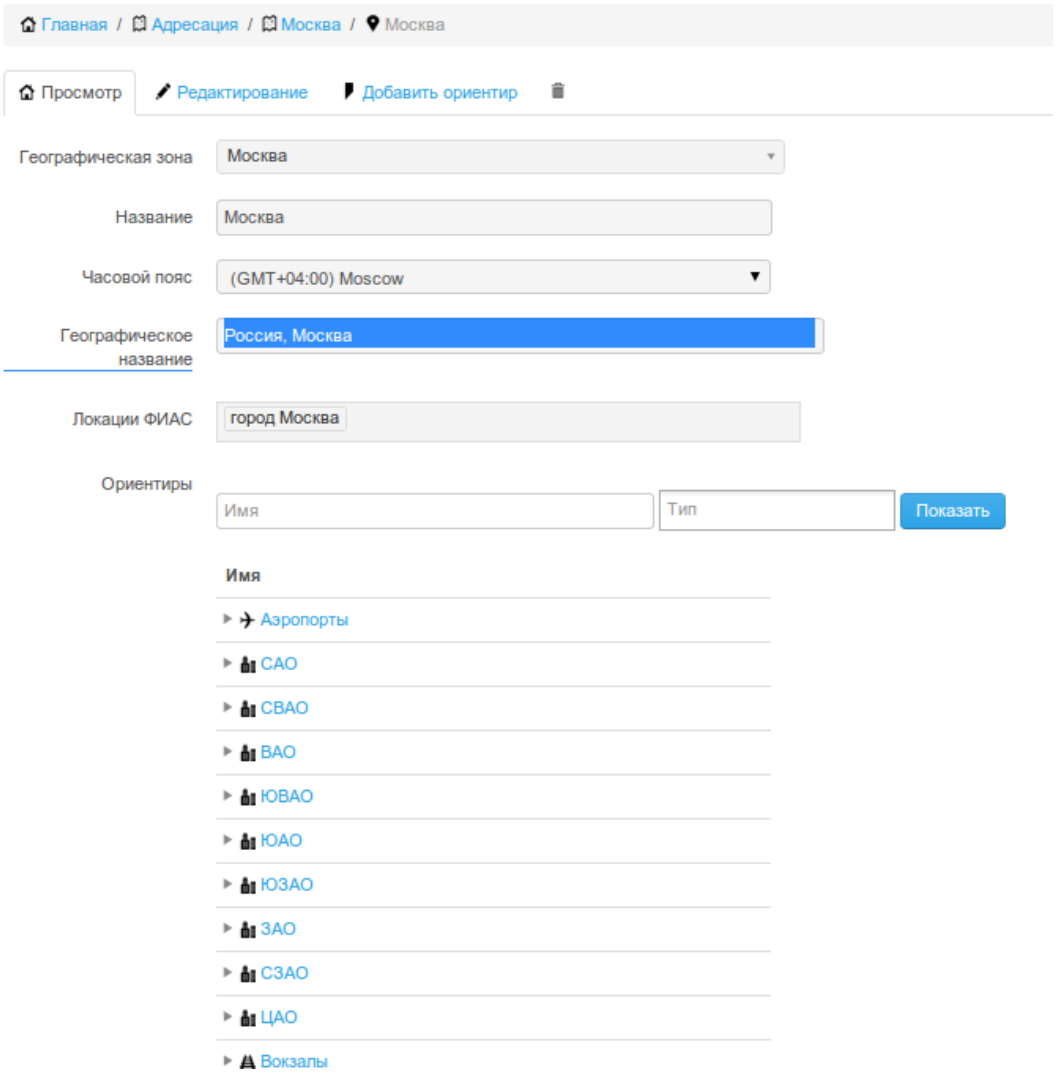

Рисунок 4.7 – Просмотр локации «Москва»

4.3.4.3. Ориентиры

Добавление и просмотр ориентира показаны на рисунках 4.8 и 4.9 соответственно

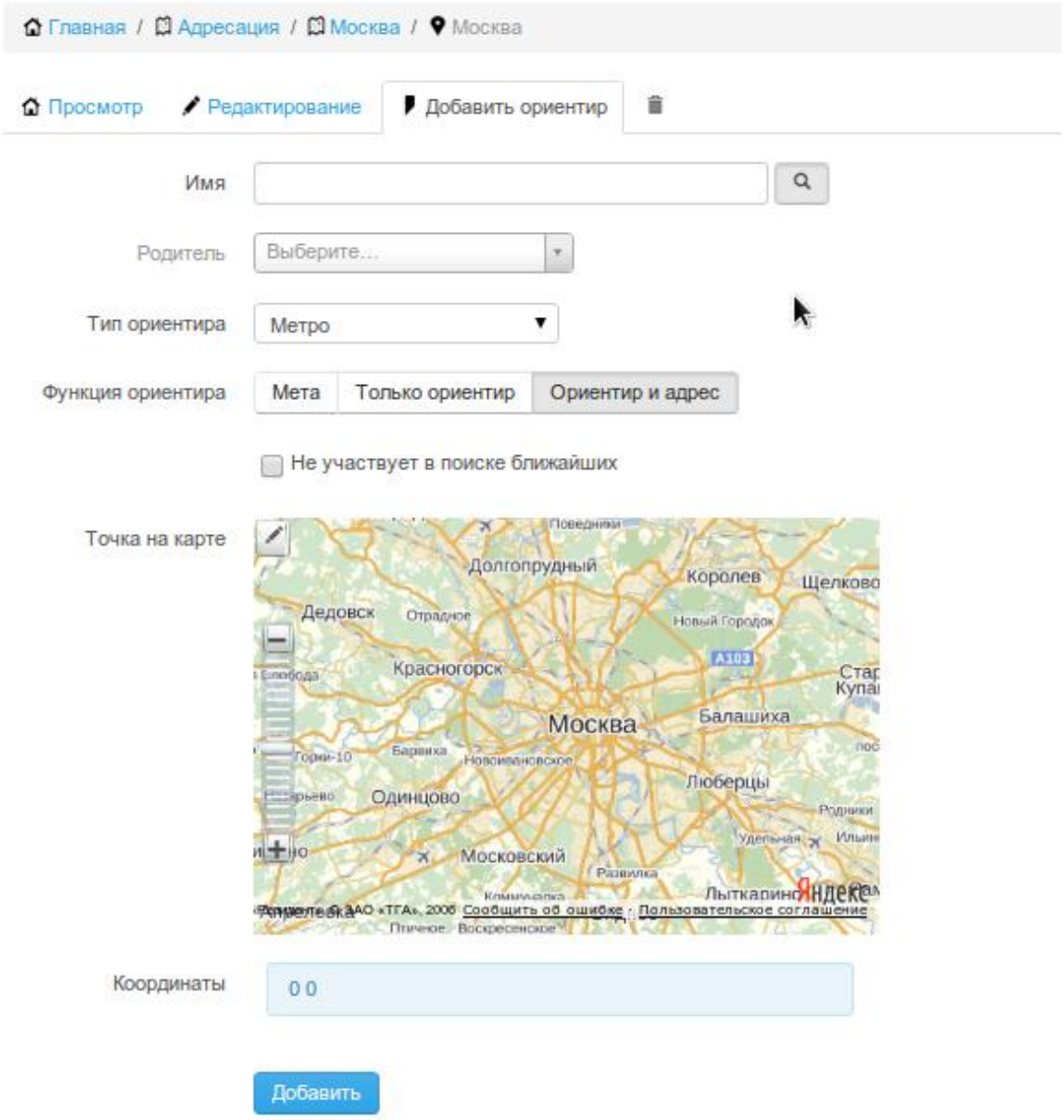

Рисунок 4.8 – Добавление ориентира

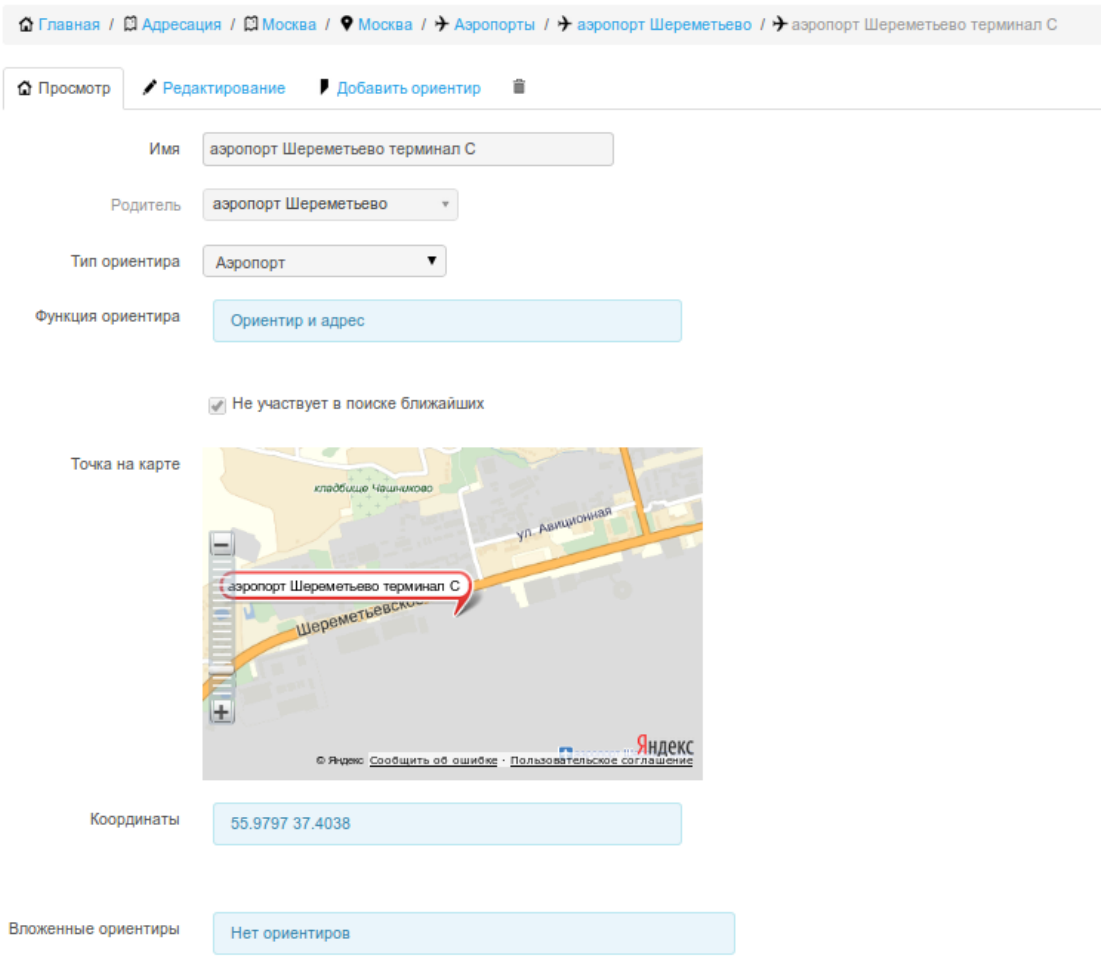

Рисунок 4.9 – Просмотр ориентира «аэропорт Шереметьево терминал С»

Добавление ориентиров шоссе

В Московской области удобно ориентироваться по километрам шоссе (например, Ярославское шоссе, 25-й километр). Для занесения таких ориентиров в базу данных был создан специальный инструмент (рис. 4.10).

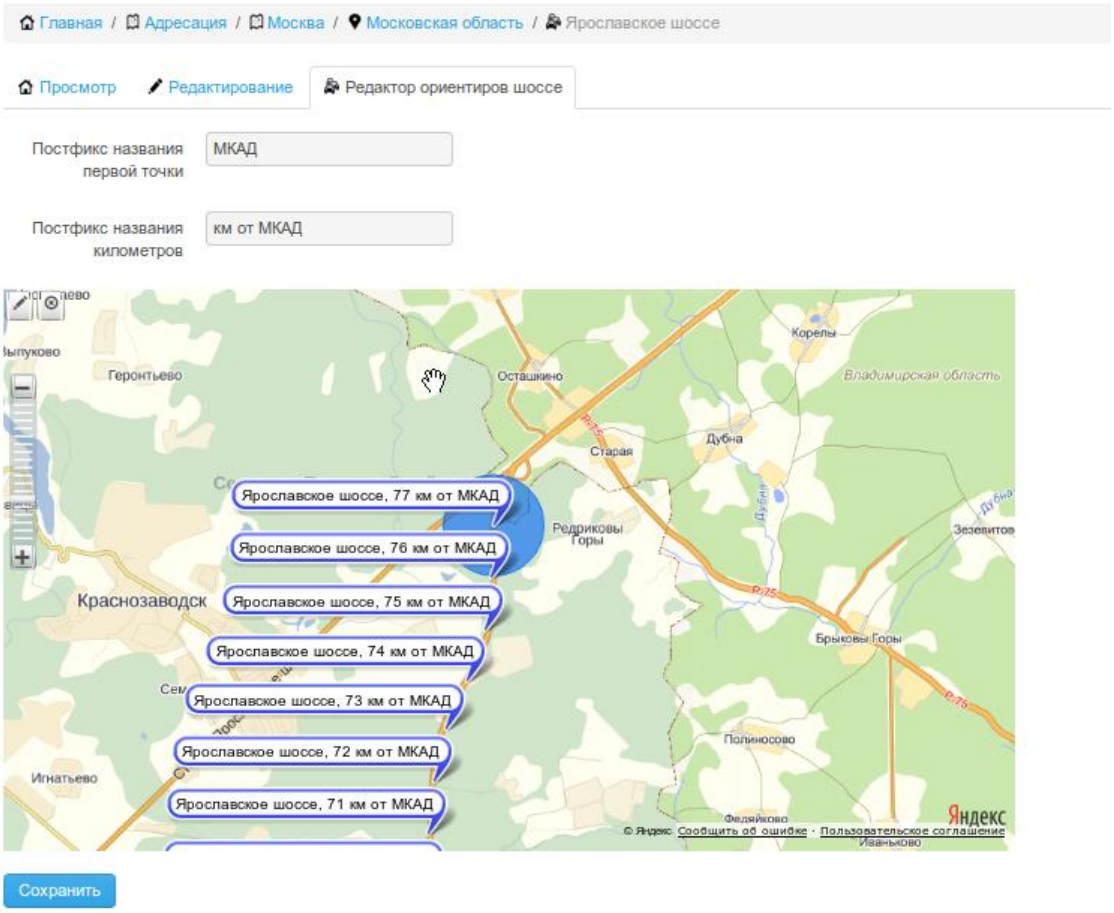

Рисунок 4.10 – Добавление ориентиров шоссе

4.3.5. Сведение всего воедино

Для распределения нагрузки, реализации кэширования и асинхронной работы большей части системы в ней был реализован специальный компонент (названный Cache). Его было принято использовать и для работы модуля. Таким образом, все действия, связанные с обращением к сети (такие, как обращения к Sphinx для поиска адресов ФИАС или ориентиров), работают через Cache. Тогда общая схема работы модуля выглядит, как на рисунке 4.11.

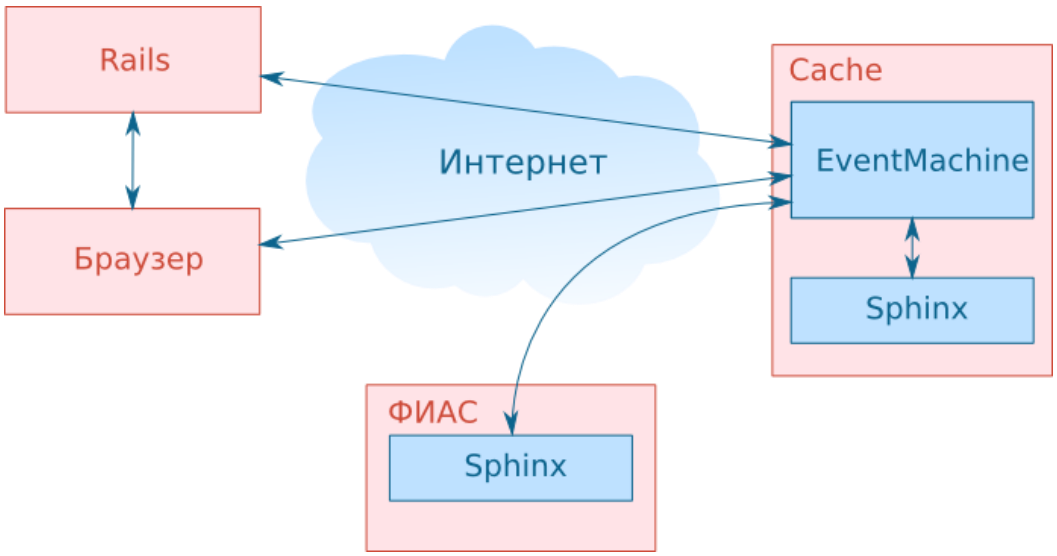

Рисунок 4.11 – Общая схема работы модуля

Например, при создании локации необходимо привязать её к глобальному идентификатору адреса ФИАС (как было рассмотрено в пункте 4.1.3.) (рис. 4.12.).

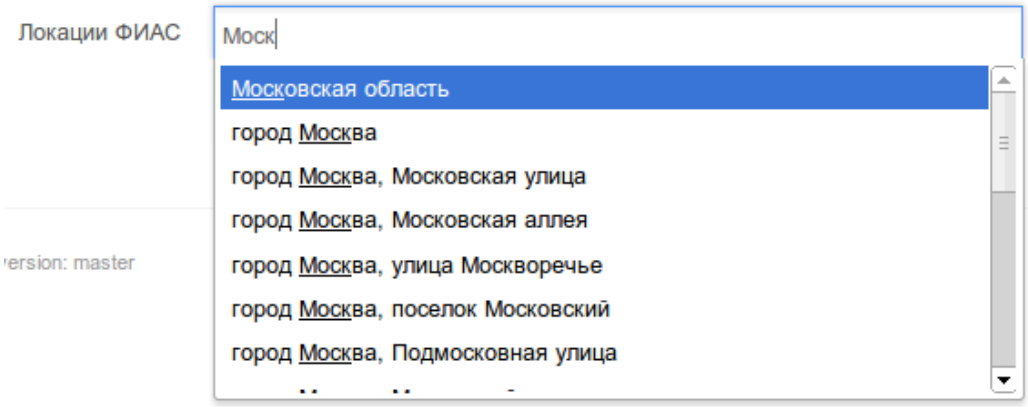

Рисунок 4.12 – Автодополнение адреса ФИАС

Процесс поиска происходит следующим образом:

1. Пользователь начинает вводить адрес;
2. Браузер посылает Ajax-запрос к Cache;
3. Cache обращается к Sphinx на машине с ФИАС, получает найденные результаты и отправляет их обратно;
4. Браузер отрисовывает полученные данные.

Похожим образом работает и автодополнение адреса в форме заказа (рис. 4.13), только для обращения браузера к Cache используется WebSocket (если поддерживается).

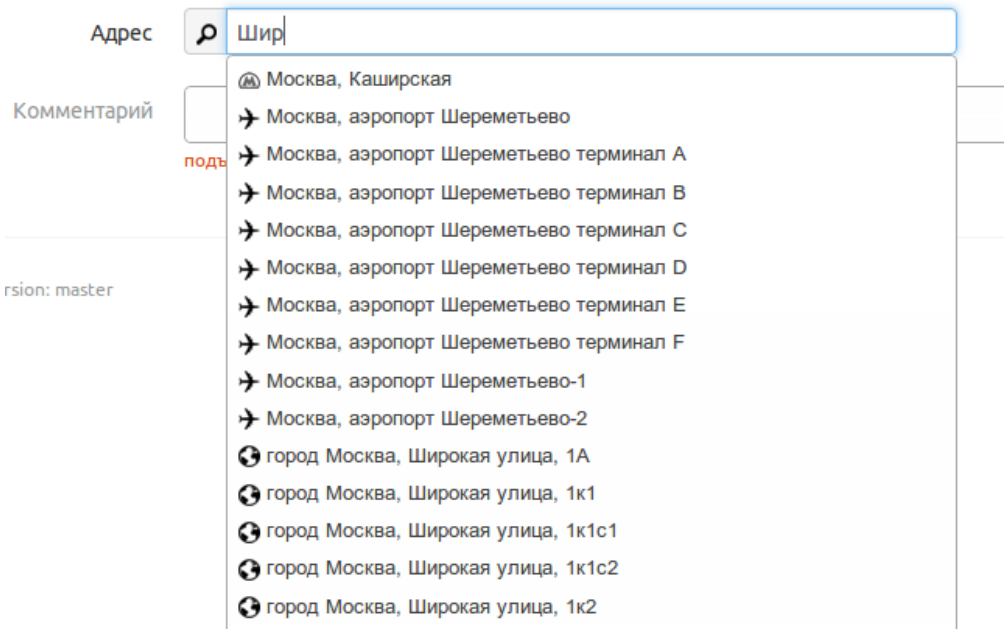

Рисунок 4.13 – Автодополнение адреса

Так же из рисунка 4.13 видно, что поиск производится сразу по адресам и ориентиром, а также, что некоторые результаты предлагаются, несмотря на возможную опечатку (по запросу «Шир» всё равно найден аэропорт «Шереметьево»)

5. Экспериментальная часть

5.1. Оценка соответствия техническому заданию

Тестирование и последующее внедрение модуля в систему показало, что он полностью соответствует техническому заданию.

5.2. Внедрение модуля в систему

Разработанный модуль был успешно внедрён в систему, где его функции используются в полной мере (например, на рисунке 5.1 показано, что тарифная зона привязана к географической, которую предоставляет модуль).

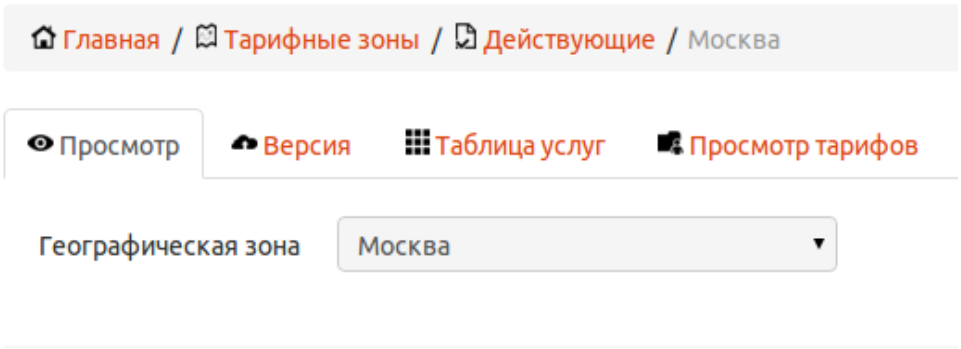

Рисунок 5.1 – Использование географической зоны в тарифной

6. Охрана труда

6.1. Исследование возможных опасных и вредных факторов при работе за компьютером и их влияния на пользователей

Персональный компьютер состоит из: системного блока, куда входит процессор, разнообразные устройства ввода/вывода информации (клавиатура, дисковые накопители, принтер, сканер), монитор. ПК часто оснащают сетевыми фильтрами, ИБП и другим вспомогательным электрооборудованием. Все эти элементы при работе компьютера образуют сложную электромагнитную обстановку на рабочем месте (таблица 7). Компьютер является источником **электромагнитных полей** (ЭМП), которые по их физическим свойствам можно разделить на электростатическое, переменные электрическое и магнитное.

Основным источником **электростатического поля** (ЭСП) является положительный потенциал, подаваемый на внутреннюю поверхность экрана. ЭСП образуется за счет разности потенциалов экрана монитора и человека. Сильное влияние на его величину оказывают потенциалы предметов окружения и влажность воздуха (при влажности выше 50% статическое поле практически отсутствует). Напряженность поля может колебаться от 8 до 75 кВ/м. Электризующиеся от трения поверхности клавиатуры и мыши тоже вносят свой вклад в общее электростатическое поле. Как показывают эксперименты, даже после работы с клавиатурой, электростатическое поле быстро возрастает с 2 до 12 кВ/м. На отдельных рабочих местах в области рук регистрировались напряженности статических электрических полей более 20 кВ/м.

Таблица 7

Источники ЭМП персонального компьютера

Источник		**Диапазон частот**
Монитор	Сетевой выпрямитель блока питания	50 Гц
	Статический преобразователь напряжения в импульсном блоке питания	20-100 Гц
	Блок кадровой развёртки и синхронизации	48-160 Гц
	Блок строчной развертки и синхронизации	15-110 кГц
	Ускоряющее анодное напряжение монитора (только для мониторов с ЭЛТ)	–
Системный блок (процессор)		50 Гц - 3000 МГц
Устройства ввода/вывода информации		- , 38 кГц
Источники бесперебойного питания		50 Гц, 20-100 кГц

6.1.2. Выводы

Во время работы за компьютером возникают следующие виды опасности:

1. Поражение электрическим током;
2. Электромагнитное излучение;
3. Статическое электричество.

6.2. Анализ влияния опасных и вредных факторов на человека

6.2.1. Анализ влияния электрического тока

По сравнению с другими видами травматизма поражение электрическим током имеет следующие особенности:

- организм человека не обладает органами, с помощью которых можно дистанционно определять наличие напряжения, и поэтому защитная реакция организма проявляется только после попадания под напряжение;
- ток, протекающий через человека, действует не только в местах контактов и по пути протекания через организм, но и вызывает рефлекторное воздействие с нарушением нормальной деятельности отдельных органов (сердечно-сосудистой, нервной системы, органов дыхания);
- возможность получения поражения током не только при прикосновении или приближении к частям электроустановки, но и без непосредственного контакта с этими частями (при поражении напряжением прикосновения или через электрическую дугу).

Электрический ток, оказывает четыре вида воздействия на организм человека: термическое, электролитическое, механическое и биологическое:

- термическое действие проявляется в ожогах отдельных участков тела, нагреве до высоких температур внутренних тканей человека, что вызывает в них серьезные функциональные расстройства;
- электролитическое действие проявляется в разложении органических жидкостей, в том числе и крови, что вызывает значительные нарушения их физико-химического состава;
- механическое действие приводит к разрыву тканей и переломам костей;
- биологическое действие проявляется в раздражении и возбуждении живых тканей в организме, а также в нарушении внутренних биоэлектрических процессов, присущих нормаль-

нодействующему организму; с биологической точки зрения исход поражения человека электрическим током может быть следствием тех физиологических реакций, которыми ткани отвечают на протекание через них электрического тока.

В физиологическом смысле действие электрического тока является **экзогенным,** то есть обусловленным факторами внешней среды. Реакции, происходящие при возникновении электрической цепи в теле человека, бывают различными, начиная от легкого раздражения и локальной судороги, кончая летальным исходом. Подобно любому другому физическому раздражителю электрический ток действует не только местно, повреждая ткани, но и рефлекторно (действия, вызванные реакцией нервной системы в ответ на раздражение электрическим током).

Электротравма – нарушение анатомических соотношений и функций тканей и органов, сопровождающееся местной и общей реакцией организма и вызванное ненормальным состоянием электрооборудования или электрических сетей.

Условно электротравмы сводятся к следующим категориям:

- местные электротравмы – ярко выраженные местные повреждения организма, нарушения целостности тканей, вызванные воздействием электрического тока;
- общие электротравмы (электрические удары) – травмы, при которых поражается весь организм в следствие нарушения деятельности жизненно важных органов и систем человека;
- смешанные электротравмы**.**

6.2.2. Анализ влияния электромагнитных полей

Установлено, что ЭМП негативно влияют на **центральную нервную систему**, вызывая головные боли, головокружения, тошноту, депрессию, бессонницу, отсутствие аппетита, возникновение синдрома стресса. Причем нервная система реагирует даже на короткие по продолжитель-

ности воздействия относительно слабых полей: изменяется гормональное состояние организма, нарушаются биотоки мозга. Особенно страдают от этого процессы обучения и запоминания. Низкочастотное ЭМП может явиться причиной **кожных заболеваний** (угревая сыпь, экзема, розовый лишай и др.), болезней **сердечно-сосудистой системы** и **кишечно-желудочного тракта**; оно воздействует на белые кровяные тельца, что приводит к возникновению опухолей, в том числе и злокачественных. Особое внимание исследователи уделяют влиянию ЭМП на женщин в период беременности. Статистика свидетельствует, что работа за компьютером нарушает нормальное течение беременности, повышает вероятность выкидыша и часто является причиной появления на свет детей с врожденными пороками, из них наиболее существенными бывают дефекты развития головного мозга. **Электростатическое поле** большой напряженности способно изменять и прерывать клеточное развитие, а также вызывать катаракту с последующим помутнением хрусталика.

6.2.3. Выводы

Из анализа воздействий опасных и вредных факторов на организм человека следует необходимость защиты от них.

6.3. Методы и средства защиты пользователей от воздействия на них опасных и вредных факторов

6.3.1. Защита от поражения электрическим током

Выполнение требований ПУЭ, ПТЭ, НПАОП 0.00-1.31-99, ГОСТ 12.1.030-87 обеспечивает электробезопасность. Оборудование, провода и кабели по степени защиты и исполнению должны соответствовать классу зоны по ПУЭ, иметь средства защиты от тока КЗ и прочих аварийных режимов. Линия электричества должна быть выполнена как отдельная групповая сеть из трёх проводов: фазового, нулевого рабочего и

нулевого защитного проводников. Нулевой защитный проводник используется для заземления. Он прокладывается от распределительного щита к розеткам питания. Не допускается подключение на щите к одному контакту двух нулевых проводников. Если в помещении используется более пяти компьютеров одновременно, то на видном и доступном месте должен быть установлен аварийный выключатель.

6.3.2. Защита от статического электричества

Для снижения влияния **электростатического поля** необходимо:

- устанавливать нейтрализаторы статического электричества;
- поддерживать в помещении относительную влажность не ниже 45-50% (чем суше воздух, тем больше электростатический заряд);
- пол в помещении застелить антистатическим линолеумом и ежедневно проводить влажную уборку;
- ограничить количество полимерных материалов в помещении;
- протирать экран и рабочее место специальной антистатической салфеткой; для снятия заряда несколько раз в день мыть руки и лицо водой, а также периодически касаться металлических предметов.

6.3.3. Защиты от электромагнитных полей

Параметры электромагнитного и электростатического полей на рабочих местах должны соответствовать требованиям ГОСТ 12.1.006-84, 12.1.005-88, ГОСТ 12.1.045-84, НПАОП 0.00-1.31-99. Для профилактики неблагоприятного влияния электромагнитного поля необходимо:

- использовать мониторы, соответствующие современным требованиям по защите от излучений (MPR II, TCO 99, TCO 03);

- соблюдать требования по площади помещения, приходящейся на одно рабочее место с ПЭВМ;
- не концентрировать на рабочем месте большого количества радиоэлектронных устройств;
- выключать мониторы, на которых временно не работают, но находятся рядом с ними.

6.4 Эргономические требования к компьютеризированным рабочим местам

Эргономика — это наука, изучающая проблемы, возникающие в системе «человек – техника - среда», с целью оптимизации трудовой деятельности оператора, создания для него комфортных и безопасных условий, повышения за счет этого его производительности, сохранения здоровья и работоспособности. Предметом эргономики является трудовая деятельность человека, а объектом исследования — система «человек – техника - среда».

Основные понятия эргономики сосредоточены в ГОСТ 26387—84 «Система «человек - машина». Термины и определения».

Цель эргономики – создание таких условий, которые делают деятельность эффективной и обеспечивают комфорт для человека.

Эффективность функционирования системы «человек – машина – среда» определяется эргономическими требованиями (ЭТ) к тем элементам системы, которые связаны с человеком при выполнении им трудовых действий. ЭТ делят на общие (характерны для группы систем) и частные. Выделяют следующие группы требований:

- ЭТ к оборудованию и параметрам его отдельных элементов;
- ЭТ к величинам рабочей (трудовой) нагрузки;
- ЭТ к организации рабочего места;
- ЭТ к факторам производственной среды.

6.4.1. Требования к помещениям и организации рабочих мест

Организация рабочего места – это система мероприятий по функциональному и пространственному размещению основных и вспомогательных средств труда для обеспечения оптимальных условий осуществления трудового процесса.

Требования к помещению

Производственное помещение — это замкнутое пространство в специально предназначенных зданиях и сооружениях, в которых осуществляется трудовая деятельность людей.

Помещение должно иметь достаточную **площадь**, быть просторным, минимальная норма – 6 м^2 на одно рабочее место с ПК. Оно должно иметь достаточный **объем** (норма – 20 м^3 на одно рабочее место), быть хорошо проветриваемым.

Для обеспечения нормированных значений микроклимата, содержания вредных веществ, ионного состава воздуха помещения для работы с компьютерами должны быть **оборудованы системами** отопления, кондиционирования воздуха или приточно-вытяжной вентиляции.

Следующие соотношения помогут определить объем воздуха, который необходимо подать в помещения:

- при объеме помещения до 20 м^3 на одного работника, на каждого необходимо подавать не меньше 30 м^3/ч;
- при объеме помещения 20-40 м^3 на одного работника – не меньше 20 м^3/ч;
- при объеме помещения больше 40 м^3 на одного работника, наличии окон и отсутствии выделений вредных веществ допускается только естественная вентиляция помещения.

Необходимые концентрации положительных и отрицательных ионов в воздухе рабочей зоны достигаются путём применения:

- генераторов отрицательных ионов;

- увлажнителей;
- кондиционеров;
- вентиляции (проветривание, система общеобменной приточно-вытяжной вентиляции, устройство местной вентиляции).

Очень важна организация **освещения.**

Из-за специфики зрительной работы с компьютерами помещения с односторонним расположением **окон** являются наиболее пригодными. Так же желательно, чтобы площадь застекления не превышала 25-50%.

Лучшим образом подходят окна, ориентированные на север или северо-восток. В этом случае устраняется нежелательное ослепляющее действие солнечных лучей. Окна необходимо оборудовать специальными средствами для регулирования освещения: жалюзями, занавесками, внешними козырьками. Коэффициент **естественной освещенности** должен составлять не менее 1,5%.

Чтобы исключить возможность попадания отраженных отблесков в глаза пользователей, поверхности в помещении должны иметь матовую или полуматовую фактуру. Коэффициент отражения пола должен быть 0.3-0.5, потолка — 0.7-0.8, стен — 0.5-0.6, других поверхностей — 0.4-0.5.

Искусственное освещение должно быть равномерным с использованием люминесцентных ламп, так что освещенность рабочих поверхностей должна быть от 300 до 500 лк. Общее освещение должно быть в виде прерывистых или сплошных линий ламп, размещаемых сбоку от рабочих мест (лучше всего — слева).

Требования к рабочему месту

Общие эргономические требования к индивидуальным рабочим местам, определяющие взаимное расположение элементов и компоновку рабочего места, изложены в ГОСТ 22 269—76, ГОСТ 12.2.032— 78, ГОСТ 12.2.033—78.

В **состав рабочего места** входят: пульт управления, средства отображения информации (СОИ), органы управления (ОУ), средства связи, рабочее сиденье (кресло) человека-оператора.

Компоновка рабочего места должна **обеспечивать**:

- оптимальную рабочую позу человека;
- обзор элементов рабочего места и пространства за его пределами;
- возможность выполнения управляющих воздействий, ведения записей, размещения документации и других материалов, осуществления всех необходимых движений, необходимых зрительных и звуковых связей;
- оптимальный режим труда и отдыха.

С помощью правильной организации рабочих мест происходит устранение дискомфорта, работник меньше утомляется, повышается его производительность. Проведенные исследования показывают, что при рациональной организации рабочих мест производительность работы возрастает на 15-25%.

При организации рабочего места необходимо:

- правильно разместить рабочее место в помещении;
- выбрать наиболее подходящее рабочее положение (с точки зрения эргономики), мебели с учётом индивидуальных особенностей работника;
- рационально расположить оснащение на рабочих местах;
- учитывать характер и особенности трудовой деятельности.

При организации рабочего места пользователя компьютера необходимо обеспечивать соответствие всех элементов рабочего места и их взаимного расположения требованиям ГОСТ 12.2.032-78 по эргономике.

Рабочие места с компьютерами лучше всего размещать в ряд, причем так, чтобы свет из окон падал сбоку, лучше, если слева. Это позволя-

ет исключить отражение на экране источников естественного света (окон) и попадание их в глаза пользователя.

При размещении рабочих мест необходимо учитывать следующие требования:

- рабочие места должны быть размещены на расстоянии не менее 1м от стен с окнами;
- расстояние между мониторами должно быть не менее 1,2м;
- расстояние между задней поверхностью одного монитора и экраном другого должно быть более 2,5м;
- проход между рядами мест должен быть не менее 1м.

Если требуется высокая концентрации внимания во время работы, рабочие места необходимо разделять перегородками высотой 1,5-2м.

6.4.2. Требования к организации работы

Режим труда и отдыха — это распорядок, регламентирующий определенное чередование времени работы и отдыха на протяжении различных промежутков времени.

Главными **целями**, которые преследует разработка рациональных режимов труда и отдыха являются эффективное использование производственных мощностей, обеспечение высокой продуктивности и работоспособности, сохранение здоровья и создание благоприятных условий для развития личности работника.

При разработке режимов труда и отдыха необходимо :

- учитывать психофизиологические особенности отдельных групп работников (в первую очередь, подростков и беременных или кормящих женщин);
- рационально чередовать работу и отдых;
- регламентировать периодичность, продолжительность отдыха, а так же его организацию в течение смены;

- унифицировать методы и принципы определения количества и продолжительности перерывов на отдых;
- устанавливать время на отдых и организацию его проведения с учетом нагрузок, испытываемых во время работы, и обусловленных условиями и содержанием труда;

Сохранение высокой производительности труда пользователей компьютера достигается путём использования методов установления рационального режима труда и отдыха:

- обеспечение постепенного вхождения в труд на старте работы – с начала выполнение простых операций и постепенный переход к более сложным;
- установление грамотного режима отдыха работников;
- понимание того, что в зависимости от характера работы, время на отдых может варьироваться, и установление наиболее подходящих вариантов.

Трудовая деятельность при работе за компьютером можно разделить на три группы:

- разработчики программ (инженер-программист);
- оператор ЭВМ;
- оператор компьютерного набора.

Длительность и частота перерывов за рабочую смену в зависимости от вида деятельности приведена в таблице 8.

Таблица 8

Длительность и частота перерывов пользователей компьютера в зависимости от категории работ

Категория работы	**Общее время регламентированных перерывов**	
	8-часовая смена	**12-часовая смена**
Инженер-программист	15 мин после каждого час работы	В первые 8 часов работы перерывы аналогичны 8-часовой смене, а в те-
Операторы ЭВМ	15 мин после каждых	

	2 часов работы	чение следующих 4 часов – 15 мин после каждого часа работы
Операторы компьютерного набора	10 мин после каждого часа работы	

Если по производственным обстоятельствам нет возможности предоставления регламентированных перерывов, то продолжительность непрерывной работы за компьютером не должна превышать 4 ч.

6.5. Выводы

Выбранные методы и способы защиты пользователей от воздействии на них опасных и вредных факторов, при соблюдении эргономических требований, позволяют обеспечить безопасную работу и сохранить здоровье.

Заключение

В результате проделанной работы были сформированы требования к модулю, проанализированы реализации данных требований в существующих системах. На основе этого анализа были выявлены функции, необходимые для успешной конкуренции в рассматриваемой области. В дальнейшем была спроектирована оптимальная архитектура, и на её основе был реализован требуемый модуль.

После внутреннего тестирования модуль был внедрён в систему, где был опробован в реальной работе. В итоге было выявлено повышение эффективности работы сотрудников в результате снижения временных затрат на составление заказа в среднем на 45%.

Список литературы

1. Томас Д., Хэнссон Д.Х. Гибкая разработка веб-приложений в среде Rails. — СПб; Питер, 2008 — 716 с.; ил.
2. «Такси Мастер», официальный сайт компании «Такси Мастер», 2013, http://www.taximaster.ru/
3. «Infinity Taxi», официальный сайт компании «Infinity Taxi», 2013 http://www.taxi-infinity.ru
4. «Интерактивное Такси», официальный сайт компании «Мадив», 2013, http://www.madiv.ru/solutions/
5. Model-View-Controller, 2013, http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller
6. Unicorn, официальная страница unicorn, 2013, http://unicorn.bogomips.org
7. EventMachine Wiki, официальная документация EventMachine, 2013, https://github.com/eventmachine/eventmachine/wiki
8. SphinxSearch, официальный сайт Sphinx Technologies Inc., 2013, http:/sphinxsearch.com
9. Федеральная информационная адресная система, официальный сайт федеральной налоговой службы, 2013, http://fias.nalog.ru/Public/NewsPage.aspx
10. Фонетические алгоритмы, 2011, http://habrahabr.ru/post/114947/
11. libxml2, 2013, http://www.xmlsoft.org/python.html
12. ГОСТ 12.0.003-99. Опасные и вредные производственные факторы.
13. ГОСТ 12.1.030-01. Электробезопасность. Защитное заземление. Зануление.

14. ГОСТ 12.1.045-01. Электростатические поля. Допустимые уровни на рабочих местах.
15. ГОСТ 12.4-99. Средства защиты от статического электричества.
16. ГОСТ Р 50948, 49-96. Общие эргономические требования и требования безопасности и ее параметры для ЭВМ.
17. ГОСТ 26387—84. Система «человек - машина». Термины и определения.
18. ГОСТ 12.2.032-78: Система стандартов безопасности труда. Рабочее место при выполнении работ сидя. Общие эргономические требования.
19. СанПиН №1340-03. Гигиенические требования к персональным ЭВМ и организация работы. Санитарно-гигиенические правила и нормы.