**Реферат**

В данной работе рассматривается метод построения велосипедных маршрутов. В первой главе производится анализ предметной области, рассматриваются различные алгоритмы построения оптимальных и субоптимальных маршрутов, проводится их анализ. Во второй главе разрабатывается метод построения велосипедных маршрутов различных видов (безопасных, гоночных). Также описываются вспомогательные алгоритмы. В третьей главе описываются средства разработки, использованные в ходе реализации разработанного метода. В четвертой главе описано исследование разработанного метода.

**Содержание**

[**Обозначения и сокращения** 7](#_Toc420416459)

[**Введение** 8](#_Toc420416460)

[**1 Аналитический раздел** 9](#_Toc420416461)

[**1.1 Цель и задачи** 9](#_Toc420416462)

[**1.2 Анализ предметной области** 9](#_Toc420416463)

[**1.3 Математическая модель** 13](#_Toc420416464)

[**1.4 Свойства дорожного графа** 14](#_Toc420416465)

[**1.5 Анализ алгоритмов поиска оптимальных путей в графах** 14](#_Toc420416466)

[**1.5.1 Таблица всех расстояний** 15](#_Toc420416467)

[**1.5.2 Поиск в ширину** 15](#_Toc420416468)

[**1.5.3 Алгоритм Беллмана-Форда** 16](#_Toc420416469)

[**1.5.4 Алгоритм Флойда-Уоршелла** 17](#_Toc420416470)

[**1.5.5 Алгоритм Дейкстры** 17](#_Toc420416471)

[**1.5.6 Алгоритм A\*** 19](#_Toc420416472)

[**1.5.7 Выбор алгоритмов для построения кратчайшего пути** 20](#_Toc420416473)

[**1.6 Анализ алгоритмов поиска субоптимальных путей в графах** 21](#_Toc420416474)

[**1.6.1 Алгоритм нахождения К кратчайших путей (алгоритм Йена)** 21](#_Toc420416475)

[**1.6.2 Алгоритм нахождения E-близких маршрутов** 22](#_Toc420416476)

[**1.6.3 Выбор алгоритмов для построения субоптимальных маршрутов** 23](#_Toc420416477)

[**1.7 Формализация разрабатываемого метода в общем виде** 23](#_Toc420416478)

[**1.7.1 Входные и выходные данные** 24](#_Toc420416479)

[**1.7.2 Ограничения разрабатываемого метода** 24](#_Toc420416480)

[**1.7.3 Функциональные требования к разрабатываемому ПО** 25](#_Toc420416481)

[**2 Конструкторский раздел** 26](#_Toc420416482)

[**2.1 Функциональная модель метода** 26](#_Toc420416483)

[**2.2 Построение графа дорог** 27](#_Toc420416484)

[**2.2.1 Структура OpenStreetMap** 27](#_Toc420416485)

[**2.2.2 Структура данных для представления графа дорог** 31](#_Toc420416486)

[**2.3 Построение кратчайшего маршрута** 34](#_Toc420416487)

[**2.4 Построение субоптимальных маршрутов** 37](#_Toc420416488)

[**2.5 Выбор маршрута в зависимости от вида** 39](#_Toc420416489)

[**2.6 Структура разрабатываемого ПО** 40](#_Toc420416490)

[**2.7 Способы тестирования ПО** 42](#_Toc420416491)

[**2.8 Выводы** 42](#_Toc420416492)

[**3 Технологический раздел** 43](#_Toc420416493)

[**4 Экспериментальный раздел** 44](#_Toc420416494)

[**5 Организационно-экономическая часть** 45](#_Toc420416495)

[**5.1 Определение этапов и содержания работ** 45](#_Toc420416498)

[**5.2 Расчет трудоемкости** 46](#_Toc420416499)

[**5.3 Определение численности исполнителей** 48](#_Toc420416500)

[**5.4 Сетевая модель проекта** 49](#_Toc420416501)

[**5.5 Календарный график выполнения проекта** 50](#_Toc420416502)

[**5.6 Анализ структуры затрат проекта** 52](#_Toc420416503)

[**5.6.1 Расчет расходов** 52](#_Toc420416504)

[**5.6.2 Расчет расходов, связанных с обеспечением работ оборудованием** 53](#_Toc420416505)

[**5.6.3 Расчет расходов, связанных с организацией рабочих мест** 53](#_Toc420416506)

[**5.6.4 Расчет накладных расходов** 54](#_Toc420416507)

[**5.7 Исследование рынка для разрабатываемого ПО** 55](#_Toc420416508)

[**5.8 Планирование цены и прогнозирование прибыли** 55](#_Toc420416509)

[**5.9 Выводы** 57](#_Toc420416510)

[**6 Охрана труда и экология** 58](#_Toc420416511)

[**6.1 Анализ опасных и вредных факторов при разработке программного обеспечения и мероприятия по их устранению** 58](#_Toc420416512)

[**6.1.1 Микроклимат** 58](#_Toc420416513)

[**6.1.2 Шум и вибрации** 59](#_Toc420416514)

[**6.1.3 Освещение** 59](#_Toc420416515)

[**6.1.4 Рентгеновское излучение** 60](#_Toc420416516)

[**6.1.5 Неионизирующие электромагнитные излучения** 61](#_Toc420416517)

[**6.1.6 Визуальные параметры** 61](#_Toc420416518)

[**6.2 Акустический расчет** 62](#_Toc420416519)

[**6.3 Расчет системы искусственного освещения** 66](#_Toc420416520)

[**Заключение** 75](#_Toc420416521)

[**Список литературы** 76](#_Toc420416522)

[**Приложение А** 77](#_Toc420416523)

# **Обозначения и сокращения**

OSM – OpenStreetMap

ПДД – Правила дорожного движения

ДТП – Дорожно-транспортное происшествие

ПО – Программное обеспечение

БД – База данных

СУБД – Система управления базами данных

# **Введение**

В последнее время популярность использования велосипедов в качестве средства передвижения или для активного отдыха растет. Поэтому многие велосипедисты нуждаются в сервисах, которые помогут им построить оптимальный, безопасный или маршрут для тренировки на шоссейном велосипеде. Существующие сервисы маршрутизации работают, в основном, только для автомобилей. Отсутствие хороших сервисов построения велосипедных маршрутов можно объяснить несколькими проблемами, связанными с данной задачей:

1. Отсутствие данных о дорожках, по которым можно проехать на велосипеде;
2. Для автоматического построения велосипедного маршрута необходимо учитывать множество факторов, относящих к отдельному велосипедисту.

В данной работе разрабатывается метод, который позволяет решить первую и, частично, вторую проблему. Этот метод основан на построении не одного, а нескольких возможных маршрутов, из которых выбирается наиболее подходящий по определенным метрикам.

1. **Аналитический раздел**

В данном разделе ставится цель данного дипломного проекта, определяются задачи для достижения этой цели. Далее проводится анализ предметной области и обосновывается необходимость данной разработки. Так как для решения некоторых задач существует несколько алгоритмов, то в этом разделе будет приведен их анализ, а также обоснование выбора алгоритмов для реализации в данной работе. В конце аналитического раздела будет приведена общая формализация задачи, входные и выходные данные разрабатываемого метода, ограничения, накладываемые на этот метод, а также требуемая функциональность, которая должна присутствовать в приложении, реализующем работу метода построения велосипедных маршрутов.

* 1. **Цель и задачи**

Целью данной работы является разработка метода построения веломаршрутов и его реализация в виде Web-приложения. Для достижения этой цели необходимо выполнить ряд задач:

* Анализ, исследование и выбор алгоритмов для построения оптимальных и субоптимальных маршрутов;
* Разработка метода построения веломаршрутов различных видов;
* Построение графа из исходных данных о карте;
* Реализация разработанного метода.
  1. **Анализ предметной области**

Самыми популярными сервисами построения маршрутов на сегодняшний момент являются Яндекс.Карты [1] и Google Maps [2]. Каждый из них хорошо справляется с задачей прокладывания оптимального пути для передвижения на машине, но ни один из них не умеет строить велосипедные маршруты. В Google Maps есть возможность проложить путь для похода пешком. На рисунке 1.1 представлен один из таких маршрутов.

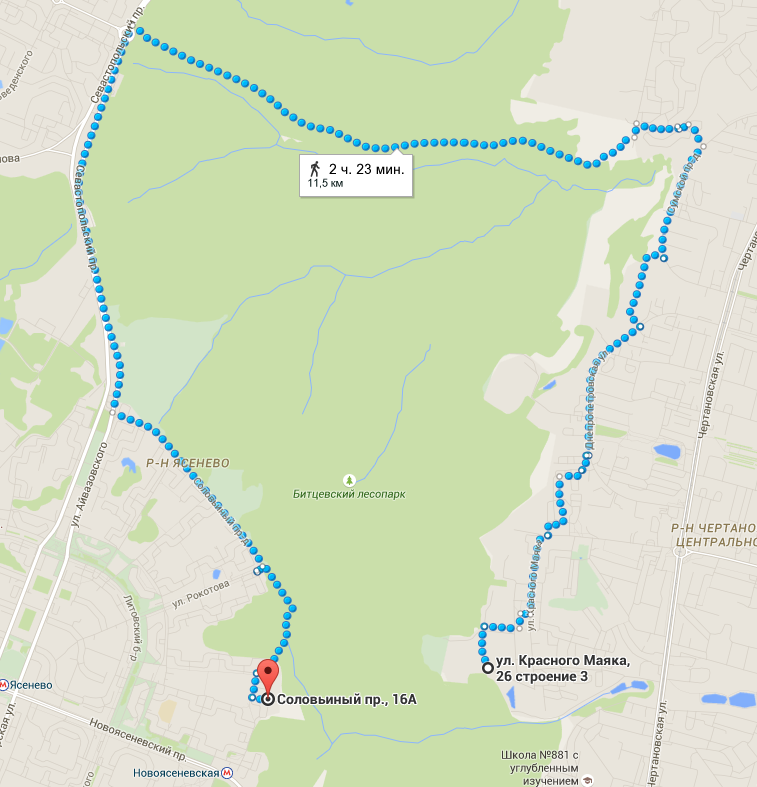


Рисунок 1.1 – Пешеходный маршрут Google Maps

Этот маршрут не является оптимальным, так как есть пешеходная дорога через лес, по которой можно пройти, затратив гораздо меньше времени. Это является следствием первой проблемы, о которой было сказано выше. Google не располагает достаточным количеством данных для того, чтобы пользователь мог получить необходимый ему путь.

Google и Яндекс используют свои данные о карте, поэтому они ощущают некоторые проблемы с неполнотой этих данных. Существуют другие сервисы, использующие данные проекта OpenStreetMap[3]. OSM ‒ некоммерческий веб-картографический проект по созданию силами сообщества участников-пользователей Интернета подробной свободной и бесплатной географической карты мира. Для создания карт используются данные с персональных GPS-трекеров, аэрофотографии, видеозаписи, спутниковые снимки и панорамы улиц, предоставленные некоторыми компаниями, а также знания человека, рисующего карту. OpenStreetMap при создании карты используется принцип вики. Каждый зарегистрированный пользователь может вносить изменения в карту. Более подробно структура OSM будет описана в технологическом разделе.

Одним из сервисов, использующих OSM является GraphHooper[4]. GraphHooper – это приложение для построения маршрутов для автомобилей, пешеходов и велосипедистов. Для того, чтобы показать, что использование OSM позволяет устранить проблему неполноты данных о дорогах, можно с помощью GraphHooper построить пешеходный маршрут, показанный на рисунке 1.1. Результат работы приложения проиллюстрирован на рисунке 1.2.

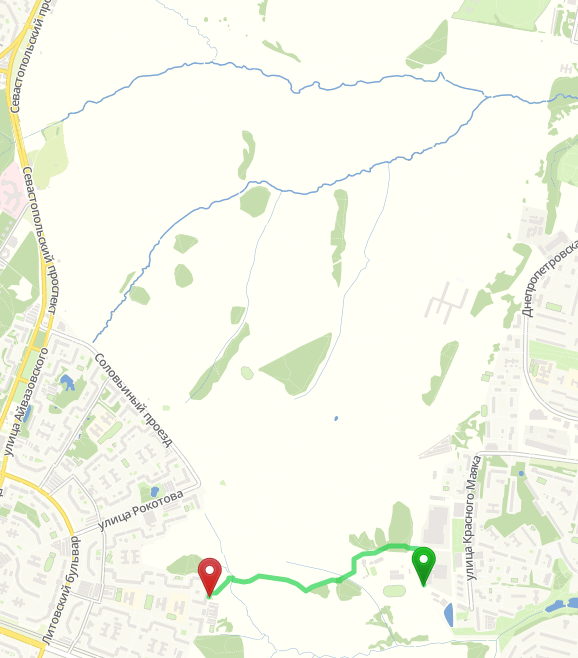


Рисунок 1.2 – Пешеходный маршрут GraphHooper

Сравнив результаты построения этого пути Google Maps и GraphHooper, можно прийти к выводу, что использование OSM для построения велосипедных маршрутов целесообразно, так как практически везде, где можно пройти пешком, можно проехать и на велосипеде.

Это приложение написано на языке Java. Преимуществом этого проекта является то, что прокладывание пути занимает доли секунды независимо от дины маршрута. К недостаткам можно отнести то, что строится только оптимальный маршрут, не учитывая предпочтения пользователя. Это как раз и является второй проблемой автоматического планирования велосипедного путешествия. Приложение, разрабатываемое в данной работе учитывает некоторые предпочтения пользователя.

Рассмотрим правила дорожного движения для велосипедистов. Для начала нужно определиться что является велосипедом. Велосипед - транспортное средство, кроме инвалидных колясок, которое имеет по крайней мере два колеса и приводится в движение как правило мускульной энергией лиц, находящихся на этом транспортном средстве, в частности при помощи педалей или рукояток, и может также иметь электродвигатель номинальной максимальной мощностью в режиме длительной нагрузки, не превышающей 0,25 кВт, автоматически отключающийся на скорости более 25 км/ч [5]. Велосипед является транспортным средством, поэтому велосипедист – полноправный участник дорожного движения. Следовательно, все пункты ПДД, относящиеся к транспортным средствам, применимы и к велосипедам. В данной работе необходимо учитывать пункты ПДД, относящиеся к дорогам, по которым могут передвигаться велосипедисты. Начиная с апреля 2014 года движению велосипедистов посвящены сразу 6 пунктов раздела 24 правил дорожного движения (24.1 - 24.6). Движение велосипедистов старше 14 лет возможно в порядке убывания:

* По велосипедной, велопешеходной дорожкам или полосе для велосипедистов.
* По правому краю проезжей части.
* По обочине.
* По тротуару или пешеходной дорожке.

Каждый последующий пункт в вышеприведенном списке подразумевает, что предыдущие пункты отсутствуют или по этим дорогам невозможно проехать.

С 2014 года на дорогах можно встретить выделенные полосы для велосипедистов, обозначенные специальными знаками, которые проиллюстрированы на рисунке 1.3.



Рисунок 1.3 – Знаки, обозначающие дороги, по которым можно передвигаться только на велосипеде или мопеде

По этим полосам разрешено движение только на велосипедах и мопедах.

Кроме того, с 2015 года велосипедисты могут двигаться и по выделенным полосам для общественного транспорта, которые обозначаются знаками, проиллюстрированными на рисунке 1.4.



Рисунок 1.4 – Знаки, обозначающие дороги для общественного транспорта

Водителям велосипеда запрещается двигаться по автомагистралям, поэтому это нужно учесть при построении графа дорог, то есть сразу исключить из него все автомагистрали.

Велосипедистов, которые будут использовать сервис маршрутизации, можно условно разделить на 3 группы. К первой группе относятся те, кто использует велосипед в качестве транспортного средства. Таким пользователям нужно добраться из одной точки в другую (например, от дома до работы) максимально быстро, поэтому для них строится оптимальный маршрут. Ко второй группе относятся те, кто использует велосипед для активного отдыха. Такие велосипедисты обычно избегают больших дорог, где достаточно оживленный поток машин, которые двигаются с большой скоростью. Исходя из этого можно сказать, что простой оптимальный маршрут не совсем подойдет этим пользователям. Очевидно, что самый безопасный маршрут построить практически невозможно, так как для этого нужно исследовать весь граф дорог, что является очень затратным по времени процессом. Поэтому в разрабатываемом методе безопасный маршрут ищется среди путей, которые являются близкими к оптимальному. Математическая формализация метрики выбора из множества путей наиболее безопасного будет рассмотрена в конструкторском разделе. К последней группе потенциальных пользователей сервиса относятся спортсмены, профессионально занимающиеся велоспортом, которые желают провести тренировку по городу. Таким велосипедистам важно, чтобы маршрут проходил по асфальтированной дороге, так как все они в основном используют шоссейные велосипеды, на которых тяжело (хотя возможно) ездить по бездорожью где-нибудь в лесу. Будем называть такие маршруты гоночными или спортивными. Метрика для выбора гоночных путей будет также формализована в конструкторском разделе.

Перед рассмотрением основных алгоритмов для решения задач поиска оптимальных и субоптимальных маршрутов опишем математическую модель задачи.

* 1. **Математическая модель**

Множество всех возможных дорог, по которым можно проехать, представляется в виде графа: , где – множество вершин графа, – множество ребер графа.

является связным ориентированным графом. Вершинам такого графа соответствуют пересечения дорог, а ребрам – сами дороги. Таким образом, вершина – это точка на карте, которая имеет географические координаты. Ребро является самой дорогой, соединяющей 2 вершины. Каждое ребро имеет несколько свойств: длина, тип дороги, количество полос, вид дорожного покрытия. Весом ребра является его длина, поэтому граф не имеет ребер с отрицательным весом.

В дальнейшем под будем понимать начальную точку движения, под конечную, под – промежуточную точку, где , – количество промежуточных точек. Также для упрощения объяснения всех алгоритмов обозначим текущую вершину, в которой алгоритм находится в данный момент как , а метку этой вершины как . – степень вершины, то есть количество инцидентных ей ребер.

* 1. **Свойства дорожного графа**

Дорожные графы, которые используются в данной работе имеют некоторые свойства, которые отличают их от обычных графов. Рассмотрим основные свойства:

* + Граф дорог редко изменяется;
  + В дорожных графах степень вершин очень редко превышает 5, поэтому он является достаточно разреженным;
  + Дорожные графы почти планарны, то есть могут быть представлены на плоскости практически без пересечения ребер (кроме небольшого количества туннелей и мостов);
  + Каждый узел вершины имеет свои координаты, это значит, что можно посчитать расстояние между двумя вершинами, если это необходимо;
  + В дорожной сети существуют более важные и менее важные дороги, то есть ребра графа можно классифицировать по типу, представляющей его дороги.

Эти свойства можно использовать для реализации алгоритмов и для создания новых. Например, в некоторых системах последнее свойство используется для того, чтобы уменьшить время поиска маршрута между двумя городами, за счет сохранения в памяти части пути, который не меняется при проезде между этими городами.

* 1. **Анализ алгоритмов поиска оптимальных путей в графах**

Существуют различные постановки задачи о кратчайшем пути:

* + Задача о кратчайшем пути в заданный пункт назначения. Требуется найти кратчайший путь в заданную вершину назначения , который начинается в каждой из вершин графа (кроме ). Поменяв направление каждого принадлежащего графу ребра, эту задачу можно свести к задаче о единой исходной вершине (в которой осуществляется поиск кратчайшего пути из заданной вершины во все остальные).
  + Задача о кратчайшем пути между заданной парой вершин. Требуется найти кратчайший путь из заданной вершины в заданную вершину .
  + Задача о кратчайшем пути между всеми парами вершин. Требуется найти кратчайший путь из каждой вершины в каждую вершину . Эту задачу тоже можно решить с помощью алгоритма, предназначенного для решения задачи об одной исходной вершине, однако обычно она решается быстрее.

В различных постановках задачи, роль длины ребра могут играть не только сами длины, но и время, стоимость, расходы, объем затрачиваемых ресурсов (материальных, финансовых, топливно-энергетических и т. п.) или другие характеристики, связанные с прохождением каждого ребра. Таким образом, задача находит практическое применение в большом количестве областей (информатика, экономика, география и др.).

В данной работе используется постановка задачи о кратчайшем пути между заданной парой вершин, а в качестве веса ребер используется длина этого ребра, которой является длина дороги, которая представляет это ребро.

Теперь рассмотрим основные алгоритмы для решения данной задачи.

* + 1. **Таблица всех расстояний**

Этот подход основан на предварительном вычислении всех возможных маршрутов и записи их в таблицу. Такой метод позволяет найти необходимый путь за постоянное время, которое требуется для извлечения данных о пути из таблицы. Но такой подход может быть применим только для маленьких графов. Дорожные графы могут содержать сотни тысяч точек, таблица расстояний для него будет требовать гигабайты или даже терабайты памяти, а время просчета всех путей может занять дни.

* + 1. **Поиск в ширину**

Впервые поиск в ширину был описан Эдвардом Муром в 1959 году в книге [6]. Поиск в ширину ‒ один из простейших алгоритмов обхода графа, который является основой для многих важных алгоритмов для работы с графами. Этот алгоритм позволяет найти кратчайший путь в невзвешенном ориентированном графе. Поиск в ширину работает путём последовательного просмотра отдельных уровней графа, начиная с узла-источника .

Алгоритм работы описывается следующим образом:

1. Поместить узел , с которого начинается поиск, в очередь.
2. Извлечь из начала очереди узел и пометить его как развёрнутый.

* Если узел является целевым узлом, то завершить поиск с результатом «успех».
* В противном случае, в конец очереди добавляются все преемники узла .

1. Если очередь пуста, то все узлы связного графа были просмотрены, следовательно, целевой узел недостижим из начального; завершить поиск с результатом «неудача».
2. Вернуться к п. 2.

Оценим время работы поиска в ширину для входного графа , где множество ребер представлено списком смежности. Операции внесения в очередь и удаления из нее требуют O(1) времени, так что общее время работы с очередью составляет операций, так как каждая вершина добавляется в список по одному разу. Для каждой вершины рассматривается не более ребер, инцидентных ей. Так как , то время, используемое на работу с ребрами, составляет Поэтому общее время работы алгоритма поиска в ширину ‒

* + 1. **Алгоритм Беллмана-Форда**

Этот алгоритм впервые был независимо предложен в 1958 году Ричардом Беллманом [7] и в 1962 году Лестером Фордом [8]. Алгоритм находит кратчайшие пути от одной вершины графа до всех остальных. Алгоритм Беллмана–Форда допускает рёбра с отрицательным весом. Отрицательным циклом называется такой цикл, сумма весов ребер которого отрицательна. Если граф содержит отрицательный цикл, то кратчайшего пути не существует, так как существует сколь угодно короткий путь от одной вершины этого цикла до другой (каждый обход цикла уменьшает длину пути). Для начала рассмотрим работу алгоритма с графом, не имеющим отрицательных весов ребер.

Заведём массив расстояний , который после отработки алгоритма будет содержать ответ на задачу. В начале работы мы заполняем его следующим образом: , а все остальные элементы массива равны бесконечности, так как расстояние до этих вершин пока неизвестно. Алгоритм состоит из нескольких фаз. На каждой фазе алгоритм просматривает все ребра графа и пытается уменьшить значение значением где – вес ребра, соединяющего вершины и , а – полученное на данный момент расстояние от вершины до вершины Для нахождения всех длин кратчайших путей достаточно выполнить фаз алгоритма. Ниже приведен псевдокод работы алгоритма:

Инициализация занимает времени, каждый из проходов требует времени. Значит алгоритм Беллмана-Форда для нахождения кратчайших путей во взвешенном графе с неотрицательными весами ребер работает за времени.

Алгоритм Беллмана–Форда позволяет очень просто определить, существует ли в графе отрицательный цикл, достижимый из вершины . Достаточно произвести внешнюю итерацию цикла не , a ровно раз. Если при исполнении последней итерации длина кратчайшего пути до какой-либо вершины строго уменьшилась, то в графе есть отрицательный цикл, достижимый из s.

* + 1. **Алгоритм Флойда-Уоршелла**

Алгоритм независимо разработан в 1962 году Робертом Флойдом [9] и Стивеном Уоршеллом [10]. Он находит длины кратчайших путей между всеми парами вершин во взвешенном ориентированном графе. Если в графе нет отрицательных циклов, то алгоритм работает корректно. Если такой цикл существует, алгоритм позволяет найти хотя бы один такой цикл.

Пусть вершины введено обозначение для длины кратчайшего пути от до, который кроме самих вершин проходит только через вершины . Очевидно, что — длина (вес) ребра если таковое существует (в противном случае его вес обозначается как ). Существует два варианта значения , ).

1. Кратчайший путь между не проходит через вершину , тогда
2. Существует более короткий путь между , проходящий через , тогда он сначала идёт от до , а потом от до . В этом случае, .

Таким образом для нахождения каждого достаточно выбрать минимум из обозначенных выше значений.

Первым этапом алгоритма является заполнение матрицы кратчайших путей значениями весов всех ребер графа. На втором этапе алгоритм пытается уменьшить длину всех путей в графе.

Ниже приведен псевдокод работы алгоритма:

Так как в ходе работы алгоритма каждый из трех циклов выполняется раз, то сложность алгоритма может быть оценена как

* + 1. **Алгоритм Дейкстры**

Классическим решением задачи о кратчайшем пути является алгоритм Дейкстры, описанный еще в 1959 году в работе [11]. Алгоритм находит кратчайшее расстояние от одной вершины до всех остальных. Работает только для графа с вершинами неотрицательного веса. Для нахождения расстояния до конкретной вершины необходимо остановить алгоритм при достижении этой вершины.

Каждой вершине из сопоставляется временная метка, которая обозначает минимальное известное расстояние от до этой вершины. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены (или алгоритм дошел до конечной вершины).

Инициализация. Метка самой вершины полагается равной 0, метки остальных вершин — бесконечности. Это означает, что расстояния от до остальных вершин пока неизвестны. Все вершины графа помечаются как непосещённые.

Шаг алгоритма. Если дошли до конечной вершины, алгоритм завершается. В противном случае, из ещё не посещённых вершин выбирается вершина , имеющая минимальную метку. Мы рассматриваем всевозможные маршруты, в которых является предпоследним пунктом. Вершины, в которые ведут рёбра из назовём соседями этой вершины. Для каждого соседа вершины , кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки и длины ребра, соединяющего с этим соседом. Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину как посещённую и повторим шаг алгоритма.

На рисунке 1.5 изображено пространство, которое алгоритм покрывает во время поиска пути.

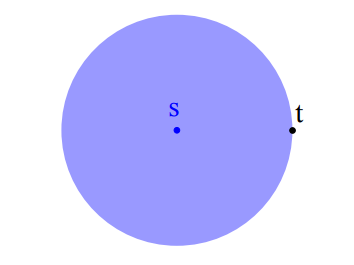


Рисунок 1.5 – Пространство поиска алгоритма Дейкстры

Проанализировав рисунок, можно сказать, что алгоритм проверяет вершины, которые ни при каких обстоятельствах не могут находится на кратчайшем маршруте. Эту проблему решает подход целенаправленного поиска (Goal-Directed search). Некоторые алгоритмы, использующие этот подход будут рассмотрены ниже.

Сложность алгоритма Дейкстры зависит от способа нахождения вершины , а также способа хранения множества непосещенных вершин и способа обновления меток. – количество вершин, а – количество ребер в графе . В простейшем случае основной цикл выполняется порядка раз, в каждом из них на нахождение минимума тратится порядка операций. На циклы по соседям каждой посещаемой вершины тратится количество операций, пропорциональное количеству рёбер . Таким образом, общее время работы алгоритма но, так как , оно составляет

* + 1. **Алгоритм A\***

Рассмотрим алгоритм, использующий подход целенаправленного поиска маршрута. Такой алгоритм впервые описан в 1968 году в работе [12]. Этот метод поиска оптимального пути основан на алгоритме Дейкстры, но он к каждой метке добавляет специальную эвристическую функцию, обозначающую минимальную оценку расстояния от текущей вершины до конечной. В данной работе это расстояние высчитывается по формуле (1), по которой определяется расстояние между двумя точками на поверхности Земли, так как это расстояние является минимально возможным.

(1)

где и — широты, и — долготы заданных начальной и конечной точки соответственно, — расстояние между этими точками, измеряемое в радианах.

Расстояние между точками, измеряемое в километрах, определяется по формуле (2).

(2)

где — средний радиус земного шара.

Метки вершин вычисляются по формуле (3).

(3)

где – итоговая метка, – расстояние, уже посчитанное от вершины до вершины , – оценка расстояния от вершины до вершины , посчитанное по формуле (1) расстояния между точками на поверхности Земли.

Алгоритм работы А\* от классического алгоритма Дейкстры отличается только тем, что для каждой просматриваемой вершины , кроме расстояния до нее от вершины вычисляется минимальная оценка расстояния от до по формуле (1). Несмотря на это, алгоритм значительно увеличивает скорость поиска оптимального маршрута, за счет просмотра меньшего количества вершин графа.

На рисунке 1.4 изображено пространство поиска алгоритма.

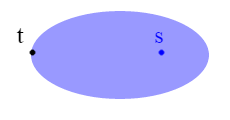


Рисунок 1.6 – Пространство поиска алгоритма А\*

Сравнив рисунки 1.6 и 1.6, можно сделать вывод о том, что алгоритм А\* является более эффективным, чем классический алгоритм Дейкстры. Несмотря на более быстрый поиск оптимального маршрута, сложность алгоритма A\* также оценивается как .

* + 1. **Выбор алгоритмов для построения кратчайшего пути**

В таблице 1.1 представлено табличное сравнение алгоритмов поиска кратчайшего пути.

Таблица 1.1 – Табличное сравнение алгоритмов поиска кратчайшего пути

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Алгоритм** | **Сложность** | **Взвешенный граф** | **Отрицательные веса ребер** | **Эвристика** |
| Поиск в ширину |  | - | - | - |
| Алгоритм Беллмана-Форда |  | + | + | - |
| Алгоритм Флойда- Уоршелла |  | + | + | - |
| Алгоритм Дейкстры |  | + | - | - |
| Алгоритм А\* |  | + | - | + |

Метод составления таблицы расстояний является слишком затратным для дорожной сети, поэтому его применение не целесообразно. В данной работе используется взвешенный ориентированный граф, поэтому алгоритм поиска в ширину для нахождения кратчайшего маршрута неприменим, так как он работает только с невзвешенными графами. Так как – граф дорожной сети, в котором отсутствуют ребра с отрицательными весами, то использовать алгоритмы Беллмана-Форда и Флойда-Уоршелла, которые выполняются за и соответственно, неэффективно, так как алгоритмы Дейкстры и А\* имеют меньшую сложность ‒ .

Таким образом, исходя из проведенного анализа алгоритмов поиска кратчайшего пути, для выполнения этой задачи в данной работе были выбраны алгоритмы Дейкстры и А\*.

* 1. **Анализ алгоритмов поиска субоптимальных путей в графах**

Задача построения оптимального маршрута не всегда может быть решена методами, приведенными выше. При большом количестве факторов, по которым необходимо найти кратчайший путь, эта задача становится практически невыполнимой. Типичным решением этой проблемы является вычисление некоторого множества оптимальных путей, а затем из него выбрать необходимый маршрут с учетом определенных критериев. Так как в данной работе кроме оптимального маршрута необходимо найти безопасный и спортивной, то использование алгоритмов поиска альтернативных путей выглядит достаточно перспективным. Существует несколько алгоритмов нахождения субоптимальных маршрутов, которые описываются далее.

* + 1. **Алгоритм нахождения К кратчайших путей (алгоритм Йена)**

Алгоритм впервые был описан в 1971 году в работе [13]. С помощью этого метода можно найти множество, содержащее путей, которые являются наиболее близкими к оптимальному. Назовем эти маршруты субоптимальными. Стоит отметить, что данный алгоритм находит пути, не имеющие циклов, в отличие от методов, описанных в некоторых других работах.

Пусть – -ый кратчайший путь от вершины до . Пусть – отклонение от пути в -ой вершине. – кратчайший путь, который совпадает с от до , а затем идущий к вершине, отличной от-ых вершин уже построенных кратчайших путей . приходит в вершину по кратчайшему подпути, не проходящему ни через одну из вершин , что означает, что путь не будет иметь циклов.

Алгоритм начинает свою работу с поиска кратчайшего маршрута по любому из алгоритмов, описанных выше. Найденный путь обозначается и добавляется в список , который в итоге будет содержать найденных субоптимальных путей. После этого вычисляются все маршруты для . На каждом шаге находятся все маршруты , а потом они добавляются в список , в котором находятся все найденные на данный момент пути. После нахождения очередного из списка выбирается путь, имеющий минимальную длину, добавляется в список , удаляется из списка . Алгоритм продолжает свою работу, пока в списке не будет записано путей.

Сложность алгоритма Йена зависит от метода, используемого для поиска кратчайшего пути. Наихудшая оценка сложности алгоритма Дейкстры или A\* ‒ . Основной цикл алгоритма близких вызывается раз. Количество итераций внутреннего цикла, который находит все пути , зависит от количества ребер в . В худшем случае это количество равно . Тогда временная сложность алгоритма будет составлять . Исходя из этого, можно сделать вывод, что этот алгоритм является очень затратным по времени.

* + 1. **Алгоритм нахождения E-близких маршрутов**

Этот алгоритм описан в статье [14]. Он находит такие альтернативные пути, которые длиннее кратчайшего маршрута не более, чем на величину Е. Данный алгоритм основан на алгоритме Дейкстры, который на каждом шаге пытается уменьшить метки вершин, которые являются соседями текущей. Этот подход отличается тем, что на каждом шаге он не пытается уменьшать метки, а добавляет их к уже существующим.

Алгоритм состоит из следующих шагов:

1. Расстояние до вершины принимаем равным 0: , так как эта вершина является начально, то , делаем текущей вершиной .
2. Для множества вершин , каждая из которых является смежной с , вычисляются новые метки по формуле:

, где

Также для каждой вершины из запоминается в качестве родительской вершина , то есть . Эта информация о метке добавляется в список меток каждой вершины из . То есть этот список хранит все возможные длины путей до вершины , а также вершину , из которой можно попасть в .

1. Далее список найденных меток сортируется по расстоянию, чтобы в любой момент можно было найти кратчайший маршрут, просто взяв первую метку из этого списка.
2. Вершина отмечается как пройденная.
3. На этом шаге ищется непройденная вершина , у которой первая метка из списка имеет минимальную длину, .
4. Если , то переходим к шагу 7, иначе к шагу 2.
5. Восстанавливаем кратчайший путь по первым меткам вершин, сохраняем его как .
6. На последнем этапе необходимо определить все маршруты, длина которых удовлетворяет формуле:

*,* где – найденный субоптимальный маршрут.

Сами альтернативные пути ищутся рекурсивно, начиная от вершины . Для нахождения всех возможных маршрутов необходимо на каждом шаге обрабатывать каждую метку всех смежных вершин.

Сложность алгоритма нахождения Е-близких маршрутов описывается следующим образом. На первом этапе, при проставлении всех меток, сложность не отличается от сложности алгоритма Дейкстры, то есть в худшем случае она оценивается как . Сложность второго этапа зависит от количества пройденных вершин. Если худший случай (пройдены все вершины), то его сложность оценивается как . Тогда суммарная сложность алгоритма поиска всех Е-близких маршрутов оценивается как .

* + 1. **Выбор алгоритмов для построения субоптимальных маршрутов**

Алгоритмов поиска альтернативных маршрутов на данный момент известно достаточно мало. Существуют методы поиска субоптимальных путей по нескольким критериям одновременно, но в данной работе они не используются, поэтому они и не были рассмотрены. Исходя из этого, для реализации в данной работе были выбраны оба представленных алгоритма для сравнения их эффективности. На данном этапе можно сказать, что алгоритм поиска Е-близких маршрутов выглядит более эффективным из-за меньшей сложности ( против ).

* 1. **Формализация разрабатываемого метода в общем виде**

Исходя из проведенного выше анализа предметной области разрабатываемый метод должен строить маршруты по трем критериям – критерий оптимальности по длине, критерий безопасности и критерий спортивности маршрута. Создаваемый метод будет состоять из трех основных этапов:

1. Построение графа дорог;
2. Построение маршрута;
3. Отображение маршрута на карте.

При этом этап “Построение маршрута” делится на 3 задачи:

1. Построение оптимального маршрута;
2. Построение множества субоптимальных маршрутов;
3. Выбор из множества альтернативных путей того маршрута, который наиболее подходит под критерий безопасности или спортивности в зависимости от выбранного вида маршрута.

В данной работе безопасным считается маршрут, который для найденного множества субоптимальных маршрутов, будет максимально избегать дорог с достаточно оживленным потоком машин, то есть он должен проходить по дорогам, не являющимися главными, если это возможно (такой маршрут есть в множестве найденных субоптимальных маршрутов). Риск попасть в ДТП на таких дорогах резко уменьшается, так как на вспомогательных дорогах меньше машин, и все они едут с достаточно небольшой скоростью. Спортивным маршрутом будем считать путь, который, в отличие, от безопасного проходит по главным дорогам. Во-первых, эти дороги всегда имеют хорошее асфальтовое покрытие, что необходимо при езде на шоссейном велосипеде. Во-вторых на этих дорогах гораздо меньше препятствий (люди, лежачие полицейские и т.д), которые мешают нормальному процессу тренировки. Математическая формализация выбора безопасного или спортивного маршрута из найденных субоптимальных будет описана в конструкторском разделе.

* + 1. **Входные и выходные данные метода**

Первый этап метода, “Построение графа дорог”, в качестве входных данных получает исходные данные OSM и область создания графа. После отработки первого тапа на выходе получается созданный граф дорог, содержащий дороги всех типов, находящихся в заданной области. Второй этап получает созданную на первом этапе структуру дорог и данные от пользователя, которые содержат в себе начальную точку пути, конечную точку, вид маршрута, коэффициент для поиска субоптимальных маршрутов. В результате работы этого этапа получается итоговый путь, состоящий из координат всех точек, составляющих этот маршрут. На вход третьего этапа метода подается эта последовательность точек. Выходными данными этого этапа являются полученная длина маршрута, время его построения и линия на карте, демонстрирующая итоговый путь.

* + 1. **Ограничения разрабатываемого метода**

Разрабатываемый метод не гарантирует того, что будет построен абсолютно кратчайший маршрут, но этот маршрут стремится к таковому. Кратчайшим маршрутом в данной работе является путь, который был построен с помощью алгоритма Дейкстры или А\*. Также, метод не сможет найти такой безопасный или спортивный маршрут, который полностью удовлетворяет всем требованиям этих критериев, так как такие пути ищутся только среди найденных близких к кратчайшему маршрутов, количество которых зависит от заданного коэффициента.

Метод предполагает, что по всем дорогам, для которых был построен граф, возможен проезд в любой момент времени. Таким образом, метод не учитывает всевозможные непредвиденные ситуации, такие как ремонтные работы, перекрытие дорог в связи с какими-то мероприятиями, ДТП, затрудняющие передвижение на конкретном участке дороги и т.д.

Граф дорог занимает достаточно много памяти, а также времени построения даже для одного города, поэтому приложение реализующее данный метод будет строить маршруты в одном районе Москвы, Чертаново Южное. Следует отметить, что это не значит, что разрабатываемый метод не будет работать в других местах. Для его корректной работы в любом месте мира необходимо предварительно создать граф дорог этой местности из электронных карт OpenStreetMap и загрузить его в приложение.

* + 1. **Функциональные требования к разрабатываемому ПО**

Разрабатываемое ПО должно выполнять следующие функции:

1. Отображение карты района построения маршрутов;
2. Ввод начальной и конечной точек маршрута с помощью клика по карте;
3. Выбор вида маршрута;
4. Построение пути в зависимости от выбранного вида;
5. Отображение построенного маршрута на карте;
6. Отображение длины пути и времени его построения.
   1. **Выводы**

В аналитическом разделе были сформулированы цель и основные задачи данной работы. Был проведен анализ предметной области, в ходе которого была обоснована необходимость данной разработки. Для решения задач поиска оптимальных и субоптимальных маршрутов были рассмотрены и проанализированы основные алгоритмы для их решения. По итогам анализа для построения кратчайшего маршрута были выбраны алгоритмы Дейкстры и А\*, а для задачи поиска альтернативных путей – алгоритм Йена и Е-близких маршрутов. В конце раздела в общем виде был формализован разрабатываемый метод, определены его входные и выходные данные, ограничения, а также требуемая функциональность разрабатываемого ПО.

1. **Конструкторский раздел**

В этом разделе приводится подробное описание разрабатываемого метода, выделяются основные этапы его работы, а также входные и выходные данные каждого этапа. Также здесь приводится выбор структуры данных для реализации метода. В конце конструкторского раздела приведена структура разрабатываемого ПО.

* 1. **Функциональная модель метода**

Работу метода можно разделить на 3 этапа:

1. Построение графа дорог;
2. Построение маршрута;
3. Вывод маршрута на карту.

Причем первый этап “Построение графа дорог” выполняется только один раз. Общая функциональная модель метода в виде диаграммы IDEF0 представлена на рисунке 2.1.



Рисунок 2.1 – Общая функциональная модель метода в виде диаграммы IDEF0

На диаграмме – это момент времени, с которого пользователи начинают взаимодействовать с приложением, так как этап построения графа дорог выполняется только при установке приложения на сервер, либо при перестроении графа для обновления информации о дорогах. Остальные 2 этапа выполняются при каждом запросе через интерфейс приложения.

На вход первого этапа метода подаются исходные данные об электронной карте OSM и область, в которой должны строится маршруты. Создание графа дорог целого города или страны занимает очень большое количество времени, поэтому необходимо определить область поиска путей. Результатом выполнения первого этапа является готовый граф, который больше не будет изменяться во времени, поэтому он передается в качестве из одного входных данных во второй этап работы метода.

Кроме дорожного графа на вход этапу построения маршрута передаются начальная точка, конечная точка и вид маршрута. Этот этап является самым сложным и трудоемким, поэтому для более подробного описания его работы необходима декомпозиция этого этапа, которая представлена на рисунке 2.2.

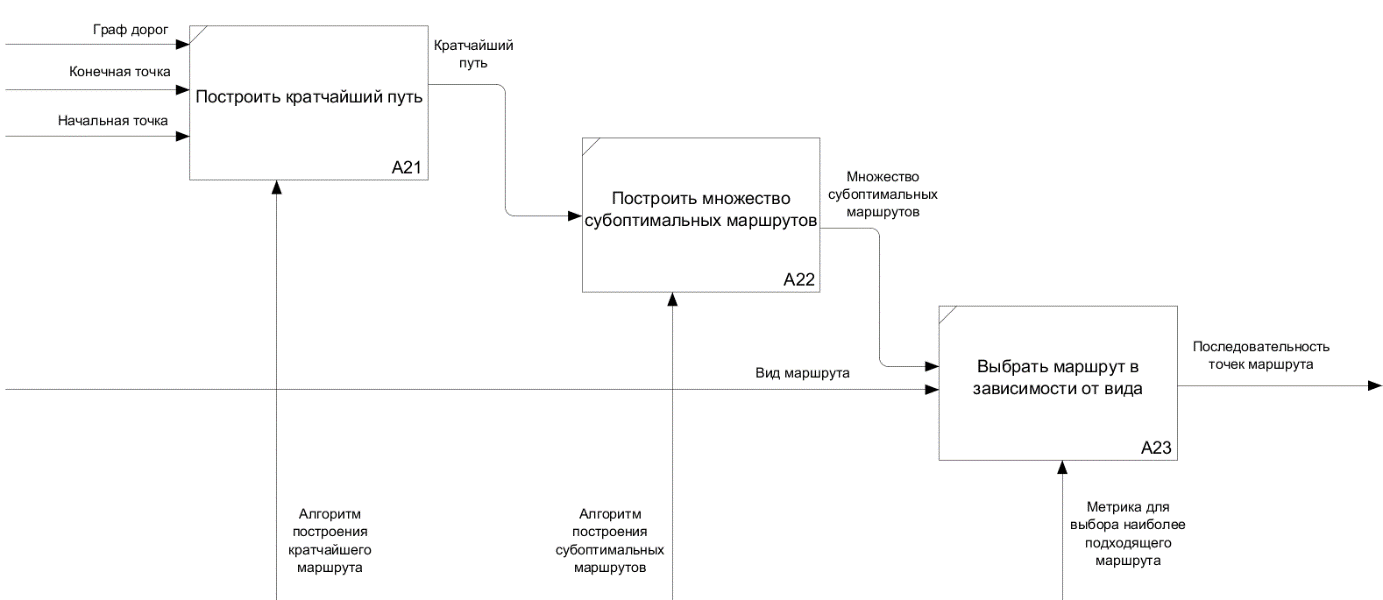


Рисунок 2.2 – Диаграмма IDEF0 для этапа построения маршрута

В этапе построения маршрута можно выделить 3 задачи:

1. Построение кратчайшего маршрута;
2. Построение множества субоптимальных маршрутов;
3. Выбор из множества альтернативных путей.

Для выполнения задачи построения кратчайшего маршрута механизмом служит алгоритм Дейкстры или А\* в зависимости от выбора пользователя. В качестве выходных данных этого этапа выступает построенный кратчайший маршрут, который далее идет на вход этапу построения множества альтернативных путей. Эта задача выполняется с помощью алгоритма Йена или поиска Е-близких маршрутов, которые выступают в качестве механизма для этого этапа метода. Полученное множество субоптимальных путей передается следующему этапу вместе с выбранным пользователем видом маршрута. С помощью механизма метрики выбора наиболее подходящего маршрута из множества альтернативных путей выбирается один, который удовлетворяет условиям метрики. На выходе этого этапа получаем совокупность точек, составляющих найденный маршрут. После этого работа этапа построения маршрута заканчивается и итоговый путь передается в качестве входных данных последнему этапу “Отображение маршрута на карте”, который выводит маршрут пользователю в виде линии.

* 1. **Построение графа дорог**
     1. **Структура OpenStreetMap**

Исходными данными о карте является XML-файл open-source проекта OpenStreetMap. В этом XML-представлены все данные о дорогах Москвы и Московской области. Для начала необходимо рассмотреть структуру этих данных. В целом ее можно представить в виде схемы, показанной на рисунке 2.3.

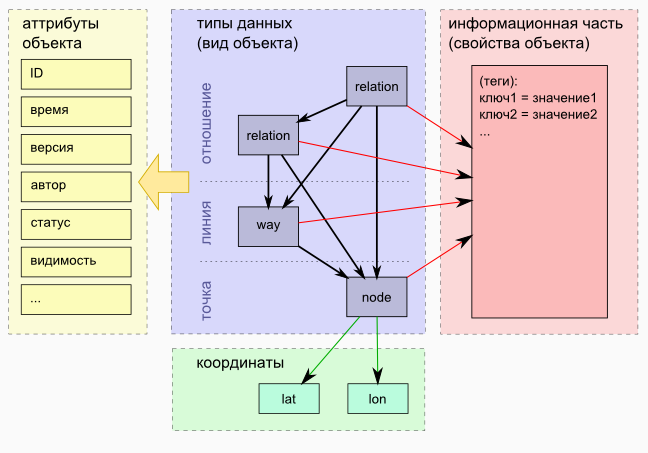


Рисунок 2.3 – Схематичное представление структуры данных OSM

Все данные делятся на три группы:

* Типы данных. Они описывают сам объект, как некоторую пространственную сущность, имеющую известные координаты всех частей этого объекта;
* Информационная часть. В ней хранятся характеристики объекта, не имеющие отношения к его географическому положению. К информационной части объекта относятся его название, физические, логические и другие свойства;
* Служебные атрибуты. Они необходимы того, чтобы правильно организовать процессы хранения и обработки объектов. Эти атрибуты включают в себя уникальный идентификатор, состояние объекта в базе, время последней правки объекта в базе и т.д.

В OSM три базовых типа: точка (node), линия (way) и отношение (relation). Абсолютно все объекты OSM описываются этими тремя типами данных. Любой объект обязательно имеет свой идентификатор (ID), который является уникальным в пределах типа данных. Модель данных в OSM строится с помощью иерархической ссылочной структуры. Это означает, что каждый последующий объект не содержит в себе информацию о предыдущих объектах, а только ссылается на некоторое их множество, образуя новую сущность. Теперь рассмотрим все типы данных по порядку.

Точка (node) является базовым типом в иерархической модели. Только этот тип данных хранит географическую информацию об объекте – координаты в виде широты и долготы. В XML нотации объект типа node выглядит следующим образом:

<node id='23' lat='58.875047918145675' lon='49.785240674006126' />

Такая запись представляет собой точку с уникальным идентификатором 23 и парой координат. Точка может являться полноценным объектом (например, станция метро) или частью объекта другого типа. Иногда она одновременно можеть быть самостоятельным объектом и являться частью другого.

Линия (way) является совокупностью указателей на точки. Линия должна содержать ссылку минимум на один, уже объявленный, объект типа node. Правильная XML нотация объекта типа линия будет заключаться в описании всех необходимых точек, после чего следует сама запись о линии, в которой перечисляются все её точки. В простейшем варианте это будет выглядеть так:

<node id='19' lat='58.888047127548994' lon='49.747870758186764' /> <node id='23' lat='58.875047918145675' lon='49.785240674006126' />

<node id='22' lat='58.86687448573524' lon='49.737090974777324' />

<way id='24'>

<nd ref='22' />

<nd ref='23' />

</way>

<way id='48'>

<nd ref='19' />

<nd ref='23' />

</way>

Порядок точек в линии важен, так как он характеризует их последовательность в линии и ее направление. В данном случае мы описали три объекта типа node и два объекта типа way. Объявленные линии ссылаются на точки, описанные выше. Оба пути ссылаются на одну и ту же точку с id=23, это означает, что линии 24 и 48 пересекаются. Такие пересечения являются вершинами для сроящегося графа дорог в том случае, если эти объекты типа way являются дорогами.

Следующим типом является отношение (relation). На самом деле все типы, кроме точек, являются отношением, но линии выделены в отдельный тип данных как наиболее используемые. Отношение отличается от линии тем, что оно является совокупностью объектов любых типов. Можно создать следующий объект типа отношение:

<relation id='31'>

<member type='way' ref='24' />

<member type='node' ref='19' />

</relation>

Так как отношения обозначают такие объекты, как маршруты общественного транспорта, границы озер, лесов, зданий, то этот тип данных не используется в настоящей работе, поэтому нет необходимости описывать его более подробно.

Тип объекта ничего не говорит о характеристиках и назначении объекта. Для этого существует информационная часть структуры данных OpenStreetMap, которая основана на принципах тегирования объектов. Теги создаются в виде пары ключ=значение. Если расширить объявленные выше объекты с помощью тегов, то можно получить следующее:

<node id='23' lat='58.87753645355202' lon='49.79290110146539'>

<tag k='highway' v='traffic\_signals' />

</node>

<node id='22' lat='58.87456113991739' lon='49.73690926857261' />

<node id='19' lat='58.89362576054878' lon='49.7492065402827' />

<way id='48'>

<nd ref='19' />

<nd ref='23' />

<tag k='embankment' v='yes' />

<tag k='highway' v='secondary' />

<tag k='incline' v='up' />

<tag k='lanes' v='2' />

<tag k='maxspeed' v='60' />

<tag k='name' v='улица Пожарского' />

</way>

<way id='24'>

<nd ref='22' />

<nd ref='23' />

<tag k='highway' v='primary' />

<tag k='lanes' v='6' />

<tag k='lit' v='yes' />

<tag k='name' v='проспект Минина' />

<tag k='oneway' v='yes' />

<tag k='ref' v='М84' />

</way>

Линия 48 стала «улицей Пожарского», с ограничением скорости в 60 км/час, с двумя полосами, имеющей положительный градиент уклона от точки 19 к точке 23, являющейся второстепенной дорогой и поднятой относительно уровня земли на насыпь. А линия 24 является основной дорогой (классом выше чем secondary) с 6 полосами движения, имеющей стационарное освещение и одностороннее движение разрешённое в направлении от точки 22 в сторону точки 23, носит название «проспект Минина» и является частью федеральной трассы М84. Обе дороги имеют общую точку 23, которая является перекрёстком со светофором. Все правила тегирования объектов описаны в wiki-документации проекта OSM [15], поэтому нет необходимости рассматривать их более подробно. В данной работе самым важным тегом типа way является highway, которым обозначаются все дороги. Следовательно, все линии, не имеющие этого тега не имеют ценности в данной работе, а значит и множество точек, которые являются частями этих линий. Для удобства обработки данных о карте Москвы было решено на основе XML-файла создать базу данных, с помощью которой будет проводиться построение графа дорог.

* + 1. **Структура данных для представления графа дорог**

Схема БД, спроектированной для хранения исходных данных представлена рисунке 2.4.

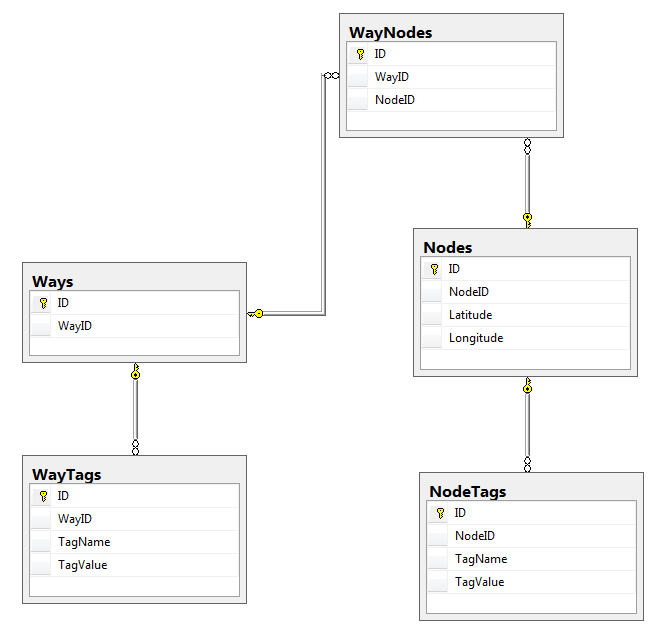


Рисунок 2.4 – Схема БД исходных данных о карте

После добавления в базу данных информации обо всех объектах из XML-файла из нее необходимо удалить избыточные данные. Во-первых, это данные о линиях, не содержащих тега highway. Во-вторых, это данные о дорогах, по которым невозможно или запрещено проехать на велосипеде (эти дороги были описаны в аналитическом разделе). После получения исходных данных о карте в удобном виде необходимо определить структуру данных для хранения графа дорог.

Существует два основных способа представления графа в памяти компьютера – это списки смежности и матрица смежности.

Матрица смежности является удобным представлением плотных графов, в которых . В данном представлении мы заполняем матрицу размером следущим образом:

(Если существует ребро из в )

(Иначе)

Данный способ подходит для ориентированных и неориентированных графов. Для неориентированных графов матрица A является симметричной (то есть , т.к. если существует ребро между и , то оно является и ребром из в , и ребром из в ). Благодаря этому свойству можно сократить почти в два раза использование памяти, храня элементы только в верхней части матрицы, над главной диагональю). С помощью данного способа представления, можно быстро проверить есть ли ребро между вершинами и , просто посмотрев в ячейку С другой стороны этот способ очень громоздкий, так как требует памяти для хранения матрицы. На рисунке 2.5 представлен граф и соотвествующая ему матрица смежности.

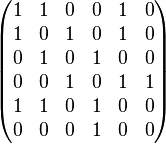
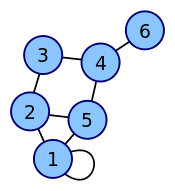


Рисунок 2.5 – Граф и соотвествующая ему матрица смежности

Списки смежности больше подходят для хранения разреженных графов, где . Такими графами как раз и являются дорожные сети, используемые в этой работе. В данном представлении используется массив содержащий списков. В каждом списке содержатся все вершины , такие что между и есть ребро. Память требуемая для представления равна что является лучшим показателем чем матрица смежности для разреженных графов. Главный недостаток этого способа представления в том, что нет быстрого способа проверить существует ли ребро Список смежности для графа из рисунка 2.5 представлен на рисунке 2.6.

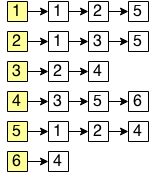


Рисунок 2.6 – Список смежности графа

Проблема описанных выше подходов для представления графа в памяти заключается в том, что дорожные сети обычно очень большого размера с количеством вершин, в лучшем случае, более 1000. Это значит, что для хранения такого графа необходимо много памяти. Кроме того, нужно учесть тот факт, что дорожный граф редко изменяется, поэтому генерировать его каждый раз при запуске приложения нецелесообразно. Исходя из этого было принято решение хранить граф в постоянной памяти на сервере в виде базы данных. Реляционная БД не совсем подходит для этой цели, так как достаточно проблематично представить граф в виде таблиц, кроме того с ним просто неудобно работать в таком виде. Поэтому было решено использовать нереляционную базу данных, которая хранит объекты в виде узлов и связей между ними. Такая БД называется графовой. Самой известной графовой СУБД является Neo4j, поэтому она и была выбрана для хранения дорожной сети. В этой СУБД используется собственный язык запросов Cypher для получения необходимых данных.

Для выделения ключевых сущностей и связей проектируемой базы данных необходимо описать ER-моедль данных. На рисунке 2.7 представлена ER-диаграмма графа дорог, используемого в данной работе.

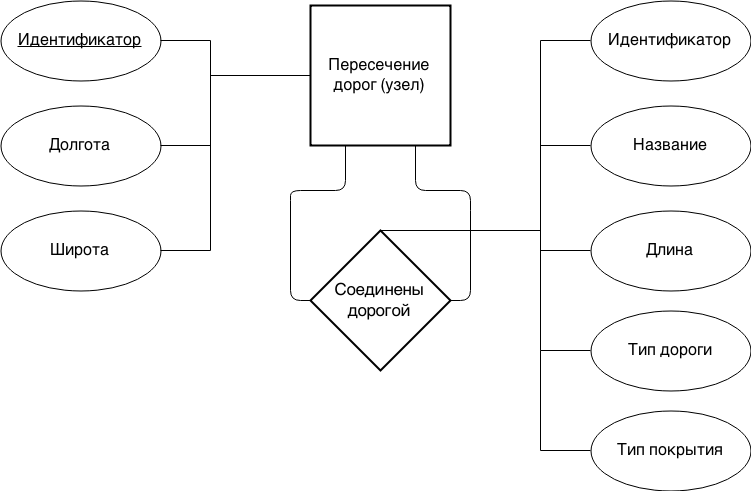


Рисунок 2.7 – ER-диаграмма в нотации Чена

В данной модели одна сущность – пересечение дорог, которая имеет следующий свойства: идентификатор, долгота, широта. Сущности связаны между собой с помощью единственного отношения – дорога, которая также имеет ряд свойств (идентификатор, название, длина, тип дороги, тип покрытия). В СУБД Neo4j пересечение дорог будет представлено в виде вершины с указанными свойствами. Neo4j поддерживает только ориентированные графы, поэтому дорога представляется в виде одного ребра, если дорога имеет одностороннее движение, двумя ребрами с разной направленностью, если движение по дороге является двусторонним. Небольшая часть дорожного графа, добавленного в БД Neo4j представлена на рисунке 2.8.

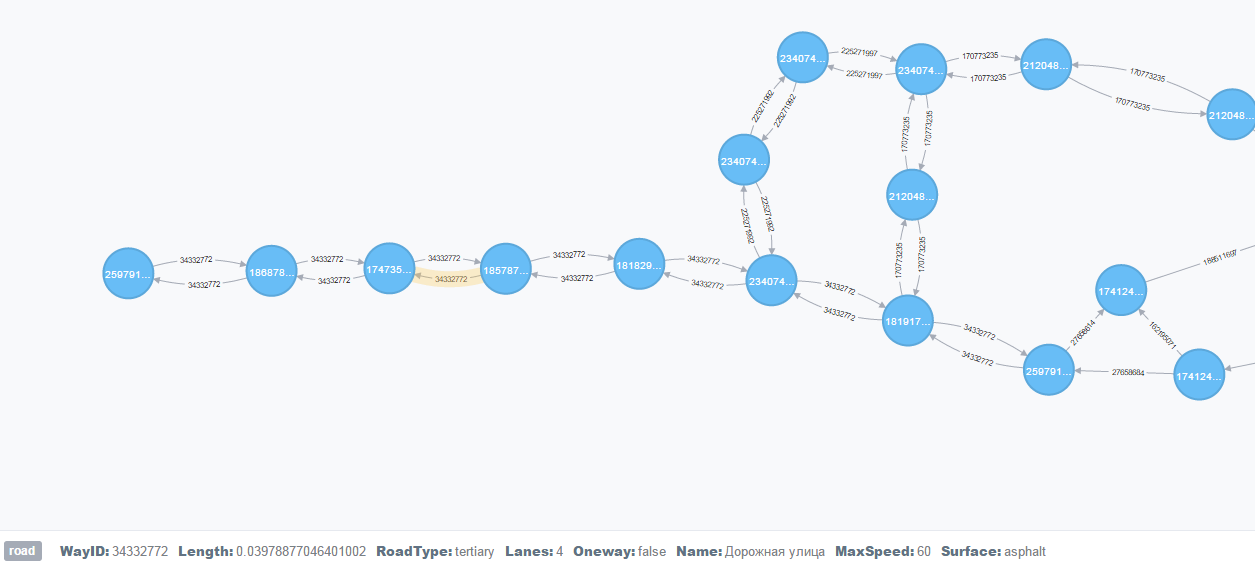


Рисунок 2.8 – Дорожный граф в СУБД Neo4j

Заполнение БД происходит с использованием базы данных с информацией о карте, описанной выше в этом пункте.

* 1. **Построение кратчайшего маршрута**

Этап построения оптимального пути выполняется при помощи одного из выбранных и описанных в аналитическом разделе алгоритмов. В данной работе для решения этой задачи используются алгоритмы Дейкстры и А\*.

На рисунке 2.9 приведена схема работы алгоритма Дейкстры.

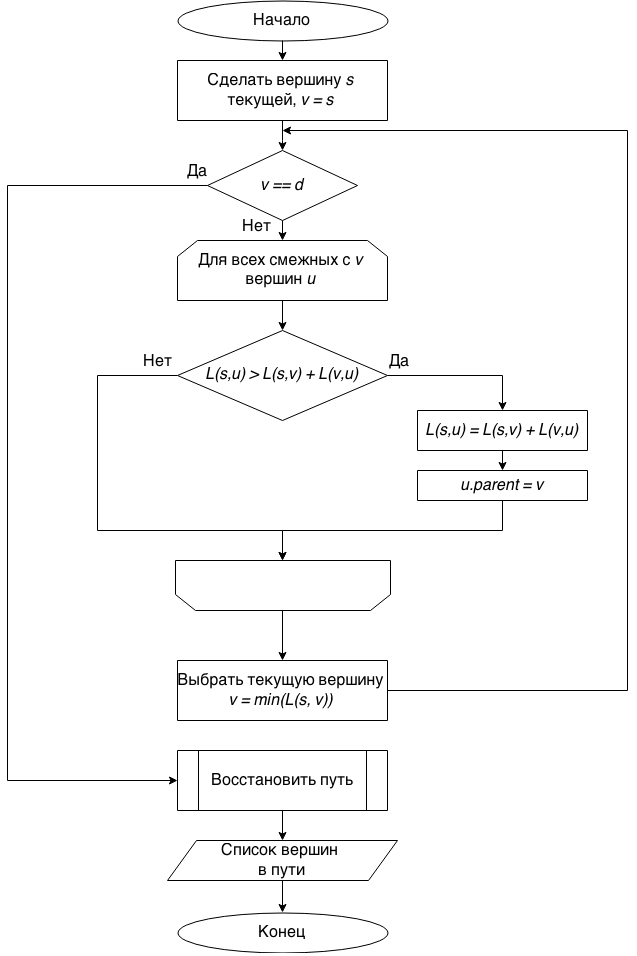


Рисунок 2.9 – Схема алгоритма Дейкстры

На рисунке 2.10 представлена схема работы алгоритма А\*.

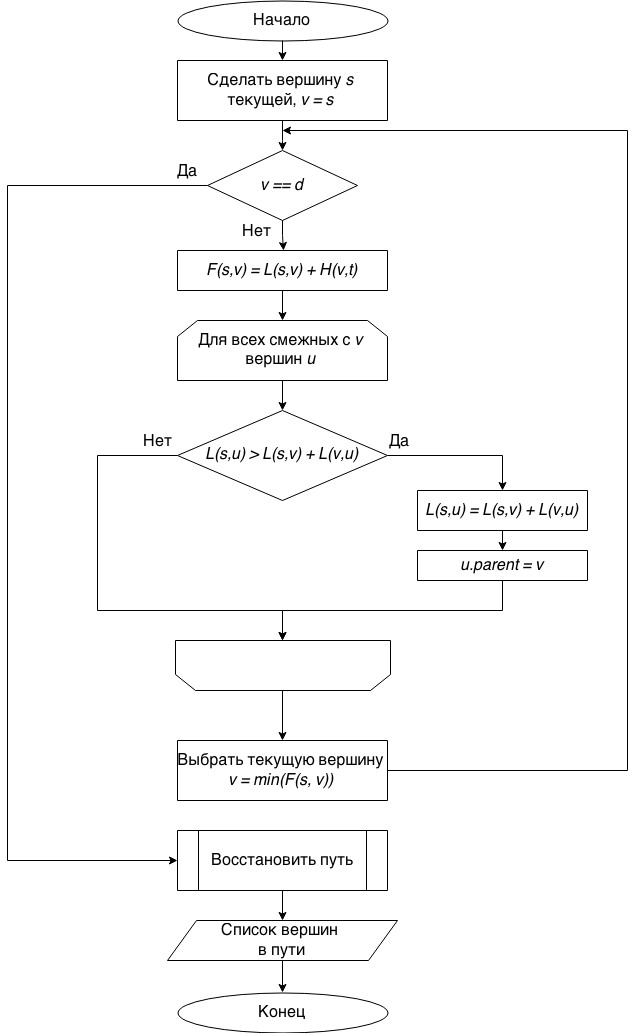


Рисунок 2.10 – Схема алгоритма А\*

* 1. **Построение субоптимальных маршрутов**

Этап построения альтернативных путей выполняется при помощи одного из выбранных и описанных в аналитическом разделе алгоритмов. В данной работе для решения этой задачи используются алгоритмы Йена и поиска Е-близких маршрутов.

На рисунке 2.11 приведена схема работы алгоритма Йена для нахождения K альтренативных маршрутов.



Рисунок 2.11 – Схема алгоритма Йена

На рисунке 2.12 представлена схема работы алгоритма поиска Е-близких маршрутов.

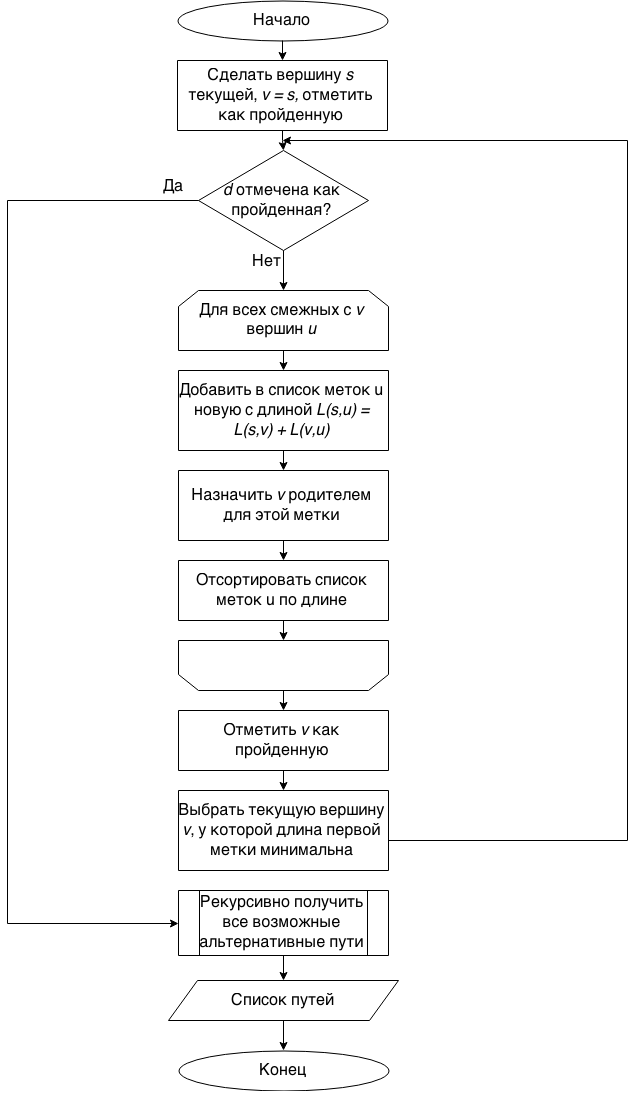


Рисунок 2.12 – Схема алгоритма поиска Е-близких маршрутов

* 1. **Выбор маршрута в зависимости от вида**

Для выбора безопасного и спортивного пути необходимо математически формализовать метрику выбора. Так как на предыдущем этапе были найдены маршруты, которые являются близкими к оптимальному, то на выбор нужного пути не будет влиять его длина. В аналитическом разделе были описаны требования к безопасному и спортивному маршруту, которые подразумевают, что пользователь, выбравший для построения безопасный маршрут, хочет избежать дорог с оживленным транспортным потоком, а тот, кто выбрал спортивный маршрут, наоборот хочет проехать по дорогам, где находится наименьшее количество различных препятствий, таких как прохожие, лежачие полицейские, парковки машин и т.д. Из этого можно сделать вывод, что безопасный и спортивный маршрут являются полярными. Это означает, что необходимо ввести такой коэффициент, который будет подсчитан для каждого маршрута из множества найденных альтернативных путей, минимум которого будет означать безопасность маршрута, а максимум – маршрут для спортивной езды. Для этого все используемые типы дорог были расположены в порядке возрастания безопасности проезда на велосипеде, а каждому типу был присвоен коэффициент, показывающий степень безопасности проезда по дороге данного типа. Типы дорог соответствуют возможным значениям тега highway. Результаты этих действий приведены в таблице 2.1.

Таблица 2.1 – Коэффициенты безопасности для дорог разных типов

|  |  |  |
| --- | --- | --- |
| Тип дороги | Описание | Коэффициент безопасности |
| trunk | Важные дороги не являющиеся автомагистралями | 0.2 |
| primary | Центральные магистрали городов | 0.3 |
| secondary | Основные магистрали районов города | 0.4 |
| teritary | Основные микрорайонные или межмикрорайонные транзитные улицы. | 0.5 |
| unclassified | Обычные мелкие улицы с нежилыми строениями вдоль улицы и проезды | 0.6 |
| residental | Дороги, которые проходят внутри жилых зон, а также используются для подъезда к ним. | 0.7 |
| service | Служебные проезды: внутридворовые, въездные, парковочные и т.д. | 0.8 |
| living\_street | Улицы, на которых пешеходы имеют преимущество перед автомобилями | 0.9 |
| pedestrian | Для обозначения улиц и площадей городов, имеющих твердое покрытие, выделенных для пешеходов | 0.9 |
| track | Дороги сельскохозяйственного назначения, лесные дороги | 0.9 |
| footway | Пешеходные дорожки и тротуары | 1 |
| cycleway | Велодорожки | 1 |

Теперь для расчета меры безопасности каждого маршрута необходимо перемножить коэффициенты из таблицы 2.1 всех ребер, входящих в этот маршрут, причем для более точного расчета коэффициента необходимо учитывать длину ребра, так как проехать, например, 100 метров по шоссе, чтобы потом ехать по велодорожке будет безопаснее, чем ехать 500 метров по улице, соединяющей микрорайоны, а только потом переехать на велодорожку. Для этого введем единицу длины, равную 100 метрам. Исходя из вышеизложенного итоговая формула расчета меры безопасности маршрута будет выглядеть следующим образом:

где – количество ребер пути, – коэффициент -го ребра пути, зависящий от типа дороги, – длина ребра, – единица длины, принимается равной 0,1км.

Маршрут является наиболее безопасным из найденного множества субоптимальных путей, если

Маршрут является наиболее пригодным для спортивной езды, если

где – количество построенных субоптимальных маршрутов.

Таким образом, на этом этапе метода из найденного множества альтернативных путей выбирается единственный маршрут, который и выводится пользователю в качестве результата.

* 1. **Структура разрабатываемого ПО**

В общем виде структуру разрабатываемого программного обеспечения можно представить в виде диаграммы компонентов, которая иллюстрирует основные модули программы, а также связи между ними. Такая диаграмма компонентов представлена на рисунке 2.13.

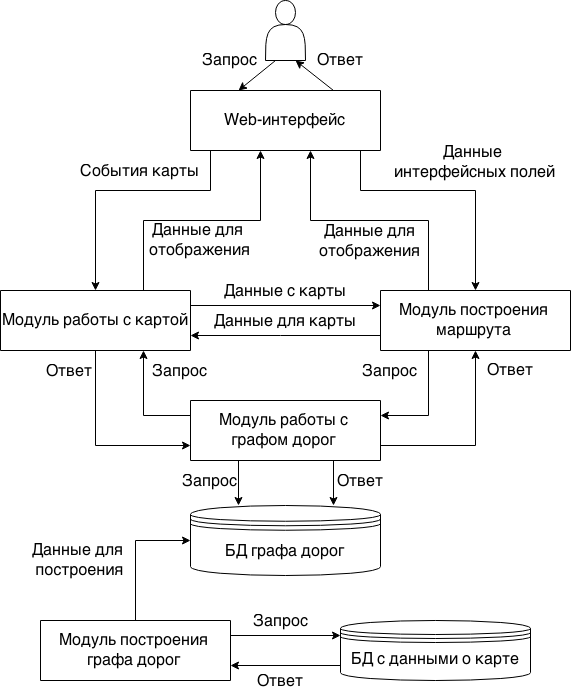


Рисунок 2.13 – Диаграмма компонентов ПО

Как уже говорилось выше, создание графа выполняется один раз, соответственно, модуль построения графа дорог вызывается на выполнение только при установке приложения на сервер. Этот модуль использует БД для получения информации о добавляемых в графовую базу данных вершинах и ребрах.

После построения графа дорог в приложение начинают поступать запросы пользователя посредством web-интерфейса. Данные, получаемые от web-интерфейса бывают двух видов: события карты (ввод начальной и конечной точки маршрута) и данные интерфейсных полей, содержащие информацию о выбранных алгоритмах поиска путей, требуемые коэффициенты, а также вид маршрута. События карты сначала поступают в модуль работы с картой, а после их обработки с помощью модуля работы с графом дорог, они поступают вместе с данными интерфейсных полей в модуль построения маршрута. Модуль построения маршрута в ходе своей работы постоянно отправляет запросы модулю работы с графом дорог, который имеет доступ к графу, хранящемуся в БД.

* 1. **Способы тестирования ПО**

Для тестирования разрабатываемого программного обеспечения были выбраны модульное и системное тестирование. С помощью модульного тестирования будет проверена работоспособность модуля работы с графом дорог, так как этот модуль является наиболее важным из-за того, что все запросы проходят через него. С помощью системного тестирования через интерфейс разработанной программы будет проверена правильность работы всей системы в целом.

* 1. **Выводы**

В ходе выполнения конструкторского раздела была представлена функциональная модель разработанного метода, определены его основные этапы. Были описаны входные и выходные данные для каждого этапа. Также, здесь был полностью описан разработанный метод с помощью необходимых формул и схем используемых алгоритмов. Далее была приведена структура данных для хранения графа дорог и обоснован ее выбор. В конце раздела была представлена структура программного обеспечения, реализующего разработанный метод, а также предложены способы тестирования приложения.

1. **Технологический раздел**

Данный раздел содержит обоснованный выбор средств программной реализации разработанного метода. Раздел содержит подробное описание установки приложения Также здесь будут описаны пользовательский интерфейс и методики тестирования программного обеспечения.

* 1. **Выбор языка программирования и средств разработки**

Для разработки приложения был выбран язык С# для операционной системы Windows с использованием программной платформы .NET Framework. C# является современным языком программирования, поддерживающим объектно-ориентированную парадигму программирования. Кроме того, язык C# имеет достаточно хорошую производительность, которая важна в данной работе. Платформа .NET содержит огромную библиотеку классов для работы с различными типами данных, что заметно ускоряет скорость разработки и уменьшает количество программного кода. Так как в данной работе используются базы данных, как реляционные, так и нереляционные, то важным фактором выбора языка являлось наличие библиотек для удобной работы с этими базами данных. Платформа .NET содержит в себе библиотеку LINQ, являющуюся удобным средством доступа к СУБД Microsoft SQL Server. Для работы с графовой базой данных Neo4j существует сторонняя библиотека для языка C#, которая будет описана в подразделе 3.2.

Так как целью данной работы является разработка Web-приложения, то необходимо было выбрать фреймворк, подходящий для его реализации. Платформа ,NET предоставляет фреймворк ASP .NET MVC, позволяющий разрабатывать Web-приложения на основе шаблона Model – View – Controller.Этот шаблон позволяет разделить обязанности различных компонентов приложения. Контроллер принимает и обрабатывает запросы, взаимодействуя с моделями, которые предоставляют доступ к данным. После обработки запроса контроллер отправляет полученные в результате данные представлению, которое возвращает пользователю результат обработки запроса в удобном для него виде.

Интерфейс приложения, отображаемый с помощью представлений, написан средствами HTML, CSS и JavaScript.

В качестве IDE для разработки было выбрана Microsoft Visual Studio 2012, которая предоставляет множество инструментальных средств для разработки на языке C# с использованием .NET. Visual Studio включает в себя графический редактор кода с автодополнением, встроенный компилятор языка C#, средства отладки и тестирования разрабатываемого ПО.

* 1. **Выбор библиотек для разработки**

Как уже говорилось выше, для языка C# существует специальная библиотека под названием Neo4jClient, которая через специальный интерфейс позволяет получать доступ к базе данных Neo4j. Благодаря этому средствами языка C# можно добавлять/удалять новые узлы и связи в граф, находящийся в БД. Кроме того, существует возможность выполнения различных запросов в базу данных с помощью языка Cypher для получения необходимых данных. Данная библиотека значительно упрощает всю работу с графом дорог.

Другой библиотекой, необходимой для разработки ПО является JavaScript-библиотека LeafLet, которая предоставляет интерфейс для работы с картами. С помощью этой библиотеки можно загружать изображение карты на сайт, обрабатывать события карты, рисовать различные линии и маркеры на карте.

Также в данной работе будет использована библиотека NUnit, с помощью которой будет проведено модульное тестирование разрабатываемого ПО.

* 1. **Развертывание приложения**

В основном все веб-приложения работают через глобальную сеть Интернет, поэтому для работоспособности приложения необходим сервер, имеющий постоянный доступ к сети. Поэтому необходимо воспользоваться услугами хостингов. Для развертывания приложения ASP .NET MVC проще всего воспользоваться облачной платформой от Microsoft – Windows Azure. Эта платформа предоставляет основные возможности выполнения различных приложений и хранения данных на серверах, постоянно находящихся в сети Интернет. Также эта платформа умеет работать с необходимыми в данной работе СУБД – Microsoft SQL Server и Neo4j.

Общая UML-диаграмма развертывания ПО представлена на рисунке 3.1.

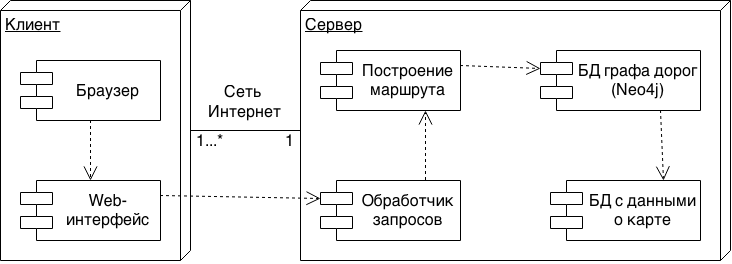


Рисунок 3.1 – UML-диаграмма развертывания приложения

На данной диаграмме сервером является рабочая станция, предоставляемая платформой Windows Azure для развертывания приложения и хранения на ней необходимых данных. Клиентом является компьютер пользователя сайта. С помощью веб-браузера через веб-интерфейс приложения пользователь посылает запрос на сервер через сеть Интернет. При получении запроса сервер обрабатывает его, строит необходимый маршрут используя БД с дорожной сетью. Сервер всегда существует в единственном экземпляре, клиентов же, наоборот, может быть много.

Для того, чтобы создать и развернуть сайт в Windows Azure необходимо зарегистрироваться на <http://azure.microsoft.com/ru-ru/pricing/free-trial/> [16] и пройти следующий шаги.

1. При успешном создании аккаунта отобразится страница, представленная на рисунке 3.2.

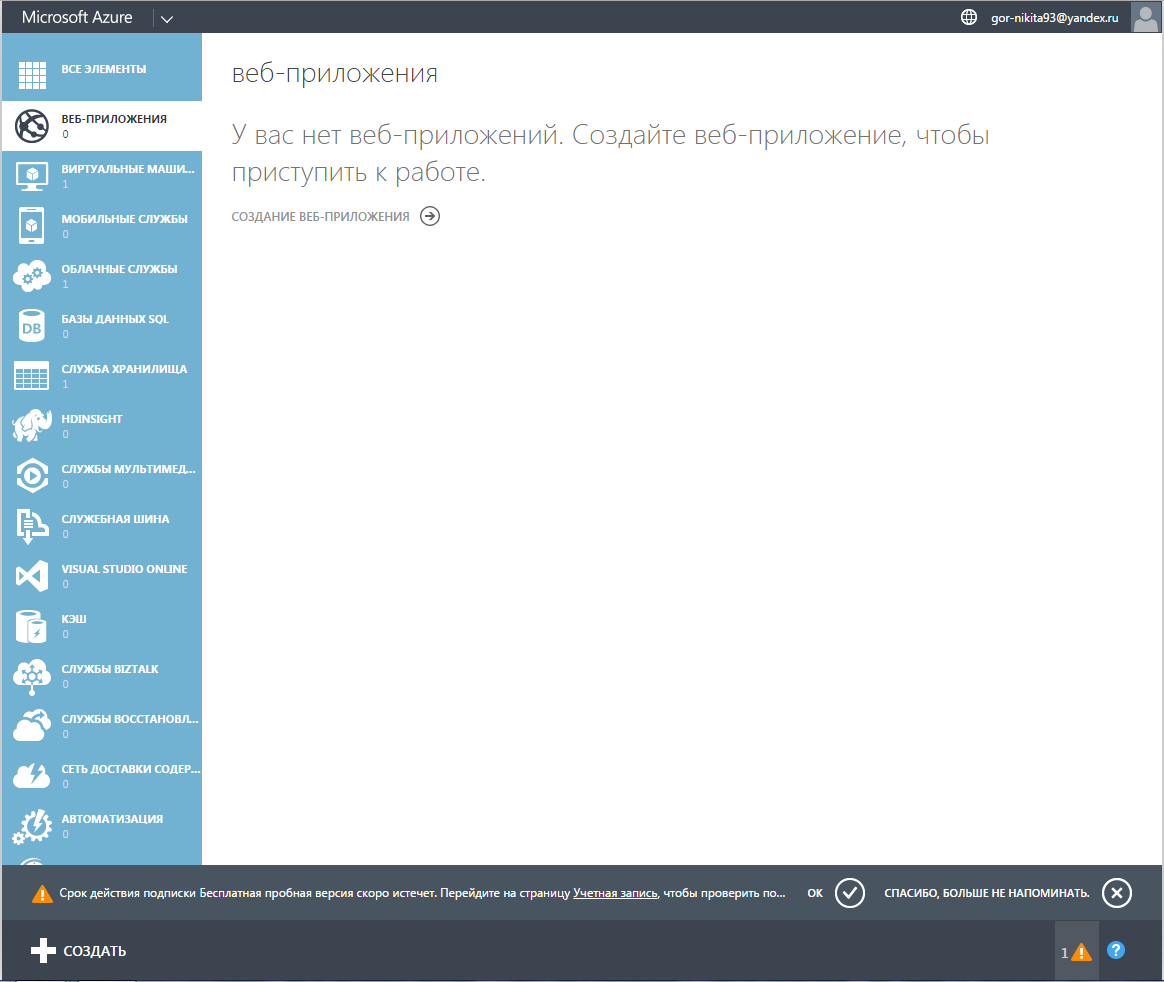


Рисунок 3.2 – Стартовая страница Windows Azure

1. Нажимаем на кнопку CREATE A WEB SITE, появится интерфейс создания с выбранным по умолчанию вариантом – быстрое создание. Этот интерфейс проиллюстрирован на рисунке 3.3.

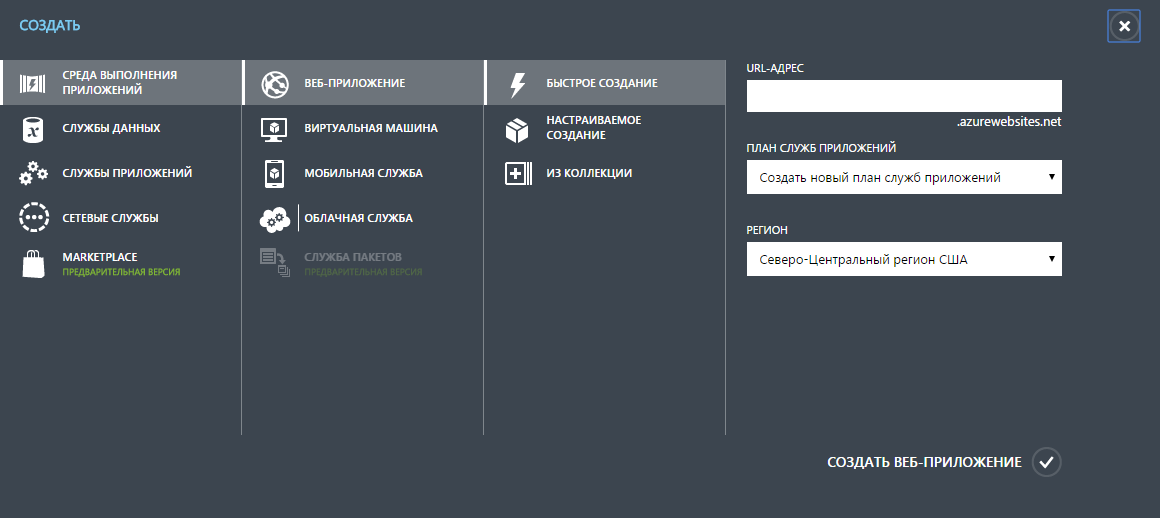


Рисунок 3.3 – Интерфейс создания сайта

1. Далее необходимо создать сайт с использованием базы данных. Для этого нажимаем на “Настраиваемое создание”. После этого появляется окно, показанное на рисунке 3.4.

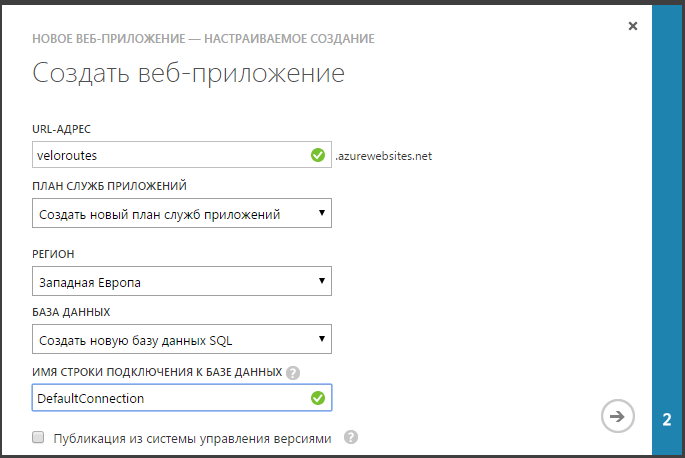


Рисунок 3.4 – Настраиваемое создание веб-приложения

Здесь заполняем необходимые поля и переходим далее.

1. Появляется окно, продемострированное на рисунке 3.5.

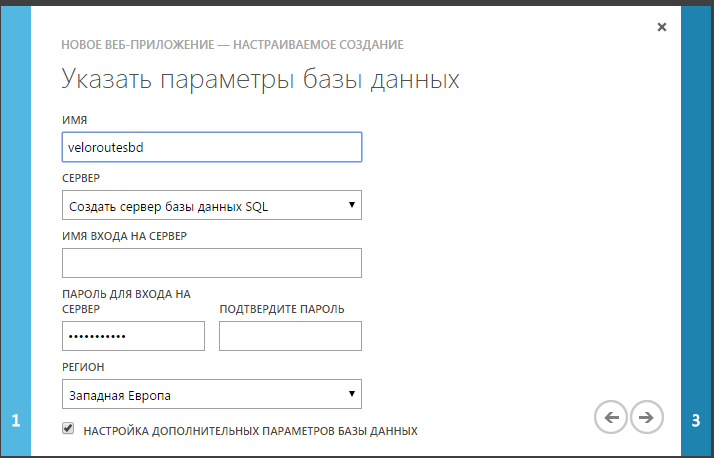


Рисунок 3.5 – Параметры БД

Указываем необходимые параметры и переходим на следующий шаг.

1. Интерфейс этого шага представлен на рисунке 3.6.

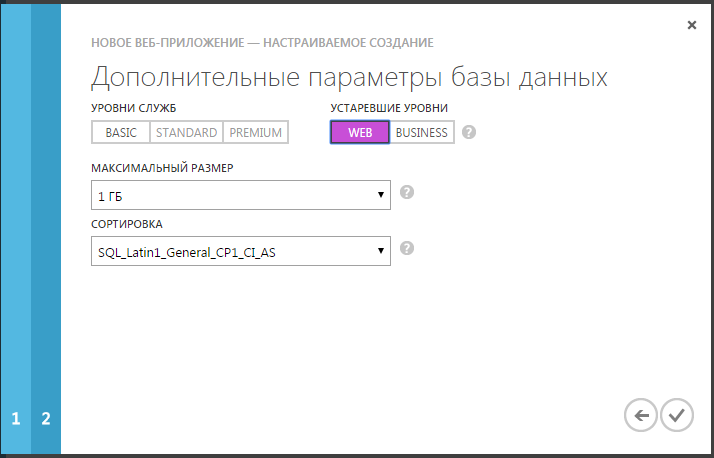


Рисунок 3.6 – Дополнительные параметры БД

На этом шаге заканчивается создание веб-приложения.

1. На этом шаге необходимо сохранить информацию о строке подключения и загрузить профиль публикации. Это можно сделать на странице приложения (рисунок 3.7).

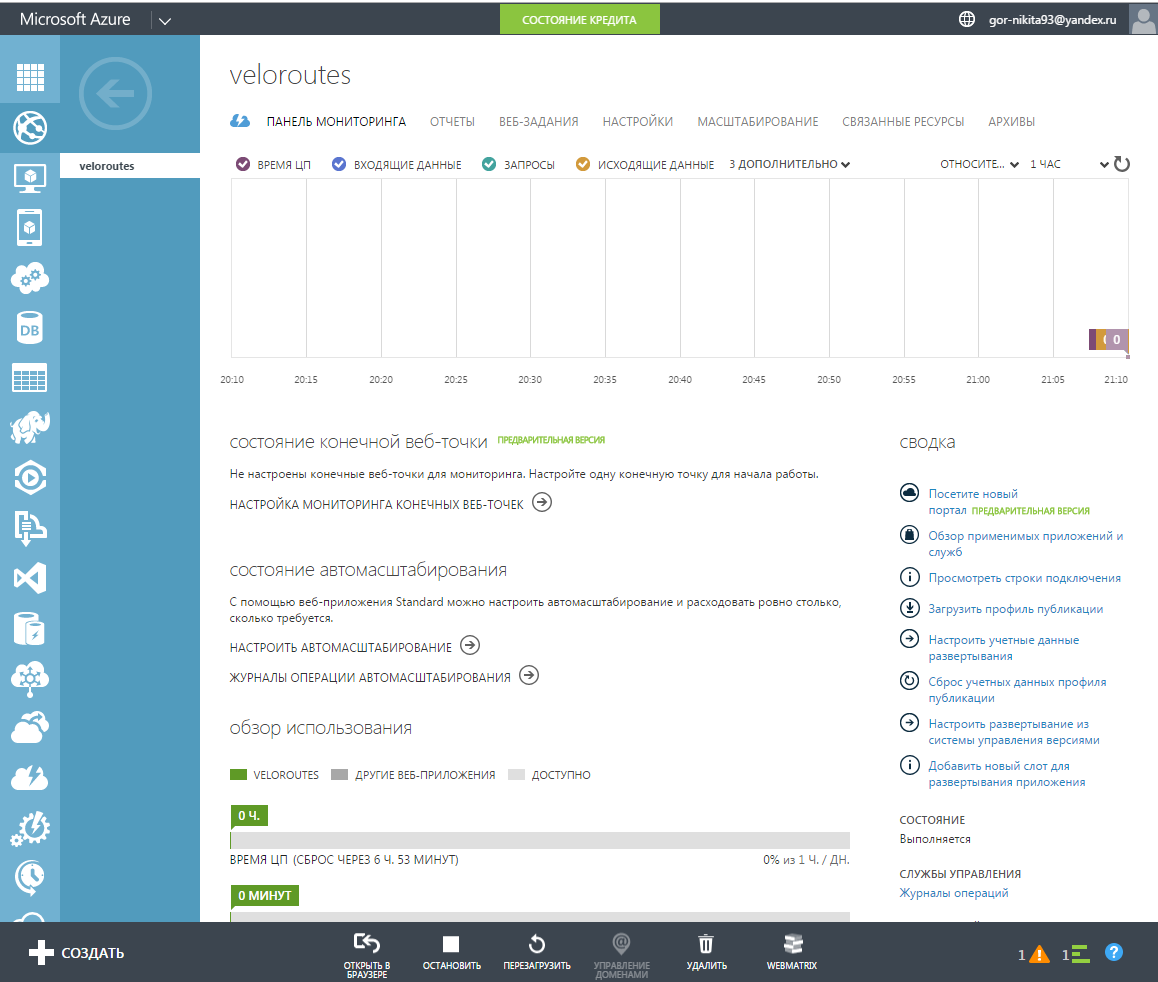


Рисунок 3.7 – Страница созданного приложения

1. Далее необходимо перейти в Visual Studio 2012 с открытым проектом приложения и в меню выбрать Построение – Публикация выбранных элементов. Появится окно, изображенное на рисунке 3.8. Нажимаем кнопку Импорт и выбираем скачанный файл профиля публикации. Нажимаем Далее.

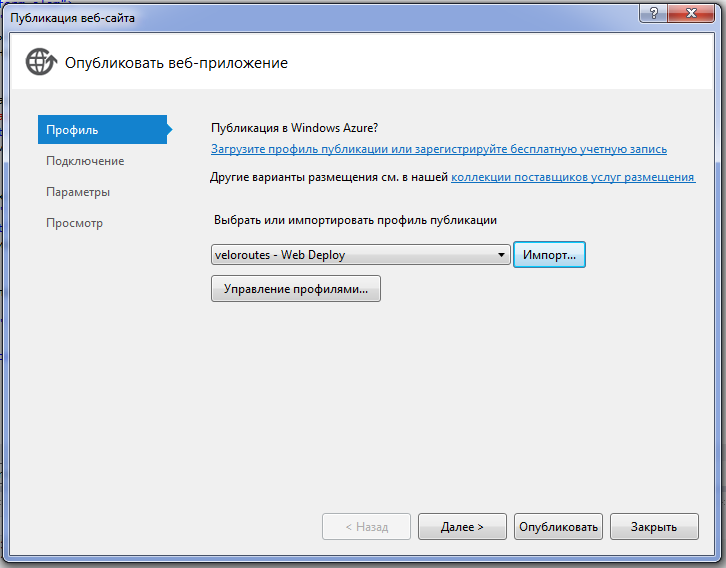


Рисунок 3.8 – Публикация приложения

1. Появится окно, изображенное на рисунке 3.9. Проверяем, жмем далее.

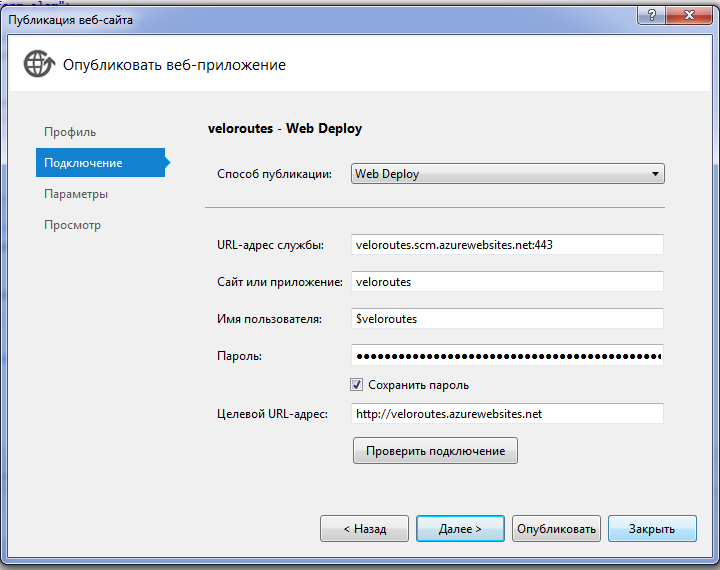


Рисунок 3.9 – Настройки подключения

1. Проверяем строку подключения, жмем опубликовать, ждем (рисунок 3.10). Открывается страница <http://veloroutes.azurewebsites.net/> , по которой теперь доступен проект.

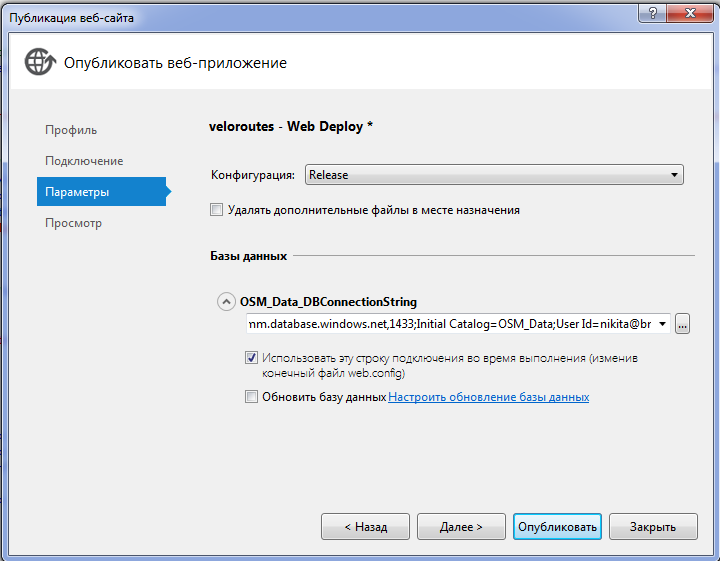


Рисунок 3.10 – Окончание публикации

1. Для работы приложения необходимо подключить обе БД. Сначала подключим Neo4j. Для этого нам необходима виртуальная машина в Windows Azure с сервером СУБД Neo4j. Перейдем на страницу создания такого сервера. В низу страницы нажмем на кнопку создать и увидим диалоговое окно с выбором образа (рисунок 3.11).

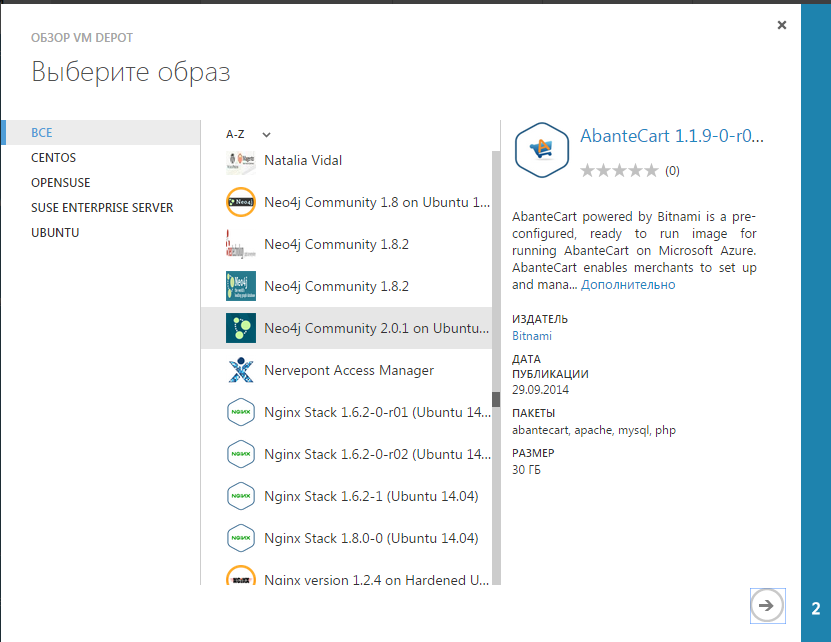


Рисунок 3.11 – Выбор образа

Выбираем Neo4j on Ubuntu и жмем далее.

1. Появляется следующее окно (рисунок 3.12). Выбираем нашу учетную запись. Жмем ОК.

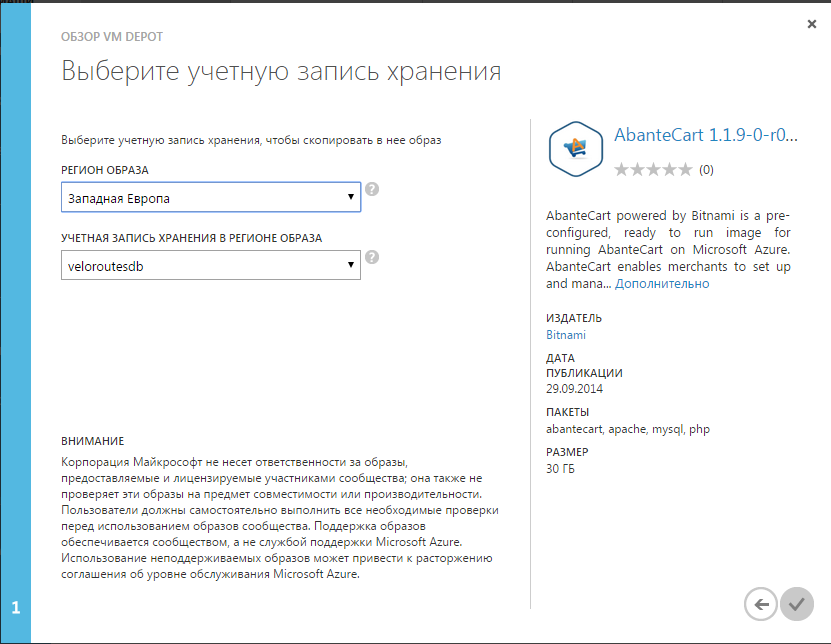


Рисунок 3.12 – Выбор учетной записи хранения

1. Далее создаем виртуальную машину, используя созданный ранее образ (рисунки 3.13 – 3.14).

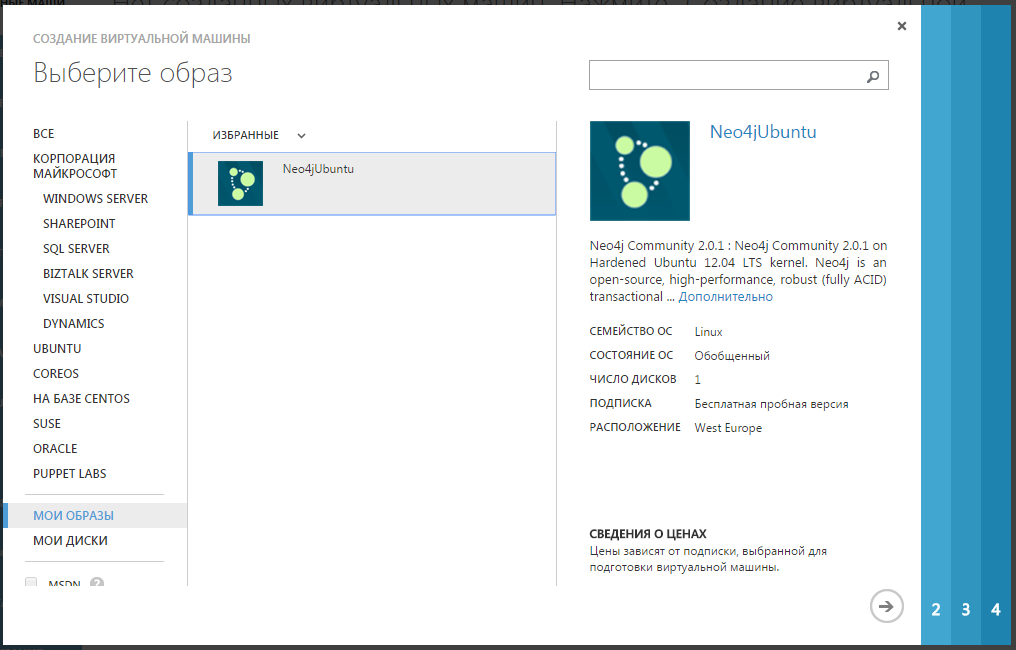


Рисунок 3.13 – Первый этап создания виртуальной машины

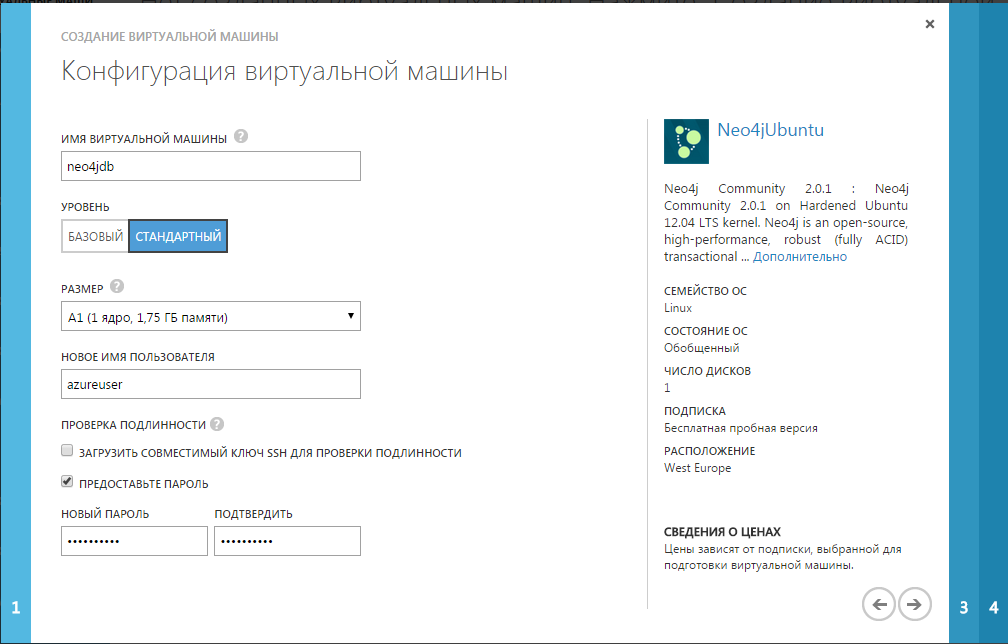


Рисунок 3.14 – Второй этап

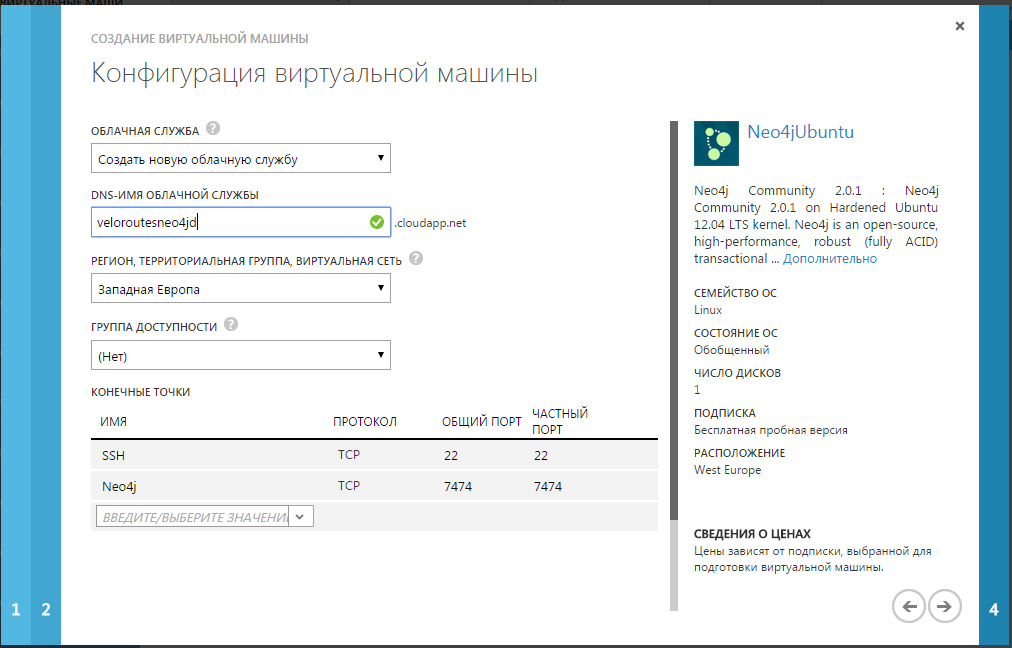


Рисунок 3.15 – Третий этап

На последнем этапе ничего не меняем и жмем ОК. Теперь интерфейс для работы с БД доступен по адресу <http://veloroutesneo4jdb.cloudapp.net:7474/browser/>.

1. Теперь необходимо развернуть бузу данных MS SQL Server с информацией о карте. Для этого необходимо соединиться с созданным ранее сервером в Windows Azure через Microsoft Management Studio. Далее нужно локальную базу данных развернуть на этом сервере. Для этого нужно правой кнопкой мыши нажать на подходящую базу данных, выбрать Задачи – Развертывание базы данных в SQL Azure. Выбираем нужный сервер Windows Azure и ждем.
2. Запускаем приложение с раскомментированной строчкой создания графовой базы данных. Ждем пока граф загрузится. После этого удаляем эту строчку, заново опубликовываем проект, переходим по ссылке <http://veloroutes.azurewebsites.net/>. Теперь приложение полностью работает.
   1. **Интерфейс пользователя**
      1. **Начало работы**

Пользовательский интерфейс разработанного ПО представлен на рисунке 3.16.



Рисунок 3.16 – Интерфейс разработанного ПО

Интерфейс приложения по сути состоит из одной страницы через, которую происходит вся работа с приложением. В левой части страницы находится форма, через которую происходит взаимодействие пользователя и приложения. Под формой расположена информация о построенном маршруте. В правой части находится карта района построения маршрутов, на которой отображается построенный путь. Рассмотрим поближе форму взаимодействия пользователя с приложением. Форма представлена на рисунке 3.17.

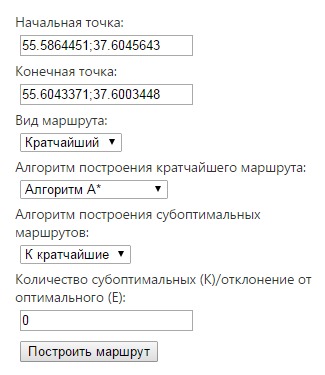


Рисунок 3.17 – Форма взаимодействия пользователя и приложения

Информация о начальной и конечной точках в полях форме появляется при клике мышкой на карте. Первый клик – начальная точка, второй клик – конечная точка. При этом поля начальной и конечной точек недоступны для ввода во избежание ввода неверных данных. После этого пользователю необходимо выбрать вид маршрута, алгоритмы поиска пути и значение коэффициента K или E в зависимости от выбранного алгоритма построения альтернативных маршрутов. Причем, выбор алгоритма поиска субоптимальных путей важен только в случае безопасного и спортивного вида маршрута. При нажатии на кнопку “Построить маршрут” на сервер посылается Ajax-запрос, который содержит всю информацию из формы. На основе этой информации строится маршрут, который по завершению построения отображается на карте района. Также после построения маршрута под формой появляется информация о его построении (длина, время построения, мера безопасности, если это необходимо).

* + 1. **Обработка исключительных ситуаций**

В связи с простотой интерфейса может быть 2 вида исключительных ситуаций: неверный ввод коэффициента E/K и невозможность построения маршрута через заданные точки. Обе ситуации корректно обрабатываются программой. Примеры их обработки представлены на рисунках 3.18 и 3.19 соответственно.

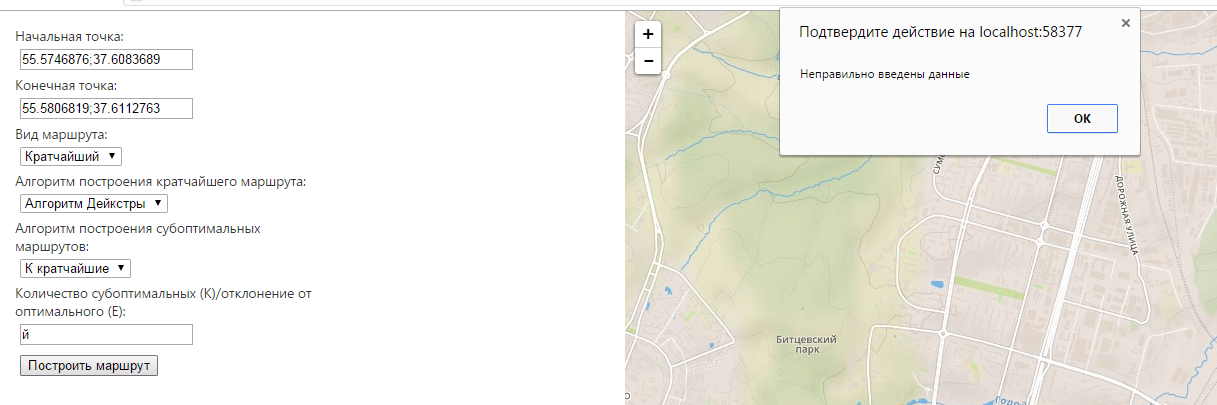


Рисунок 3.18 – Неверный ввод коэффициента K/E

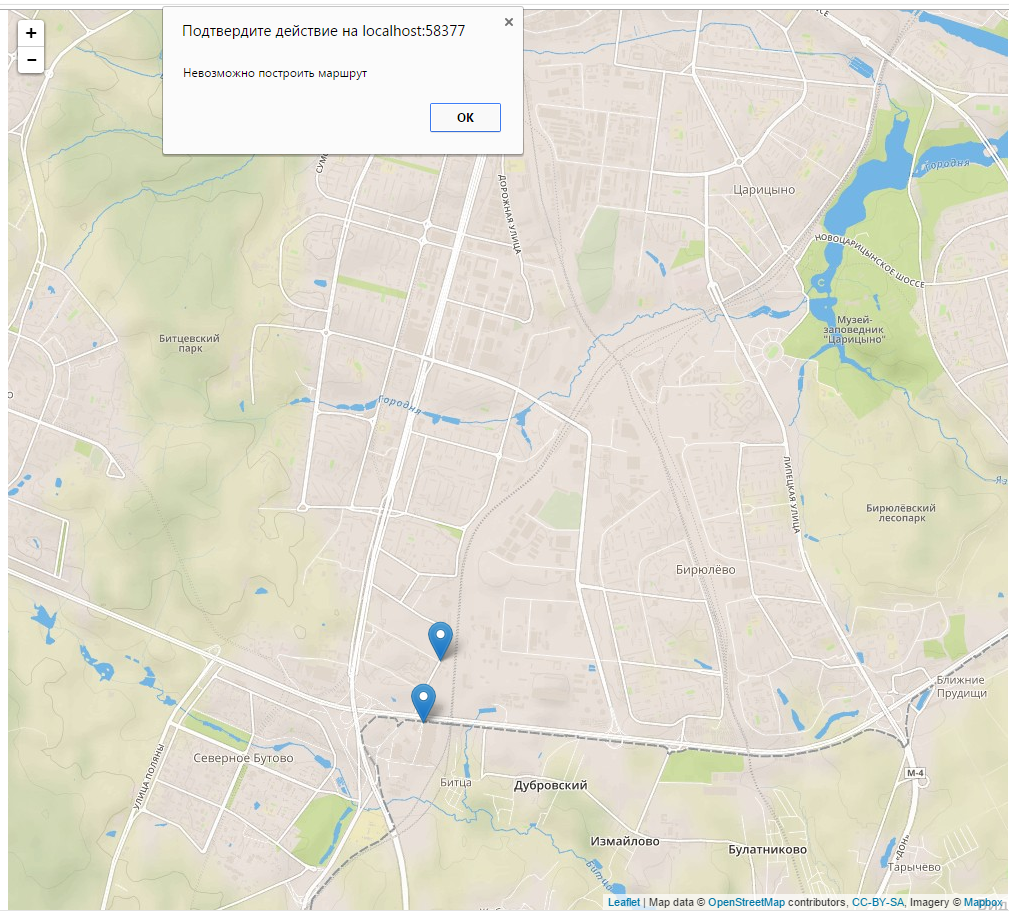


Рисунок 3.19 – Невозможность построения маршрута

* 1. **Системные требования????????????**

Работа данного программного продукта тестировалась со следующими системными требованиями:

* Процессор Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz 2.30GHz;
* Оперативная память 4 ГБ;
* Клавиатура;
* Мышь.
  1. **Тестирование и отладка программы**

В ходе разработки программного обеспечения было произведено и системное тестирование программного обеспечения, которые позволили выявить и своевременно устранить недостатки.

* 1. **Выводы**

В соответствии с описанными в конструкторском разделе алгоритмами, на языке C# с использованием фреймворка ASP .NET MVC было реализовано веб-приложение. Это приложение было развернуто в интернете с помощью Windows Azure по адресу <http://veloroutes.azurewebsites.net/>. Реализованное приложение всегда доступно через сеть Интернет. Произведено тестирование и отладка программного продукта.

1. **Экспериментальный раздел**
2. **Организационно-экономическая часть**
4. 1. **Определение этапов и содержания работ**

Планирование длительности этапов и содержания работ осуществляется в соответствии с ГОСТ 19.102-77.

На стадии разработки технического задания определены стадии и этапы работ, указанные в таблице 5.1.

Таблица 5.1 ‒ Этапы выполнения проекта

|  |  |  |
| --- | --- | --- |
| Стадия разработки | Этап работ | Содержание работ |
| 1. Техническое задание | Обоснование необходимости разработки программы | Постановка задачи.  Сбор исходных материалов. |
| Научно-исследовательские работы | Предварительный выбор методов решения задач. |
| Разработка и утверждение технического задания | Определение требований к программе.  Определение стадий, этапов и сроков разработки программы и документации на неё. |
| 1. Эскизный проект | Разработка эскизного проекта | Предварительная разработка структуры входных и выходных данных.  Уточнение методов решения задачи. |
| 1. Технический проект | Разработка технического проекта | Уточнение структуры входных и выходных данных.  Разработка алгоритма решения задачи.  Определение формы представления входных и выходных данных.  Разработка структуры программы. |
| Продолжение таблицы 5.1 | | |
|  | Утверждение технического проекта | Разработка пояснительной записки.  Согласование и утверждение технического проекта. |
| 1. Рабочий проект | Разработка программы | Разработка и реализация ядра системы.  Разработка и реализация пользовательского интерфейса.  Тестирование и отладка программы. |
| 1. Внедрение | Подготовка и передача программы | Развертывание приложения на сервере.  Опытная эксплуатация.  Обучение пользователей. |

* 1. **Расчет трудоемкости**

Вероятные трудозатраты на выполнение данного проекта определяются с помощью экспертных оценок. Для этой цели опрошено четверо экспертов-разработчиков, которые дали оценку трудовых затрат на данную разработку, указанную в таблице 5.2.

Таблица 5.2 ‒ Результаты экспертных оценок

|  |  |
| --- | --- |
| Эксперт 1 | 1000 часов |
| Эксперт 2 | 1150 часов |
| Эксперт 3 | 950 часов |
| Эксперт 4 | 1300 часов |

Ожидаемая продолжительность:

 часов

Затраты труда на программирование выражаются формулой:

,

где  ‒ время на проектирование системы,

‒ время на написание программы,

 ‒ время на проведение тестирования, исправлений и написание документации.

Коэффициент затрат на проектирование определяется равным = 0.7, так как этап проектирования наиболее важный.

Затраты труда на проведения тестирования, внесения исправлений и подготовки сопроводительной документации определяются суммой затрат на выполнение каждой работы этапа тестирования:

 = ++,

где ‒ затраты труда на проведение тестирования, – затраты труда на внесение исправлений, – затраты труда на написание документации.

Значение определяется посредством ввода соответствующих коэффициентов к значениям затрат труда на непосредственно программирование ():

Коэффициент затрат на проведение тестирования отражает отношение затрат труда на тестирование программы по отношению к затратам труда на ее разработку. В связи с высокими требованиями к надежности разрабатываемой системы коэффициент принимается равным 0.6.

Коэффициент затрат на исправление программы  принимается равным 0.3, как наиболее используемый.

Коэффициент затрат на написание документации , в связи с простотой пользования разрабатываемой программой принимается равным 0.2.

Затраты на выполнение этапа тестирования:

 = (0.6+0.3+0.2) = 1.1

Тогда можно записать:



Отсюда находим затраты труда на написание программы:

 часов

Далее можно рассчитать все затраты труда:

 часов

 часов

 часов

 часов

Заказчик установил, что опытный срок эксплуатации для программного комплекса 8 чел/дней. При 8-ми часовом рабочем дне для одного программиста затраты на внедрение будут составлять 64 часа.

Общее значение трудозатрат тогда:

 часов.

* 1. **Определение численности исполнителей**

Средняя численность исполнителей определяется по формуле:

,

где F – фонд рабочего времени и определяется по формуле:



- это фонд времени в текущем месяце и вычисляется для каждого месяца с учетом выходных и праздничных дней.

На реализацию проекта отведено 4 месяца рабочего времени при односменной работе с продолжительностью рабочего дня 8 часов.

Найдем  для каждого месяца. Результаты приведены в таблице 5.3.

Таблица 5.3 ‒ Фонд времени для каждого месяца

|  |  |
| --- | --- |
| Февраль | (28 – 8 – 1) \* 8 = 152 часа |
| Март | (31 – 7 – 3) \* 8 = 168 часов |
| Апрель | (30 – 8) \* 8 = 176 часов |
| Май | (31 – 6 ­– 7) \* 8 = 144 часа |

Отсюда: F = 152+168+176+144= 640 часов

Отсюда средняя численность исполнителей равна:



В выполнении проекта будут заняты ведущий программист и программист.

Продолжительность отдельных работ определяем по формуле:

,

где - расчетная продолжительность работы

 - количество исполнителей

 - коэффициент выполнения нормы, равен 1.

Результаты расчетов приведены в таблице 5.4.

Таблица 5.4 ‒ Расчетная и итоговая продолжительность каждой задачи

|  |  |  |  |
| --- | --- | --- | --- |
| Задача | Расчетная продолжительность, часы | Количество исполнителей | Итоговая продолжительность, часы |
| Анализ предметной области | 72 | 1 | 72 |
| Выбор средств реализации работы с картой дорог | 32 | 1 | 32 |
| Построение архитектуры | 128 | 2 | 64 |
| Проектирование и наполнение баз данных | 144 | 2 | 72 |
| Реализация построения маршрутов | 112 | 2 | 56 |
| Реализация отображения маршрутов на карте | 96 | 2 | 48 |
| Реализация пользовательского интерфейса | 48 | 2 | 24 |
| Объединение компонентов в единую систему | 32 | 2 | 16 |
| Тестирование и отладка | 304 | 2 | 152 |
| Написание документации | 80 | 2 | 40 |
| Развертывание приложения на сервере | 8 | 1 | 8 |
| Опытная эксплуатация | 80 | 2 | 40 |

* 1. **Сетевая модель проекта**

Для определения временных затрат и трудоемкости разработки программного обеспечения систем используется метод сетевого планирования. Метод сетевого планирования позволяет установить единой схемой связь между всеми работами в виде наглядного и удобного для восприятия изображения (сетевого графика), представляющего собой информационно-динамическую модель, позволяющую определить продолжительность и трудоёмкость, как отдельных этапов, так и всего комплекса работ в целом.

Составление сетевой модели включает в себя оценку степени детализации комплекса работ и определения логической связи между отдельными работами.

С этой целью составляется перечень всех основных работ. В перечне указываются кодовые номера работ, наименования, длительности и номера их предшественников с типом связи (окончание-начало, начало-начало, окончание-окончание).

Для построения сетевой модели проекта используется Microsoft Project 2012. На рисунке 5.1 проиллюстрирован расчет дней и часов.

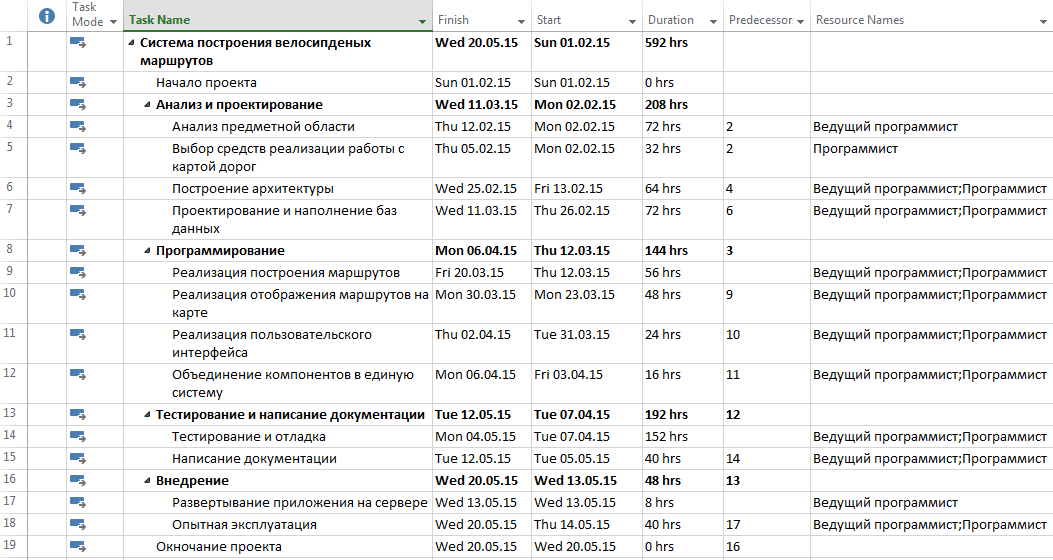


Рисунок 5.1 ‒ Расчет дней и часов

Для того, чтобы получить длительность в днях, необходимо итоговое количество часов разделить на 8: дня

* 1. **Календарный график выполнения проекта**

По сетевой модели, в частности, может быть построена диаграмма Гантта, иллюстрирующая проведение последовательности работ, отражённых в сетевой модели на рисунке 5.1.

На рисунке 5.2 изображена диаграмма Гантта с отслеживанием. Красным выделен критический путь проекта длительностью в 592 часа. Его составляет последовательность задач 4-6-7-9-10-11-12-14-15-17-18.

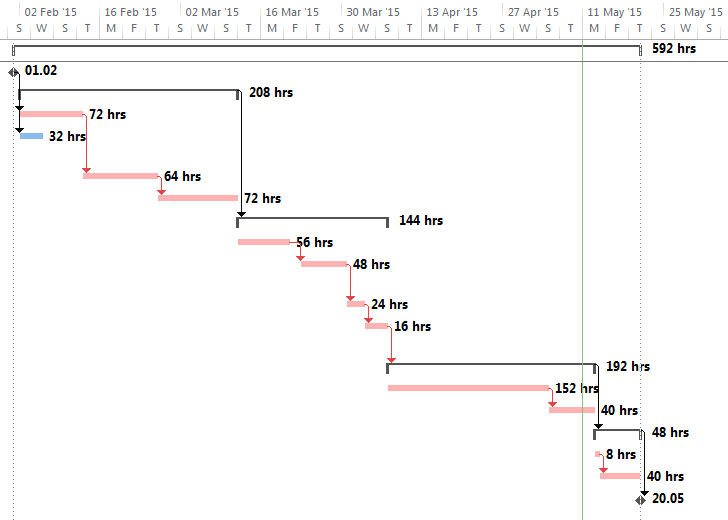


Рисунок 5.2 ‒ Диаграмма Гантта с отслеживанием

Дальнейшая оптимизация критического пути невозможна в виду малого количества разработчиков и жёстко установленной последовательности выполнения работ. В таблице 5.5 отражена занятость каждого из работников в каждой задаче проекта.

Таблица 5.5 ‒ Использование ресурсов

|  |  |  |
| --- | --- | --- |
| Задача | Ведущий программист | Программист |
| Анализ предметной области | 72 часа | 0 часов |
| Выбор средств реализации работы с картой | 0 часов | 32 часов |
| Построение архитектуры | 64 часа | 64 часа |
| Проектирование и наполнение баз данных | 72 часа | 72 часа |
| Реализация построения маршрутов | 56 часов | 56 часов |
| Реализация отображения маршрутов на карте | 48 часов | 48 часов |
| Реализация пользовательского интерфейса | 24 часа | 24 часа |
| Объединение компонентов в единую систему | 16 часов | 16 часов |
| Тестирование и отладка | 152 часа | 152 часов |
| Написание документации | 40 часов | 40 часов |
| Развертывание приложения на сервере | 8 часов | 0 часов |
| Опытная эксплуатация | 40 часов | 40 часов |
| Итого: | 592 часа | 544 часа |

* 1. **Анализ структуры затрат проекта**

Затраты на выполнение проекта состоят из затрат на заработную плату исполнителям, затрат на закупку или аренду оборудования, затрат на организацию рабочих мест, и затрат на накладные расходы (21).

,

где ‒ заработная плата исполнителей, ‒ затраты на обеспечение необходимым оборудованием, ‒ затраты на организацию рабочих мест, ‒ накладные расходы.

* + 1. **Расчет расходов**

Затраты на выплату исполнителям заработной платы определяется следующим соотношением:



где ‒ основная заработная плата, ‒ дополнительная заработная плата, ‒ отчисления с заработной платы.

Расчет основной заработной платы при дневной оплате труда исполнителей следует проводить на основе данных по окладам и графику занятости исполнителей:



где ‒ число дней, отработанных исполнителем проекта, дневной оклад исполнителя. При 8-и часовом рабочем дне он рассчитывается по следующей формуле:



где - месячный оклад,  - месячный фонд рабочего времени

По данным кадровых агентств и Интернет-ресурса https://rabota.yandex.ru средний месячный оклад специалистов требуемой квалификации следующий:

* ведущий программист – 80000 руб.
* программист – 75000 руб.

Указанная зарплата не является «чистой» и из неё ещё будет вычтена сумма подоходного налога. Дополнительная заработная плата не предусматривается в связи с тем, что проект короткий и исполнители нанимаются на срок менее полугода.

MS Project позволяет автоматизировать расчёт затрат на ресурсы: как трудовые, так и материальные (рисунок 5.3). Расходы на заработную плату составляют 551000р.

Отчисления на заработную плату в виде выплаты единого социального налога в 30% составляют 165300р.

* + 1. **Расчет расходов, связанных с обеспечением работ оборудованием**

Для работ, связанных с выполнением проекта необходимы два компьютера с процессором Intel Core i5. Возможна закупка двух моноблоков Lenovo IdeaCentre B50-30 по 48900р. на общую сумму 97800 руб.

* + 1. **Расчет расходов, связанных с организацией рабочих мест**

В соответствии с санитарными нормами, расстояние между рабочими столами с видеомониторами должно быть не менее 2 м., а между боковыми поверхностями видеомониторов - не менее 1,2 м. Площадь на одно рабочее место с терминалом или ПК должна составлять не менее 6 кв.м., а объем - не менее 20 куб.м. Расположение рабочих мест в подвальных помещениях не допускается. Помещения должны быть оборудованы системами отопления, кондиционирования воздуха или эффективной приточно-вытяжной вентиляцией.

Для организации рабочих мест на два компьютера потребуется рабочая площадь 16 кв.м.

Стоимость годовой аренды помещений требуемого класса в различных районах г. Москвы приведена в таблице 5.6 (источник irr.ru).

Таблица 5.6 – Стоимость аренды помещений

|  |  |  |
| --- | --- | --- |
| Район | Площадь | Стоимость аренды (за год за кв.м) |
| м. Кузнецкий мост | 18 | 34800 |
| м. Полянка | 15 | 23000 |
| м. Полежаевская | 16 | 13000 |
| м. Третьяковская | 17 | 15000 |
| м. Партизанская | 16 | 8400 |

Аренда самого дешевого офиса требуемого класса обходится в 8400 рублей за квадратный метр в районе станции метро Партизанская. Затраты на аренду помещения на этапе разработки вычисляются по формуле:



где - стоимость аренды одного квадратного метра за год;

*S* - арендуемая площадь офиса;

- срок аренды.

Офис не будет использоваться на этапе внедрения, тогда учитывая, что написание документации заканчивается 13.05.15, офис необходимо арендовать на 3,5 месяца:



* + 1. **Расчет накладных расходов**

Накладные расходы вычисляют в расчете 60%-100% от расходов на основную заработную плату и в данном проекте планируются 60%:



Накладные расходы на разработку программы, рассчитанные по формуле, составляют: .

На рисунке 5.3 показаны все ресурсы, требуемые для разработки проекта.

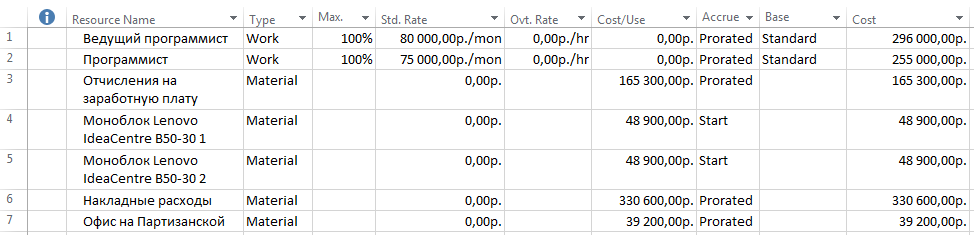


Рисунок 5.3 ‒ Ресурсы проекта

Суммарные затраты на выполнение проекта при одном внедрении:

рублей

Они показаны на рисунке 5.4.

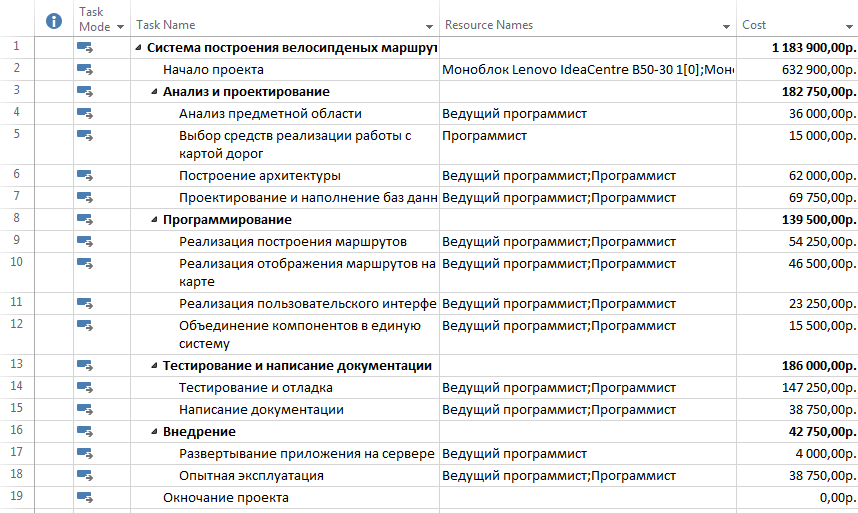


Рисунок 5.4 ‒ Общие затраты на выполнение проекта

Затраты только на разработку (без внедрения) равны:

рублей

На рисунке 5.5 представлена структура затрат на разработку.

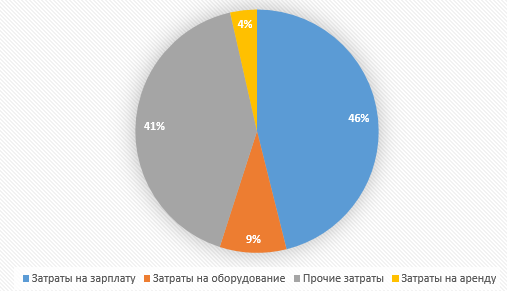


Рисунок 5.5 ‒ Затраты на разработку

* 1. **Исследование рынка для разрабатываемого ПО**

В настоящее время существует множество ресурсов и приложений, позволяющих строить маршруты, но все они, в основном, работают только для автомобилей и не могут проложить путь для велосипедиста. Учитывая рост популярности использования велосипеда в качестве транспортного средства, то такие компании как Google, Яндекс, карты которых являются самыми посещаемыми, но не поддерживают построение велосипедных маршрутов, могут быть заинтересованы в данной разработке. Внедрение такого сервиса обеспечит более частое посещение их online-карт. Существует еще несколько сервисов, которые за счёт внедрения построения велосипедных маршрутов, могли бы привлечь на больше посетителей.

Существует очень мало разработок в данной области, поэтому конкуренция будет минимальной.

Исходя из вышеизложенного, можно предположить, что число потенциальных покупателей на годовом интервале времени будет не более 3, т.е. .

## **Планирование цены и прогнозирование прибыли**

На основе данных о затратах на разработку и внедрение, результатах прогнозирования объема продаж, определим стоимость одного комплекта программного обеспечения.

Стоимость поставляемого на рынок ПО определяется частью стоимости разработки ПО, затрат на внедрение и прибыли фирмы-разработчика.

Стоимость ПО можно рассчитать, используя следующее соотношение:



где ‒ часть стоимости разработки, приходящаяся на одну копию программы, ‒ стоимость внедрения программы, ‒ процент прибыли, заложенный в стоимость.

Стоимость внедрения: = 42750+25650+12825=81225р.

Стоимость внедрения остается постоянной для каждой установки ПО, а частичная стоимость разработки, приходящаяся на каждый комплект ПО, определяется исходя из данных о планируемом объеме установок:



где ‒ стоимость разработки проекта (без внедрения),

‒ планируемое число копий ПО,

‒ ставка банковского процента по долгосрочным кредитам.

Если, в качестве ставки процента по долгосрочным кредитам использовать 25% годовых и использовать рассчитанные значения затрат на разработку и планируемое число установок, то можно определить частичную стоимость разработки.

1102675 / 3 \* (1 + 0,25) = 459448 рублей

* = (459448 +81225)\*(1+0.15) = 621774 рубля

Сумма 1102675 р., необходимая для разработки ПО, берётся в кредит в «Банке Москвы» под ставку 25% годовых на срок 6 месяцев. В таблице 5.7 представлен фрагмент таблицы общего баланса. В ней учтен налог на прибыль, который составляет 20%.

Таблица 5.7 ‒ Фрагмент таблицы общего баланса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Период расчета | Баланс начальный | Сумма продаж | Сумма погашения кредита | Фактическая совокупная прибыль | Фактическая чистая  прибыль | Баланс конечный |
| 2.2015 | -307564 | 0 | 229724 | -229724 | -229724 | -394693 |
| 3.2015 | -604457 | 0 | 229724 | -229724 | -229724 | -691586 |
| 4.2015 | -901350 | 0 | 229724 | -229724 | -229724 | -988479 |
| 5.2015 | -1059112 | 0 | 229724 | -229724 | -229724 | -1102675 |
| 6-7.2015 | -1183900 | 621774 | 459448 | 162326 | 129860,8 | -594591,2 |
| 8-9.2015 | -675816,2 | 621774 | 0 | 621774 | 497419,2 | -178397 |
| 10-11.015 | -259522 | 621774 | 0 | 621774 | 497419,2 | 237897,2 |

## **Выводы**

В результате выполнения организационно-экономических расчетов установлены следующие показатели данного проекта:

Общие затраты труда на реализацию проекта, которые составили 1156 чел/часов.

Исходя из временных требований к реализации проекта, численность исполнителей составила 2 человека: ведущий программист и программист.

Составлена сетевая модель с контролем затрат на каждом этапе и построена диаграмма Гантта – временной график выполнения проекта.

Рассчитаны денежные затраты на реализацию проекта, которые составили в общей сумме 1 102 675 р. Источник денежных средств – кредит в «Банке Москвы» на срок 6 месяцев под 25% годовых.

Стоимость продукта составила 621 774 рублей при условии продажи трех экземпляров в год.

Ориентировочный срок окупаемости продукта – 6 месяцев.

На основании вышеизложенного можно сделать вывод об экономической целесообразности проведения работ и внедрения в производство данной разработки.

1. **Охрана труда и экология**
   1. **Анализ опасных и вредных факторов при разработке программного обеспечения и мероприятия по их устранению**

Разработка программного обеспечения требует длительного взаимодействия с вычислительными системами. Работа с персональными электронно-вычислительными машинами связана с рядом вредных и опасных факторов, таких как статическое электричество, рентгеновское излучение, электромагнитные поля, блики и отраженный свет, ультрафиолетовое излучение, мерцание изображения. При длительном воздействии на организм эти факторы негативно влияют на здоровье человека.

* + 1. **Микроклимат**

Работа как программиста, так и пользователя относится к категории 1а, поскольку не предполагает больших физических усилий. Поэтому оптимальные нормы микроклимата для рабочего помещения программиста определяются таблицей 6.1 (СанПиН 2.2.2/2.4.1340-03):

Таблица 6.1 ‒ Оптимальные нормы микроклимата

|  |  |  |  |
| --- | --- | --- | --- |
|  | Температура воздуха, oC | Относительная влажность воздуха, % | Скорость движения воздуха, м/с |
| Холодный | 22-24 | 40-60 | 0,1 |
| Теплый | 23-25 | 40-60 | 0,1 |

Вредным фактором при работе с ЭВМ является также запыленность помещения. Этот фактор усугубляется влиянием на частицы пыли электростатических полей персональных компьютеров.

Для устранения несоответствия параметров указанным нормам проектом предусмотрено использование системы кондиционирования как наиболее эффективного и автоматически функционирующего средства.

Нормы СанПиН 2.2.4.1294-03 «Санитарно-гигиенические нормы допустимых уровней ионизации воздуха» определяют уровни положительных и отрицательных ионов в воздухе, которые указаны в таблице 6.2.

Таблица 6.2 ‒ Уровни ионизации воздуха помещений при работе на ВДТ и ПЭВМ

|  |  |  |
| --- | --- | --- |
| Уровни | Число ионов в 1 см куб. воздуха | |
| n+ | n- |
| Минимально необходимые | 400 | 600 |
| Оптимальные | 1500-3000 | 3000-5000 |
| Предельно допустимые | 50000 | 50000 |

Для обеспечения требуемых уровней предусмотрено использование системы ионизации Сапфир-4А.

Концентрация вредных химических веществ в помещениях с ПЭВМ не должна превышать «ПДК загрязняющих веществ в атмосферном воздухе населенных мест» ГН 2.1.6.789-99. Для выполнения указанных требований предусмотрено применение фильтров из активированного угля.

* + 1. **Шум и вибрации**

Уровень шума на рабочем месте программиста не должен превышать 55 дБА, а уровень вибрации не должен превышать допустимых норм вибрации. СанПиН 2.2.2.542-96 устанавливает следующие нормы на вибрацию (таблица 6.3).

Таблица 6.3 ‒ Допустимые нормы вибрации на рабочих местах с ВДТ и ПЭВМ

|  |  |  |
| --- | --- | --- |
| Среднегеометрические частоты октавных полос, Гц | Допустимые значения | |
| по виброскорости | |
| м/с | дБ |
| 2 | 4,5x10 | 79 |
| 4 | 2,2x10 | 73 |
| 8 | 1,1x10 | 67 |
| 16 | 1,1x10 | 67 |
| 31,5 | 1,1x10 | 67 |
| 63 | 1,1x10 | 67 |
| Корректированные значения и их уровни в дБ | 2,0x10 | 72 |

При разработке программного обеспечения внутренними источниками шума являются вентиляторы, а также принтеры и другие периферийные устройства ЭВМ.

Внешние источники шума – прежде всего, шум с улицы и из соседних помещений. Постоянные внешние источники шума, превышающего нормы, отсутствуют.

Для устранения превышения нормы проектом предусмотрено применение звукопоглощающих материалов для облицовки стен и потолка помещения, в котором осуществляется работа с вычислительной техникой.

* + 1. **Освещение**

Наиболее важным условием эффективной работы программистов и пользователей является соблюдение оптимальных параметров системы освещения в рабочих помещениях.

Естественное освещение осуществляется через светопроемы, ориентированные в основном на север и северо-восток (для исключения попадания прямых солнечных лучей на экраны компьютеров) и обеспечивает коэффициент естественной освещенности (КЕО) не ниже 1,5%.

В качестве искусственного освещения проектом предусмотрено использование системы общего равномерного освещения. В соответствии с СанПиН 2.2.2/2.4.1340-03, освещенность на поверхности рабочего стола находится в пределах 300-500 лк. Разрешается использование светильников местного освещения для работы с документами (при этом светильники не должны создавать блики на поверхности экрана).

Правильное расположение рабочих мест относительно источников освещения, отсутствие зеркальных поверхностей и использование матовых материалов ограничивает прямую (от источников освещения) и отраженную (от рабочих поверхностей) блескость. При этом яркость светящихся поверхностей не превышает 200 кд/кв.м, яркость бликов на экране ПЭВМ не превышает 40 кд/кв.м, и яркость потолка не превышает 200 кд/кв.м.

В соответствии с СанПиН 2.2.2/2.4.1340-03 проектом предусмотрено использование люминесцентных ламп типа ЛБ в качестве источников света при искусственном освещении. В светильниках местного освещения допускается применение ламп накаливания.

Применение газоразрядных ламп в светильниках общего и местного освещения обеспечивает коэффициент пульсации не более 5%.

Таким образом, проектом обеспечиваются оптимальные условия освещения рабочего помещения.

* + 1. **Рентгеновское излучение**

В соответствии с СанПиН 2.2.2/2.4.1340-03, проектом предусмотрено использование ПЭВМ, конструкция которых обеспечивает мощность экспозиционной дозы рентгеновского излучения в любой точке на расстоянии 0,05 м. от экрана и корпуса монитора не более 0,1 мбэр/час (100 мкР/час). Результаты сравнения норм излучения приведены в таблице 6.4.

Таблица 6.4 ‒ Сравнение норм рентгеновского излучения в различных стандартах

|  |  |
| --- | --- |
| Стандарт | Допустимоезначение, мкР/час, не более |
| СанПиН 2.2.2/2.4.1340-03 | 100 |
| TCO-99 | 500 |
| MPR II | 500 |

Как видно из таблицы, стандарты MPR II и TCO-99 предъявляют менее жесткие требования к рентгеновскому излучению, чем СанПиН. Но при соблюдении оптимального расстояния между пользователем и монитором дозы рентгеновского излучения не опасны для большинства людей.

* + 1. **Неионизирующие электромагнитные излучения**

В соответствии с СанПиН 2.2.2/2.4.1340-03, допустимые значения параметров неионизирующих излучений приводятся в таблицах 6.5, 6.6, 6.7 и 6.8.

Таблица 6.5 ‒ Предельно допустимые значения напряженности электрического поля

|  |  |
| --- | --- |
| Диапазон частот | Допустимые значения |
| 5 Гц – 2 кГц | 25 В/м |
| 2 – 400 кГц | 2,5 В/м |

Таблица 6.6 ‒ Предельно допустимые значения плотности магнитного потока

|  |  |
| --- | --- |
| Диапазон частот | Допустимые значения |
| 5 Гц – 2 кГц | 250 нТл |
| 2 – 400 кГц | 25 нТл |

Величина поверхностного электростатического потенциала не должна превышать 500 В.

Мониторы, используемые в настоящее время, удовлетворяют нормам MPR II (или более жестким требованиям) и имеют следующие предельные значения:

Таблица 6.7 ‒ Предельно допустимые значения напряженности электромагнитного поля

|  |  |
| --- | --- |
| Диапазон частот | Допустимые значения |
| 5 Гц – 2 кГц | 25 В/м |
| 2 – 400 кГц | 2,5 В/м |

Таблица 6.8 ‒ Предельно допустимые значения магнитной индукции

|  |  |
| --- | --- |
| Диапазон частот | Допустимые значения |
| 5 Гц – 2 кГц | 200 нТл |
| 2 – 400 кГц | 25 нТл |

Поверхностный электростатический потенциал не превышает 500 В.

Таким образом, параметры электрических и магнитных (неионизирующих) полей удовлетворяют требованиям СанПиН.

* + 1. **Визуальные параметры**

Неправильный выбор визуальных эргономических параметров приводит к ухудшению здоровья пользователей, быстрой утомляемости, раздражительности. В этой связи, проектом предусмотрено, что конструкция вычислительной системы и ее эргономические параметры обеспечивают комфортное и надежное считывание информации.

Требования к визуальным параметрам, их внешнему виду, дизайну, возможности настройки представлены в СанПиН 2.2.2/2.4.1340-03. Визуальные эргономические параметры монитора и пределы их изменений приведены в таблице 6.9.

Таблица 6.9 ‒ Визуальные эргономические параметры ВДТ и пределы их изменений

|  |  |  |
| --- | --- | --- |
| Наименование параметров | Пределы значений параметров | |
| миним. (не менее) | максим. (не более) |
| Яркость знака (яркость фона), кд/кв.м (измеренная в темноте) | 35 | 120 |
| Внешняя освещенность экрана, лк | 100 | 250 |
| Угловой размер знака, угл. мин. | 16 | 60 |

Для выполнения этих требований проектом предусмотрено использование современных мониторов, имеющих достаточно широкий набор регулируемых параметров. В частности, для удобного считывания информации реализована возможность настройки положения монитора по горизонтали и вертикали. Мониторы оснащены специальными устройствами и средствами настройки ширины, высоты, яркости, контраста и разрешения изображения. Кроме того, в современных мониторах зерно изображения имеет размер в пределах 0,27 мм, что обеспечивает высокую четкость и непрерывность изображения. Наконец, на поверхность дисплея нанесено матовое покрытие, чтобы избавиться от солнечных бликов.

* 1. **Акустический расчет**

В помещении размером 15x30x3 м3находится 1 источник шума - окно на улицу. Схема его расположения представлена на рисунке 6.1.

В

А

R2

R1

Рисунок 6.1 – Схема расположения источников шума

Расстояние до расчетной точки равно R1 = 13 м. Окно находится на стене выше уровня пола. Уровни звуковой мощности источников шума представлены в таблице 6.10.

Таблица 6.10 – Уровни звуковой мощности источников шума

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Источник шума | Lp=f(fсг), дБ | | | | | | | |
| 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |
| 1 | 72 | 72 | 68 | 68 | 68 | 68 | 71 | 70 |

УЗД (уровни звукового давления) в расчетной точке при заданных уровнях звуковой мощности нескольких источников шума на каждой (*j*-той) из восьми октавных полос определяем по следующей формуле:

где - октавный уровень звуковой мощности *i*-го источника, дБ;

- коэффициент, учитывающий влияние ближнего поля в тех случаях, когда расстояние *r* меньше удвоенного максимального габарита источника (*r* < 2*l*макс). Так как максимальный габарит любого источника много меньше расстояния до расчетной точки, то по таблице принимаем ;

- фактор направленности источника шума, безразмерный, определяется по опытным данным. Для источников шума с равномерным излучением звука следует принимать Ф*i* = 1;

Ω*i* - пространственный угол излучения источника, рад. По таблице 3 из СНиП 23-03-2003 принимаем Ω1 = 2π (источник подвешен у стены), Ω2 = (источник у потолка близко от двух стен);

*ri* - расстояние от акустического центра источника шума до расчетной точки, м;

*m* - число источников шума, находящихся на расстоянии *ri* ≤ 5*r*мин, где *r*мин - расстояние от расчетной точки до акустического центра ближайшего источника шума. Так как *R*1 *= 13 м,,* то *r*мин = *R*2 = 13 м, то 5*r*мин = 5 ∙ 13 = 65 м, следовательно, источник находятся в указанной зоне, и *m* = 1;

*n* - общее число источников шума в помещении;

*В –* акустическая постоянная помещения в м2, определяемая по формуле:

*В = В*1000*mj*,

Где *В*1000 - постоянная помещения в м2 на среднегеометрической частоте 1000 Гц, определяемая по [таблице 3](http://snipov.net/c_4626_snip_95877.html#i261787) из СНиП II-12–77 в зависимости от объема *V*в м3 и типа помещения. Так как объем помещения *V*= *LBH* = 15 \* 30 \* 3 = 1350 м3и помещение типа 1, то *В*1000 = V / 20 = 1350 / 20 = 67,5 м2;

*mj* - частотный множитель, определяемый по [таблице](http://snipov.net/c_4626_snip_95877.html#i261787) 4 СНиП II-12–77, его значения указаны в таблице 6.11.

Таблица 6.11 – Значения частотного множителя *mj*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Частотный множитель *mj* из среднегеометрических частотах октавных полос в Гц | | | | | | | |
| Объем помещения *V* | 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |
| 1350 м3 | 0,5 | 0,5 | 0,55 | 0,7 | 1 | 1,6 | 3 | 6 |

Отсюда получаем значения B, указанные в таблице 6.12.

Таблица 6.12 – Значения акустической постоянной помещения B

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Параметр | Среднегеометрические частоты октавных полос в Гц | | | | | | | |
| 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |
| *В*, м2 | 33,75 | 33,75 | 37,125 | 47,25 | 67,5 | 108 | 202,5 | 405 |

*k* - коэффициент, учитывающий нарушение диффузности звукового поля в помещении, обратно пропорциональный принимаемому значению по графику на [рисунке 3](http://snipov.net/c_4626_snip_95877.html#i236007) из СНиП II-12–77 в зависимости от отношения акустической постоянной помещения *В* к площади ограждающих поверхностей *S*огр (стен, пола, потолка) помещения (k = 1 / ψ), которая определяется по формуле:

*S*огр = 2C(A+ B) + 2AB= 2 ∙ 3 ∙ (15 + 30) + 2 ∙ 15 ∙ 30 = 1170 м2

Значения коэффициента k указаны в таблице 6.13.

Таблица 6.13 – Значения коэффициента нарушения диффузности звукового поля

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Среднегеометрические частоты октавных полос в Гц | | | | | | | |
| 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |
| *В/S*огр | 0,03 | 0,03 | 0,03 | 0,04 | 0,06 | 0,09 | 0,17 | 0,35 |
| *k* | 1 | 1 | 1 | 1,03 | 1,08 | 1,14 | 1,19 | 1,3 |

Рассчитываем УЗД в расчетной точке при заданных уровнях звуковой мощности источников на каждой (*j*-той) из восьми октавных полос *Lj*, дБ, и сравниваем с нормативным значением предельно допустимых уровней звукового давления в помещении для работ, связанных с программированием *Lj*пред, дБ. Для определения нормативного УЗД в октавных полосах были выбраны следующие вид деятельности и рабочее место:

Вид деятельности - Творческая деятельность, руководящая работа с повышенными требованиями, научная деятельность, конструирование и проектирование, программирование, преподавание и обучение, врачебная деятельность.

Рабочие места в помещениях дирекции, проектно-конструкторских бюро, расчетчиков, программистов вычислительных машин, в лабораториях для теоретических работ и обработки данных, приема больных в здравпунктах.

Результаты расчетов представлены в таблице 6.14.

Таблица 6.14 – Результаты расчетов УЗД

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| УЗД | Среднегеометрические частоты октавных полос в Гц | | | | | | | |
| 63 | 125 | 250 | 500 | 1000 | 2000 | 4000 | 8000 |
| *L1j*, дБ | 62,7722 | 62,7722 | 58,3617 | 57,3246 | 55,7960 | 53,7954 | 54,1587 | 50,3417 |
| *Ljобщ*, дБ | 62,7722 | 62,7722 | 58,3617 | 57,3246 | 55,7960 | 53,7954 | 54,1587 | 50,3417 |
| *Lj*пред, дБ | 71 | 61 | 54 | 49 | 45 | 42 | 40 | 38 |

На рисунке 6.2 показано сравнение полученных результатов с предельно допустимыми.

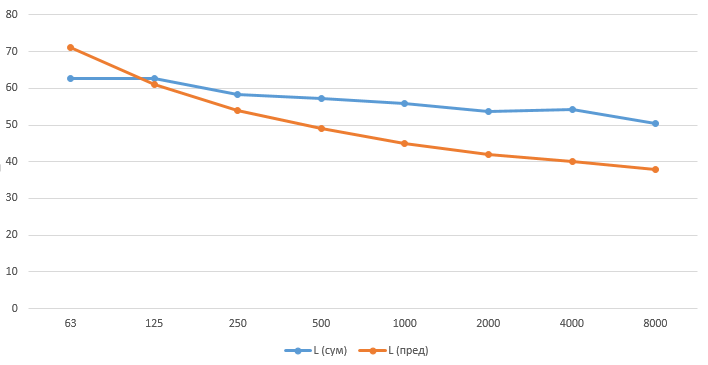


Рисунок 6.2 – Сравнение полученных УЗД с предельно допустимыми

По результатам вычислений в октавных полосах со среднегеометрическими частотами от 125Гц уровень звукового давления от 1 источника шума становится выше предельно допустимых норм.

В таком случае необходимо предпринять меры по защите от излишне высокого звукового давления. Для снижения шума можно применить различные методы коллективной защиты: рациональное размещение оборудования; борьба с шумом на путях его распространения, в том числе изменение направленности излучения шума, использование средств звукоизоляции, звукопоглощение и установка глушителей шума, в том числе акустическая обработка поверхностей помещения.

Например, для уменьшения механического шума следует своевременно проводить ремонт оборудования, заменять ударные процессы на безударные, шире использовать принудительное смазывание трущихся поверхностей, применять балансировку вращающихся частей. Широкое применение получили методы снижения шума на пути его распространения посредством установки звукоизолирующих и звукопоглощающих преград в виде экранов, перегородок, кожухов, кабин и др.

* 1. **Расчет системы искусственного освещения**

В зависимости от цели расчета при проектировании искусственного освещения приходится решать следующий ряд вопросов:

1. Выбрать или определить типы ламп и светильников. Для освещения предприятий службы быта следует применять газоразрядные лампы. Применение ламп накаливания целесообразно при температуре воздуха ниже 10 оС и падении напряжения в сети более 10% от номинального. Выбор светильника должен производится с учетом его крепления, подвода электроэнергии, защиты от механических повреждений, взрыво- и пожароопасности (открытые, закрытые, пылевлагонепроницаемые, взрывоопасные, взрывозащищенные светильники).
2. Выбрать систему освещения. Наиболее экономичной является система комбинированного освещения, так как она создает наиболее равномерное светораспределение. При комбинированном освещении доля общего освещения в нем не должна быть меньше 10%.
3. Выбрать расположение светильников и определить их количество. Светильники, расположенные симметрично вдоль или поперек помещения, в шахматном порядке, рядами, ромбовидно, обеспечивают равномерное по площади освещение. Локализованное неравномерное размещение светильников производят с учетом местонахождения ПЭВМ, оборудования и т.д.

Экспериментально установлено, что наибольшая равномерность достигается:

* При шахматном расположении, если ;
* При расположении прямоугольником, если ,

где *r* – расстояние между светильниками; *H*p – высота подвеса светильника над рабочей поверхностью:

*H*p = *H* – *h*c – *h*р.м.,

где *H* – высота помещения, м; *h*c – высота подвеса светильника, м; *h*р.м. – высота рабочего места (*h*р.м. = 0,8 м), м.

Оптимальное расстояние от крайнего ряда светильников до стены: .

При отсутствии рабочих поверхностей у стены: .

Для исключения слепящего действия светильников общего освещения должно выполняться правило  м при мощности ламп *P*л ≤ 200 Вт. Необходимое число светильников при расположении квадратом составляет:

,

где *S* – площадь помещения, м2; *r* – длина стороны квадрата, м.

1. Определить нормируемую освещенность рабочего места по минимальному размеру объекта различия, фону, контрасту объекта с фоном в системе освещения.

Для расчета искусственного освещения используют три метода:

* Метод светового потока для общего равномерного освещения горизонтальной рабочей поверхности.
* Точечный метод для любой системы освещения.
* Метод удельной мощности для ориентировочных расчетов общего равномерного освещения.

Световой поток определяется по формуле:

,

где *F*л – световой поток лампы, лк; *Е*н – нормированная освещенность, лк; *S* – площадь освещаемого помещения, м2; K – коэффициент запаса (в соответствии со СНиП 23-05-95 для люминесцентных ламп производственных цехов предприятий службы быта ; для остальных помещений K = 1,5); *Z* – коэффициент минимальной освещенности, равный отношению средней освещенности к минимальной; *N* – число ламп; *η* – коэффициент использования светового потока, равный отношению потока, падающего на рабочую поверхность, к общему потоку ламп.

Коэффициент использования светового потока *η* зависит от к.п.д. светильника, коэффициента отражения потолка (*ρ*п), стен (*ρ*с), величины показателя помещения *i*, учитывающего геометрические параметры помещения, высоту подвеса светильника (*H*p):

,

где *a* и *b* – ширина и длина помещения, м.

При длине рабочего помещения *a* = 30 м, ширине *b* = 15 м и высоте *H* = 3 м, потребуется следующее освещение:

*Е*н = 300 лк

*F*л = 5383 лм – световой поток 1 лампы

Тогда

.

Следовательно, *η* = 0,57.

Число ламп равно , что при использовании светильников с использованием одной лампы потребует использования 78 светильников.

Расстояние между двумя светильниками составит: *r* = 1,4 (3,0 – 0,1 – 0,8) = 2,94 м.

Расстояние от стены до светильников: *r*к = 0,25 ∙ 2,94 = 0,74 м.

Следовательно, светильники следует расположить в 6 рядов по 11 светильников (рисунок 6.1).

1,25

м

0,74м

30

м

15

м

2,94

м

2,5

м

Рисунок 6.3 ‒ Схема освещения помещения

При этом отклонение от расчетного светового потока на (66-78)/78= -15% (допустимым является 20% отклонение).

Таким образом, в проекте используются 66 светильников с высотой подвеса 0,1 м и, соответственно, 66 люминесцентных ламп ЛБ-80 со световым потоком 5383 лм и световой отдачей 65,3 лм/Вт.

Расчёт системы искусственного освещения в программе DIALux:

Была выбрана лампа BASE LED 1200x600, которая показана на рисунке 6.4.



Рисунок 6.4 ‒ Ведомость светильников

В помещении расположены два рабочих места, состоящие из моноблока, клавиатуры, офисного стола с тумбочкой и стула. В углу расположен кондиционер. Также, в помещении находятся стеллаж, шкаф, 2 дивана. В комнате есть окно и дверь. Схема показана на рисунке 6.5.

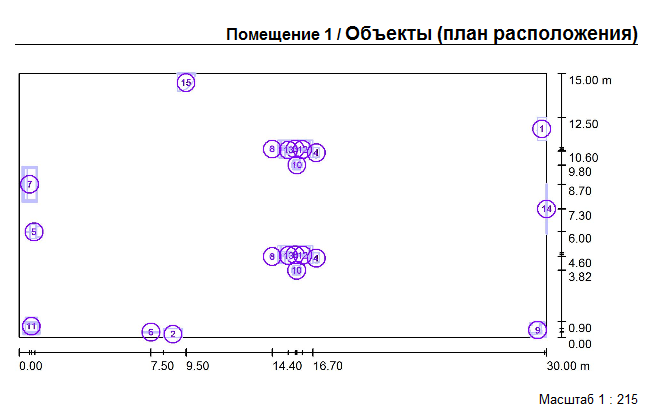


Рисунок 6.5 ‒ Объекты помещения

3D-визуализация помещения представлена на рисунке 6.6.

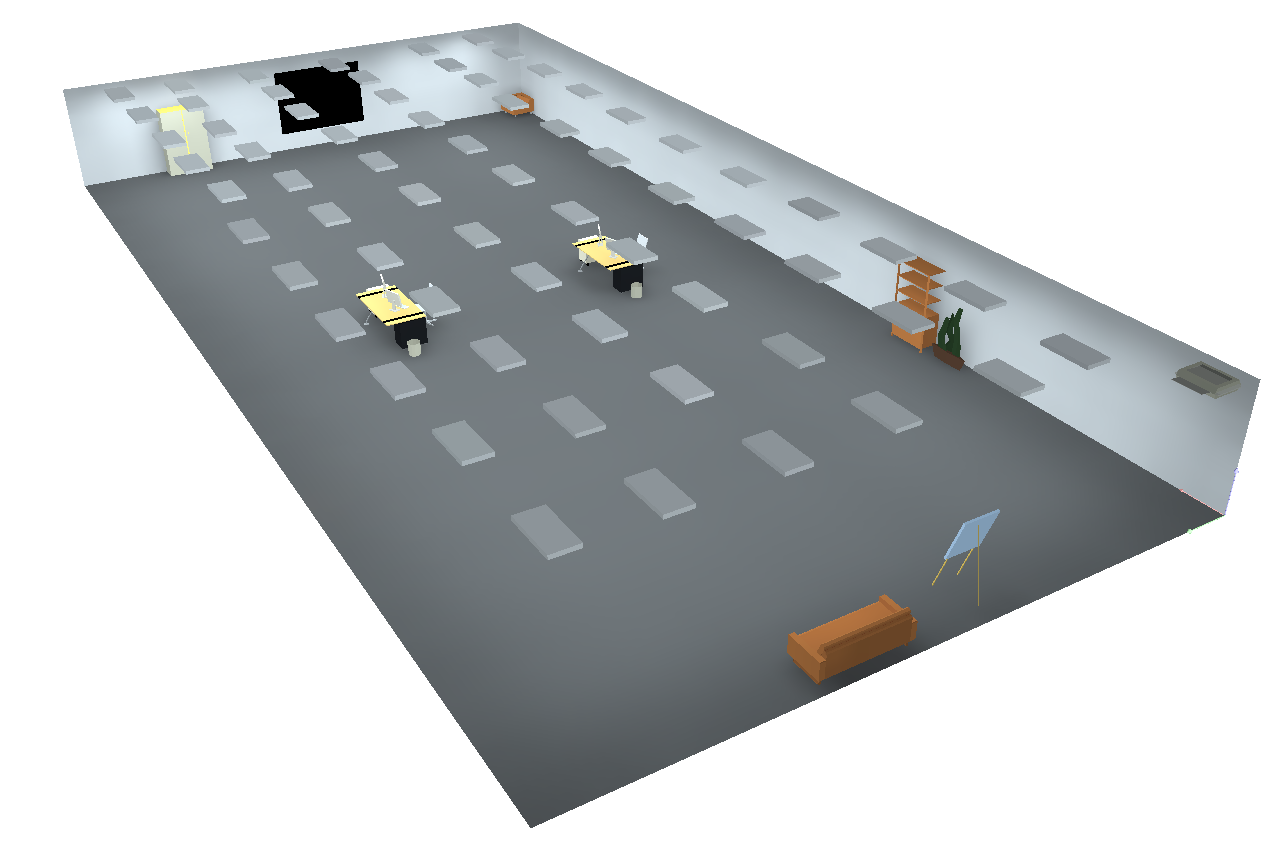


Рисунок 6.6 – 3D-визуализация помещения

Создано две схемы освещения: день (прямой солнечный свет из окна, светильники задействованы на 80%) и ночь (отсутствие света из окна, светильники задействованы на 100%).

На рисунке 6.7 показано резюме схемы освещения «День».

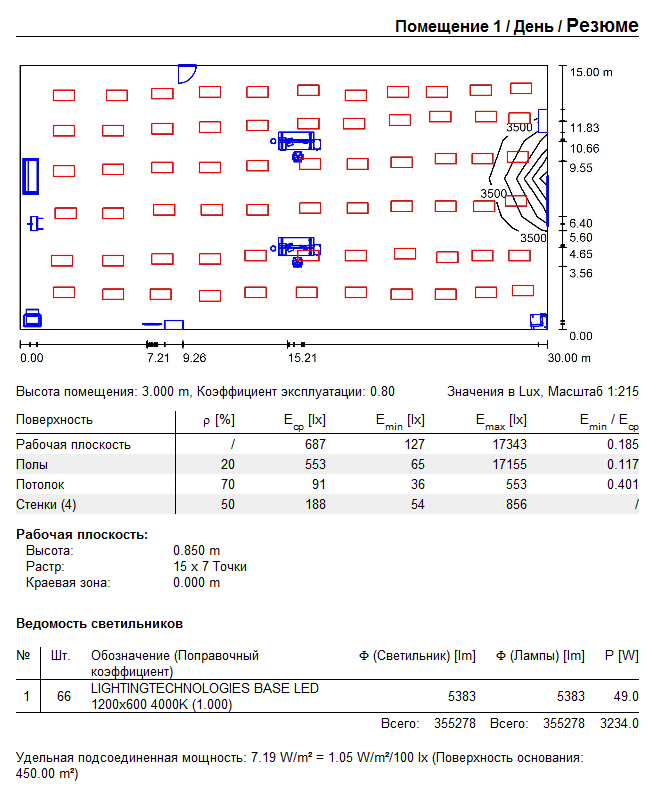


Рисунок 6.7 ‒ Схема освещения «День», резюме

На рисунке 6.8. показан график значений светового потока для рабочей плоскости при схеме освещения «День». На рисунке видно, что превышение допустимых значений (>500 лк) наблюдается только около окна, а в рабочем месте значения светового потока находятся в районе 300 – 450, что по СанПиН 2.2.2/2.4.1340-03 является нормой (300 – 500).

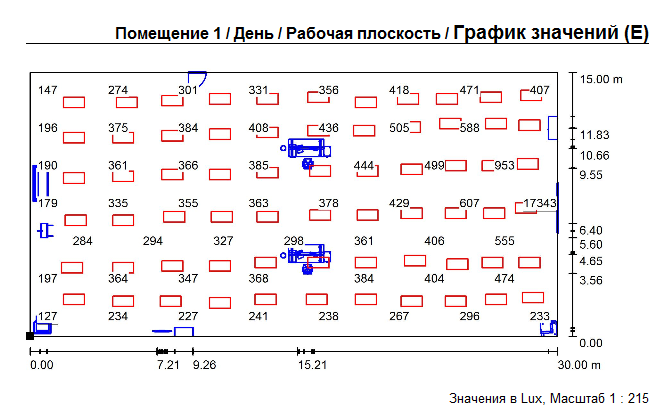


Рисунок 6.8 ‒ График значений светового потока для рабочей плоскости, схема «День»

Далее аналогично рассмотрены параметры схемы освещения «Ночь». На рисунке 6.9 показано резюме схемы освещения «Ночь».

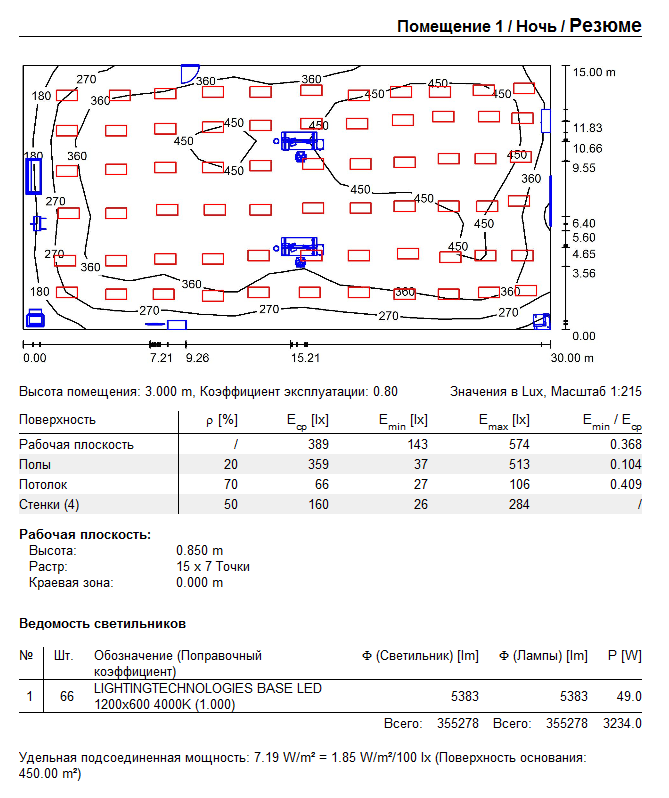


Рисунок 6.9 ‒ Схема освещения «Ночь», резюме

На рисунке 6.10 показан график значений светового потока для рабочей плоскости при схеме освещения «Ночь». Превышения допустимых значений светового потока не наблюдается.

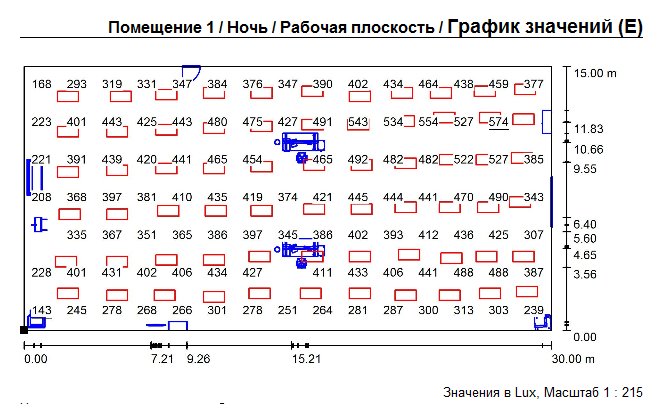


Рисунок 6.10 ‒ График значений светового потока для рабочей плоскости, схема «Ночь»

**6.4 Выводы**

В данном разделе был проведен анализ опасных и вредных факторов при разработке программного обеспечения, проведены акустический расчет и расчет системы искусственного освещения для рабочих мест.

# **Заключение**

# **Список литературы**

1. Яндекс карты
2. Google Maps
3. OSM
4. GraphHooper
5. ПДД http://www.pdd24.com/
6. E. F. Moore (April 2–5, 195). «The shortest path through a maze» (Harvard University Press): 285–292.
7. Bellman <http://bioinfo.ict.ac.cn/~dbu/AlgorithmCourses/Lectures/Bellman1958.pdf>
8. L. R. Ford, Jr. & D. R. Fulkerson Flows in Networks
9. Floyd, Robert W. (June 1962). "Algorithm 97: Shortest Path".
10. Stephen Marshall Theorem on Boolean Matrices
11. Дейкстра
12. http://ai.stanford.edu/~nilsson/OnlinePubs-Nils/PublishedPapers/astar.pdf
13. Йен <http://people.csail.mit.edu/minilek/yen_kth_shortest.pdf>
14. Статья Степанова
15. <http://wiki.openstreetmap.org/wiki/Map_Features>
16. <http://azure.microsoft.com/ru-ru/pricing/free-trial/>

# **Приложение А**