**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

междисциплинарный проект

Тема: Объектно-ориентированные технологии разработки программного обеспечения

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4304 |  | Шурыгина М. |
| Студент гр. 430 |  | Пестерев Д. |
| Студентка гр. 4303 |  | Попова Д. Е. |
| Студентка гр. 430 |  | Колосова О. |
| Студентка гр. 430 |  | Неверова Е. |
| Руководитель |  | Спицын А.В. |

Санкт-Петербург

2018

**ЗАДАНИЕ на** междисциплинарный проект

|  |  |  |  |
| --- | --- | --- | --- |
| Студентка Шурыгина М. | | Группа 4304 | |
| Студент Пестерев Д. | | Группа 4304 | |
| Студентка Попова Д. Е. | | Группа 4303 | |
| Студентка Колосова О. | | Группа 4304 | |
| Студентка Неверова Е. | | Группа 4303 | |
| Тема НИР: Объектно-ориентированные технологии разработки программного обеспечения | | | |
| Исходные данные (технические требования):  На языке программирования C++ реализовать:   * шаблон полиморфного контейнера (мультиграф) на основе STL-контейнеров, использовать шаблоны проектирования; * систему классов исключительных ситуаций, стратегию контроля аномального поведения программы; * сериализацию, манипулятор, аллокатор; * графический интерфейс, визуализацию приложения.   Разработать набор диаграмм UML, диаграмму Ганта, требуемую документацию и артефакты по методологии разработки RUP.  Содержание пояснительной записки: «Содержание», «Введение», «Заключение», «Список используемых источников».  Предполагаемый объём пояснительной записки: не менее 20 страниц. | | | |
| Дата выдачи задания: 08.02.2018 | | | |
| Дата сдачи проекта: 31.05.2018 | | | |
| Дата защиты проекта: 31.05.2018 | | | |
| Студентка гр. 4304 |  | | Шурыгина М. |
| Студент гр. 430 |  | | Пестерев Д. |
| Студентка гр. 4303 |  | | Попова Д. Е. |
| Студентка гр. 430 |  | | Колосова О. |
| Студентка гр. 430 |  | | Неверова Е. |
| Руководитель |  | | Спицын А.В. |

**Аннотация**

В данном междисциплинарном проекте при использовании объектно-ориентированной технологии разработки был реализован программный комплекс. Программный комплекс решает задачу создания собственной модели проверки правописания на основе скрытых марковских цепей (СММ) и создания модели на основе сбора статистики по заданному словарю для конкретного языка. Программный комплекс состоит из трёх модулей: полиморфный контейнер (мультиграф) на базе STL, алгоритм построения СММ и графический интерфейс пользователя. Реализация полиморфного контейнера осуществлялась в среде программирования Qt. Разработка программного комплекса производилась в соответствии с методологией разработки программного обеспечения RUP. Определены и разработаны необходимые RUP-артефакты.

**Summary**

In this interdisciplinary project, using object-oriented development technology, a software package was implemented. The program complex solves the problem of building the optimal way to travel between the sights of St. Petersburg. The software complex consists of three modules: a polymorphic container (multigraph) based on STL, an algorithm for finding the optimal path and a graphical user interface. The polymorphic container was implemented in the Qt programming environment. Development of the software package was carried out in accordance with the methodology of software development RUP. The necessary RUP-artifacts are defined and developed.

Содержание

**[введение](#__RefHeading___Toc514881415)** [5](#__RefHeading___Toc514881415)

**[1. Анализ и проектирование](#__RefHeading___Toc514881416)** [6](#__RefHeading___Toc514881416)

[1.1. Анализ предметной области 6](#__RefHeading___Toc514881417)

[1.2. Структура проекта 7](#__RefHeading___Toc514881418)

[1.3. Планирование 10](#__RefHeading___Toc514881419)

[1.4. Проектирование 12](#__RefHeading___Toc514881420)

**[2. Реализация](#__RefHeading___Toc514881421)** [16](#__RefHeading___Toc514881421)

[2.1. Используемые технологии 16](#__RefHeading___Toc514881422)

[2.2. Шаблоны проектирования 17](#__RefHeading___Toc514881423)

[2.3. Реализация требований 17](#__RefHeading___Toc514881424)

[2.4. Рефакторинг 19](#__RefHeading___Toc514881425)

[2.4. Тестирование 20](#__RefHeading___Toc514881425)

**[3. Результаты разработки](#__RefHeading___Toc514881426)** [21](#__RefHeading___Toc514881426)

[3.1. Пример работы программы 21](#__RefHeading___Toc514881427)

[3.2. Дальнейшие перспективы 22](#__RefHeading___Toc514881428)

**[заключение](#__RefHeading___Toc514881429)** [23](#__RefHeading___Toc514881429)

**[список использованных источников](#__RefHeading___Toc514881430)** [24](#__RefHeading___Toc514881430)

**[приложение А. Глоссарий](#__RefHeading___Toc514881441)** [25](#__RefHeading___Toc514881441)

**[приложение Б. Видение](#__RefHeading___Toc514881448)** [30](#__RefHeading___Toc514881448)

**[приложение В. План проекта](#__RefHeading___Toc514881455)** [35](#__RefHeading___Toc514881455)

**[приложение Г. Экономическое обоснование](#__RefHeading___Toc514881465)** [37](#__RefHeading___Toc514881465)

**[приложение Д. Список рисков](#__RefHeading___Toc514881521)** [39](#__RefHeading___Toc514881521)

**[приложение Е.](#__RefHeading___Toc514881548)****[ПРЕЦЕДЕНТ РАЗРАБОТКИ](#__RefHeading___Toc514881548)** [44](#__RefHeading___Toc514881548)

**[приложение Ж. СПЕЦИФИКАЦИЯ ПРЕЦЕДЕНТОВ](#__RefHeading___Toc514881558)** [49](#__RefHeading___Toc514881558)

**[приложение З. прототип интерфейсов](#__RefHeading___Toc514881569)** [55](#__RefHeading___Toc514881569)

**[приложение И. ДОКУМЕНТ АРХИТЕКТУРЫ ПО](#__RefHeading___Toc514881582)** [57](#__RefHeading___Toc514881582)

**[приложение К. Компиляция](#__RefHeading___Toc514881608)**66

**[приложение л. план тестирования](#__RefHeading___Toc514881622)** [6](#__RefHeading___Toc514881622)8

**[приложение М. ТЕСТИРОВАНИЯ ШАБЛОНА «МУЛЬТИГРАФ»](#__RefHeading___Toc514881698)** [7](#__RefHeading___Toc514881698)2

**[приложение Н. Тестирование графического интерфейса](#__RefHeading___Toc514881716)** [7](#__RefHeading___Toc514881716)6

# **введение**

Цель данного проекта – создать модель проверки правописания на основе скрытых марковских цепей (СММ) и сформировать модель на основе сбора статистики по заданному словарю для конкретного языка.

Задачи проекта – разработать на языке программирования C++ программную систему, в рамках которой реализуется полиморфный контейнер на базе STL-контейнера. Реализуемый контейнер использует мультиграф для поиска оптимального пути.

В качестве среды разработки используется Qt.

Разработка проекта ведется в соответствии с методологией RUP. В рамках методологии RUP было определено количество итераций каждой стадии разработки. Были распределены роли в команде разработчиков. Определен перечень поставляемых документов (RUP-артефактов) в отчет.

В работе над проектом были применены знания таких дисциплин как: Объектно-ориентированное программирование, Алгоритмы и структуры данных, Конструирование ПО, Тестирование ПО.

# **1. Анализ и проектирование**

## 1.1. Анализ предметной области

Предметную область можно определить как сферу человеческой деятельности, выделенную и описанную, согласно установленным критериям. В описываемое понятие должны входить сведения об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты этой деятельности. Одна из первых задач, с решением которых сталкивается разработчик программной системы – это изучение и анализ предметной области. Дело в том, что предметная область сильно влияет на все аспекты проекта: требования к системе, взаимодействие с пользователем, модель хранения данных, реализацию и т.д. Анализ предметной области позволяет выделить её сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации. Для документирования могут быть использованы различные средства. Анализ предметной области является основой для анализа осуществимости проекта и определения образа (концепции) продукта и границ проекта.

По данным информационного портала ТАСС порядка 7,5 млн российских и иностранных туристов посетили Санкт-Петербург в 2017 году. При этом в 2012 году культурную столицу посетили 3,5 млн туристов. При постоянно увеличивающемся турпотоке и ограниченном времени пребывания в городе появляется задача поиска оптимального пути для посещения наибольшего количества достопримечательностей.

Объектом разработки является приложение, которое позволит строить оптимальный маршрут посещения достопримечательностей либо пешком, либо в такси, либо в метро и обеспечит более комфортное и выгодное перемещение между выбранными пунктами. Ввод маршрута должен быть возможен в автоматическом режиме (выбор достопримечательностей на карте).

Альтернативой разработке системы была бы доработка имеющихся в открытом доступе программ, посредством добавления функция учитывающих платежеспособность пользователя и местонахождение достопримечательностей. Но имеющиеся на рынке программы коммерческие, что не позволяет сделать вышеуказанные манипуляции. Таким образом, система «SightsNavigator», предназначенная для построения оптимального маршрута между достопримечательностями, будет разработана под актуальные запросы туристов.

## 1.2. Структура проекта

Проект выполнялся согласно методологии разработки программного обеспечения – Rational Unified Process (RUP). RUP – это процесс, направленный на поддержку коллективной разработки ПС. Все участники проекта используют единую базу знаний, единый процесс, единый взгляд на разработку, единый язык моделирования. Все модели в RUP представляются в нотации Unified Modeling Language (UML). RUP – это технологический процесс по созданию программных средств, позволяющий улучшить производительность коллективной разработки путем предоставления для всех этапов жизненного цикла методик выполнения основных видов деятельности, шаблонов документов, инструкций по работе с инструментальными средствами.

В основе RUP лежат следующие принципы:

* Ранняя идентификация и непрерывное (до окончания проекта) устранение основных рисков.
* Концентрация на выполнении требований заказчиков к исполняемой программе (анализ и построение модели прецедентов (вариантов использования)).
* Ожидание изменений в требованиях, проектных решениях и реализации в процессе разработки.
* Компонентная архитектура, реализуемая и тестируемая на ранних стадиях проекта.
* Постоянное обеспечение качества на всех этапах разработки проекта (продукта).
* Работа над проектом в сплочённой команде, ключевая роль в которой принадлежит архитекторам.

RUP предоставляет структурированный подход к итерационной разработке программного обеспечения, подразделяя процесс на четыре основные фазы во времени: Inception (исследование, начало), Elaboration (уточнение плана), Construction (конструирование, построение) и Transition (переход, развертывание). На рисунке 1 представлено широкораспространенное изображение фаз RUP. Целями каждой из данных фаз являются:

* Inception — фаза сбора информации и анализа требований, определение образа проекта в целом.
* Elaboration — фаза анализа требований и проектирования системы, планирование необходимых действий и ресурсов, спецификация функций и особенностей дизайна.
* Construction — основная фаза разработки и кодирования, построение продукта как восходящей последовательности версий кода.
* Transition — фаза внедрения продукта, поставка продукта конкретному пользователю.

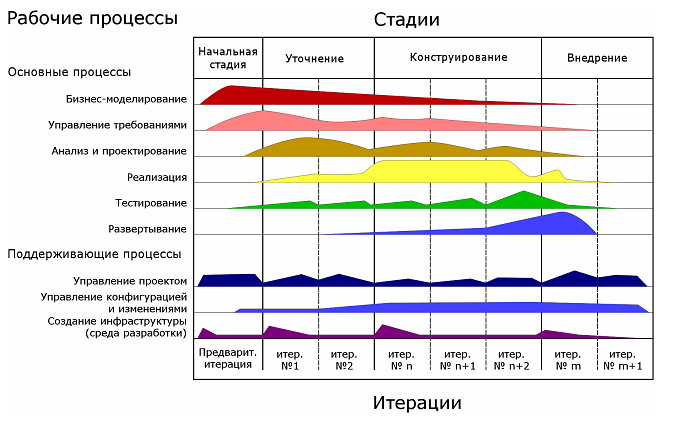


Рисунок 1– Графическое представление процесса разработки по RUP.

В проекте также были выделены 4 фазы:

* Фаза Начало. Были определены цели и границы проекта. План задач, требующийся при разработке программной системы, описан в Плане проекта (приложение В**)**. Более подробно цели проекта отражены в Экономическом обосновании (приложение Г). Также был произведён обзор функциональности системы, который отражён в Видении проекта (приложение Б) и оценена выполнимость проекта в Списке рисков проекта (приложение Д).
* Фаза Уточнение. Были проанализированы требования проекта. При анализе требований проекта были составлены такие отчёты, как Спецификация прецедентов (приложение Ж), Прототип интерфейсов (приложение З), Документ архитектуры ПО (приложение И). Были разработаны диаграммы UML и выбраны шаблоны проектирования.
* Фаза Реализация. При разработке проекта были составлены такие отчёты, как Документ по компиляции (приложение К), План тестирования (приложение Л), Тестирование конвейера (приложение М), Тестирование GUI (приложение Н). Производилась доработка диаграмм UML.
* Фаза Внедрение. Была произведена конечная оценка проект, ее результаты были добавлены в пояснительную записку к разработанному приложению, составленную при внедрении продукта.

## 1.3. Планирование

Для иллюстрации плана, графика работ по проекту используется диаграмма Гантта, которая представлена на рисунке 2 (более подробно данная диаграмма представлена в конце приложения Б). Менеджер проекта по согласованию с координатором проекта от Заказчика и участниками проектной команды может перераспределять должным образом сроки задач внутри каждой фазы.

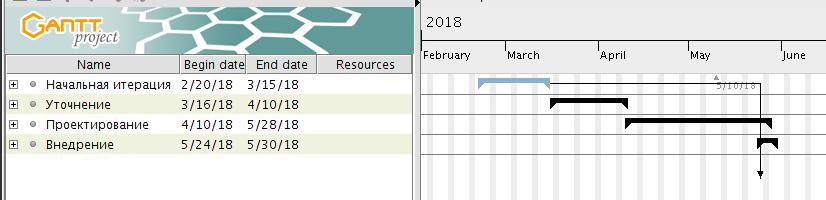


Рисунок 2 – План проекта с указанием основных вех.

Диаграмма Гантта состоит из полос, ориентированных вдоль оси времени. Каждая полоса на диаграмме представляет отдельную задачу в составе проекта (вид работы), её концы — моменты начала и завершения работы, её протяженность — длительность работы. Вертикальной осью диаграммы служит перечень задач. Также на диаграмме могут быть отмечены совокупные задачи, проценты завершения, указатели последовательности и зависимости работ, метки ключевых моментов (вехи) и др.

В RUP определено 9 технологических процессов, для каждого из которых предложена методика выполнения. Технологические процессы делятся на две категории – основные процессы и процессы поддержки.

Основными технологическими процессами являются: бизнес-анализ, управление требованиями, анализ и проектирование, реализация, тестирование, развертывание.

Вспомогательные процессы включают: управление проектом, управление конфигурацией, управление средой.

Для каждого технологического процесса предусмотрены роли, определяющие поведение и обязанности отдельных лиц и групп, работающих в одной команде, виды деятельности, определяющие работы, выполняемые исполнителями и артефакты – документы, используемые, порождаемые или модифицируемые процессом.

Распределение членов команды по ролям в проекте представлено в таблице 1:

Таблица 1 – Распределение ролей в проекте.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Шурыгина Мария | Пестерев Дмитрий | Попова Дарья | Неверова Елена | Колосова Ольга |
| Руководитель проекта (Project Leader) | + |  |  |  |  |
| Разработчик архитектуры (Architect) |  | + |  | + |  |
| Проектировщик (Designer) |  |  | + |  | + |
| Технический писатель (Technical Writer) |  |  | + | + | + |
| Разработчик (Developer) | + | + |  | + |  |
| Специалист по пользовательскому интерфейсу (User Interface Designer) |  |  | + | + |  |
| Тестировщик (Tester) |  | + |  |  | + |
| Специалист по инструментам (Tool Specialist) | + |  |  |  | + |
| Аналитик (Analyst) | + | + | + |  | + |

Проект построен по принципу одной итерации на каждую фазу. Поэтому цели каждой итерации совпадают с целями, определенными соответствующей фазой. Проект будет иметь только один релиз в конце фазы Разработка. График проекта с датами начала и окончания фаз представлен в таблице 2.

Таблица 2 – График проекта.

|  |  |  |  |
| --- | --- | --- | --- |
| Фаза | Число итераций | Начало | Конец |
| Начало | 1 | 20.02.2019 | 15.03.2019 |
| Уточнение | 1 | 16.03.2019 | 10.04.2019 |
| Реализация | 1 | 10.04.2019 | 23.05.2019 |
| Внедрение | 1 | 20.05.2019 | 31.05.2019 |

## 1.4. Проектирование

Цель процесса анализа и проектирования состоит в разработке технических инструкций, предписывающих, как реализовать программное средство, удовлетворяющую сформулированным требованиям. Для этого следует хорошо понять требования к системе и преобразовать их в проект системы, выбрав правильную стратегию реализации. На ранних стадиях процесса должна быть создана устойчивая архитектура, на основе которой можно спроектировать программное обеспечение, легкая для понимания, построения и развертывания. Архитектура должна быть согласована со средой реализации с целью удовлетворения требований к производительности, устойчивости, безопасности, расширяемости и тестируемости.

К числу решаемых задач при этом относятся:

* разработка точной архитектуры распределенной программной системы;
* преобразование модели требований в проектную модель разрабатываемой системы;
* адаптация проекта системы к среде реализации с целью повышения производительности разработки;
* выбор механизмов реализации и определение ограничений на реализацию;
* разработка компонентной структуры;
* распределение компонентов по узлам.

Главной задачей анализа является преобразование требований в форму, понятную разработчику, то есть, определение подсистем, компонентов и классов, с помощью которых реализуется требуемое поведение системы. В основе такого преобразования лежат варианты использования, созданные при определении требований к программе. В связи, с чем был произведен сбор, анализ и определение высокоуровневых требований, потребностей пользователей для разрабатываемого приложения ««SightsNavigator» (см. Приложение).

Далее была выявлены профили пользователей, разработана модель вариантов использования (ВИ), составлена спецификация для каждого ВИ и взаимодействий между ними.

Выявленные профили пользователей:

* Пользователь системы – пользователи могут добавлять и удалять метки достопримечательностей, вводить имеющееся количество денег, просматривать предложенный путь. Прокладка маршрута возможна с помощью курсора путём выбора достопримечательностей на карте.

Разработанная модель вариантов использования представлена на рисунке 3 в виде диаграммы прецедентов в UML.

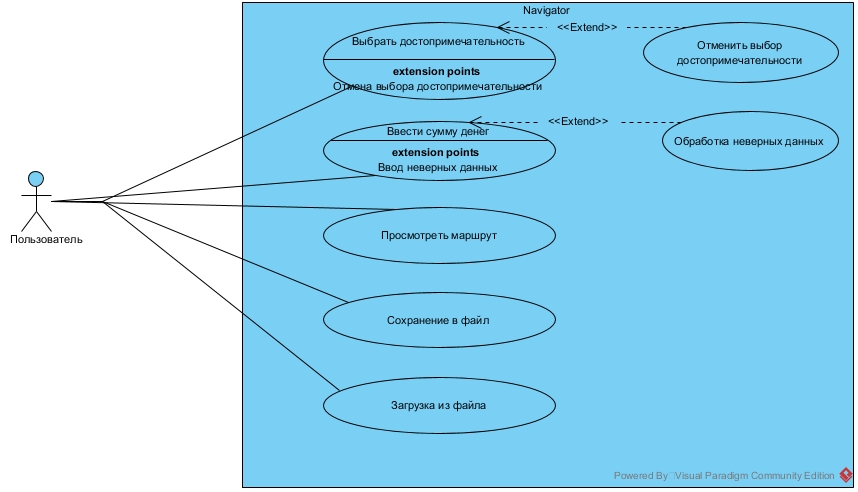
****

Рисунок 3 – Use case диаграмма.

Диаграмма отражает отношения между актёрами и прецедентами и является составной частью модели прецедентов. Прецедент — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Далее было разработано логическое представление системы. Оно состоит из пакета MultiGraph. Данный пакет предоставляет контейнер для хранения графа и возможности работы с контейнером. Также предоставляет пользовательский интерфейс и реализует основные алгоритмы программы.

В соответствии с диаграммой использования построена диаграмма состояний системы, представленная на рисунке 4.

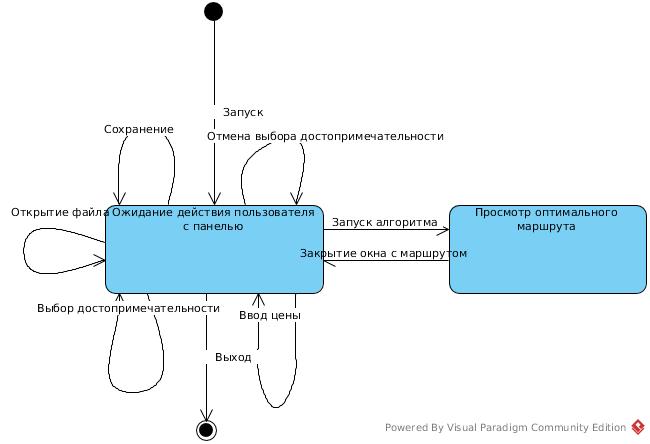


Рисунок 4 – Диаграмма состояний системы

1. 2. РЕАЛИЗАЦИЯ
   1. Система реализована как единая исполняемая программа. Саму систему можно разделить на 3 слоя: слой интерфейса, слой контейнера и внешний слой.
2. **Слой интерфейса -** набор классов, реализующих интерфейс и функции активных элементов.
3. **Слой контейнера -** набор шаблонных классов, реализующих функции контейнера **MultiGraph.**
4. **Внешний слой -** набор классов и возможностей, полученных из внешних библиотек.
5. Требования к реализации:
6. 1) Система должна работать на операционной системе Windows.
7. 2) Все функции должны быть доступны и понятны пользователю.
8. 3) Обязательное наличие контейнера Мультиграф с использованием библиотеки STL.
9. 4) Контейнер должен использовать собственный аллокатор, манипулятор и поддерживать собственные исключения.
10. **2.1. Использованные технологии**
11. Git – распределенная система управления версиями;
12. GitHub – веб-сервис для хостинга IT-проектов и их совместной разработки;
13. Google Disk – облачный сервис хранения данных;
14. STL – библиотека согласованных обобщённых алгоритмов, контейнеров, средств доступа к их содержимому и различных вспомогательных функций в C++;
15. Qt – кроссплатформенный фреймворк для разработки ПО на языке программирования С++. Далее подробнее об использованных частях, входящих в Qt:
16. QtCreator – кроссплатформенный свободный IDE для разработки на С, С++ и QML;
17. QMake – утилита облегчающая процесс сборки кроссплатформенных приложений;
19. Использованные библиотеки:
20. QtCore – классы ядра библиотеки, используемые другими модулями;
21. QtGui – компоненты графического интерфейса;
22. QtWidgets – содержит классы для классических приложений на основе виджетов;
23. **2.2. Шаблоны проектирования**
24. Пакет MultiGraph реализует контейнер Мультиграф(класс Multigraph), который является шаблоном, содержащим в себе список вершин. Для его разработки используются такие шаблоны проектирования как Пул объектов и Итератор. Пул объектов представлен классом Allocator, предоставляющим интерфейс для выделения и освобождения памяти для объектов контейнера. Итератор представлен классом **VertexIterator** и предоставляет возможности для получения доступа к элементам контейнера и используется для обхода графа по вершинам.
25. **2.3. Реализация требований**
26. **2.3.1. Реализация шаблона полиморфного контейнера (мультиграф), построенного на основе STL-контейнера**
27. В теории графов мультиграфом (или псевдографом) называется граф, в котором разрешается присутствие кратных рёбер (их также называют «параллельными»), то есть рёбра, имеющие те же самые конечные вершины. Таким образом, две вершины могут быть соединены более чем одним ребром.

Шаблон полиморфного контейнера (мультиграф) реализуется на основе последовательного STL-контейнера вектор (vector).

1. **2.3.2. Реализация системы классов исключительных ситуаций**
2. В разработанной системе реализована обработка следующих исключительных ситуаций:
3. **MultigraphVertexExistingException -** исключительные ситуации, связанные с обработкой обращений к вершинам графа. Исключение возникает при обращении к несуществующей вершине или при попытке повторно создать идентичную вершину.

**MultigraphIncorrectIndexException -** исключительные ситуации, связанные с обращением по индексу к несуществующему элементу.

**MultigraphAllocatorException** - Исключительные ситуации, связанные с управлением и размещением в памяти объектов контейнера.

1. **MultigraphException -** обработка прочих исключительных ситуаций. Базовый класс для вышеперечисленных исключений, наследуется от std::exception.
2. **2.3.3. Реализация сериализации и манипулятора**
3. Сериализация происходит с помощью std::stream, очередь сериализации имеет следующий вид: число вершин, вершины, число дуг, дуги. Сериализация происходит в специальный файл с расширением “.mg”.
4. **2.3.4. Реализация аллокатора**
5. Разработан класс Allocator с использованием порождающего шаблона проектирования "объектный пул", хранит указатели на всю выделенную память, при очищении или разрешении контейнера, безопасно возвращает всю зарезервированную память.

**2.3.5. Реализация проверки инвариантов контейнера предусловий и постусловий для контейнера**

1. Для контейнера проверяются два инварианта:

1. Инвариант предусловия — проверка того, что выбранные вершины находятся внутри одного связного подграфа.

2. Инвариант постусловия - проверка того, что найденные оптимальный путь не пуст.

## 2.4. Рефакторинг

Рефакторинг, или перепроектирование кода, переработка кода, равносильное преобразование алгоритмов – процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы. В основе рефакторинга лежит последовательность небольших эквивалентных (то есть сохраняющих поведение) преобразований. Поскольку каждое преобразование маленькое, программисту легче проследить за его правильностью, и в то же время вся последовательность может привести к существенной перестройке программы и улучшению её согласованности и чёткости.

Перечислены произведённые рефакторинги при разработке приложения:

* Извлечение метода – удаление повторяющегося кода в методах класса графа.
* Скрытие метода – классы узла и ребер графа используются только классом графа и классом-наследником графа, поэтому методы, осуществляющие поиск объектов класса узлов и ребер, отмечены как защищенные.
* Замещение поля структурой – каждое ребро содержит не одну, а три метки. Для группировки всех меток одного ребра была создана структура.
* Переименование метода – методы классов переименовывались в соответствии с производимыми операциями.
* Удаление избыточного кода - Удален избыточный код, затрудняющий чтение и понимание кода, удалены закомментированные функции и переменные.
* Группировка функций - Сгруппированы функции класса Multigraph по их назначению.

**2.5 Тестирование**

Тестирование было проведено по ряду следующих вещей: функциональными и нефункциональными требованиями, вариантами использования, элементами дизайна, а так же запросами на изменение.

В ходе тестирования ПС использовались следующие группы тестов:

* Функциональное тестирование, которое позволит проверить на соответствие требованиям, описанным в документации;
* Модульное тестирование, необходимое для тестирования шаблона контейнера и алгоритма;
* Тестирование пользовательского интерфейса для тестирования приложения.

Такая система тестирования позволяет выполнять детальное тестирование и предотвращать, исправлять ошибки на ранних этапах.

После создания приложения для разработанного шаблона были выполнены следующие функциональные тесты, результаты представлены в приложении М:

1. Создание вершины
2. Проверка существования вершины
3. Получение меток ребер
4. Добавление ребра
5. Добавление уже существующей вершины
6. Поиск оптимального маршрута
7. Очистка контейнера
8. Сериализация контейнера

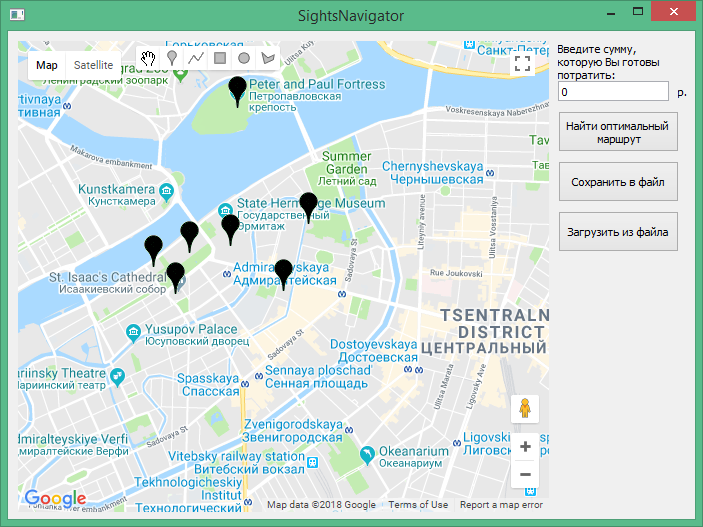
Также было проведено тестирование графического интерфейса пользователя (приложение Н)

# **3. Результаты разработки**

## 3.1. Пример работы программы

На основе описанных в п. 1.4 этапа проектирования были разработаны: use-case диаграмма, диаграммы последовательностей, диаграммы деятельностей и разработаны спецификации вариантов использования, диаграммы классов. На основе разработанной архитектуры было реализовано приложение, назначение которого - построение оптимального маршрута между достопримечательностями города Санкт-Петербург.

На рисунке 7 представлен пример вида главного экрана приложения.

1. 
2. Рис. 1 – Главное окно приложения

## 3.2. Дальнейшие перспективы

Дальнейшим развитием для разработанного приложения могут быть:

* усовершенствование визуализации приложения;
* увеличение производительности приложения для расширения использования.
* увеличение количества валют
* добавление интерфейса на иностранных языках
* добавление администраторской панели
* добавление кроссплатформенности.

# **заключение**

В ходе работы над междисциплинарным проектом были достигнуты поставленные цели.

Процесс разработки сопровождался согласно методологии RUP. Для каждой итерации разработан соответствующий пакет документов.

Была спроектирована и реализована программная система на языке C++ (с использованием стандартной библиотеки шаблонов STL). При выполнении междисциплинарного проекта получен дополнительный опыт владения языком программирования C++, опыт проектирования пользовательских интерфейсов и разработки документации и управление проектом.

Был получен практический опыт разработки программного продукта в составе проектной команды.

# **список использованных источников**

1. Мейерс С. Эффективное использование STL //Библиотека программиста. СПб.: Питер. – 2002. 224 с.

2. Страуструп Б. Программирование. Принципы и практика использования C++.: Пер. с англ. – М. : ООО «И.Д. Вильямс», 2011. – 1248 с.

3. Фаулер М. UML Основы. СПб.: Символ-Плюс, 2004. 192 c.

4. Введение в RUP // Rational Method Composer. URL: [http://dit.isuct.ru/Publish\_RUP/#core.base\_rup/guidances/supportingmaterials/welcome\_2BC5187F.html](http://dit.isuct.ru/Publish_RUP/" \l "core.base_rup/guidances/supportingmaterials/welcome_2BC5187F.html) (дата обращения: 21.02.2018)

5. Lee Z. E. Qt5 C++ GUI Programming Cookbook. Birbingham: Packt Publishing Ltd., 2016. 339 c.

6. ДеМарко Т. Роман об управлении проектами Том Демарко; пер. с англ. А. Максимовой.-3-е изд.-Сер. 061 Хороший перевод. – Манн, Иванов и Фербер ББК: Ю959. 18, 0, 2012. 352 с.

7. Вигерс К. И. Разработка требований к программному обеспечению. – М : Рус. Ред., 2004. 575 с.

8. Роберт К. Чистый код. Создание, анализ и рефакторинг. //Библиотека программиста. СПб.: Питер. – 2010. 464 с.

9. Поток туристов в Санкт-Петербург увеличился вдвое за последние пять лет // TASS. URL: <http://tass.ru/obschestvo/4887154> (дата обращения: 25.02.2018).

10. Библиотека виджетов Qt Material Design Desktop Widgets // Github. URL: <https://github.com/laserpants/qt-material-widgets> (дата обращения: 22.03.2018).

# ПРИЛОЖЕНИЕ А. ГЛОССАРИЙ

SightsNavigator

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 30.04.18 | 1.0 | Создание документа | Сандин А.В. |
| 15.05.18 | 1.1 | Добавление новых сокращений | Аксёнов Г.Н. |

**Введение**

Данный документ содержит определения терминов в системе подбора персонала. Документ будет расширяться на протяжении всего жизненного цикла проекта. Любые определения, не включенные в данный документ, могут быть включены.

**Глоссарий**

**GUI**

Графический пользовательский интерфейс - разновидность интерфейса, в котором элементы интерфейса исполнены в виде графических изображений.

**RUP**

Rational Unified Process - процесс, направленный на коллективную разработку ПС.

**Use-cases**

Сценарии использования - описание поведения ПС при взаимодействии с пользователем.

**Алгоритм**

Набор функций, реализующих взаимодействие между контейнером и приложением.

**Артефакт**

Практический результат, который производится во время выполнения мероприятий, связанных с проектом. В качестве артефактов выступают: документы, диаграммы, исходный код.

**Версия**

Состояние рабочего артефакта, которое может быть восстановлено в любой момент времени независимо от истории изменения.

**Граф**

Абстрактный математический объект, представляющий из себя множество вершин графа и набор рёбер, то есть соединений между парами вершин.

**Диаграмма Ганта**

Тип столбчатых диаграмм, который используется для иллюстрации плана, графика работ по какому-либо проекту.

**Диаграмма деятельности**

Диаграмма, на которой показаны действия, состояния которых описано на диаграмме состояний

**Диаграмма классов**

Диаграмма описывающая типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами.

**Диаграмма коммуникации**

Диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации.

**Диаграмма компонентов**

Диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

**Диаграмма объектов.**

Снимок объектов системы в какой-то момент времени. Поскольку она показывает экземпляры, а не классы, то диаграмму объектов часто называют диаграммой экземпляров.

**Диаграмма последовательности**

Диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определённого объекта (создание-деятельность-уничтожение некой сущности) и взаимодействие актёров (действующих лиц) ИС в рамках какого-либо определённого прецедента (отправка запросов и получение ответов).

**Диаграмма развертывания**

Диаграмма, представляющая физическое расположение системы, показывая, на каком физическом оборудовании запускается та или иная составляющая программного обеспечения.

**Диаграмма состояний**

Диаграмма, определяющая все возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий. Диаграммы состояний строятся для единственного класса и описывают поведение единственного объекта.

**Интерфейс**

Совокупность методов и правил взаимодействия программ между собой или с пользователем, а также средств, реализующих это взаимодействие.

**Компонент (в приложении)**

Графический элемент пользовательского интерфейса, предоставляющий функциональность для работы с остальными частями ПС.

**Контейнер**

Структура, позволяющая инкапсулировать в себя объекты разных типов.

**Конфигурация**

Совокупность версий рабочих артефактов.

**Маршрут**

Путь следования объекта, учитывающий направление движения относительно географических ориентиров или координат, с указанием начальной, конечной и промежуточных точек, в случае их наличия.

**Мультиграф**

Граф, в котором разрешается присутствие кратных ребер, то есть ребер, имеющих те же самые конечные вершины.

**Пользователь ПС**

Работодатель, использующий приложение для взаимодействия с программной системой. Целью взаимодействия является поиск специалистов.

**Приложение**

Часть ПС, предоставляющая графический интерфейс пользователя для работы с другими частями ПС.

**Проект**

Экземпляр выполнения ПС со своим набором специальностей и выбранных специалистов.

**Прототип интерфейса**

Действующая модель пользовательского интерфейса. Он имитирует работу системы, так что ее можно оценить в действии еще до того, как начата разработка.

**ПС**

Программная система - совокупность приложений и программ, конструктивно объединенных в единое изделие для выполнения определенной задачи.

**Рефакторинг**

Процесс изменения внутренней структуры программы, не затрагивая ее внешнего поведения и имеющий целью облегчить понимание работы программы.

**Риск**

Сочетание вероятности и последствий наступления неблагоприятных событий.

**Система**

Совокупность элементов, находящихся в отношениях и связях между собой и образующих определенную целостность, единство.

**Спецификация**

Документ, который точно, полностью и в поддающейся проверке форме определяет требования, устройство, поведение или другие особенности системы, компонента, продукта, результата или услуги, а также процедуры, способные определить, были ли выполнены эти условия.

**Средство разработки**

Система программных средств, используемая программистами для разработки программного обеспечения.

**Фаза проекта**

Отдельные части в рамках проекта, требующие дополнительного контроля для эффективного управления достижением основного результата проекта.

**Файл**

Именованная область данных на носителе информации.

# **приложение Б. видение**

# SightsNavigator

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 30.04.17 | 1.0 | Создание документа | Аксёнов Г.Н.  Сандин А.В. |
| 10.05.17 | 2.0 | Изменение разделов Видение и обзор продукта | Аксёнов Г.Н. |

Оглавление

[1. Введение 3](#__RefHeading___Toc508652485)0

[1.1 Цель 3](#__RefHeading___Toc508652486)0

[1.2 Сфера применения 3](#__RefHeading___Toc508652493)1

[1.3 Определения, акронимы и сокращения. 3](#__RefHeading___Toc508652500)1

[1.4 Ссылки 3](#__RefHeading___Toc508652507)1

[2. Позиционирование 3](#__RefHeading___Toc508652521)1

[2.1 Решаемая проблема 3](#__RefHeading___Toc508652486)1

[2.2 Положение продукта на рынке 3](#__RefHeading___Toc508652493)1

[2.3 Заинтересованные лица 3](#__RefHeading___Toc508652500)2

[2.4 Рабочая среда пользователя 3](#__RefHeading___Toc508652507)2

[3. Обзор продукта 3](#__RefHeading___Toc508652521)2

[3.1 Перспективы продукта 3](#__RefHeading___Toc508652486)2

[3.2 Допущения и зависимости 3](#__RefHeading___Toc508652493)2

[3.3 Потребности и возможности 3](#__RefHeading___Toc508652500)3

[3.4 Альтернативы и конкуренция 3](#__RefHeading___Toc508652507)3

[4. Другие требования к продукту 3](#__RefHeading___Toc508652521)3

**1. Введение**

Целью данного документа является определение представления заинтересованных лиц о разрабатываемом продукте “Путеводитель по достопримечательностям города”. Данный документ содержит структуру представляемых основных требований для данного приложения, обеспечивая договорную основу для более подробных технических требований.

**1.1. Цель**

Цель этого документа состоит в том, чтобы определить общие требования пользователей к продукту.

**1.2. Сфера применения.**

Данный документ описывает продукт “Путеводитель по достопримечательностям города”. Эта программная система позволяет пользователям находить оптимальный маршрут для посещения выбранных достопримечательностей.

**1.3. Определения, акронимы и сокращения.**

См. документ “Глоссарий”.

**1.4. Ссылки**

Отсутствуют

1. **Позиционирование**

**2.1 Решаемая проблема**

Таблица 1 – Решаемая проблема

|  |  |
| --- | --- |
| Проблема | Нахождение оптимального способа посещения выбранных достопримечательностей города выбранного туристом. У туриста при посещении незнакомого города возникают проблемы при выборе идеального маршрута для посещения как можно большего количества достопримечательностей. |
| Затрагивает | Туристов |
| Влияет | На время, затрачиваемое на передвижение между достопримечательностями. |
| Успешное решение | Позволит туристу передвигаться как можно экономнее и быстрее. |

**2.2. Положение продукта на рынке**

Продукт подходит для туристов, возрастом до 25 лет, желающим в одиночку или в небольшой компании посетить множество достопримечательностей города за оптимальное время, без экскурсий от туристических агентств.

**2.3. Заинтересованные лица**

Таблица 2 – Заинтересованные лица

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Responsibilities** |
| Студенты | Разработчики ПО | Создание конечного программного продукта, создание документации. |
| Пользователь | Турист | Использование ПО |

**2.4. Рабочая среда пользователя**

Работа проводится на персональном компьютере с ОС Windows. Пользователю предоставляется графический интерфейс. В один момент времени приложением может пользоваться только один пользователь. Все необходимые расчеты выполняются автоматически.

1. **Обзор продукта**
   1. **Перспективы продукта**

Данный продукт не имеет зависимостей от стороннего ПО, не требует установки сторонних библиотек, поддерживает операционные системы семейства Windows.

* 1. **Допущения и зависимости**

Изменения следующих факторов повлечет за собой изменение текущего документа:

* изменения списка поддерживаемых ОС,
* изменение перечня ключевых потребностей пользователей,
* расширения списка возможностей ПО.
  1. **Потребности и возможности**

Таблица 3 – Потребности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Необходимость | Приоритет | Проблема | Текущее решение | Предполагаемое решение |
| Экономить время и деньги при передвижении между достопримечательностями | Высокий | Да | Использование навигаторов. | Добавление достопримечательностей на карту с указанием затрачиваемых средств и времени на передвижение между ними (метро, такси, пешком). |

* 1. **Альтернативы и конкуренция**

Преимущества данного программного продукта перед аналогами (2gis, MAPS.ME) следующие:

1) Более точное описание достопримечательностей.

2) Более подходящая функциональность для создания экскурсий.

3) Определение оптимальности пути по денежному критерию.

**4. Другие требования к продукту**

* Язык программирования – C++.
* Реализовать шаблон полиморфного контейнера (мультиграф), построенного на основе STL-контейнера.
* Реализовать систему классов исключительных ситуаций и продуманную стратегию контроля аномального поведения программы.
* Реализовать сериализацию и манипулятор (программный инструмент, который позволяется изменять состояние потока).
* Реализовать аллокатор (программный инструмент, позволяющий управлять распределением памяти под контейнер).
* Описать и реализовать проверку инвариантов контейнера предусловий и постусловий для контейнера.
* Создать, описать и использовать пространства имен.
* Реализовать графический интерфейс, визуализацию приложения.
* Разработать полный набор диаграмм UML.
* Производить рефакторинг.
* Использовать шаблоны проектирования (паттерны) в проекте.
* Разработать документацию по тестированию.
* Разработать требуемые артефакты RUP.
* Описать архитектуру программы.

**приложение В. ПЛАН ПРОЕКТ****А**

**SightsNavigator**

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 16.05.2017 | 1.0 | Первоначальная версия | Аксёнов Г.Н. |

Оглавление

[1. Введение 3](#__RefHeading___Toc508652485)5

[2. План проекта 3](#__RefHeading___Toc508652485)5

* 1. **Введение**

**Цель**

Документ определяет план задач при разработке программной системы.

**Определения и сокращения**

Смотрите артефакт Глоссарий.

* 1. **План проекта**

**Начальный этап**

1. Разработать артефакт Видение.
2. Разработать артефакт Глоссарий.
3. Разработать артефакт Экономическое обоснование
4. Разработать артефакт План проекта.
5. Разработать артефакт Список рисков.

**Этап уточнения**

1. Разработать артефакт Спецификация прецедентов
2. Разработать артефакт Документ Архитектуры ПО.
3. Разработать артефакт Прототип интерфейса.

**Этап разработки**

1. Контейнер Мультиграф

* Реализовать шаблон полиморфного контейнера "Мультиграф", используя STL-контейнеры.
* Создание итератора.
* Реализовать систему классов исключительных ситуаций, применимых к шаблону контейнера.
* Реализовать манипулятор и аллокатор.
* Реализовать инварианты контейнера, пред- и пост- условий.
* Реализовать свое пространство имен.

1. Приложение (логика):

* Реализовать алгоритм сокращения дуг.
* Реализовать функционал UI.
* Реализовать сериализацию.
* Реализовать систему классов исключительных ситуаций, применимых к приложению.

1. Приложение (UI):

* Разработать интерфейс приложения.
* Разработать визуализатор контейнера и функционал связанный с ним
* Реализовать систему интуитивного пользования приложением (справочные и информационные сообщения).

1. Разработать артефакт Документ по компиляции
2. Разработать артефакт План тестирования.
3. Разработать артефакт Тестирование шаблона.
4. Разработать артефакт Тестирование пользовательского интерфейса.

**Этап внедрения.**

* + - 1. Разработать артефакт Оценка проекта.

# **приложение Г. Экономическое обоснование**

# **SightsNavigator**

1. Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 12.03.2018 | 1.0 | Первоначальная версия документа | Аксёнов Г.Н. |
| 24.03.2018 | 1.1 | Дополненная версия документа | Сандин А.В.  Аксёнов Г.Н. |

Оглавление

[1. Введение 3](#__RefHeading___Toc508653175)7

[2. Описание продукта 3](#__RefHeading___Toc508653180)7

[3. Бизнес-контекст 38](#__RefHeading___Toc508653181)

[4. Цели продукта 3](#__RefHeading___Toc508653182)8

[5. Ограничения 3](#__RefHeading___Toc508653183)8

1. **Введение**
   1. Цель

Цель создания данного документа состоит в том, чтобы оценить проект с точки зрения экономической эффективности проекта.

* 1. Контекст

Данный документ содержит экономическое обоснование системы анализа эффективности распределения ресурсов между участниками программных проектов.

* 1. Определения, акронимы и сокращения

См. Глоссарий проекта.

* 1. Ссылки

Отсутствуют.

1. **Описание продукта**

Система «SightsNavigator», предназначена для построения оптимального маршрута между достопримечательностями города Санкт-Петербург. Система подбирает маршрут, исходя из количества введенных денежных средств пользователя и выбранных достопримечательностей.

1. **Бизнес-контекст**

Разрабатываемый продукт предназначен для использования туристами посещающими Санкт-Петербург и позволяет оптимизировать маршрут между выбранными достопримечательностями.

1. **Цели продукта**

Цель создания данного продукта – минимизировать время и деньги, затрачиваемые туристами на передвижение между достопримечательностями.

Предполагаемыми рисками проекта являются:

* риски проектной команды (болезни и т.д.);
* риск несоответствия профессиональной квалификации;
* риски ошибок при проектировании, реализации и тестировании;
* риски выбора неверных технологий и средств разработки.

1. **Ограничения**

Основными ограничениями разрабатываемой системы являются:

* Нацеленность на решение конкретной задачи (распределение ответственности между членами группы).
* Интерфейс системы должен соответствовать стандартам интерфейса пользователя Microsoft Windows.

В дальнейшем возможно развитие системы и дополнение ее различными функциональными возможностями.

# **приложение Д. Список рисков**

# **SightsNavigator**

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 12.03.2018 | 1.0 | Первоначальная версия документа | Сандин А.В. |
| 23.03.2018 | 1.1 | Добавлены дополнительные риски, внесены исправления по полученным замечаниям | Аксёнов Г.Н. |

Оглавление

[1. Введение](#__RefHeading___Toc508652480) 39

[2. Риски](#__RefHeading___Toc508652485) 40

[2.1 Персонал 4](#__RefHeading___Toc508652486)0

[2.2 Профессиональная квалификация 4](#__RefHeading___Toc508652493)0

[2.3 Ошибки проектирования 4](#__RefHeading___Toc508652507)1

[2.4 Ошибки реализации 4](#__RefHeading___Toc508652514)2

[2.5 Ошибки тестирования 4](#__RefHeading___Toc508652521)3

[2.6 Выбор средств разработки 4](#__RefHeading___Toc508652521)4

**Введение**

1.1 Цель

Целью данного документа является определение возможных рисков проекта, а также планирование стратегии управления рисками. Своевременное определение рисков позволяет предотвратить срыв сроков проекта, уменьшение его стоимости, а также обеспечить максимальное соответствие продукта требованиям и необходимый уровень его качества.

1.2 Контекст

Данный документ содержит оценку рисков, их тип, ранжирование по возможной степени и стратегии управления рисками проекта.

1.3 Определения, акронимы и сокращения

См. Глоссарий.

1.4 Ссылки

Отсутствуют.

1. **Риски**

2.1 Персонал

2.1.1 Степень риска

Средняя

2.1.2 Описание

Члены проектной команды могут не выполнить возложенные на них задачи вследствие болезни или других причин.

2.1.3 Влияние

* Невыполнение сроков проекта.
* Увеличение нагрузки на других членов проектной команды.

2.1.4 Показатели появления

Выявление проявления данного риска следует вести путем периодического опроса участников команды о статусе их работы, а также слежением за ходом выполнения задач в системе управления проектом.

2.1.5 Стратегия предотвращения

Составление плана выполнения проекта должно вестись с учетом возможности перераспределения задач между участниками проектной команды.

* + 1. План реагирования

1. При временной невозможности выполнения задач участником проектной команды необходимо проанализировать задачи, решаемые им на текущий момент, и оценить возможность переноса сроков этих задач.

1. В случае полной невозможности выполнения задач участником проекта и при возникновении трудностей изменения сроков задач необходимо выполнить перераспределение ресурсов между проблемными задачами.

2.2 Профессиональная квалификация

2.2.1 Степень риска

Средняя

2.2.2 Описание

Члены проектной команды могут частично или полностью не обладать навыками работы с используемыми технологиями для решения поставленных задач, а также не иметь необходимых знаний в предметной области.

2.2.3 Влияние

* Невыполнение поставленных задач и сроков проекта.
* Увеличение затрат на изучение соответствующих технологий и материала.

2.2.4 Показатели появления

Выявление проявления данного риска проводится путем предварительного опроса участников проекта на предмет владения используемыми технологиями и знаниями в предметной области.

2.2.5 Стратегия предотвращения

При составлении плана выполнения проекта и планировании распределения задач между участниками необходимо выполнить оценку текущей квалификации каждого участника. В том случае, если будут выявлены проблемы, связанные с незнанием используемых технологий, запланировать задачи по их изучению.

2.2.6 План реагирования

* Оценить возможность выполнения определенных задач другими участниками проекта.
* Обеспечить возможности (время, материалы, помощь более квалифицированных участников проекта) для обучения нужным технологиям.

2.3 Ошибки проектирования

2.3.1 Степень риска

Высокая

2.3.2 Описание

На этапе проектирования системы могут быть допущены ошибки при определении подхода проектирования системы, а также выборе вариантов реализации частей системы.

2.3.3 Влияние

* Изменение сроков выполнения проекта из-за возможных проблем с добавлением нового функционала в проект.
* Частичное или полное несоответствие поставленным требованиям.
* Частично или полностью нереализованная функциональность.

2.3.4 Показатели появления

Внесение изменений в проект и добавление нового функционала проходит с высокими трудозатратами.

2.3.5 Стратегия предотвращения

* Прототипирование системы и предоставление прототипа Заказчику.
* Запланировать задачи на возможную доработку архитектуры системы.

2.3.6 План реагирования

Осуществить поиск решений, позволяющих решить возникшие проблемы в рамках существующей архитектуры.

2.4 Ошибки реализации

2.4.1 Степень риска

Высокая

2.4.2 Описание

При реализации проекта участники могут использовать сложные и неоптимальные алгоритмы. Код системы может стать сложным для понимания и доработки.

2.4.3 Влияние

* Несоответствие продукта необходимому качеству.
* Сложности с внесением изменений.
* Сложности с исправлением ошибок и отладкой.

2.4.4 Показатели появления

Исправление ошибок, внесение изменений в проект и добавление нового функционала проходит с высокими трудозатратами.

2.4.5 Стратегия предотвращения

* Выработать единый стиль написания кода.
* Проводить периодический просмотр кода.
* При реализации использовать шаблоны проектирования.

2.4.6 План реагирования

* Выполнить пересмотр и рефакторинг кода.
* Если необходимо, провести оптимизацию алгоритмов.

2.5 Ошибки тестирования

2.5.1 Степень риска

Средняя

2.5.2 Описание

На этапе тестирования могут выявиться критические ошибки, которые не позволят полноценно работать с системой. Кроме того, качество тестирования может не выявить некоторые ошибки, которые повлекут за собой несоответствие продукта поставленным требованиям.

2.5.3 Влияние

* Увеличение затрат на исправление критических ошибок.
* Проявление в конечном продукте не выявленных ошибок.

2.5.4 Показатели появления

* Выявлены ошибки в работе разрабатываемого продукта.
* Выявлено несоответствие продукта поставленным требованиям.

2.5.5 Стратегия предотвращения

* На каждом этапе тестирования формулировать описание функциональности, предоставляемой на тестирование.
* Запланировать достаточное время на выполнение тестирования.

2.5.6 План реагирования

* Своевременно исправить найденные ошибки.
* Уделить время на изучение влияния внесенных изменений на поведение системы.

2.6 Выбор средств разработки

2.6.1 Степень риска

Низкая

2.6.2 Описание

Неверный выбор средств разработки (среда разработки, инструментарий) влияет на качество и время реализации продукта.

2.6.3 Влияние

Невыполнение сроков проекта.

2.6.4 Показатели появления

* Участники проекта тратят много времени на задачи, не связанные с написанием кода (написание скриптов сборки, контроль версий и т.д.).
* Сроки выполнения задач увеличиваются.

2.6.5 Стратегия предотвращения

* Определить требования к средствам разработки.
* Провести анализ решений для разработки и выбрать необходимые средства.

2.6.6 План реагирования

* Выполнить анализ возможности смены средств разработки.
* В случае невозможности изменения средств разработки определить методы решения проблемы в текущей среде, с помощью дополнительных инструментальных средств.

# **приложение Е. Прецедент разработки**

# SightsNavigator

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 30.04.18 | 1.0 | Создание документа | Сандин А.В. |

**Цель**

Данный документ представляет собой описание того, как RUP будет использоваться в проекте “SightsNavigator”.

**Определения и сокращения**

Смотрите артефакт Глоссарий.

**Обзор**

В оставшейся части документа, будут описаны адаптации RUP к данному проекту, артефакты, появляющиеся в процессе прохождения итераций.

**Обзор процесса разработки**

Данный проект будет состоять из начальной стадии, стадии уточнения и трех этапов стадии конструирования (анализ и проектирование, реализация, тестирование). Обзор дизайна и кода будет проводится на ключевых этапах итерации, а проверка на соответствие требованиям в конце каждого этапа.

**Этапы**

**Начальный этап**

Для описания требований мы будем использовать use-cases. Следующие артефакты будут производиться в течение данного процесса:

Таблица 1 - Описание артефактов, создающихся на начальном этапе

|  |  |
| --- | --- |
| **Артефакт** | **Используемый инструмент** |
| Видение (Vision) | Microsoft Word, LibreOffice Writer |
| Глоссарий (Glossary) | Microsoft Word, LibreOffice Writer |
| Список рисков(Risk List) | Microsoft Word, LibreOffice Writer |
| План проекта (Project plan) | Microsoft Word, LibreOffice Writer |
| Экономическое обоснование | Microsoft Word, LibreOffice Writer |

**Этап уточнения**

На основе разработанных требований формируется основа для последующей реализации программного продукта. Используя use-cases производиться создание архитектуры программного продукта, создание прототипа пользовательского интерфейса. Следующие артефакты буду производиться в течение данного процесса:

Таблица 2 - Описание артефактов, создающихся на этапе уточнения

|  |  |
| --- | --- |
| **Артефакт** | **Используемый инструмент** |
| Документ архитектуры ПО (Soft Architecture Document) | Microsoft Word, LibreOffice Writer |
| Спецификация прецедентов (Use-cases specificaion) | Microsoft Word, LibreOffice Writer,  Visual Paradigm CE |
| Протопип интерфейса(GUI Prototype) | Microsoft Word, LibreOffice Writer, Qt Creator |

**Этап реализации**

На этапе реализации идеология процесса управления переключается с разработки интеллектуальной собственности, проходившей на начальном этапе и на этапе уточнения, на изготовление готовых к внедрению продуктов на этапах реализации и внедрения.

Следующие артефакты буду производиться в течение данного процесса:

Таблица 3 - Описание артефактов, создающихся

на этапе реализации

|  |  |
| --- | --- |
| **Артефакт** | **Используемый инструмент** |
| Документ по компиляции | Microsoft Word, LibreOffice Writer |
| План тестирования (Master Test Plan) | Microsoft Word, LibreOffice Writer |
| Тестирование шаблона (Container test cases) | Microsoft Word, LibreOffice Writer, Qt Creator |
| Тестирование пользовательского интерфейса (GUI Test cases) | Microsoft Word, LibreOffice Writer |

**Этап внедрения**

Этап внедрения включает в себя финальное тестирование продукта и внесение незначительных корректив с учетом отзывов пользователей

Следующие артефакты буду производиться в течение данного процесса:

Таблица 4 - Описание артефактов, создающихся

на этапе внедрение

|  |  |
| --- | --- |
| **Артефакт** | **Используемый инструмент** |
| Оценка проекта | Microsoft Word, LibreOffice Writer |

**Конфигурация и управление изменениями**

Рабочий продукт хранит все версии файлов и каталогов проекта. В нем также хранятся все порожденные данные и метаданные, связанные с этими файлами и каталогами

Таблица 5 – Распределение ролей в проекте.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Леончик Александр | Федоров Андрей | Голян Тан | Аксёнов  Григорий | Сандин  Артем |
| Руководитель проекта (Project Leader) | + |  |  |  |  |
| Разработчик архитектуры (Architect) | + | + |  |  |  |
| Проектировщик (Designer) | + |  |  |  | + |
| Эксперт предметной области (Domain Expert) |  |  |  | + |  |
| Разработчик (Developer) | + | + |  |  |  |
| Специалист по пользовательскому интерфейсу (Human Factors Engineer) |  | + |  | + |  |
| Тестировщик (Tester) |  |  | + | + |  |
| Специалист по инструментам (Instrument Specialist) |  |  |  |  | + |
| Аналитик (Analyst) |  |  | + |  | + |

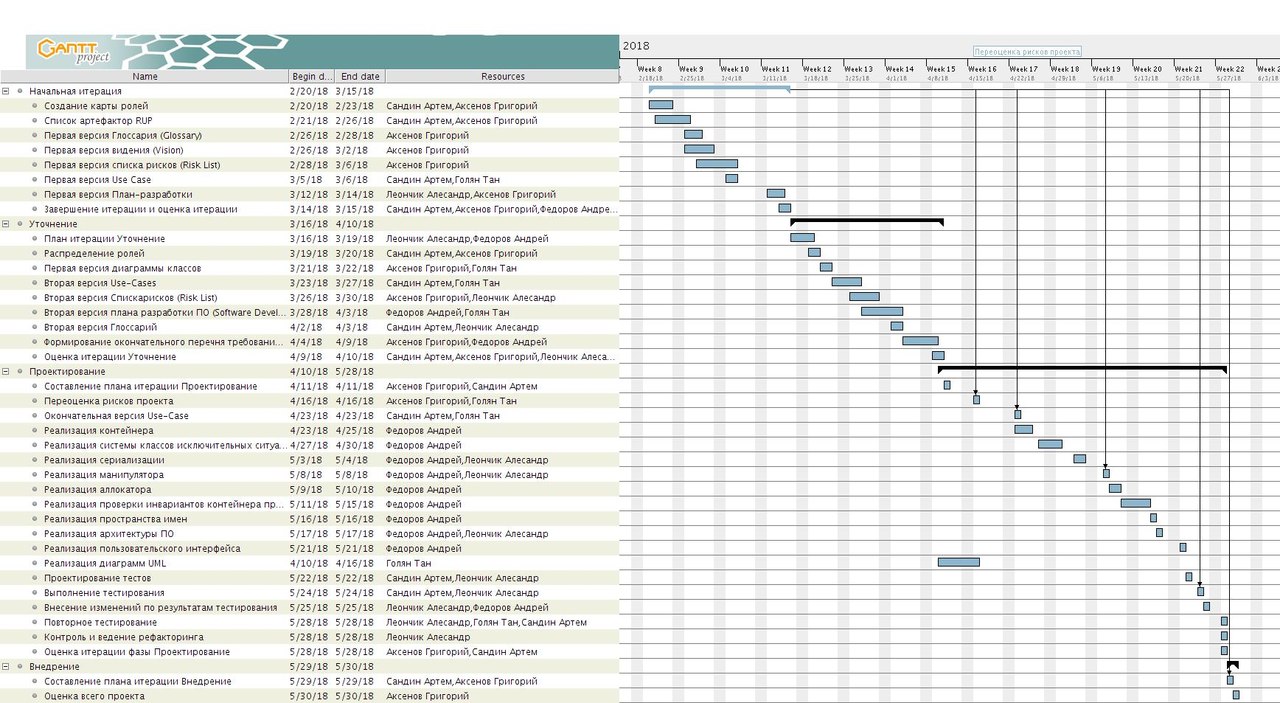


Рис.1 Диаграмма Ганта

**ПРИЛОЖЕНИЕ Ж. СПЕЦИФИКАЦИЯ ПРЕЦЕДЕНТОВ**

1. **SightsNavigator**
2. Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 19.05.17 | 1.0 | Создание и заполнение документа | Аксёнов Г.Н.  Сандин А.В. |
| 23.05.17 | 2.0 | Исправление формулировок | Сандин А.В. |

Оглавление

[1. Введение 4](#__RefHeading___Toc508652480)9

[1.1 Цель](#__RefHeading___Toc508652486) 49

[1.2 Сфера](#__RefHeading___Toc508652486) 49

[1.3 Определение, акронимы и аббревиатуры](#__RefHeading___Toc508652486) 49

[2. Действующие лица 5](#__RefHeading___Toc508652485)0

[3. Прецедент - Сохранение в файл 5](#__RefHeading___Toc508652485)0

[4. Прецедент - Загрузка из файла 5](#__RefHeading___Toc508652485)0

[5. Прецедент — Выбор достопримечательности 5](#__RefHeading___Toc508652485)1

[6. Прецедент — Отмена выбора достопримечательности 5](#__RefHeading___Toc508652485)1

[7. Прецедент — Ввод суммы денег 5](#__RefHeading___Toc508652485)2

[8. Другие прецеденты 5](#__RefHeading___Toc508652485)3

[9. Диаграмма вариантов использования 5](#__RefHeading___Toc508652485)3

# **1. Введение**

## 1.1 Цель

В этом документе представлен подробный обзор вариантов использования, существенно влияющих на архитектуру разрабатываемой системы, а так же UML диаграмма вариантов использования.

## 1.2 Сфера

Данный документ относится к системе MultiDinner, разрабатываемой командой MultiTeam.

## 1.3 Определение, акронимы и аббревиатуры

Определения, акронимы и аббревиатуры определены в артефакте Глоссарий.

# **2. Действующие лица**

Единственным действующим лицом на все прецеденты является Пользователь.

# **3. Прецедент - Сохранение в файл**

## Описание

Данный прецедент имеет место, когда пользователь хочет сериализовать мультиграф и сохранить его.

## Поток событий

## Основной поток

* 1. Пользователь выбирает Сохранить в файл
  2. Система открывает диалоговое окно сохранения файла
  3. Пользователь выбирает путь сохранения и файл
  4. Система создает либо перезаписывает файл по заданному пути

## Предусловие

Граф должен быть не пустым.

## Постусловие

По заданному пути существует файл выбранного системой формата с сериализованным графом.

# **4. Прецедент – Загрузка из файла**

## Описание

Данный прецедент имеет место, когда пользователь хочет загрузить мультиграф и начать работать с ним в системе.

## Поток событий

## Основной поток

1. Пользователь выбирает Загрузка из файла
2. Система открывает диалоговое окно загрузки файла
3. Пользователь выбирает файл
4. Система десериализует мультиграф из выбранного файла и отображает его на интерфейсе.

**Альтернативные потоки**

1. Система не может найти выбранный файл либо файл имеет неверный формат
2. Система выдает ошибку пользователю и просит выбрать другой файл.

**5. Прецедент — Выбор достопримечательности**

## Описание

Данный прецедент имеет место, когда пользователь хочет выбрать достопримечательность из предложенных.

## Поток событий

## Основной поток

* 1. Пользователь нажимает на достопримечательность левой кнопкой мыши.
  2. Система добавляет выбранную достопримечательность в список содержащий названия выбранных достопримечательностей.

## Предусловие

Достопримечательность должна существовать на карте и не быть выбранной.

## Постусловие

Список содержит название выбранной достопримечательности.

1. **Прецедент — Отмена выбора достопримечательности**

## Описание

Данный прецедент имеет место, когда пользователь хочет отменить выбор достопримечательности.

## Поток событий

## Основной поток

1) Пользователь нажимает на достопримечательность левой кнопкой мыши.

2) Система удаляет выбранную достопримечательность из списка содержащего названия выбранных достопримечательностей.

## Предусловие

Достопримечательность должна быть выбрана.

## Постусловие

Список не содержит название выбранной достопримечательности.

1. **Прецедент — Ввод суммы денег**

## Описание

Данный прецедент имеет место, когда пользователь хочет ввести имеющуюся для посещения достопримечательностей сумму денег.

## Поток событий

## Основной поток

1) Пользователь вводит количество денег в рублях.

**Альтернативные потоки**

1) Пользователь ввел некорректные данные. (Корректность заключается в том, что число является целым неотрицательным)

1. Система выдает ошибку пользователю и просит ввести целое неотрицательное число.

## Предусловие

Должно существовать поле ввода.

## Постусловие

Число записывается в поле MaxCost класса MapWidget.

# **8. Другие прецеденты**

## Просмотр достопримечательностей на карте

Пользователь получает изображение достопримечательностей на карте.

**9. Диаграмма Вариантов использования**

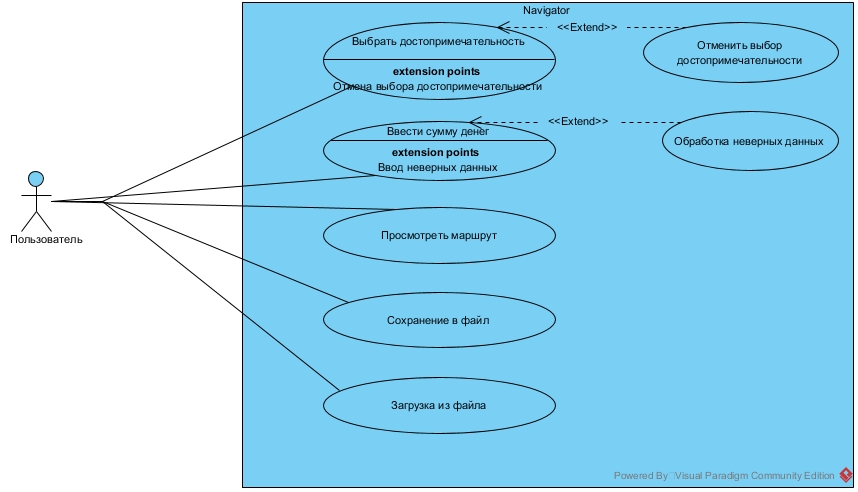
****

Рис. 1. Диаграмма Вариантов использования

**ПРИЛОЖЕНИЕ З. ПРОТОТИП ИНТЕРФЕЙСА**

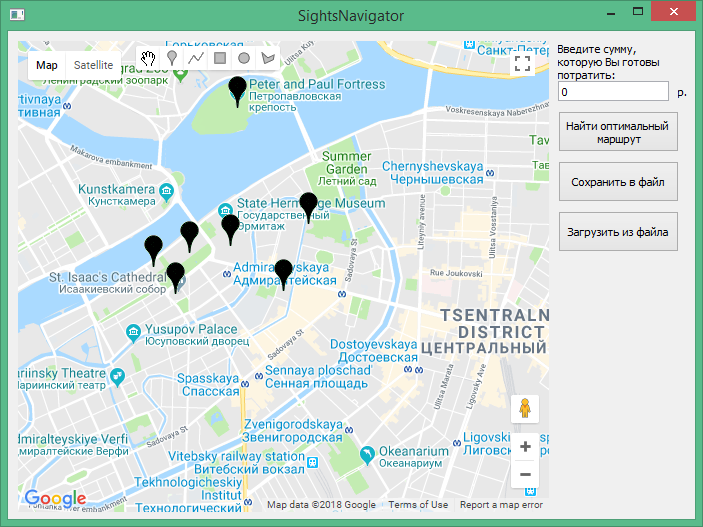
1. **SightsNavigator**
2. Лист регистрации изменений

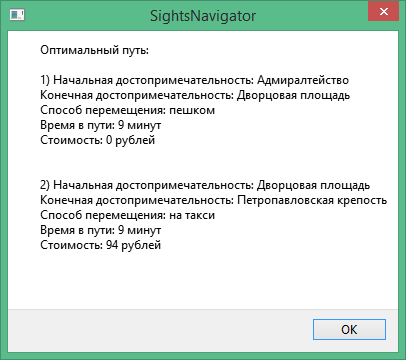
|  |  |  |  |
| --- | --- | --- | --- |
| 1. **Дата** | 1. **Версия** | 1. **Описание** | 1. **Автор** |
| 1. 07.03.2017 | 1. 1.0 | 1. Первоначальная версия | 1. Аксёнов Г.Н. |

1. Оглавление

[1. Введение 5](#__RefHeading___Toc508652480)4

[2. Обзор прототипа пользовательского интерфейса 5](#__RefHeading___Toc508652480)4

1. **Введение**
2. Данный документ представляет собой описание того, как должен выглядеть пользовательский интерфейс в приложении.
3. **Обзор прототипа пользовательского интерфейса**
4. 
5. Рис. 1 – Главное окно приложения



1. Рис. 2 – Окно с выведенным оптимальным маршрутом

**ПРИЛОЖЕНИЕ И. ДОКУМЕНТ АРХИТЕКТУРЫ ПО**

1. **SightsNavigator**
2. Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 12.04.2018 | 1.0 | Первоначальная версия документа | Голян Т.;  Аксёнов Г.Н.;  Сандин А.В. |
| 04.05.2018 | 1.1 | Внесение изменений | Голян Т.;  Аксёнов Г.Н.;  Сандин А.В. |
| 20.05.2018 | 1.2 | Внесение изменений | Федоров А.М.;  Леончик А.В.; |

Оглавление

[1. Введение 57](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793960)

[1.1 Цель 57](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc508652486)

[1.2 Контекст 58](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc508652486)

[1.3 Определения, акронимы, сокращения 58](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc508652486)

[2. Архитектурные цели и ограничения 58](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793964)

[3. Представление вариантов использования и взаимодействия 59](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793966)

[4. Логическое представление 60](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793966)

[5. Представление структуры 61](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793967)

[6. Представление процесса 6](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793968)1

[7. Представление развертывания 62](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793969)

[8. Представление поведения](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793970) 62

[9. Размер и работа 6](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793971)5

[10. Качество 6](file:///C:\\Users\\Andrei\\Desktop\\MDP_1(1).docx" \l "__RefHeading___Toc514793971)5

# **1. Введение**

При разработке программного продукта задачей разработчиков является проектирование архитектуры приложения.

## 1.1. Цель

Документ архитектуры ПО призван провести краткий обзор разрабатываемой системы и описать различные архитектурные решения, принятые при проектировании ПО.

## 1.2. Контекст

Документ архитектуры ПО предоставляет обзор архитектуры приложения «SightsNavigator». «SightsNavigator» разработан командой в рамках курса «Объектно-ориентированные технологии разработки программного обеспечения».

## 1.3. Определения, акронимы, сокращения

См. глоссарий проекта.

# **2. Архитектурные цели и ограничения**

Основной целью разработки является процесс создания рабочего продукта в соответствии с методологией разработки RUP. К реализации продукт имеет следующие требования и ограничения:

1) Наличие аллокатора.

2) Наличие итератора.

3) Наличие системы классов исключительных ситуаций.

4) Возможность сериализации.

5) Наличие манипулятора.

6) Наличие инварианта и проверки предусловия, постусловия.

7) Пространство имен.

Этот документ описывает архитектуру программной системы с помощью UML-диаграмм:

* Диаграмма классов.
* Диаграмма объектов.
* Диаграмма компонентов.
* Диаграмма состояний.
* Диаграмма деятельности.
* Диаграмма коммуникации.
* Диаграмма последовательности.
* Диаграмма развёртывания.

# **3. Представление вариантов использования и взаимодействия**

Use-case диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне, представлена на рисунке 1.

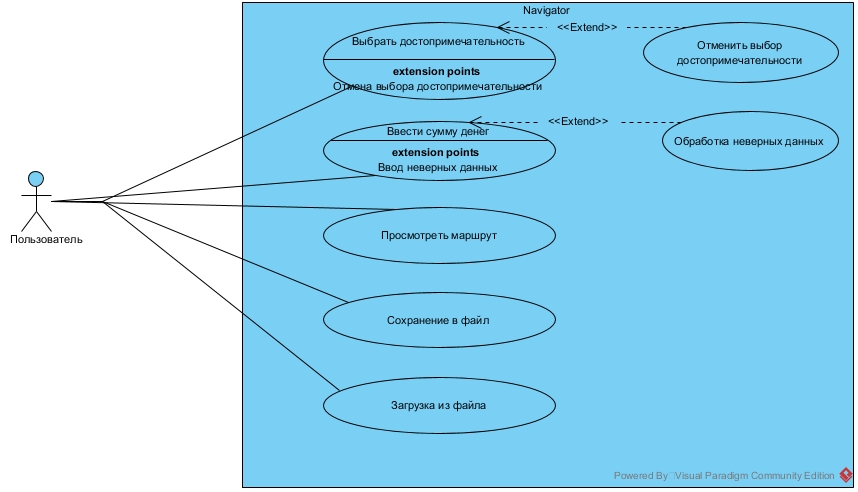
****

Рисунок 1 – Use-case диаграмма.

Диаграмма последовательности отражает   взаимодействие объектов системы во времени, а также обмена сообщениями между ними, представлена на рисунке 2.

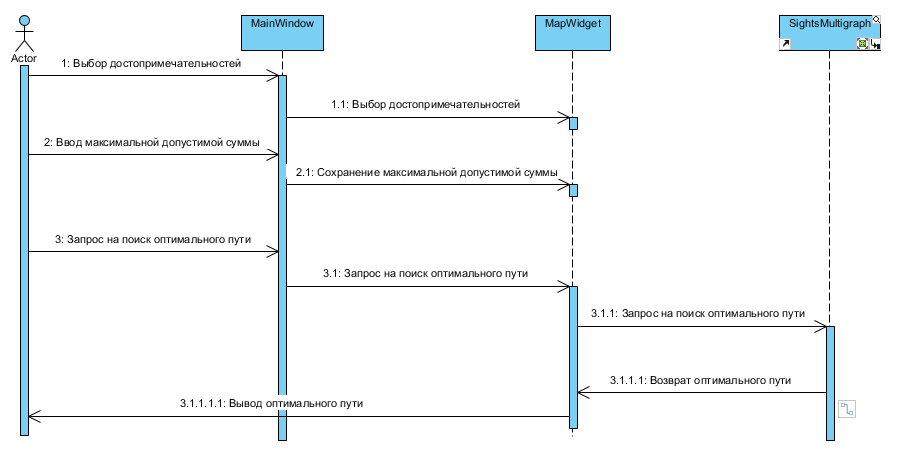


Рисунок 2 – Диаграмма последовательности

# **4. Логическое представление**

Для представления архитектуры используется диаграмма компонентов. Диаграмма компонентов показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п. Диаграмма компонентов представлена на рисунке 3.

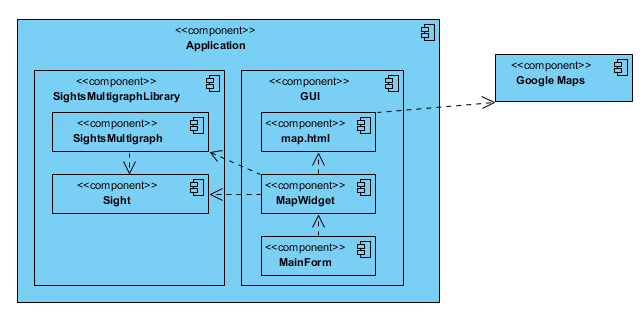


Рисунок 3 – Диаграмма компонентов.

# **5. Представление структуры**

Для демонстрации классов системы, их атрибутов, методов и взаимосвязей между ними используется диаграмма классов. Диаграмма классов представлена на рисунке 4.

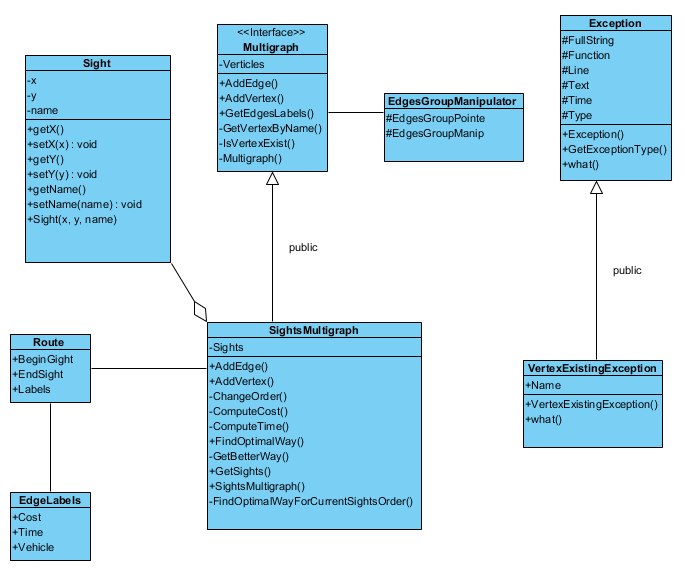


Рисунок 4 – Диаграмма классов.

# **6. Представление процесса**

Для представления процесса работы приложения используется диаграмма объектов, которая демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием связей между объектами. Диаграмма объектов представлена на рисунке 5.

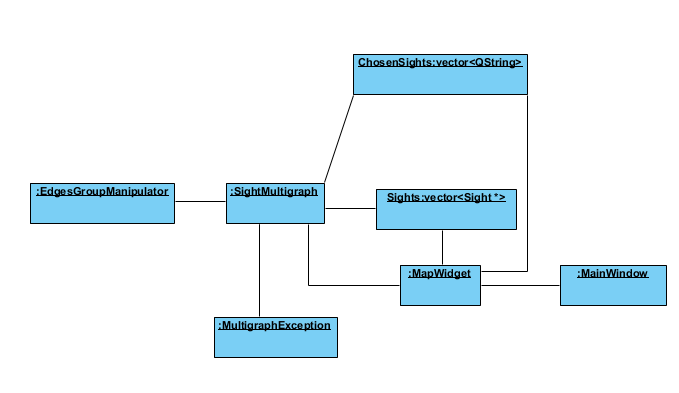


Рисунок 5 – Диаграмма объектов.

# **7. Представление развертывания**

Программное обеспечение будет развертываться на ОС Microsoft Windows 7 и более старших версиях данной операционной системы. Диаграмма развёртывания представлена на рисунке 6.

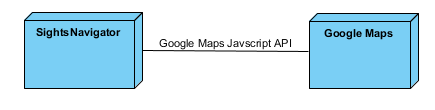


Рисунок 6 – Диаграмма развёртывания.

# **8. Представление поведения**

Основное представление поведения показано на диаграммах действия, состояний и коммуникации. В частности, на ней изображаются участвующие во взаимодействии объекты и действия. На рисунке 7 представлена диаграммы деятельности, которая показывает алгоритм поиска оптимального пути.

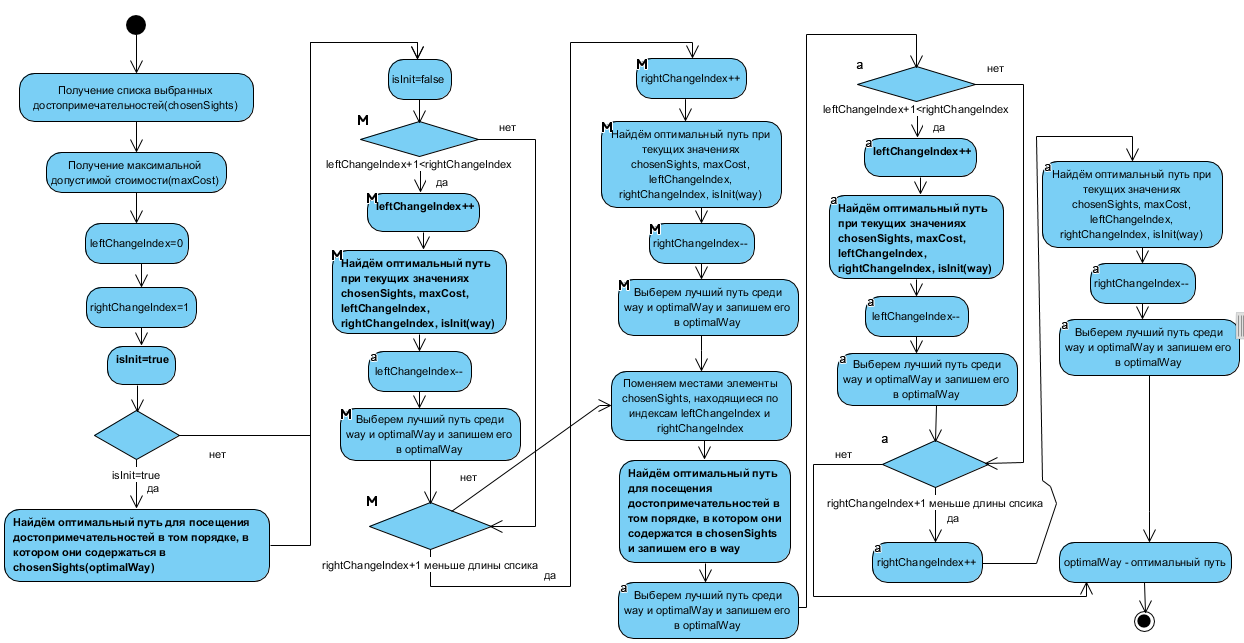


Рисунок 7 – Диаграмма действия.

На рисунке 8 представлена диаграмма состояний. Диаграмма коммуникации приведена на рисунке 9.

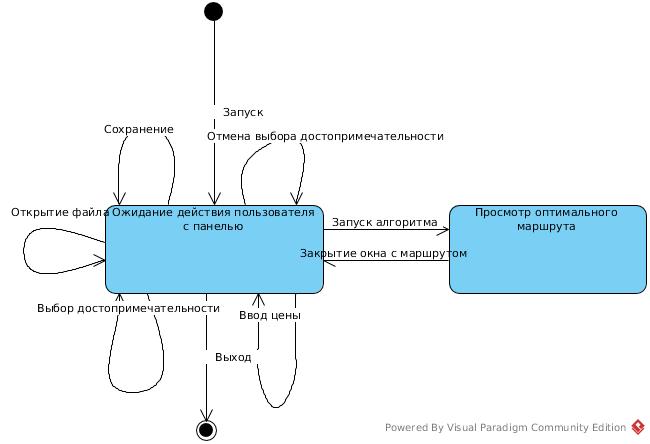


Рисунок 8 – Диаграмма состояний.

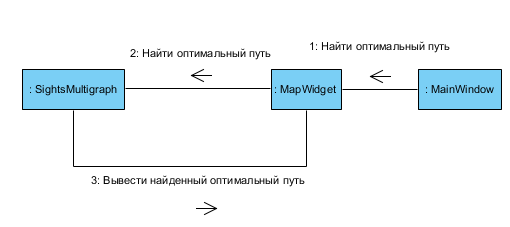


Рисунок 9 – Диаграмма коммуникации.

* 1. Система реализована как единая исполняемая программа. Саму систему можно разделить на 3 слоя: слой интерфейса, слой контейнера и внешний слой.

1. **Слой интерфейса -** набор классов, реализующих интерфейс и функции активных элементов.
2. **Слой контейнера -** набор шаблонных классов, реализующих функции контейнера **MultiGraph.**
3. **Внешний слой -** набор классов и возможностей, полученных из внешних библиотек.

При разработке использовались следующие шаблоны проектирования:

* Итератор.

Назначение

Итератор – поведенческий шаблон проектирования. Представляет собой объект, позволяющий получить последовательный доступ к элементам объекта-агрегата без использования описаний каждого из агрегированных объектов. Например, такие элементы как дерево, связанный список, хэш-таблица и массив могут быть пролистаны (и модифицированы) с помощью объекта Итератор. Перебор элементов выполняется объектом итератора, а не самой коллекцией. Это упрощает интерфейс и реализацию коллекции, а также способствует более логичному разделению обязанностей.

Применение

1. Пакет MultiGraph реализует контейнер Мультиграф(класс Multigraph), который является шаблоном, содержащим в себе список вершин. Для его разработки используются такие шаблоны проектирования как Пул объектов и Итератор. Пул объектов представлен классом Allocator, предоставляющим интерфейс для выделения и освобождения памяти для объектов контейнера. Итератор представлен классом **VertexIterator** и предоставляет возможности для получения доступа к элементам контейнера и используется для обхода графа по вершинам.

# **8. Размер и работа**

Размеры проекта ограничены учебным курсом. Временные рамки сужены, поэтому проект имеет достаточно большие размеры по сравнению со стандартными курсовыми проектами, но достаточно малые размеры относительно реальных проектов.

# **9. Качество**

Целью разработки является освоение самого процесса разработки, поэтому к качеству предъявляются минимально возможные требования, с условием выполнения дополнительных функций, обеспечения надёжности работы программы.

**ПРИЛОЖЕНИЕ К. КОМПИЛЯЦИЯ**

1. **SightsNavigator**

Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 10.05.2017 | 1.0 | Первоначальная версия | Сандин А.В.  Голян Т. |

Оглавление

[1. Введение 6](#__RefHeading___Toc514793960)6

[2. Обзор 6](#__RefHeading___Toc514793964)6

[3. План компоновки 6](#__RefHeading___Toc514793965)7

**1. Введение**

Данный документ описывает рабочую версию программной системы, демонстрирующей подмножество функций, предоставляемых программной системой. Компоновка (обычно в процессе компиляции и компоновки исходного кода) включает один или несколько элементов реализации (часто исполняемых), состоящих из других элементов.

**2. Обзор**

Программная система состоит из нескольких взаимосвязанных частей: Контейнер, Графический Пользовательский Интерфейс, Алгоритм.

На этапе компиляции используются:

1) Контейнер, обеспечивающий интерфейс необходимой структуры данных (в данном случае – двудольный граф) для Алгоритма.

2) Алгоритм, реализующий функциональную часть программной системы. Для корректной работы Алгоритма необходим Контейнер, а также класс, обеспечивающий передачу данных в графического пользовательский интерфейс.

3) Графический пользовательский интерфейс, реализующий отображение итогов работы Алгоритма в текущей модели, а также взаимодействующий с базой данных.

**3. План компоновки**

При изменении частей программы предлагается следующий план компоновки:

1. Изменение Контейнера – соответствующее изменение Алгоритма.
2. Изменение Алгоритма – соответствующее изменение Графического Пользовательского Интерфейса.
3. Изменение Графического Пользовательского Интерфейса - соответствующее изменение Алгоритма.

ПРИЛОЖЕНИЕ Л. План тестирования

1. **SightsNavigator**

# Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 12.05.2017 | 1.0 | Первоначальная версия | Сандин А.В.  Аксенов Г.Н.  Леончик А.В. |

# Оглавление

[1. Введение 6](#__RefHeading___Toc508652480)8

[1.1 Цель 6](#__RefHeading___Toc508652486)8

[1.2 Обзор 6](#__RefHeading___Toc508652486)9

[2. Оценка задачи 6](#__RefHeading___Toc508652480)9

[2.1 Предпосылки 6](#__RefHeading___Toc508652486)9

[2.2 Оценка 6](#__RefHeading___Toc508652486)9

[3. Мотивация тестирования 6](#__RefHeading___Toc508652485)9

[4. Цель тестирования элементов 6](#__RefHeading___Toc508652485)9

[5. Описание планируемых тестов 6](#__RefHeading___Toc508652485)9

[6. Типы тестирования](#__RefHeading___Toc508652485) 70

[6.1 Функциональное тестирование](#__RefHeading___Toc508652486) 70

[6.2 Модульное тестирование](#__RefHeading___Toc508652486) 70

[6.3 Тестирование пользовательского интерфейса 7](#__RefHeading___Toc508652486)0

[7. Ресурсы 7](#__RefHeading___Toc508652480)1

[7.1 Система 7](#__RefHeading___Toc508652486)1

# **1. Введение**

## 1.1 Цель

Целью данного документа является описание подхода к тестированию готового продукта.

Включает следующие цели:

* Определить компоненты для тестирования;
* Перечислить рекомендуемые требования к проведению тестирования;
* Описать рекомендуемые стратегии тестирования;
* Определить требуемые ресурсы.

## 1.2 Обзор

В следующих разделах описаны предпосылки, ресурсы, основные группы тестирования, цели и технологии.

# **2. Оценка задачи**

## 2.1 Предпосылки

В процессе разработки ПС необходимо сформулировать план тестирования для дальнейшего использования. Данный документ поможет проверить ПС на соответствие требованиям при помощи тестов.

## 2.2 Оценка

Задачей данного документа является определения ключевых моментов тестирования с целью:

* Выявления ошибок ПС;
* Выявление важных проблем и рисков;
* Формирование представления и оценки предполагаемых рисков;
* Проверки спецификации;
* Формирование представления о качестве продукта.

# **3. Мотивация тестирования**

Тестирование мотивировано рядом вещей, необходимых для проверки, а именно: функциональными и нефункциональными требованиями, вариантами использования, элементами дизайна, а так же запросами на изменение.

# **4. Цель тестирования элементов**

Целью тестирования элементов ПС является выявления и исправления ошибок для получения более качественного продукта, удовлетворяющего требованиям заказчика.

# **5. Описание планируемых тестов**

В ходе тестирования ПС планируется использовать следующие группы тестов:

* *Функциональное тестирование*, которое позволит проверить на соответствие требованиям, описанным в документации;
* *Модульное тестирование*, необходимое для тестирования шаблона контейнера и алгоритма;
* *Тестирование пользовательского интерфейса* для тестирования приложения.

# **6. Типы тестирования**

## 6.1 Функциональное тестирование

Цель тестирования*:* определить, соответствует ли продукт требованиям, указанным в документации.

Технология тестирования*:* на каждой итерации проверяется соответствие реализованных функциональных возможностей ПС в соответствии с документацией.

Критерий завершенности*:* соответствие всех требований, заявленных в спецификации к ПС, заявленные на данную итерацию.

Примечания*:* отсутствуют.

## 6.2 Модульное тестирование

Цель тестирования*:* определить соответствие функциональным требованиям отдельных модулей ПС.

Технология тестирования*:* рассматриваются два модуля ПС - шаблон контейнера и работа алгоритма сокращения дуг в мультиграфе. Для каждого модуля определяется из спецификации набор требований, определяющих тесты.

Критерий завершенности*:* Выполнение всех тестовых случаев, определенных для конкретного модуля.

Примечания*:* отсутствуют.

## 6.3 Тестирование пользовательского интерфейса

Цель тестирования*:* определить наибольшее количество ошибок при работе с пользовательским интерфейсом программы и исправить их.

Технология тестирования:для тестирования необходимо использовать ручное тестирование графических форм. Тестовые случаи определяются в зависимости от спецификации требований и вариантов использования.

Критерий завершенности*:* Выполнение всех тестовых случаев.

Примечания: отсутствуют.

# **7. Ресурсы**

# **7.1 Система**

Тестирование проводится на персональных компьютерах и планшетах, соответствующих необходимым требованиям ПС.

ПРИЛОЖЕНИЕ М. Тестирования шаблона «Мультиграф»

1. **SightsNavigator**

# Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 12.05.2017 | 1.0 | Первоначальная версия | Сандин А.В.  Леончик А.В. |

# Оглавление

[1. Введение 7](#__RefHeading___Toc508652480)2

[1.1 Цель 7](#__RefHeading___Toc508652486)2

[2. Описание функциональных особенностей 7](#__RefHeading___Toc508652480)2

[3. Тест-кейсы функциональных возможностей 7](#__RefHeading___Toc508652485)3

# **1. Введение**

## 1.1 Цель

Целью данного документа является описание тестирования функциональных возможностей шаблона «Мультиграф», который был реализован в ходе разработки ПС.

## 2. Описание функциональных особенностей

Список функциональных возможностей контейнера:

1. Создать вершину - создает экземпляр класса Vertex и добавляет в объект класса контейнера.
2. Проверка существования вершины — проверяет, что вершина с данным именем существует.
3. Получение меток ребер — получает метки всех ребер между двумя выбранными вершинами.
4. Добавить ребро - создает экземпляр класса Edge и добавляет в объект класса контейнера.
5. Добавление уже существующей вершины — добавить вершину с именем, которая принадлежит другой вершине.
6. Поиск оптимального маршрута — выполнить поиск наиболее быстрого маршрута не превышающего введенную пользователем стоимость.
7. Очистка контейнера - очищает контейнер от всех вершин и ребер.
8. Сериализация контейнера - вывод контейнера в поток.

# **3. Тест-кейсы функциональных возможностей**

# **Создание**

Шаги

1. Создается мультиграф, с помощью конструктора по умолчанию.

Ожидаемый результат: мультиграф пуст.

***Создание вершины***

Шаги:

1. Вызывается метод AddVertex класса Multigraph.
2. Проверяется равенство данных переданных в метод, с данными, записанными в вершине.

Ожидаемый результат: равенство данных из п. 2.

***Добавление ребра***

Шаги:

1. Создаются две вершины с помощью метода AddVertex класса Multigraph.
2. Вызывается метод AddEdge класса Multigraph для созданных в пункте 1 вершин.
3. Проверяется наличие созданного ребра среди исходящих рёбер для вершин из пункта 1.

Ожидаемый результат: у вершин появилось добавленное исходящее ребро.

***Проверка существования вершины***

Шаги:

1) Создается вершина с помошью метода AddVertex класса Multigraph

2) Вызывается метод isVertexExist класса Multigraph для имени вершины, созданной в пункте 1.

3) Проверяется, что выбранный метод вернул значение true.

Ожидаемый результат: проверка прошла успешно.

***Получение меток ребер.***

Шаги:

1) Создаются две вершины, два ребра между эти вершинами.

2) Вызывается метод getEdgesLabels класса Multigraph для вершин, созданных в пункте 1

Ожидаемый результат: метод вернул список меток ребер из пункта 1

***Добавление уже существующей вершины***

Шаги:

1) Создается вершина.

2) Создается вершина с именем вершины из пункта 1.

3) Проверяется, что возникла исключительная ситуация.

Ожидаемый результат: возникновение исключительной ситуации

***Поиск оптимального маршрута***

1) Создаем мультиграф.

2) Создаем список выбранных вершин.

3) Вызываем метод findOptimalWay класса SightsMultigraph.

Ожидаемый результат: метод вернул оптимальный путь.

***Очистка контейнера***

Шаги

1. Создается мультиграф.
2. В мультиграф добавляются две вершины, с уникальными именами.
3. В мультиграф добавляются две дуги, связывающие вершины, добавленные в предыдущем пункте.
4. Мультиграф очищается.

Ожидаемый результат: мультиграф пуст.

***Сериализация***

Шаги

1. Создается мультиграф.
2. В мультиграф добавляются две вершины, с уникальными именами.
3. В мультиграф добавляются две дуги, связывающие вершины, добавленные в предыдущем пункте.
4. Мультиграф сериализуется в строковый поток.
5. Мультиграф очищается.
6. Мультиграф востанавливается из строкового потока.

Ожидаемый результат: мультиграф имеет две вершины и две дуги между ними.

ПРИЛОЖЕНИЕ Н. Тестирования графического интерфейса

1. **SightsNavigator**

# Лист регистрации изменений

|  |  |  |  |
| --- | --- | --- | --- |
| **Дата** | **Версия** | **Описание** | **Автор** |
| 12.05.2017 | 1.0 | Первоначальная версия | Аксенов Г.В.  Федоров А.М. |

# Оглавление

[1. Введение 7](#__RefHeading___Toc508652480)6

[1.1 Цель 7](#__RefHeading___Toc508652486)6

[2. Описание функциональных возможностей 7](#__RefHeading___Toc508652480)6

[2.1 Главное окно 7](#__RefHeading___Toc508652486)6

[2.2 Контрольная панель 7](#__RefHeading___Toc508652486)7

[3. Тест-кейсы функциональных возможностей 7](#__RefHeading___Toc508652485)7

# **1. Введение**

## 1.1 Цель

Целью данного документа является описание подхода к тестированию функциональных возможностей приложения.

# **2. Описание функциональных возможностей**

## 2.1 Главное окно

Главное окно состоит из следующих компонентов:

1. Рабочая область – предоставляет пользователю карту, в котором возможно выбирать предложенные достопримечательности и изменять масштаб карты для удобства восприятия.
2. Контрольная панель — предаставляет собой панель, на которой находится поле для ввода количества имеющихся денег и кнопки для построения оптимального маршрута, сохранения оптимального пути в файл и извлечения оптимального пути из файла.

## 2.2 Контрольная панель

Контрольная панель состоит из следующих компонентов:

1. Поле «Введите сумму которую Вы готовы потратить:» - поле, позволяющее пользователю вводить количество денег, которое он готов потратить на передвижение между достопримечательностями.
2. Кнопка «Найти оптимальный маршрут» - кнопка при нажатии которой запускается алгоритм поиска оптимального пути.
3. Кнопка «Сохранить в файл» - кнопка, предоставляющая пользователю сохранить оптимальный маршрут для дальнейшего использования.
4. Кнопка «Загрузить из файла» - кнопка, предоставляющая возможность открытия ранее сохраненного файла с оптимальным маршрутом.

# **3. Тест-кейсы функциональных возможностей**

## Выбор достопримечательности

Шаги:

1. Создать / открыть модуль.
2. Нажатие на невыбранную достопримечательность левой кнопкой мыши.

Ожидаемый результат: достопримечательность отобразилась на карте как выбранная.

## Отмена выбора достопримечательности

Шаги:

1. Создать / открыть модуль.
2. Нажатие на выбранную достопримечательность.

Ожидаемый результат: достопримечательность отобразилась на карте как не выбранная.

## Выполнение алгоритма

Шаги:

1. Создать / открыть модуль.
2. Создание или открытие карты с множеством выбранных достопримечательностей.
3. Ввод денег.
4. Нажатие на кнопку нахождения оптимального пути.

Ожидаемый результат: будет построен оптимальный маршрут с указанием необходимого количества денег.

## Сохранение мультиграфа.

Шаги:

1. Создать / открыть модуль.
2. Нажатие на кнопку сохранения.
3. Ввод названия файла.
4. Ожидаемый результат: создан файл с заданным названием.

## Открытие мультиграфа.

Шаги:

1. Создать / открыть модуль.
2. Нажатие на кнопку открыть.
3. Выбор файла из сохранённых ранее.

Ожидаемый результат: мультиграф, содержащийся в файле отображён на карте.