МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

Кафедра ИС

ОТЧЕТ ПО ТЕХНИЧЕСКИМ ТРЕБОВАНИЯМ К СЕРВИСУ «Карта Маршрутов»

Выполнили студенты группы № 2372:
Соколовский В.Д., Гечис В.Р. Мельникова М.А.

Проверил: проф. Водяхо А. И.

Санкт-Петербург
2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Полное наименование проекта:

Картографический сервис «Карта Маршрутов».

Сокращенное наименование проекта:

КС «Карта Маршрутов».

Описание проекта:

Сервис «Карта Маршрутов» предназначен для анализа окружения и нахождения кратчайшего пути.

Цели проекта:

Целью разрабатываемого сервиса является предоставление пользователю возможности быстро и удобно планировать маршруты.

Аудитория пользователя:

Сервис разрабатывается для всех пользователей ПК.

Локализация проекта:

Приложение будет поддерживать английский язык.

ГЛОССАРИЙ

№	Термины	Определения
1	Use case	Описание поведения системы взаимодействия с внешней средой, например, как система реагирует на запросы пользователей при построении маршрутов.
2	Интерфейс	Совокупность возможностей, методов и способов взаимодействия пользователя с сервисом, включая пользовательский интерфейс и управляющие элементы.
3	База данных	Совокупность организованных данных, описывающих характеристики картографической информации, маршрутов, объектов на карте и их взаимосвязи для обеспечения функционирования и поддержки сервиса "Карта Маршрутов".
4	Администратор	Пользователь, наделённый всеми возможными полномочиями д
5	Пользователь	Сущность, которая обладает ограниченными полномочиями для использования приложения

Таблица 1 - Глоссарий

ПОЛЬЗОВАТЕЛЬСКИЕ ТРЕБОВАНИЯ

Пользовательские требования - определяют набор пользовательских задач, которые должна решать программа, а также сценарии (use case) их решения в системе.

Набор пользовательских задач:

- В главном окне приложения должен быть представлен список из кнопок: изменений карты и нахождения кратчайшего пути;
- При нажатии кнопки «Update Map» должно выводить список xml-файлов, которые можно использовать;
- При нажатии кнопки «Add» можно будет нарисовать препятствие;
- При нажатии кнопки «Edit» можно будет изменять карту;
- При нажатии кнопки «Remove» можно будет удалить препятствие;
- При нажатии кнопки «Start» и «Finish» можно будет задать начальную точку и конечную, после чего будет найдет оптимальный маршрут;
- Приложение должно работать на Windows/Linux;
- Быстродействие приложения.

Требования руководителя проекта и проектной команды:

1. Требуется соответствующая документация с требованиями потенциальных пользователей, а также требованиями менеджера по продажам;
2. Для выполнения проекта требуется среда разработки Qt Creator.

Требования менеджера по продажам:

Приложение должно быть интуитивно понятно для использования.

Требования спонсора проекта:

Приложение должно окупить себя и принести прибыль в 100% стоимости.

Заинтересованные лица и пользователи

Группа заинтересованных лиц	Основные требования
Администраторы	• Управление базой данных с картографической информацией • Мониторинг производительности и безопасности системы • Обновление и поддержка программного обеспечения • Поддержка пользователей и решение проблем
Пользователи	• Интуитивный интерфейс для поиска и создания маршрутов • Быстрое и точное построение маршрутов • Возможность просмотра информации о местоположении, достопримечательностей, транспортных средств и т. д.

Таблица 2 – Заинтересованные лица

Системный сценарий (use case) использования приложения показан на рисунке 1.

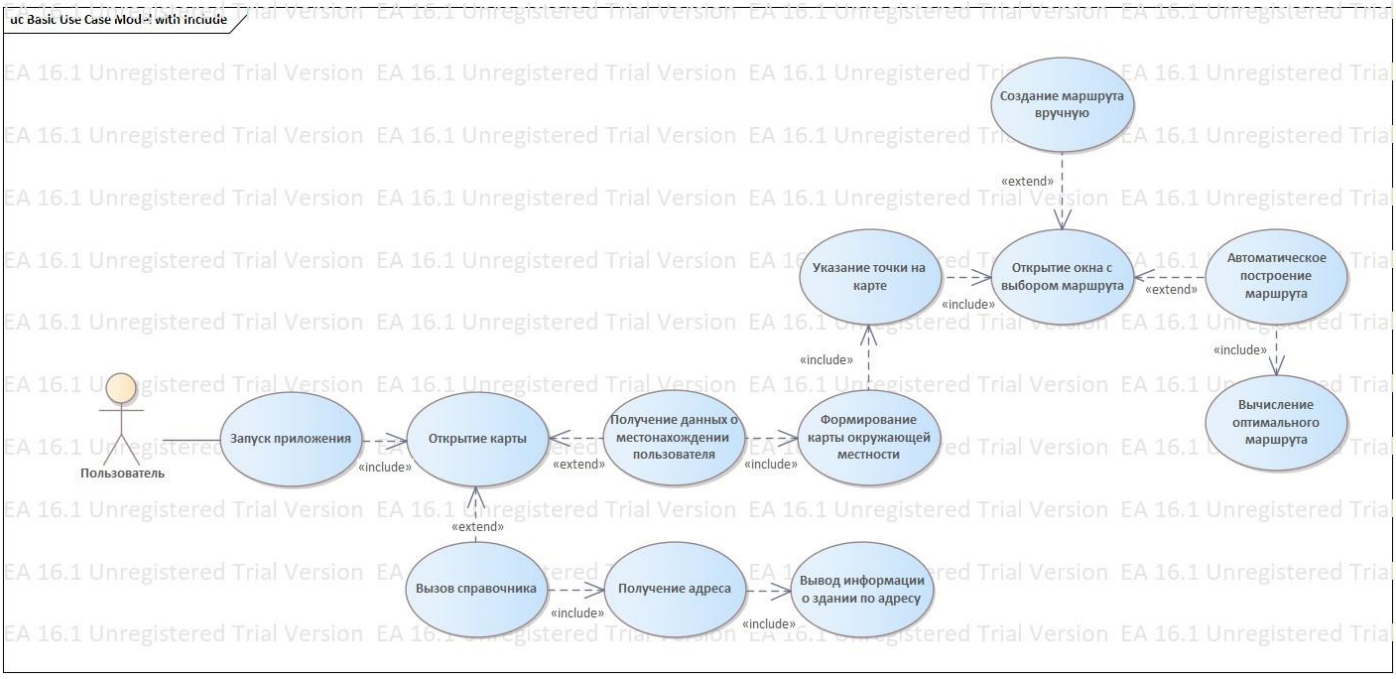

Рисунок 1 - Системный сценарий (use case)

- В таблицах 3,5,7 описаны варианты использования системы с кратким описанием и условиями.
- В таблицах 4,6,8 описаны последовательности действий, приводящие к успешному выполнению соответствующего варианта использования.

Вариант использования	Проложение автоматического маршрута на карте
Актер	• Пользователь
Предусловия	• Пользователь должен запустить приложение и быть авторизированным в системе
Краткое описание	• Пользователь устанавливает стартовую и конечную точку с помощью адреса или точки на карте, сервис предоставляет возможность проложение автоматического маршрута.
Постусловия	• Маршрут представлен пользователю на экране

Таблица 3 – Проложение автоматического маршрута на карте

Ход действий для проложения автоматического маршрута на карте

Действия актора	**Отклик системы**
1. Пользователь вводит свой адрес	• Система отображает стартовую точку
2. Пользователь вводит конечную точку	• Система вычисляет оптимальный путь и отображает маршрут на экране

Таблица 4 *– Ход действий для проложения автоматического маршрута на карте*

Вариант использования	Проложение маршрута на карте вручную

Актер	• Пользователь
Предусловия	• Пользователь должен запустить приложение и быть авторизированным в системе
Краткое описание	• Пользователь устанавливает стартовую и конечную точку с помощью адреса или точки на карте, сервис предоставляет возможность проложения пути вручную
Постусловия	• Маршрут представлен пользователю на экране

Таблица 5 – Проложение маршрута на карте вручную

Ход действий для проложения ручного пути на карте

Действия актора	**Отклик системы**
1. Пользователь вводит свой адрес	• Система отображает стартовую точку
2. Пользователь проводит на карте непрерывную линию своего пути	• Система корректирует линию пути с учетом препятствий. Конец линии принимается на конечную точку маршрута

Таблица 6 – Ход действий для проложения маршрута на карте вручную

Вариант использования	Получение информации по адресу
Актер	• Пользователь
Предусловия	• Пользователь должен запустить приложение и быть авторизированным в системе
Краткое описание	• Пользователь открывает приложение и вызывает встроенный справочник, вводит адрес, справочник получает адрес и выводит информацию о здании по нему.
Постусловия	• Информация выведена на экране

Таблица 7 – Получение информации по адресу

Действия актора	**Отклик системы**
1. Пользователь вызывает справочник	• Система отображает открывает интерфейс справочника на экране
2. Пользователь вводит адрес в поисковую строку справочника	• Система получает информацию по заданному ключу адреса и выводит информацию на экран

Таблица 8 – Ход действий для получения информации по адресу

СИСТЕМНЫЕ ТРЕБОВАНИЯ

Системные требования указывают разработчику какие методы и свойства должны иметься у объектов

Системные требования представлены в таблице 9.

Объект	Методы	Свойства
TInterface	Открыть карту, поменять карту, добавить препятствие, редактировать препятствие, удалить препятствие, точка старта, точка завершения, вывод результатов и сохранение в файл	void *updateMap*(); void *addObstacle*(); void *editObstacle*(); void *removeObstacle*(); void *setStartPoint*(); void *setFinishPoint*(); void *resultAndSave*();
TMap	Нахождение оптимального маршрута, добавление препятствия	QVector<TObstacle> obstacles;
TObstacle	Получение позиции препятствия и получение индекса тупика	QPointF m_pos; int m_impasseIndex;
TParsing	Получение пути, парсинг файла, обработка ошибок	TMap* map; string **getPath**(); short **parsingFile**(*const* string&); void **error**(*const* short&);
TRoute	Получение длины, точек, времени	QPolygonF m_points; qreal m_length; qreal m_time;

Таблица 9 – Спецификация классов

АЛЬТЕРНАТИВНЫЕ ПОДХОДЫ РЕАЛИЗАЦИИ

Приложение разрабатывается в QT Creator, на языке C++. Выбран QT Creator, потому что является более современным и удобным для данной цели.

Преимущества:

- Удобное межпроцессное взаимодействие (сигналы/слоты);
- Так как QT - полноценный фреймворк, то он обладает всем, что нужно - контейнеры, строки, алгоритмы, GUI и т.д;
- Низкий порог вхождения;
- Переносимость на уровне исходного кода. (Windows, Linux, Mac, QNX, Android, IOS, WinRT);
- Активно развивается;
- Обладает неплохой IDE, специфицированной под QT;
- Обилие информации и материалов как на русском, английском и многих других языках;
- Множество примеров использования.

Недостатки:

- Большой вес приложений. (библиотеки, в зависимости от того, что используется, будут весить от 15 Мб и больше);
- Под нестандартные случаи сложно найти примеры.

Qt предоставляет программисту не только удобный набор библиотек классов, но и определённую модель разработки приложений, определённый каркас их структуры. Следование принципам и правилам «хорошего стиля программирования на C++/Qt» существенно снижает частоту таких трудно отлавливаемых ошибок в приложениях, как утечки памяти (memory leaks), необработанные исключения, незакрытые файлы или неосвобождённые дескрипторы ресурсных объектов, чем нередко страдают программы, написанные «на голом C++» без использования библиотеки Qt.

Важным преимуществом Qt является хорошо продуманный, логичный и стройный набор классов, предоставляющий программисту очень высокий уровень абстракции. Благодаря этому программистам, использующим Qt, приходится писать значительно меньше кода, чем это имеет место при использовании, например, библиотеки классов MFC. Сам же код выглядит стройнее и проще, логичнее и

понятнее, чем аналогичный по функциональности код MFC или код, написанный с использованием «родного» для X11 тулкита Xt. Его легче поддерживать и развивать.

В случае использования Qt для создания приложения под другие системы понадобится всего лишь перекомпиляция исходного кода. В случае же использования, например, MFC или «родных» системных API понадобится много тяжёлой работы по портированию, адаптации и отладке, а то и переписыванию с нуля существующего исходного кода для другой ОС или аппаратной платформы.

Многие компании-разработчики приложений Windows используют Qt ещё по одной причине: даже если код пишется и в обозримом будущем будет писаться только для платформы Windows и тестируется только на ней, возможность откомпилировать один и тот же исходный код на одной и той же платформе Windows двумя разными компиляторами (Microsoft Visual C++ и GCC/Win32) гарантирует лучшее качество исходного кода и лучшую его совместимость со стандартом C++. Что немаловажно для кода, который планируется длительно поддерживать и развивать.

АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Архитектура показывает выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов. Показывает соединение выбранных элементов структуры.

Диаграмма классов изображена на рисунке 2.

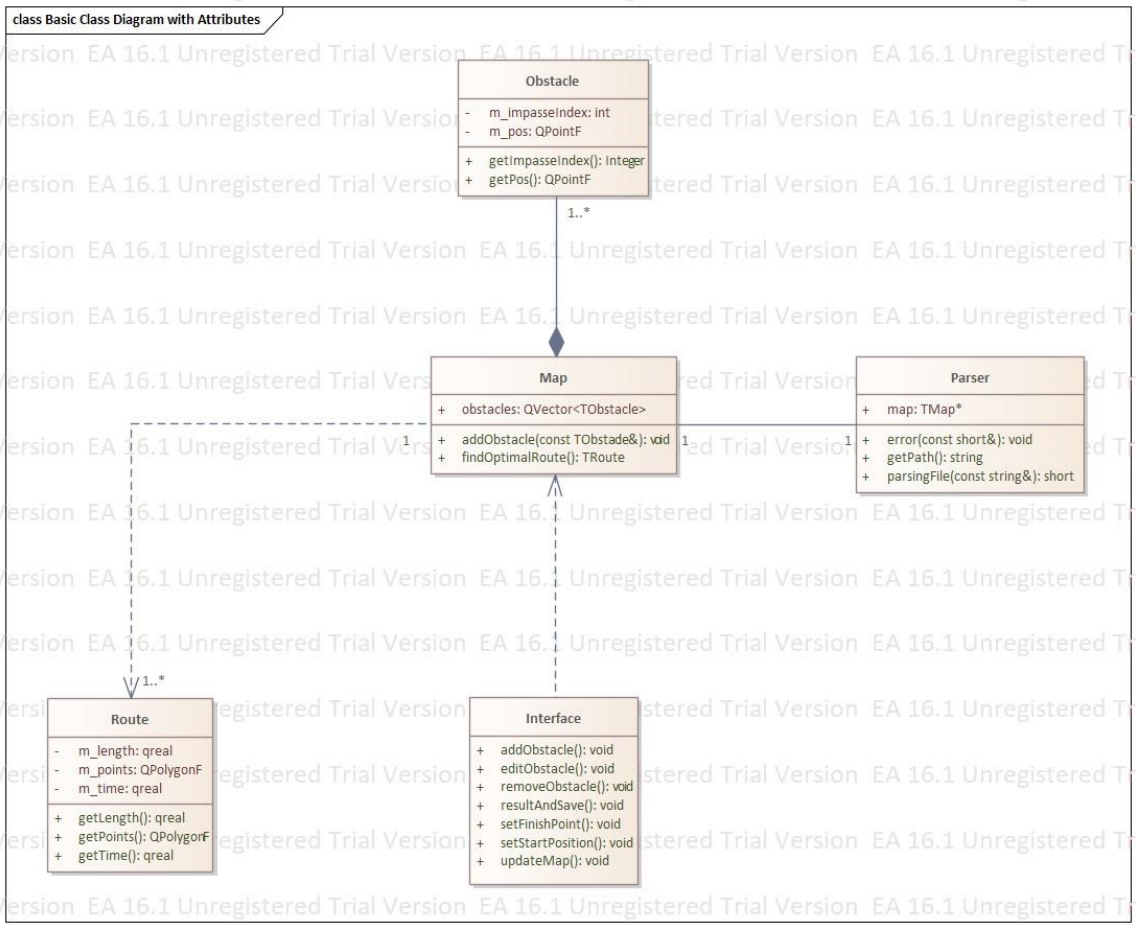

Рисунок 2 - Диаграмма классов

- На диаграмме классов изображены отношения между 5 реализованными классами:
    - Карта(Map) – главный класс, в атрибутах имеет вектор объектов класса препятствий(Obstacle), а также методы для добавления препятствий и поиска оптимального пути(Route)
    - Препятствие(Obstacle) – атрибуты и методы используются для реализации собственного инструментария класса
    - Синтаксический анализатор(Parser) – для расшифровки карты из файла.
    - Маршрут(Route) – класс формируемых маршрутов

- Интерфейс(Interface)

- Полную структуру классов представлена в спецификации классов(см. **Табл. 9. Спецификация классов**)

Диаграмма состояний изображена на рисунке 3.

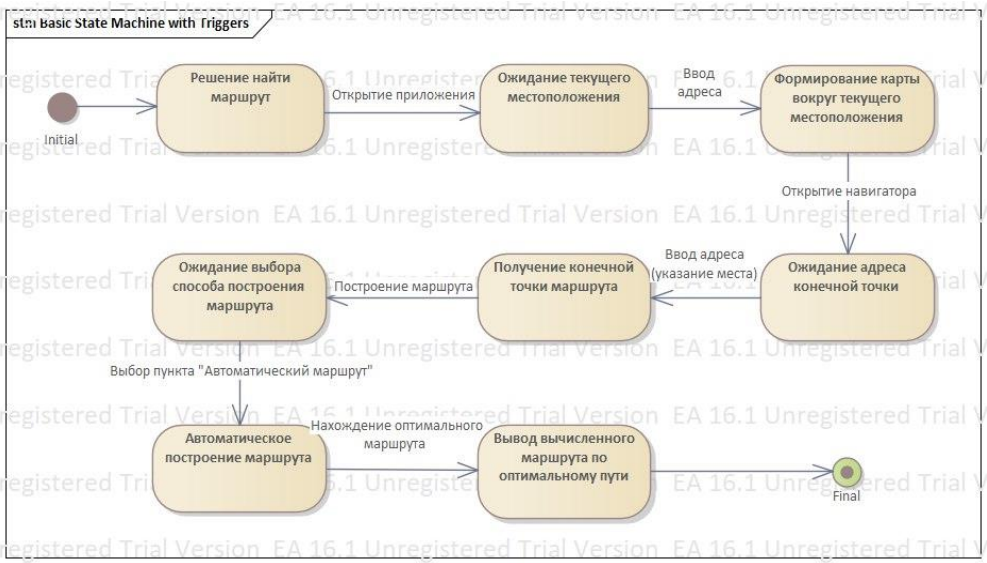

Рисунок 3 - Диаграмма состояний

- На диаграмме состояний для моделирования динамических аспектов системы представлены состояния и переходы системы.
- В элементах состояния представлены контрольные точки работы системы, такие как:
  - ожидание необходимых данных или действий от пользователя
  - проведение вычислений(построение маршрутов)
- В переходах показаны промежуточные контрольные точки, в которых происходит либо ввод необходимых данных от пользователя(в т.ч. выбор пунктов меню), либо происходят промежуточные вычисления

Диаграмма деятельности работы модуля "Карта" изображена на рисунке 4.

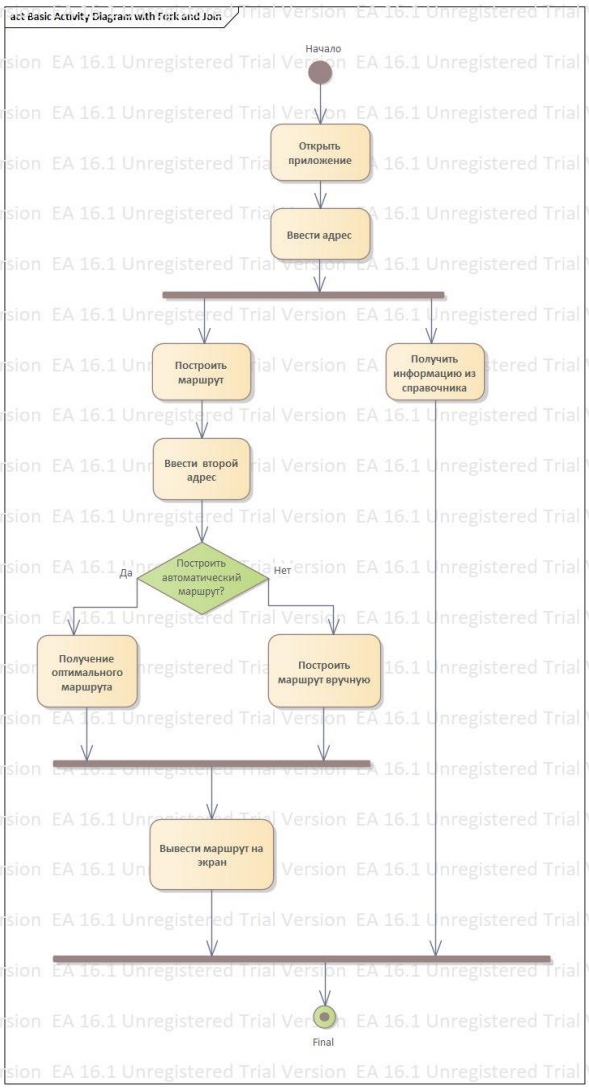

Рисунок 4 - Диаграмма деятельности

- На диаграмме деятельности(активности) представлены динамические аспекты поведения системы. На диаграмме деятельности в виде блок-схемы визуализирован случай использования с учётом принятия разных решений, в зависимости от намерения пользователя(построить маршрут/получить информацию из справочника), а так же представлена конструкция условия(построение автоматического/ручного маршрута).

Диаграмма последовательностей изображена на рисунке 5.

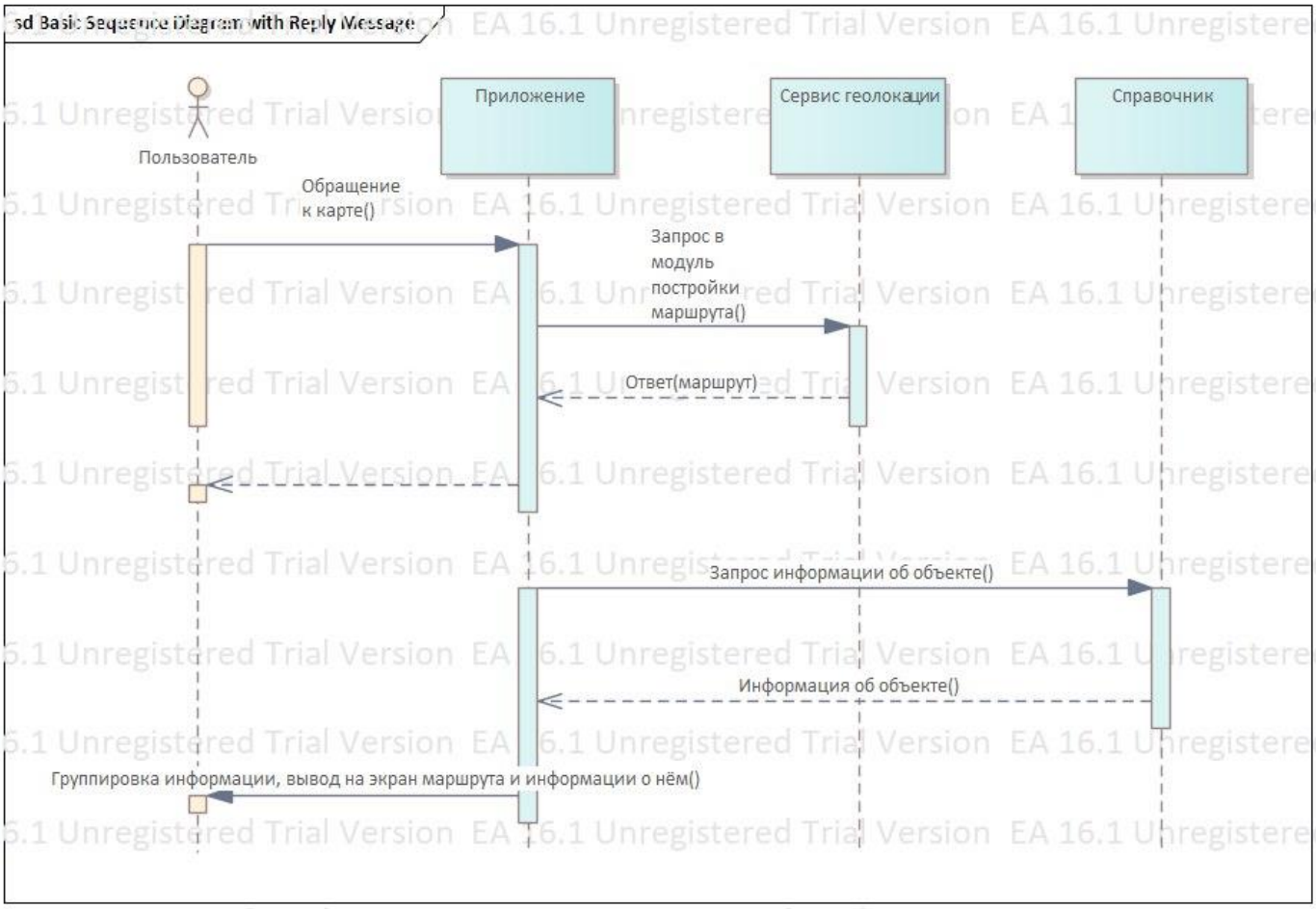

Рисунок 5 – Диаграмма последовательностей

- На диаграмме последовательностей на единой временной оси представлен процесс взаимодействия частей информационной системы(приложение/сервис геолокации/справочник) между собой с помощью конструкции вида:”Запрос-Ответ”, инициируемые обращениями пользователя к карте с помощью интерфейса модуля “Приложение”.

ЭСКИЗ ЭКРАННОЙ ФОРМЫ

Главное меню:

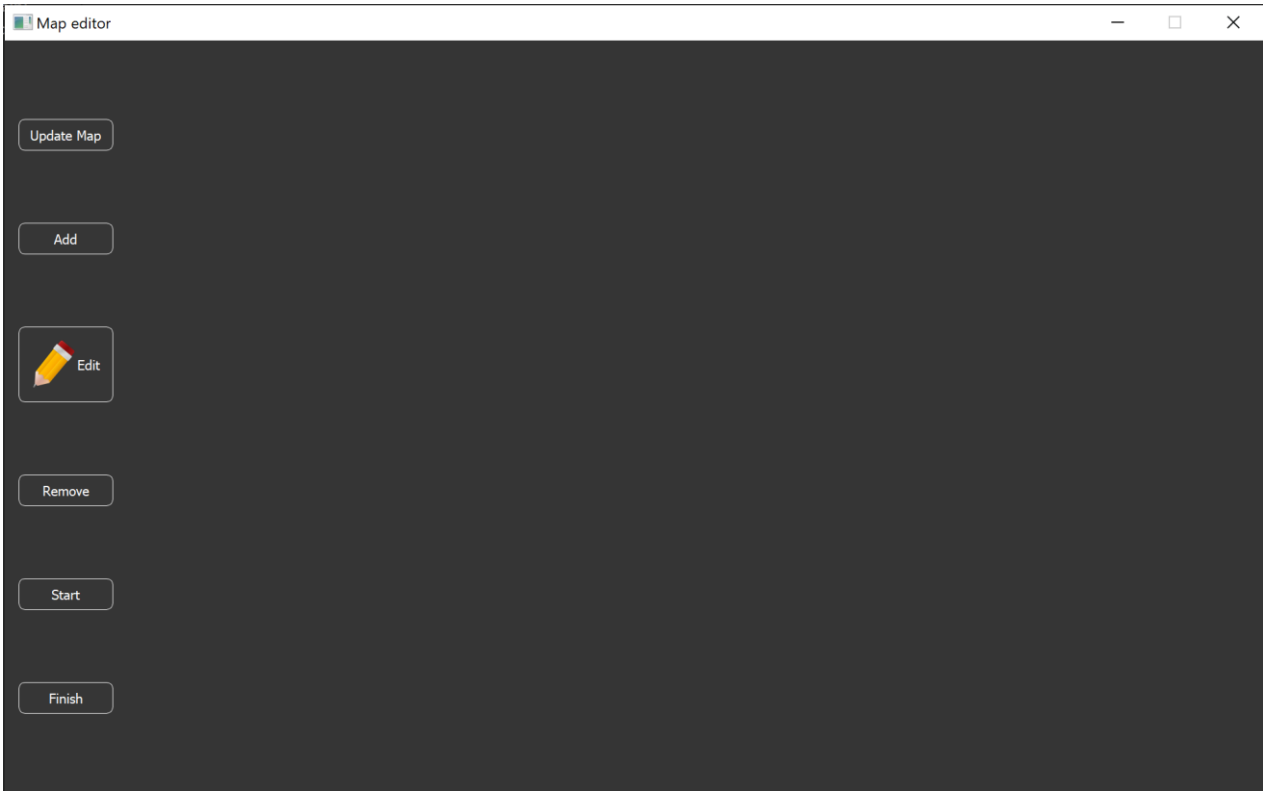

Рисунок 6 - Главное меню

Рисунок 7 – Прототип интерфейса карты

ТЕСТИРОВАНИЕ ПРИЛОЖЕНИЯ

Целью тестирования приложения является выявление соответствия разработанного приложения требованиям заинтересованных сторон.

Условия тестирования

Приложение тестировалось в операционных системах:

- Windows
- Linux

Тестирование

Прогоним приложение с проверкой на выполнение функциональных требований пользователей. Для модульного тестирования созданы тест-кейсы и разработаны сценарии. Системное тестирование полной системы будет проводиться в виде ручного тестировнаия.

В ходе тестирования приложения на выполнение функциональных требований ошибок не обнаружено. Все требования соблюдены. Приложение работает стабильно на всех операционных системах, указанных в требованиях пользователя.

Список тестов:

1. Модульное тестирование функции «**editObstacle**()»;
2. Модульное тестирование функции «**TParsing**()»;
3. Тест на обновление карты;
4. Тест на выявление оптимального маршрута;
5. Тест на функциональность сервиса.
6. Системное тестирование полной системы на соответствие требованиям.

Возможные направления эволюции

Сервис «Карта Маршрутов» в своих будущих версиях может быть дополнено новыми функциями, такими как:

- Реализация для ОС Android, IOS;
- Улучшение алгоритмов маршрутизации;
- Улучшение экологической устойчивости.

Данный пункт нужен для разработчиков для того, чтобы они могли учесть в разработке первой версии приложения возможные пути эволюции. То есть в будущем добавить новые функции не должно возникать противоречий между существующей архитектурой проекта и внедряемыми функциям.

ЗАКЛЮЧЕНИЕ

В ходе выполнения практической работы была разработана Карта Маршрутов. Был проведен анализ существующих решений и выбраны подходящие технологии для реализации картографического сервиса.

Были сформированы технические требования, также было составлено архитектурное описание и тесты для проекта. При разработке архитектуры было обозначено несколько видов классов: интерфейс, карта, препятствие, парсинг и подсчет.

Разработанный сервис может быть использован в качестве основы для создания более сложных и функциональных карт, а полученные знания и опыт могут быть применены в дальнейшей работе над подобными проектами.

ИСТОЧНИКИ

1. Водяхо А.И., Выговский Л.С., Дубенецкий В.А., Цехановский В.В., Архитектурные решения информационных систем. - СПб.: Издательство «Лань», 20 с.