

Санкт-Петербургский Политехнический Университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

# Программирование

Отчет по лабораторной работе  
Транспортное расписание

**Работу выполнил:**

Мальцев М.С.

Группа: 13501/4

**Преподаватель:**

Вылегжанина К.Д.

Санкт-Петербург  
2016

# Содержание

<b>1</b>	<b>Транспортное расписание</b>	<b>2</b>
1.1	Задание . . . . .	2
1.2	Концепция . . . . .	2
1.3	Минимально работоспособный продукт . . . . .	2
1.4	Диаграмма прецедентов использования . . . . .	2
1.5	Диаграмма последовательностей . . . . .	3
<b>2</b>	<b>Проектирование приложения</b>	<b>3</b>
2.1	Архитектуру приложения . . . . .	3
2.2	Диаграмма компонентов . . . . .	4
2.3	Файлы создаваемые в процессе работы приложения . . . . .	4
2.4	Интерфейс ядра . . . . .	4
<b>3</b>	<b>Реализация Транспортного расписания</b>	<b>5</b>
3.1	Используемые версии . . . . .	5
3.2	Основные классы . . . . .	5
3.3	Скриншоты основных экранов пользовательского интерфейса . . . . .	6
<b>4</b>	<b>Процесс обеспечения качества и тестирование</b>	<b>7</b>
4.1	О ревью . . . . .	7
4.2	О демо . . . . .	8
4.3	Список использованных утилит . . . . .	8
4.4	Автоматические тесты . . . . .	8
<b>5</b>	<b>Выводы</b>	<b>9</b>
<b>6</b>	<b>Листинги</b>	<b>9</b>

# 1 Транспортное расписание

## 1.1 Задание

Реализовать проект Транспортное расписание

"Транспортное расписание программа позволяющая создать, редактировать и использовать расписание для поездов метрополитена

## 1.2 Концепция

Программа должна предоставлять обычному пользователю возможность просмотра маршрута поездов, показывать информацию о станций и помогать найти способ проезда до нужной станции. У администратора в отличие от обычного пользователя должны присутствовать права на редактирования данных.

## 1.3 Минимально работоспособный продукт

Программа, которая позволяет пользователю просмотреть маршруты поездов и информацию о станций

## 1.4 Диаграмма прецедентов использования

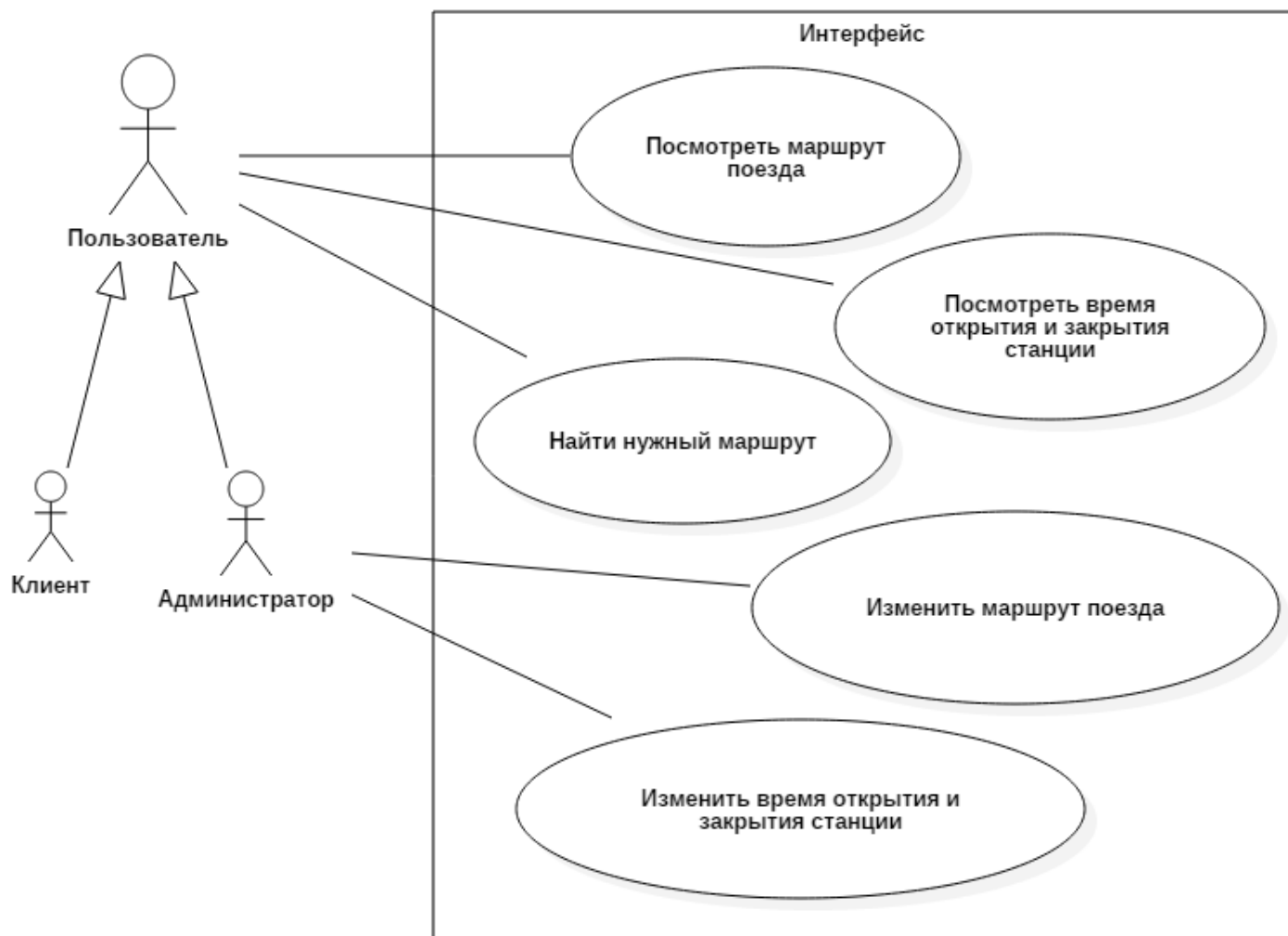


Рис. 1: Диаграмма прецедентов использования

## 1.5 Диаграмма последовательностей

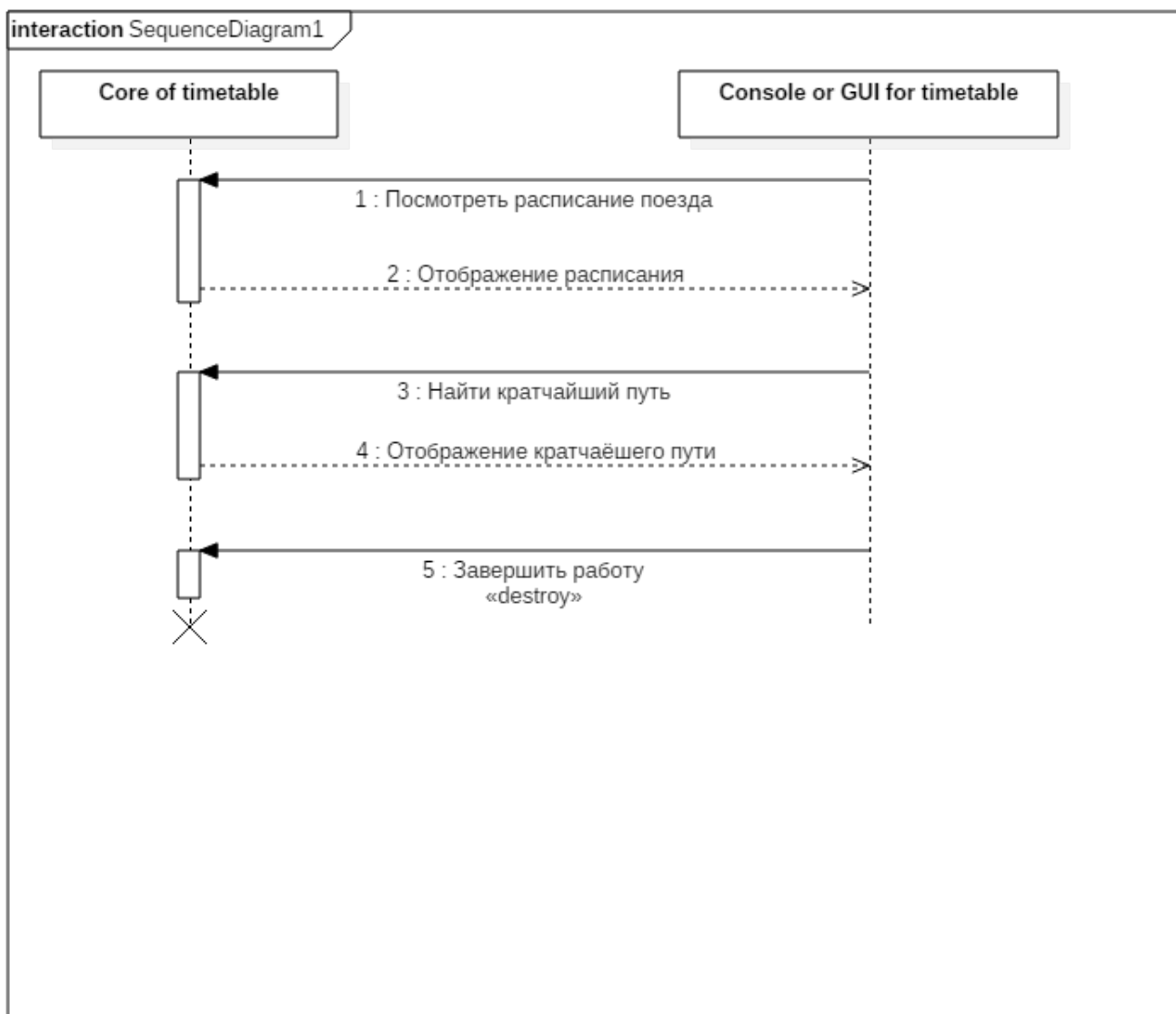


Рис. 2: Диаграмма последовательностей

## 2 Проектирование приложения

### 2.1 Архитектуру приложения

Было решено выделить 4 подпроекта:

1. Консольное приложение - подпроект, цель которого предоставить пользователю функциональности ядра с помощью консоли
2. Библиотека - подпроект, содержащий основную бизнес-логику всего проекта
3. Графическое приложение - подпроект, созданный для того, чтобы с помощью графического интерфейса предоставить пользователю функциональности ядра
4. Тесты - подпроект, созданный для того, чтобы тестировать библиотеку, содержащую основную бизнес-логику

## 2.2 Диаграмма компонентов

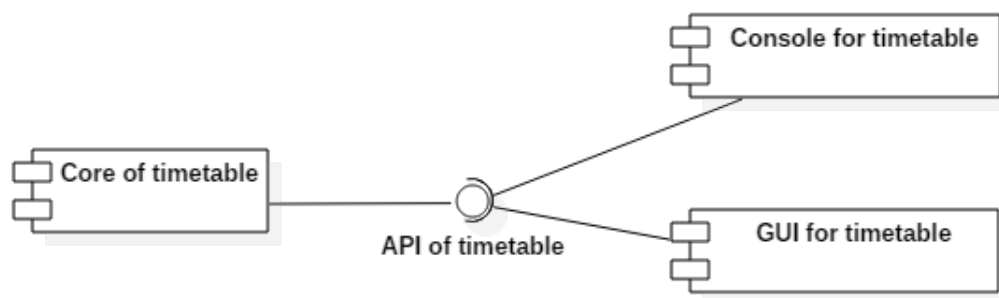


Рис. 3: Диаграмма компонентов

## 2.3 Файлы создаваемые в процессе работы приложения

Называются файлы могут как угодно пользователю.

Формат данных должен быть следующий:

Для маршрутов: Devyatkino,Grazhdansky Prospekt/Parnas,Prospekt Prosvescheniya

Для станций: Parnas 5.47-0.00/Prospekt Prosvescheniya 5.37-0.40

## 2.4 Интерфейс ядра

В библиотека предоставляет следующую функциональность:

1. `void putInfoAboutMetro(const std::string &infoAboutRoutes, const std::string &infoAboutStations) noexcept;`

Один из способ передать входные данные, на вход принимаются две строки в определённом формате, которые ядро будет парсить, а потом передаст классам, отвечающим за хранение информации

2. `void loadInfoFromFile(const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station);`

Ввод данных с помощью файлов. В этот метод передаются два параметра, это название файлов, один файл - информация о маршрутах, второй о станциях. Информация, как и в предыдущем методе должна находится в определённом формате

3. `int howManyRoutes() const noexcept;`

Возвращает информацию о том, сколько маршрутов существуетна данный момент

4. `std::vector<std::string> getRoute(const int number_of_the_route);`

Возвращает запрашиваемый маршрут

5. `std::string getInfoAboutStation(const std::string &name_of_the_station);`

Возвращает информацию о запрашиваемой станции

6. `std::string getInfoAboutStation(const int number_of_the_route, const int number_of_the_station);`

Альтернативный способ получения информации о станции

7. `void changeStationInRoute(const int number_of_the_route, const int number_of_the_station, const std::string &new_marking);`

Изменить название станции в маршруте

8. `void addStationInRoute(const int number_of_the_route, const std::string &what_to_add);`
- Добавить новую станцию в маршрут
9. `int addRoute() noexcept;`
- Добавляет новый маршрут
10. `void deleteStationFromRoute(const int number_of_the_route, const int number_of_the_station);`
- Удаляет станцию из маршрута
11. `void deleteRoute(const int number_of_the_route);`
- Удаляет маршрут
12. `void addInfoAboutStation(const std::string &name_of_the_station, const std::string &station_description) noexcept;`
- Добавление информации о станции
13. `void addInfoAboutStation(const int number_of_the_route, const int number_of_the_station, const std::string &station_description);`
- Альтернативный способ добавления информации о станции
14. `void removeInfoAboutStation(const std::string &what_station_to_remove);`
- Удаление информации о станции
15. `void removeInfoAboutStation(const int number_of_the_station);`
- Альтернативный способ удаления информации о станции
16. `std::vector<std::pair<std::string, std::string>> getAllStationsWhichHaveDescription() noexcept;`
- Возвращает станции к которым существует описание
17. `void saveChanges(const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station) noexcept;`
- Сохраняет все изменения в файлы

## 3 Реализация Транспортного расписания

### 3.1 Используемые версии

- Qt Creator 4.0.0 (opensource)
- Стандарт c++11
- GCC 5.4.2 и GCC 5.6.0
- Операционная система: Windows 10
- cprcheckgui версии 1.7.2

### 3.2 Основные классы

Из библиотеке хотелось бы упомянуть про следующие классы:

- `CoreOfInfoAboutMetro` - класс, который отвечает за перенаправление задач к подконтрольным ему классам
- `ParsingInfo` - класс, цель которого парсинг информации

- RoutesInfo - класс, отвечающий за обработку информации связанной с маршрутами
- StationsInfo - класс, отвечающий за обработку информации связанной со станциями

### 3.3 Скриншоты основных экранов пользовательского интерфейса

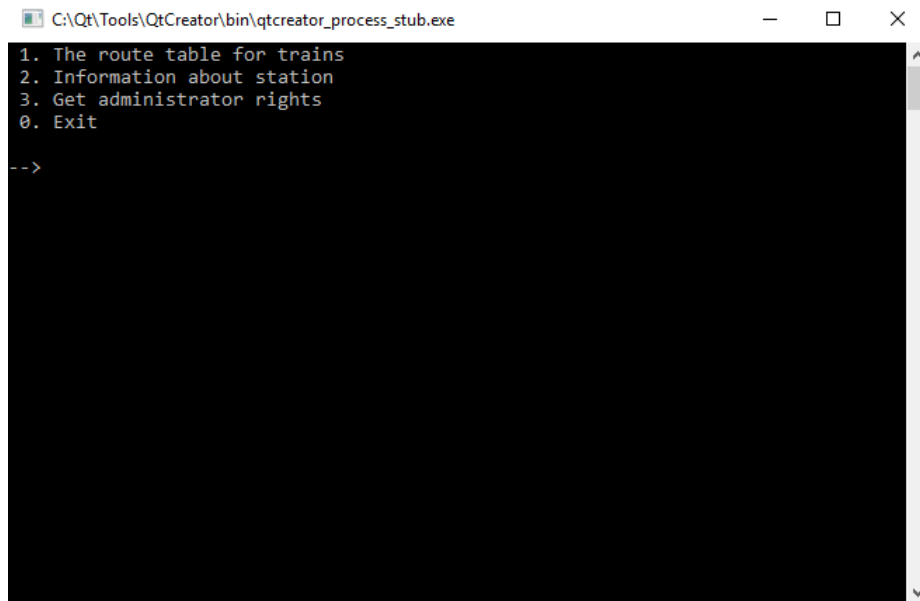
A screenshot of a console window titled "C:\Qt\Tools\QtCreator\bin\qtcreator\_process\_stub.exe". The window contains a menu with the following options: "1. The route table for trains", "2. Information about station", "3. Get administrator rights", and "0. Exit". Below the menu, there is a prompt "-->" followed by a cursor.

Рис. 4: Меню консольного приложения

Здесь предоставлены основные функциональность предлагаемые пользователю:

- Получить информацию о маршруте
- Получить информацию о станции
- Выход

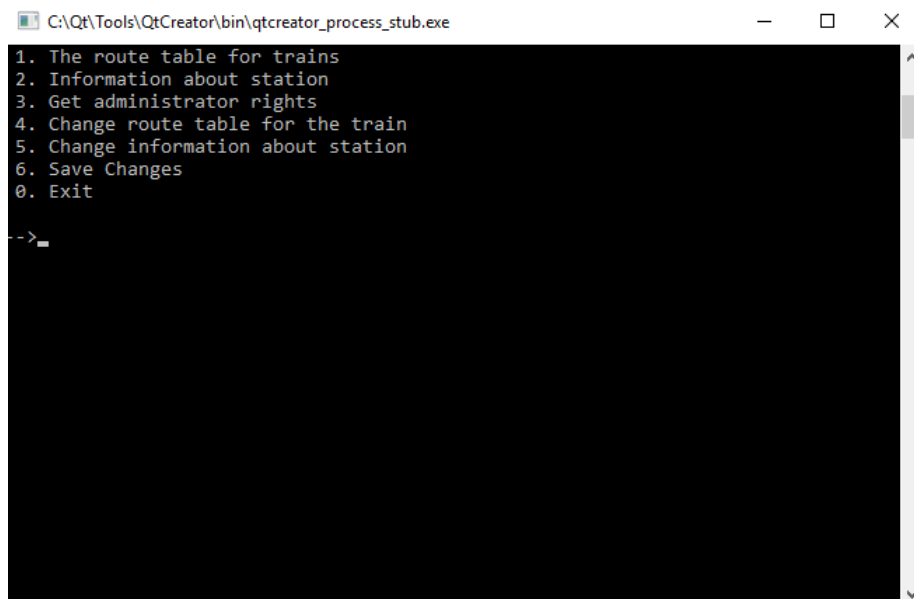
A screenshot of a console window titled "C:\Qt\Tools\QtCreator\bin\qtcreator\_process\_stub.exe". The window contains an expanded menu with the following options: "1. The route table for trains", "2. Information about station", "3. Get administrator rights", "4. Change route table for the train", "5. Change information about station", "6. Save Changes", and "0. Exit". Below the menu, there is a prompt "-->" followed by a cursor.

Рис. 5: Меню консольного приложения, расширенное для администрирования

Кроме предидущих для администратора становятся доступны функциональности связанные с редактированием:

- Изменить информацию о маршрутах
- Изменить информацию о станциях
- Сохранить изменения

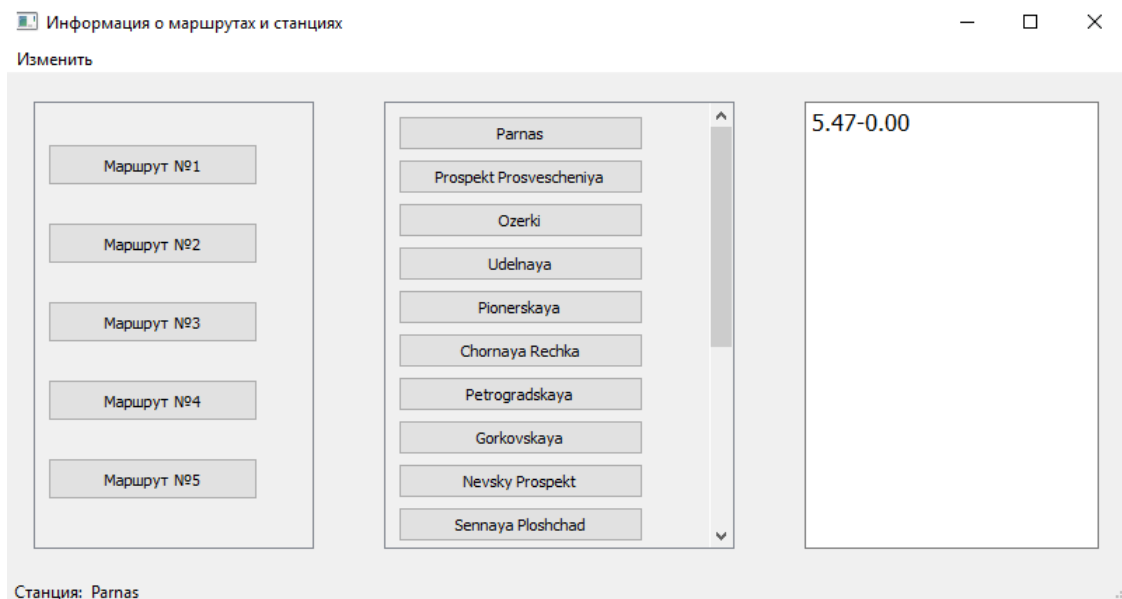


Рис. 6: Главное окно графического приложения

Здесь показано главное окно, в нём можно выбрать маршрут и получить информацию, о станции находящейся в нём

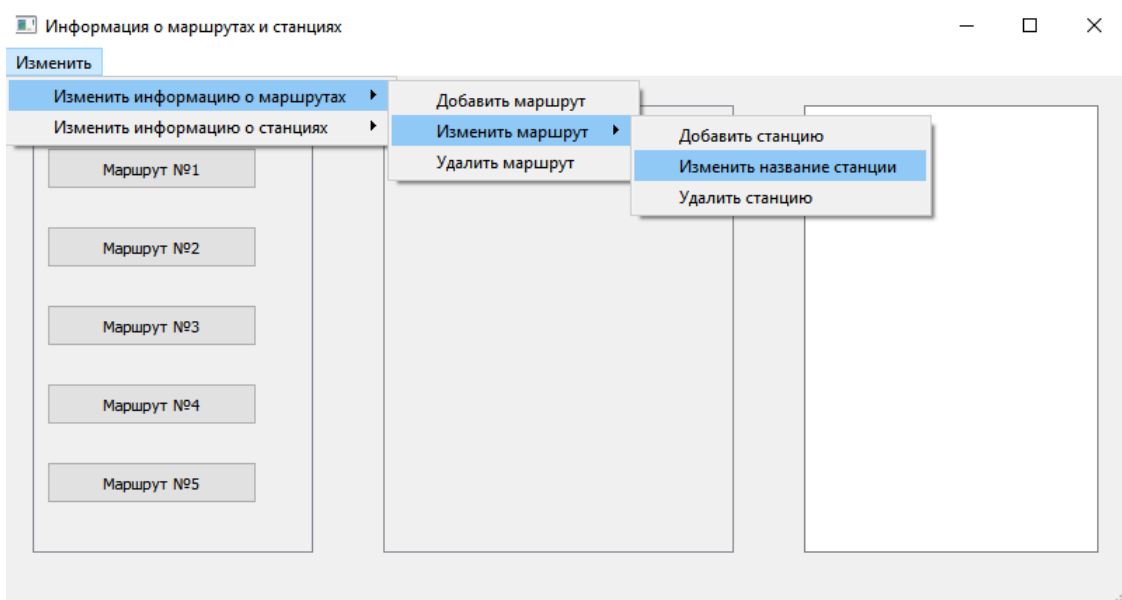


Рис. 7: Открыто меню изменения маршрутов

Здесь продемонстрировано меню редактирования информации о маршрутах и станциях

## 4 Процесс обеспечения качества и тестирование

### 4.1 О ревью

Было получено 6 ревью:



- в первом 9 замечаний
- во втором 15 замечаний
- в третьем 21 замечаний
- в четвёртом 29 замечаний
- в пятом 80 замечаний
- в шестом 68 замечаний

## 4.2 О демо

Было 2 демо:

- Замечания из первого демо:
  - +1. Добавить кнопку help или выводить меню, чтобы показать перечень того, из чего может выбрать пользователь.
  - +2. Мы хотим, чтобы пользователю выводилось сообщение о том, что добавлена станция не за пределами ветки, а в конец ветки.
  - +3. Сделать так, чтобы в любом случае ввода необходимы были только цифры, а не только цифры и буквы. Пользователю проще ввести цифры, чем длинное название станции.
  - +4. Feature-request. Запрос на функциональность. В минутах получать расстояние между станциями.
  - +5. Разнести добавление, удаление и замену станций.
  - +6. Сделать ревью строк английского.
 Исправлено всё, кроме пункта 4
- Во втором демо, замечание было очень маленькое, поставить ещё один отступ при выводе в консоль. Было сразу же исправлено

## 4.3 Список использованных утилит

Были использованы:

- `srccheck`, в данном случае он, в основном, выдавал преупреждения связанные с производительностью (например, "лучше передавать по ссылке"), это очень помогло
- `jenkins` – именно, благодаря ему было найдено наибольшее количество замечаний и ошибок. Где-то забыл комментарий, потерялась кроссплатформенность кода, всё это было выявлено благодаря `jenkins`

## 4.4 Автоматические тесты

При разработке приложения были написаны автоматические тесты, целью которых было выявить неправильную работу библиотеки, и по моему мнению, они со своей задачей справились. Автоматические тесты – это очень удобно, потому что когда вы пытаетесь изменить или добавить бизнес-логику, старый код может перестать работать, а это очень просто выявляется с помощью тестов.

Сценарий автоматических тестов, был следующий: методам были переданы определённые параметры и то, что методы вернули сравнивалось с ожидаемым.

На графике продемонстрирован процент покрытия тестами:

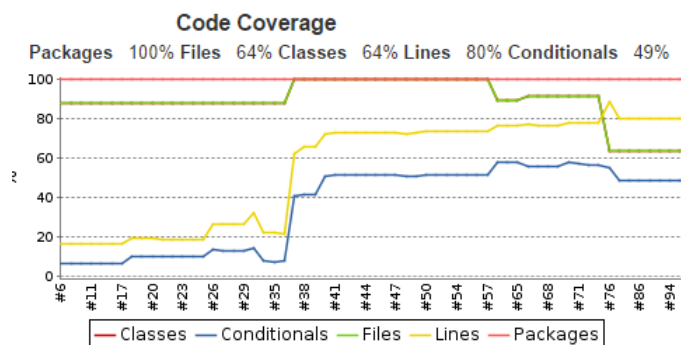


Рис. 8: Процент покрытия

Процент покрытия в середине разработки упал, потому что появилась функциональность которую нельзя было протестировать автоматически, например "Сохранение данных она создавала файл, а jenkins не разрешал этого делать, поэтому пришлось протестировать вручную

Под конец разработки процент покрытия упал, потому что большая часть библиотеки была переписана, а все силы были брошены на графический интерфейс

Вручную также было протестировано графическое и консольное приложение. Тестирование проводилось путём использования всех предоставляемых приложением возможностей. По очереди вызывались функциональности приложения и делался вывод, о том что что-то стоит переделать, а что-то работает достаточно хорошо, чтобы это оставить

## 5 Выводы

За этот семестр я :

- Узнал о существовании разных стандартов языка, в частности про c++11
- Осознал значимость ревью, понял что важно как просить, так и самому делать его другим
- Получил опыт в написание ядра приложения, теперь при написании нового будет проще
- Узнал и опробывал среду непрерывной интеграции jenkins
- Научился делать простейший графический интерфейс
- Осознал необходимость наследования классов, для эффективной разработки проекта
- И возможно ещё что-то, о чём я забыл упомянуть

В целом этот семестр мне понравился, был получен достаточно большой объём знаний, и были получены полезные практические навыки. Особенно понравилось то, что в течение семестра проводились семинары. На них было получено очень много полезной информации о том, как улучшить структуру и код программ, а так же как использовать возможности языка наиболее эффективно.

## 6 Листинги

```
1 #ifndef API_H
2 #define API_H
3
4 #include "handling_info/routes_info.h"
5 #include "handling_info/stations_info.h"
6 #include "handling_info/parsing_info.h"
7 #include "exception_of_core/exception_of_core.h"
8
9 enum class Rights_of_customers{user = 0, administrator = 1};
10
11 /**
12  * @brief Абстрактный класс, которым описывается функциональность предоставляемая ядром приложения
13  */
14 class API
15 {
16 public:
17
18     API() = default;
19
20     API(const API&) = delete;
21     API& operator= (const API&) = delete;
22
23     virtual ~API(){}
24
25     /**
26      * @brief Выдача прав
27      * @param rights — какие права нужно выдать
28      */
29     virtual void putOfRights(const Rights_of_customers rights) noexcept = 0;
30
31     /**
32      * @return Информация о правах
```

```

33     */
34     virtual Rights_of_customers getInformationOfTheRights() const noexcept = 0;
35
36     /**
37     * @brief Один из способ передать входные данные
38     * @param infoAboutRoutes – информация о маршрутах в формате: Devyatkinoy ,Grazhdansky
39     *   Prospekt/Parnas ,Prospekt Prosvescheniya
40     *   где через запятую указываются название станций, а слеш отделяет маршруты
41     * @param infoAboutStations – информация о станциях в формате: Parnas~5.47–0.00/Prospekt
42     *   Prosvescheniya~5.37–0.40
43     *   где слеш отделяет станции, а тильда отделяет название станции от информации связанной с ней
44     */
45     virtual void putInfoAboutMetro(const std::string &infoAboutRoutes, const std::string &
46     * infoAboutStations) noexcept = 0;
47
48     /**
49     * @brief Загрузить входные данные из файла
50     * @param name_of_the_file_with_route – название файла, для хранения информации о маршрутах
51     * @param name_of_the_file_with_station – название файла, для хранения информации о станциях
52     * Информация должна находиться в таком же формате, как и в putInfoAboutMetro
53     * @exception При неудачном открытии файла бросается MissingFile, от объекта которого можно
54     *   получить информацию о неверном имени файла
55     */
56     virtual void loadInfoFromFile(const std::string &name_of_the_file_with_route, const std::
57     * string &name_of_the_file_with_station) = 0;
58
59     /**
60     * @return Сколько маршрутов существует на данный момент
61     */
62     virtual int howManyRoutes() const noexcept = 0;
63
64     /**
65     * @param number_of_the_route – номер запрашиваемого маршрута
66     * @return Запрашиваемый маршрут в формате vector<string> , где string – это название станций
67     * @exception При запросе несуществующего маршрута бросается RouteDoesNotExist, от объекта
68     *   которого можно получить информацию, какой маршрут запрашивали
69     */
70     virtual std::vector<std::string> getRoute(const int number_of_the_route) = 0;
71
72     /**
73     * @param name_of_the_station – название станции, о которой нужно получить информацию
74     * @return информация о запрашиваемой станции
75     * @exception При запросе несуществующей станции бросается StationDoesNotExist
76     */
77     virtual std::string getInfoAboutStation(const std::string &name_of_the_station) = 0;
78
79     /**
80     * @brief Способ альтернативный предыдущему
81     * @param number_of_the_route – номер маршрута, содержащего станцию
82     * @param number_of_the_station – номер станции в маршруте
83     * @return информация о запрашиваемой станции
84     * @exception При запросе несуществующей станции бросается StationDoesNotExist, а
85     *   RouteDoesNotExist, если маршрута не существует
86     */
87     virtual std::string getInfoAboutStation(const int number_of_the_route, const int
88     * number_of_the_station) = 0;
89
90     /**
91     * @brief Изменение станции в маршруте
92     * @param number_of_the_route – номер маршрута
93     * @param number_of_the_station – номер станции
94     * @param new_marking – то, что нужно поставить в замен
95     * @exception При неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
96     */
97     virtual void changeStationInRoute(const int number_of_the_route, const int
98     * number_of_the_station, const std::string &new_marking) = 0;
99
100     /**
101     * @brief Добавляет станцию в маршрут
102     * @param number_of_the_route – номер маршрута, в который нужно добавить станцию
103     * @param what_to_add – название станции, которую нужно добавить
104     * @exception При неверном запросе бросается RouteDoesNotExist
105     */
106     virtual void addStationInRoute(const int number_of_the_route, const std::string &
107     * what_to_add) = 0;

```

```

101  /**
102   * @brief Добавляет новый пустой() маршрут
103   * @return Номер добавленного маршрута
104   */
105  virtual int addRoute() noexcept = 0;
106
107  /**
108   * @brief Удаляет станцию из маршрута
109   * @param number_of_the_route — номер маршрута, из которого нужно удалить станцию
110   * @param number_of_the_station — номер станции, которую нужно удалить
111   * @exception При неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
112   */
113  virtual void deleteStationFromRoute(const int number_of_the_route, const int
↪ number_of_the_station) = 0;
114
115  /**
116   * @brief Удаляет маршрут
117   * @param number_of_the_route — номер маршрута, который нужно удалить
118   * @exception При неверном запросе бросается RouteDoesNotExist
119   */
120  virtual void deleteRoute(const int number_of_the_route) = 0;
121
122  /**
123   * @brief Добавляет информацию о станции
124   * @param name_of_the_station — номер станции, о которой информация
125   * @param station_description — информация о станции
126   */
127  virtual void addInfoAboutStation(const std::string &name_of_the_station, const std::string
↪ &station_description) noexcept = 0;
128
129  /**
130   * @brief Альтернативный путь добавления информации о станции
131   * @param number_of_the_route — номер маршрута, в котором содержится станция, про которую
↪ информация
132   * @param number_of_the_station — номер станции, про которую информация
133   * @param station_description — информация о станции
134   * @exception При неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
135   */
136  virtual void addInfoAboutStation(const int number_of_the_route, const int
↪ number_of_the_station, const std::string &station_description) = 0;
137
138  /**
139   * @brief Удаляет информацию о станции
140   * @param what_station_to_remove — название станции, которую нужно удалить
141   * @exception При неверном запросе бросается StationDoesNotExist
142   */
143  virtual void removeInfoAboutStation(const std::string &what_station_to_remove) = 0;
144
145  /**
146   * @brief Альтернативный путь удаления информации о станции, в месте с ним используется метод
147   * getAllStationsWhichHaveDescription(), именно из него берётся номер станции
148   * @param number_of_the_station — номер станции из getAllStationsWh..
149   * @exception При неверном запросе бросается StationDoesNotExist
150   */
151  virtual void removeInfoAboutStation(const int number_of_the_station) = 0;
152
153  /**
154   * @return Всё станции, про которые содержится информация, первая часть пары название станции,
155   * вторая информация о ней
156   */
157  virtual std::vector<std::pair<std::string, std::string>>
↪ getAllStationsWhichHaveDescription() noexcept = 0;
158
159  /**
160   * @brief Сохранить информацию
161   * @param name_of_the_file_with_route — название файла, для хранения информации о маршрутах
162   * @param name_of_the_file_with_station — название файла, для хранения информации о станциях
163   */
164  virtual void saveChanges(const std::string &name_of_the_file_with_route, const std::string
↪ &name_of_the_file_with_station) noexcept = 0;
165 };
166
167 #endif // API_H

```

```

1 #ifndef CORE_OF_TIMETABLE_H
2 #define CORE_OF_TIMETABLE_H

```

```

3
4 #include "api.h"
5
6 /**
7  * @brief Класс в котором содержится основная бизнеслогика— приложения
8  */
9 class CoreOfInfoAboutMetro : public API
10 {
11
12 public:
13
14     CoreOfInfoAboutMetro() : rights(Rights_of_customers::user) {}
15
16     void putOfRights(const Rights_of_customers rights) noexcept override;
17
18     Rights_of_customers getInformationOfTheRights() const noexcept override;
19
20     void putInfoAboutMetro(const std::string &infoAboutRoutes, const std::string &
21 ↪ infoAboutStations) noexcept override;
22
23     void loadInfoFromFile(const std::string &name_of_the_file_with_route, const std::string &
24 ↪ name_of_the_file_with_station) override;
25
26     int howManyRoutes() const noexcept override;
27
28     std::vector<std::string> getRoute(const int number_of_the_route) override;
29
30     std::string getInfoAboutStation(const std::string &name_of_the_station) override;
31
32     std::string getInfoAboutStation(const int number_of_the_route, const int
33 ↪ number_of_the_station) override;
34
35     void changeStationInRoute(const int number_of_the_route, const int number_of_the_station,
36 ↪ const std::string &new_marking) override;
37
38     void addStationInRoute(const int number_of_the_route, const std::string &what_to_add)
39 ↪ override;
40
41     int addRoute() noexcept override;
42
43     void deleteStationFromRoute(const int number_of_the_route, const int number_of_the_station
44 ↪ ) override;
45
46     void deleteRoute(const int number_of_the_route) override;
47
48     void addInfoAboutStation(const std::string &name_of_the_station, const std::string &
49 ↪ station_description) noexcept override;
50
51     void addInfoAboutStation(const int number_of_the_route, const int number_of_the_station,
52 ↪ const std::string &station_description) override;
53
54     void removeInfoAboutStation(const std::string &what_station_to_remove) override;
55
56     void removeInfoAboutStation(const int number_of_the_station) override;
57
58     std::vector<std::pair<std::string, std::string>> getAllStationsWhichHaveDescription()
59 ↪ noexcept override;
60
61     void saveChanges(const std::string &name_of_the_file_with_route, const std::string &
62 ↪ name_of_the_file_with_station) noexcept override;
63
64 private:
65
66     Rights_of_customers rights;
67
68     RoutesInfo routeInfo;
69
70     StationsInfo stationInfo;
71
72     ParsingInfo parsingInformation;
73 };
74
75 #endif // CORE_OF_TIMETABLE_H

```

```

1 #include "core.h"
2

```

```

3 void CoreOfInfoAboutMetro::putOfRights(const Rights_of_customers rights) noexcept
4 {
5     if (rights == Rights_of_customers::administrator)
6     {
7         this->rights = Rights_of_customers::administrator;
8     }
9     else
10    {
11        this->rights = Rights_of_customers::user;
12    }
13 }
14
15 Rights_of_customers CoreOfInfoAboutMetro::getInformationOfTheRights() const noexcept
16 {
17     return rights;
18 }
19
20 void CoreOfInfoAboutMetro::putInfoAboutMetro(const std::string &infoAboutRoutes, const std::
    ↪ string &infoAboutStations) noexcept
21 {
22     parsingInformation.putInfoAboutRoutes(infoAboutRoutes, routeInfo);
23
24     parsingInformation.putInfoAboutStations(infoAboutStations, stationInfo);
25 }
26
27 void CoreOfInfoAboutMetro::loadInfoFromFile(const std::string &name_of_the_file_with_route,
    ↪ const std::string &name_of_the_file_with_station)
28 {
29     parsingInformation.loadFromFile(name_of_the_file_with_route, name_of_the_file_with_station
    ↪ , routeInfo, stationInfo);
30 }
31
32 int CoreOfInfoAboutMetro::howManyRoutes() const noexcept
33 {
34     return routeInfo.getHowManyRoutes();
35 }
36
37 std::vector<std::string> CoreOfInfoAboutMetro::getRoute(const int number_of_the_route)
38 {
39     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
40     {
41         throw RouteDoesNotExist(number_of_the_route);
42     }
43
44     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
45
46     return routeInfo.getRoute(number_of_the_route_in_the_vector);;
47 }
48
49 std::string CoreOfInfoAboutMetro::getInfoAboutStation(const std::string &name_of_the_station)
50 {
51     std::string info_about_station = stationInfo.getInfoAboutStation(name_of_the_station);
52     std::string void_string;
53
54     if(info_about_station == void_string)
55     {
56         throw StationDoesNotExist(name_of_the_station);
57     }
58
59     return info_about_station;
60 }
61
62 std::string CoreOfInfoAboutMetro::getInfoAboutStation(const int number_of_the_route, const int
    ↪ number_of_the_station)
63 {
64     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
65     {
66         throw RouteDoesNotExist(number_of_the_route);
67     }
68
69     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
70     const int number_of_the_station_in_the_vector = number_of_the_station - 1;
71
72     std::vector<std::string> route = routeInfo.getRoute(number_of_the_route_in_the_vector);
73
74     int size_of_vector = route.size();

```

```

75     if(number_of_the_station > size_of_vector || number_of_the_station < 1)
76     {
77         throw StationDoesNotExist(number_of_the_station);
78     }
79
80
81     std::string info_about_station = stationInfo.getInfoAboutStation(route[
82     ↪ number_of_the_station_in_the_vector]);
83
84     if(info_about_station.empty())
85     {
86         throw StationDoesNotExist(route[number_of_the_station_in_the_vector]);
87     }
88
89     return info_about_station;
90 }
91 void CoreOfInfoAboutMetro::changeStationInRoute(const int number_of_the_route, const int
92     ↪ number_of_the_station, const std::string &new_marking)
93 {
94     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
95     {
96         throw RouteDoesNotExist(number_of_the_route);
97     }
98
99     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
100     const int number_of_the_station_in_the_vector = number_of_the_station - 1;
101
102     int size_of_vector = (routeInfo.getRoute(number_of_the_route_in_the_vector)).size();
103
104     if(number_of_the_station < 1 || number_of_the_station > size_of_vector)
105     {
106         throw StationDoesNotExist(number_of_the_station);
107     }
108
109     routeInfo.changeStationInRoute(number_of_the_route_in_the_vector,
110     ↪ number_of_the_station_in_the_vector, new_marking);
111 }
112 void CoreOfInfoAboutMetro::addStationInRoute(const int number_of_the_route, const std::string
113     ↪ &what_to_add)
114 {
115     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
116     {
117         throw RouteDoesNotExist(number_of_the_route);
118     }
119
120     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
121
122     routeInfo.addStationInRoute(number_of_the_route_in_the_vector, what_to_add);
123 }
124 int CoreOfInfoAboutMetro::addRoute() noexcept
125 {
126     routeInfo.addRoute();
127
128     return routeInfo.getHowManyRoutes();
129 }
130 void CoreOfInfoAboutMetro::deleteStationFromRoute(const int number_of_the_route, const int
131     ↪ number_of_the_station)
132 {
133     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
134     {
135         throw RouteDoesNotExist(number_of_the_route);
136     }
137
138     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
139     const int number_of_the_station_in_the_vector = number_of_the_station - 1;
140
141     int size_of_vector = (routeInfo.getRoute(number_of_the_route_in_the_vector)).size();
142
143     if(number_of_the_station < 1 || number_of_the_station > size_of_vector)
144     {
145         throw StationDoesNotExist(number_of_the_station);
146     }

```

```

146
147     routeInfo.deleteStationFromRoute(number_of_the_route_in_the_vector,
    ↪ number_of_the_station_in_the_vector);
148 }
149
150 void CoreOfInfoAboutMetro::deleteRoute(const int number_of_the_route)
151 {
152     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
153     {
154         throw RouteDoesNotExist(number_of_the_route);
155     }
156
157     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
158
159     routeInfo.deleteRoute(number_of_the_route_in_the_vector);
160 }
161
162 void CoreOfInfoAboutMetro::addInfoAboutStation(const std::string &name_of_the_station, const
    ↪ std::string &station_description) noexcept
163 {
164     stationInfo.addInfoAboutStation(name_of_the_station, station_description);
165 }
166
167 void CoreOfInfoAboutMetro::addInfoAboutStation(const int number_of_the_route, const int
    ↪ number_of_the_station, const std::string &station_description)
168 {
169     if(number_of_the_route > routeInfo.getHowManyRoutes() || number_of_the_route < 1)
170     {
171         throw RouteDoesNotExist(number_of_the_route);
172     }
173
174     const int number_of_the_route_in_the_vector = number_of_the_route - 1;
175     const int number_of_the_station_in_the_vector = number_of_the_station - 1;
176
177     std::vector<std::string> route = routeInfo.getRoute(number_of_the_route_in_the_vector);
178
179     int size_of_vector = route.size();
180
181     if(number_of_the_station < 1 || number_of_the_station > size_of_vector)
182     {
183         throw StationDoesNotExist(number_of_the_station);
184     }
185
186     stationInfo.addInfoAboutStation(route[number_of_the_station_in_the_vector],
    ↪ station_description);
187 }
188
189 void CoreOfInfoAboutMetro::removeInfoAboutStation(const std::string &what_station_to_remove)
190 {
191     if (stationInfo.getInfoAboutStation(what_station_to_remove) == "")
192     {
193         throw StationDoesNotExist(what_station_to_remove);
194     }
195
196     stationInfo.removeInfoAboutStation(what_station_to_remove);
197 }
198
199 void CoreOfInfoAboutMetro::removeInfoAboutStation(const int number_of_the_station)
200 {
201     const int number_of_the_station_in_the_vector = number_of_the_station - 1;
202
203     int size_of_vector = (stationInfo.getAllStations()).size();
204
205     if (size_of_vector < number_of_the_station || number_of_the_station < 1 || size_of_vector
    ↪ == 0)
206     {
207         throw StationDoesNotExist(number_of_the_station);
208     }
209
210     auto info_about_station = (stationInfo.getAllStations())[
    ↪ number_of_the_station_in_the_vector];
211
212     stationInfo.removeInfoAboutStation(info_about_station.first);
213 }
214

```



```

215 std::vector<std::pair<std::string, std::string>> CoreOfInfoAboutMetro::
    ↪ getAllStationsWhichHaveDescription() noexcept
216 {
217     auto all_station = stationInfo.getAllStations();
218
219     for(unsigned int i = 0; i < all_station.size(); i++)
220     {
221         if ((all_station[i]).second == "")
222         {
223             all_station.erase(all_station.begin() + i);
224         }
225     }
226
227     return all_station;
228 }
229
230 void CoreOfInfoAboutMetro::saveChanges(const std::string &name_of_the_file_with_route, const
    ↪ std::string &name_of_the_file_with_station) noexcept
231 {
232     parsingInformation.saveChanges(name_of_the_file_with_route, name_of_the_file_with_station,
    ↪ routeInfo, stationInfo);
233 }

```

```

1  #ifndef FILESCCHEDULE_H
2  #define FILESCCHEDULE_H
3
4  #include <string>
5  #include <vector>
6  #include "exception_of_core/exception_of_core.h"
7  #include <map>
8
9  enum class Part_of_buffer{name = 0, value = 1};
10
11 /**
12  * @brief Класс отвечающий за работу с информацией связанной со станциями
13  */
14 class StationsInfo
15 {
16 public:
17
18     /**
19     * @brief Получить информацию о станции
20     * @param name_of_the_station — название станции, о которой хотят получить информацию
21     * @return Информация о станции
22     */
23     std::string getInfoAboutStation(const std::string &name_of_the_station) noexcept;
24
25     /**
26     * @brief Добавляет информацию о станции, если станция уже существует, то старая информация
    ↪ удаляется
27     * @param name_of_the_station — название станции, которую нужно добавить
28     * @param station_description — информация о станции
29     */
30     void addInfoAboutStation(const std::string &name_of_the_station, const std::string &
    ↪ station_description) noexcept;
31
32     /**
33     * @brief Удаляет информацию о станции
34     * @param what_station_to_remove — название станции, информацию, о которой нужно удалить
35     */
36     void removeInfoAboutStation(const std::string &what_station_to_remove) noexcept;
37
38     /**
39     * @return Возвращает все станции, к которым присутствует описание
40     * Этот метод используется, для вывода всех станций с описанием, чтобы потом выбрать определённую и
    ↪ удалить
41     */
42     std::vector<std::pair<std::string, std::string>> getAllStations() noexcept;
43
44 private:
45
46     std::map<std::string, std::string> infoAboutStationsInMap;
47 };
48
49 #endif // FILESCCHEDULE_H

```

```

1 #include "stations_info.h"
2
3 std::string StationsInfo::getInfoAboutStation(const std::string &name_of_the_station) noexcept
4 {
5     return infoAboutStationsInMap[name_of_the_station];
6 }
7
8 void StationsInfo::addInfoAboutStation(const std::string &name_of_the_station, const std::
9     ↪ string &station_description) noexcept
10 {
11     infoAboutStationsInMap[name_of_the_station] = station_description;
12 }
13
14 void StationsInfo::removeInfoAboutStation(const std::string &what_station_to_remove) noexcept
15 {
16     infoAboutStationsInMap.erase(infoAboutStationsInMap.find(what_station_to_remove));
17 }
18
19 std::vector<std::pair<std::string, std::string>> StationsInfo::getAllStations() noexcept
20 {
21     std::vector<std::pair<std::string, std::string>> infoAboutStationsInVector;
22     std::pair<std::string, std::string> name_and_value;
23
24     for(auto it_for_map = infoAboutStationsInMap.begin(); it_for_map != infoAboutStationsInMap
25     ↪ .end(); ++it_for_map)
26     {
27         name_and_value.first = it_for_map->first;
28         name_and_value.second = it_for_map->second;
29
30         infoAboutStationsInVector.push_back(name_and_value);
31     }
32
33     return infoAboutStationsInVector;
34 }

```

```

1 #ifndef FILETIMETABLE_H
2 #define FILETIMETABLE_H
3
4 #include <string>
5 #include <vector>
6 #include "exception_of_core/exception_of_core.h"
7
8 /**
9  * @brief Класс отвечающий за работу с информацией связанной с маршрутами
10  */
11 class RoutesInfo
12 {
13 public:
14
15     /**
16      * @param number_of_the_route — номер запрашиваемого маршрута
17      * @return Запрашиваемый маршрут
18      */
19     std::vector<std::string> getRoute(const int number_of_the_route) noexcept;
20
21     /**
22      * @return Возвращает информацию, о том что сколько маршрутов существует на данный момент
23      */
24     int getHowManyRoutes() const noexcept;
25
26     /**
27      * @brief Изменяет название станции в маршруте
28      * @param number_of_the_route — номер маршрута
29      * @param number_of_the_station — номер станции
30      * @param new_marking — новое название станции
31      */
32     void changeStationInRoute(const int number_of_the_route, const int number_of_the_station,
33     ↪ const std::string &new_marking) noexcept;
34
35     /**
36      * @brief Добавляет новую станцию в маршрут
37      * @param number_of_the_route — номер маршрута
38      * @param what_add — название добавляемой станции
39      */

```

```

39     void addStationInRoute(const int number_of_the_route, const std::string &what_add)
    ↪ noexcept;
40
41     /**
42     * @brief Добавляет новый маршрут
43     */
44     void addRoute() noexcept;
45
46     /**
47     * @brief Удаляет станцию из маршрута
48     * @param number_of_the_route — номер маршрута, из которого надо удалить станцию
49     * @param number_of_the_station — номер станции, которую надо удалить
50     */
51     void deleteStationFromRoute(const int number_of_the_route, const int number_of_the_station
    ↪ ) noexcept;
52
53     /**
54     * @brief Удаляет маршрут
55     * @param number_of_the_route — номер маршрута, который нужно удалить
56     */
57     void deleteRoute(const int number_of_the_route) noexcept;
58
59 private:
60
61     std::vector<std::vector<std::string>> many_routes;
62 };
63
64 #endif // FILETIMETABLE_H

```

```

1 #include "routes_info.h"
2
3 std::vector<std::string> RoutesInfo::getRoute(const int number_of_the_route) noexcept
4 {
5     return many_routes[number_of_the_route];
6 }
7
8 int RoutesInfo::getHowManyRoutes() const noexcept
9 {
10    return many_routes.size();
11 }
12
13 void RoutesInfo::changeStationInRoute(const int number_of_the_route, const int
    ↪ number_of_the_station, const std::string &new_marking) noexcept
14 {
15     many_routes[number_of_the_route][number_of_the_station] = new_marking;
16 }
17
18 void RoutesInfo::addStationInRoute(const int number_of_the_route, const std::string &what_add)
    ↪ noexcept
19 {
20     auto it = many_routes.begin() + number_of_the_route;
21
22     std::vector<std::string> route = *it;
23
24     route.push_back(what_add);
25
26     *it = route;
27 }
28
29 void RoutesInfo::addRoute() noexcept
30 {
31     std::vector<std::string> what_add;
32
33     many_routes.push_back(what_add);
34 }
35
36 void RoutesInfo::deleteStationFromRoute(const int number_of_the_route, const int
    ↪ number_of_the_station) noexcept
37 {
38     auto it = many_routes.begin() + number_of_the_route;
39
40     std::vector<std::string> route = *it;
41
42     route.erase(route.begin() + number_of_the_station);
43
44     *it = route;

```

```

45 }
46
47 void RoutesInfo::deleteRoute(const int number_of_the_route) noexcept
48 {
49     many_routes.erase(many_routes.begin() + number_of_the_route);
50 }

```

```

1  #ifndef FILE_H
2  #define FILE_H
3
4  #include <fstream>
5  #include <iostream>
6  #include <string>
7  #include <vector>
8  #include "exception_of_core/exception_of_core.h"
9  #include "routes_info.h"
10 #include "stations_info.h"
11
12 /**
13  * @brief The ParsingInfo class
14  */
15 class ParsingInfo
16 {
17 public:
18
19     /**
20      * @brief Получает на вход строку, парсит её, а потом заполняет информацию о маршрутах
21      * @param info — строка, которую надо парсить
22      * @param routeInfo — то, с чем работают заполнители
23      */
24     void putInfoAboutRoutes(const std::string &info, RoutesInfo &routeInfo) noexcept;
25
26     /**
27      * @brief Получает на вход строку, парсит её, а потом заполняет информацию о станциях
28      * @param info — строка, которую нужно парсить
29      * @param stationInfo — то, с чем работают заполнители
30      */
31     void putInfoAboutStations(const std::string &info, StationsInfo &stationInfo) noexcept;
32
33     /**
34      * @brief Загружает данные из файла и заполняет маршруты и информацию о станциях, путём вызова
35      * ↪ двух пред идущих функций
36      * @param name_of_the_file_with_route — название файла, содержащего информацию о маршрутах
37      * @param name_of_the_file_with_station — название файла, содержащего информацию о станциях
38      * @param routeInfo — то, с чем работают заполнители
39      * @param stationInfo — то, с чем работают заполнители
40      * @exception При неудачном открытии файла бросается MissingFile
41      */
42     void loadFromFile(const std::string &name_of_the_file_with_route, const std::string &
43     ↪ name_of_the_file_with_station, RoutesInfo &routeInfo, StationsInfo &stationInfo);
44
45     /**
46      * @brief Сохраняет изменения
47      * @param name_of_the_file_with_route — название файла, содержащего информацию о маршрутах
48      * @param name_of_the_file_with_station — название файла, содержащего информацию о станциях
49      * @param routeInfo — то, с чем работают заполнители
50      * @param stationInfo — то, с чем работают заполнители
51      */
52     void saveChanges(const std::string &name_of_the_file_with_route, const std::string &
53     ↪ name_of_the_file_with_station, RoutesInfo &routeInfo, StationsInfo &stationInfo);
54
55     /**
56      * @brief Сохраняет изменения проведённые со станциями
57      * @param name_of_the_file_with_route — название файла, содержащего информацию о станциях
58      * @param routeInfo — то, с чем работают заполнители
59      */
60     void saveChangesForRoute(const std::string &name_of_the_file_with_route, RoutesInfo &
61     ↪ routeInfo);
62
63     /**
64      * @brief Сохраняет изменения проведённые с маршрутами
65      * @param name_of_the_file_with_station — название файла, содержащего информацию о маршрутах
66      * @param stationInfo — то, с чем работают заполнители
67      */
68     void saveChangesForStation(const std::string &name_of_the_file_with_station, StationsInfo &
69     ↪ stationInfo);

```

```

65
66 private:
67
68 /**
69  * @brief Функция использующаяся для разделения строки на название станции и информацию о ней
70  * @param blockWithInfo — строка, которую нужно парсить
71  * @return Пара, первый элемент которой название станции, а второй информация о ней
72  */
73 std::pair<std::string, std::string> parsingForMap(const std::string &blockWithInfo)
74 ↪ noexcept;
75 };
76 #endif // FILE_H

```

```

1 #include "parsing_info.h"
2
3 void ParsingInfo::putInfoAboutRoutes(const std::string &info, RoutesInfo &routeInfo) noexcept
4 {
5     std::string station;
6
7     routeInfo.addRoute();
8
9     int number_of_the_route = 0;
10
11     for (auto it = info.begin(); it != info.end(); ++it)
12     {
13         if( *it == '/' )
14         {
15             routeInfo.addRoute();
16
17             routeInfo.addStationInRoute(number_of_the_route, station);
18
19             station = "";
20
21             number_of_the_route++;
22         }
23         else
24         {
25             if ( *it == ',' )
26             {
27                 routeInfo.addStationInRoute(number_of_the_route, station);
28
29                 station = "";
30             }
31             else
32             {
33                 station += *it;
34             }
35         }
36     }
37     routeInfo.addStationInRoute(number_of_the_route, station);
38 }
39
40 void ParsingInfo::putInfoAboutStations(const std::string &info, StationsInfo &stationInfo)
41 ↪ noexcept
42 {
43     std::string blockWithInfo;
44     std::pair<std::string, std::string> pairNameValue;
45
46     for (auto it = info.begin(); it != info.end(); ++it)
47     {
48         if( *it == '/' )
49         {
50             pairNameValue = parsingForMap(blockWithInfo);
51
52             stationInfo.addInfoAboutStation(pairNameValue.first, pairNameValue.second);
53
54             blockWithInfo = "";
55         }
56         else
57         {
58             blockWithInfo += *it;
59         }
60     }
61     pairNameValue = parsingForMap(blockWithInfo);

```

```

62     stationInfo.addInfoAboutStation(pairNameValue.first , pairNameValue.second);
63 }
64 }
65
66 std::pair<std::string , std::string> ParsingInfo::parsingForMap(const std::string &
67 ↪ blockWithInfo) noexcept
68 {
69     Part_of_buffer part = Part_of_buffer::name;
70     std::string name_of_buffer = "";
71     std::string value_of_bufer = "";
72
73     for (auto it = blockWithInfo.begin(); it != blockWithInfo.end(); ++it)
74     {
75         if (*it == '~')
76         {
77             part = Part_of_buffer::value;
78         }
79         else
80         {
81             if (part == Part_of_buffer::name)
82             {
83                 name_of_buffer += *it;
84             }
85             else
86             {
87                 value_of_bufer += *it;
88             }
89         }
90     }
91
92     std::pair<std::string , std::string> pairNameValue;
93
94     pairNameValue.first = name_of_buffer;
95     pairNameValue.second = value_of_bufer;
96
97     return pairNameValue;
98 }
99
100 void ParsingInfo::loadFromFile(const std::string &name_of_the_file_with_route, const std::
101 ↪ string &name_of_the_file_with_station,
102                               RoutesInfo &routeInfo, StationsInfo &stationInfo)
103 {
104     std::ifstream reading_file_with_routes(name_of_the_file_with_route);
105     if (!reading_file_with_routes.is_open())
106     {
107         throw MissingFile(name_of_the_file_with_route);
108     }
109
110     std::string line_from_file;
111     std::getline(reading_file_with_routes, line_from_file);
112
113     putInfoAboutRoutes(line_from_file, routeInfo);
114
115     reading_file_with_routes.close();
116
117
118     std::ifstream reading_file_with_stations(name_of_the_file_with_station);
119     if (!reading_file_with_stations.is_open())
120     {
121         throw MissingFile(name_of_the_file_with_station);
122     }
123
124     std::getline(reading_file_with_stations, line_from_file);
125
126     putInfoAboutStations(line_from_file, stationInfo);
127
128     reading_file_with_stations.close();
129 }
130
131 void ParsingInfo::saveChanges(const std::string &name_of_the_file_with_route, const std::
132 ↪ string &name_of_the_file_with_station, RoutesInfo &routeInfo, StationsInfo &stationInfo)
133 {
134     saveChangesForRoute(name_of_the_file_with_route, routeInfo);

```

```

135     saveChangesForStation(name_of_the_file_with_station, stationInfo);
136 }
137
138 void ParsingInfo::saveChangesForRoute(const std::string &name_of_the_file_with_route,
    ↳ RoutesInfo &routeInfo)
139 {
140     std::ofstream rewriting_file_with_routes(name_of_the_file_with_route);
141
142     std::vector<std::string> route;
143
144     if (routeInfo.getHowManyRoutes() != 0)
145     {
146         route = routeInfo.getRoute(0);
147
148         rewriting_file_with_routes << route[0];
149
150
151         for(auto it = route.begin() + 1; it != route.end(); ++it)
152         {
153             rewriting_file_with_routes << ',' << *it;
154         }
155
156         for(int i = 1; i < routeInfo.getHowManyRoutes(); i++)
157         {
158             route = routeInfo.getRoute(i);
159
160             rewriting_file_with_routes << '/';
161
162             rewriting_file_with_routes << route[0];
163
164             for(auto it = route.begin() + 1; it != route.end(); ++it)
165             {
166                 rewriting_file_with_routes << ',' << *it;
167             }
168         }
169     }
170
171     rewriting_file_with_routes.close();
172 }
173
174 void ParsingInfo::saveChangesForStation(const std::string &name_of_the_file_with_station,
    ↳ StationsInfo &stationInfo)
175 {
176     std::ofstream rewriting_file_with_stations(name_of_the_file_with_station);
177
178     auto all_station = stationInfo.getAllStations();
179
180     if (all_station.size() != 0)
181     {
182         rewriting_file_with_stations << (all_station[0]).first << '~' << (all_station[0]).
    ↳ second;
183
184
185         for(auto it = all_station.begin() + 1; it != all_station.end(); ++it)
186         {
187             rewriting_file_with_stations << '/' << (*it).first << '~' << (*it).second;
188         }
189     }
190
191     rewriting_file_with_stations.close();
192 }

```

```

1  #ifndef STATION_DOES_NOT_EXIST_H
2  #define STATION_DOES_NOT_EXIST_H
3
4  #include <string>
5  #include <sstream>
6  #include <exception>
7
8  /**
9   * @brief Исключение бросается в том случае, если оказывается, что запрашиваемой станции не существует
10   */
11 class StationDoesNotExist : public std::exception
12 {
13 public:
14     StationDoesNotExist() = default;

```

```

15
16 /**
17  * @brief Используется при необходимости передать информацию об ошибке, в том случае, если она в
18  * ↳ название станции
19  * @param requested — название станции, которая не существует, но была запрошена пользователем
20  */
21 explicit StationDoesNotExist(const std::string &requested) : requested_by_customer(
22  * ↳ requested) {}
23
24 /**
25  * @brief Используется при необходимости передать информацию об ошибке, в том случае, если она в
26  * ↳ номере станции
27  * @param requested — номер станции, которая не существует, но была запрошена пользователем
28  */
29 explicit StationDoesNotExist(int requested)
30 {
31     std::stringstream print_int;
32     print_int << requested;
33     requested_by_customer = print_int.str();
34 }
35
36 /**
37  * @return Возвращаем информацию о том, изза— чего было возбужденно исключение
38  */
39 std::string getWhatRequested() noexcept {return requested_by_customer;}
40
41 private:
42
43 /**
44  * @brief Здесь храниться информация о том, изза— чего было возбужденно исключение
45  */
46 std::string requested_by_customer;
47 };
48
49 #endif // STATION_DOES_NOT_EXIST_H

```

```

1 #ifndef ROUTE_DOES_NOT_EXIST_H
2 #define ROUTE_DOES_NOT_EXIST_H
3
4 #include <string>
5 #include <sstream>
6 #include <exception>
7
8 /**
9  * @brief Исключение бросается в том случае, если оказывается, что запрашиваемого маршрута не существует
10  */
11 class RouteDoesNotExist : public std::exception
12 {
13 public:
14
15     RouteDoesNotExist() = default;
16
17     /**
18     * @brief Используется, при необходимости передать информацию об ошибке
19     * @param requested — номер маршрута, который не существует, но был запрошен пользователем
20     */
21     explicit RouteDoesNotExist(int requested) : requested_by_customer(requested) {}
22
23     /**
24     * @return Возвращаем информацию о том, изза— чего было возбужденно исключение
25     */
26     int getWhatRequested() noexcept {return requested_by_customer;}
27
28 private:
29
30     /**
31     * @brief Здесь храниться информация о том, изза— чего было возбужденно исключение
32     */
33     int requested_by_customer;
34 };
35
36 #endif // ROUTE_DOES_NOT_EXIST_H

```

```

1 #ifndef THERE_ARE_NO_ROUTES_H
2 #define THERE_ARE_NO_ROUTES_H

```



```

3
4 #include <exception>
5 #include <string>
6
7 /**
8  * @brief Исключение, которое бросается при неудачном открытии файла, от него можно получить,
9  * ↪ информацию о том, какой файл
10  * запрашивал пользователь, перед появлением исключения
11  */
12 class MissingFile : public std::exception
13 {
14 public:
15     MissingFile() = default;
16
17     /**
18      * @brief Используется, при необходимости передать информацию об ошибке
19      * @param requested — название файла, который хотел открыть пользователь
20      */
21     explicit MissingFile(std::string requested) : requested_by_customer(requested) {}
22
23     /**
24      * @return Возвращаем информацию о том, из-за чего было возбуждено исключение
25      */
26     std::string getWhatRequested() noexcept {return requested_by_customer;}
27 private:
28     std::string requested_by_customer;
29 };
30
31 #endif // THERE_ARE_NO_ROUTES_H

```

```

1 #ifndef EXCEPTION_OF_CORE_H
2 #define EXCEPTION_OF_CORE_H
3
4 #include "route_does_not_exist.h"
5 #include "station_does_not_exist.h"
6 #include "missing_file.h"
7
8 #endif // EXCEPTION_OF_CORE_H

```

```

1 #include <QString>
2 #include <QtTest>
3 #include <core.h>
4
5 const std::string infoAboutRoute = "Devyatkino ,Grazhdansky_Prospekt/Parnas ,Prospekt_
6     ↪ Prosvescheniya";
7 const std::string infoAboutStation = "Devyatkino~5.32-0.00/Grazhdansky_Prospekt~5.30-0.44/
8     ↪ Parnas~5.47-0.00/Prospekt_Prosvescheniya~5.37-0.40";
9
10 /**
11  * @brief Тесты для функциональностей предоставляемых ядром
12  */
13 class Test_for_coreTest : public QObject
14 {
15     Q_OBJECT
16 public:
17     Test_for_coreTest();
18 private Q_SLOTS:
19     void putOfRights();
20
21     void putInfoAboutRoutes();
22
23     void changeStationInRoute();
24
25     void addStationInRoute();
26
27     void deleteStationFromRoute();

```

```

33
34     void addRoute();
35
36     void deleteRoute();
37
38
39     void testingException();
40
41
42     void putInfoAbouteStations();
43
44     void addInfoAboutStation();
45
46     void removeInfoAboutStation();
47
48     void getAllStation();
49 };
50
51 Test_for_coreTest::Test_for_coreTest()
52 {
53 }
54
55 void Test_for_coreTest::putOfRights()
56 {
57     CoreOfInfoAboutMetro startTest;
58
59     QCOMPARE(startTest.getInformationOfTheRights(), Rights_of_customers::user);
60     startTest.putOfRights(Rights_of_customers::administrator);
61     QCOMPARE(startTest.getInformationOfTheRights(), Rights_of_customers::administrator);
62 }
63
64 void Test_for_coreTest::putInfoAbouteRoutes()
65 {
66     CoreOfInfoAboutMetro startTest;
67     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
68
69     std::vector<std::string> route1;
70     route1.push_back("Devyatkino");
71     route1.push_back("Grazhdansky_Prospekt");
72
73     std::vector<std::string> route123 = startTest.getRoute(1);
74
75     std::cout << route123[0] << std::endl;
76
77     QCOMPARE(startTest.getRoute(1), route1);
78
79     std::vector<std::string> route2;
80     route2.push_back("Parnas");
81     route2.push_back("Prospekt_Prosvescheniya");
82
83     QCOMPARE(startTest.getRoute(2), route2);
84 }
85
86 void Test_for_coreTest::changeStationInRoute()
87 {
88     CoreOfInfoAboutMetro startTest;
89     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
90
91     std::vector<std::string> route;
92     route.push_back("Parnas");
93     route.push_back("Prospekt_Prosvescheniya");
94
95     QCOMPARE(startTest.getRoute(2), route);
96
97     std::string new_marking = "Avtovo";
98     startTest.changeStationInRoute(2, 2, new_marking);
99
100     route[1] = new_marking;
101
102     QCOMPARE(startTest.getRoute(2), route);
103 }
104
105 void Test_for_coreTest::addStationInRoute()
106 {
107     CoreOfInfoAboutMetro startTest;
108     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);

```

```

109
110     std::vector<std::string> route;
111     route.push_back("Parnas");
112     route.push_back("Prospekt_Prosvescheniya");
113
114     QCOMPARE(startTest.getRoute(2), route);
115
116     std::string what_to_add = "Avtovo";
117     startTest.addStationInRoute(2, what_to_add);
118
119     route.push_back(what_to_add);
120
121     QCOMPARE(startTest.getRoute(2), route);
122 }
123
124 void Test_for_coreTest::deleteStationFromRoute()
125 {
126     CoreOfInfoAboutMetro startTest;
127     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
128
129     std::vector<std::string> route;
130     route.push_back("Parnas");
131     route.push_back("Prospekt_Prosvescheniya");
132
133     QCOMPARE(startTest.getRoute(2), route);
134
135     startTest.deleteStationFromRoute(2,1);
136
137     route.erase(route.begin());
138
139     QCOMPARE(startTest.getRoute(2), route);
140 }
141
142 void Test_for_coreTest::addRoute()
143 {
144     CoreOfInfoAboutMetro startTest;
145     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
146
147     QCOMPARE(startTest.howManyRoutes(), 2);
148
149     startTest.addRoute();
150
151     QCOMPARE(startTest.howManyRoutes(), 3);
152
153
154     std::string what_to_add = "Why_you_read_this?";
155     startTest.addStationInRoute(3, what_to_add);
156
157     std::vector<std::string> route;
158     route.push_back(what_to_add);
159
160     QCOMPARE(startTest.getRoute(3), route);
161 }
162
163 void Test_for_coreTest::deleteRoute()
164 {
165     CoreOfInfoAboutMetro startTest;
166     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
167
168     QCOMPARE(startTest.howManyRoutes(), 2);
169
170     startTest.deleteRoute(1);
171
172     QCOMPARE(startTest.howManyRoutes(), 1);
173
174     std::vector<std::string> route;
175     route.push_back("Parnas");
176     route.push_back("Prospekt_Prosvescheniya");
177
178     QCOMPARE(startTest.getRoute(1), route);
179 }
180
181 void Test_for_coreTest::testingException()
182 {
183     CoreOfInfoAboutMetro startTest;
184     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);

```

```

185
186     std::string name_first_file = "abraradabra";
187     std::string name_second_file = "ba-ba-duk";
188     QVERIFY_EXCEPTION_THROWN(startTest.loadInfoFromFile(name_first_file, name_second_file),
189         ↪ MissingFile);
189
190     QVERIFY_EXCEPTION_THROWN(startTest.getRoute(56), RouteDoesNotExist);
191     QVERIFY_EXCEPTION_THROWN(startTest.getRoute(0), RouteDoesNotExist);
192
193     QVERIFY_EXCEPTION_THROWN(startTest.getInfoAboutStation("Pargolovo"), StationDoesNotExist);
194     QVERIFY_EXCEPTION_THROWN(startTest.getInfoAboutStation(""), StationDoesNotExist);
195     QVERIFY_EXCEPTION_THROWN(startTest.getInfoAboutStation(1, 124), StationDoesNotExist);
196     QVERIFY_EXCEPTION_THROWN(startTest.getInfoAboutStation(3564, 124), RouteDoesNotExist);
197
198     QVERIFY_EXCEPTION_THROWN(startTest.changeStationInRoute(45, 1, "Pararuram"),
199         ↪ RouteDoesNotExist);
200     QVERIFY_EXCEPTION_THROWN(startTest.changeStationInRoute(1, 2345, "Pararuram"),
201         ↪ StationDoesNotExist);
202 }
203
204 void Test_for_coreTest::putInfoAbouteStations()
205 {
206     CoreOfInfoAboutMetro startTest;
207     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
208
209     std::string name_of_the_station = "Grazhdansky_Prospekt";
210     std::string info_about_station = "5.30-0.44";
211
212     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
213
214     name_of_the_station = "Parnas";
215     info_about_station = "5.47-0.00";
216
217     const int number_of_the_route = 2;
218     const int number_of_the_station = 1;
219
220     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
221     QCOMPARE(startTest.getInfoAboutStation(number_of_the_route, number_of_the_station),
222         ↪ info_about_station);
223
224     name_of_the_station = "Prospekt_Proshchaniya";
225     info_about_station = "5.37-0.40";
226
227     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
228 }
229
230 void Test_for_coreTest::addInfoAboutStation()
231 {
232     CoreOfInfoAboutMetro startTest;
233     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
234
235     std::string name_of_the_station = "Avtovo";
236     std::string info_about_station = "It_is_Avtovo";
237
238     startTest.addInfoAboutStation(name_of_the_station, info_about_station);
239
240     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
241
242     name_of_the_station = "Parnas";
243     const int number_of_the_route = 2;
244     const int number_of_the_station = 1;
245
246     info_about_station = "5.47-0.00";
247
248     QCOMPARE(startTest.getInfoAboutStation(number_of_the_route, number_of_the_station),
249         ↪ info_about_station);
250     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
251
252     info_about_station = "It_is_Parnas";
253
254     startTest.addInfoAboutStation(number_of_the_route, number_of_the_station,
255         ↪ info_about_station);
256
257     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
258 }

```

```

255
256 void Test_for_coreTest::removeInfoAboutStation()
257 {
258     CoreOfInfoAboutMetro startTest;
259     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
260
261     std::string name_of_the_station = "Parnas";
262     std::string info_about_station = "5.47-0.00";
263
264     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
265
266     std::string what_station_to_remove = "Parnas";
267     startTest.removeInfoAboutStation(what_station_to_remove);
268
269     QVERIFY_EXCEPTION_THROWN(startTest.getInfoAboutStation(what_station_to_remove),
270     ↪ StationDoesNotExist);
271
272     name_of_the_station = "Prospekt_Prosvescheniya";
273     info_about_station = "5.37-0.40";
274
275     QCOMPARE(startTest.getInfoAboutStation(name_of_the_station), info_about_station);
276
277     const int number_of_the_station = 4;
278
279     startTest.removeInfoAboutStation(number_of_the_station);
280
281     QVERIFY_EXCEPTION_THROWN(startTest.getInfoAboutStation(name_of_the_station),
282     ↪ StationDoesNotExist);
283 }
284 void Test_for_coreTest::getAllStation()
285 {
286     CoreOfInfoAboutMetro startTest;
287     startTest.putInfoAboutMetro(infoAboutRoute, infoAboutStation);
288
289     auto AllStation = startTest.getAllStationsWhichHaveDescription();
290
291     std::string what_get = (AllStation[2]).first;
292     std::string what_expect = "Parnas";
293
294     QCOMPARE(what_get, what_expect);
295
296     what_get = (AllStation[2]).second;
297     what_expect = "5.47-0.00";
298
299     QCOMPARE(what_get, what_expect);
300
301     what_get = (AllStation[3]).first;
302     what_expect = "Prospekt_Prosvescheniya";
303
304     QCOMPARE(what_get, what_expect);
305
306     what_get = (AllStation[3]).second;
307     what_expect = "5.37-0.40";
308
309     QCOMPARE(what_get, what_expect);
310 }
311
312 QTEST_APPLESS_MAIN(Test_for_coreTest)
313
314 #include "tst_test_for_coretest.moc"

```

```

1 #include <app.h>
2
3 int main()
4 {
5     ConsoleForTimetable start;
6
7     return 0;
8 }

```

```

1 #ifndef CONSOLE_FOR_TIMETABLE_H
2 #define CONSOLE_FOR_TIMETABLE_H
3

```

```

4 #include "workwithinfo/work_with_routes.h"
5 #include "workwithinfo/work_with_stations.h"
6
7 /**
8  * @brief Класс служащий для работы с ядром через консоль
9  */
10 class ConsoleForTimetable
11 {
12 public:
13
14     ConsoleForTimetable();
15
16 private:
17
18     /**
19      * @brief Вывод в консоль меню с предоставляемыми возможностями
20      * @return Возвращает "fals", когда пользователь хочет выйти из приложения
21      */
22     bool menu();
23
24     /**
25      * @brief Выдача прав администратора или обычного пользователя
26      */
27     void definitionOfAdministrator();
28
29     /**
30      * @return Символ считанный из консоли
31      */
32     char getCharFromConsole();
33
34
35     /// Методы для администратора доступны( только в режиме администратора):
36
37     /**
38      * @brief Все изменения совершенные пользователем записываются в файл
39      */
40     void saveChanges();
41
42     void loadFromFile();
43
44     CoreOfInfoAboutMetro core;
45
46     WorkWithRoutes route_info;
47
48     WorkWithStations stations_info;
49 };
50
51 #endif // CONSOLE_FOR_TIMETABLE_H

```

```

1 #ifndef WORKWITHSTATIONS_H
2 #define WORKWITHSTATIONS_H
3
4 #include "work_with_info.h"
5
6 /**
7  * @brief Класс, отвечающий за обработку запросов связанных со станциями, с помощью консоли
8  */
9 class WorkWithStations : public WorkWithInfo
10 {
11 public:
12
13     /**
14      * @brief В зависимости от запрашиваемой станции, печатает информацию о ней
15      */
16     void informationAboutStation(CoreOfInfoAboutMetro &core);
17
18     /// Методы для администратора доступны( только в режиме администратора):
19
20     /**
21      * @brief Предоставляет пользователю возможность изменения информации о станции
22      * @brief Добавить информацию о станции, удалить её
23      */
24     void changeInfoAboutStation(CoreOfInfoAboutMetro &core);
25
26     /**
27      * @brief Добавить информацию о станции

```

```

28     */
29     void addOrChangeInformationAboutStation (CoreOfInfoAboutMetro &core);
30
31     /**
32     * @brief Удалить информацию о станции
33     */
34     void removeInformationAboutStation (CoreOfInfoAboutMetro &core);
35 };
36
37 #endif // WORKWITHSTATIONS_H

```

```

1 #include "work_with_stations.h"
2
3 void WorkWithStations::informationAboutStation (CoreOfInfoAboutMetro &core)
4 {
5     unsigned how_many_routes = 0;
6     try
7     {
8         how_many_routes = core.howManyRoutes();
9
10        if(how_many_routes == 0)
11        {
12            std::cout << std::endl << "_At_the_moment_there_are_no_routes,_contact_the_
→ administrator_for_help" << std::endl;
13            std::cout << std::endl << "_Press_Enter..." << std::endl << std::endl;
14            std::cin.get();
15            return;
16        }
17
18
19        std::cout << "_What_route_are_you_interested_in?_(Enter_number:_1-" <<
→ how_many_routes << ')<< std::endl << std::endl << "—>";
20        int choice_of_the_route = getIntFromConsole();
21        std::cout << std::endl;
22
23        std::vector<std::string> output_for_console = core.getRoute(choice_of_the_route);
24
25        displayRoute(output_for_console);
26
27        std::cout << "_What_station_are_you_interested_in?" << std::endl << std::endl << "—>"
→ ;
28        int choice_number_of_the_station = getIntFromConsole();
29        std::cout << std::endl;
30
31        try
32        {
33            std::cout << std::endl << "_Information_about_the_station:" << std::endl << std::
→ endl << ' '
34            << core.getInfoAboutStation(choice_of_the_route,
→ choice_number_of_the_station) << std::endl;
35        }
36        catch (StationDoesNotExist& exception)
37        {
38            std::cout << "_The_station_" << ' ' << exception.whatRequested() << ' ' << "_
→ does_not_exist";
39        }
40    }
41    catch (RouteDoesNotExist& exception)
42    {
43        std::cout << "_The_route_"
44        << ' ' << exception.whatRequested() << ' ' << "_does_not_exist" << std::
→ endl
45        << std::endl << "_At_the_moment_there_are_1-" << how_many_routes << "_routes
→ " << std::endl
46        << std::endl << "_Enter_number_of_the_route,_for_example:_1" << std::endl;
47    }
48    std::cout << std::endl << std::endl << "_Press_Enter..." << std::endl;
49    std::cin.get();
50 }
51
52 void WorkWithStations::changeInfoAboutStation (CoreOfInfoAboutMetro &core)
53 {
54     if (core.getInformationOfTheRights() == Rights_of_customers::user)
55     {
56         return;
57     }

```

```

58     try
59     {
60         std::cout << "_1.Add_or_change_information_about_station" << std::endl
61             << "_2.Remove_information_about_station" << std::endl << std::endl << "—>";
62
63         int choice_of_action = getIntFromConsole();
64
65         std::cout << std::endl;
66
67         switch(choice_of_action)
68         {
69             case 1:
70             {
71                 addOrChangeInformationAboutStation(core);
72                 break;
73             }
74             case 2:
75             {
76                 removeInformationAboutStation(core);
77                 break;
78             }
79             default:
80             {
81                 std::cout << "_You_have_entered_something_unclear" << std::endl;
82                 break;
83             }
84         }
85         std::cout << std::endl << "_The_changes_have_been_well_accepted" << std::endl;
86     }
87     catch(StationDoesNotExist& exception)
88     {
89         std::cout << std::endl << "_The_station_with_number_"
90             << "'_'<<<exception.whatRequested()<<<'_'<< "_does_not_exist" << std::
91     ↪ endl;
92     }
93     catch(RouteDoesNotExist& exception)
94     {
95         if(core.howManyRoutes() == 0)
96         {
97             std::cout << std::endl << "_At_the_moment_there_are_no_routes,_contact_the_
98     ↪ administrator_for_help" << std::endl;
99             std::cout << std::endl << "_Press_Enter..." << std::endl << std::endl;
100             std::cin.get();
101             return;
102         }
103         std::cout << "_The_route_"
104             << "'_'<<<exception.whatRequested()<<<'_'<< "_does_not_exist" << std::
105     ↪ endl
106             << std::endl << "_At_the_moment_there_are_1—" << core.howManyRoutes() << "_
107     ↪ routes" << std::endl
108             << std::endl << "_Enter_number_of_the_route,_for_example:_1" << std::endl;
109     }
110     std::cout << std::endl << std::endl << "_Press_Enter..." << std::endl << std::endl;
111     std::cin.get();
112 }
113 void WorkWithStations::addOrChangeInformationAboutStation(CoreOfInfoAboutMetro &core)
114 {
115     if(core.howManyRoutes() == 0)
116     {
117         std::cout << std::endl << "_At_the_moment_there_are_no_routes,_contact_the_
118     ↪ administrator_for_help" << std::endl;
119         std::cout << std::endl << "_Press_Enter..." << std::endl << std::endl;
120         std::cin.get();
121         return;
122     }
123     std::cout << "_What_route_are_you_interested_in?_(Enter_number:_1—"
124         << core.howManyRoutes() << ')_' << std::endl << std::endl << "—>";
125
126     int choice_of_the_route = getIntFromConsole();
127
128     std::cout << std::endl;

```



```

129
130     std::vector<std::string> output_for_console = core.getRoute(choice_of_the_route);
131
132     displayRoute(output_for_console);
133
134     std::cout << std::endl << "_Which_information_about_station_do_you_want_to_change?" << std
    ↪ ::endl << std::endl << "—>";
135
136     int choice_number_of_the_station = getIntFromConsole();
137
138     std::cout << std::endl << "_What_is_known_about_the_station?" << std::endl << std::endl <<
    ↪ "—>";
139     std::string station_description;
140     std::getline(std::cin, station_description);
141
142     core.addInfoAboutStation(choice_of_the_route, choice_number_of_the_station,
    ↪ station_description);
143 }
144
145 void WorkWithStations::removeInformationAboutStation(CoreOfInfoAboutMetro &core)
146 {
147     auto AllItemFromTimetable = core.getAllStationsWhichHaveDescription();
148     for(unsigned i = 0; i < AllItemFromTimetable.size(); i++)
149     {
150         std::cout << ' ' << i+1 << ' ' << (AllItemFromTimetable[i]).first << ":_:" <<
151             (AllItemFromTimetable[i]).second << std::endl;
152     }
153
154     std::cout << std::endl << "_What_do_you_want_to_remove?" << std::endl << std::endl << "—>
    ↪ ";
155
156     int number_of_what_remove = getIntFromConsole();
157
158     try
159     {
160         core.removeInfoAboutStation(number_of_what_remove);
161     }
162     catch(StationDoesNotExist& exception)
163     {
164         std::cout << std::endl << "_The_station_with_number_"
165             << ' ' << exception.whatRequested() << ' ' << "_does_not_exist" << std::
    ↪ endl;
166     }
167 }

```

```

1  #ifndef WORKWITHROUTES_H
2  #define WORKWITHROUTES_H
3
4  #include "work_with_info.h"
5
6  /**
7   * @brief Класс, отвечающий за обработку запросов связанных с маршрутами, с помощью консоли
8   */
9  class WorkWithRoutes : public WorkWithInfo
10 {
11 public:
12
13     /**
14     * @brief Предоставляет информацию о запрашиваемом маршруте
15     */
16     void routeInformation(CoreOfInfoAboutMetro &core);
17
18     /// Методы для администратора доступны( только в режиме администратора):
19
20     /**
21     * @brief Предоставляет пользователю возможность изменения маршрутов
22     * Изменить маршрут, удалить маршрут, добавить маршрут
23     */
24     void changeItinerarys(CoreOfInfoAboutMetro &core);
25
26     /**
27     * @brief Предоставляет пользователю возможность изменить конкретный маршрут
28     * Добавить, удалить и переименовать станцию
29     */
30     void changeRoute(CoreOfInfoAboutMetro &core);
31

```

```

32     /**
33      * @brief Добавляет новый маршрутпустой() с номером, на один больше чем последний существующий
34      */
35     void addRoute(CoreOfInfoAboutMetro &core);
36
37     /**
38      * @brief Удаляет маршрут с указанным номером, причём маршруты чей номер больше указанного
39      * ↪ сдвигаются вниз
40      */
41     void deleteRoute(CoreOfInfoAboutMetro &core);
42 };
43 #endif // WORKWITHROUTES_H

```

```

1  #include "work_with_routes.h"
2
3  void WorkWithRoutes::routeInformation(CoreOfInfoAboutMetro &core)
4  {
5      unsigned how_many_routes;
6      try
7      {
8          how_many_routes = core.howManyRoutes();
9
10         if(how_many_routes == 0)
11         {
12             std::cout << std::endl << "_At_the_moment_there_are_no_routes,_contact_the_"
13             ↪ administrator_for_help" << std::endl;
14             std::cout << std::endl << "_Press_Enter..." << std::endl << std::endl;
15             std::cin.get();
16             return;
17         }
18
19         std::cout << "_What_route_are_you_interested_in?_(Enter_number:_1-" <<
20         ↪ how_many_routes << ' ' << std::endl << std::endl << "—>";
21         int choice_of_the_route = getIntFromConsole();
22         std::cout << std::endl;
23
24         std::vector<std::string> output_for_console = core.getRoute(choice_of_the_route);
25
26         displayRoute(output_for_console);
27     }
28     catch(RouteDoesNotExist& exception)
29     {
30         std::cout << "_The_route_"
31         ↪ << "'_'<<<exception.whatRequested()<<<_'_' << "_does_not_exist" << std::
32         ↪ endl
33         << std::endl << "_At_the_moment_there_are_1-" << how_many_routes << "_routes" <<
34         ↪ std::endl
35         << std::endl << "_Enter_number_of_the_route,_for_example:_1" << std::endl;
36     }
37     std::cout << std::endl << "_Press_Enter..." << std::endl << std::endl;
38     std::cin.get();
39 }
40
41 void WorkWithRoutes::changeItinerarys(CoreOfInfoAboutMetro &core)
42 {
43     if (core.getInformationOfTheRights() == Rights_of_customers::user)    /// Пользователь не
44     ↪ сможет вызвать метод, если он не админ
45     {
46         return;
47     }
48
49     std::cout << "_What_do_you_want_to_do_with_route_table?_";
50
51     unsigned how_many_routes = 0;
52     how_many_routes = core.howManyRoutes();
53
54     if(how_many_routes == 0)
55     {
56         std::cout << std::endl << "_At_the_moment_there_are_no_routes" << std::endl << std::
57         ↪ endl;
58     }
59     else
60     {
61         std::cout << "(There_are_routes:_1-" << how_many_routes << ' ' << std::endl;
62     }
63 }

```

```

57
58     std::cout << "_1.Add_route" << std::endl
59         << "_2.Change_route" << std::endl
60         << "_3.Delete_route" << std::endl << std::endl << "—>";
61
62     int choice_action_with_route_table = getIntFromConsole();
63
64     switch(choice_action_with_route_table)
65     {
66     case 1:
67     {
68         addRoute(core);
69         break;
70     }
71     case 2:
72     {
73         changeRoute(core);
74         break;
75     }
76     case 3:
77     {
78         deleteRoute(core);
79         break;
80     }
81     default:
82     {
83         std::cout << std::endl << "_You_have_entered_something_unclear" << std::endl;
84         break;
85     }
86     }
87     std::cout << std::endl << "_Press_Enter..." << std::endl << std::endl;
88     std::cin.get();
89 }
90
91
92 void WorkWithRoutes::addRoute(CoreOfInfoAboutMetro &core)
93 {
94     std::cout << std::endl << "_Was_created_the_route_" << core.addRoute() << std::endl;
95 }
96
97 void WorkWithRoutes::deleteRoute(CoreOfInfoAboutMetro &core)
98 {
99     std::cout << std::endl << "_Which_route_you_want_to_delete" << std::endl << std::endl << "
100     ↪ —>";
101     int choice_route = getIntFromConsole();
102
103     try
104     {
105         core.deleteRoute(choice_route);
106         std::cout << std::endl << "_The_route_" << choice_route << "_was_successfully_deleted"
107         ↪ << std::endl;
108     }
109     catch(RouteDoesNotExist& exception)
110     {
111         std::cout << std::endl << "_The_route_"
112         << "'_'<<<exception.whatRequested()<<<'_' << "_does_not_exist" << std::
113         ↪ endl;
114     }
115     std::cout << std::endl;
116 }
117
118 void WorkWithRoutes::changeRoute(CoreOfInfoAboutMetro &core)
119 {
120     bool how_successful_changes = 1;
121     std::cout << std::endl << "_Which_do_route_you_want_to_change?" << std::endl << std::endl
122     ↪ << "—>";
123     int choice_route = getIntFromConsole();
124     std::cout << std::endl;
125
126     try
127     {
128         std::vector<std::string> output_for_console = core.getRoute(choice_route);
129
130         displayRoute(output_for_console);
131
132         std::cout << "_What_do_you_want?" << std::endl

```

```

129         << "_1.Add_station" << std::endl
130         << "_2.Change_station" << std::endl
131         << "_3.Delete_station" << std::endl
132         << std::endl << "—>";
133     int choice_action_with_rrote = getIntFromConsole();
134
135     switch (choice_action_with_rrote)
136     {
137     case 1:
138     {
139         std::cout << "_What_station_do_you_want_to_add?" << std::endl
140         << "__(the_station_will_be_added_to_the_end_of_the_branch)" << std::endl <<
141         ↪ std::endl << "—>";
142
143         std::string what_to_add;
144         std::getline(std::cin, what_to_add);
145         std::cout << std::endl << std::endl;
146
147         core.addStationInRoute(choice_route, what_to_add);
148         break;
149     }
150     case 2:
151     {
152         std::cout << "_What_station_do_you_want_to_change?" << std::endl << std::endl << "
153         ↪ —>";
154         int choice_station = getIntFromConsole();
155
156         std::cout << std::endl << "_What_do_you_want_to_put_in_replacement?" << std::endl
157         ↪ << std::endl << "—>";
158         std::string what_to_replace;
159         std::getline(std::cin, what_to_replace);
160         std::cout << std::endl << std::endl;
161
162         core.changeStationInRoute(choice_route, choice_station, what_to_replace);
163         break;
164     }
165     case 3:
166     {
167         std::cout << "_What_station_do_you_want_to_delete?" << std::endl << std::endl << "
168         ↪ —>";
169         int choice_station = getIntFromConsole();
170
171         core.deleteStationFromRoute(choice_route, choice_station);
172         break;
173     }
174     default:
175     {
176         std::cout << std::endl << "_You_have_entered_something_unclear" << std::endl;
177         how_successful_changes = 0;
178         break;
179     }
180 }
181 catch (RouteDoesNotExist& exception)
182 {
183     std::cout << "_The_route_"
184     << "'_'<<<exception.whatRequested()<<<'_'<< "_does_not_exist" << std::
185     ↪ endl;
186     how_successful_changes = 0;
187 }
188 catch (StationDoesNotExist& exception)
189 {
190     std::cout << std::endl << "_The_station_with_number_"
191     << "'_'<<<exception.whatRequested()<<<'_'<< "_does_not_exist" << std::
192     ↪ endl;
193     how_successful_changes = 0;
194 }
195
196 if (how_successful_changes == 1)
197 {
198     std::cout << std::endl << "_The_changes_have_been_well_accepted" << std::endl;
199 }
200 }

```

```

1 #ifndef WORKWITHINFO_H
2 #define WORKWITHINFO_H

```

```

3
4 #include <core.h>
5 #include <algorithm>
6
7 /**
8  * @brief Абстрактный класс, содержащий несколько функций помогающих при работе с пользователем
9  */
10 class WorkWithInfo
11 {
12 public:
13     WorkWithInfo() = default;
14
15     /**
16      * @return Число считанное из консоли
17      */
18     int getIntFromConsole();
19
20     /**
21      * @brief Удобный вывод маршрута в консоль
22      * @param output_for_console – что за маршрут нужно вывести
23      */
24     void displayRoute(std::vector<std::string> &output_for_console);
25
26     /**
27      * @brief Так как наследники должны иметь свой
28      */
29     virtual ~WorkWithInfo() {}
30
31     /// Они мне не нужны, но по правилу: переопределил деструктор, переопределяй и эти конструкторы
32     /// Конструктор копирования
33     WorkWithInfo(const WorkWithInfo&) = default;
34     /// Копирующее присваивание
35     WorkWithInfo& operator= (const WorkWithInfo&) = default;
36 };
37
38 #endif // WORKWITHINFO_H

```

```

1 #include "work_with_info.h"
2
3 int WorkWithInfo::getIntFromConsole()
4 {
5     int number;
6
7     std::cin >> number;
8     std::cin.clear();
9     std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n'); /// Мешает считать
    ↪ кучу символовведь( нам нужен один)
10
11     return number;
12 }
13
14 void WorkWithInfo::displayRoute(std::vector<std::string> &output_for_console)
15 {
16     for(unsigned i = 0; i < output_for_console.size(); i++)
17     {
18         std::cout << ' ' << i+1 << ' ' << output_for_console[i] << std::endl; /// Выводится в
    ↪ виде: 1.Parnas                                     2.
19     }
    ↪ Prospekt Prosvescheniya
20     std::cout << std::endl;
21 }

```

```

1 #include "mainwindow.h"
2 #include <QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication a(argc, argv);
7
8     MainWindow w;
9     w.show();
10
11     return a.exec();
12 }

```

```

1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <QMessageBox>
6 #include <QPushButton>
7 #include <QWidget>
8 #include <QScrollBar>
9 #include <QVBoxLayout>
10 #include <QDialog>
11 #include <core.h>
12 #include "dialogaboutaddingstationinroute.h"
13 #include "dialogaboutdeletingroute.h"
14 #include "dialogaboutchangingnameofthestations.h"
15 #include "dialogaboutdeletingstation.h"
16 #include "dialogaboutaddinginfoaboutstation.h"
17 #include "dialogaboutdeletinginfoaboutstation.h"
18
19 namespace Ui {
20 class MainWindow;
21 }
22
23 class MainWindow : public QMainWindow
24 {
25     Q_OBJECT
26
27     CoreOfInfoAboutMetro core;
28
29     QVector<QVector<QPushButton*>> stations_buttons_vector;
30     QVector<QPushButton*> routes_buttons;
31
32     QVBoxLayout* stations_layout;
33     QVBoxLayout* routes_layout;
34
35     int index;
36
37     int number_of_the_route;
38     int number_of_the_station;
39     QString string_with_info;
40
41 public:
42     explicit MainWindow(QWidget *parent = 0);
43     ~MainWindow();
44
45 private:
46     Ui::MainWindow *ui;
47
48
49 private slots:
50     void showStations();
51     void showInfoAboutStation();
52     void deleteRouteSlot();
53     void changeNameOfStation();
54     void addStation();
55     void deletingStation();
56     void addInformationAboutStation();
57     void deleteInfoAboutStation();
58
59     void on_action_2_triggered();
60     void on_action_4_triggered();
61     void on_action_9_triggered();
62     void on_action_5_triggered();
63     void on_action_6_triggered();
64     void on_action_7_triggered();
65     void on_action_10_triggered();
66 };
67
68 #endif // MAINWINDOW_H

```

```

1 #include "mainwindow.h"
2 #include "ui_mainwindow.h"
3
4 MainWindow::MainWindow(QWidget *parent) :
5     QMainWindow(parent),
6     ui(new Ui::MainWindow)

```

```

7 {
8     ui->setupUi(this);
9     this->setWindowTitle("Информация_о_маршрутах_и_станциях");
10    ui->textBrowser->setText("");
11    ui->textBrowser->setStyleSheet("font-size:_16px;");
12    this->setMinimumSize(800,400);
13
14    ui->scrollAreaForStations->setFixedWidth(250);
15    ui->scrollAreaForRoutes->setFixedWidth(200);
16
17
18    try
19    {
20        core.loadInfoFromFile("metro_Saint-Petersburg_route_info.txt", "metro_Saint-
21        ↪ Petersburg_station_info.txt");
22    }
23    catch (MissingFile&)
24    {
25        QMessageBox* file_does_not_exist = new QMessageBox;
26        file_does_not_exist->setWindowTitle("Отсутствуют_Файлы");
27        file_does_not_exist->setText("Не_обнаружены_файлы_с_информацией_ожидались\n"
28        ↪ "metro_Saint-Petersburg_route_info.txt_и_metro_Saint-
29        ↪ Petersburg_station_info.txt");
30        file_does_not_exist->show();
31
32        core.addRoute();
33        core.addStationInRoute(1,"Парнас");
34        core.addStationInRoute(1,"Вовсе_не_Парнас");
35        core.addStationInRoute(1,"Супер_Парнас");
36        core.addInfoAboutStation("Супер_Парнас","Это_самый_лучший_Парнас_в_мире");
37        core.addInfoAboutStation("Вовсе_не_Парнас","Это_совсем_не_Парнас");
38        core.addRoute();
39        core.addStationInRoute(2,"Дыбенко");
40        core.addStationInRoute(2,"Дыбенко");
41        core.addStationInRoute(2,"Пацаны_с_Дыбенко");
42        core.addInfoAboutStation("Пацаны_с_Дыбенко","ДЫБЕНКООООО");
43        core.addInfoAboutStation("Дыбенко","Дыбена_-_это_самый_благоустроенный_район_города_
44        ↪ СанктПетербург-,
45        ↪ "в_котором_живёт_элита_общества");
46
47        statusBar()->showMessage("Файлы_не_найлены");
48    }
49
50    stations_layout = new QVBoxLayout;
51    QWidget* stations = new QWidget;
52
53    routes_layout = new QVBoxLayout;
54    QWidget* routes = new QWidget;
55
56    for (int i = 0; i < core.howManyRoutes(); i++)
57    {
58        std::stringstream print_int;
59        print_int << i + 1;
60
61        std::string name_of_the_route = "Маршрут_№" + print_int.str();
62
63        QPushButton* route_button = new QPushButton(name_of_the_route.c_str(), this);
64
65        routes_buttons.push_back(route_button);
66
67        routes_buttons[i]->setFixedSize(150,30);
68
69        routes_layout->addWidget(routes_buttons[i]);
70
71        routes_buttons[i]->show();
72
73        routes_buttons[i]->setProperty("index", i);
74
75        connect(routes_buttons[i], SIGNAL(clicked()), this, SLOT(showStations()));
76
77        QVector<QPushButton*> stations_buttons;
78
79        std::vector<std::string> route = core.getRoute(i + 1);

```

```

80         for(unsigned int j = 0; j < route.size(); j++)
81         {
82             QPushButton* station_button = new QPushButton((route[j]).c_str(), this);
83
84             stations_buttons.push_back(station_button);
85             stations_buttons[j]->hide();
86             stations_buttons[j]->setFixedSize(175,25);
87             stations_buttons[j]->setProperty("name",(route[j]).c_str());
88
89             stations_layout->addWidget(stations_buttons[j]);
90
91             connect(stations_buttons[j], SIGNAL(clicked()), this, SLOT(
92 → showInfoAboutStation()));
93         }
94
95         stations_buttons_vector.push_back(stations_buttons);
96     }
97
98     stations->setLayout(stations_layout);
99     ui->scrollAreaForStations->setWidget(stations);
100
101     routes->setLayout(routes_layout);
102     ui->scrollAreaForRoutes->setWidget(routes);
103 }
104
105 void MainWindow::showStations()
106 {
107     for(int i = 0; i < core.howManyRoutes(); i++)
108     {
109         std::vector<std::string> route = core.getRoute(i + 1);
110
111         for(unsigned int j = 0; j < route.size(); j++)
112         {
113             (stations_buttons_vector[i][j])->hide();
114         }
115     }
116
117     QPushButton *button = qobject_cast<QPushButton*>(sender());
118     QVariant index = button->property("index");
119
120     int number_of_the_route = index.toInt() + 1;
121
122     std::vector<std::string> route = core.getRoute(number_of_the_route);
123
124     for(unsigned int j = 0; j < route.size(); j++)
125     {
126         (stations_buttons_vector[number_of_the_route - 1][j])->show();
127     }
128
129     std::stringstream print_int;
130     print_int << number_of_the_route;
131     std::string new_name = "Маршрут №" + print_int.str();
132     statusBar()->showMessage(new_name.c_str());
133 }
134
135 void MainWindow::showInfoAboutStation()
136 {
137     QPushButton *button = qobject_cast<QPushButton*>(sender());
138     QVariant name = button->property("name");
139
140     QString qt_name_of_the_stations = name.toString();
141     std::string name_of_the_station = qt_name_of_the_stations.toStdString();
142
143     std::string name_of_the_label_std = "";
144     QString name_of_the_label_qt;
145
146     try
147     {
148         name_of_the_label_std += core.getInfoAboutStation(name_of_the_station);
149         name_of_the_label_qt = name_of_the_label_std.c_str();
150     }
151     catch (StationDoesNotExist&)

```



```

155     {
156         name_of_the_label_qt = "Станция_не_найдена";
157     }
158
159     ui->textBrowser->setText(name_of_the_label_qt);
160
161     QString toStatusBar = "Станция: _" + qt_name_of_the_stations;
162
163     statusBar()->showMessage(toStatusBar);
164 }
165
166 MainWindow::~MainWindow()
167 {
168     delete ui;
169 }
170
171 void MainWindow::on_action_2_triggered()
172 {
173     core.addRoute();
174     int i = core.howManyRoutes() - 1;
175
176     std::stringstream print_int;
177     print_int << i + 1;
178     std::string name_of_the_route = "Маршрут_№" + print_int.str();
179     QPushButton* route_button = new QPushButton(name_of_the_route.c_str(), this);
180     route_button->setFixedSize(150,30);
181     routes_buttons.push_back(route_button);
182     routes_layout->addWidget(route_button);
183     route_button->show();
184     route_button->setProperty("index", i);
185     connect(route_button, SIGNAL(clicked()), this, SLOT(showStations()));
186
187     QVector<QPushButton*> stations_buttons;
188     stations_buttons_vector.push_back(stations_buttons);
189
190     std::string new_name = "Добавлен: _";
191     new_name += name_of_the_route;
192     statusBar()->showMessage(new_name.c_str());
193 }
194
195 void MainWindow::on_action_4_triggered()
196 {
197     DialogAboutDeletingRoute* dialog_about_delete_route = new DialogAboutDeletingRoute(&core
198     → ,&index, this);
199     dialog_about_delete_route->show();
200 }
201
202 void MainWindow::deleteRouteSlot()
203 {
204     std::vector<std::string> route = core.getRoute(index+1);
205     for(unsigned int j = 0; j < route.size(); j++)
206     {
207         delete(stations_buttons_vector[index][j]);
208     }
209
210     delete(routes_buttons[core.howManyRoutes()-1]);
211
212     for(int i = 0; i < core.howManyRoutes()-1; i++)
213     {
214         std::stringstream print_int;
215         print_int << i + 1;
216         std::string name_of_the_route = "Маршрут_№" + print_int.str();
217         routes_buttons[i]->setText(name_of_the_route.c_str());
218         routes_buttons[i]->setProperty("index", i);
219     }
220
221     if (stations_buttons_vector.size() > index)
222     {
223         stations_buttons_vector.remove(index);
224     }
225
226     core.deleteRoute(index+1);
227
228     std::stringstream print_int;
229     print_int << index + 1;
230     std::string new_name = "Удалён: _Маршрут_№" + print_int.str();

```

```

230     statusBar()->showMessage(new_name.c_str());
231 }
232
233 void MainWindow::on_action_6_triggered()
234 {
235     DialogAboutChangingNameOfTheStations* dialog_about_changing_name_of_stations =
236         new DialogAboutChangingNameOfTheStations(&core,&number_of_the_route,
237             &number_of_the_station,&string_with_info,this)
238     ↪ ;
239     dialog_about_changing_name_of_stations->show();
240 }
241
242 void MainWindow::changeNameOfStation()
243 {
244     core.changeStationInRoute(number_of_the_route + 1, number_of_the_station + 1,
245         string_with_info.toStdString());
246     QString old_name = (stations_buttons_vector[number_of_the_route][number_of_the_station])->
247     ↪ text();
248     (stations_buttons_vector[number_of_the_route][number_of_the_station])->setText(
249     ↪ string_with_info);
250     stations_buttons_vector[number_of_the_route][number_of_the_station]->
251     ↪ setProperty("name",string_with_info);
252     QString new_name = "Станция:_" + old_name + "_переименована_в_" + string_with_info;
253     statusBar()->showMessage(new_name);
254 }
255
256 void MainWindow::on_action_5_triggered()
257 {
258     DialogAboutAddingStationInRoute* dialog_about_adding_station_in_route = new
259     ↪ DialogAboutAddingStationInRoute(&core,&index,&string_with_info,this);
260     dialog_about_adding_station_in_route->show();
261 }
262
263 void MainWindow::addStation()
264 {
265     core.addStationInRoute(index+1,string_with_info.toStdString());
266     QPushButton* station_button = new QPushButton(string_with_info, this);
267     stations_buttons_vector[index].push_back(station_button);
268     station_button->hide();
269     station_button->setFixedSize(175,25);
270     station_button->setProperty("name",(string_with_info));
271     stations_layout->addWidget(station_button);
272     connect(station_button, SIGNAL(clicked()), this, SLOT(showInfoAboutStation()));
273
274     QString new_name = "Добавлена_станция:_" +
275     ↪ new_name = new_name + string_with_info;
276     statusBar()->showMessage(new_name);
277 }
278
279 void MainWindow::on_action_7_triggered()
280 {
281     DialogAboutDeletingStation* dialog_about_deleting_station =
282     ↪ new DialogAboutDeletingStation(&core,&number_of_the_route,
283         &number_of_the_station,this);
284     dialog_about_deleting_station->show();
285 }
286
287 void MainWindow::deletingStation()
288 {
289     QString old_name = (stations_buttons_vector[number_of_the_route][number_of_the_station])->
290     ↪ text();
291     core.deleteStationFromRoute(number_of_the_route+1,number_of_the_station+1);
292     delete(stations_buttons_vector[number_of_the_route][stations_buttons_vector[
293     ↪ number_of_the_route].size()-1]);
294     std::vector<std::string> route = core.getRoute(number_of_the_route+1);
295     QString name_of_station;
296     for(unsigned int j = 0; j < route.size(); j++)

```

```

301     {
302         name_of_station = (route[j]).c_str();
303         stations_buttons_vector[number_of_the_route][j]->setText(name_of_station);
304     }
305
306     QString information_about_deleting = "Удалена станция: _";
307     information_about_deleting = information_about_deleting + old_name;
308     statusBar()->showMessage(information_about_deleting);
309 }
310
311 void MainWindow::on_action_9_triggered()
312 {
313     DialogAboutAddingInfoAboutStation* dialog_about_adding_info_about_station =
314         new DialogAboutAddingInfoAboutStation(&core,&number_of_the_route,
315         &number_of_the_station,&string_with_info,this)
316     ↪ ;
317     dialog_about_adding_info_about_station->show();
318 }
319
320 void MainWindow::addInformationAboutStation()
321 {
322     core.addInfoAboutStation(number_of_the_route+1,number_of_the_station+1,string_with_info.
323     ↪ toString());
324
325     QString name_of_the_station = (stations_buttons_vector[number_of_the_route][
326     ↪ number_of_the_station])>text();
327
328     QString information_about_adding = "Добавленна информация о станции: _";
329     information_about_adding = information_about_adding + name_of_the_station;
330     statusBar()->showMessage(information_about_adding);
331 }
332
333 void MainWindow::on_action_10_triggered()
334 {
335     DialogAboutDeletingInfoAboutStation* dialog_about_deleting_info_about_station =
336         new DialogAboutDeletingInfoAboutStation(&core,&number_of_the_station,this);
337     dialog_about_deleting_info_about_station->show();
338 }
339
340 void MainWindow::deleteInfoAboutStation()
341 {
342     std::vector<std::pair<std::string,std::string>> AllStaions = core.
343     ↪ getAllStationsWhichHaveDescription();
344
345     core.removeInfoAboutStation(number_of_the_station+1);
346
347     QString information_about_deleting = "Добавленна информация о станции: _";
348     information_about_deleting = information_about_deleting + ((AllStaions[
349     ↪ number_of_the_station]).first).c_str();
350     statusBar()->showMessage(information_about_deleting);
351 }

```

```

1  #ifndef DIALOGABOUTDELETINGSTATION_H
2  #define DIALOGABOUTDELETINGSTATION_H
3
4  #include <QDialog>
5  #include <core.h>
6  #include "mainwindow.h"
7
8  namespace Ui {
9  class DialogAboutDeletingStation;
10 }
11
12 class DialogAboutDeletingStation : public QDialog
13 {
14     Q_OBJECT
15
16     CoreOfInfoAboutMetro* core;
17
18     int *number_of_the_route;
19     int *number_of_the_station;
20
21 public:
22     explicit DialogAboutDeletingStation(CoreOfInfoAboutMetro* core, int *number_of_the_route,
23     ↪ int *number_of_the_station, QMainWindow* m_window,
24     ↪ QWidget *parent = 0);

```

```

24     ~DialogAboutDeletingStation();
25
26 signals:
27     void deletingStation();
28
29 private slots:
30     void on_comboBox_activated(int index);
31
32     void on_comboBox_2_activated(int index);
33
34     void on_pushButton_clicked();
35
36 private:
37     Ui::DialogAboutDeletingStation *ui;
38
39 };
40
41 #endif // DIALOGABOUTDELETINGSTATION_H

```

```

1 #include "dialogaboutdeletingstation.h"
2 #include "ui_dialogaboutdeletingstation.h"
3
4 DialogAboutDeletingStation::DialogAboutDeletingStation(CoreOfInfoAboutMetro *core, int *
    ↪ number_of_the_route, int *number_of_the_station, QMainWindow *m_window, QWidget *parent)
    ↪ :
5     core(core),
6     number_of_the_route(number_of_the_route),
7     number_of_the_station(number_of_the_station),
8     QDialog(parent),
9     ui(new Ui::DialogAboutDeletingStation)
10 {
11     ui->setupUi(this);
12
13     this->setWindowTitle("Удаление_станции");
14
15     for(int i = 0; i < core->howManyRoutes(); i++)
16     {
17         std::stringstream print_int;
18         print_int << i + 1;
19         ui->comboBox->addItem((print_int.str()).c_str());
20     }
21
22     connect(this, SIGNAL(deletingStation()), m_window, SLOT(deletingStation()));
23 }
24
25 DialogAboutDeletingStation::~DialogAboutDeletingStation()
26 {
27     delete ui;
28 }
29
30 void DialogAboutDeletingStation::on_comboBox_activated(int index)
31 {
32     *number_of_the_route = index;
33
34     ui->comboBox_2->clear();
35     ui->comboBox_2->setEnabled(true);
36     std::vector<std::string> route = core->getRoute(index + 1);
37     for(unsigned int i = 0; i < route.size(); i++)
38     {
39         ui->comboBox_2->addItem(route[i].c_str());
40     }
41 }
42
43 void DialogAboutDeletingStation::on_comboBox_2_activated(int index)
44 {
45     *number_of_the_station = index;
46
47     ui->pushButton->setEnabled(true);
48 }
49
50 void DialogAboutDeletingStation::on_pushButton_clicked()
51 {
52     emit deletingStation();
53     close();
54 }

```

```

1  #ifndef DIALOGABOUTDELETINGROUTE_H
2  #define DIALOGABOUTDELETINGROUTE_H
3
4  #include <QDialog>
5  #include <core.h>
6  #include "mainwindow.h"
7
8  namespace Ui {
9  class DialogAboutDeletingRoute;
10 }
11
12 class DialogAboutDeletingRoute : public QDialog
13 {
14     Q_OBJECT
15
16     CoreOfInfoAboutMetro* core;
17
18     int* index_out_of;
19
20 public:
21     explicit DialogAboutDeletingRoute(CoreOfInfoAboutMetro* core, int* index_out_of,
22     ↪ QMainWindow* m_window, QWidget *parent = 0);
23     ~DialogAboutDeletingRoute();
24
25 signals:
26     void deleteRouteSignal();
27
28 private slots:
29     void on_comboBox_activated(int index);
30
31     void on_pushButton_clicked();
32
33 private:
34     Ui::DialogAboutDeletingRoute *ui;
35 };
36
37 #endif // DIALOGABOUTDELETINGROUTE_H

```

```

1  #include "dialogaboutdeletingroute.h"
2  #include "ui_dialogaboutdeletingroute.h"
3
4  DialogAboutDeletingRoute::DialogAboutDeletingRoute(CoreOfInfoAboutMetro *core, int *
5  ↪ index_out_of, QMainWindow *m_window, QWidget *parent) :
6  core(core),
7  index_out_of(index_out_of),
8  QDialog(parent),
9  ui(new Ui::DialogAboutDeletingRoute)
10 {
11     ui->setupUi(this);
12
13     this->setFixedSize(250,150);
14     this->setWindowTitle("Удаление_маршрута");
15
16     for(int i = 0; i < core->howManyRoutes(); i++)
17     {
18         std::stringstream print_int;
19         print_int << i + 1;
20         ui->comboBox->addItem((print_int.str()).c_str());
21     }
22
23     connect(this, SIGNAL(deleteRouteSignal()), m_window, SLOT(deleteRouteSlot()));
24
25 DialogAboutDeletingRoute::~~DialogAboutDeletingRoute()
26 {
27     delete ui;
28 }
29
30 void DialogAboutDeletingRoute::on_comboBox_activated(int index)
31 {
32     *index_out_of = index;
33     ui->pushButton->setEnabled(true);
34 }

```

```

35
36 void DialogAboutDeletingRoute::on_pushButton_clicked()
37 {
38     emit deleteRouteSignal();
39     close();
40 }

```

```

1  #ifndef DIALOGABOUTDELETINGINFOABOUTSTATION_H
2  #define DIALOGABOUTDELETINGINFOABOUTSTATION_H
3
4  #include <QDialog>
5  #include <core.h>
6  #include "mainwindow.h"
7
8  namespace Ui {
9  class DialogAboutDeletingInfoAboutStation;
10 }
11
12 class DialogAboutDeletingInfoAboutStation : public QDialog
13 {
14     Q_OBJECT
15
16     CoreOfInfoAboutMetro* core;
17
18     int *number_of_the_station;
19
20 public:
21     explicit DialogAboutDeletingInfoAboutStation(CoreOfInfoAboutMetro* core, int *
↪ number_of_the_station, QMainWindow* m_window, QWidget *parent = 0);
22     ~DialogAboutDeletingInfoAboutStation();
23
24 signals:
25     void deleteInfoAboutStation();
26
27 private slots:
28     void on_comboBox_2_activated(int index);
29     void on_pushButton_clicked();
30
31 private:
32     Ui::DialogAboutDeletingInfoAboutStation *ui;
33 };
34
35 #endif // DIALOGABOUTDELETINGINFOABOUTSTATION_H

```

```

1  #include "dialogaboutdeletinginfoaboutstation.h"
2  #include "ui_dialogaboutdeletinginfoaboutstation.h"
3
4  DialogAboutDeletingInfoAboutStation::DialogAboutDeletingInfoAboutStation(CoreOfInfoAboutMetro*
↪ core, int *number_of_the_station, QMainWindow* m_window,
5                                     QWidget *parent) :
6     core(core),
7     number_of_the_station(number_of_the_station),
8     QDialog(parent),
9     ui(new Ui::DialogAboutDeletingInfoAboutStation)
10 {
11     ui->setupUi(this);
12
13     this->setWindowTitle("Удаление_информации_о_станции");
14
15     QString name_of_station;
16     std::vector<std::pair<std::string, std::string>> AllStaions = core->
↪ getAllStationsWhichHaveDescription();
17
18     for(unsigned int i = 0; i < AllStaions.size(); i++)
19     {
20         name_of_station = ((AllStaions[i]).first).c_str();
21         ui->comboBox_2->addItem(name_of_station);
22     }
23
24     connect(this, SIGNAL(deleteInfoAboutStation()), m_window, SLOT(deleteInfoAboutStation()));
25 }
26
27 DialogAboutDeletingInfoAboutStation::~DialogAboutDeletingInfoAboutStation()
28 {
29     delete ui;

```

```

30 }
31
32 void DialogAboutDeletingInfoAboutStation::on_comboBox_2_activated(int index)
33 {
34     *number_of_the_station = index;
35
36     ui->textBrowser->clear();
37     std::vector<std::pair<std::string, std::string>> AllStaions = core->
    ↪ getAllStationsWhichHaveDescription();
38     QString info_about_station = ((AllStaions[index]).second).c_str();
39     ui->textBrowser->setText(info_about_station);
40
41     ui->pushButton->setEnabled(true);
42 }
43
44 void DialogAboutDeletingInfoAboutStation::on_pushButton_clicked()
45 {
46     emit deleteInfoAboutStation();
47     close();
48 }

```

```

1 #ifndef DIALOGABOUTCHANGINGNAMEOFTHESTATIONS_H
2 #define DIALOGABOUTCHANGINGNAMEOFTHESTATIONS_H
3
4 #include <QDialog>
5 #include <core.h>
6 #include "mainwindow.h"
7
8 namespace Ui {
9 class DialogAboutChangingNameOfTheStations;
10 }
11
12 class DialogAboutChangingNameOfTheStations : public QDialog
13 {
14     Q_OBJECT
15
16     CoreOfInfoAboutMetro* core;
17
18     int *number_of_the_route;
19     int *number_of_the_station;
20     QString* new_name;
21
22
23 public:
24     explicit DialogAboutChangingNameOfTheStations(CoreOfInfoAboutMetro* core, int *
    ↪ number_of_the_route,
25
    ↪ new_name, QMainWindow* m_window,
    int *number_of_the_station, QString *
    ↪ parent = 0);
26     ~DialogAboutChangingNameOfTheStations();
27
28 signals:
29     void changeNameOfStation();
30
31 private slots:
32     void on_comboBox_activated(int index);
33     void on_comboBox_2_activated(int index);
34     void on_pushButton_clicked();
35     void on_textEdit_textChanged();
36
37 private:
38     Ui::DialogAboutChangingNameOfTheStations *ui;
39 };
40
41
42 #endif // DIALOGABOUTCHANGINGNAMEOFTHESTATIONS_H

```

```

1 #include "dialogaboutchangingnameofthestations.h"
2 #include "ui_dialogaboutchangingnameofthestations.h"
3
4 DialogAboutChangingNameOfTheStations::DialogAboutChangingNameOfTheStations(
    ↪ CoreOfInfoAboutMetro *core, int *number_of_the_route, int *number_of_the_station,
    ↪ QString *new_name, QMainWindow *m_window, QWidget *parent) :
5     core(core),
6     number_of_the_route(number_of_the_route),
7     number_of_the_station(number_of_the_station),

```

```

8     new_name(new_name),
9     QDialog(parent),
10    ui(new Ui::DialogAboutChangingNameOfTheStations)
11 {
12     ui->setupUi(this);
13
14     this->setWindowTitle("Изменение_названия_станции");
15
16     for(int i = 0; i < core->howManyRoutes(); i++)
17     {
18         std::stringstream print_int;
19         print_int << i + 1;
20         ui->comboBox->addItem((print_int.str()).c_str());
21     }
22
23     connect(this, SIGNAL(changeNameOfStation()), m_window, SLOT(changeNameOfStation()));
24 }
25
26 DialogAboutChangingNameOfTheStations::~DialogAboutChangingNameOfTheStations()
27 {
28     delete ui;
29 }
30
31 void DialogAboutChangingNameOfTheStations::on_comboBox_activated(int index)
32 {
33     *number_of_the_route = index;
34
35     ui->comboBox_2->clear();
36     ui->comboBox_2->setEnabled(true);
37     std::vector<std::string> route = core->getRoute(index + 1);
38     for(unsigned int i = 0; i < route.size(); i++)
39     {
40         ui->comboBox_2->addItem(route[i].c_str());
41     }
42 }
43
44 void DialogAboutChangingNameOfTheStations::on_comboBox_2_activated(int index)
45 {
46     *number_of_the_station = index;
47
48     ui->textEdit->setEnabled(true);
49 }
50
51 void DialogAboutChangingNameOfTheStations::on_pushButton_clicked()
52 {
53     *new_name = ui->textEdit->toPlainText();
54
55     emit changeNameOfStation();
56     close();
57 }
58
59 void DialogAboutChangingNameOfTheStations::on_textEdit_textChanged()
60 {
61     ui->pushButton->setEnabled(true);
62 }

```

```

1  #ifndef DIALOGABOUTADDINGSTATIONINROUTE_H
2  #define DIALOGABOUTADDINGSTATIONINROUTE_H
3
4  #include <QDialog>
5  #include <core.h>
6  #include "mainwindow.h"
7
8  namespace Ui {
9  class DialogAboutAddingStationInRoute;
10 }
11
12 class DialogAboutAddingStationInRoute : public QDialog
13 {
14     Q_OBJECT
15
16     CoreOfInfoAboutMetro* core;
17
18     int* index_out_of;
19     QString* name_of_adding_station;
20

```



```

21 public:
22     explicit DialogAboutAddingStationInRoute(CoreOfInfoAboutMetro* core, int* index_out_of,
23                                             QString* name_of_adding_station, QMainWindow*
24         ↪ m_window, QWidget *parent = 0);
25     ~DialogAboutAddingStationInRoute();
26 signals:
27     void addStationSignal();
28
29 private slots:
30
31     void on_comboBox_activated(int index);
32
33     void on_pushButton_clicked();
34
35     void on_textEdit_textChanged();
36
37 private:
38     Ui::DialogAboutAddingStationInRoute *ui;
39 };
40
41 #endif // DIALOGABOUTADDINGSTATIONINROUTE_H

```

```

1 #include "dialogaboutaddingstationinroute.h"
2 #include "ui_dialogaboutaddingstationinroute.h"
3
4 DialogAboutAddingStationInRoute::DialogAboutAddingStationInRoute(CoreOfInfoAboutMetro *core,
5     ↪ int *index_out_of, QString* name_of_adding_station, QMainWindow *m_window, QWidget *parent
6     ↪ ) :
7     core(core),
8     index_out_of(index_out_of),
9     name_of_adding_station(name_of_adding_station),
10    QDialog(parent),
11    ui(new Ui::DialogAboutAddingStationInRoute)
12 {
13     ui->setupUi(this);
14
15     this->setFixedSize(300,200);
16     this->setWindowTitle("Добавление_станции_в_маршрут");
17
18     for(int i = 0; i < core->howManyRoutes(); i++)
19     {
20         std::stringstream print_int;
21         print_int << i + 1;
22         ui->comboBox->addItem((print_int.str()).c_str());
23     }
24
25     connect(this, SIGNAL(addStationSignal()), m_window, SLOT(addStation()));
26 }
27
28 DialogAboutAddingStationInRoute::~DialogAboutAddingStationInRoute()
29 {
30     delete ui;
31 }
32
33 void DialogAboutAddingStationInRoute::on_comboBox_activated(int index)
34 {
35     *index_out_of = index;
36     ui->textEdit->setEnabled(true);
37 }
38
39 void DialogAboutAddingStationInRoute::on_pushButton_clicked()
40 {
41     *name_of_adding_station = ui->textEdit->toPlainText();
42
43     emit addStationSignal();
44     close();
45 }
46
47 void DialogAboutAddingStationInRoute::on_textEdit_textChanged()
48 {
49     ui->pushButton->setEnabled(true);
50 }

```

```

1 #ifndef DIALOGABOUTADDINGINFOABOUTSTATION_H

```

```

2 #define DIALOGABOUTADDINGINFOABOUTSTATION_H
3
4 #include <QDialog>
5 #include <core.h>
6 #include "mainwindow.h"
7
8 namespace Ui {
9 class DialogAboutAddingInfoAboutStation;
10 }
11
12 class DialogAboutAddingInfoAboutStation : public QDialog
13 {
14     Q_OBJECT
15
16     CoreOfInfoAboutMetro* core;
17
18     int *number_of_the_route;
19     int *number_of_the_station;
20     QString* information_about_station;
21
22 public:
23     explicit DialogAboutAddingInfoAboutStation(CoreOfInfoAboutMetro* core, int *
        ↪ number_of_the_route,
24
        int *number_of_the_station, QString *
        ↪ information_about_station, QMainWindow* m_window,
        QWidget *parent = 0);
25     ~DialogAboutAddingInfoAboutStation();
26
27 signals:
28     void addInformationAboutStation();
29
30 private slots:
31     void on_comboBox_activated(int index);
32     void on_comboBox_2_activated(int index);
33     void on_pushButton_clicked();
34     void on_textEdit_textChanged();
35
36 private:
37     Ui::DialogAboutAddingInfoAboutStation *ui;
38 };
39
40 #endif // DIALOGABOUTADDINGINFOABOUTSTATION_H

```

```

1 #include "dialogaboutaddinginfoaboutstation.h"
2 #include "ui_dialogaboutaddinginfoaboutstation.h"
3
4 DialogAboutAddingInfoAboutStation::DialogAboutAddingInfoAboutStation(CoreOfInfoAboutMetro *
    ↪ core, int *number_of_the_route, int *number_of_the_station, QString *
    ↪ information_about_station, QMainWindow *m_window, QWidget *parent) :
5     core(core),
6     number_of_the_route(number_of_the_route),
7     number_of_the_station(number_of_the_station),
8     information_about_station(information_about_station),
9     QDialog(parent),
10     ui(new Ui::DialogAboutAddingInfoAboutStation)
11 {
12     ui->setupUi(this);
13
14     this->setWindowTitle("Добавление_информации_о_станции");
15
16     for(int i = 0; i < core->howManyRoutes(); i++)
17     {
18         std::stringstream print_int;
19         print_int << i + 1;
20         ui->comboBox->addItem((print_int.str()).c_str());
21     }
22
23     connect(this, SIGNAL(addInformationAboutStation()), m_window, SLOT(addInformationAboutStation
        ↪ ()));
24 }
25
26 DialogAboutAddingInfoAboutStation::~DialogAboutAddingInfoAboutStation()
27 {
28     delete ui;
29 }
30

```

```

31 void DialogAboutAddingInfoAboutStation::on_comboBox_activated(int index)
32 {
33     *number_of_the_route = index;
34
35     ui->comboBox_2->clear();
36     ui->comboBox_2->setEnabled(true);
37     std::vector<std::string> route = core->getRoute(index + 1);
38     for(unsigned int i = 0; i < route.size(); i++)
39     {
40         ui->comboBox_2->addItem(route[i].c_str());
41     }
42 }
43
44 void DialogAboutAddingInfoAboutStation::on_comboBox_2_activated(int index)
45 {
46     *number_of_the_station = index;
47
48     ui->textEdit->setEnabled(true);
49 }
50
51 void DialogAboutAddingInfoAboutStation::on_pushButton_clicked()
52 {
53     *information_about_station = ui->textEdit->toPlainText();
54
55     emit addInformationAboutStation();
56     close();
57 }
58
59 void DialogAboutAddingInfoAboutStation::on_textEdit_textChanged()
60 {
61     ui->pushButton->setEnabled(true);
62 }

```