

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

1	Транспортное расписание	2
1.1	Задание	2
1.2	Концепция	2
1.3	Минимально работоспособный продукт	2
1.4	Диаграмма прецедентов использования	3
1.5	Диаграмма последовательностей	4
1.6	Вывод	4
2	Проектирование приложения	4
2.1	Архитектуру приложения	4
2.2	Диаграмма компонентов	5
2.3	Файлы создаваемые в процессе работы приложения	5
2.4	Интерфейс ядра	5
2.5	Вывод	6
3	Реализация Транспортного расписания	6
3.1	Используемые версии	6
3.2	Основные классы	7
3.3	Скриншоты основных экранов пользовательского интерфейса	7
3.4	Вывод	9
4	Процесс обеспечения качества и тестирование	9
4.1	О ревью	9
4.2	О демо	10
4.3	Список использованных утилит	10
4.4	Автоматические тесты	10
5	Выводы	11
6	Приложение	11
6.1	Листинги	11
6.2	Doxygen	57

1 Транспортное расписание

Современный мир переполнен различной информацией, поэтому было решено её как-то систематизировать, в связи с этим темой проекта было выбрано Транспортное расписание. Так как это очень обширная тема, было решено выбрать метрополитен и для него создать приложение, позволяющее хранить информацию о маршрутах и станциях.

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 станциях 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 станциях

1.4 Диаграмма прецедентов использования

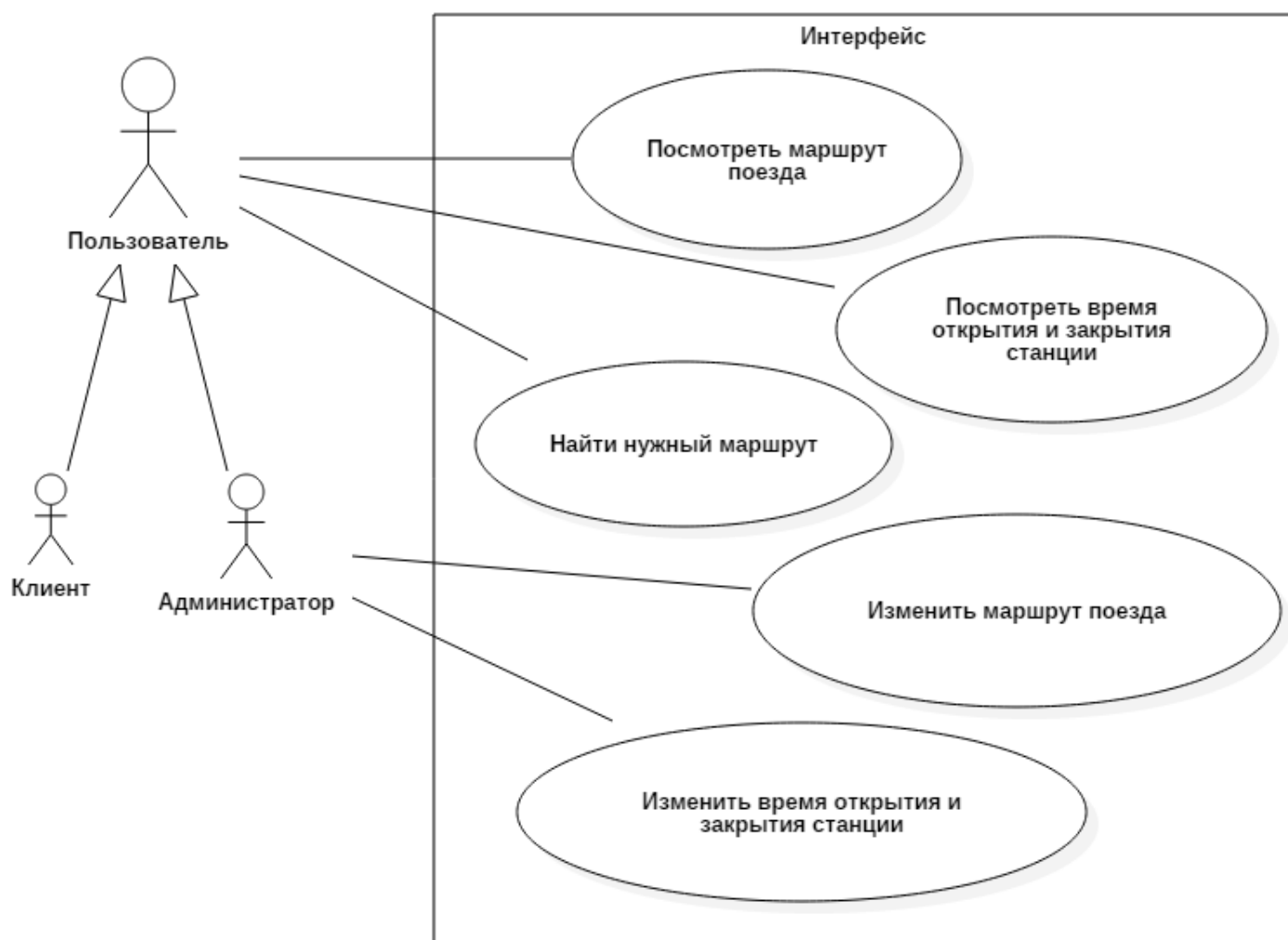


Рис. 1: Диаграмма прецедентов использования

Здесь продемонстрирована запланированная функциональность приложения

1.5 Диаграмма последовательностей

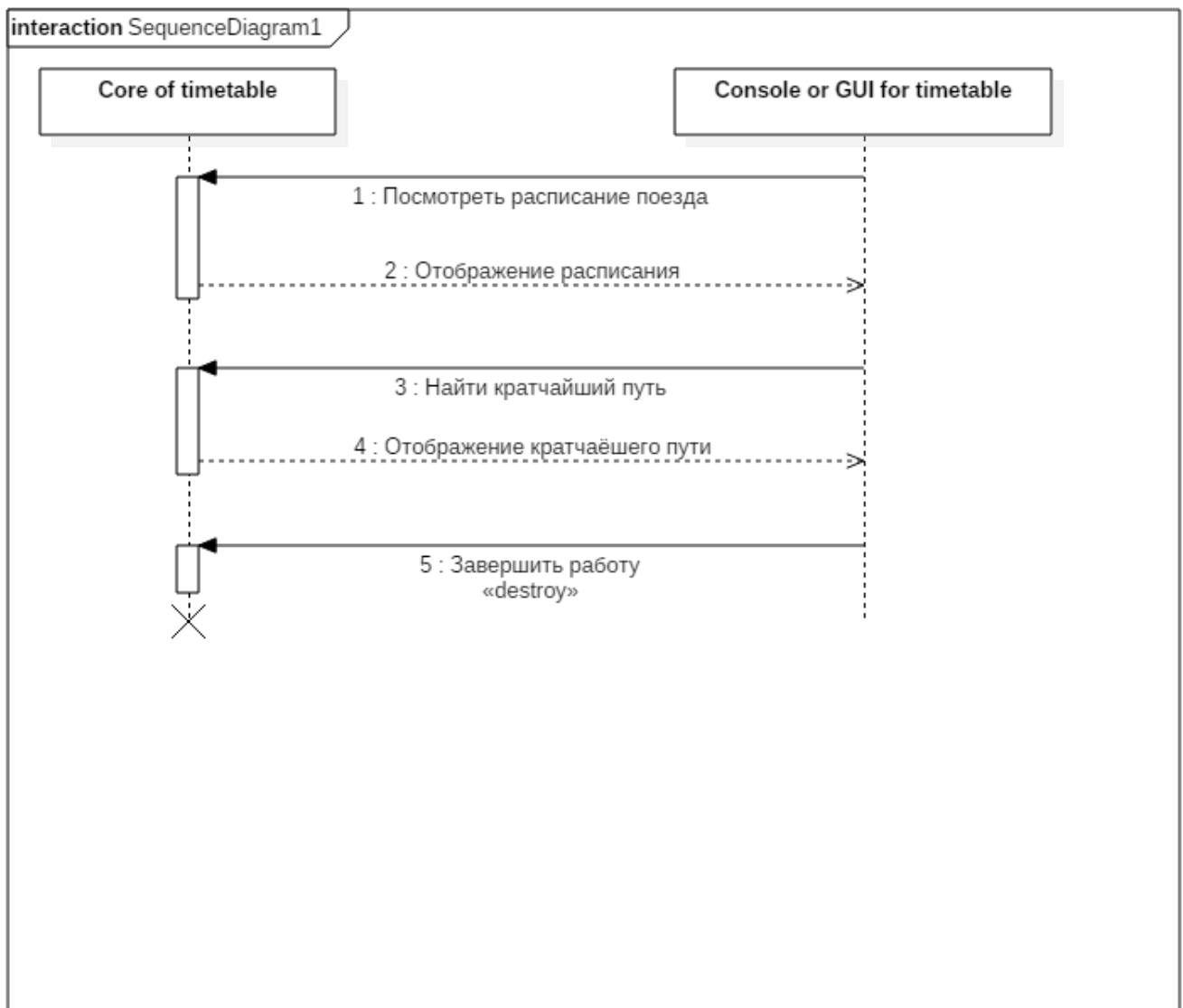


Рис. 2: Диаграмма последовательностей

Здесь продемонстрирован процесс общения ядра с его оболочкой

1.6 Вывод

Было решено создать приложение позволяющее пользователю хранить информацию о маршрутах и станциях

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

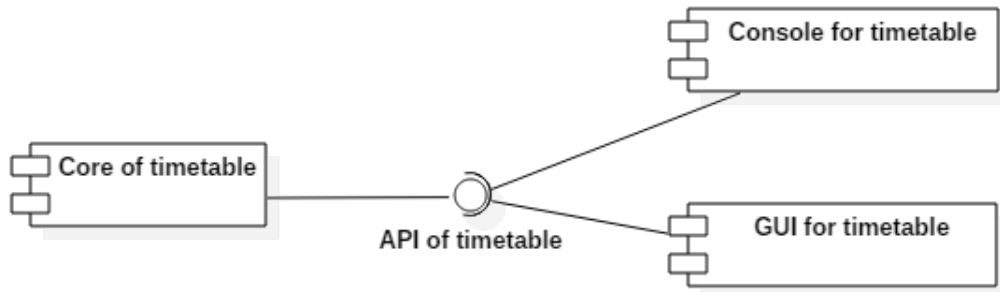


Рис. 3: Диаграмма компонентов

Предусмотренно две оболочки вокруг ядра, которые будут давать пользователю возможность использовать функциональности ядра

2.3 Файлы создаваемые в процессе работы приложения

Называются файлы могут как угодно пользователю.

Формат данных должен быть следующий:

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

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

2.4 Интерфейс ядра

В библиотеку предоставляет следующую функциональность:

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

Один из способов 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 находиться в определённом формате

3. `int howManyRoutes() const noexcept;`

Возвращает информацию о том, сколько маршрутов существует на 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, const std::string &marking) 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

2.5 Вывод

Было выделено 4 подпроекта: подпроект отвечающий за бизнес-логику, подпроект отвечающий за тестирование бизнес-логики, и два подпроекта отвечающих за предоставление функциональности ядра пользователю

3 Реализация Транспортного расписания

3.1 Используемые версии

- Qt Creator 4.0.0 (opensource)

- Стандарт c++11
- Qt 5.4.2 MinGW 32bit и Qt 5.6.0 MinGW 32bit
- Операционная система: Windows 10
- Cppcheck версии 1.7.2
- Doxygen 1.8.11

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



Рис. 4: Меню консольного приложения

Здесь предоставлены основные функциональность предлагаемые пользователю:

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

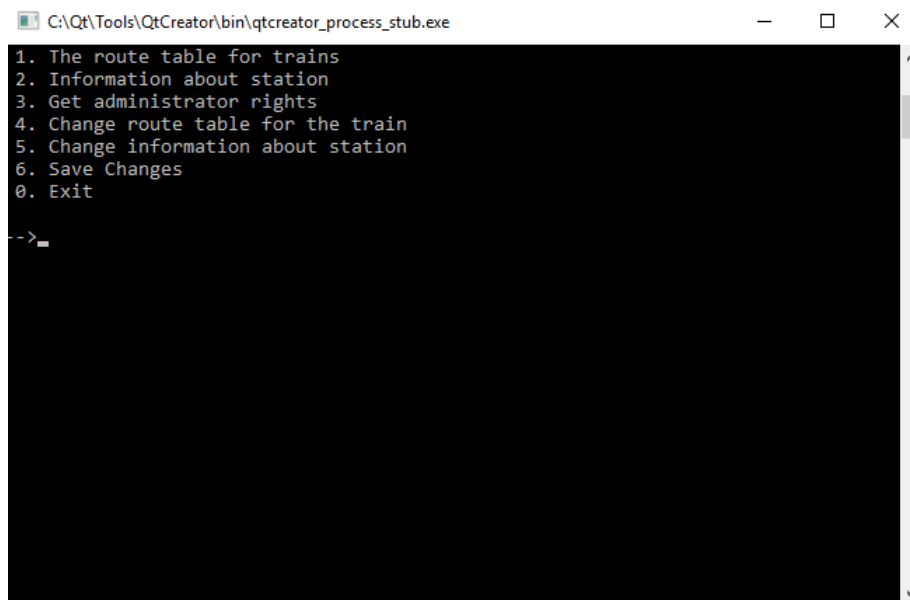


Рис. 5: Меню консольного приложения, расширенное для администрирования

Кроме предидущих для администратора становятся доступны функциональности связанные с редактированием:

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

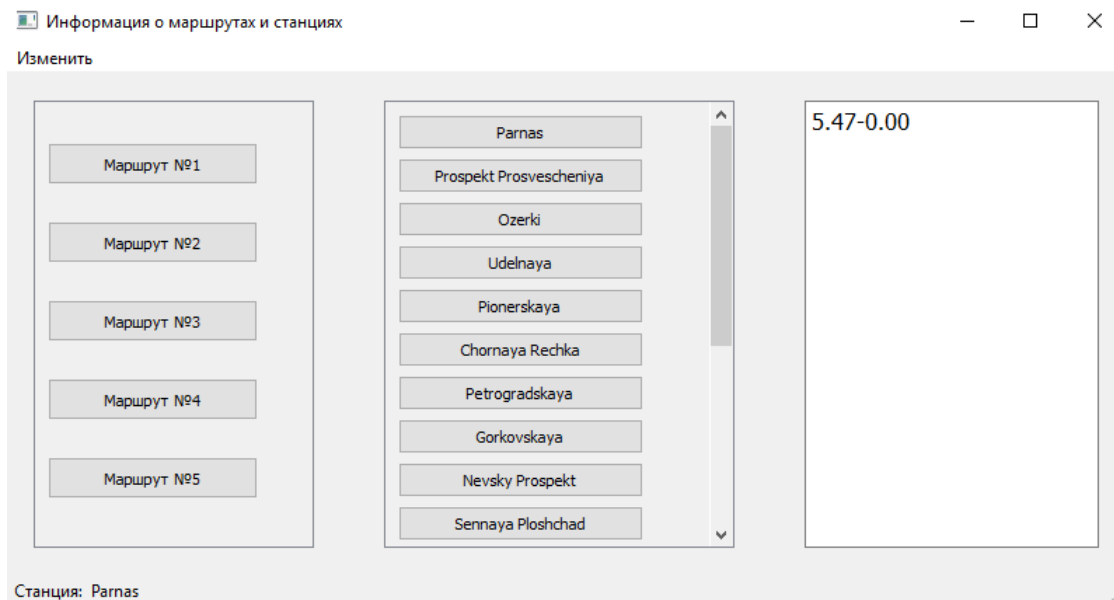


Рис. 6: Главное окно графического приложения

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

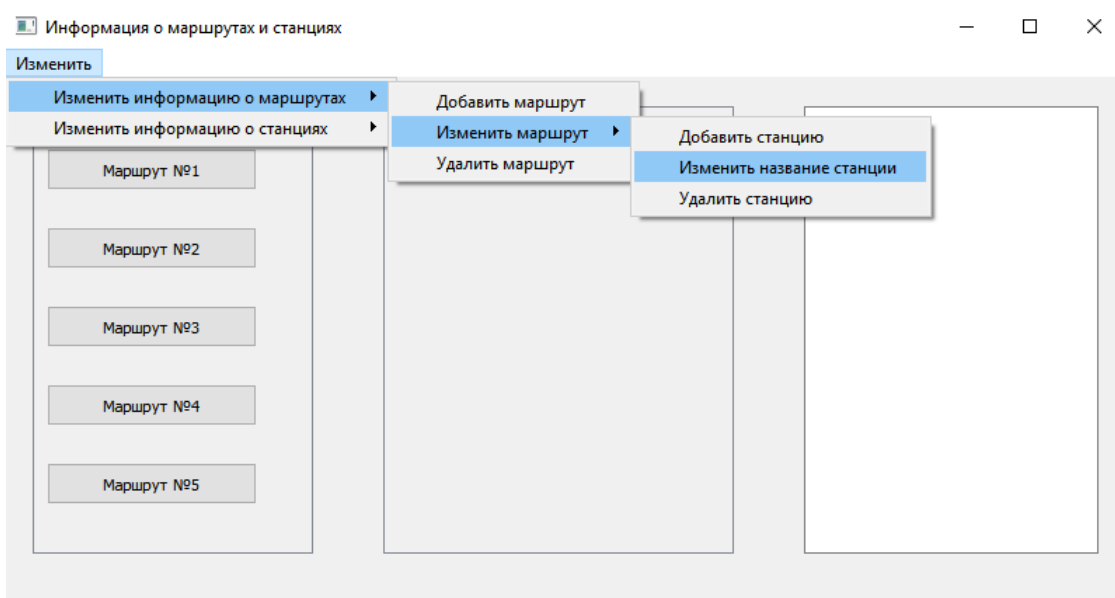


Рис. 7: Открыто меню изменения маршрутов

Здесь продемонстрировано меню редактирования информации о маршрутах и станциях

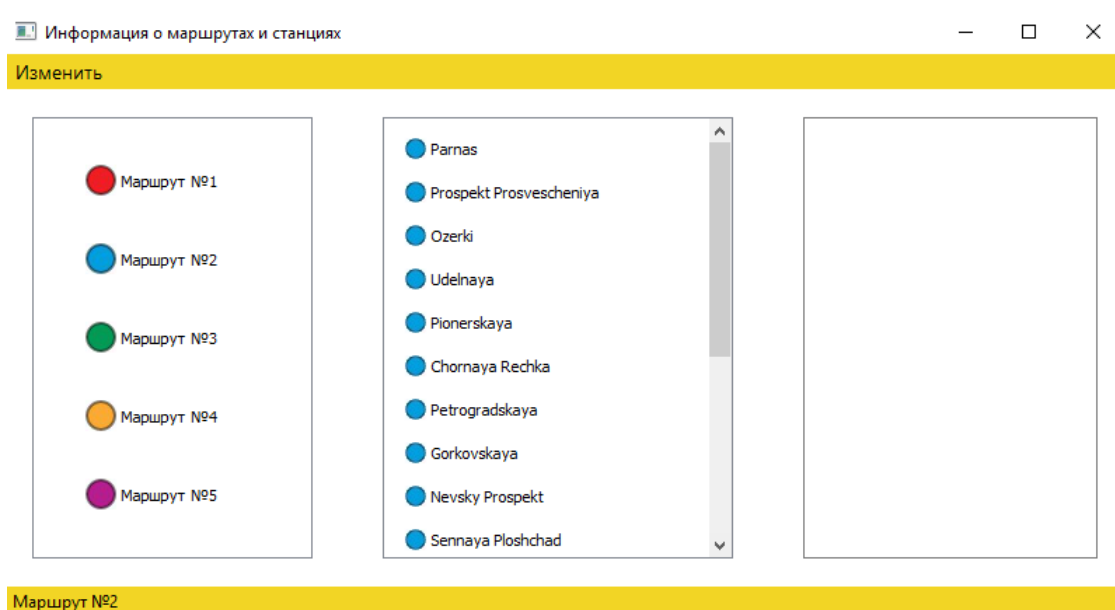


Рис. 8: Меню после изменения дизайна

Реализация функциональностей приложения затронута не была, а вот дизайн кнопок и окна изменился

3.4 Вывод

Рализация происходила в несколько этапов, сперва добавлялась рабочая функциональность, а потом она приводилась в опрятный вид

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 вывести 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:

- srpccheck, в данном случае он, в основном, выдавал предупреждения 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:

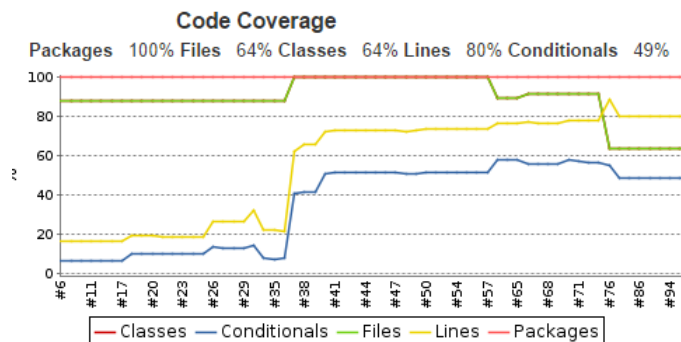


Рис. 9: Процент покрытия

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 опробовал 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 Приложение

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

```

```

99     virtual void addStationInRoute(const int number_of_the_route, const std::string &
    ↪ what_to_add) = 0;
100
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     {

```

```

144         throw StationDoesNotExist(number_of_the_station);
145     }
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
63     stationInfo.addInfoAboutStation(pairNameValue.first , pairNameValue.second);
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 void ParsingInfo::loadFromFile(const std::string &name_of_the_file_with_route, const std::
100 ↪ string &name_of_the_file_with_station,
101                               RoutesInfo &routeInfo, StationsInfo &stationInfo)
102 {
103     std::ifstream reading_file_with_routes(name_of_the_file_with_route);
104     if (!reading_file_with_routes.is_open())
105     {
106         throw MissingFile(name_of_the_file_with_route);
107     }
108
109     std::string line_from_file;
110     std::getline(reading_file_with_routes, line_from_file);
111
112     putInfoAboutRoutes(line_from_file, routeInfo);
113
114     reading_file_with_routes.close();
115
116
117     std::ifstream reading_file_with_stations(name_of_the_file_with_station);
118     if (!reading_file_with_stations.is_open())
119     {
120         throw MissingFile(name_of_the_file_with_station);
121     }
122
123     std::getline(reading_file_with_stations, line_from_file);
124
125     putInfoAboutStations(line_from_file, stationInfo);
126
127     reading_file_with_stations.close();
128 }
129
130 void ParsingInfo::saveChanges(const std::string &name_of_the_file_with_route, const std::
131 ↪ string &name_of_the_file_with_station, RoutesInfo &routeInfo, StationsInfo &stationInfo)
132 {
133     saveChangesForRoute(name_of_the_file_with_route, routeInfo);
134

```



```

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
19 private Q_SLOTS:
20     void putOfRights();
21
22     void putInfoAboutRoutes();
23
24     void changeStationInRoute();
25
26     void addStationInRoute();
27
28     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),
    ↪ StationDoesNotExist);
270
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),
    ↪ StationDoesNotExist);
282 }
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 #include <QSplashScreen>
4 #include <QTimer>
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9
10    MainWindow w;
11
12    QSplashScreen *splash = new QSplashScreen;
13    splash->setPixmap(QPixmap(":/img/img/metro.jpg"));

```

```

14     splash->show();
15
16     QTimer::singleShot(2000,splash,SLOT(close()));
17     QTimer::singleShot(2000,&w,SLOT(show()));
18
19     return a.exec();
20 }

```

```

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 <QPainter>
12 #include <QPixmap>
13 #include <QSize>
14 #include <QPoint>
15 #include <QRect>
16 #include <QTimer>
17 #include <QPainter>
18 #include <QRect>
19 #include <QPointF>
20 #include <QRadialGradient>
21 #include <QPen>
22 #include <QBrush>
23 #include <core.h>
24 #include "dialogaboutaddingstationinroute.h"
25 #include "dialogaboutdeletingroute.h"
26 #include "dialogaboutchangingnameofthestations.h"
27 #include "dialogaboutdeletingstation.h"
28 #include "dialogaboutaddinginfoaboutstation.h"
29 #include "dialogaboutdeletinginfoaboutstation.h"
30
31 namespace Ui {
32 class MainWindow;
33 }
34
35 class MainWindow : public QMainWindow
36 {
37     Q_OBJECT
38
39     CoreOfInfoAboutMetro core;
40
41     QVector<QVector<QPushButton*>> stations_buttons_vector;
42     QVector<QPushButton*> routes_buttons;
43
44     QVBoxLayout* stations_layout;
45     QVBoxLayout* routes_layout;
46
47     int index;
48
49     int number_of_the_route;
50     int number_of_the_station;
51     QString string_with_info;
52
53     QString button_style_for_route = "QPushButton_"
54         "{"
55         "border:_0px_"
56         "border-radius:_9px;"
57         "background:_#FFFFFF;_"
58         "}"
59         "QPushButton: hover_"
60         "{"
61         "border:_1px_solid_#bbbbbb;"
62         "}"
63         "QPushButton: pressed_"
64         "{"
65         "border:_2px_solid_#4d4d4d;"
66         "}"
67
68     QString button_style_for_station = "QPushButton_"

```



```

69         "{"
70         "padding: 6px;"
71         "text-align: left;"
72         "border: 0px;"
73         "border-radius: 9px;"
74         "background: #FFFFFF;"
75         "}"
76         "QPushButton: hover_"
77         "{"
78         "border: 1px solid #bbbbbb;"
79         "}"
80         "QPushButton: pressed_"
81         "{"
82         "border: 2px solid #4d4d4d;"
83         "}"
84
85 public:
86     explicit MainWindow(QWidget *parent = 0);
87     ~MainWindow();
88
89 private:
90     Ui::MainWindow *ui;
91
92 private slots:
93     void showStations();
94     void showInfoAboutStation();
95     void deleteRouteSlot();
96     void changeNameOfStation();
97     void addStation();
98     void deletingStation();
99     void addInformationAboutStation();
100    void deleteInfoAboutStation();
101
102    void on_action_2_triggered();
103    void on_action_4_triggered();
104    void on_action_9_triggered();
105    void on_action_5_triggered();
106    void on_action_6_triggered();
107    void on_action_7_triggered();
108    void on_action_10_triggered();
109
110 };
111
112 #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     QPalette pal_for_main_window;
15     pal_for_main_window.setBrush(this->backgroundRole(), Qt::white);
16     this->setPalette(pal_for_main_window);
17
18     ui->statusBar->setStyleSheet("background: #f2d525;");
19     ui->menu->setStyleSheet("background: #f2d525;");
20     ui->menuBar->setStyleSheet("QMenuBar{"
21         "background: #f2d525;"
22         "font-size: 14px;"
23     "}"
24     "QMenuBar::item{"
25         "background: transparent;"
26     "}"
27     "QMenuBar::item:selected{"
28         "color: #ffffff"
29     "}"
30     "QMenuBar::item:pressed{"
31         "color: #ffffff"

```

```

32         "});
33
34     ui->scrollAreaForStations->setFixedWidth(250);
35     ui->scrollAreaForRoutes->setFixedWidth(200);
36
37     try
38     {
39         core.loadInfoFromFile("metro_Saint-Petersburg_route_info.txt", "metro_Saint-
40         ↪ Petersburg_station_info.txt");
41     }
42     catch (MissingFile&)
43     {
44         QMessageBox* file_does_not_exist = new QMessageBox;
45
46         file_does_not_exist->setWindowTitle("Отсутствуют_Файлы");
47         file_does_not_exist->setText("Не_обнаружены_файлы_с_информацией,_ожидались\n"
48         ↪ "metro_Saint-Petersburg_route_info.txt_и_metro_Saint-
49         ↪ Petersburg_station_info.txt");
50         QTimer::singleShot(2000, file_does_not_exist, SLOT(show()));
51
52         core.addRoute();
53         core.addStationInRoute(1, "Парнас");
54         core.addStationInRoute(1, "Вовсе_не_Парнас");
55         core.addStationInRoute(1, "Супер_Парнас");
56         core.addInfoAboutStation("Супер_Парнас", "Это_самый_лучший_Парнас_в_мире");
57         core.addInfoAboutStation("Вовсе_не_Парнас", "Это_совсем_не_Парнас");
58         core.addRoute();
59         core.addStationInRoute(2, "Дыбенко");
60         core.addStationInRoute(2, "Дыбенко");
61         core.addStationInRoute(2, "Пацаны_с_Дыбенко");
62         core.addInfoAboutStation("Пацаны_с_Дыбенко", "ДЫБЕНКООООО");
63         core.addInfoAboutStation("Дыбенко", "Дыбена_—_это_самый_благоустроенный_район_города_
64         ↪ Санкт_Петербурга_—,"
65         ↪ "в_котором_живёт_элита_общества");
66
67         statusBar()->showMessage("Файлы_не_найжены");
68     }
69
70     stations_layout = new QVBoxLayout;
71     QWidget* stations = new QWidget;
72
73     routes_layout = new QVBoxLayout;
74     QWidget* routes = new QWidget;
75
76     for (int i = 0; i < core.howManyRoutes(); i++)
77     {
78         std::stringstream print_int_for_number_of_the_route;
79         print_int_for_number_of_the_route << i + 1;
80
81         std::string name_of_the_route = "Маршрут_№" + print_int_for_number_of_the_route.
82         ↪ str();
83
84         QPushButton* route_button = new QPushButton(name_of_the_route.c_str());
85
86         QString name_of_file_for_route = ":/img/img/button_for_routes";
87         std::stringstream print_int_for_number_of_the_img;
88         print_int_for_number_of_the_img << (i % 14) + 1;
89         name_of_file_for_route = name_of_file_for_route + (print_int_for_number_of_the_img
90         ↪ .str()).c_str() + ".jpg";
91
92         QPixmap* pix_for_route = new QPixmap(name_of_file_for_route);
93         route_button->setIcon(*pix_for_route);
94         route_button->setIconSize(pix_for_route->size());
95
96         route_button->setStyleSheet(button_style_for_route);
97
98         routes_buttons.push_back(route_button);
99         routes_buttons[i]->setFixedSize(150,30);
100         routes_layout->addWidget(routes_buttons[i]);
101         routes_buttons[i]->show();
102         routes_buttons[i]->setProperty("index", i);
103         connect(routes_buttons[i], SIGNAL(clicked()), this, SLOT(showStations()));

```

```

103     QVector<QPushButton*> stations_buttons;
104
105     std::vector<std::string> route = core.getRoute(i + 1);
106
107     for(unsigned int j = 0; j < route.size(); j++)
108     {
109         QPushButton* station_button = new QPushButton((route[j]).c_str(), this);
110
111         QString name_of_file_for_station = "/img/img/button_for_stations";
112         name_of_file_for_station = name_of_file_for_station + (
113     ↪ print_int_for_number_of_the_img.str()).c_str() + ".jpg";
114         QPixmap* pix_for_stations = new QPixmap(name_of_file_for_station);
115         station_button->setIcon(*pix_for_stations);
116         station_button->setIconSize(pix_for_stations->size());
117         station_button->setStyleSheet(button_style_for_station);
118
119         stations_buttons.push_back(station_button);
120         stations_buttons[j]->hide();
121         stations_buttons[j]->setFixedHeight(25);
122         stations_buttons[j]->setProperty("name", (route[j]).c_str());
123
124         stations_layout->addWidget(stations_buttons[j]);
125
126         connect(stations_buttons[j], SIGNAL(clicked()), this, SLOT(
127     ↪ showInfoAboutStation()));
128     }
129
130     stations_buttons_vector.push_back(stations_buttons);
131 }
132
133 stations->setLayout(stations_layout);
134 ui->scrollAreaForStations->setWidget(stations);
135
136 routes->setLayout(routes_layout);
137 ui->scrollAreaForRoutes->setWidget(routes);
138
139 QPainter *p = new QPainter(this);
140 p->drawLine(1,1,100,100);
141 }
142
143 void MainWindow::showStations()
144 {
145     for(int i = 0; i < core.howManyRoutes(); i++)
146     {
147         std::vector<std::string> route = core.getRoute(i + 1);
148
149         for(unsigned int j = 0; j < route.size(); j++)
150         {
151             (stations_buttons_vector[i][j])->hide();
152         }
153     }
154
155     QPushButton *button = qobject_cast<QPushButton*>(sender());
156     QVariant index = button->property("index");
157
158     int number_of_the_route = index.toInt() + 1;
159
160     std::vector<std::string> route = core.getRoute(number_of_the_route);
161
162     for(unsigned int j = 0; j < route.size(); j++)
163     {
164         (stations_buttons_vector[number_of_the_route - 1][j])->show();
165     }
166
167     std::stringstream print_int;
168     print_int << number_of_the_route;
169     std::string new_name = "Мапи́р №" + print_int.str();
170     statusBar()->showMessage(new_name.c_str());
171 }
172
173 void MainWindow::showInfoAboutStation()
174 {
175
176

```

```

177     QPushButton *button = qobject_cast<QPushButton*>(sender());
178     QVariant name = button->property("name");
179
180     QString qt_name_of_the_stations = name.toString();
181     std::string name_of_the_station = qt_name_of_the_stations.toStdString();
182
183
184     std::string name_of_the_label_std = "";
185     QString name_of_the_label_qt;
186
187     try
188     {
189         name_of_the_label_std += core.getInfoAboutStation(name_of_the_station);
190         name_of_the_label_qt = name_of_the_label_std.c_str();
191     }
192     catch (StationDoesNotExist&)
193     {
194         name_of_the_label_qt = "Станция_не_найдена";
195     }
196
197     ui->textBrowser->setText(name_of_the_label_qt);
198
199     QString toStatusBar = "Станция: _" + qt_name_of_the_stations;
200
201     statusBar()->showMessage(toStatusBar);
202 }
203
204 MainWindow::~MainWindow()
205 {
206     delete ui;
207 }
208
209 void MainWindow::on_action_2_triggered()
210 {
211     core.addRoute();
212     int i = core.howManyRoutes() - 1;
213
214     std::stringstream print_int;
215     print_int << i + 1;
216     std::string name_of_the_route = "Маршрут_№" + print_int.str();
217     QPushButton* route_button = new QPushButton(name_of_the_route.c_str(), this);
218     route_button->setFixedSize(150,30);
219     routes_buttons.push_back(route_button);
220     routes_layout->addWidget(route_button);
221     route_button->show();
222     route_button->setProperty("index", i);
223
224     QString name_of_file = ":/img/img/button_for_routes";
225     std::stringstream print_int_for_number_of_the_img;
226     print_int_for_number_of_the_img << (i % 14) + 1;
227     name_of_file = name_of_file + (print_int_for_number_of_the_img.str()).c_str() + ".jpg";
228
229     QPixmap* pix = new QPixmap(name_of_file);
230
231     route_button->setIcon(*pix);
232     route_button->setIconSize(pix->size());
233     route_button->setStyleSheet(button_style_for_route);
234
235     connect(route_button, SIGNAL(clicked()), this, SLOT(showStations()));
236
237     QVector<QPushButton*> stations_buttons;
238     stations_buttons_vector.push_back(stations_buttons);
239
240     std::string new_name = "Добавлен: _";
241     new_name += name_of_the_route;
242     statusBar()->showMessage(new_name.c_str());
243 }
244
245 void MainWindow::on_action_4_triggered()
246 {
247     DialogAboutDeletingRoute* dialog_about_delete_route = new DialogAboutDeletingRoute(&core
248     → ,&index, this);
249     dialog_about_delete_route->show();
250 }
251 void MainWindow::deleteRouteSlot()

```

```

252 {
253     std::vector<std::string> route = core.getRoute(index+1);
254     for(unsigned int j = 0; j < route.size(); j++)
255     {
256         delete(stations_buttons_vector[index][j]);
257     }
258
259     delete(routes_buttons[core.howManyRoutes()-1]);
260
261     for(int i = 0; i < core.howManyRoutes()-1; i++)
262     {
263         std::stringstream print_int;
264         print_int << i + 1;
265         std::string name_of_the_route = "Маршрут_№" + print_int.str();
266         routes_buttons[i]->setText(name_of_the_route.c_str());
267         routes_buttons[i]->setProperty("index", i);
268     }
269
270     if (stations_buttons_vector.size() > index)
271     {
272         stations_buttons_vector.remove(index);
273     }
274
275     core.deleteRoute(index+1);
276
277     std::stringstream print_int;
278     print_int << index + 1;
279     std::string new_name = "Удалён: _Маршрут_№" + print_int.str();
280     statusBar()->showMessage(new_name.c_str());
281 }
282
283 void MainWindow::on_action_6_triggered()
284 {
285     DialogAboutChangingNameOfTheStations* dialog_about_changing_name_of_stations =
286         new DialogAboutChangingNameOfTheStations(&core,&number_of_the_route,
287             &number_of_the_station,&string_with_info,this)
288     ↪ ;
289     dialog_about_changing_name_of_stations->show();
290 }
291
292 void MainWindow::changeNameOfStation()
293 {
294     core.changeStationInRoute(number_of_the_route + 1, number_of_the_station + 1,
295         string_with_info.toStdString());
296     QString old_name = (stations_buttons_vector[number_of_the_route][number_of_the_station])->
297     ↪ text();
298
299     (stations_buttons_vector[number_of_the_route][number_of_the_station])->setText(
300     ↪ string_with_info);
301     stations_buttons_vector[number_of_the_route][number_of_the_station]->
302     ↪ setProperty("name",string_with_info);
303
304     QString new_name = "Станция: _" + old_name + "_переименована_в_" + string_with_info;
305     statusBar()->showMessage(new_name);
306 }
307
308 void MainWindow::on_action_5_triggered()
309 {
310     DialogAboutAddingStationInRoute* dialog_about_adding_station_in_route = new
311     ↪ DialogAboutAddingStationInRoute(&core,&index,&string_with_info,this);
312     dialog_about_adding_station_in_route->show();
313 }
314
315 void MainWindow::addStation()
316 {
317     core.addStationInRoute(index+1,string_with_info.toStdString());
318
319     QPushButton* station_button = new QPushButton(string_with_info, this);
320
321     stations_buttons_vector[index].push_back(station_button);
322     station_button->hide();
323     station_button->setFixedHeight(25);
324     station_button->setProperty("name",(string_with_info));
325     stations_layout->addWidget(station_button);

```

```

325
326 QString name_of_file_for_station = ":/img/img/button_for_stations";
327 std::stringstream print_int_for_number_of_the_img;
328 print_int_for_number_of_the_img << (index % 14) + 1;
329 name_of_file_for_station = name_of_file_for_station + (print_int_for_number_of_the_img.str
→ ()) .c_str() + ".jpg";
330 QPixmap* pix_for_stations = new QPixmap(name_of_file_for_station);
331 station_button->setIcon(*pix_for_stations);
332 station_button->setIconSize(pix_for_stations->size());
333 station_button->setStyleSheet(button_style_for_station);
334
335
336 connect(station_button, SIGNAL(clicked()), this, SLOT(showInfoAboutStation()));
337
338
339 QString new_name = "Добавлена_станция:..";
340 new_name = new_name + string_with_info;
341 statusBar()->showMessage(new_name);
342 }
343
344 void MainWindow::on_action_7_triggered()
345 {
346     DialogAboutDeletingStation* dialog_about_deleting_station =
347         new DialogAboutDeletingStation(&core,&number_of_the_route,
348                                         &number_of_the_station,this);
349     dialog_about_deleting_station->show();
350 }
351
352 void MainWindow::deletingStation()
353 {
354     QString old_name = (stations_buttons_vector[number_of_the_route][number_of_the_station])->
→ text();
355
356     core.deleteStationFromRoute(number_of_the_route+1,number_of_the_station+1);
357
358     delete(stations_buttons_vector[number_of_the_route][stations_buttons_vector[
→ number_of_the_route].size()-1]);
359
360     std::vector<std::string> route = core.getRoute(number_of_the_route+1);
361     QString name_of_station;
362     for(unsigned int j = 0; j < route.size(); j++)
363     {
364         name_of_station = (route[j]).c_str();
365         stations_buttons_vector[number_of_the_route][j]->setText(name_of_station);
366     }
367
368     QString information_about_deleting = "Удалена_станция:..";
369     information_about_deleting = information_about_deleting + old_name;
370     statusBar()->showMessage(information_about_deleting);
371 }
372
373 void MainWindow::on_action_9_triggered()
374 {
375     DialogAboutAddingInfoAboutStation* dialog_about_adding_info_about_station =
376         new DialogAboutAddingInfoAboutStation(&core,&number_of_the_route,
377                                               &number_of_the_station,&string_with_info,this)
→ ;
378     dialog_about_adding_info_about_station->show();
379 }
380
381 void MainWindow::addInformationAboutStation()
382 {
383     core.addInfoAboutStation(number_of_the_route+1,number_of_the_station+1,string_with_info.
→ toString());
384
385     QString name_of_the_station = (stations_buttons_vector[number_of_the_route][
→ number_of_the_station])->text();
386
387     QString information_about_adding = "Добавленна_информация_о_станции:..";
388     information_about_adding = information_about_adding + name_of_the_station;
389     statusBar()->showMessage(information_about_adding);
390 }
391
392 void MainWindow::on_action_10_triggered()
393 {
394     DialogAboutDeletingInfoAboutStation* dialog_about_deleting_info_about_station =

```

```

395         new DialogAboutDeletingInfoAboutStation(&core,&number_of_the_station,this);
396         dialog_about_deleting_info_about_station->show();
397     }
398
399 void MainWindow::deleteInfoAboutStation()
400 {
401     std::vector<std::pair<std::string,std::string>> AllStaions = core.
    ↪ getAllStationsWhichHaveDescription();
402
403     core.removeInfoAboutStation(number_of_the_station+1);
404
405     QString information_about_deleting = "Добавленна информация о станции: _";
406     information_about_deleting = information_about_deleting + ((AllStaions[
    ↪ number_of_the_station]).first).c_str();
407     statusBar()->showMessage(information_about_deleting);
408 }

```

```

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,
    ↪ 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
42 #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 ui->pushButton->setStyleSheet("QPushButton_"
14     "{"
15     "background: _#fde910;"
16     "border: _1px_solid_000000;"
17     "border-radius: _9px;"
18     "}"
19     "QPushButton: hover_"
20     "{"
21     "background: _#ffd919;"
22     "}"
23     "QPushButton: pressed_"
24     "{"
25     "background: _#e6bf00;"
26     "}"");
27 ui->comboBox->setStyleSheet("QComboBox_"
28     "{"
29     "background: _#fced3f;"
30     "border-radius: _1px;"
31     "}"
32     "QComboBox: editable_"
33     "{"
34     "background: _white;"
35     "}"
36     "QComboBox_QAbstractItemView_"
37     "background: _#fcf172;"
38     "selection-color: _black;"
39     "selection-background-color: _#fced3f;"
40     "}"");
41 ui->comboBox_2->setStyleSheet("QComboBox_"
42     "{"
43     "background: _#fced3f;"
44     "border-radius: _1px;"
45     "}"
46     "QComboBox: editable_"
47     "{"
48     "background: _white;"
49     "}"
50     "QComboBox_QAbstractItemView_"
51     "background: _#fcf172;"
52     "selection-color: _black;"
53     "selection-background-color: _#fced3f;"
54     "}"");
55
56 this->setWindowTitle("Удаление_станции");
57 QPalette pal;
58 pal.setBrush(this->backgroundRole(), Qt::white);
59 this->setPalette(pal);
60
61 for(int i = 0; i < core->howManyRoutes(); i++)
62 {
63     std::stringstream print_int;
64     print_int << i + 1;
65     ui->comboBox->addItem((print_int.str()).c_str());
66 }
67
68 connect(this, SIGNAL(deletingStation()), m_window, SLOT(deletingStation()));
69 }
70
71 DialogAboutDeletingStation::~DialogAboutDeletingStation()
72 {
73     delete ui;
74 }
75
76 void DialogAboutDeletingStation::on_comboBox_activated(int index)
77 {
78     *number_of_the_route = index;
79
80     ui->comboBox_2->clear();
81     ui->comboBox_2->setEnabled(true);
82     std::vector<std::string> route = core->getRoute(index + 1);
83     for(unsigned int i = 0; i < route.size(); i++)
84     {
85         ui->comboBox_2->addItem(route[i].c_str());
86     }
87 }
88

```



```

89 void DialogAboutDeletingStation::on_comboBox_2_activated(int index)
90 {
91     *number_of_the_station = index;
92
93     ui->pushButton->setEnabled(true);
94 }
95
96 void DialogAboutDeletingStation::on_pushButton_clicked()
97 {
98     emit deletingStation();
99     close();
100 }

```

```

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,
    ↪ QMainWindow* m_window, QWidget *parent = 0);
22     ~DialogAboutDeletingRoute();
23
24 signals:
25     void deleteRouteSignal();
26
27 private slots:
28
29     void on_comboBox_activated(int index);
30
31     void on_pushButton_clicked();
32
33 private:
34     Ui::DialogAboutDeletingRoute *ui;
35 };
36
37
38 #endif // DIALOGABOUTDELETINGROUTE_H

```

```

1  #include "dialogaboutdeletingroute.h"
2  #include "ui_dialogaboutdeletingroute.h"
3
4  DialogAboutDeletingRoute::DialogAboutDeletingRoute(CoreOfInfoAboutMetro *core, int *
    ↪ index_out_of, QMainWindow *m_window, QWidget *parent) :
5     core(core),
6     index_out_of(index_out_of),
7     QDialog(parent),
8     ui(new Ui::DialogAboutDeletingRoute)
9 {
10     ui->setupUi(this);
11
12     ui->pushButton->setStyleSheet("QPushButton_"
13                                "{"
14                                "background: ~#fde910;"
15                                "border: ~1px_solid_000000;"
16                                "border-radius: ~9px;"
17                                "}"
18                                "QPushButton: hover_"
19                                "{"
20                                "background: ~#ffd919;"
21                                "}"

```

```

22         "QPushButton:pressed_"
23         "{"
24         "background:~#e6bf00;"
25         "}")";
26     ui->comboBox->setStyleSheet("QComboBox_"
27         "{"
28         "background:~#fced3f;"
29         "border-radius:~1px;"
30         "}"
31         "QComboBox:editable"
32         "{"
33         "background:~white;"
34         "}"
35         "QComboBox_QAbstractItemView_"
36         "background:~#fcf172;"
37         "selection-color:~black;"
38         "selection-background-color:~#fced3f;"
39         "}")";
40
41     this->setFixedSize(250,150);
42     this->setWindowTitle("Удаление_маршрута");
43     QPalette pal;
44     pal.setBrush(this->backgroundRole(), Qt::white);
45     this->setPalette(pal);
46
47     for(int i = 0; i < core->howManyRoutes(); i++)
48     {
49         std::stringstream print_int;
50         print_int << i + 1;
51         ui->comboBox->addItem((print_int.str()).c_str());
52     }
53
54     connect(this,SIGNAL(deleteRouteSignal()),m_window,SLOT(deleteRouteSlot()));
55 }
56
57 DialogAboutDeletingRoute::~DialogAboutDeletingRoute()
58 {
59     delete ui;
60 }
61
62 void DialogAboutDeletingRoute::on_comboBox_activated(int index)
63 {
64     *index_out_of = index;
65     ui->pushButton->setEnabled(true);
66 }
67
68 void DialogAboutDeletingRoute::on_pushButton_clicked()
69 {
70     emit deleteRouteSignal();
71     close();
72 }

```

```

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 *
22     ↪ number_of_the_station, QMainWindow* m_window, QWidget *parent = 0);
23     ~DialogAboutDeletingInfoAboutStation();

```

```

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,
    QWidget *parent) :
5     core(core),
6     number_of_the_station(number_of_the_station),
7     QDialog(parent),
8     ui(new Ui::DialogAboutDeletingInfoAboutStation)
9 {
10     ui->setupUi(this);
11
12     ui->pushButton->setStyleSheet("QPushButton_"
13         "{
14         \"background: _#fde910;\"
15         \"border: _1px_solid_000000;\"
16         \"border-radius: _9px;\"
17         }\"
18         \"QPushButton: hover_\"
19         \"{
20         \"background: _#ffd919;\"
21         }\"
22         \"QPushButton: pressed_\"
23         \"{
24         \"background: _#e6bf00;\"
25         }\"");
26
27     ui->comboBox_2->setStyleSheet("QComboBox_"
28         "{
29         \"background: _#fced3f;\"
30         \"border-radius: _1px;\"
31         }\"
32         \"QComboBox: editable\"
33         \"{
34         \"background: _white;\"
35         }\"
36         \"QComboBox_QAbstractItemView_{\"
37         \"background: _#fcf172;\"
38         \"selection-color: _black;\"
39         \"selection-background-color: _#fced3f;\"
40         }\"");
41
42     this->setWindowTitle("Удаление_информации_о_станции");
43     QPalette pal;
44     pal.setBrush(this->backgroundRole(), Qt::white);
45     this->setPalette(pal);
46
47     QString name_of_station;
48     std::vector<std::pair<std::string, std::string>> AllStaions = core->
    ↪ getAllStationsWhichHaveDescription();
49
50     for(unsigned int i = 0; i < AllStaions.size(); i++)
51     {
52         name_of_station = ((AllStaions[i]).first).c_str();
53         ui->comboBox_2->addItem(name_of_station);
54     }
55
56     ui->textBrowser->setStyleSheet("font-size: _16px;");
57
58     connect(this, SIGNAL(deleteInfoAboutStation()), m_window, SLOT(deleteInfoAboutStation()));
59 }
60
61 DialogAboutDeletingInfoAboutStation::~DialogAboutDeletingInfoAboutStation()

```

```

62 {
63     delete ui;
64 }
65
66 void DialogAboutDeletingInfoAboutStation::on_comboBox_2_activated(int index)
67 {
68     *number_of_the_station = index;
69
70     ui->textBrowser->clear();
71     std::vector<std::pair<std::string, std::string>> AllStaions = core->
    ↪ getAllStationsWhichHaveDescription();
72     QString info_about_station = ((AllStaions[index]).second).c_str();
73     ui->textBrowser->setText(info_about_station);
74
75     ui->pushButton->setEnabled(true);
76 }
77
78 void DialogAboutDeletingInfoAboutStation::on_pushButton_clicked()
79 {
80     emit deleteInfoAboutStation();
81     close();
82 }

```

```

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                                                     int *number_of_the_station, QString *
    ↪ new_name, QMainWindow* m_window,
26                                                     QWidget *parent = 0);
27     ~DialogAboutChangingNameOfTheStations();
28
29 signals:
30     void changeNameOfStation();
31
32 private slots:
33     void on_comboBox_activated(int index);
34     void on_comboBox_2_activated(int index);
35     void on_pushButton_clicked();
36     void on_textEdit_textChanged();
37
38 private:
39     Ui::DialogAboutChangingNameOfTheStations *ui;
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     ui->pushButton->setStyleSheet("QPushButton_"
15                                  "{
16                                  \"background: ~#fde910;\"
17                                  \"border: ~1px_solid_000000;\"
18                                  \"border-radius: ~9px;\"
19                                  }\"
20                                  \"QPushButton: hover_\"
21                                  \"{
22                                  \"background: ~#ffd919;\"
23                                  }\"
24                                  \"QPushButton: pressed_\"
25                                  \"{
26                                  \"background: ~#e6bf00;\"
27                                  }\"");
28     ui->comboBox->setStyleSheet("QComboBox_"
29                                 "{
30                                 \"background: ~#fced3f;\"
31                                 \"border-radius: ~1px;\"
32                                 }\"
33                                 \"QComboBox: editable \"
34                                 \"{
35                                 \"background: ~white;\"
36                                 }\"
37                                 \"QComboBox_QAbstractItemView_{\"
38                                 \"background: ~#fcf172;\"
39                                 \"selection-color: ~black;\"
40                                 \"selection-background-color: ~#fced3f;\"
41                                 }\"");
42     ui->comboBox_2->setStyleSheet("QComboBox_"
43                                  "{
44                                  \"background: ~#fced3f;\"
45                                  \"border-radius: ~1px;\"
46                                  }\"
47                                  \"QComboBox: editable \"
48                                  \"{
49                                  \"background: ~white;\"
50                                  }\"
51                                  \"QComboBox_QAbstractItemView_{\"
52                                  \"background: ~#fcf172;\"
53                                  \"selection-color: ~black;\"
54                                  \"selection-background-color: ~#fced3f;\"
55                                  }\"");
56
57     this->setWindowTitle("Изменение_названия_станции");
58     QPalette pal;
59     pal.setBrush(this->backgroundRole(), Qt::white);
60     this->setPalette(pal);
61
62     for(int i = 0; i < core->howManyRoutes(); i++)
63     {
64         std::stringstream print_int;
65         print_int << i + 1;
66         ui->comboBox->addItem((print_int.str()).c_str());
67     }
68
69     connect(this, SIGNAL(changeNameOfStation()), m_window, SLOT(changeNameOfStation()));
70 }
71
72 DialogAboutChangingNameOfTheStations::~DialogAboutChangingNameOfTheStations()
73 {
74     delete ui;
75 }
76
77 void DialogAboutChangingNameOfTheStations::on_comboBox_activated(int index)
78 {
79     *number_of_the_route = index;
80
81     ui->comboBox_2->clear();

```

```

82     ui->comboBox_2->setEnabled(true);
83     std::vector<std::string> route = core->getRoute(index + 1);
84     for(unsigned int i = 0; i < route.size(); i++)
85     {
86         ui->comboBox_2->addItem(route[i].c_str());
87     }
88 }
89
90 void DialogAboutChangingNameOfTheStations::on_comboBox_2_activated(int index)
91 {
92     *number_of_the_station = index;
93
94     ui->textEdit->setEnabled(true);
95 }
96
97 void DialogAboutChangingNameOfTheStations::on_pushButton_clicked()
98 {
99     *new_name = ui->textEdit->toPlainText();
100
101     emit changeNameOfStation();
102     close();
103 }
104
105 void DialogAboutChangingNameOfTheStations::on_textEdit_textChanged()
106 {
107     ui->pushButton->setEnabled(true);
108 }

```

```

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

```

```

1  #include "dialogaboutaddingstationinroute.h"
2  #include "ui_dialogaboutaddingstationinroute.h"
3

```

```

4 DialogAboutAddingStationInRoute::DialogAboutAddingStationInRoute(CoreOfInfoAboutMetro *core,
  ↪ int *index_out_of, QString* name_of_adding_station, QMainWindow *m_window, QWidget *parent
  ↪ ) :
5     core(core),
6     index_out_of(index_out_of),
7     name_of_adding_station(name_of_adding_station),
8     QDialog(parent),
9     ui(new Ui::DialogAboutAddingStationInRoute)
10 {
11     ui->setupUi(this);
12
13     this->setFixedSize(300,200);
14
15     ui->pushButton->setStyleSheet("QPushButton_"
16                                   "{"
17                                   "background: ~#fde910;"
18                                   "border: ~1px_solid_000000;"
19                                   "border-radius: ~9px;"
20                                   "}"
21                                   "QPushButton: hover_"
22                                   "{"
23                                   "background: ~#ffd919;"
24                                   "}"
25                                   "QPushButton: pressed_"
26                                   "{"
27                                   "background: ~#e6bf00;"
28                                   "}"");
29     ui->comboBox->setStyleSheet("QComboBox_"
30                                   "{"
31                                   "background: ~#fced3f;"
32                                   "border-radius: ~1px;"
33                                   "}"
34                                   "QComboBox: editable_"
35                                   "{"
36                                   "background: ~white;"
37                                   "}"
38                                   "QComboBox_QAbstractItemView_"
39                                   "{"
40                                   "background: ~#fcf172;"
41                                   "selection-color: ~black;"
42                                   "selection-background-color: ~#fced3f;"
43                                   "}"");
44
45     this->setWindowTitle("Добавление_станции_в_маршрут");
46     QPalette pal;
47     pal.setBrush(this->backgroundRole(), Qt::white);
48     this->setPalette(pal);
49
50     for(int i = 0; i < core->howManyRoutes(); i++)
51     {
52         std::stringstream print_int;
53         print_int << i + 1;
54         ui->comboBox->addItem((print_int.str()).c_str());
55     }
56
57     connect(this, SIGNAL(addStationSignal()), m_window, SLOT(addStation()));
58 }
59
60 DialogAboutAddingStationInRoute::~DialogAboutAddingStationInRoute()
61 {
62     delete ui;
63 }
64
65 void DialogAboutAddingStationInRoute::on_comboBox_activated(int index)
66 {
67     *index_out_of = index;
68     ui->textEdit->setEnabled(true);
69 }
70
71 void DialogAboutAddingStationInRoute::on_pushButton_clicked()
72 {
73     *name_of_adding_station = ui->textEdit->toPlainText();
74
75     emit addStationSignal();
76     close();
77 }

```

```

78
79 void DialogAboutAddingStationInRoute::on_textEdit_textChanged()
80 {
81     ui->pushButton->setEnabled(true);
82 }

```

```

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,
25                                             QWidget *parent = 0);
26     ~DialogAboutAddingInfoAboutStation();
27
28 signals:
29     void addInformationAboutStation();
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::DialogAboutAddingInfoAboutStation *ui;
39 };
40
41 #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     ui->pushButton->setStyleSheet("QPushButton_"
15                                   "{"
16                                   "background: ~#fde910;"
17                                   "border: ~1px_solid_000000;"
18                                   "border-radius: ~9px;"
19                                   "}"
20                                   "QPushButton: hover_"
21                                   "{"
22                                   "background: ~#ffd919;"
23                                   "}"

```



```

24         "QPushButton:pressed_"
25         "{"
26         "background:~#e6bf00;"
27         "}")";
28     ui->comboBox->setStyleSheet("QComboBox_"
29     "{"
30     "background:~#fced3f;"
31     "border-radius:~1px;"
32     "}"
33     "QComboBox:editable"
34     "{"
35     "background:~white;"
36     "}"
37     "QComboBox_QAbstractItemView_"
38     "background:~#fcf172;"
39     "selection-color:~black;"
40     "selection-background-color:~#fced3f;"
41     "}")";
42     ui->comboBox_2->setStyleSheet("QComboBox_"
43     "{"
44     "background:~#fced3f;"
45     "border-radius:~1px;"
46     "}"
47     "QComboBox:editable"
48     "{"
49     "background:~white;"
50     "}"
51     "QComboBox_QAbstractItemView_"
52     "background:~#fcf172;"
53     "selection-color:~black;"
54     "selection-background-color:~#fced3f;"
55     "}")";
56
57     this->setWindowTitle("Добавление_информации_о_станции");
58     QPalette pal;
59     pal.setBrush(this->backgroundRole(), Qt::white);
60     this->setPalette(pal);
61
62     for(int i = 0; i < core->howManyRoutes(); i++)
63     {
64         std::stringstream print_int;
65         print_int << i + 1;
66         ui->comboBox->addItem((print_int.str()).c_str());
67     }
68
69     connect(this, SIGNAL(addInformationAboutStation()), m_window, SLOT(addInformationAboutStation
70     ↪ ()));
71 }
72 DialogAboutAddingInfoAboutStation::~DialogAboutAddingInfoAboutStation()
73 {
74     delete ui;
75 }
76
77 void DialogAboutAddingInfoAboutStation::on_comboBox_activated(int index)
78 {
79     *number_of_the_route = index;
80
81     ui->comboBox_2->clear();
82     ui->comboBox_2->setEnabled(true);
83     std::vector<std::string> route = core->getRoute(index + 1);
84     for(unsigned int i = 0; i < route.size(); i++)
85     {
86         ui->comboBox_2->addItem(route[i].c_str());
87     }
88 }
89
90 void DialogAboutAddingInfoAboutStation::on_comboBox_2_activated(int index)
91 {
92     *number_of_the_station = index;
93
94     ui->textEdit->setEnabled(true);
95 }
96
97 void DialogAboutAddingInfoAboutStation::on_pushButton_clicked()
98 {

```

```
99      *information_about_station = ui->textEdit->toPlainText();
100
101      emit addInformationAboutStation();
102      close();
103  }
104
105  void DialogAboutAddingInfoAboutStation::on_textEdit_textChanged()
106  {
107      ui->pushButton->setEnabled(true);
108  }
```

6.2 Doxygen

My Project

Generated by Doxygen 1.8.11

Оглавление

Глава 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

API	??
CoreOfInfoAboutMetro	??
ConsoleForTimetable	??
exception	
MissingFile	??
RouteDoesNotExist	??
StationDoesNotExist	??
ParsingInfo	??
QDialog	
DialogAboutAddingInfoAboutStation	??
DialogAboutAddingStationInRoute	??
DialogAboutChangingNameOfTheStations	??
DialogAboutDeletingInfoAboutStation	??
DialogAboutDeletingRoute	??
DialogAboutDeletingStation	??
QMainWindow	
MainWindow	??
QObject	
Test_for_coreTest	??
RoutesInfo	??
StationsInfo	??
WorkWithInfo	??
WorkWithRoutes	??
WorkWithStations	??

Глава 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

API

Абстрактный класс, которым описывается функциональность предоставляемая ядром приложения	??
ConsoleForTimetable Класс служащий для работы с ядром через консоль	??
CoreOfInfoAboutMetro Класс в котором содержится основная бизнес-логика приложения	??
DialogAboutAddingInfoAboutStation	??
DialogAboutAddingStationInRoute	??
DialogAboutChangingNameOfTheStations	??
DialogAboutDeletingInfoAboutStation	??
DialogAboutDeletingRoute	??
DialogAboutDeletingStation	??
MainWindow	??
MissingFile Исключение, которое бросается при неудачном открытии файла, от него можно получить, информацию о том, какой файл запрашивал пользователь, перед появлением исключения	??
ParsingInfo The ParsingInfo class	??
RouteDoesNotExist Исключение бросается в том случае, если оказывается, что запрашиваемого маршрута не существует	??
RoutesInfo Класс отвечающий за работу с информацией связанной с маршрутами	??
StationDoesNotExist Исключение бросается в том случае, если оказывается, что запрашиваемой станции не существует	??
StationsInfo Класс отвечающий за работу с информацией связанной со станциями	??
Test_for_coreTest Тесты для функциональностей предоставляемых ядром	??
WorkWithInfo Абстрактный класс, содержащий несколько функций помогающих при работе с пользователем	??

[WorkWithRoutes](#)

Класс, отвечающий за обработку запросов связанных с маршрутами, с помощью консоли ??

[WorkWithStations](#)

Класс, отвечающий за обработку запросов связанных со станциями, с помощью консоли ??

Глава 3

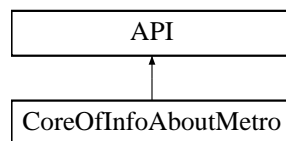
Class Documentation

3.1 API Class Reference

Абстрактный класс, которым описывается функциональность предоставляемая ядром приложения

#include <api.h>

Inheritance diagram for API:



Public Member Functions

- API (const [API](#) &)=delete
- [API](#) & operator= (const [API](#) &)=delete
- virtual void [putOffRights](#) (const Rights_of_customers rights) noexcept=0
Выдача прав
- virtual Rights_of_customers [getInformationOfTheRights](#) () const noexcept=0
- virtual void [putInfoAboutMetro](#) (const std::string &infoAboutRoutes, const std::string &infoAboutStations) noexcept=0
Один из способ передать входные данные
- virtual void [loadInfoFromFile](#) (const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station)=0
Загрузить входные данные из файла
- virtual int [howManyRoutes](#) () const noexcept=0
- virtual std::vector< std::string > [getRoute](#) (const int number_of_the_route)=0
- virtual std::string [getInfoAboutStation](#) (const std::string &name_of_the_station)=0
- virtual std::string [getInfoAboutStation](#) (const int number_of_the_route, const int number_of_the_station)=0
Способ альтернативный предидущему
- virtual void [changeStationInRoute](#) (const int number_of_the_route, const int number_of_the_station, const std::string &new_marking)=0
Изменение станции в маршруте

- virtual void [addStationInRoute](#) (const int number_of_the_route, const std::string &what_to_↵ add)=0
Добавляет станцию в маршрут
- virtual int [addRoute](#) () noexcept=0
Добавляет новый (пустой) маршрут
- virtual void [deleteStationFromRoute](#) (const int number_of_the_route, const int number_of_↵ the_station)=0
Удаляет станцию из маршрута
- virtual void [deleteRoute](#) (const int number_of_the_route)=0
Удаляет маршрут
- virtual void [addInfoAboutStation](#) (const std::string &name_of_the_station, const std::string &station_description) noexcept=0
Добавляет информацию о станции
- virtual void [addInfoAboutStation](#) (const int number_of_the_route, const int number_of_the_↵ station, const std::string &station_description)=0
Альтернативный путь добавления информации о станции
- virtual void [removeInfoAboutStation](#) (const std::string &what_station_to_remove)=0
Удаляет информацию о станции
- virtual void [removeInfoAboutStation](#) (const int number_of_the_station)=0
Альтернативный путь удаления информации о станции, в месте с ним используется метод [get↵ AllStationsWhichHaveDescription\(\)](#), именно из него берётся номер станции
- virtual std::vector< std::pair< std::string, std::string > > [getAllStationsWhichHaveDescription](#) () noexcept=0
- virtual void [saveChanges](#) (const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station) noexcept=0
Сохранить информацию

3.1.1 Detailed Description

Абстрактный класс, которым описывается функциональность предоставляемая ядром приложения

3.1.2 Member Function Documentation

- 3.1.2.1 virtual void API::addInfoAboutStation (const std::string & name_of_the_station, const std::string & station_description) [pure virtual], [noexcept]

Добавляет информацию о станции

Parameters

name_of_the_station	- номер станции, о которой информация
station_description	- информация о станции

Implemented in [CoreOfInfoAboutMetro](#).

- 3.1.2.2 virtual void API::addInfoAboutStation (const int number_of_the_route, const int number_of_the_station, const std::string & station_description) [pure virtual]

Альтернативный путь добавления информации о станции

Parameters

number_of_the_route	- номер маршрута, в котором содержится станция, про которую информация
number_of_the_station	- номер станции, про которую информация
station_description	- информация о станции

Exceptions

При	неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.3 virtual int API::addRoute () [pure virtual], [noexcept]

Добавляет новый (пустой) маршрут

Returns

Номер добавленного маршрута

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.4 virtual void API::addStationInRoute (const int number_of_the_route, const std::string & what_to_add) [pure virtual]

Добавляет станцию в маршрут

Parameters

number_of_the_route	- номер маршрута, в который нужно добавить станцию
what_to_add	- название станции, которую нужно добавить

Exceptions

При	неверном запросе бросается RouteDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.5 virtual void API::changeStationInRoute (const int number_of_the_route, const int number_of_the_station, const std::string & new_marking) [pure virtual]

Изменение станции в маршруте

Parameters

number_of_the_route	- номер маршрута
number_of_the_station	- номер станции
new_marking	- то, что нужно поставить в замен

Exceptions

При	неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.6 virtual void API::deleteRoute (const int number_of_the_route) [pure virtual]

Удаляет маршрут

Parameters

number_of_the_route	- номер маршрута, который нужно удалить
---------------------	---

Exceptions

При	неверном запросе бросается RouteDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.7 virtual void API::deleteStationFromRoute (const int number_of_the_route, const int number_of_the_station) [pure virtual]

Удаляет станцию из маршрута

Parameters

number_of_the_route	- номер маршрута, из которого нужно удалить станцию
number_of_the_station	- номер станции, которую нужно удалить

Exceptions

При	неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.8 virtual std::vector<std::pair<std::string, std::string> > API::getAllStationsWhichHave↵
Description () [pure virtual], [noexcept]

Returns

Всё станции, про которые содержится информация, первая часть пары название станции, вторая информация о ней

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.9 virtual std::string API::getInfoAboutStation (const std::string & name_of_the_station)
[pure virtual]

Parameters

name_of_the_station	- название станции, о которой нужно получить информацию
---------------------	---

Returns

информация о запрашиваемой станции

Exceptions

При	запросе несуществующей станции бросается StationDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.10 virtual std::string API::getInfoAboutStation (const int number_of_the_route, const int number_of_the_station) [pure virtual]

Способ альтернативный предыдущему

Parameters

number_of_the_route	- номер маршрута, содержащего станцию
number_of_the_station	- номер станции в маршруте

Returns

информация о запрашиваемой станции

Exceptions

При	запросе несуществующей станции бросается StationDoesNotExist , а RouteDoesNotExist , если маршрута не существует
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.11 virtual Rights_of_customers API::getInformationOfTheRights () const [pure virtual],
[noexcept]

Returns

Информация о правах

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.12 `virtual std::vector<std::string> API::getRoute (const int number_of_the_route) [pure virtual]`

Parameters

<code>number_of_the_route</code>	- номер запрашиваемого маршрута
----------------------------------	---------------------------------

Returns

Запрашиваемый маршрут в формате `vector<string>` , где `string` - это название станций

Exceptions

При	запросе несуществующего маршрута бросается RouteDoesNotExist , от объекта которого можно получить информацию, какой маршрут запрашивали
-----	---

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.13 `virtual int API::howManyRoutes () const [pure virtual], [noexcept]`

Returns

Сколько маршрутов существует на данный момент

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.14 `virtual void API::loadInfoFromFile (const std::string & name_of_the_file_with_route, const std::string & name_of_the_file_with_station) [pure virtual]`

Загрузить входные данные из файла

Parameters

<code>name_of_the_file_with_route</code>	- название файла, для хранения информации о маршрутах
<code>name_of_the_file_with_station</code>	- название файла, для хранения информации о станциях Информация должна находиться в таком же формате, как и в <code>putInfoAboutMetro</code>

Exceptions

При	неудачном открытии файла бросается MissingFile , от объекта которого можно получить информацию о неверном имени файла
-----	---

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.15 virtual void API::putInfoAboutMetro (const std::string & infoAboutRoutes, const std::string & infoAboutStations) [pure virtual], [noexcept]

Один из способ передать входные данные

Parameters

infoAboutRoutes	- информация о маршрутах в формате: Devyatkino,Grazhdansky Prospekt/Parnas,Prospekt Prosvescheniya где через запятую указываются название станций, а слеш отделяет маршруты
infoAboutStations	- информация о станциях в формате: Parnas~5.47-0.00/Prospekt Prosvescheniya~5.37-0.40 где слеш отделяет станции, а тильда отделяет название станции от информации связаной с ней

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.16 virtual void API::putOfRights (const Rights_of_customers rights) [pure virtual], [noexcept]

Выдача прав

Parameters

rights	- какие права нужно выдать
--------	----------------------------

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.17 virtual void API::removeInfoAboutStation (const std::string & what_station_to_remove) [pure virtual]

Удаляет информацию о станции

Parameters

what_station_to_remove	- название станции, которую нужно удалить
------------------------	---

Exceptions

При	неверном запросе бросается StationDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.18 virtual void API::removeInfoAboutStation (const int number_of_the_station) [pure virtual]

Альтернативный путь удаления информации о станции, в месте с ним используется метод [getAllStationsWhichHaveDescription\(\)](#), именно из него берётся номер станции

Parameters

number_of_the_station	- номер станции из getAllStationsWh..
-----------------------	---------------------------------------

Exceptions

При	неверном запросе бросается StationDoesNotExist
-----	--

Implemented in [CoreOfInfoAboutMetro](#).

3.1.2.19 virtual void API::saveChanges (const std::string & name_of_the_file_with_route, const std::string & name_of_the_file_with_station) [pure virtual], [noexcept]

Сохранить информацию

Parameters

name_of_the_file_with_route	- название файла, для хранения информации о маршрутах
name_of_the_file_with_station	- название файла, для хранения информации о станциях

Implemented in [CoreOfInfoAboutMetro](#).

The documentation for this class was generated from the following file:

- Core/api.h

3.2 ConsoleForTimetable Class Reference

Класс служащий для работы с ядром через консоль

```
#include <app.h>
```

3.2.1 Detailed Description

Класс служащий для работы с ядром через консоль

The documentation for this class was generated from the following files:

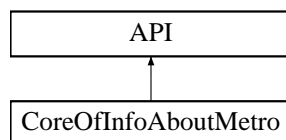
- Console/app.h
- Console/app.cpp

3.3 CoreOfInfoAboutMetro Class Reference

Класс в котором содержится основная бизнес-логика приложения

```
#include <core.h>
```

Inheritance diagram for CoreOfInfoAboutMetro:



Public Member Functions

- void [putOfRights](#) (const Rights_of_customers rights) noexcept override
Выдача прав
- Rights_of_customers [getInformationOfTheRights](#) () const noexcept override
- void [putInfoAboutMetro](#) (const std::string &infoAboutRoutes, const std::string &infoAboutStations) noexcept override
Один из способ передать входные данные
- void [loadInfoFromFile](#) (const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station) override
Загрузить входные данные из файла
- int [howManyRoutes](#) () const noexcept override
- std::vector< std::string > [getRoute](#) (const int number_of_the_route) override
- std::string [getInfoAboutStation](#) (const std::string &name_of_the_station) override
- std::string [getInfoAboutStation](#) (const int number_of_the_route, const int number_of_the_station) override
Способ альтернативный предидущему
- void [changeStationInRoute](#) (const int number_of_the_route, const int number_of_the_station, const std::string &new_marking) override
Изменение станции в маршруте
- void [addStationInRoute](#) (const int number_of_the_route, const std::string &what_to_add) override
Добавляет станцию в маршрут
- int [addRoute](#) () noexcept override
Добавляет новый (пустой) маршрут
- void [deleteStationFromRoute](#) (const int number_of_the_route, const int number_of_the_station) override
Удаляет станцию из маршрута
- void [deleteRoute](#) (const int number_of_the_route) override
Удаляет маршрут
- void [addInfoAboutStation](#) (const std::string &name_of_the_station, const std::string &station_description) noexcept override
Добавляет информацию о станции
- void [addInfoAboutStation](#) (const int number_of_the_route, const int number_of_the_station, const std::string &station_description) override
Альтернативный путь добавления информации о станции
- void [removeInfoAboutStation](#) (const std::string &what_station_to_remove) override
Удаляет информацию о станции

- void [removeInfoAboutStation](#) (const int number_of_the_station) override
Альтернативный путь удаления информации о станции, в месте с ним используется метод [get↔AllStationsWhichHaveDescription\(\)](#), именно из него берётся номер станции
- std::vector< std::pair< std::string, std::string > > [getAllStationsWhichHaveDescription](#) () noexcept override
- void [saveChanges](#) (const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station) noexcept override
Сохранить информацию

3.3.1 Detailed Description

Класс в котором содержится основная бизнес-логика приложения

3.3.2 Member Function Documentation

3.3.2.1 void CoreOfInfoAboutMetro::addInfoAboutStation (const std::string &name_of_the_station, const std::string &station_description) [override], [virtual], [noexcept]

Добавляет информацию о станции

Parameters

name_of_the_station	- номер станции, о которой информация
station_description	- информация о станции

Implements [API](#).

3.3.2.2 void CoreOfInfoAboutMetro::addInfoAboutStation (const int number_of_the_route, const int number_of_the_station, const std::string &station_description) [override], [virtual]

Альтернативный путь добавления информации о станции

Parameters

number_of_the_route	- номер маршрута, в котором содержится станция, про которую информация
number_of_the_station	- номер станции, про которую информация
station_description	- информация о станции

Exceptions

При	неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
-----	--

Implements [API](#).

3.3.2.3 `int CoreOfInfoAboutMetro::addRoute () [override], [virtual], [noexcept]`

Добавляет новый (пустой) маршрут

Returns

Номер добавленного маршрута

Implements [API](#).

3.3.2.4 `void CoreOfInfoAboutMetro::addStationInRoute (const int number_of_the_route, const std::string & what_to_add) [override], [virtual]`

Добавляет станцию в маршрут

Parameters

<code>number_of_the_route</code>	- номер маршрута, в который нужно добавить станцию
<code>what_to_add</code>	- название станции, которую нужно добавить

Exceptions

При	неверном запросе бросается RouteDoesNotExist
-----	--

Implements [API](#).

3.3.2.5 `void CoreOfInfoAboutMetro::changeStationInRoute (const int number_of_the_route, const int number_of_the_station, const std::string & new_marking) [override], [virtual]`

Изменение станции в маршруте

Parameters

<code>number_of_the_route</code>	- номер маршрута
<code>number_of_the_station</code>	- номер станции
<code>new_marking</code>	- то, что нужно поставить в замен

Exceptions

При	неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
-----	--

Implements [API](#).

3.3.2.6 `void CoreOfInfoAboutMetro::deleteRoute (const int number_of_the_route) [override], [virtual]`

Удаляет маршрут

Parameters

number_of_the_route	- номер маршрута, который нужно удалить
---------------------	---

Exceptions

При	неверном запросе бросается RouteDoesNotExist
-----	--

Implements [API](#).

3.3.2.7 void CoreOfInfoAboutMetro::deleteStationFromRoute (const int number_of_the_route, const int number_of_the_station) [override], [virtual]

Удаляет станцию из маршрута

Parameters

number_of_the_route	- номер маршрута, из которого нужно удалить станцию
number_of_the_station	- номер станции, которую нужно удалить

Exceptions

При	неверном запросе бросаются StationDoesNotExist и RouteDoesNotExist
-----	--

Implements [API](#).

3.3.2.8 std::vector< std::pair< std::string, std::string > > CoreOfInfoAboutMetro::getAllStationsWhichHaveDescription () [override], [virtual], [noexcept]

Returns

Всё станции, про которые содержится информация, первая часть пары название станции, вторая информация о ней

Implements [API](#).

3.3.2.9 std::string CoreOfInfoAboutMetro::getInfoAboutStation (const std::string & name_of_the_station) [override], [virtual]

Parameters

name_of_the_station	- название станции, о которой нужно получить информацию
---------------------	---

Returns

информация о запрашиваемой станции

Exceptions

При	запросе несуществующей станции бросается StationDoesNotExist
-----	--

Implements [API](#).

3.3.2.10 `std::string CoreOfInfoAboutMetro::getInfoAboutStation (const int number_of_the_route, const int number_of_the_station) [override], [virtual]`

Способ альтернативный предидущему

Parameters

<code>number_of_the_route</code>	- номер маршрута, содержащего станцию
<code>number_of_the_station</code>	- номер станции в маршруте

Returns

информация о запрашиваемой станции

Exceptions

При	запросе несуществующей станции бросается StationDoesNotExist , а RouteDoesNotExist , если маршрута не существует
-----	--

Implements [API](#).

3.3.2.11 `Rights_of_customers CoreOfInfoAboutMetro::getInformationOfTheRights () const [override], [virtual], [noexcept]`

Returns

Информация о правах

Implements [API](#).

3.3.2.12 `std::vector< std::string > CoreOfInfoAboutMetro::getRoute (const int number_of_the_route) [override], [virtual]`

Parameters

<code>number_of_the_route</code>	- номер запрашиваемого маршрута
----------------------------------	---------------------------------

Returns

Запрашиваемый маршрут в формате `vector<string>` , где `string` - это название станций

Exceptions

При	запросе несуществующего маршрута бросается RouteDoesNotExist , от объекта которого можно получить информацию, какой маршрут запрашивали
-----	---

Implements [API](#).

3.3.2.13 `int CoreOfInfoAboutMetro::howManyRoutes () const` `[override]`, `[virtual]`, `[noexcept]`

Returns

Сколько маршрутов существует на данный момент

Implements [API](#).

3.3.2.14 `void CoreOfInfoAboutMetro::loadInfoFromFile (const std::string & name_of_the_file_with_route, const std::string & name_of_the_file_with_station)` `[override]`, `[virtual]`

Загрузить входные данные из файла

Parameters

<code>name_of_the_file_with_route</code>	- название файла, для хранения информации о маршрутах
<code>name_of_the_file_with_station</code>	- название файла, для хранения информации о станциях Информация должна находиться в таком же формате, как и в <code>putInfoAboutMetro</code>

Exceptions

При	неудачном открытии файла бросается MissingFile , от объекта которого можно получить информацию о неверно имени файла
-----	--

Implements [API](#).

3.3.2.15 `void CoreOfInfoAboutMetro::putInfoAboutMetro (const std::string & infoAboutRoutes, const std::string & infoAboutStations)` `[override]`, `[virtual]`, `[noexcept]`

Один из способ передать входные данные

Parameters

<code>infoAboutRoutes</code>	- информация о маршрутах в формате: Devyatkin, Grazhdansky Prospekt/Parnas, Prospekt Prosvescheniya где через запятую указываются названия станций, а слеш отделяет маршруты
------------------------------	--

Parameters

infoAboutStations	- информация о станциях в формате: Parnas~5.47-0.00/Prospekt Prosvescheniya~5.37-0.40 где слеш отделяет станции, а тильда отделяет название станции от информации связанной с ней
-------------------	---

Implements [API](#).

3.3.2.16 void CoreOfInfoAboutMetro::putOfRights (const Rights_of_customers rights) [override], [virtual], [noexcept]

Выдача прав

Parameters

rights	- какие права нужно выдать
--------	----------------------------

Implements [API](#).

3.3.2.17 void CoreOfInfoAboutMetro::removeInfoAboutStation (const std::string & what_station_to_remove) [override], [virtual]

Удаляет информацию о станции

Parameters

what_station_to_remove	- название станции, которую нужно удалить
------------------------	---

Exceptions

При	неверном запросе бросается StationDoesNotExist
-----	--

Implements [API](#).

3.3.2.18 void CoreOfInfoAboutMetro::removeInfoAboutStation (const int number_of_the_station) [override], [virtual]

Альтернативный путь удаления информации о станции, в месте с ним используется метод [getAllStationsWhichHaveDescription\(\)](#), именно из него берётся номер станции

Parameters

number_of_the_station	- номер станции из getAllStationsWh..
-----------------------	---------------------------------------

Exceptions

При	неверном запросе бросается StationDoesNotExist
-----	--

Implements [API](#).

```
3.3.2.19 void CoreOfInfoAboutMetro::saveChanges ( const std::string & name_of_the_file_with↵
        _route, const std::string & name_of_the_file_with_station ) [override], [virtual],
        [noexcept]
```

Сохранить информацию

Parameters

name_of_the_file_with_route	- название файла, для хранения информации о маршрутах
name_of_the_file_with_station	- название файла, для хранения информации о станциях

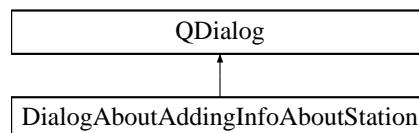
Implements [API](#).

The documentation for this class was generated from the following files:

- Core/core.h
- Core/core.cpp

3.4 DialogAboutAddingInfoAboutStation Class Reference

Inheritance diagram for DialogAboutAddingInfoAboutStation:



Signals

- void addInformationAboutStation ()

Public Member Functions

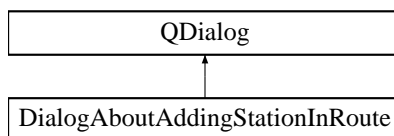
- DialogAboutAddingInfoAboutStation ([CoreOfInfoAboutMetro](#) *core, int *number_of_the_route, int *number_of_the_station, QString *information_about_station, QMainWindow *m_window, QWidget *parent=0)

The documentation for this class was generated from the following files:

- GUI/dialogaboutaddinginfoaboutstation.h
- GUI/dialogaboutaddinginfoaboutstation.cpp

3.5 DialogAboutAddingStationInRoute Class Reference

Inheritance diagram for DialogAboutAddingStationInRoute:



Signals

- void addStationSignal ()

Public Member Functions

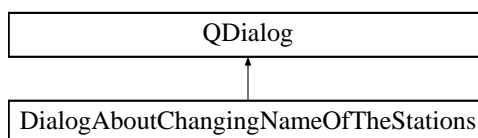
- DialogAboutAddingStationInRoute ([CoreOfInfoAboutMetro](#) *core, int *index_out_of, QString *name_of_adding_station, QMainWindow *m_window, QWidget *parent=0)

The documentation for this class was generated from the following files:

- GUI/dialogaboutaddingstationinroute.h
- GUI/dialogaboutaddingstationinroute.cpp

3.6 DialogAboutChangingNameOfTheStations Class Reference

Inheritance diagram for DialogAboutChangingNameOfTheStations:



Signals

- void changeNameOfStation ()

Public Member Functions

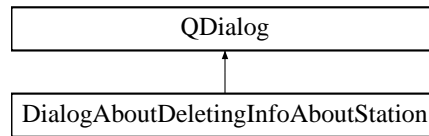
- DialogAboutChangingNameOfTheStations ([CoreOfInfoAboutMetro](#) *core, int *number_of_the_route, int *number_of_the_station, QString *new_name, QMainWindow *m_window, QWidget *parent=0)

The documentation for this class was generated from the following files:

- GUI/dialogaboutchangingnameofthestations.h
- GUI/dialogaboutchangingnameofthestations.cpp

3.7 DialogAboutDeletingInfoAboutStation Class Reference

Inheritance diagram for DialogAboutDeletingInfoAboutStation:



Signals

- void deleteInfoAboutStation ()

Public Member Functions

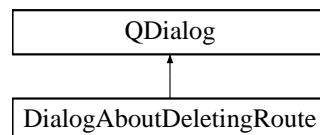
- DialogAboutDeletingInfoAboutStation ([CoreOfInfoAboutMetro](#) *core, int *number_of_the_station, QMainWindow *m_window, QWidget *parent=0)

The documentation for this class was generated from the following files:

- GUI/dialogaboutdeletinginfoaboutstation.h
- GUI/dialogaboutdeletinginfoaboutstation.cpp

3.8 DialogAboutDeletingRoute Class Reference

Inheritance diagram for DialogAboutDeletingRoute:



Signals

- void deleteRouteSignal ()

Public Member Functions

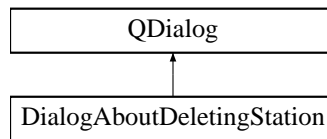
- DialogAboutDeletingRoute ([CoreOfInfoAboutMetro](#) *core, int *index_out_of, QMainWindow *m_window, QWidget *parent=0)

The documentation for this class was generated from the following files:

- GUI/dialogaboutdeletingroute.h
- GUI/dialogaboutdeletingroute.cpp

3.9 DialogAboutDeletingStation Class Reference

Inheritance diagram for DialogAboutDeletingStation:



Signals

- void deletingStation ()

Public Member Functions

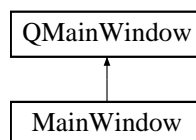
- DialogAboutDeletingStation ([CoreOffInfoAboutMetro](#) *core, int *number_of_the_route, int *number_of_the_station, QMainWindow *m_window, QWidget *parent=0)

The documentation for this class was generated from the following files:

- GUI/dialogaboutdeletingstation.h
- GUI/dialogaboutdeletingstation.cpp

3.10 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=0)

The documentation for this class was generated from the following files:

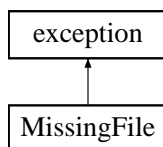
- GUI/mainwindow.h
- GUI/mainwindow.cpp

3.11 MissingFile Class Reference

Исключение, которое бросается при неудачном открытии файла, от него можно получить, информацию о том, какой файл запрашивал пользователь, перед появлением исключения

```
#include <missing_file.h>
```

Inheritance diagram for MissingFile:



Public Member Functions

- [MissingFile](#) (std::string requested)
Используется, при необходимости передать информацию об ошибке
- std::string [getWhatRequested](#) () noexcept

3.11.1 Detailed Description

Исключение, которое бросается при неудачном открытии файла, от него можно получить, информацию о том, какой файл запрашивал пользователь, перед появлением исключения

3.11.2 Constructor & Destructor Documentation

3.11.2.1 MissingFile::MissingFile (std::string requested) [inline], [explicit]

Используется, при необходимости передать информацию об ошибке

Parameters

requested	- название файла, который хотел открыть пользователь
-----------	--

3.11.3 Member Function Documentation

3.11.3.1 std::string MissingFile::getWhatRequested () [inline], [noexcept]

Returns

Возвращаем информацию о том, из-за чего было возбуждено исключение

The documentation for this class was generated from the following file:

- Core/exception_of_core/missing_file.h

3.12 ParsingInfo Class Reference

The [ParsingInfo](#) class.

```
#include <parsing_info.h>
```

Public Member Functions

- void [putInfoAboutRoutes](#) (const std::string &info, [RoutesInfo](#) &routeInfo) noexcept
Получает на вход строку, парсит её, а потом заполняет информацию о маршрутах
- void [putInfoAboutStations](#) (const std::string &info, [StationsInfo](#) &stationInfo) noexcept
Получает на вход строку, парсит её, а потом заполняет информацию о станциях
- void [loadFromFile](#) (const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station, [RoutesInfo](#) &routeInfo, [StationsInfo](#) &stationInfo)
Загружает данные из файла и заполняет маршруты и информацию о станциях, путём вызова двух пред идущих функций
- void [saveChanges](#) (const std::string &name_of_the_file_with_route, const std::string &name_of_the_file_with_station, [RoutesInfo](#) &routeInfo, [StationsInfo](#) &stationInfo)
Сохраняет изменения
- void [saveChangesForRoute](#) (const std::string &name_of_the_file_with_route, [RoutesInfo](#) &routeInfo)
Сохраняет изменения проведённые со станциями
- void [saveChangesForStation](#) (const std::string &name_of_the_file_with_station, [StationsInfo](#) &stationInfo)
Сохраняет изменения проведённые с маршрутами

3.12.1 Detailed Description

The [ParsingInfo](#) class.

3.12.2 Member Function Documentation

- 3.12.2.1 void [ParsingInfo::loadFromFile](#) (const std::string & name_of_the_file_with_route, const std::string & name_of_the_file_with_station, [RoutesInfo](#) & routeInfo, [StationsInfo](#) & stationInfo)

Загружает данные из файла и заполняет маршруты и информацию о станциях, путём вызова двух пред идущих функций

Parameters

name_of_the_file_with_route	- название файла, содержащего информацию о маршрутах
name_of_the_file_with_station	- название файла, содержащего информацию о станциях
routeInfo	- то, с чем работают заполнители
stationInfo	- то, с чем работают заполнители

Exceptions

При	неудачном открытии файла бросается MissingFile
-----	--

3.12.2.2 void ParsingInfo::putInfoAboutRoutes (const std::string & info, RoutesInfo & routeInfo)
[noexcept]

Получает на вход строчку, парсит её, а потом заполняет информацию о маршрутах

Parameters

info	- строка, которую надо парсить
routeInfo	- то, с чем работают заполнители

3.12.2.3 void ParsingInfo::putInfoAboutStations (const std::string & info, StationsInfo & stationInfo)
[noexcept]

Получает на вход строчку, парсит её, а потом заполняет информацию о станциях

Parameters

info	- строка, которую нужно парсить
stationInfo	- то, с чем работают заполнители

3.12.2.4 void ParsingInfo::saveChanges (const std::string & name_of_the_file_with_route, const std::string & name_of_the_file_with_station, RoutesInfo & routeInfo, StationsInfo & stationInfo)

Сохраняет изменения

Parameters

name_of_the_file_with_route	- название файла, содержащего информацию о маршрутах
name_of_the_file_with_station	- название файла, содержащего информацию о станциях
routeInfo	- то, с чем работают заполнители
stationInfo	- то, с чем работают заполнители

3.12.2.5 void ParsingInfo::saveChangesForRoute (const std::string & name_of_the_file_with_route, RoutesInfo & routeInfo)

Сохраняет изменения проведённые со станциями

Parameters

name_of_the_file_with_route	- название файла, содержащего информацию о станциях
routeInfo	- то, с чем работают заполнители

3.12.2.6 void ParsingInfo::saveChangesForStation (const std::string & name_of_the_file_with_station, StationsInfo & stationInfo)

Сохраняет изменения проведённые с маршрутами

Parameters

name_of_the_file_with_station	- название файла, содержащего информацию о маршрутах
stationInfo	- то, с чем работают заполнители

The documentation for this class was generated from the following files:

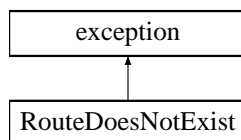
- Core/handling_info/parsing_info.h
- Core/handling_info/parsing_info.cpp

3.13 RouteDoesNotExist Class Reference

Исключение бросается в том случае, если оказывается, что запрашиваемого маршрута не существует

```
#include <route_does_not_exist.h>
```

Inheritance diagram for RouteDoesNotExist:



Public Member Functions

- [RouteDoesNotExist](#) (int requested)
Используется, при необходимости передать информацию об ошибке
- int [getWhatRequested](#) () noexcept

3.13.1 Detailed Description

Исключение бросается в том случае, если оказывается, что запрашиваемого маршрута не существует

3.13.2 Constructor & Destructor Documentation

3.13.2.1 RouteDoesNotExist::RouteDoesNotExist (int requested) [inline], [explicit]

Используется, при необходимости передать информацию об ошибке

Parameters

requested	- номер маршрута, который не существует, но был запрошен пользователем
-----------	--

3.13.3 Member Function Documentation

3.13.3.1 int RouteDoesNotExist::getWhatRequested () [inline], [noexcept]

Returns

Возвращаем информацию о том, из-за чего было возбуждено исключение

The documentation for this class was generated from the following file:

- Core/exception_of_core/route_does_not_exist.h

3.14 RoutesInfo Class Reference

Класс отвечающий за работу с информацией связанной с маршрутами

```
#include <routes_info.h>
```

Public Member Functions

- std::vector< std::string > [getRoute](#) (const int number_of_the_route) noexcept
- int [getHowManyRoutes](#) () const noexcept
- void [changeStationInRoute](#) (const int number_of_the_route, const int number_of_the_station, const std::string &new_marking) noexcept
Изменяет название станции в маршруте
- void [addStationInRoute](#) (const int number_of_the_route, const std::string &what_add) noexcept
Добавляет новую станцию в маршрут
- void [addRoute](#) () noexcept
Добавляет новый маршрут
- void [deleteStationFromRoute](#) (const int number_of_the_route, const int number_of_the_station) noexcept
Удаляет станцию из маршрута
- void [deleteRoute](#) (const int number_of_the_route) noexcept
Удаляет маршрут

3.14.1 Detailed Description

Класс отвечающий за работу с информацией связанной с маршрутами

3.14.2 Member Function Documentation

3.14.2.1 void RoutesInfo::addStationInRoute (const int number_of_the_route, const std::string & what_add) [noexcept]

Добавляет новую станцию в маршрут

Parameters

number_of_the_route	- номер маршрута
what_add	- название добавляемой станции

3.14.2.2 void RoutesInfo::changeStationInRoute (const int number_of_the_route, const int number_of_the_station, const std::string & new_marking) [noexcept]

Изменяет название станции в маршруте

Parameters

number_of_the_route	- номер маршрута
number_of_the_station	- номер станции
new_marking	- новое название станции

3.14.2.3 void RoutesInfo::deleteRoute (const int number_of_the_route) [noexcept]

Удаляет маршрут

Parameters

number_of_the_route	- номер маршрута, который нужно удалить
---------------------	---

3.14.2.4 void RoutesInfo::deleteStationFromRoute (const int number_of_the_route, const int number_of_the_station) [noexcept]

Удаляет станцию из маршрута

Parameters

number_of_the_route	- номер маршрута, из которого надо удалить станцию
number_of_the_station	- номер станции, которую надо удалить

3.14.2.5 int RoutesInfo::getHowManyRoutes () const [noexcept]

Returns

Возвращает информацию, о том что сколько маршрутов существует на данный момент

3.14.2.6 std::vector< std::string > RoutesInfo::getRoute (const int number_of_the_route) [noexcept]

Parameters

<code>number_of_the_route</code>	- номер запрашиваемого маршрута
----------------------------------	---------------------------------

Returns

Запрашиваемый маршрут

The documentation for this class was generated from the following files:

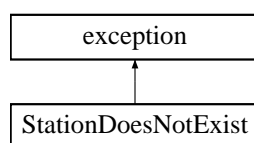
- Core/handling_info/routes_info.h
- Core/handling_info/routes_info.cpp

3.15 StationDoesNotExist Class Reference

Исключение бросается в том случае, если оказывается, что запрашиваемой станции не существует

```
#include <station_does_not_exist.h>
```

Inheritance diagram for StationDoesNotExist:



Public Member Functions

- [StationDoesNotExist](#) (const std::string &requested)
Используется при необходимости передать информацию об ошибке, в том случае, если она в название станции
- [StationDoesNotExist](#) (int requested)
Используется при необходимости передать информацию об ошибке, в том случае, если она в номере станции
- std::string [getWhatRequested](#) () noexcept

3.15.1 Detailed Description

Исключение бросается в том случае, если оказывается, что запрашиваемой станции не существует

3.15.2 Constructor & Destructor Documentation

3.15.2.1 `StationDoesNotExist::StationDoesNotExist (const std::string & requested) [inline], [explicit]`

Используется при необходимости передать информацию об ошибке, в том случае, если она в название станции

Parameters

requested	- название станции, которая не существует, но была запрошена пользователем
-----------	--

3.15.2.2 StationDoesNotExist::StationDoesNotExist (int requested) [inline], [explicit]

Используется при необходимости передать информацию об ошибке, в том случае, если она в номере станции

Parameters

requested	- номер станции, которая не существует, но была запрошена пользователем
-----------	---

3.15.3 Member Function Documentation

3.15.3.1 std::string StationDoesNotExist::getWhatRequested () [inline], [noexcept]

Returns

Возвращаем информацию о том, из-за чего было возбуждено исключение

The documentation for this class was generated from the following file:

- Core/exception_of_core/station_does_not_exist.h

3.16 StationsInfo Class Reference

Класс отвечающий за работу с информацией связанной со станциями

```
#include <stations_info.h>
```

Public Member Functions

- std::string [getInfoAboutStation](#) (const std::string &name_of_the_station) noexcept
Получить информацию о станции
- void [addInfoAboutStation](#) (const std::string &name_of_the_station, const std::string &station←_description) noexcept
Добавляет информацию о станции, если станция уже существует, то старая информация удаляется
- void [removeInfoAboutStation](#) (const std::string &what_station_to_remove) noexcept
Удаляет информацию о станции
- std::vector< std::pair< std::string, std::string > > [getAllStations](#) () noexcept

3.16.1 Detailed Description

Класс отвечающий за работу с информацией связанной со станциями

3.16.2 Member Function Documentation

3.16.2.1 void StationsInfo::addInfoAboutStation (const std::string & name_of_the_station, const std::string & station_description) [noexcept]

Добавляет информацию о станции, если станция уже существует, то старая информация удаляется

Parameters

name_of_the_station	- название станции, которую нужно добавить
station_description	- информация о станции

3.16.2.2 `std::vector< std::pair< std::string, std::string > > StationsInfo::getAllStations ()`
`[noexcept]`

Returns

Возвращает все станции, к которым присутствует описание Этот метод используется, для вывода всех станци с описанием, чтобы потом выбрать определённую и удалить

3.16.2.3 `std::string StationsInfo::getInfoAboutStation (const std::string & name_of_the_station)`
`[noexcept]`

Получить информацию о станции

Parameters

name_of_the_station	- название станции, о которой хотят получить информацию
---------------------	---

Returns

Информация о станции

3.16.2.4 `void StationsInfo::removeInfoAboutStation (const std::string & what_station_to_remove)`
`[noexcept]`

Удаляет информацию о станции

Parameters

what_station_to_remove	- название станции, информацию, о которой нужно удалить
------------------------	---

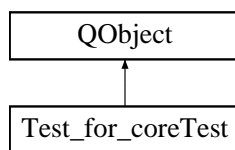
The documentation for this class was generated from the following files:

- Core/handling_info/stations_info.h
- Core/handling_info/stations_info.cpp

3.17 Test_for_coreTest Class Reference

Тесты для функциональностей предоставляемых ядром

Inheritance diagram for Test_for_coreTest:



Public Member Functions

- [Test_for_coreTest](#) ()
Слабоватенькие тесты

3.17.1 Detailed Description

Тесты для функциональностей предоставляемых ядром

The documentation for this class was generated from the following file:

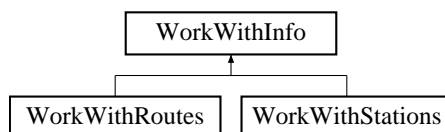
- Tests/tst_test_for_coretest.cpp

3.18 WorkWithInfo Class Reference

Абстрактный класс, содержащий несколько функций помогающих при работе с пользователем

```
#include <work_with_info.h>
```

Inheritance diagram for WorkWithInfo:



Public Member Functions

- int [getIntFromConsole](#) ()
- void [displayRoute](#) (std::vector< std::string > &output_for_console)
Удобный вывод маршрута в консоль
- virtual [~WorkWithInfo](#) ()
Так как наследники должны иметь свой
- [WorkWithInfo](#) (const [WorkWithInfo](#) &)=default
- [WorkWithInfo](#) & [operator=](#) (const [WorkWithInfo](#) &)=default
Копирующее присваивание

3.18.1 Detailed Description

Абстрактный класс, содержащий несколько функций помогающих при работе с пользователем

3.18.2 Constructor & Destructor Documentation

3.18.2.1 WorkWithInfo::WorkWithInfo (const WorkWithInfo &) [default]

Они мне не нужны, но по правилу: переопределил деструктор, переопределяй и эти конструкторы
Конструктор копирования

3.18.3 Member Function Documentation

3.18.3.1 void WorkWithInfo::displayRoute (std::vector< std::string > & output_for_console)

Удобный вывод маршрута в консоль

Parameters

output_for_console	- что за маршрут нужно вывести
--------------------	--------------------------------

Выводится в виде: 1.Parnas

2.Prospekt Prosvescheniya

3.18.3.2 int WorkWithInfo::getIntFromConsole ()

Returns

Число считанное из консоли

Мешает считать кучу символов(ведь нам нужен один)

The documentation for this class was generated from the following files:

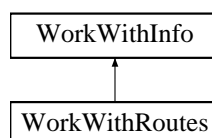
- Console/workwithinfo/work_with_info.h
- Console/workwithinfo/work_with_info.cpp

3.19 WorkWithRoutes Class Reference

Класс, отвечающий за обработку запросов связанных с маршрутами, с помощью консоли

```
#include <work_with_routes.h>
```

Inheritance diagram for WorkWithRoutes:



Public Member Functions

- void `routeInformation` (`CoreOfInfoAboutMetro &core`)
Предоставляет информацию о запрашиваемом маршруте
- void `changeItinerary` (`CoreOfInfoAboutMetro &core`)
Методы для администратора(доступны только в режиме администратора):
- void `changeRoute` (`CoreOfInfoAboutMetro &core`)
Предоставляет пользователю возможность изменить конкретный маршрут. Добавить, удалить и переименовать станцию
- void `addRoute` (`CoreOfInfoAboutMetro &core`)
Добавляет новый маршрут(пустой) с номером, на один больше чем последний существующий
- void `deleteRoute` (`CoreOfInfoAboutMetro &core`)
Удаляет маршрут с указанным номером, причём маршруты чей номер больше указанного сдвигаются вниз

3.19.1 Detailed Description

Класс, отвечающий за обработку запросов связанных с маршрутами, с помощью консоли

3.19.2 Member Function Documentation

3.19.2.1 void WorkWithRoutes::changeItinerary (CoreOfInfoAboutMetro & core)

Методы для администратора(доступны только в режиме администратора):

Предоставляет пользователю возможность изменения маршрутов. Изменить маршрут, удалить маршрут, добавить маршрут. Пользователь не сможет вызвать метод, если он не админ.

The documentation for this class was generated from the following files:

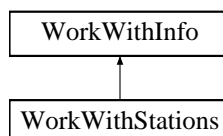
- Console/workwithinfo/work_with_routes.h
- Console/workwithinfo/work_with_routes.cpp

3.20 WorkWithStations Class Reference

Класс, отвечающий за обработку запросов связанных со станциями, с помощью консоли

```
#include <work_with_stations.h>
```

Inheritance diagram for WorkWithStations:



Public Member Functions

- void [informationAboutStation](#) ([CoreOfInfoAboutMetro](#) &core)
В зависимости от запрашиваемой станции, печатает информацию о ней
- void [changeInfoAboutStation](#) ([CoreOfInfoAboutMetro](#) &core)
Методы для администратора(доступны только в режиме администратора):
- void [addOrChangeInformationAboutStation](#) ([CoreOfInfoAboutMetro](#) &core)
Добавить информацию о станции
- void [removeInformationAboutStation](#) ([CoreOfInfoAboutMetro](#) &core)
Удалить информацию о станции

3.20.1 Detailed Description

Класс, отвечающий за обработку запросов связанных со станциями, с помощью консоли

3.20.2 Member Function Documentation

3.20.2.1 void [WorkWithStations::changeInfoAboutStation](#) ([CoreOfInfoAboutMetro](#) & core)

Методы для администратора(доступны только в режиме администратора):

Предоставляет пользователю возможность изменения информации о станции Добавить информацию о станции, удалить её

The documentation for this class was generated from the following files:

- Console/workwithinfo/work_with_stations.h
- Console/workwithinfo/work_with_stations.cpp