Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра экономической информатики

Дисциплина «Технологии разработки программного обеспечения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.И.Полоско |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«РАЗРАБОТКА ПРОГРАММЫ РАСПИСАНИЯ ДВИЖЕНИЯ АВТОБУСОВ »**

БГУИР КР 1-40 05 01-12 003 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 224402  Бурая Александра Константиновна  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**РЕФЕРАТ**

БГУИР КР 1-40 05 01-12 003 ПЗ

**Бурая, А. К.** Разработка программы расписания движения автобусов/А.К. Бурая. – Минск: БГУИР, 2023. – 43 с.

Пояснительная записка 43 с., 43 рис., 4 табл., 4 источника, 2 приложения.

Расписание движения автобусов, модели *UML*, *IDEF*0, *BPMN*, схемы алгоритмов, программное средство

*Цель* *курсовой работы*: уменьшение временных затрат на поиск расписания автобусов. Программа позволяет вносить корректировки в уже имеющиеся данные, создавать новые записи, удалять старые и сохранять отредактированные записи.

*Методология проведения работы*: в процессе решения поставленных задач использованы принципы системного подхода, аналитические методы, методы компьютерной обработки экспериментальных данных и компьютерного моделирования.

*Результаты работы*: выполнена постановка задачи и определены основные методы ее решения; в ходе объектного моделирования системы построен ряд *UML*-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0 и BPMN; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку.

Программный продукт разработан на языке C++ с применением MS Visual Studio 2022.

*Область применения результатов*: c помощью разработанного программного средства можно проанализировать расписание автобусов.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при учете и анализе данных общественного транспорта

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc136411581)

[1 Анализ и моделирование предметной области программного средства 5](#_Toc136411582)

[1.1 Описание предметной области 5](#_Toc136411583)

[1.2 Разработка функциональной модели предметной области 5](#_Toc136411584)

[1.3 Разработка BPMN-модели основного процесса предметной области 9](#_Toc136411585)

[1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 10](#_Toc136411586)

[1.5 UML-модели представления программного обеспечения и их описание10](#_Toc136411587)

[2 Проектирование и конструирование программного средства 14](#_Toc136411588)

[2.1 Постановка задачи 14](#_Toc136411589)

[2.2 Разработка модульной структуры 14](#_Toc136411590)

[2.3 Выбор способа организации данных 15](#_Toc136411591)

[2.4 Разработка перечня пользовательских функций программы 15](#_Toc136411592)

[2.5 Разработка схем алгоритмов программы 17](#_Toc136411593)

[3 Тестирование и проверка работоспособности программного средства 24](#_Toc136411594)

[4 Инструкция по развертыванию приложения и сквозной тестовый пример 28](#_Toc136411595)

[4.1 Авторизация 28](#_Toc136411596)

[4.2 Модуль пользователя 29](#_Toc136411597)

[4.3 Модуль администратора 30](#_Toc136411598)

[Заключение 35](#_Toc136411599)

[Список использованных источников 36](#_Toc136411600)

[Приложение А 37](#_Toc136411601)

[Приложение В 42](#_Toc136411602)

# **Введение**

В современном мире, где населенность городов продолжает расти, создание эффективной транспортной инфраструктуры является важной задачей для городской администрации. Современный общественный транспорт является недорогим и удобным средством передвижения по городу: часто на метро или автобусе можно добраться до заданной точки быстрее, чем на собственной машине.

Использование общественного транспорта способствует уменьшению количества автомобилей на дорогах и, как следствие, помогает снизить загрязнение окружающей среды. Транспорт городов по всему миру развиваются с учетом запросов на комфорт, экологичность и безопасность. Традиционные средства передвижения (работающие на дизельном топливе и двигателях внутреннего сгорания) уступают место более экологичным электромобилям и электробусам.

Благодаря развитию технологий пассажирам больше не нужно гадать, когда приедет их автобус, информацию в реальном времени можно получить на информационных табло на остановках или через интернет; а купить билет– не только в киосках или транспорте, но и онлайн. Информационные системы, которые получают, сортируют и хранят данные о существующих маршрутах, названиях остановок и расписании автобусов, позволяют пассажирам получать актуальную информацию о городском транспорте [1].

Целью данного курсового проекта является уменьшение временных затрат на поиск расписания автобусов. Программа предназначена для просмотра актуального расписания и информации об автобусах, поиска маршрута по номеру, расчета времени до прибытия ближайшего транспортного средства на определенную остановку.

Для достижения поставленной цели необходимо решить следующие задачи:

* ознакомиться с предметной областью;
* проанализировать особенности создания маршрутов и расписания автобусов;
* разработать алгоритмы работы программного средства;
* разработать модули структуры программного средства;
* протестировать работу программного средства.

Объектом исследования является расписание автобусов.

Таким образом, разработка программы расписания движения автобусов является актуальной и важной задачей, направленной на повышение эффективности и удобства использования общественного транспорта.

# **1 Анализ и моделирование предметной области программного средства**

## **1.1 Описание предметной области**

Самым распространённым видом общественного транспорта является – городской автобус, он и был взят за основу данного проекта. Чтобы воспользоваться этим средством передвижения пассажиру необходимо спланировать оптимальный маршрут, затем определить возможные остановки для пересадки на другой транспорт, найти ближайшую остановку и сесть в автобус. Также необходимо оплатить проезд, купив проездной талон в киоске или у водителя, а затем пробить его в транспорте. Оплатить проезд можно и с помощью проездного билета-радиокарты или воспользовавшись онлайн сервисами. После посадки в транспорт необходимо следить за остановками, чтобы не пропустить нужную, а после выйти из автобуса. При приближении к остановочному пункту названия остановок обычно объявляются или отображаются на табло.

Маршруты и расписание составляются специалистами по маршрутизации. Они разрабатывают расписание движения автобусов и других видов транспорта, а также контролируют их выполнение. В расписание включается информация о маршруте, времени отправления и прибытия, интервалах между автобусами и другие данные. Кроме того, расписание может изменяться в зависимости от сезона, праздничных и выходных дней.

Процесс составления маршрутов обычно проходит следующим образом. Сначала анализируется спрос на транспорт на основе данных о плотности населения, транспортной инфраструктуре, плотности транспортных потоков, географических особенностях города, затем составляются маршруты. После этого начинается составление расписания движения на каждом маршруте, включая время отправления и прибытия, интервалы между остановками и прибытием автобусов. Специалист по маршрутизации также оценивает, сколько автобусов и водителей необходимо для обслуживания каждого маршрута. Затем готовое расписание утверждается, и маршруты и расписание становятся доступными для пассажиров, которые могут использовать их для планирования своих поездок.

## **1.2 Разработка функциональной модели предметной области**

Рассмотрим информационную систему автопарка, выполненную с помощью средств моделирования нотации IDEF0 (см. рисунок 1.1). Для начала необходимо создать контекстную модель информационной системы. На контекстной диаграмме представлен один функциональный блок «Составить расписание автобусов». Были определены данные входа, механизма, контроля и выхода.

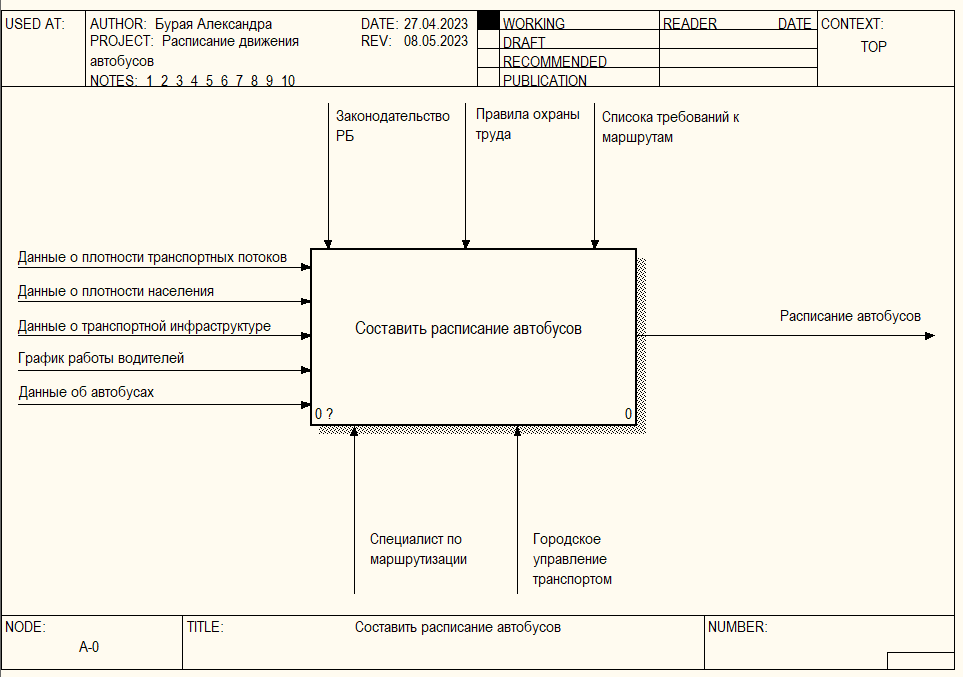


Рисунок 1.1 – Контекстная диаграмма

Контекстная диаграмма содержит следующие параметры:

* входные параметры: данные о плотности транспортных потоков, данные о плотности населения, данные о транспортной инфраструктуре, график работы водителей, данные об автобусах;
* выходной параметр: расписание автобусов;
* управляющие параметры: законодательство РБ, правила охраны труда, список требований к маршрутам;
* исполнительные параметры: специалист по маршрутизации, городское управление транспортом.

Нотация IDEF0 поддерживает последовательную декомпозициюпроцесса до требуемого уровня детализации. Декомпозиция позволяет представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко усваиваемой [2]. Диаграмма декомпозиции процесса «Составить расписание автобусов» представлена на рисунке 1.2.

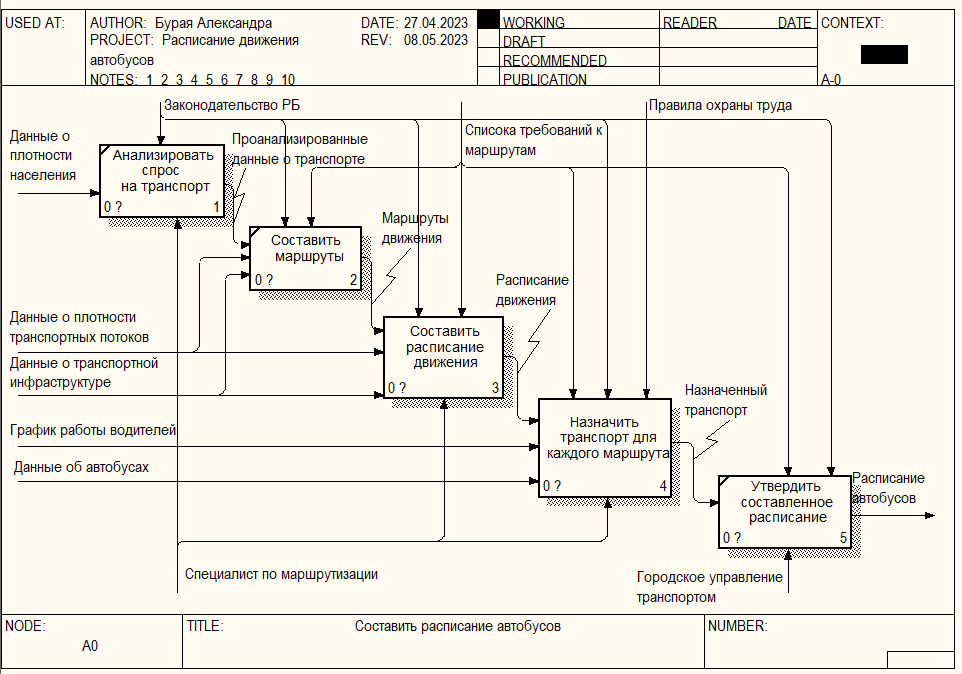


Рисунок 1.2 – Диаграмма декомпозиции

Данная диаграмма состоит из пяти функциональных блоков: «Анализировать спрос на транспорт», «Составить маршруты», «Составить расписание движения», «Назначить транспорт для каждого маршрута», «Утвердить составленное расписание».

Чтобы более детально описать процесс «Составить расписание движения», происходит декомпозиция второго уровня (см. рисунок 1.3).

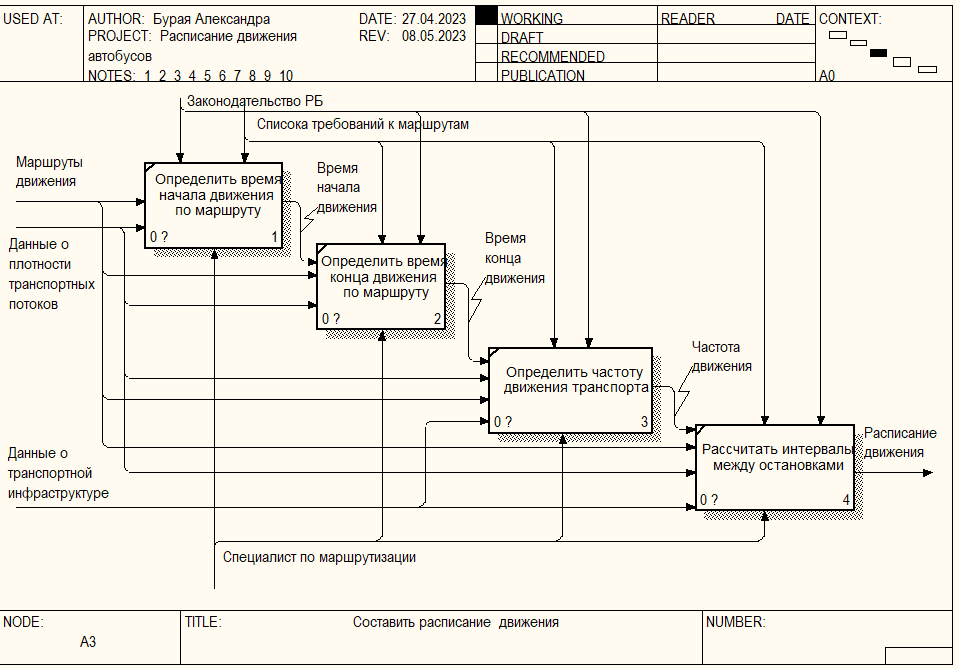


Рисунок 1.3 – Диаграмма декомпозиции процесса «Составить расписание движения»

Данная диаграмма состоит из четырех функциональных блоков: «Определить время начала движения по маршруту», «Определить время конца движения по маршрута», «Определить частоту движения транспорта», «Рассчитать интервалы между остановками».

Для более детального представления процесса «Назначить транспорт для каждого маршрута», происходит декомпозиция второго уровня (см. рисунок 1.4).

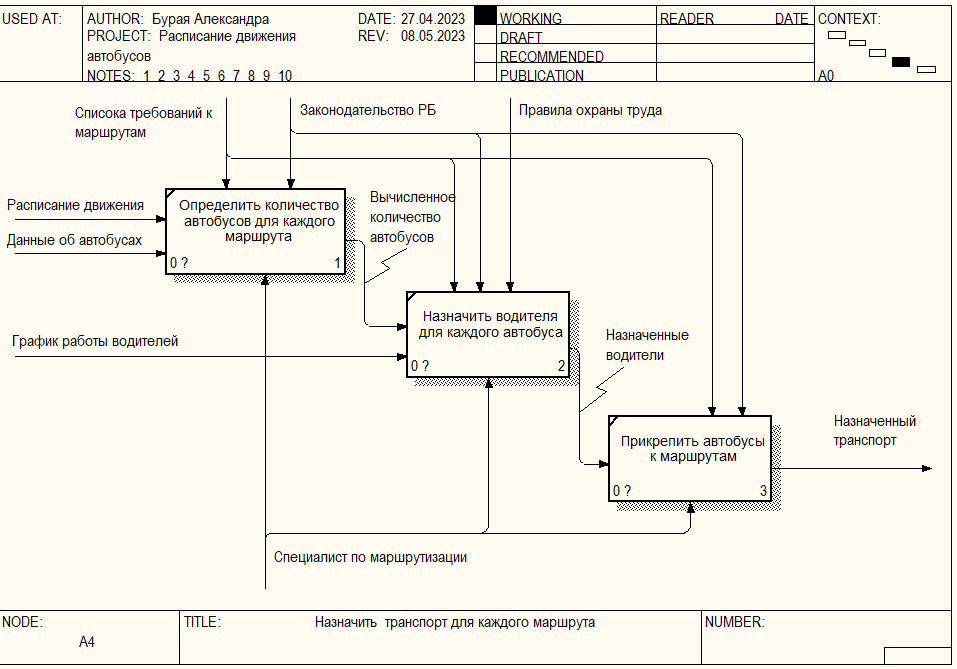


Рисунок 1.4 – Диаграмма декомпозиции процесса «Назначить транспорт для каждого маршрута»

Данная диаграмма состоит из трех функциональных блоков: «Определить количество автобусов для каждого маршрута», «Назначить водителя для каждого автобуса», «Прикрепить автобусы к маршрутам».

## **1.3 Разработка BPMN-модели основного процесса предметной области**

BPMN диаграмма показывает в какой последовательности совершаются рабочие действия и перемещаются потоки информации [3].

В процессе составления расписания автобусов участвует специалист по маршрутизации и городское управление транспортом. На рисунке 1.5 указаны работы, выполняемые каждым участником процесса.

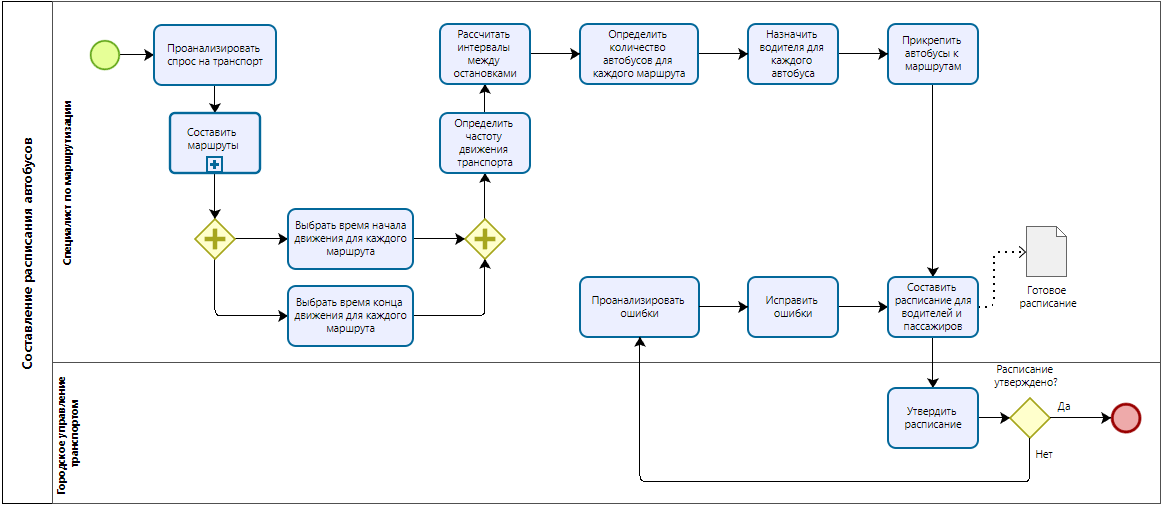


Рисунок 1.5 – BPMN-модель основного процесса предметной области

## **1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований**

Анализ требований к разрабатываемому программному средству является важной частью процесса разработки, поскольку он помогает определить функциональные требования к программному продукту.

Разрабатываемое в ходе курсовой работы программное средство должно соответствовать следующим требованиям:

* ввод данных осуществляется с помощью клавиатуры;
* программа должна иметь простой и понятный интерфейс;
* должно присутствовать разделение на пользователя и администратора с дальнейшим разделением их в правах;
* пользователь должен иметь возможность видеть выполняемые им действия и полученный результат;
* обработка ошибок: приложение должно обрабатывать возможные ошибки, такие как неверный синтаксис выражения, деление на ноль и другие некорректные операции. В случае возникновения ошибки, приложение должно информировать пользователя об этом.

## **1.5 UML-модели представления программного обеспечения и их описание**

UMLявляется графическим языком для визуализации, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. С его помощью разрабатывается детальный план создаваемой системы, содержащей не только ее концептуальные элементы, такие как системные функции и бизнес-процессы, но и конкретные особенности, например классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования.

Диаграмма вариантов использования представляет совокупность прецедентов и актеров. Они описывают систему с точки зрения возможностей ее использования. На рисунке 1.6 представлена диаграмма, на которой изображены два актёра (Пользователь, Администратор), от которых идет разветвление на различные варианты использования [4].

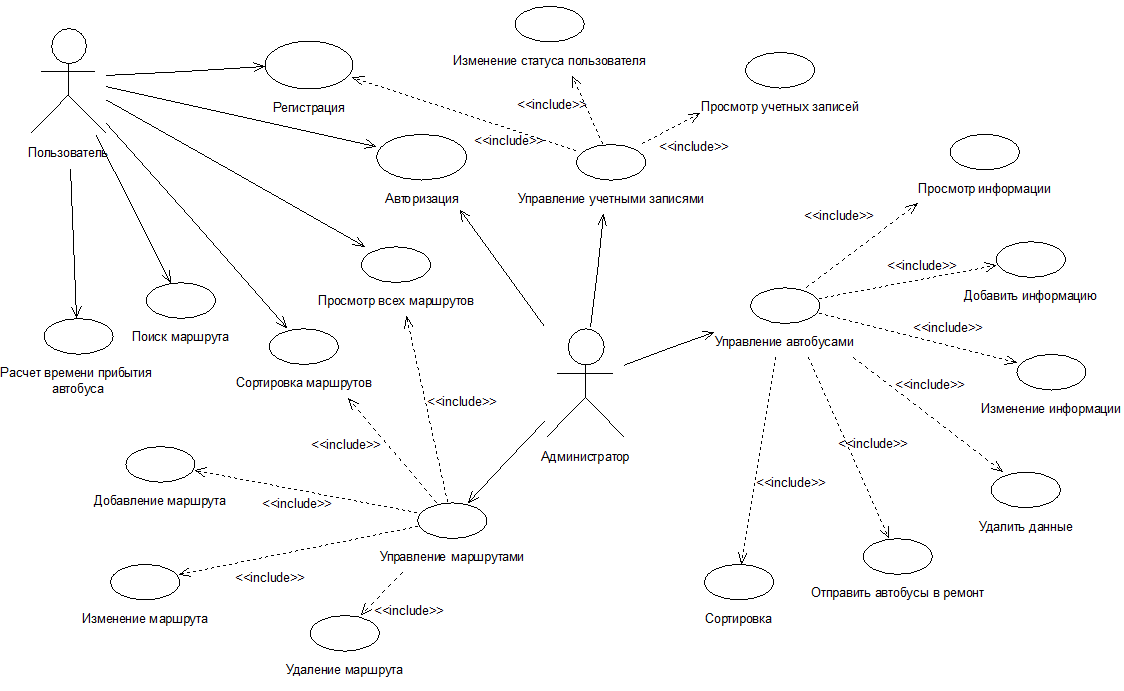


Рисунок 1.6– Диаграмма вариантов использования

Диаграммы компонентов используются для представления организации совокупности компонентов и существующих между ними зависимостей. Данный тип диаграмм описывает систему с точки зрения реализации [4].

На рисунке 1.7 представлена диаграмма компонентов. На которой представлены компоненты, участвующие в процессе использования приложения.

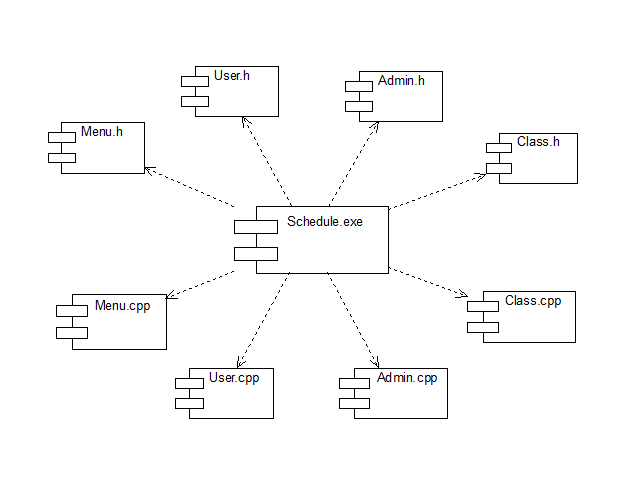


Рисунок 1.7 – Диаграмма компонентов

Диаграммы состояний используются для моделирования поведения интерфейса и классов. Они заостряют внимание на поведении объекта, которое зависит от последовательности событий. На рисунке 1.8 представлена диаграмма состояний, которая демонстрирует процесс входа в систему и ввода запроса с последующей обработкой [4].

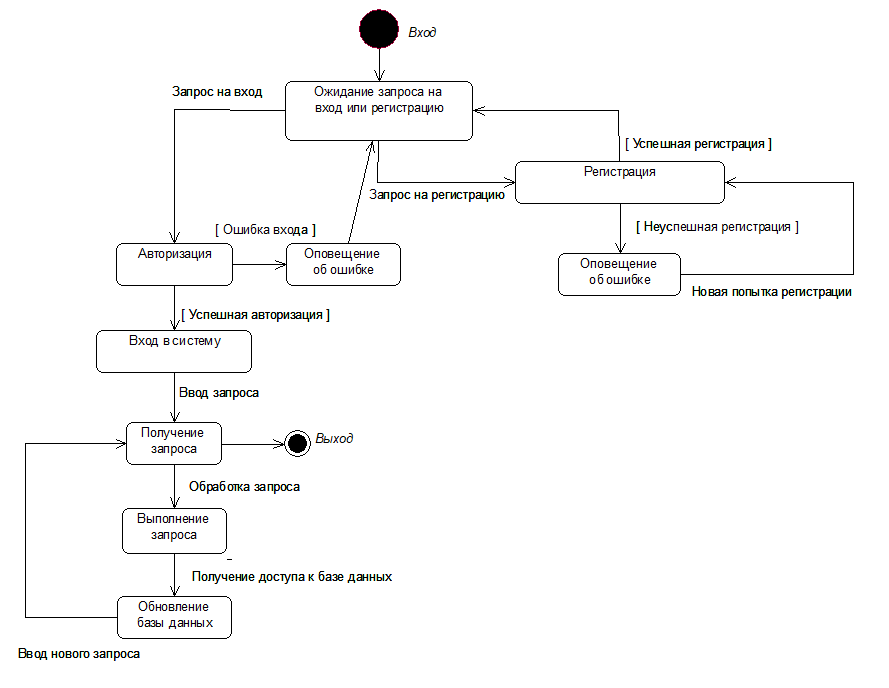


Рисунок 1.8 – Диаграмма состояний

Диаграмма развертывания представляет конфигурацию узлов и размещенных в них компонентов (см. рисунок 1.9)

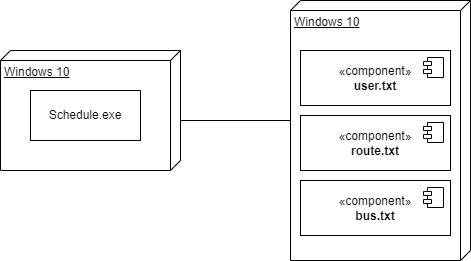


Рисунок 1.9 – Диаграмма развертывания

Диаграммы последовательностей моделируют взаимодействия между объектами в едином сценарии использования. Они иллюстрируют, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования [4].

На рисунке 1.10 представлена диаграмма последовательности. На данной диаграмме изображены сообщения между используемыми классами (остановка, маршрут, автобус, пользователь).

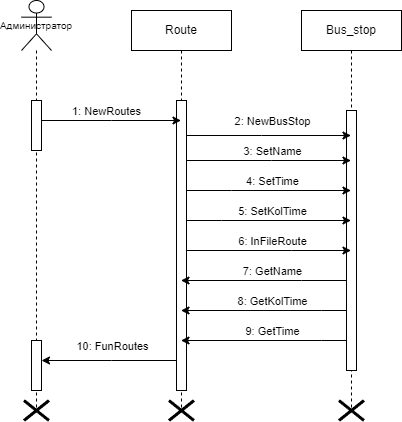


Рисунок 1.10 – Диаграмма последовательности

# **Проектирование и конструирование программного средства**

## **2.1 Постановка задачи**

Разработать программу расписания движения автобусов. В справочной службе автопарка хранится расписание движения автобусов. Для каждого маршрута указаны: номер рейса, названия остановок, на каждой остановке есть свое расписание. Кроме маршрутов в ИС хранится информация об автобусах. О каждом автобусе известны: его регистрационный номер, ФИО водителя, номер маршрута, по которому ходит автобус, требование ремонта.

Индивидуальное задание: вывести информацию о времени прибытия трех ближайших автобусов на определенной остановке на выбранном маршруте (текущее время считывается автоматически).

Общее для всех вариантов задание: реализовать авторизацию для входа в систему, функционал администратора и функционал пользователя.

## **2.2 Разработка модульной структуры**

На рисунке 2.1 представлена схема работы всей программы.



Рисунок 2.1 – Модульная структура программы

## **2.3 Выбор способа организации данных**

В качестве способа организации данных в ходе проектирования были выбраны классы, так как благодаря им можно быстро создать объекты, с которыми должна работать программа. Описание используемых классов представлено ниже.

Таблица 2.3.1 – Описание класса Bus\_stop

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Назначение поля* |
| name | string | Название остановки |
| kol\_time | int | Количество автобусов в день |
| time[300] | double | Расписание |

Таблица 2.3.2 – Описание класса Route

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Назначение поля* |
| number | int | Номер маршрута |
| kol\_ost | int | Количество остановок |
| bus\_stop [10] | class Bus\_Stop | Название остановок и расписание |

Таблица 2.3.3 – Описание класса Bus

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Назначение поля* |
| id | string | Регистрационный номер автобуса |
| FIO | string | ФИО водителя |
| number | int | Номер маршрута |
| repair | bool | Требование ремонта |

Таблица 2.3.4 – Описание класса User

|  |  |  |
| --- | --- | --- |
| *Название поля* | *Тип данных* | *Назначение поля* |
| login | string | Логин |
| password | string | Пароль |
| status | int | Статус |

## **2.4 Разработка перечня пользовательских функций программы**

void StartMenu(); // Меню первого уровня

void UserMenu(); // Меню пользователя

void AdminMenu(); // Меню администратора

void FunctionsUser(); // Функции пользователя

void FunctionsAdmin(); // Функции администратора

void FunRoutes(); // Функции администратора для работы с маршрутами

void FunBuses(); // Функции администратора для работы с автобусами

void FunUsers(); // Функции администратора для работы с учетными записями

void LoginMenu(int status, int x, int y); // Меню для регистрации (принимает 0 для создания пользователя, 1 - для администратора)

void GoToXY(short x, short y); // Курсор в точку x,y

void ConsoleCursorVisible(bool show, short size); // Видимость и размеры курсора

void SetColor(ConsoleColor text, ConsoleColor background); // Установка цвета текста

void SorteRoute(); // Сортировка маршрутов по номеру

void Search(); // Поиск маршрута по номеру

void Time(); // Расчет времени до прибытия ближайшего автобуса

void ChangeStatus(); // Изменение статуса пользователя

void ChangeRoute(); // Редактирование маршрута

void DeleteRoute(); // Удаление маршрута

void ChangeBus(); // Изменение данных об автобусе

void DeleteBus(); // Удаление автобуса

void Repair(); // Отправление автобусов в ремонт

void SorteBus(); // Сортировка автобусов

void Encrypt(string& pass); // Зашифровка

void Decrypt(string& pass); // Расшифровка

int Length\_file(string name); // Получение размера файла для "user.txt" и для "bus.txt"

int Count\_route(string name); // Получение размера файла для "route.txt"

int CinIntErrorCheck(int min, int max); // Проверка целых чисел в заданном диапазоне

double CinDoubleErrorCheck(double min, double max); // Проверка дробных чисел в заданном диапазоне

int CinIntErrorCheck(); // Проверка целых чисел

bool CinBoolErrorCheck(); // Проверка булевых значений

string CinStringSymAngErrorCheck(int &x, int &y); // Проверка строк (принимается только латиница)

string CinStringSymRusErrorCheck(); // Проверка строк (принимается только кириллица)

string CinNomerErrorCheck(); // Проверка номеров автобусов

void OutFileRoute(Route\* (&arr\_route), int kol\_route); // Получение данных из файла "route.txt"

void OutFileBus(Bus\* (&arr\_buses)); // Получение данных из файла "bus.txt"

void OutFileUser(User\* (&arr\_users)); // Получение данных из файла "user.txt"

void NewBusStop(); // Создание новых остановок

friend void NewRoutes(); // Создание нового маршрута

void OutputRoute(); // Вывод всех маршрутов

void InFileRoute(); // Запись данных в файл "route.txt"

friend void OutFileRoute(Route\* (&arr\_route), int kol\_route); // Получение данных из файла "route.txt"

void Copy(Route &arr\_route); // Копирование полей одного маршрута в другой

void SearchTime(int j); // Расчет времени до прибытия ближайшего автобуса

void InFileBus(); // Запись данных в файл "bus.txt"

friend void OutFileBus(Bus\* (&arr\_buses)); // Получение данных из файла "bus.txt"

void NewBus(); // Создание нового автобуса

void OutputBus(); // Вывод всех автобусов

void CopyBus(Bus& arr); // Копирование полей одного автобуса в другой

void InFileUser(); // Запись данных в файл "user.txt"

friend void OutFileUser(User\* (&arr\_users)); // Получение данных из файла "user.txt"

void OutputUser(); // Вывод всех учетных записей пользователей

## **Разработка схем алгоритмов программы**

Реализация функции StartMenu():

void StartMenu()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

SetConsoleTitle(L"Расписание маршрутов автобусов города ");

ConsoleCursorVisible(false, 100);

int y=10, x=35;

SetColor(Yellow, Black);

GoToXY(x, y); y++;

cout << " ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶ ¶¶ ¶¶¶ ¶¶¶ ¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶ ¶¶ ¶¶¶ ¶¶¶ ¶¶ "; GoToXY(x, y++);

cout << " ¶¶ ¶¶ ¶¶¶ ¶¶¶ ¶¶ "; GoToXY(x, y++);

cout << " ¶¶ ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶¶¶ ¶¶¶¶¶¶¶¶¶¶¶¶ ¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶ ¶¶¶¶¶ ¶¶¶¶¶¶¶¶¶¶ ¶¶¶¶¶ ¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶0¶¶¶ ¶¶¶0¶¶¶ "; GoToXY(x, y); y+=2;

cout << " ¶¶¶¶¶ ¶¶¶¶¶ "; GoToXY(x, y);

cout << " Загрузка. "; GoToXY(x, y);

Sleep(500);

cout << " Загрузка.. "; GoToXY(x, y); y += 2;

Sleep(500);

cout << " Загрузка... "; GoToXY(x, y);

Sleep(1000);

cout << " Нажмите любую клавишу для продолжения "; GoToXY(x, y);

\_getch();

system("CLS");

string m[] = { "Войти как пользователь", "Войти как администратор", "Регистрация", "Выход" };

int active\_menu = 0; // Для подсветки пункта меню

char ch; // Для кода нажатой клавиши

while (true){

x = 45, y = 12;

GoToXY(x, y);

for (int i = 0; i < size(m); i++)

{

if (i == active\_menu) SetColor(LightMagenta,Black);

else SetColor(Yellow, Black);

GoToXY(x, y++);

cout << m[i] << endl;}

ch = \_getch();

if (ch == -32) ch = \_getch();

switch (ch){

case ESC:

system("CLS");

exit(0);

case UP:

if (active\_menu > 0)

active\_menu--;

else

active\_menu = size(m) - 1;

break;

case DOWN:

if (active\_menu < size(m) - 1)

active\_menu++;

else

active\_menu = 0;

break;

case ENTER:

switch (active\_menu){

case 0:

system("CLS");

GoToXY(45,12);

SetColor(Yellow, Black);

UserMenu();

break;

case 1:

system("CLS");

GoToXY(45, 12);

SetColor(Yellow, Black);

AdminMenu();

break;

case 2:

system("CLS");

SetColor(Yellow, Black);

LoginMenu(0, 45, 12);

\_getch();

system("CLS");

break;

case 3:

system("CLS");

GoToXY(45,13);

SetColor(Yellow, Black);

cout << "Выход из программы...";

\_getch();

system("CLS");

GoToXY(0,0);

exit(0);

default:

cout << "Используйте стрелки" << endl;

\_getch();

system("CLS");

}

break;

default:

GoToXY(45, 17);

cout << "Используйте стрелки" << endl;

\_getch();

system("CLS");}

}

}

На рисунке 2.2 представлена схема алгоритма начального меню.

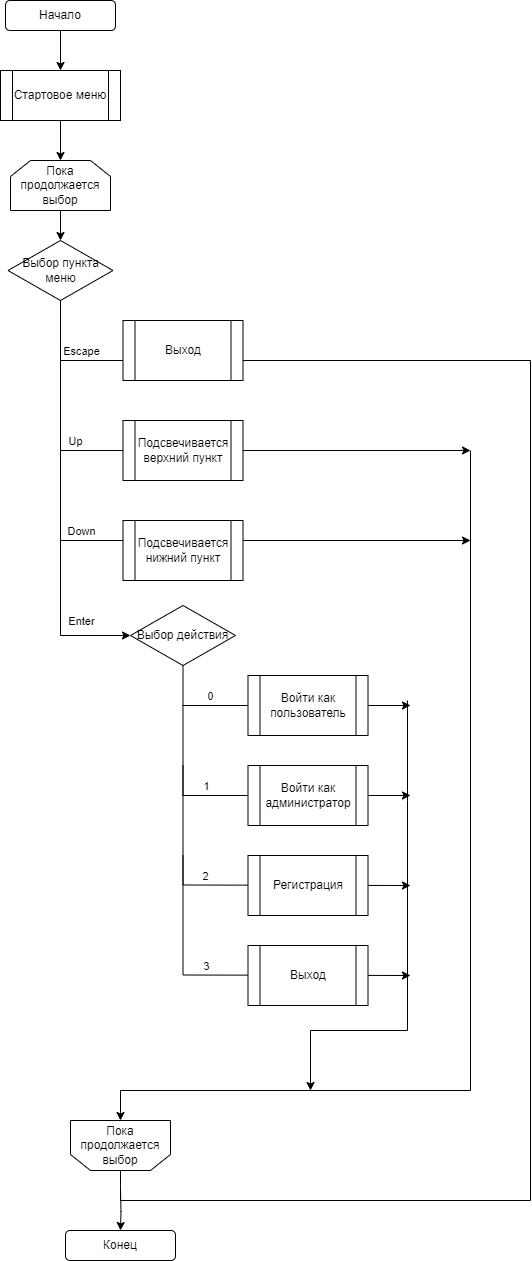


Рисунок 2.2 – Алгоритм функции StartMenu

Индивидуальное задание: вывести информацию о времени прибытия трех ближайших автобусов на определенной остановке на выбранном маршруте (текущее время считывается автоматически). На рисунке 2.3 представлена схема алгоритма функции Time, для выполнения этого задания.

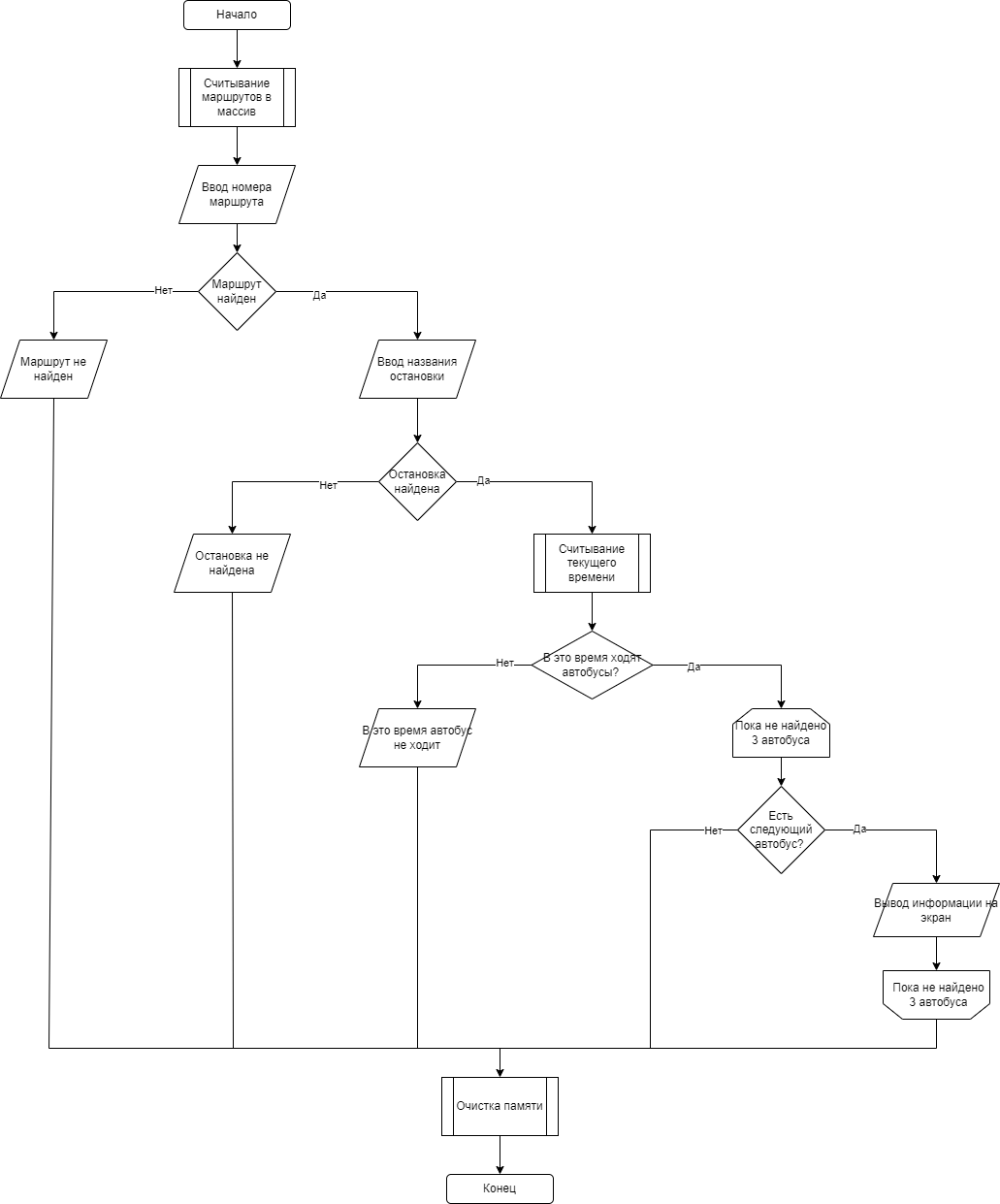


Рисунок 2.3 – Алгоритм функции Time

Далее приведена реализация функции Time();

void Time()

{

bool n = 0, o = 0;

int size = Count\_route("route.txt"), nomer, kol;

string name;

Route\* arr = new Route[size];

OutFileRoute(arr, size);

cout << "Введите номер искомого маршрута: " << endl;

nomer=CinIntErrorCheck(1,999);

for (int i = 0; i < size; i++)

{

if (arr[i].GetNumber() == nomer)

{

n = 1;

cout << "Введите остановку искомого маршрута: " << endl;

name = CinStringSymRusErrorCheck();

kol = arr[i].GetKolost();

for (int j = 0; j < kol; j++)

{

if (arr[i].GetBusStopName(j) == name)

{

o = 1;

arr[i].SearchTime(j);

}

}

}

}

if (n == 0)

{

delete[]arr;

cout << "Маршрута с таким номером не найдено" << endl;

return;

}

else if (o == 0)

{

delete[]arr;

cout << "Маршрута с такой остановкой не найдено" << endl;

return;

}

delete[]arr;}

void Route::SearchTime(int j)

{

double timenow;

int kol = bus\_stop[j].GetKolTime();

time\_t now = time(0);

tm\* ltm = localtime(&now);

timenow = ltm->tm\_hour + ltm->tm\_min / 100.;

cout << "Текущее время: " << fixed << setprecision(2) << timenow<<endl;

if (timenow > bus\_stop[j].GetTime(kol - 1) || timenow < bus\_stop[j].GetTime(0))

{

cout << "В это время данный автобус не ходит" << endl;

return;

}

for (int i = 0; i < kol; i++)

{

if (bus\_stop[j].GetTime(i) > timenow)

{

cout << "Ближайший автобус приедет в " << fixed << setprecision(2)<< bus\_stop[j].GetTime(i);

if (i+1<kol)

{

cout << ", " << bus\_stop[j].GetTime(i + 1);

if(i + 2 < kol)

{

cout << ", " << bus\_stop[j].GetTime(i + 2) << endl;

}

}

cout.unsetf(ios::fixed);

return;

}

}

}

# **3 Тестирование и проверка работоспособности программного средства**

При написании кода программы необходимо предусмотреть обработку исключительных ситуаций, таких как:

* введенные пользователем данные не соответствуют формату поля (например, символы в числовом поле);
* введенные пользователем данные нелогичны (например, год рождения больше 2023);
* ничего не найдено по результатам поиска;
* проверки на уникальность введенных данных (если необходимо).

Выбор в меню осуществляется только при помощи стрелок вверх-вниз. В противном случае появляется сообщение о необходимости использовать стрелки (см. рисунок 3.1).

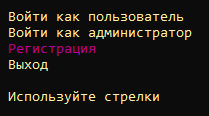


Рисунок 3.1 – Проверка выбора пункта меню

Проверка на уникальность логина при регистрации (см. рисунок 3.2). Если учетная запись с таким именем уже существует, то появляется сообщение об ошибке.

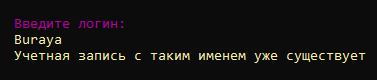


Рисунок 3.2 – Проверка ввода логина при регистрации

Проверка пароля при входе и регистрации (см. рисунок 3.3). При входе в роли администратора и регистрации у пользователя 3 попытки ввода пароля (подтверждения пароля), при входе в роли обычного пользователя – 5 попыток ввода.

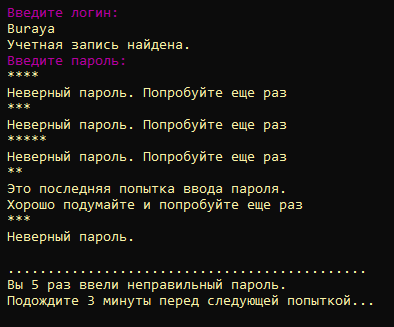


Рисунок 3.3 – Подтверждение пароля

Проверка на ввод нечисловых данных (принимаются только целые значения >0) представлена на рисунке 3.4.

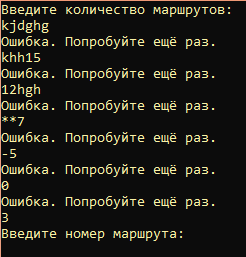


Рисунок 3.4 – Проверка на ввод целых чисел

Проверка на ввод строк (принимаются только русские буквы) представлена на рисунке 3.5.

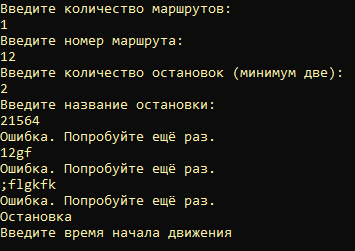


Рисунок 3.5 – Проверка на ввод строк

Ввод некорректного времени для начала движения (см. рисунок 3.6). На первой остановке время начала движения должно быть меньше, чем на второй. Также присутствует проверка на ввод корректного времени.

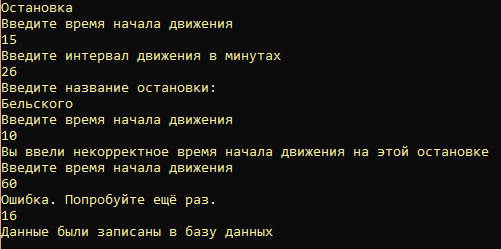


Рисунок 3.6 – Проверка на ввод времени

Проверка на существования маршрута (см. рисунок 3.7). Если маршрута с таким номером не существует, то появляется сообщение об ошибке.



Рисунок 3.7 – Проверка на существование маршрута

Проверка ввода регистрационного номера автобуса (должен состоять из 6 символов: первые два – это заглавные латинские буквы A, B, E, I, K, M. H, O, P, C, T, X, остальные 4 – цифры) представлена на рисунке 3.8.

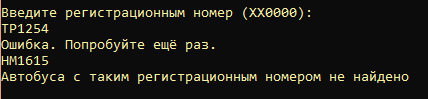


Рисунок 3.8 – Проверка на ввод регистрационного номера

Проверка на уникальность вводимого имени водителя (см. рисунок 3.9). Если водитель с таким ФИО уже есть, то появляется сообщение об ошибке.

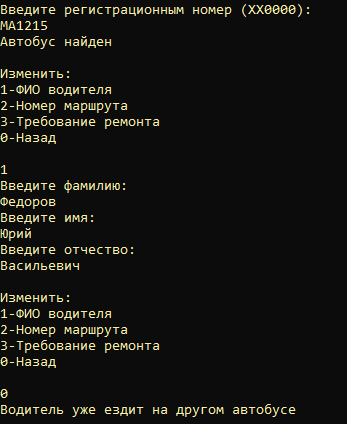


Рисунок 3.9 – Проверка на уникальность ФИО

# **4 Инструкция по развертыванию приложения и сквозной тестовый пример**

## **4.1 Авторизация**

В данной части работы представлен пример работы программы при авторизации и регистрации пользователя. Загрузка программы представлена на рисунке 4.1.

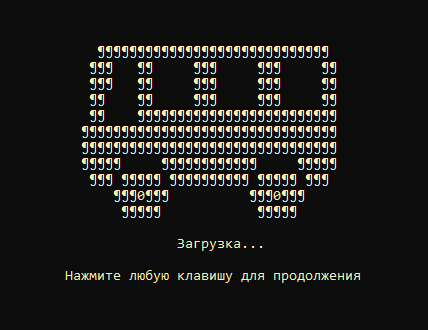


Рисунок 4.1 – Загрузочный экран

Далее появляется меню, с помощью которого производится вход в систему. При выборе варианта «Войти как пользователь» осуществляется вход от имени пользователя, при выборе варианта «Войти как администратор» – от имени администратора, если выбрать поле «Регистрация», начнется создание учетной записи пользователя. При выборе «Выхода» программа завершает свою работу (см. рисунок 4.2).

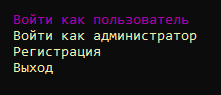


Рисунок 4.2 – Меню авторизации

Непосредственный вход в аккаунт представлен на рисунке 4.3. В целях безопасности пароль замаскирован с помощью символа «\*».

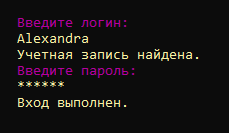


Рисунок 4.3 –Авторизация

## **4.2 Модуль пользователя**

При входе в систему под именем пользователя, появляется главное меню пользователя, которое включает функции, представленные на рисунке 4.4.

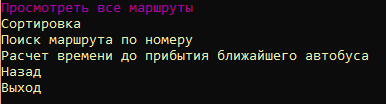


Рисунок 4.4 –Меню пользователя

Функция просмотра всех маршрутов, отображает на экран все существующие маршруты (см. рисунок 4.5)

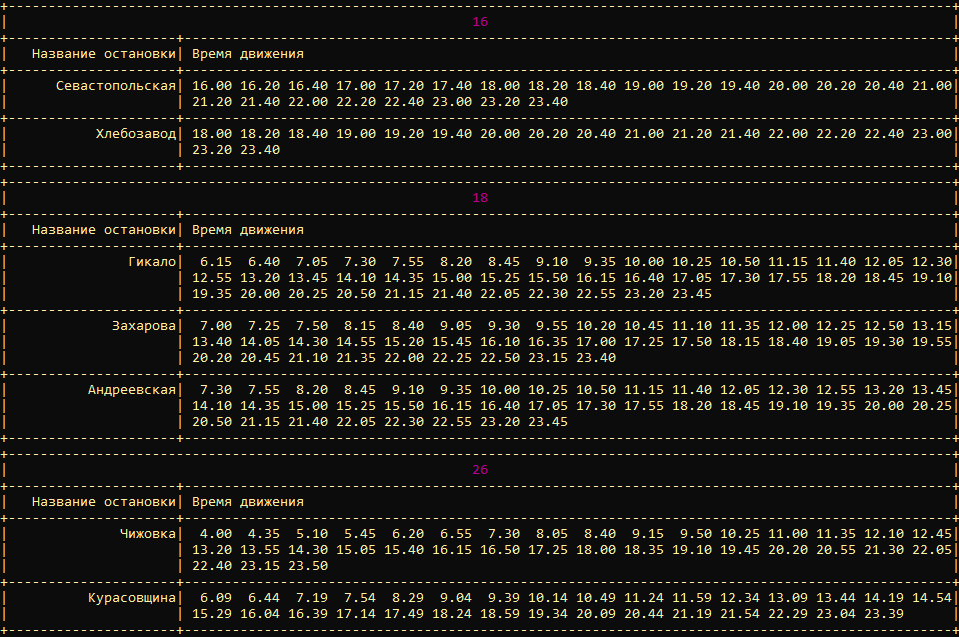


Рисунок 4.5 – Просмотр маршрутов

Функция выбора определенного маршрута, отображает на экран только один маршрут и все его остановки (см. рисунок 4.6-4.7)

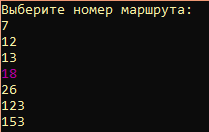


Рисунок 4.6 – Выбор номера

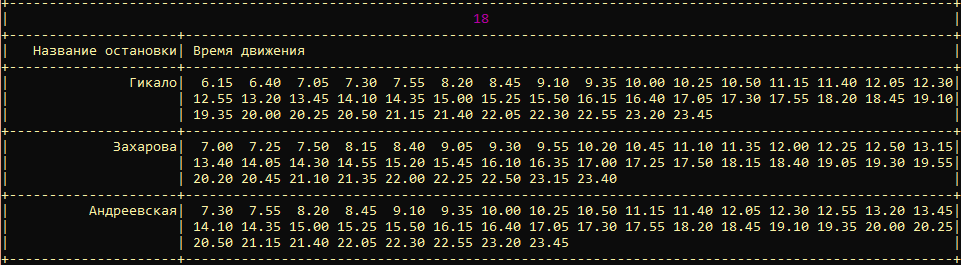


Рисунок 4.7 – Просмотр маршрута

Функция поиска для расчета времени прибытия ближайшего автобуса считывает текущее время и выводит на экран 3 автобуса, которые скоро приедут. Если автобус не ходит в текущее время, выводится информация об этом. (см. рисунок 4.8).

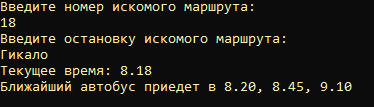


Рисунок 4.8 – Расчет времени прибытия

## **4.3 Модуль администратора**

При входе в систему под именем администратора, появляется главное меню администратора, которое включает функции, представленные на рисунке 4.9.

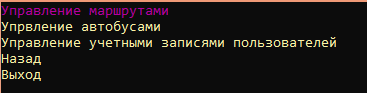


Рисунок 4.9 – Главное меню администратора

При выборе пункта «Управление маршрутами», появляется меню для работы с маршрутами, которое включает функции, представленные на рисунке 4.10.

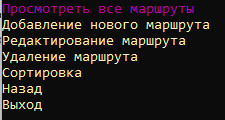


Рисунок 4.10 – Меню управления маршрутами

Функция добавления нового маршрута позволяет создать новый маршрут (см. рисунок 4.11).

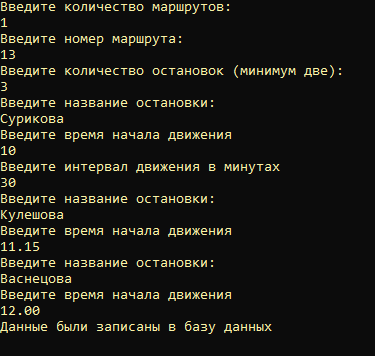


Рисунок 4.11 – Добавление маршрута

Функция сортировки маршрутов сортирует их по возрастанию порядкового номера (см. рисунок 4.12).

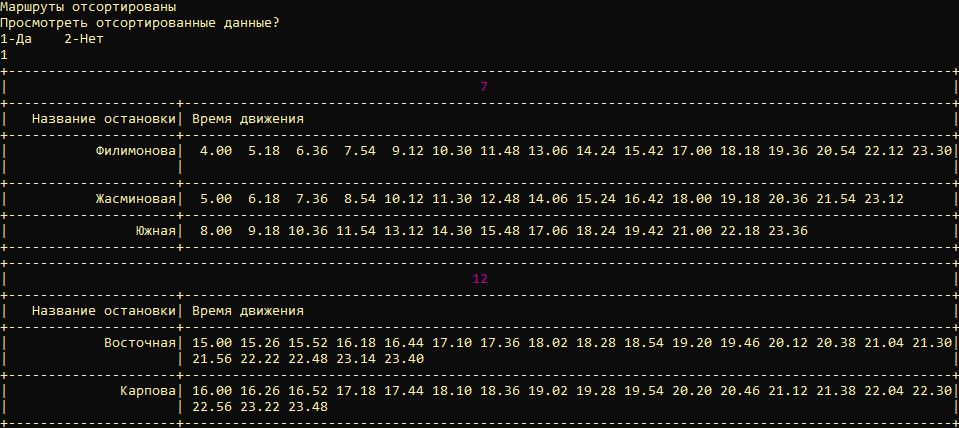


Рисунок 4.12 – Сортировка маршрутов

При выборе пункта «Управление автобусами», появляется меню для работы с автобусами, которое включает функции, представленные на рисунке 4.13.

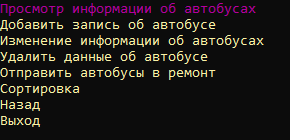


Рисунок 4.13 – Меню управления автобусами

Функция просмотра всех данных об автобусах, отображает на экран всю соответствующую информацию о транспортных средствах (см. рисунок 4.14). При несоответствии маршрута, по которому ходит автобус, и информации об актуальных маршрутах из базы данных (если маршрута уже не существует), выводится «-».

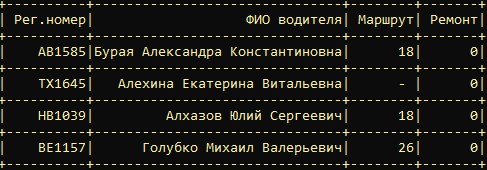


Рисунок 4.14 – Просмотр информации об автобусах

Функция изменения данных автобуса позволяет выбрать одно из трех полей и изменить соответствующую информацию о транспортном средстве. (см. рисунок 4.15)

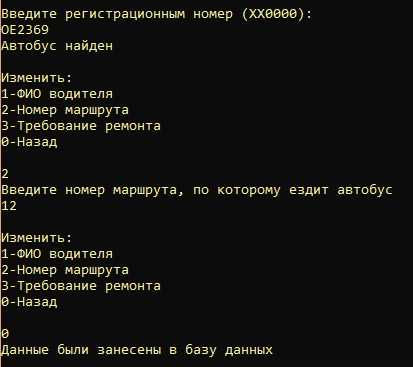


Рисунок 4.15 – Изменение данных автобуса

Функция сортировки автобусов сортирует их по регистрационному номеру, по номеру маршрута или ФИО водителя в зависимости от выбранного пункта (см. рисунок 4.16).

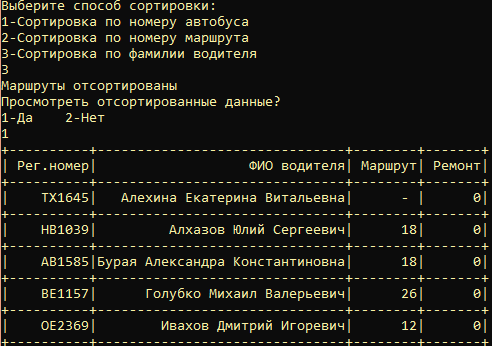


Рисунок 4.16 – Сортировка автобусов

При выборе пункта «Управление учетными записями пользователей», появляется меню для работы с учетными записями, которое включает функции, представленные на рисунке 4.17.

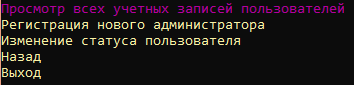


Рисунок 4.17 – Меню управления учетными записями

Функция просмотра всех учетных записей, отображает на экран всю соответствующую информацию о них. Статус 0 – статус пользователя, статус 1 – статус администратора (см. рисунок 4.18).

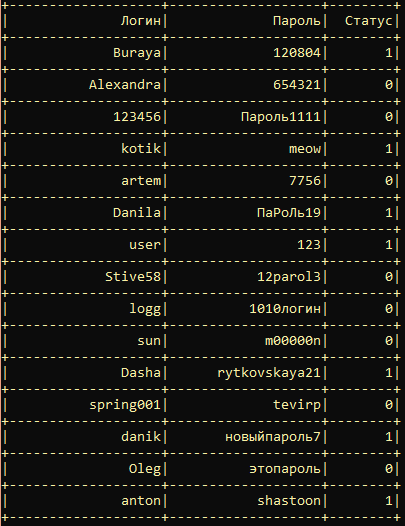


Рисунок 4.18 – Меню управления учетными записями

Функция регистрации нового администратора (см. рисунок 4.19)

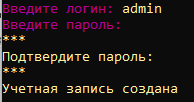


Рисунок 4.19 – Регистрация администратора

Функция изменения статуса пользователя позволяет из обычного пользователя делать администратора, а из администратора – обычного пользователя (см. рисунок 4.20).

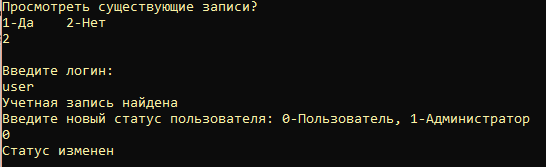


Рисунок 4.20 – Изменение статуса пользователя

# **Заключение**

При выполнении курсовой работы был изучен процесс составления расписания автобусов. Было установлено, что правильная организация и хранение данных позволяет уменьшить трудоемкость выполнения этой операции вручную.

В ходе объектного моделирования системы был построен ряд UML-диаграмм; разработаны модели бизнес-процессов предметной области на основе нотаций IDEF0 и BPMN; описаны основные алгоритмы работы программного средства; разработано руководство пользователя; выполнено тестирование программного средства, показавшее его соответствие функциональным требованиям, поставленным в задании на разработку.

В ходе работы были созданы пользовательские функции приложений: функции поиска, сортировки, получения данных из файла, авторизации, редактирования, проверки данных и другие. Были созданы следующие алгоритмы функций: стартового меню, индивидуального задания. Были рассмотрены исключительные ситуации и их обработка с помощью функций проверок.

Цель курсовой работы (уменьшение временных затрат на поиск расписания автобусов) была успешно достигнута.

Итог данной курсовой работы – консольное приложение, которое помогает создавать, редактировать, просматривать и удалять маршруты и расписание автобусов. В этом приложении присутствуют конструкции объектно-ориентированного программирования, стандартные и пользовательские функции.

Также стоит отметить, что с данной программой может комфортно работать как обычный пассажир с правами пользователя, так и работник автопарка с правами администратора. Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при учете и анализе данных общественного транспорта.

# **Список использованных источников**

1. Роль общественного транспорта в развитии современных городов - Статья [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/rol-obschestvennogo-transporta-v-razvitii-sovremennyh-gorodov/viewer>.
2. Методология IDEF0 [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №2.docx](file:///C:\\Users\\Анюта\\трпо\\ТРПО%20Лабораторная%20работа%20№2.docx).
3. Методическое пособие для выполнение лабораторной работы «Моделирование бизнес-процессов на основе нотации BPMN в BizagiProcess Modeler» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №4.docx](file:///C:\\Users\\Анюта\\трпо\\ТРПО%20Лабораторная%20работа%20№4%20(2).docx).
4. Методическое пособие для выполнения лабораторной роботы «Язык Uml. Диаграммы UML» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО - Лабораторные работы №7-8.doc](file:///C:\\Users\\Анюта\\трпо\\ТРПО%20-%20Лабораторные%20работы%20№7-8.doc)x.

# **Приложение А**

**Листинг кода**

// Стартовое меню

void StartMenu()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

SetConsoleTitle(L"Расписание маршрутов автобусов города ");

ConsoleCursorVisible(false, 100);

int y=10, x=35;

SetColor(Yellow, Black);

GoToXY(x, y); y++;

cout << " ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶ ¶¶ ¶¶¶ ¶¶¶ ¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶ ¶¶ ¶¶¶ ¶¶¶ ¶¶ "; GoToXY(x, y++);

cout << " ¶¶ ¶¶ ¶¶¶ ¶¶¶ ¶¶ "; GoToXY(x, y++);

cout << " ¶¶ ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶¶¶ ¶¶¶¶¶¶¶¶¶¶¶¶ ¶¶¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶ ¶¶¶¶¶ ¶¶¶¶¶¶¶¶¶¶ ¶¶¶¶¶ ¶¶¶ "; GoToXY(x, y++);

cout << " ¶¶¶0¶¶¶ ¶¶¶0¶¶¶ "; GoToXY(x, y); y+=2;

cout << " ¶¶¶¶¶ ¶¶¶¶¶ "; GoToXY(x, y);

cout << " Загрузка. "; GoToXY(x, y);

Sleep(500);

cout << " Загрузка.. "; GoToXY(x, y); y += 2;

Sleep(500);

cout << " Загрузка... "; GoToXY(x, y);

Sleep(1000);

cout << " Нажмите любую клавишу для продолжения "; GoToXY(x, y);

\_getch();

system("CLS");

string m[] = { "Войти как пользователь", "Войти как администратор", "Регистрация", "Выход" };

int active\_menu = 0; // Для подсветки пункта меню

char ch; // Для кода нажатой клавиши

while (true){

x = 45, y = 12;

GoToXY(x, y);

for (int i = 0; i < size(m); i++){

if (i == active\_menu) SetColor(LightMagenta,Black);

else SetColor(Yellow, Black);

GoToXY(x, y++);

cout << m[i] << endl;}

ch = \_getch();

if (ch == -32) ch = \_getch();

switch (ch){

case ESC:

system("CLS");

exit(0);

case UP:

if (active\_menu > 0)

active\_menu--;

else

active\_menu = size(m) - 1;

break;

case DOWN:

if (active\_menu < size(m) - 1)

active\_menu++;

else

active\_menu = 0;

break;

case ENTER:

switch (active\_menu){

case 0:

system("CLS");

GoToXY(45,12);

SetColor(Yellow, Black);

UserMenu();

break;

case 1:

system("CLS");

GoToXY(45, 12);

SetColor(Yellow, Black);

AdminMenu();

break;

case 2:

system("CLS");

SetColor(Yellow, Black);

LoginMenu(0, 45, 12);

\_getch();

system("CLS");

break;

case 3:

system("CLS");

GoToXY(45,13);

SetColor(Yellow, Black);

cout << "Выход из программы...";

\_getch();

system("CLS");

GoToXY(0,0);

exit(0);

default:

cout << "Используйте стрелки" << endl;

\_getch();

system("CLS");}

break;

default:

GoToXY(45, 17);

cout << "Используйте стрелки" << endl;

\_getch();

system("CLS");}

}

}

// Меню регистрации

void LoginMenu(int status, int x, int y)

{

User temp, temp\_pass;

string log;

bool st=1, p=0;

int k=0;

temp.SetStatus(status);

GoToXY(x, y++);

SetColor(LightMagenta, Black);

cout << "Введите логин: ";

SetColor(Yellow, Black);

rewind(stdin);

log = CinStringSymAngErrorCheck(x,y);

int size = Length\_file("user.txt");

User\* arr\_users = new User[size];

OutFileUser(arr\_users);

for (int i = 0; i < size; i++){

if (arr\_users[i].GetLogin() == log)

st = 0;}

if (st==1){

st = 1;

temp.SetLogin(log);

GoToXY(x, y++);

SetColor(LightMagenta, Black);

cout << "Введите пароль: ";

SetColor(Yellow, Black);

GoToXY(x, y++);

temp.Password();

while (temp.GetPassword() == "NOPASSWORD"){

cout << "Вы не ввели пароль, попробуйте еще раз" << endl;

GoToXY(x, y++);

temp.Password();}

GoToXY(x, y++);

cout << "Подтвердите пароль: ";

while (k != 3){

GoToXY(x, y++);

temp\_pass.Password();

while (temp\_pass.GetPassword() == "NOPASSWORD"){

cout << "Вы не ввели пароль, попробуйте еще раз" << endl;

GoToXY(x, y++);

temp\_pass.Password();}

if (temp.GetPassword() == temp\_pass.GetPassword()){

delete[]arr\_users;

p = 1;

GoToXY(x, y++);

cout << "Учетная запись создана";

GoToXY(x, y++);

temp.InFileUser();

break;

}

else if (k == 1){

GoToXY(x, y++);

cout << "Это последняя попытка ввода пароля.";

k++;}

else if (k == 2){

GoToXY(x, y++);

cout << "Неверный пароль.";

k++;}

else{

GoToXY(x, y++);

cout << "Неверный пароль. Попробуйте еще раз";

k++;}

}

}

else{

delete[]arr\_users;

GoToXY(x, y++);

cout << "Учетная запись с таким именем уже существует";

p = 1;}

if (p == 0){

delete[]arr\_users;

y++;

GoToXY(x, y++);

cout << ".............................................";

GoToXY(x, y++);

cout << "Вы 3 раза ввели неправильный пароль.";

GoToXY(x, y++);

cout << "Начните регистрацию заново.";

\_getch;}

}

# **Приложение В**

**Отчет о проверке на заимствование в системе «Антиплагиат»**

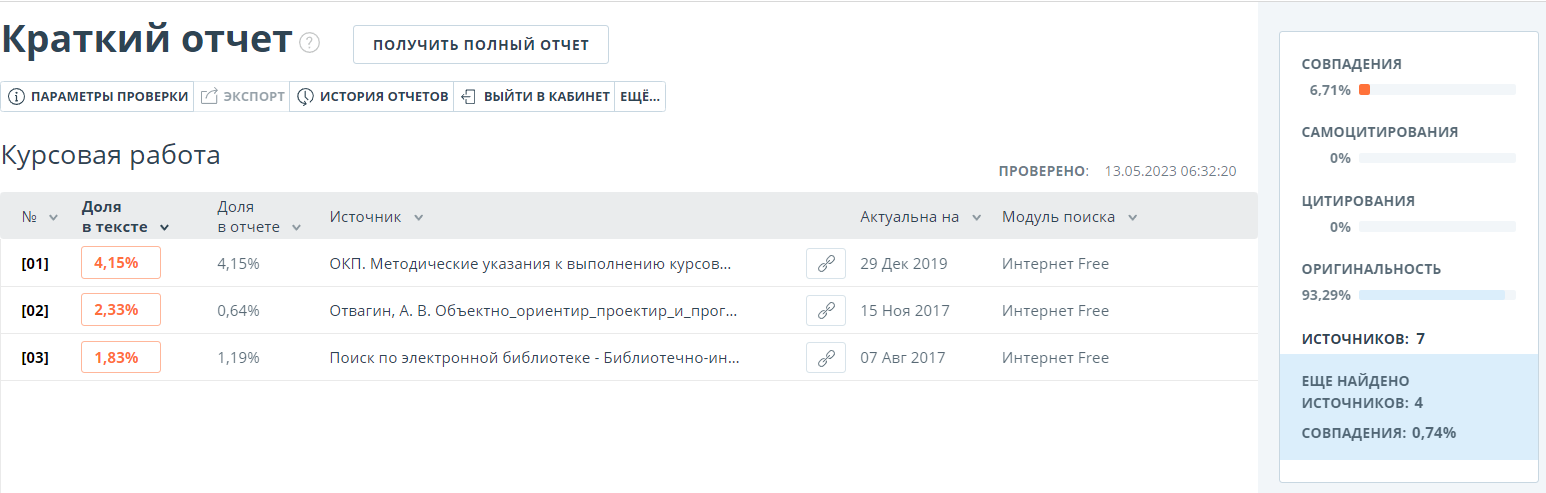


Рисунок Б.1 – Отчет о проверке на заимствование