СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Павловская, Т. А. *C*/*C*++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2010. —461 с.

[2] Лафоре, Р. Объектно-ориентированное программирование в C++/ Р. Лафоре – СПб: Питер Ком, 2019. – 923 с.

[3] Вендров, А.М. Практикум по проектированию программного обеспечения экономических информационных систем : учебное пособие / А.М. Вендров. – Москва : Финансы и статистика, 2004. – 192 с. : ил. – (*UML* CASE).

[4] Шилдт, Г. Самоучитель *C*++: Пер. с англ. – 3-е изд. – СПб.: БХВ-Петербург, 2003. — 688 с.

[5] Страуструп, Б.Б Язык программирования *C*++/ Б. Страуструп – М.: Бином, 2022. – 369 с.

[6] Вайсфельд, М. Объектно-ориентированное мышление. – СПб: Питер, 2014. – 27с.

[7] СТП 01-2017. Стандарт предприятия. Дипломные проекты (работы). Общие требования. - Минск: БГУИР, 2017. - 169 с.

[8] *Drawio* [Электронный ресурс]. – Режим доступа: *https://app.diagrams.net/*

[9] БЧ. Мой поезд [Электронный ресурс]. – Режим доступа: *https://pass.rw.by/ru/*

ПРИЛОЖЕНИЕ А
(обязательное)
Фрагмент исходного кода приложения

```
#include "../Admin.h"

#include "../../Utils.h"


Admin::Admin(std::string name, int userId) {
    this->name = name;
    this->userId = userId;
}

void Admin::start() {
    std::cout << "Hi, " << name << "!" << std::endl;

    if (!db.openDatabase(dbName)) {
        std::cerr << "Failed to open the database. Terminating." << std::endl;
        std::exit(EXIT_FAILURE);
    }

    clearScreen();

    while (isWork) {
        std::cout << "============================" <<
std::endl;
        std::cout << "        MAIN MENU        " << std::endl;
        std::cout << "============================" <<
std::endl;
        std::cout << "1. Trains info" << std::endl;
        std::cout << "2. Carriages info" << std::endl;
        std::cout << "3. Stations info" << std::endl;
        std::cout << "4. Places info" << std::endl;
        std::cout << "5. Routes info" << std::endl;
        std::cout << "6. Tickets info" << std::endl;
        std::cout << "-------------------------" << std::endl;
        std::cout << "0. Exit" << std::endl;
        std::cout << "============================" <<
std::endl;
        std::cout << "Enter your choice: ";

        int command;
        std::cin >> command;
        handleInvalidInput();

        switch (command) {
          case 1:
            clearScreen();
            std::cout << "\n===== TRAINS INFO =====\n" << std::endl;
            printTrainsMenu();
            break;

          case 2:
            clearScreen();
            std::cout << "\n===== CARRIAGES INFO =====\n" <<
std::endl;
            printCarriagesMenu();
            break;

          case 3:
            clearScreen();
            std::cout << "\n===== STATIONS INFO =====\n" <<
std::endl;
            printStationsMenu();
            break;

          case 4:
            clearScreen();
            std::cout << "\n===== PLACES INFO =====\n" << std::endl;
            printPlacesMenu();
            break;

          case 5:
            clearScreen();
            std::cout << "\n===== ROUTES INFO =====\n" << std::endl;
```

```
                printRoutesMenu();
                break;

            case 6:
                clearScreen();
                std::cout << "\n===== TICKETS INFO =====\n" << std::endl;
                printTicketsMenu();
                break;

            case 0:
                clearScreen();
                std::cout << "\nExiting... Goodbye!\n" << std::endl;
                exit();
                break;

            default:
                clearScreen();
                std::cout << "\nInvalid choice. Please try again.\n" << std::endl;

                pressToContinue();
                break;
        }
    }
}

void Admin::createStation() {
    clearScreen();
    std::string name, city;

    std::cout << "===============================" <<
std::endl;
    std::cout << "       CREATE STATION       " << std::endl;
    std::cout << "===============================" <<
std::endl;

    std::cout << "Enter the station name: ";
    std::cin.ignore();
    std::getline(std::cin, name);
    std::cout << "Enter the city: ";
    std::getline(std::cin, city);

    if (db.createLocation(name, city) == SQLITE_OK) {
        std::cout << "\nStation added successfully.\n";
    } else {
        std::cerr << "\nError adding the station.\n";
    }

    pressToContinue();
}

void Admin::readStations() {
    clearScreen();

    std::cout << "==============================" <<
std::endl;
    std::cout << "        STATIONS LIST        " << std::endl;
    std::cout << "==============================" <<
std::endl;

    std::vector<std::string> stations = db.readLocations();

    if (stations.empty()) {
        std::cout << "\nNo stations in the database.\n";
    } else {
        std::cout << "\nStations:\n";
        for (const auto &station: stations) {
            std::cout << " - " << station << std::endl;
        }
    }
    std::cout << "==============================\n" <<
std::endl;

    pressToContinue();
}
```

```
    void Admin::deleteStation() {
        clearScreen();
        readStations();

        int station_id;
        std::cout << "==================================" <<
std::endl;
        std::cout << "        DELETE STATION         " << std::endl;
        std::cout << "==================================" <<
std::endl;

        std::cout << "Enter the station ID to delete: ";
        std::cin >> station_id;

        if (db.deleteLocation(station_id) == SQLITE_OK) {
            std::cout << "\nStation deleted successfully.\n";
        } else {
            std::cout << "\nError deleting the station.\n";
        }

        pressToContinue();
    }

    void Admin::printStationsMenu() {
        clearScreen();

        std::cout << "==================================" <<
std::endl;
        std::cout << "        STATIONS MENU          " << std::endl;
        std::cout << "==================================" <<
std::endl;
        std::cout << "1. Add a station" << std::endl;
        std::cout << "2. Delete station" << std::endl;
        std::cout << "3. View all stations" << std::endl;
        std::cout << "0. Exit" << std::endl;

        int command;
        std::cout << "Enter your choice: ";
        std::cin >> command;
        handleInvalidInput();

        switch (command) {
            case 1: {
                clearScreen();
                createStation();
                break;
            }
            case 2: {
                clearScreen();
                deleteStation();
                break;
            }
            case 3: {
                clearScreen();
                readStations();
                break;
            }
            case 0: {
                clearScreen();
                return;
            }
            default: {
                clearScreen();
                std::cout << "\nInvalid choice. Please try again.\n";
                pressToContinue();
                break;
            }
        }
    }
    void Admin::createTrain() {
        clearScreen();

        std::string train_number, type;
        std::cout    <<    "================================"    <<
std::endl;
        std::cout << "        CREATE TRAIN         " << std::endl;
        std::cout << "=============================" <<
std::endl;

        std::cout << "Enter the train number: ";
        std::cin.ignore();
        std::getline(std::cin, train_number);

        std::cout << "Enter the train type: ";
        std::getline(std::cin, type);

        if (db.createTrain(train_number, type) == SQLITE_OK) {
            std::cout << "\nTrain added successfully.\n";
        } else {
            std::cout << "\nError adding the train.\n";
        }

        pressToContinue();
    }

    void Admin::deleteTrain() {
        clearScreen();

        readTrains();
        int train_id;
        std::cout << "=============================" <<
std::endl;
        std::cout << "        DELETE TRAIN        " << std::endl;
        std::cout << "=============================" <<
std::endl;

        std::cout << "Enter the train ID to delete: ";
        std::cin >> train_id;

        if (db.deleteTrain(train_id) == SQLITE_OK) {
            std::cout << "\nTrain deleted successfully.\n";
        } else {
            std::cout << "\nError deleting the train.\n";
        }
```

```
        pressToContinue();
    }


    void Admin::readTrains() {
        clearScreen();

        std::vector<std::string> trains = db.readTrains();

        std::cout    <<    "=================================="    <<
std::endl;
        std::cout << "        TRAINS LIST        " << std::endl;
        std::cout    <<    "=================================="    <<
std::endl;

        if (trains.empty()) {
            std::cout << "\nNo trains in the database.\n";
        } else {
            std::cout << "\nTrains:\n";
            for (const auto &train: trains) {
                std::cout << " - " << train << std::endl;
            }
        }
        std::cout   <<   "==================================\n"   <<
std::endl;
    }

    void Admin::printTrainsMenu() {
        clearScreen();

        std::cout    <<    "=================================="    <<
std::endl;
        std::cout << "        TRAIN MENU         " << std::endl;
        std::cout    <<    "=================================="    <<
std::endl;
        std::cout << "1. Add Train               " << std::endl;
        std::cout << "2. Delete Train            " << std::endl;
```

```
            std::cout << "3. View All Trains          " << std::endl;
            std::cout << "-----------------------------" << std::endl;
            std::cout << "0. Exit                     " << std::endl;
            std::cout << "==================================" <<
std::endl;
            std::cout << "Enter your choice: ";

            int command;
            std::cin >> command;
            handleInvalidInput();
            std::cin.ignore();
            switch (command) {
                case 1: {
                    createTrain();
                    break;
                }
                case 2: {
                    deleteTrain();
                    break;
                }
                case 3: {
                    readTrains();
                    pressToContinue();
                    break;
                }
                case 0: {
                    clearScreen();
                    return;
                }
                default: {
                    clearScreen();
                    std::cout << "\nInvalid choice. Please try again.\n";

                    pressToContinue();
                    break;
                }
            }
        }
```

```
void Admin::createCarriage() {
    clearScreen();

    std::cout  <<  "================================"  <<
std::endl;
    std::cout << "        CREATE CARRIAGE        " << std::endl;
    std::cout  <<  "================================"  <<
std::endl;

    std::cout << "Choose train ID: " << std::endl;
    std::vector<int> train_ids = db.getTrainIds();
    if (train_ids.empty()) {
        std::cout << "No trains available.\n";
        return;
    }

    for (int id: train_ids) {
        std::cout << "Train ID: " << id << std::endl;
    }

    int train_id;
    std::cout << "Enter train ID: ";
    std::cin >> train_id;

    if (std::cin.fail()) {
        std::cout << "Invalid input for train ID.\n";
        return;
    }

    auto it = std::find(train_ids.begin(), train_ids.end(), train_id);
    if (it == train_ids.end()) {
        std::cout << "Invalid train ID.\n";
        return;
    }

    int number;
    std::string type;
    std::cout << "Enter carriage number: ";
    std::cin >> number;
    std::cout << "Enter carriage type (Compartment, Economy, Luxury): ";
    std::cin >> type;

    if (db.createCarriage(train_id, number, type) == SQLITE_OK) {
        std::cout << "\nCarriage added successfully.\n";
    } else {
        std::cout << "\nError adding carriage.\n";
    }


    pressToContinue();
}

void Admin::deleteCarriage() {
    clearScreen();

    int carriage_id;
    std::cout << "==============================" <<
std::endl;
    std::cout << "       DELETE CARRIAGE       " << std::endl;
    std::cout << "==============================" <<
std::endl;

    std::cout << "Enter carriage ID to delete: ";
    std::cin >> carriage_id;

    if (db.deleteCarriage(carriage_id) == SQLITE_OK) {
        std::cout << "\nCarriage deleted successfully.\n";
    } else {
        std::cout << "\nError deleting carriage.\n";
    }


    pressToContinue();
}
```

```
    void Admin::readCarriages() {
        clearScreen();

        std::vector<std::string> carriages = db.readCarriages();

        std::cout  <<  "================================"  <<
std::endl;
        std::cout << "        CARRIAGES LIST        " << std::endl;
        std::cout  <<  "================================"  <<
std::endl;

        if (carriages.empty()) {
            std::cout << "\nNo carriages available.\n";
        } else {
            std::cout << "\nCarriages:\n";
            for (const auto &carriage: carriages) {
                std::cout << " - " << carriage << std::endl;
            }
        }

        std::cout  <<  "================================"  <<
std::endl;

        pressToContinue();
    }

    void Admin::printCarriagesMenu() {
        clearScreen();

        std::cout  <<  "================================"  <<
std::endl;
        std::cout << "       CARRIAGES MENU         " << std::endl;
        std::cout  <<  "================================"  <<
std::endl;
        std::cout << "1. Add carriage              " << std::endl;
        std::cout << "2. Delete carriage           " << std::endl;
        std::cout << "3. View all carriages        " << std::endl;
        std::cout << "4. Exit                      " << std::endl;
        std::cout << "================================" <<
std::endl;
        std::cout << "Enter your choice: ";

        int command;
        std::cin >> command;
        handleInvalidInput();
        std::cin.ignore();
        switch (command) {
            case 1: {
                createCarriage();
                break;
            }
            case 2: {
                deleteCarriage();
                break;
            }
            case 3: {
                readCarriages();
                break;
            }
            case 4: {
                return;
            }
            default: {
                clearScreen();
                std::cout << "\nInvalid choice. Please try again.\n";
                pressToContinue();
                break;
            }
        }
    }
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Блок-схема алгоритма входа в приложение