**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

***Факультет Информационных технологий***

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**КУРСОВАЯ РАБОТА**

**Дисциплина:**Объектно-ориентированное программирование

**Тема**: Консольное приложения для управления системой учета студентов учебного заведения

**Выполнил:** студент **группы 231-332**

Алимов Аслан Рустамович

(Фамилия И.О.)

**Дата, подпись** *\_\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись

**Проверил:**  *\_\_\_\_\_\_\_\_\_\_\_\_*

(Фамилия И.О., степень, звание) (Оценка)

**Дата, подпись** *\_\_\_\_\_\_\_\_\_\_\_\_*

(Дата) (Подпись)

**Замечания:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва**

**2024**

АННОТАЦИЯ

Курсовой проект. Московский политехнический университет. Тема работы: «Разработка консольного приложения для управления системой учета студентов учебного заведения».

Выполнил — Алимов А. Р.

Руководитель работы — Полубояринова А.С..

Направление подготовки — 09.03.02 «Информационные системы и технологии».

Курсовой проект состоит из пояснительной записки и разработанного консольного приложения.

Пояснительная записка состоит из 19 стр. текст. приложения. Включает в себя разделы: введение, описание предметной области и назначение проектируемого приложения, проектирование и реализация приложения, заключение, список использованных источников приложение.

В курсовой работе описан процесс проектирования и разработки консольного приложения для управления системой учета студентов учебного заведения, использующего возможности командной строки для взаимодействия с пользователем. В ходе выполнения проекта были разработаны и протестированы основные модули приложения, включая добавление новых студентов, редактирование личных данных, учет успеваемости (оценки), поиск студентов по заданным параметрам, генерацию отчетов об успеваемости.

Ключевые слова: консольное приложение, учет студентов, программирование, командная строка, управление успеваемостью.

ОГЛАВЛЕНИЕ

[АННОТАЦИЯ 2](#_Toc185529401)

[ОГЛАВЛЕНИЕ 3](#_Toc185529402)

[ВВЕДЕНИЕ 4](#_Toc185529403)

[1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И НАЗНАЧЕНИЕ ПРОЕКТИРУЕМОГО 5](#_Toc185529404)

[1.2 Описание функционала информационной системы 5](#_Toc185529405)

[1.3 Описание классов и их иерархии 8](#_Toc185529406)

[2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ КОНСОЛЬНОГО ПРИЛОЖЕНИЯ 13](#_Toc185529407)

[2.1 Этапы проектирования консольного приложения 13](#_Toc185529408)

[2.2 Выбор и описание программных средств реализации информационной системы 13](#_Toc185529409)

[ЗАКЛЮЧЕНИЕ 15](#_Toc185529410)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc185529411)

[Приложение 17](#_Toc185529412)

[Листинг кода класса DataNotes: 17](#_Toc185529413)

[Листинг кода класса DataUser: 21](#_Toc185529414)

[Листинг кода класса ChoiceCheck: 24](#_Toc185529415)

[Листинг кода класса ColorConsole: 25](#_Toc185529416)

[Листинг кода класса Gui: 26](#_Toc185529417)

[Листинг кода класса Initialization: 26](#_Toc185529418)

[Листинг кода класса Main: 28](#_Toc185529419)

[Листинг кода класса Notes: 30](#_Toc185529420)

[Листинг кода класса User: 32](#_Toc185529421)

[Листинг кода класса Program: 33](#_Toc185529422)

[Пример работы приложения: 33](#_Toc185529423)

ВВЕДЕНИЕ

Современный мир быстро развивается, и с каждым годом технологии становятся неотъемлемой частью нашей жизни. В образовательных учреждениях, где объем данных о студентах и учебном процессе постоянно увеличивается, эффективное управление этой информацией становится ключевым фактором успешной работы. Сложности в учете студентов, контроле их успеваемости и организации отчетности являются актуальной проблемой для большинства учебных заведений.

Многие существующие системы учета студентов могут быть либо слишком сложными для использования, либо требовать значительных ресурсов для их внедрения и поддержки. Консольное приложение для управления системой учета студентов учебного заведения было разработано с целью упростить процесс управления данными о студентах и улучшить организацию образовательного процесса. Простота в использовании, а также удобство работы через командную строку делают приложение доступным для широкой аудитории и позволяют легко интегрировать его в повседневную деятельность учебного заведения.

Главная цель разработки данного приложения — создать инструмент, который позволит пользователям эффективно управлять данными о студентах, включая их личную информацию, успеваемость и посещаемость, не перегружая интерфейс ненужными функциями. Использование текстового интерфейса позволяет сосредоточиться на главных задачах — учете данных и генерации отчетов, исключая необходимость отвлекаться на сложные настройки или графические элементы.

Разработанное приложение предоставляет функциональность, которая охватывает ключевые аспекты управления учебным процессом, включая добавление, редактирование, поиск студентов и создание отчетов об их успеваемости.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И НАЗНАЧЕНИЕ ПРОЕКТИРУЕМОГО

**1.1** Описание назначения информационной системы

Проектируемое приложение представляет собой консольную систему управления данными студентов (Student Management System). Основное назначение — автоматизация процессов управления информацией о студентах, их группах и академической успеваемости.

Цель системы:

* Упрощение работы преподавателей и учебных отделов.
* Повышение точности и удобства хранения данных.
* Обеспечение инструментов для анализа успеваемости студентов и создания отчетов.

Система находит применение в образовательных учреждениях, таких как школы, колледжи и университеты, поддерживая управление информацией в рамках учебного процесса.

#### **1.2 Описание функционала информационной системы**

Приложение обеспечивает выполнение основных задач управления данными студентов:

**I. Управление студентами**

1. **Добавление студентов**:
   * Ввод имени, фамилии и группы.
   * Сохранение данных в файл базы.
   * Цель: создание базы данных студентов.
2. **Редактирование данных студентов**:
   * Возможность изменения группы студента.
   * Цель: актуализация информации о студентах.
3. **Удаление студентов**:
   * Удаление записей студентов из базы.
   * Цель: устранение данных о выпускниках или удаленных записях.

**II. Управление успеваемостью**

1. **Добавление оценок**:
   * Ввод оценок по каждому предмету.
   * Привязка оценок к конкретному студенту.
   * Цель: формирование базы успеваемости.
2. **Редактирование оценок**:
   * Изменение существующих оценок.
   * Цель: коррекция ошибок или обновление данных.

**III. Формирование отчетов**

1. **Генерация общего отчета**:
   * Вывод всех студентов с их успеваемостью.
   * Цель: обзор академических показателей.
2. **Генерация отчетов по группам**:
   * Формирование отчета для выбранной группы.
   * Цель: детализированный анализ успеваемости в конкретной группе.
3. **Расчет средней успеваемости**:
   * Подсчет средней оценки по студенту или группе.
   * Цель: оценка академических результатов.

**IV. Интерфейс взаимодействия**

1. **Удобное меню выбора**:
   * Пользователь выбирает действия из предложенного списка.
   * Цель: интуитивное управление системой.
2. **Обратная связь**:
   * Информирование пользователя о результатах выполнения действий.
   * Указание на ошибки или успех выполнения операций.

***1.3 Описание классов и их иерархии***

**Основные классы системы**:

* **Student**
  + Назначение: Представляет студента с уникальным ID, именем, фамилией и группой.
  + Методы:
    - ShowInfo(): Вывод данных студента.
    - UpdateGroup(): Изменение группы.
* **Grade**
  + Назначение: Управляет данными об оценках студентов.
  + Методы:
    - AddGrade(): Добавление новой оценки.
    - EditGrade(): Редактирование существующей оценки.
    - CalculateAverage(): Подсчет средней оценки.
* **DataManager**
  + Назначение: Обеспечивает работу с файлами и базой данных.
  + Методы:
    - SaveData(): Сохранение данных в файлы.
    - LoadData(): Загрузка данных при запуске.
* **ReportManager**
  + Назначение: Формирование отчетов.
  + Методы:
    - GenerateGroupReport(): Создание отчета по группе.
    - GenerateFullReport(): Создание общего отчета.
* **UIManager**
  + Назначение: Управление взаимодействием с пользователем.
  + Методы:
    - DisplayMenu(): Вывод меню.
    - HandleInput(): Обработка ввода пользователя.
* **MainProgram**
  + Назначение: Основной контроллер приложения.
  + Методы:
    - Run(): Запуск основного цикла приложения.

#### **Пример использования:**

1. Пользователь запускает приложение и выбирает действие из меню: добавить студента, ввести оценки, просмотреть отчеты.
2. Система сохраняет изменения в базу данных и предоставляет отчеты по запросу.
3. При следующем запуске приложение восстанавливает данные из файлов.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ КОНСОЛЬНОГО ПРИЛОЖЕНИЯ

2.1 Этапы проектирования консольного приложения

Проектирование консольного приложения осуществлялось по следующим этапам:

1. **Сбор требований**:
   * Определение основных целей и задач системы;
   * Анализ потребностей целевой аудитории, включая преподавателей и администраторов;
   * Формирование и согласование технического задания.
2. **Анализ предметной области**:
   * Изучение существующих решений для управления данными студентов;
   * Выделение ключевых функций, необходимых для реализации в системе.
3. **Разработка архитектуры**:
   * Построение UML-диаграмм классов, прецедентов и последовательностей;
   * Проектирование структуры данных, включая классы, их атрибуты и методы;
   * Определение способов взаимодействия между компонентами системы.
4. **Дизайн интерфейса**:
   * Проектирование структуры меню и консольных интерфейсов;
   * Упрощение процесса взаимодействия пользователя с системой;
   * Разработка механизмов для информирования пользователя о ходе выполнения операций.
5. **Реализация**:
   * Создание функциональных модулей для работы с файлами, управления студентами и формирования отчетов;
   * Проведение модульного и интеграционного тестирования системы.
6. **Документирование**:
   * Разработка технической документации, включая описание структуры системы, классов и методов;
   * Создание руководства пользователя для эффективного взаимодействия с приложением.

2.2 Выбор и описание программных средств реализации информационной системы

Для реализации приложения были выбраны следующие инструменты и технологии:

* **Microsoft Visual Studio**:
  + Интегрированная среда разработки (IDE) для написания, отладки и тестирования приложения;
* **.NET Framework**:
  + Фреймворк, обеспечивающий поддержку основных функций приложения, включая работу с файлами и обработку данных;
  + Библиотеки для реализации консольного интерфейса и управления вводом/выводом данных.
* **Git**:
  + Система контроля версий, использованная для сохранения изменений в коде и работы с различными ветками;
  + Обеспечение совместной разработки и возможности отката к предыдущим версиям проекта.
* **CSV**:
  + Формат данных для хранения информации о студентах и успеваемости;
  + Простота чтения и записи данных без необходимости использования сложных систем управления базами данных.
* **UML-диаграммы**:
  + Использовались для моделирования архитектуры системы и документирования логики взаимодействия компонентов.

Данный набор инструментов позволил реализовать проект эффективно, обеспечивая производительность и легкость дальнейшего расширения функциональности.

ЗАКЛЮЧЕНИЕ

В ходе разработки консольного приложения для учета студентов в учебном заведении была создана система, которая позволяет хранить информацию о личных данных, успеваемости, а также генерировать отчеты по успеваемости. Приложение реализовано с использованием объектно-ориентированного подхода, что обеспечивает модульность, расширяемость и удобство дальнейшей разработки.

Разработанный функционал включает в себя добавление студентов, редактирование их данных, хранение оценок и поиск студентов по различным параметрам. Удобное меню и простота взаимодействия с пользователем позволяют легко управлять данными и генерировать отчеты. Система может быть доработана и адаптирована для различных нужд учебных заведений, что делает её универсальной и полезной в реальных условиях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Microsoft Docs. Руководство по языку C# / URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/> (дата обращения: 20.12.2024).
2. Пол Джонс. Основы объектно-ориентированного программирования / URL: <https://www.oreilly.com/library/view/learning-object-oriented-programming/%20>
3. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. —  
   Ссылка: <https://www.ozon.ru/context/detail/id/13060764/>
4. Соммерс Дж. Объектно-ориентированное программирование: Современные подходы и инструменты. — М.: Бином, 2020.  
   Ссылка: <https://www.ozon.ru/context/detail/id/151218303/>
5. Шмидт Д. Шаблоны проектирования и архитектуры в объектно-ориентированном программировании. — М.: Вильямс, 2020.  
   Ссылка: https://www.ozon.ru/context/detail/id/13638371/
6. Чакон С., Штрауб Б. Pro Git. — 2-е изд. / URL: <https://git-scm.com/book/ru/v2> (дата обращения: 20.12.2024).
7. Освой программирование играючи. Сайт Александра Климова / URL: <http://developer.alexanderklimov.ru/android/> (дата обращения: 25.01.2021).

Приложение

**Листинг кода класса Grade:**

class Grade {

String subject

int grade

Grade(String subject, int grade) {

this.subject = subject

this.grade = grade

}

}

**Листинг кода класса ReportGenerator:**

class ReportGenerator {

static void generatePerformanceReport(List<Student> students) {

students.each { student ->

println "Студент: ${student.getFullName()}, Группа: ${student.group}, Средний балл: ${student.getAverageGrade()}"

student.printGrades()

println "-----------------------------"

}

}

static void generateGroupReport(List<Student> students, String group) {

def groupStudents = students.findAll { it.group == group }

if (groupStudents.isEmpty()) {

println "Студентов в группе $group нет."

} else {

println "Отчет по группе: $group"

generatePerformanceReport(groupStudents)

}

}

}

**Листинг кода класса Student:**

class Student {

String firstName

String lastName

String group

List<Grade> grades = []

Student(String firstName, String lastName, String group) {

this.firstName = firstName

this.lastName = lastName

this.group = group

}

void addGrade(String subject, int grade) {

this.grades.add(new Grade(subject, grade))

}

void updateGrade(String subject, int newGrade) {

def grade = this.grades.find { it.subject == subject }

if (grade) {

grade.grade = newGrade

}

}

void updateGroup(String newGroup) {

this.group = newGroup

}

String getFullName() {

return "$firstName $lastName"

}

double getAverageGrade() {

if (grades.isEmpty()) return 0

return grades.sum { it.grade } / grades.size()

}

void printGrades() {

grades.each { grade ->

println " ${grade.subject}: ${grade.grade}"

}

}

}

**Листинг кода класса StudentDataBase:**

class StudentDatabase {

List<Student> students = []

void addStudent(Student student) {

students.add(student)

}

void removeStudent(String fullName) {

students.removeIf { it.getFullName() == fullName }

}

Student findStudentByFullName(String fullName) {

return students.find { it.getFullName() == fullName }

}

List<Student> findStudentsByGroup(String group) {

return students.findAll { it.group == group }

}

List<Student> findStudentsByName(String name) {

return students.findAll { it.firstName.contains(name) || it.lastName.contains(name) }

}

List<Student> getAllStudents() {

return students

}

List<Student> findStudentsByAverageGrade(double grade) {

return students.findAll { it.getAverageGrade() >= grade }

}

}

**Листинг кода класса** **StudentManagementSystem:**

import java.util.Scanner

class StudentManagementSystem {

static StudentDatabase database = new StudentDatabase()

static void main(String[] args) {

Scanner scanner = new Scanner(System.in)

while (true) {

println "Меню:"

println "1. Добавить студента"

println "2. Редактировать данные студента"

println "3. Удалить студента"

println "4. Найти студента"

println "5. Поиск студентов по средней оценке"

println "6. Сгенерировать отчет по успеваемости"

println "7. Сгенерировать отчет по группе"

println "8. Выйти"

print "Выберите опцию: "

String choice = scanner.nextLine()

switch (choice) {

case "1":

addStudent(scanner)

break

case "2":

editStudent(scanner)

break

case "3":

removeStudent(scanner)

break

case "4":

searchStudent(scanner)

break

case "5":

searchByAverageGrade(scanner)

break

case "6":

generateReport()

break

case "7":

generateGroupReport(scanner)

break

case "8":

println "Выход..."

return

default:

println "Некорректный выбор. Попробуйте снова."

}

}

}

static void addStudent(Scanner scanner) {

print "Введите имя студента: "

String firstName = scanner.nextLine()

print "Введите фамилию студента: "

String lastName = scanner.nextLine()

print "Введите группу студента: "

String group = scanner.nextLine()

Student student = new Student(firstName, lastName, group)

database.addStudent(student)

println "Студент ${student.getFullName()} добавлен!"

}

static void editStudent(Scanner scanner) {

print "Введите фамилию и имя студента для редактирования: "

String fullName = scanner.nextLine()

Student student = database.findStudentByFullName(fullName)

if (student) {

println "Выбрали студента: ${student.getFullName()}"

println "1. Изменить группу"

println "2. Изменить оценку"

print "Выберите опцию: "

String option = scanner.nextLine()

if (option == "1") {

print "Введите новую группу: "

String newGroup = scanner.nextLine()

student.updateGroup(newGroup)

println "Группа изменена на $newGroup."

} else if (option == "2") {

print "Введите название предмета: "

String subject = scanner.nextLine()

print "Введите новую оценку: "

int grade = scanner.nextInt()

scanner.nextLine() // consume newline

student.updateGrade(subject, grade)

println "Оценка для предмета $subject обновлена на $grade."

}

} else {

println "Студент не найден."

}

}

static void removeStudent(Scanner scanner) {

print "Введите фамилию и имя студента для удаления: "

String fullName = scanner.nextLine()

database.removeStudent(fullName)

println "Студент $fullName удален."

}

static void searchStudent(Scanner scanner) {

print "Введите фамилию или имя для поиска: "

String name = scanner.nextLine()

List<Student> foundStudents = database.findStudentsByName(name)

if (foundStudents.isEmpty()) {

println "Студенты не найдены."

} else {

foundStudents.each { student ->

println "Студент: ${student.getFullName()}, Группа: ${student.group}"

}

}

}

static void searchByAverageGrade(Scanner scanner) {

print "Введите минимальную среднюю оценку для поиска: "

double grade = scanner.nextDouble()

List<Student> foundStudents = database.findStudentsByAverageGrade(grade)

if (foundStudents.isEmpty()) {

println "Студенты с такой средней оценкой не найдены."

} else {

foundStudents.each { student ->

println "Студент: ${student.getFullName()}, Группа: ${student.group}, Средний балл: ${student.getAverageGrade()}"

}

}

}

static void generateReport() {

println "Генерация отчета по успеваемости..."

ReportGenerator.generatePerformanceReport(database.getAllStudents())

}

static void generateGroupReport(Scanner scanner) {

print "Введите группу для генерации отчета: "

String group = scanner.nextLine()

ReportGenerator.generateGroupReport(database.getAllStudents(), group)

}

}

]

Пример работы приложения:

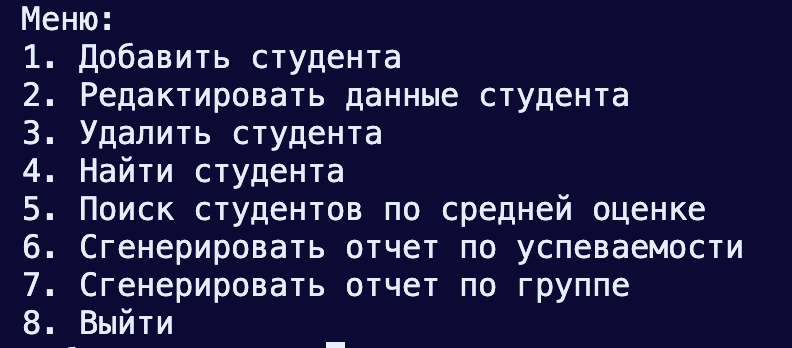


Рисунок 1 Меню

Главное меню приложения представляет собой центральный интерфейс для взаимодействия пользователя с системой. Оно отображает список доступных действий, упрощая навигацию и обеспечивая доступ к основным функциям.

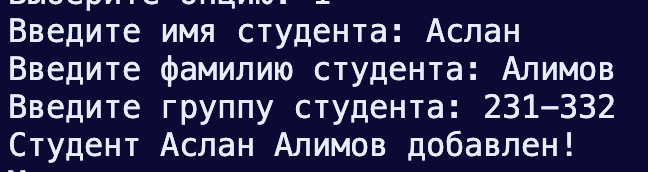


Рисунок - Добавление студента

Этот экран позволяет пользователю ввести данные нового студента для добавления в базу.

Описание:

1. Пользователь вводит имя, фамилию, номер группы и дополнительную информацию (при необходимости).
2. Система проверяет корректность ввода (например, уникальность записи или обязательность полей).
3. После успешного добавления выдается подтверждающее сообщение.

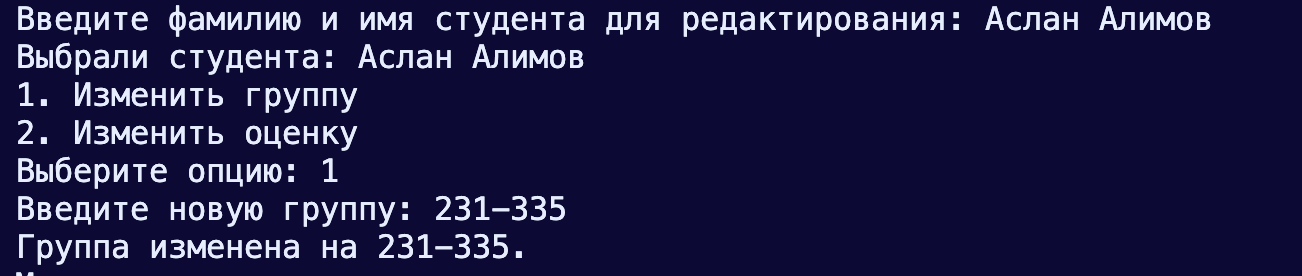


Рисунок - Редактирование студента

Позволяет изменять информацию о выбранном студенте.

Описание:

1. Пользователь выбирает студента из списка или вводит идентификатор.
2. На экране отображаются текущие данные, которые можно редактировать (например, фамилию, группу).
3. Сохраненные изменения обновляют базу данных, а система выдает подтверждение.

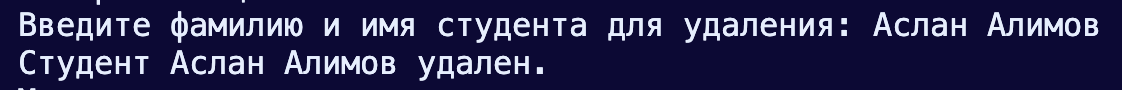


Рисунок – Удаление

Позволяет удалить студента из базы



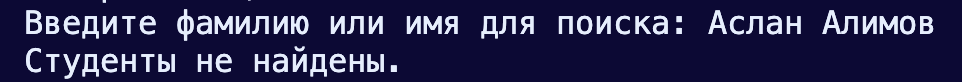
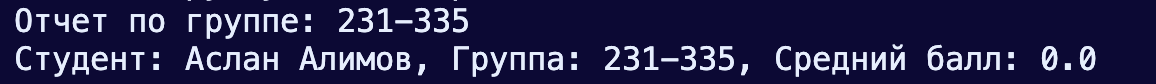


Рисунок - Поиск студента

Позволяет найти студента по его данным



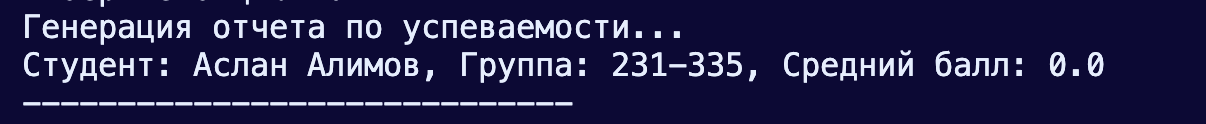


Рисунок - Генерация отсчета

Этот раздел формирует отчет о результатах обучения.

Описание:

1. Пользователь выбирает параметры отчета (например, группу или временной период).
2. Система генерирует отчет с указанием фамилий студентов, предметов и средних оценок.

Ссылка на репозиторий на [GitHub](https://github.com/Velial11/kursach.git)