МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка алгоритма учёта посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом

Обучающийся Н.И. Желтов
(И.О. Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень, звание, И.О. Фамилия)

Консультант к.п.н., доцент О.Н. Брега
(ученая степень, звание, И.О. Фамилия)

Тольятти 2023

Аннотация

Тема данной выпускной квалификационной работы «Разработка алгоритма учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом».

В работе представлена исполняемая программа, в которой реализован алгоритм учета посещаемости. Программа запускается через терминал и не принимает никаких дополнительных ключей.

Данная выпускная квалификационная работа представляет из себя: введение, три основные главы, заключение, а также список используемой литературы и задействованных источников.

Введение раскрывает актуальность темы, поставленную цель и задачи для реализации автоматического учета посещаемости студентов и преподавателей.

Первая глава содержит описание существующих алгоритмов, системы контроля и управления доступом, поставлены задачи на разработку алгоритма учета посещаемости.

Вторая глава содержит описание разработки алгоритма учета посещаемости и его реализации.

Третья глава представляет собой описание тестирования работоспособности реализованной программы по учету посещаемости студентов и преподавателей.

Заключение содержит из себя выводы, которые были сделаны в результате проведенной работы.

В работе использовано 1 таблица, 32 рисунка и 25 ссылок на внешние ресурсы. Общий объем выпускной квалификационной работы составил 56 страницы.

Abstract

The topic of this graduate qualification work "Development of an algorithm for accounting attendance by students and teachers based on events of the access control and management system".

The paper presents an executable program that implements the attendance accounting algorithm. The program runs through the terminal and does not accept any additional keys.

This graduate qualification work is: an introduction, the three main chapters, the conclusion, as well as a list of references and sources used.

The introduction reveals the relevance of the topic, the set goal and objectives for the implementation of automatic accounting for student and teacher attendance.

The first chapter contains a description of the existing algorithms, access control and management system, the tasks for the development of the attendance accounting algorithm.

The second chapter contains a description of the development of the attendance accounting algorithm and its implementation.

The third chapter is a description of testing the performance of the implemented program to account for the attendance of students and teachers.

The conclusion contains the conclusions that were made as a result of the work.

One table, 32 pictures and 25 references to external resources are used in the work. The total volume of the graduate qualification work was 56 pages.

Оглавление

Введение

В настоящее время вопросы контроля посещаемости студентами учебных заведений являются одним из главных приоритетов в развитии профессионального государственного образования в России.

Система учета посещаемости занятий является важным компонентом образовательных учреждений, так как она позволяет отслеживать активность студентов и преподавателей на уроках, оценивать их работу и успеваемость. Однако, традиционные методы учета посещаемости, такие как ручное заполнение журналов и табелей, являются неэффективными и недостаточно точными. В связи с этим, задача разработки алгоритма учета посещаемости занятий имеет большое значение для повышения качества образования и эффективности управления учебным процессом.

Цель данной работы – разработать алгоритм учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом, который позволит автоматизировать процесс учета и обработки данных о посещаемости и улучшить качество управления учебным процессом.

Для достижения цели были поставлены следующие задачи:

- провести анализ существующих алгоритмов;
- провести обзор системы контроля и управления доступом;
- проанализировать существующие методы учета посещаемости занятий;
- выбрать систему контроля и управлением доступом, которая поможет в реализации алгоритма учета посещаемости занятий;
- собрать данные о посещаемости занятий с помощью системы контроля и управления доступом;
- собрать данные о расписании занятий;
- предварительно обработать данные и отобрать признаки для анализа;

– разработать алгоритм учета посещаемости студентов и преподавателей;
– оценить точность и эффективность алгоритма на основе данных о посещаемости занятий;
– проанализировать преимущества и недостатки разработанного алгоритма.

Разработка алгоритма учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом имеет важное значение для улучшения качества образования и эффективности управления учебным процессом. Введение автоматизированного алгоритма позволит упростить процесс учета и обработки данных, а также повысить качество управления учебным процессом.

В первой главе проведен анализ существующих алгоритмов, которые входят в алгоритм разработки. Проведен обзор системы контроля и управления доступом, где детально рассмотрены включенные в них технологии. Также выполнен анализ методов учета посещаемости, применяемых на данный момент.

Во второй главе была выбрана подходящая система контроля и управления доступом. Был проведен первичный сбор данных о посещаемости занятий при помощи данной системы, а также собраны данные о расписании занятий. После обработки полученных данных, был реализован алгоритм для учета посещаемости занятий.

Третья глава посвящена оценке эффективности разработанного алгоритма учета посещаемости занятий. Было проведено тестирование работы алгоритма, в ходе которого были выявлены его преимущества и недостатки.

Разработанный в работе алгоритм учета посещаемости на основе событий систем контроля и управления доступом является главным результатом выпускной квалификационной работы.

Глава 1 Теоретические основы алгоритма учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом

1.1 Описание и анализ существующих алгоритмов

Алгоритм – это последовательность шагов или действий, которые необходимо выполнить для решения определенной задачи или достижения определенной цели.

В контексте программирования, существуют три основных типа алгоритмов: линейные, разветвляющиеся и циклические.

Линейный алгоритм представляет собой последовательность действий, которые выполняются по порядку. Такой алгоритм не имеет ветвлений или циклов. Примером может быть алгоритм вычисления площади прямоугольника: сначала берется длина стороны, затем ширина, и затем они перемножаются.

Разветвляющийся алгоритм, как следует из названия, включает в себя ветвления, которые позволяют программе выполнять различные действия в зависимости от условий. Примером может быть алгоритм определения, является ли число четным или нечетным: если число делится на 2 без остатка, оно четное, в противном случае – нечетное [18].

Циклический алгоритм позволяет повторять определенную последовательность действий несколько раз. Цикл может быть организован как с фиксированным количеством повторений, так и с переменным количеством повторений, зависящим от условий. Примером может быть алгоритм вычисления факториала числа: умножение числа на все целые числа от 1 до этого числа. В данном случае цикл будет повторяться фиксированное количество раз – столько, сколько равно данному числу [18].

В разработке программных решений важную роль играют вспомогательные алгоритмы, которые помогают повысить эффективность основных алгоритмов. Они выполняют конкретные задачи, которые

используются в основном алгоритме для достижения общей цели. В этом контексте, понимание различных типов функций и процедур, которые могут использоваться в алгоритмах, также является важным. Разнообразие возможностей, которые предоставляют эти элементы, позволяют решать задачи более эффективно и гибко [6].

Существуют различные типы функций и процедур, которые могут использоваться в алгоритмах. Процедуры – это фрагменты кода, которые выполняют определенные действия и могут принимать параметры. Функции – это фрагменты кода, которые также выполняют определенные действия, но при этом возвращают значение. Также существуют процедуры без параметров и функции без возвращаемого значения [6].

Вспомогательный алгоритм может содержать как процедуры, так и функции, которые могут быть использованы в основном алгоритме для выполнения задачи.

Для реализации основного алгоритма учета посещаемости необходимо использовать несколько алгоритмов, которые будут являться подпрограммами и выполнять различные задачи.

Для контроля посещаемости студентов и преподавателей необходимо иметь данные о расписании занятий, поскольку только при наличии такой информации можно определить, какие занятия проходят в конкретный день и время. Эти данные могут быть получены с помощью парсинга веб-страницы сайта образовательной организации, где размещено расписание занятий.

Алгоритм парсинга – это подпрограмма, которая используется для извлечения данных с веб-страницы. Расписание занятий на сайте обновляется регулярно, и использование парсинга в данном случае помогает автоматизировать процесс сбора информации, обеспечивая более точную и актуальную информацию для учета посещаемости [14].

Основная задача алгоритма парсинга – это преобразование HTML-кода страницы в структурированные данные, которые можно использовать в программе. Алгоритм парсинга может быть написан на различных языках

программирования, таких как Python, Java, JavaScript и других. В основе работы алгоритма парсинга лежит использование регулярных выражений и DOM-модели, которые позволяют извлекать нужную информацию с веб-страницы [25].

Для работы основного алгоритма учета посещаемости необходимо иметь данные о каждом проходе студента и преподавателя, которые обычно хранятся в базе данных системы контроля и управления доступом. Для получения этой информации можно использовать алгоритм извлечения данных из базы данных. Он позволит получить нужную информацию и сформировать ее в удобном виде для дальнейшей обработки. Кроме того, после извлечения данных о проходах, необходимо связать эти данные с данными о студентах и преподавателях, чтобы провести учет посещаемости. Для этого подойдет алгоритм связывания, который позволит соединить данные из разных таблиц и получить полную информацию о каждом проходе студента и преподавателя.

Алгоритм связывания – это процесс соединения данных из нескольких таблиц в единую таблицу для облегчения доступа к информации. Связывание таблиц может быть выполнено, используя SQL запросы. Это позволяет избежать дублирования данных и обеспечить целостность информации [23].

Существует несколько типов связей между таблицами, таких как один к одному, один ко многим, многие ко многим. Алгоритм связывания зависит от типа связи и используемой базы данных. Однако, общая схема связывания состоит из нескольких шагов, таких как определение связей между таблицами, создание внешних ключей, выполнение запросов для объединения таблиц и обработки полученных данных [24].

После того, как данные будут извлечены из базы данных, необходимо провести сравнение полученных данных с данными о расписании занятий. Для этой задачи подойдет алгоритм сравнения, который позволит выявить пропущенные занятия и сформировать отчет об учете посещаемости.

Алгоритм сравнения данных даты и времени – это алгоритм, который используется для сравнения двух или более значений даты и времени. Он может быть реализован с использованием стандартных функций и операторов, которые поддерживаются большинством языков программирования [17].

Для сравнения двух дат и времен используются операторы сравнения, такие как "меньше", "больше", "меньше или равно" и "больше или равно". Операторы сравнения работают на основе значения временной метки и позволяют определить, какая из дат является более ранней или более поздней [22].

Таким образом, использование этих алгоритмов позволит автоматизировать процесс учета посещаемости, сократить время, необходимое для его проведения, а также обеспечить более точную и актуальную информацию о посещаемости студентов и преподавателей.

1.2 Обзор системы контроля и управления доступом

Система контроля и управления доступом (СКУД) – это комплексная система, которая позволяет контролировать доступ персонала в здания и помещения. Она обычно состоит из нескольких компонентов: считывателей, контроллеров доступа, программного обеспечения и базы данных [15].

Основные задачи системы контроля и управления доступом в образовательной организации включают в себя:

- обеспечение безопасности зданий и помещений учебного заведения [15];
- – контроль доступа преподавателей и студентов в здания и помещения [15];
- учет рабочего времени преподавателей и посещаемости студентов [15];
- подсчет всех проходов турникета за день одного человека [15].

Считыватели в системе контроля и управления доступом устанавливаются на входах в здания и помещения, а также на других стратегически важных местах. Они считывают данные с карт доступа, которые используются для идентификации персонала и студентов [15].

Контроллеры доступа являются главным узлом системы контроля и управления доступом. Они обрабатывают данные, которые поступают от считывателей, и решают, кто может получить доступ к зданиям и помещениям учебного заведения [15].

Сервер системы контроля и управления доступом позволяет управлять контроллерами доступа, конфигурировать правила доступа и управлять базой данных, в которой хранится информация о персонале и студентах, а также об их правах доступа [2].

Система контроля и управления доступом для образовательной организации может также включать модули для учета посещаемости студентов, которые позволяют контролировать и анализировать посещаемость занятий, и модули для учета рабочего времени преподавателей.

Помимо обеспечения безопасности и контроля доступа, система контроля и управления доступом для образовательной организации может быть использована для повышения эффективности управления учебным процессом, сокращения затрат на административную работу и улучшения качества образования.

Таким образом, система контроля и управления доступом для образовательной организации является важным инструментом для обеспечения безопасности зданий и помещений учебного заведения, учета посещаемости студентов и преподавателей, а также контроля доступа к определенным учебным материалам и ресурсам.

1.3 Обзор реализованных методов учета посещаемости

В процессе исследования задачи по системе учета посещаемости были обнаружены уже существующие методы, используемые для учета посещаемости в образовательных учреждениях. Давайте более подробно рассмотрим эти методы, которые в настоящее время используются в образовательном процессе.

Ручной учет посещаемости – это процесс, при котором преподаватель или другой ответственный сотрудник учебного заведения вручную отмечает, кто присутствует на занятии. В прошлом это часто делалось в бумажных журналах, но сейчас многие учебные заведения перешли на электронные журналы.

Одним из преимуществ ручного учета является его простота. Этот метод не требует никаких специальных технических знаний, а также не требует дополнительных инвестиций в оборудование и программное обеспечение.

Однако, ручной учет имеет несколько недостатков. Прежде всего, он требует значительных усилий со стороны преподавателей и других ответственных сотрудников. Кроме того, этот метод не идеально точен – возможны ошибки внесения данных, а также мошенничество со стороны студентов (например, если один студент отметился за другого).

Использование бесконтактных карт – это один из наиболее распространенных способов учета посещаемости в образовательных учреждениях. Каждый студент или преподаватель получает уникальную бесконтактную карту, которая затем используется для регистрации их посещений. Карты могут быть считывающими устройствами, установленными на входе в каждое помещение. Когда карты проходят через считывающее устройство, информация об этом событии регистрируется в базе данных [10].

Преимуществом такого подхода является то, что он довольно прост в использовании и не требует специальных навыков. Кроме того, использование

бесконтактных карт позволяет быстро и точно записывать посещаемость, что облегчает контроль посещаемости для преподавателей и администрации учебного заведения.

Однако недостатком этого метода является возможность потери или повреждения карты, что может привести к проблемам с учетом посещаемости. Кроме того, считывающее устройство должно быть установлено на каждом входе, что может быть затратным для крупных учебных заведений с большим количеством зданий.

Использование биометрических технологий в системах учета посещаемости в образовательных учреждениях становится все более популярным и эффективным методом. Биометрические технологии позволяют идентифицировать студентов и преподавателей по их физическим параметрам, таким как отпечатки пальцев, распознавание лица и др [10].

Для использования биометрических технологий в системе учета посещаемости в образовательном учреждении необходимо установить соответствующее оборудование, которое обычно включает в себя сканеры отпечатков пальцев, камеры для распознавания лица. В процессе работы системы, биометрические данные считываются с помощью оборудования и отправляются на сервер для их обработки.

Преимущества использования биометрических технологий в системе учета посещаемости включают в себя высокую точность и быстроту идентификации, а также отсутствие необходимости в физической карте доступа или других идентификационных средствах. Кроме того, биометрические технологии могут помочь предотвратить мошенничество и злоупотребления системой, так как данные студентов и преподавателей уникальны и не могут быть подделаны [9].

Однако, использование биометрических технологий также имеет свои недостатки. Во-первых, они могут быть дорогостоящими и требовать значительных затрат на установку и обслуживание оборудования. Во-вторых,

есть риски в отношении защиты данных и приватности, так как биометрические данные являются чувствительной информацией [9].

Тем не менее, использование биометрических технологий в системе учета посещаемости может быть эффективным и надежным методом, если будут приняты соответствующие меры для обеспечения безопасности и защиты данных.

Для более наглядного сравнения этих методов, представим их в виде сравнительной таблице. Таблица 1 содержит сравнительную сводку по каждому методу.

Таблица 1 – Сравнительная таблица качеств представленных методов

Название	Достоинства	Недостатки
Ручной учет	Простота в использовании; Доступность	Ошибки при ведении учета; Неэффективность для больших групп
Использование бесконтактных карт	Удобство использования; Ускорение процесса контроля	Возможность потери или кражи карты; Считывающее устройство должно быть установлено на каждом входе в здании
Использование биометрических технологий	Высокая точность и быстрота идентификации; Данные уникальны	Требуют значительных затрат; Риски в отношении защиты данных и приватности; Возможность отказа системы из-за технических проблем

На основе анализа был выбран метод бесконтактных карт, так как он позволяет сохранять информацию в базу данных о каждом входе и выходе каждого студента и преподавателя.

Рассмотрим блок-схему метода использования бесконтактных карт (рисунок 1).

Рисунок 1 – Блок-схема метода бесконтактных карт

Теперь рассмотрим подробно выбранный метод учета посещаемости. Выглядит он следующим образом:

При входе в корпус студент/преподаватель прикладывает свою бесконтактную карту к считывающему устройству. Бесконтактная карта содержит уникальный идентификатор, который связан с соответствующей записью о студенте/преподавателе в базе данных.

Для учета посещаемости необходимо регистрировать время прихода и ухода каждого студента/преподавателя в корпус. Это реализовано с помощью двух отдельных записей в базе данных: одна запись содержит информацию о времени прихода, а другая – о времени ухода.

Каждая запись в базе данных содержит следующую информацию: идентификатор карты, дата и время события (приход/уход), ФИО студента/преподавателя.

В результате проведенного исследования было обнаружено, что для учета посещаемости в образовательных учреждениях существуют различные методы. После анализа этих методов было принято решение использовать метод бесконтактных карт, который позволяет регистрировать информацию о каждом входе и выходе студентов и преподавателей в базе данных. Этот метод является эффективным и удобным для использования в образовательных учреждениях, что обеспечит более точный учет посещаемости и поможет избежать ошибок при подсчете.

1.4 Постановка задачи на разработку алгоритма учета посещаемости

Цель разработки линейного алгоритма в образовательной организации заключается в том, чтобы автоматизировать процесс учета посещаемости занятий студентами и преподавателями на основе данных системы контроля доступа (СКУД).

Для достижения этой цели необходимо сформулировать задачи:

1. Необходимо разработать алгоритм, который будет определять, находился ли студент/преподаватель в здании во время занятия, в соответствии с расписанием.
2. Необходимо разработать подпрограммы, которые будут собирать данные из системы контроля доступа и расписания учебных занятий.
3. Необходимо разработать подпрограмму, которая будет сравнивать данные о входах и выходах студентов/преподавателей из корпусов с данными о расписании занятий.
4. Программа должна генерировать отчет посещаемости студентов/преподавателей во время занятий.

Для создания такой системы потребуется использование современных технологий, включая языки программирования, такие как «C++», «C#», «Java», «Python», система управления базами данных, а также различные библиотеки для обработки и анализа данных.

Таким образом, разработка данного алгоритма необходима для обеспечения эффективного контроля за посещаемостью студентов и преподавателей в образовательной организации.

Вывод по первой главе

Можно сделать вывод, что система контроля и управления доступом является важным инструментом для обеспечения безопасности зданий и помещений образовательных организаций, а также может использоваться для учета посещаемости занятий и рабочего времени учащихся, что позволяет повысить эффективность управления учебным процессом и сократить затраты на административную работу.

При этом, для реализации алгоритма учета посещаемости занятий были рассмотрены существующие алгоритмы, которые помогут в разработке основного алгоритма. Так же были рассмотрены различные методы, включая использование биометрических данных и различных видов идентификации.

Так же были поставлены задачи на разработку алгоритма, что является важным шагом и позволяет определить основные требования и цели, которые должны быть достигнуты при разработке.

Глава 2 Разработка алгоритма учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом

2.1 Разработка алгоритма учета посещаемости занятий студентами и преподавателями

2.1.1 Описание алгоритма учета посещаемости занятий студентами и преподавателями

В данной главе будет показана подробная составляющая разрабатываемого алгоритма.

Разрабатываемый алгоритм должен обладать такими основные свойствами как:

- понятность: алгоритм должен быть понятным для исполнителя, который работает с ним [1];
- дискретность: алгоритм состоит из отдельных простых шагов или операций, которые выполняются последовательно [1];
- определенность: алгоритм должен быть определенным и точно определять каждый шаг выполнения. Определенность позволяет достичь одинакового результата при выполнении алгоритма в одинаковых условиях [1];
- массовость: алгоритм должен быть применим к различным наборам данных или ситуациям. Он должен быть общим и способным работать с разными входными данными [1];
- результативность: алгоритм должен быть эффективным и способным достичь своей цели или решить поставленную задачу. Результативность алгоритма оценивается по его способности давать корректные и ожидаемые результаты при выполнении [1].

Для данного алгоритма исполнитель будет являться программа, в которой будет реализовано его выполнение [1]. Среда для данного алгоритма

будет включать в себя компьютерную систему, оборудованную необходимым программным обеспечением [1]. В данном случае, это операционная система «Windows», с установленным языком программирования «Python», системой управления базами данных, средой разработки, а также дополнительными библиотеками и модулями, необходимыми для работы с СКУД и другими компонентами системы.

Среди множества языков программирования был выбран «Python». Так как он имеет простой синтаксис и обширные возможности для анализа и обработки данных. Кроме того, «Python» является интерпретируемым языком, что означает, что он не требует компиляции перед запуском программы и это упрощает процесс разработки. Не мало важным считается тот факт, что он имеет множество библиотек для работы с базами данных, делая его подходящим для разработки системы учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом [7].

Итак, алгоритм выглядит следующим образом. Он состоит из следующих этапов.

1. На первом собираются данные о расписании занятий с сайта учебного заведения.

2. На втором происходит сбор данных от системы контроля и управления доступом.

3. На третьем происходит сравнение данных о расписании занятий и данных о проходах от СКУД. Это позволяет определить, был ли студент или преподаватель в здании в соответствии с расписанием.

4. На четвертом шаге формируется отчет о текущей посещаемости студента/преподавателя.

2.1.2 Функциональное моделирование программы учета посещаемости занятий

Разрабатываемая программа, необходимая для осуществления учета посещаемости студентов и преподавателей, должна отвечать функциональным требованиям [5].

На рисунке 2 представлена функциональная модель разрабатываемой программы, на которой отображены основные функции.

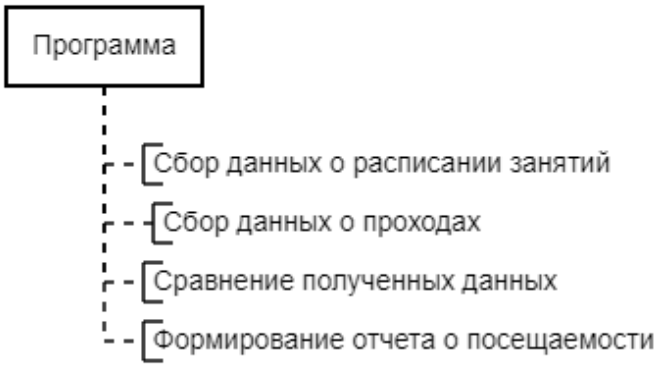


Рисунок 2 – Функциональная модель разрабатываемой программы

Основные функции разрабатываемой программы и краткое их содержание:

- сбор данных о расписании занятий – данная функция отвечает за сбор информации о расписании занятий и их времени проведения;
- сбор данных о проходах – данная функция отвечает за получение информации о посещении студентов и преподавателей занятий с помощью системы контроля и управления доступом;
- сравнение полученных данных – функция сравнение данных о расписании занятий и данных СКУД позволяет определить, был ли студент или преподаватель в здании вовремя занятии;
- формирование отчета о посещаемости – функция формирования отчетов отвечает за создание отчета о посещаемости занятий.

Таким образом, разработанный алгоритм будет позволять эффективно и надежно учитывать посещаемость студентами и преподавателями на основе данных, полученных от системы контроля и управления доступом, сравнивая их с расписанием занятий, собранным с сайта учебного заведения.

2.2 Выбор системы контроля и управления доступом для реализации программы по учету посещаемости

В предыдущей главе были рассмотрены различные способы учета посещаемости, а также методы, которые входят в систему контроля и управления доступом. Наиболее подходящим под поставленные задачи был выбран метод использования бесконтактных карт и теперь необходимо выбрать систему контроля и управления доступом под выбранный метод.

Существуют многие решения для данной задачи, а именно: «APACS 3000», «HID Global», «LEGIC Indensystems», «ARMO-SYSTEMS» и многие другие. Выбор пал именно на «APACS 3000», и это обосновано несколькими причинами:

- надежность и безопасность: «APACS 3000» является проверенной и надежной системой, которая имеет высокий уровень безопасности и защиты от несанкционированного доступа;
- гибкость и масштабируемость: «APACS 3000» обладает гибкими настройками и может быть легко адаптирована к различным потребностям и требованиям. Кроме того, система легко масштабируется и может быть расширена в зависимости от изменения потребностей [13];
- удобство использования: «APACS 3000» имеет интуитивно понятный интерфейс и легко интегрируется с другими системами. Также система имеет широкие возможности по анализу и обработке данных, что позволяет получать ценную информацию о посещаемости и деятельности сотрудников [13];

– тестовая версия: наличие тестовой версии «APACS 3000» с тестовой базой данных «Firebird» позволяет провести тщательное тестирование системы перед ее внедрением в работу. Это позволяет убедиться в правильности выбора СКУД и избежать ошибок в процессе внедрения.

Данная система контроля и управления доступом является одной из наиболее надежных и гибких систем для учета посещаемости, и важным фактором выбора данной системы является наличие тестовой версии, где можно будет взять данные о проходах для проведения полноценного тестирования алгоритма учета посещаемости.

Для работы с системой «APACS 3000», необходимо наличие сервера, который обеспечивает функционирование системы. В тестовой версии «APACS 3000» также предусмотрен демонстрационный режим работы сервера.

Сервер является центральным узлом системы и отвечает за управление и контроль доступа, а также за сбор и обработку данных о проходах сотрудников. Сервер обрабатывает запросы от контроллеров, которые подключены к считывателям, установленным на входах и выходах здания, и принимает решения о разрешении или запрете доступа на основе предварительно настроенных правил и прав доступа [2].

На рисунке 3 происходит подключение сервера для работы с «APACS 3000».

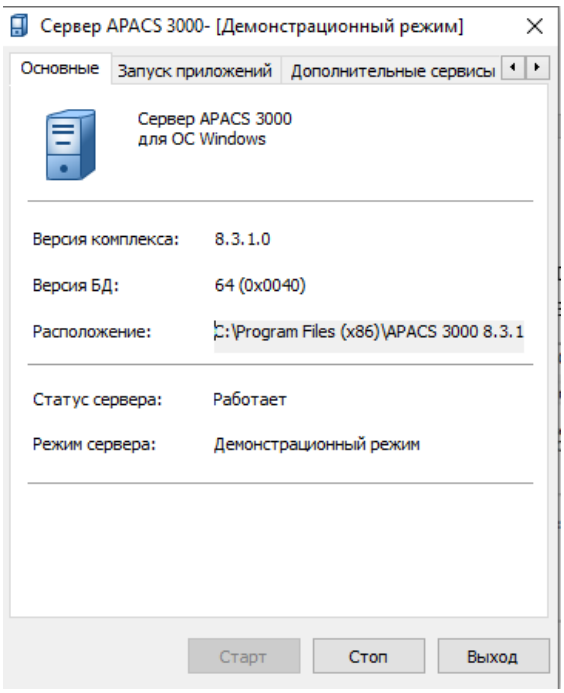


Рисунок 3 – Демонстрационный режим работы сервера «APACS 3000»

Рассмотрим основные функции, которые выполняет сервер:

– осуществление проверки прав доступа, аудит действий операторов и подтверждение сообщений, чтобы обеспечить безопасность и контроль доступа к системе [12];
– периодический опрос панелей, чтобы получать информацию о состоянии устройств, статусе событий и других данных, необходимых для работы системы [12];
– регистрация полученных сообщений от панелей, сохранение их в базе данных для дальнейшей обработки и анализа [12];
– выполнение команд, приходящие от клиентских приложений, таких как запросы на открытие дверей, управление режимами работы системы, управление пользователями и другие операции, которые могут быть выполнены на сервере [12].

Далее необходимо осуществить описание работы программного комплекса управления системами безопасности и контроля доступа.

Для демонстрации полного функционала системы необходимо осуществить авторизацию в системе под тестовым именем пользователя (рисунок 4).

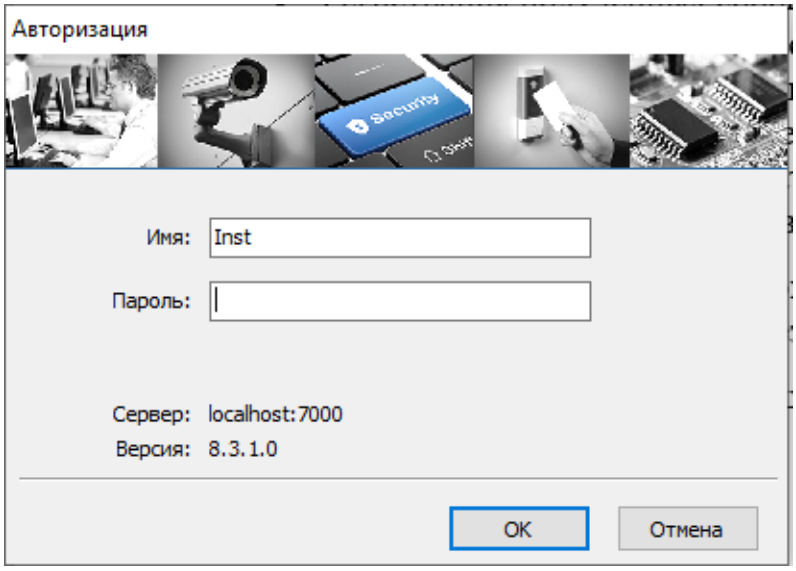


Рисунок 4 – Авторизация в системе «APACS 3000»

Модуль авторизации является встроенным в программное обеспечение «APACS 3000».

Далее необходимо осуществить переход в «Консоль + Картотека» для входа в клиентское приложение (рисунок 5).

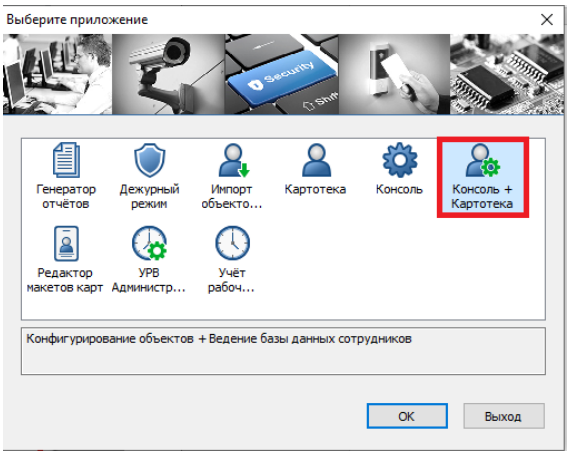


Рисунок 5 – Выбор приложения

Ниже представлено демонстрационное клиентское приложение с тестовой базой данных, которая работает в режиме симуляции (рисунок 6).

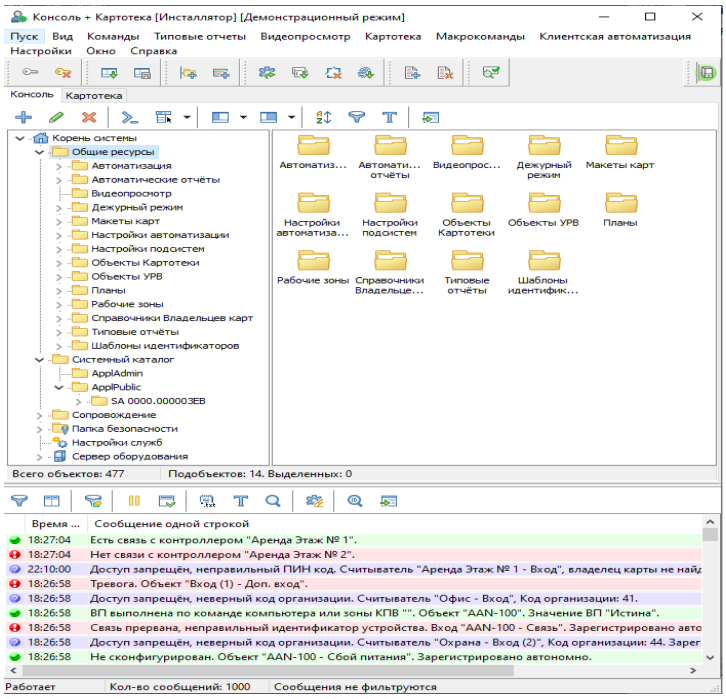


Рисунок 6 – Клиентское приложение «APACS 3000»

Во вкладке «Картотека» содержится тестовая база данных сотрудников. Эта картотека включает информацию о каждом сотруднике, такую как персональные данные, идентификационные данные (номер карты доступа), период активности его идентификационной карты и другие соответствующие данные (рисунок 7).

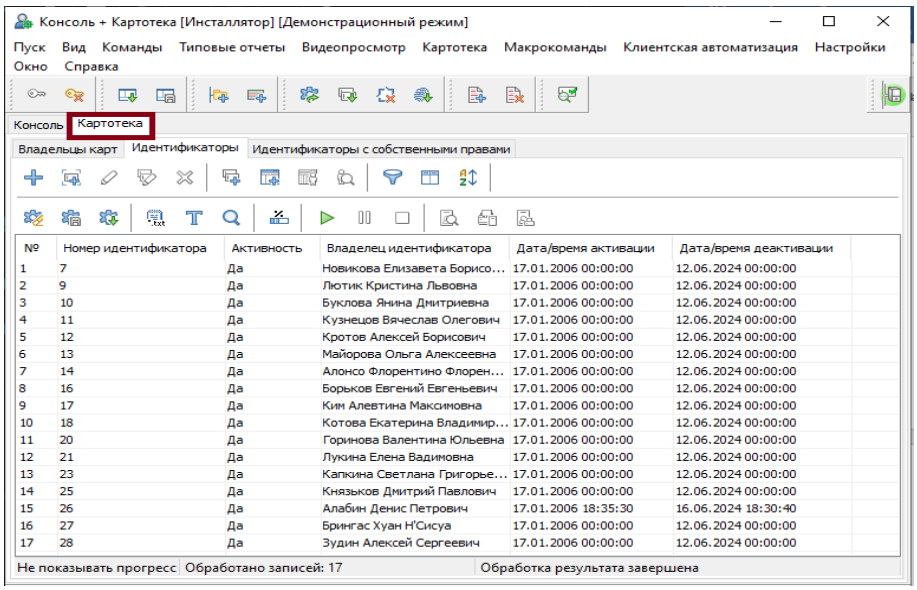


Рисунок 7 – Идентификационные данные каждого сотрудника

Для учета посещаемости сотрудников необходимо открыть вкладку «Консоль», далее выбрать папку «Общие ресурсы» и перейти в папку «Дежурный режим», после выбрать «Центральная проходная» (рисунок 8).

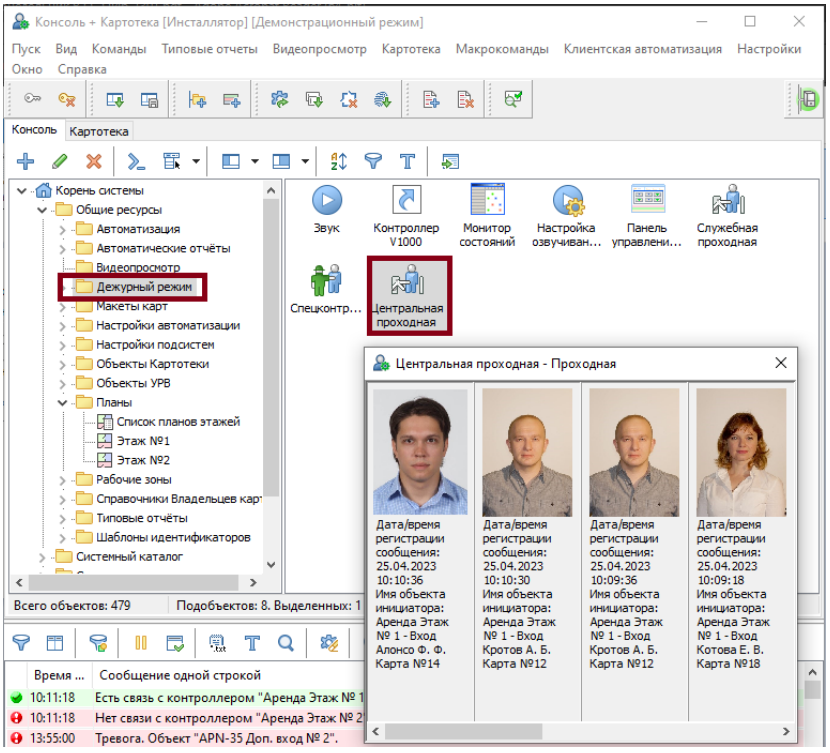


Рисунок 8 – Модуль «Центральная проходная – Проходная»

Модуль центральной проходной отвечает за фиксацию входа/выхода каждого сотрудника. Этот модуль предоставляет возможность регистрировать проход каждого сотрудника через считывающее устройство при входе и выходе из зоны контролируемого доступа.

Когда сотрудник проходит через считывающее устройство, модуль центральной проходной сервера получает данные о проходе, такие как номер карты доступа, дату и время прохода. После происходит запись данных в базу данных.

За работу контроллера «AAN-100» отвечает драйвер. Драйвер представляет собой программное обеспечение, предназначенное для работы с системой контроля и управления доступом. Он обеспечивает взаимодействие между считывателями, установленными на входах и выходах здания, и сервером системы, осуществляет считывание данных, передает данные на сервер, после чего происходит обработка команд от сервера и хранение данных о проходах (рисунок 9) [11].

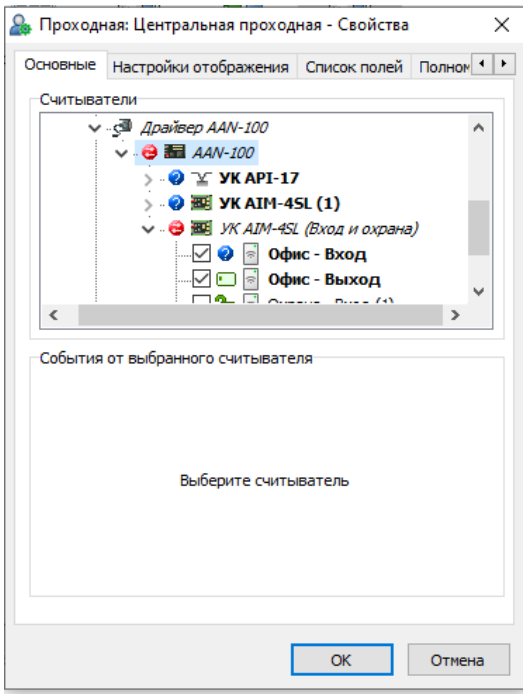


Рисунок 9 – Контроллер «AAN-100»

На основе изученных данных и информации о тестовой версии системы контроля и управления доступом «APACS 3000», можно сделать вывод о том, что эта система предоставляет функциональные возможности для контроля доступа и учета посещаемости в организациях. Драйвер, который обеспечивает взаимодействие с считывателями в проходной, играет важную роль в передаче данных с считывателей на сервер системы, обработке команд от сервера и хранении данных о проходах. В следующей под главой будет происходить сбор данных с базы данных «FireBird» о проходах, что позволит более детально рассмотреть анализ посещаемости.

2.3 Подготовка данных для реализации программы

2.3.1 Сбор данных о посещаемости занятий студентами и преподавателями

Для сбора данных о посещаемости занятий студентами и преподавателями, воспользуемся тестовой базой данных «FireBird» от системы контроля доступа «APACS 3000». Эта база данных содержит информацию о каждом проходе в здание, что делает ее подходящим источником данных для анализа посещаемости. Данная база данных соответствует нашему алгоритму учета посещаемости и позволит провести детальный анализ присутствия студентов и преподавателей на занятиях.

Для реализации сбора данных был разработан программный код на основе алгоритма связывания на языке «Python», который осуществляет взаимодействие с базой данных «FireBird» с использованием SQL-запросов. Программный код также включает определение параметров подключения с базой данных, таких как адрес сервера, имя базы данных, имя пользователя и пароль.

После успешного подключения к базе данных, код выполняет SQL-запрос, который извлекает требуемые данные о посещаемости в указанный период времени (рисунок 10) [16], [21].

```
cur.execute("""SELECT sc.FREALTIME, sc.FINITOBJNAME, cr.FHOLDERNAME
    FROM TAPCSYSEVENTSCOMMON sc
    INNER JOIN TAPCCARDHOLDERREF cr ON sc.FSEK1 = cr.FSEK1
    WHERE sc.FREALTIME >= ? AND sc.FREALTIME <= ? AND sc.FINITOBJNAME IN (?, ?)""",
    (datetime.datetime.combine(start_date, start_time), datetime.datetime.combine(end_date, end_time),
     "Офис - Вход", "Офис - Выход"))
```

Рисунок 10 – Выполнение SQL-запроса об данных прохода

Данные, связанные с событиями прохода, такими как время и информация о месте (вход или выход из здания), хранились в одной таблице в базе данных. Однако, для получения данных в удобном виде, необходимо было связать эти данные с информацией о фамилиях и инициалах, хранящейся в другой таблице. Это позволило получить полные данные о событиях прохода вместе с соответствующими фамилиями и инициалами в нормализованной форме. Далее приведен кусочек результат работы программного кода (рисунок 11).

	A	B	C
1	Время	Вход/Выход	Фамилия И.О.
2	02.01.2023 8:51	УЛК - Вход	Ворогов И. С.
3	02.01.2023 9:24	УЛК - Выход	Ворогов И. С.
4	02.01.2023 9:39	УЛК - Вход	Ворогов И. С.
5	02.01.2023 13:14	УЛК - Выход	Ворогов И. С.
6	02.01.2023 14:01	УЛК - Вход	Ворогов И. С.
7	02.01.2023 17:29	УЛК - Выход	Ворогов И. С.
8	02.01.2023 17:42	УЛК - Вход	Ворогов И. С.
9	02.01.2023 17:51	УЛК - Выход	Ворогов И. С.
10	03.01.2023 8:55	УЛК - Вход	Ворогов И. С.
11	03.01.2023 9:27	УЛК - Выход	Ворогов И. С.
12	03.01.2023 9:39	УЛК - Вход	Ворогов И. С.
13	03.01.2023 13:11	УЛК - Выход	Ворогов И. С.

Рисунок 11 – Данные о событии прохода

Таким образом, данный подход обеспечивает эффективный и удобный способ сбора и хранения данных о посещаемости для последующего использования в разрабатываемой программе.

2.3.2 Сбор данных о расписании занятий

Для эффективной организации учебного процесса необходимо иметь точную информацию о расписании занятий. Сбор и учет данных о расписании занятий является важной задачей, которая позволяет оптимизировать использование учебных аудиторий и ресурсов, а также предоставлять студентам и преподавателям актуальную информацию о расписании и изменениях в нем. Для решения задачи сбора данных о расписании занятий был разработан парсер на языке программирования «Python».

При разработке парсера были использованы современные технологии, такие как библиотеки для парсинга данных из «HTML» и других форматов, а также для обработки и анализа данных. Парсер автоматически обходит веб-страницу учебного заведения, извлекает необходимую информацию о расписании занятий, такую как даты, время, аудитории, предметы и другие детали, и сохраняет ее в удобном формате для дальнейшего использования [14].

Простое взятие информации о расписании занятий с учебного сайта оказалось непростой задачей из-за структуры и формата данных на веб-странице, а также ограничений доступа к сайту. Однако, с использованием библиотеки «Selenium» на языке программирования «Python», удалось успешно собрать информацию о расписании занятий.

«Selenium» – это популярный инструмент для автоматизации веб-браузера, который позволяет взаимодействовать с веб-страницами, выполнять действия, такие как клики, ввод данных, и получать информацию из элементов веб-страницы. Он широко используется для тестирования веб-приложений, автоматизации рутинных задач, и, как в данном случае, для сбора данных с веб-страницы, которая требует динамического взаимодействия с пользователем [8].

Для решения задачи сбора данных о расписании занятий с учебного сайта с использованием «Selenium», был разработан скрипт на языке «Python», который автоматически открывает браузер, переходит на веб-страницу с расписанием, выполняет необходимые действия для загрузки данных и затем извлекает нужную информацию из элементов веб-страницы. Полученная информация сохраняется в файл, чтобы избежать повторных запросов к сайту и снизить нагрузку на сервер.

Использование «Selenium» позволяет эффективно собирать информацию о расписании занятий с учебного сайта, обходя ограничения доступа и взаимодействуя с веб-страницей так, как это делает пользователь.

Библиотека «selenium» является основной библиотекой «Selenium» и содержит основные функции и методы для взаимодействия с веб-браузером. Она позволяет управлять браузером, выполнять различные действия на веб-странице, такие как клики, ввод данных, и извлечение информации из элементов страницы [4].

Библиотека «selenium.webdriver» содержит классы и методы, необходимые для настройки и управления веб-драйверами, которые являются основным инструментом для автоматизации браузера в «Selenium» [4].

Для начала работы с «Selenium», необходимо подключить соответствующие библиотеки в коде (рисунок 12).

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
```

Рисунок 12 – Подключение библиотек «Selenium»

После подключения необходимых библиотек, можно начать использовать функции и методы «Selenium» для автоматизации взаимодействия с веб-браузером и сбора данных с веб-страницы. Далее будут прилагаться кусочки программного кода и комментарии к ним.

Создаем экземпляр веб-драйвера, настраиваем его, открываем веб-страницу и извлекаем нужную информацию, в нашем случае данные о расписании занятий (рисунок 13).

```
def get_data_with_selenium(url):

    options = webdriver.ChromeOptions()
    options.add_argument("user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/111.0.0.0 Safari/537.36")
    options = Service("C:\\Users\\Никита\\PycharmProjects\\diplom\\chromedriver.exe")
    driver = webdriver.Chrome(service=options)
    try:
        driver.get(url=url)
        time.sleep(8)

        with open("index_selenium_group.html", "w", encoding="utf-8") as file:
            file.write(driver.page_source)

    except Exception as _ex:
        print(_ex)
    finally:
        driver.close()
        driver.quit()
```

Рисунок 13 – Разработанный скрипт для автоматизации взаимодействия с веб-страницей с использованием библиотеки «Selenium»

Разберем основные блоки данного скрипта:

– «options» – это объект, представляющий настройки и параметры для настройки веб-драйвера «webdriver.Chrome». В данном коде, «options» создается с использованием класса «webdriver.ChromeOptions» и позволяет настраивать различные аспекты работы с веб-драйвером, такие как указание пользовательского агента браузера [4];

– «try» – это блок, в котором выполняются операции, которые могут вызвать исключения (ошибки) во время выполнения программы. Если внутри блока «try» возникает исключение, то выполнение кода в блоке «try» прерывается, и код в блоке «except» выполняется [19];

– «finally» – это блок, который выполняется всегда, независимо от того, возникли ошибки в блоке «try» или нет. В данном коде, в блоке «finally» выполняются операции по закрытию веб-драйвера и освобождению ресурсов, даже если возникло исключение [19].

После использования «Selenium» для взаимодействия с веб-страницей расписания занятий и получения необходимых данных, необходимо использовать библиотеки, такие как «BeautifulSoup», «lxml» для парсинга (анализа и извлечения данных) полученной веб-страницы.

«BeautifulSoup» – это библиотека, предназначенная для облегчения парсинга «HTML» и «XML» документов. Она предоставляет удобные методы и функции для поиска, извлечения и манипуляции элементами «HTML» или «XML» документа. «BeautifulSoup» также автоматически выполняет разбор документа и представляет его в удобном виде для работы с данными [20].

«lxml» – это библиотека, которая предоставляет высокопроизводительный и гибкий инструментарий для парсинга и обработки «XML» и «HTML» документов в «Python». Она имеет мощные функции для поиска, извлечения и манипуляции элементами документов [3].

После успешного подключения библиотек «BeautifulSoup» и «lxml» приступаем к «парсингу» данных расписания занятий с веб-сайта.

Скрипт представленный на рисунке 14 открывает файл, указанный в переменной «file_path», который был сохранен после использования библиотеки «Selenium», с использованием кодировки «UTF-8». Считывает содержимое файла и передает его в объект «BeautifulSoup» с использованием парсера «lxml». Затем производится поиск элемента с классом «schedule-table» и находятся все элементы с классом «date-cell» внутри этого элемента. Текстовые данные (дни недели) из этих элементов извлекаются и сохраняются в отдельных переменных «Monday», «Tuesday», «Wednesday», «Thursday», «Friday» и «Saturday».

```
with open(file_path, encoding="utf-8") as file:
    src = file.read()
soup = BeautifulSoup(src, "lxml")

# Собираем дни недели

info_data = soup.find(class_="schedule-table").find_all(class_="date-cell")
monday = info_data[0].text
tuesday = info_data[1].text
wednesday = info_data[2].text
thursday = info_data[3].text
friday = info_data[4].text
saturday = info_data[5].text

with open(f"data/GroupName.csv", "w", newline="") as file:
    writer = csv.writer(file, delimiter=';')
    writer.writerow([monday, tuesday, wednesday, thursday, friday, saturday])
```

Рисунок 14 – Скрипт извлечения текстового содержимого о днях недели

В коде на рисунке 15, «soup» представляет объект «BeautifulSoup», созданный на основе ранее загруженного и спарсенного HTML-кода в переменной «info_data_group» с использованием библиотеки «BeautifulSoup». Затем, в цикле происходит извлечение текстовых данных из элементов «HTML» с классом «schedule-cell» с помощью метода «find_all», и эти данные сохраняются в соответствующие списки «row0», «row1», «row2» и «row3». Затем, данные из этих списков записываются в CSV-файл с использованием библиотеки «csv» и метода «writerows».

```python
  #Собираем информацию по каждому дню недели
info_data_group = soup.find(class_="schedule-table")
rows=[]
rows1=[]
rows2=[]
rows3=[]
for item in info_data_group:
    rasp_tlt = item.find_all(class_="schedule-cell")
    row0 = [rasp_tlt[0].text, rasp_tlt[1].text, rasp_tlt[2].text, rasp_tlt[3].text, rasp_tlt[4].text,rasp_tlt[5].text]
    row1 = [rasp_tlt[6].text, rasp_tlt[7].text, rasp_tlt[8].text, rasp_tlt[9].text, rasp_tlt[10].text,rasp_tlt[11].text]
    row2 = [rasp_tlt[12].text, rasp_tlt[13].text, rasp_tlt[14].text, rasp_tlt[15].text, rasp_tlt[16].text,rasp_tlt[17].text]
    row3 = [rasp_tlt[18].text, rasp_tlt[19].text, rasp_tlt[20].text, rasp_tlt[21].text, rasp_tlt[22].text,rasp_tlt[23].text]
    rows.append(row0)
    rows1.append(row1)
    rows2.append(row2)
    rows3.append(row3)
with open(f"data/GroupName.csv", "a", newline="") as file:
    writer = csv.writer(file, delimiter=";")
    writer.writerows((rows))
    writer.writerows((rows1))
    writer.writerows((rows2))
    writer.writerows((rows3))
```

Рисунок 15 – Скрипт извлечения информации по каждому дню недели

Результат работы функций и информация, спарсенная с веб-страницы, была сохранена в файле. В этом файле содержится расписание занятий на определенную неделю, разбитое по дням недели и временным интервалам (рисунок 16).

ПН 3	ВТ 4	СР 5	ЧТ 6	ПТ 7	СБ 8
Нет занятий	Нет занятий	Преддипломная практика/Производственная практика (преддипломная практика) (Иные формы)ПМИб_1902бДП[08:30-10:00]	Системы искусственного интеллекта 2 (Пр)ПМИб_1902б Тонких А.П.УЛК-314[08:30-10:00]	Нет занятий	Нет занятий
Нет занятий	Информационная безопасность (Пр)ПМИб_1902бРаченко Т.А.УЛК-408[10:15-11:45]	Преддипломная практика/Производственная практика (преддипломная практика) (Иные формы)ПМИб_1902бДП[10:15-11:45]	Дополнительные главы прикладной математики и информатики (Пр)ПМИб_1902б Тырыгина Г.А.УЛК-702[10:15-11:45]	Нет занятий	Нет занятий
Нет занятий	Информационная безопасность (Лек)ПМИб_1902аПМИб_1902бПМИб_1902вРаченко Т.А.УЛК-418[12:45-14:15]	Нет занятий	Системы искусственного интеллекта 2 (Лек)ПМИб_1902аПМИб_1902бПМИб_1902вКлимов В.С.УЛК-418[12:45-14:15]	Нет занятий	Вычислительный эксперимент 2 (Пр)ПМИб_1902бТырыгина Г.А.УЛК-314[12:45-14:15]
Нет занятий	Нет занятий	Нет занятий	Нет занятий	Нет занятий	Модели принятия решений (Пр)ПМИб_1902бТырыгина Г.А.УЛК-305[14:30-16:00]

Рисунок 16 – Результат работы функций представлен в Excel таблице

Таким образом, разработанный парсер для сбора данных о расписании занятий представляет собой мощный инструмент, который может быть

полезен для разрабатываемой программы. Этот парсер позволяет автоматически собирать данные о расписании занятий, такие как даты, время, место проведения и группа, из источника данных, такого как веб-страница.

С использованием разработанного парсера, разрабатываемая программа может получать актуальную информацию о расписании занятий без необходимости вручную собирать и вводить данные. Это позволяет снизить трудозатраты на обновление расписания и минимизировать возможность ошибок в данных.

2.4 Реализация алгоритма по учету посещаемости студентов и преподавателей

Реализация алгоритма по учету посещаемости студентов и преподавателей на основе контроля и управления доступом будет реализована в программном коде.

Перед тем, как приступать к реализации программы, нужно составить блок-схему алгоритма, которая описывает последовательность выполнения действий в программе (рисунок 17).

Рисунок 17 – Блок-схема алгоритма программы

Важно разделить алгоритм на несколько этапов, чтобы упростить задачу и избежать ошибок. Каждый этап должен быть реализован отдельно и протестирован перед переходом к следующему. Это позволит убедиться в корректности работы каждой части алгоритма и облегчит отладку, если что-то пойдет не так. Кроме того, разделение алгоритма на этапы облегчит понимание его работы и позволит сосредоточиться на каждом этапе по отдельности, что улучшит качество кода и ускорит процесс разработки:

– этап первый подготовительный: в рамках данного этапа необходимо произвести извлечение полученных данных о расписании занятий и данные о проходах, а также отобрать признаки для анализа;
– этап второй реализация: в рамках текущего этапа необходимо реализовать алгоритм сравнения, который будет осуществлять учет посещаемости студентов и преподавателей, на основе

предоставленных данных. Важно убедиться, что алгоритм будет правильно обрабатывать все возможные варианты посещения, чтобы избежать ошибок при подсчете;

– этап третий вывод: в рамках текущего этапа формируется отчет о посещаемости студента/преподавателя на основе работы алгоритма сравнения.

В рамках первого этапа необходимо взять входные данные, которые были получены при реализации программ из предыдущих под глав.

Данные о расписании были получены из парсинга данных с веб-страницы учебного учреждения, содержат информацию о предметах, датах, времени, местах проведения. Данные о проходах были взяты из тестовой базы данных «FireBird», содержащее информацию о фиксации входа и выхода из здания с использованием бесконтактных карт. Эти данные включают в себя информацию о дате, времени, ФИО и событии прохода (вход/выход).

Перед использованием данных необходимо произвести отбор признаков, которые будут использоваться в подпрограмме сравнения. Из данных о расписания следует выбрать признаки, такие как дата, время и место проведения. В данных о проходах необходимо отобрать признаки, такие как дата, время, событие прохода и место проведения. Этот отбор позволит упростить работу с данными и сократить объем анализируемой информации.

В рамках второго этапа необходимо реализовать алгоритм сравнения. Для сравнения посещаемости выбираются главные признаки, такие как дата, время и место проведения. Сначала производится проверка даты. Если дата сходится, то происходит сравнение места проведения занятия с локацией прохода. Если место проведения сходится, то производится проверка временного промежутка занятия с временем прохода студента или преподавателя. Если время прохода входит в промежуток занятия, то это означает, что он присутствовал в здании во время занятия. Если же время прохода не совпадает с промежутком занятия, то это означает отсутствие студента или преподавателя в здании во время занятия. Также важно

учитывать, если студент или преподаватель зашел до начала занятия и не выходил во время занятия, это также означает их присутствие. Такой алгоритм сравнения помогает правильно обработать данные и избежать ошибок при подсчете посещаемости.

Рассмотрим на рисунке 18 блок-схему подпрограммы сравнения данных.

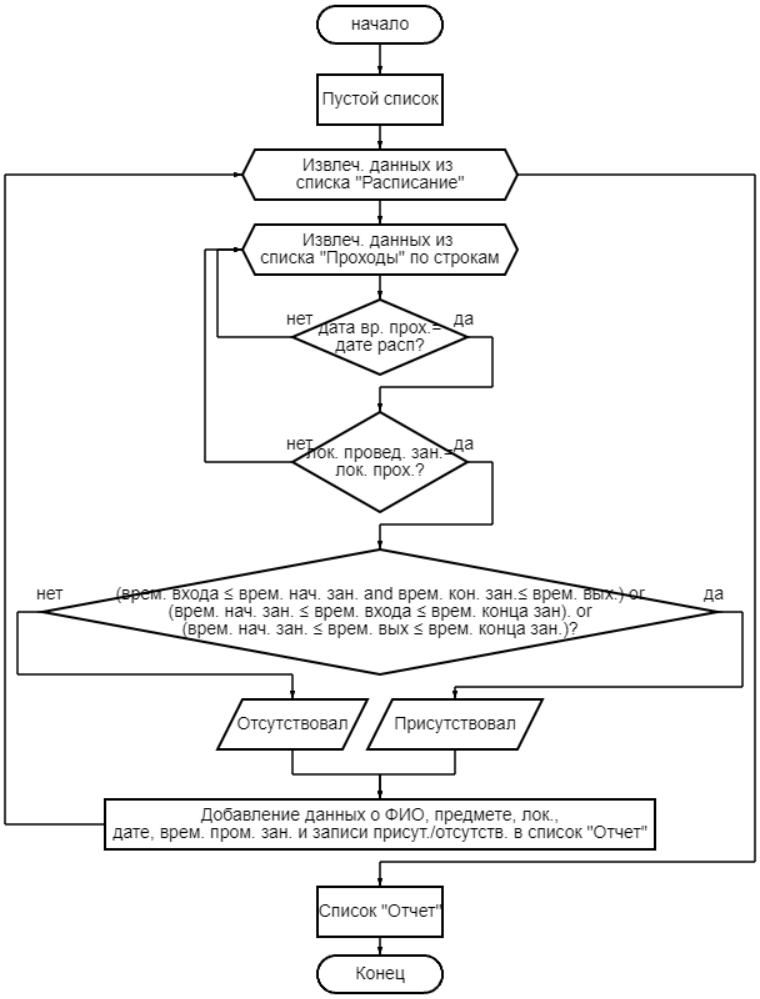


Рисунок 18 – Блок-схема подпрограммы сравнения данных

В рамках третьего этапа происходит формирование отчета о присутствии/отсутствии студента или преподавателя в здании во время занятия на основе работы алгоритма сравнения, реализованного на втором этапе.

Рассмотрим архитектуру программы, которая включает в себя несколько функций, каждая из которых отвечает за свой этап обработки данных.

На рисунке 19 происходит импорт необходимых модулей и библиотек, таких как «csv», «datetime», «collections», «re» и «locale». А также описание функций, предназначенных для работы алгоритма.

Функция «parse_schedule_date» преобразует строку с датой из расписания в формат «datetime».

Функция «parse_time» преобразует строку со временем из расписания в формат «time».

Функция «parse_date_time» преобразует строку с датой и временем в формат «datetime».

```
import csv
import datetime as dt
from collections import defaultdict

def parse_schedule_date(date_str):
    return dt.datetime.strptime(date_str, "%d.%m.%Y")

def parse_time(time_str):
    return dt.datetime.strptime(time_str, "%H:%M").time()

def parse_date_time(datetime_str):
    return dt.datetime.strptime(datetime_str, "%d.%m.%Y %H:%M")
```

Рисунок 19 – Функции преобразования

Функция «process_schedule» считывает файл с расписанием, парсит его и возвращает список кортежей с датой, временным интервалом и названием предмета (рисунок 20).

```python
def process_schedule(schedule_file):
    schedule = []
    with open(schedule_file, encoding="cp1251") as f:
        reader = csv.reader(f, delimiter=";")
        dates = next(reader)
        for row in reader:
            for date_str, cell in zip(dates, row):
                if "Нет занятий" not in cell:
                    date = parse_schedule_date(date_str)
                    time_range = [parse_time(t) for t in cell.split("[")[-1].strip("]").split("-")]
                    subject = cell.split("(")[0].strip()

                    # Извлечение местоположения
                    location_match = re.search(r"(\w+)-\d+", cell)
                    if location_match:
                        location = location_match.group(1)
                    else:
                        location = "Неизвестно"

                    schedule.append((date, time_range, subject, location))
    return schedule
```

Рисунок 20 – Функция «process_schedule»

Функция «process_entry_exit» считывает файл с данными о входе и выходе студентов и преподавателей, парсит его и возвращает словарь, где ключом является имя студента/преподавателя, а значением – список кортежей с датой/временем и действием (вход или выход) (рисунок 21).

```python
def process_entry_exit(entry_exit_file):
    entry_exit_data = defaultdict(list)
    with open(entry_exit_file, encoding="cp1251") as f:
        reader = csv.reader(f, delimiter=";")
        next(reader)
        for row in reader:
            timestamp, action, name = row
            entry_exit_data[name].append((parse_date_time(timestamp), action))
    return entry_exit_data
```

Рисунок 21 – Функция «process_entry_exit»

Функция «check_attendance» сравнивает данные о посещении студентов/преподавателей с данными о расписании и формирует отчет, содержащий информацию о присутствии/отсутствии на каждом занятии (рисунок 22).

```python
def check_attendance(schedule, entry_exit_data):
    report = []
    # Установка локали на русскую
    locale.setlocale(locale.LC_TIME, "ru_RU")
    for date, time_range, subject, location in schedule:
        for name, timestamps in entry_exit_data.items():
            attended = False
            entered = None
            for timestamp, action in timestamps:
                if timestamp.date() == date.date():
                    if action.startswith(location + " - Вход"):
                        entered = timestamp
                    elif entered is not None and action.startswith(location + " - Выход"):
                        if (entered.time() <= time_range[0]) or \
                           (time_range[0] <= entered.time() <= time_range[1] or time_range[0] <= timestamp.time() <= time_range[1]):
                            attended = True
                            break
                        entered = None

            # Форматирование даты в формате "d.m.Y"
            formatted_date = date.strftime("%d.%m.%Y")
            report.append((name, subject + " " + location, formatted_date, f"{time_range[0]}-{time_range[1]}", "Присутствовал" if attended else "Отсутствовал"))
    return report
```

Рисунок 22 – Функция «check_attendance»

Главная функция «main» объединяет все функции воедино и позволяет пользователю ввести фамилию студента/преподавателя для получения отчета о его посещаемости (рисунок 23).

```python
def main():
    schedule = process_schedule("Расписание.csv")
    entry_exit_data = process_entry_exit("Вход_Выход.csv")

    # Ввод фамилии с клавиатуры
    name = input("Введите фамилию студента: ")

    # Проверка наличия фамилии в данных входа/выхода
    found_name = None
    for entry_exit_name in entry_exit_data.keys():
        if name.lower() in entry_exit_name.lower():
            found_name = entry_exit_name
            break

    if found_name is not None:
        attendance_report = check_attendance(schedule, {found_name: entry_exit_data[found_name]})
        with open("Отчет.csv", "w", encoding="cp1251", newline="") as f:
            writer = csv.writer(f, delimiter=";")
            writer.writerow(["ФИО", "Предмет", "Дата", "Время", "Присутствие/Отсутствие"])
            writer.writerows(attendance_report)
    else:
        print("Фамилия не найдена в данных входа/выхода.")

if __name__ == "__main__":
    main()
```

Рисунок 23 – Главная функция main

Теперь перейдем к модели программы, где подробно будут рассмотрены функции, реализованные в коде:

Функция «parse_schedule_date(date_str)» – принимает строку «date_str» и возвращает дату в формате «datetime», полученную из строки с помощью метода «strptime».

Функция «parse_time(time_str)» – принимает строку «time_str» и возвращает время в формате «time», полученное из строки с помощью метода «strptime».

Функция «parse_date_time(datetime_str)» – принимает строку «datetime_str» и возвращает дату и время в формате «datetime», полученные из строки с помощью метода «strptime».

Функция «process_schedule(schedule_file)» – принимает имя файла «schedule_file», считывает расписание занятий из CSV-файла, обрабатывает его и возвращает список кортежей, каждый из которых содержит дату занятия, время начала и окончания занятия, название предмета и местоположение занятия.

Функция «process_entry_exit(entry_exit_file)» – принимает имя файла «entry_exit_file», считывает данные о входах и выходах студентов и преподавателей из CSV-файла, обрабатывает их и возвращает словарь, в котором ключами являются имена студентов и преподавателей, а значениями – список кортежей, каждый из которых содержит дату и время входа или выхода и тип действия (вход или выход).

Функция «check_attendance(schedule, entry_exit_data)» – принимает расписание занятий и данные о входах и выходах студентов и преподавателей, обрабатывает их и возвращает список кортежей, каждый из которых содержит имя студента или преподавателя, название предмета и местоположение занятия, дату занятия, время начала и окончания занятия и статус посещения занятия (присутствовал или отсутствовал).

Функция «main» – основная функция программы, в которой вызываются все остальные функции. Сначала вызываются функции «process_schedule» и «process_entry_exit» для получения расписания занятий и данных о входах и выходах студентов и преподавателей. Затем пользователю предлагается

ввести фамилию студента, после чего происходит поиск соответствующей фамилии в данных входа/выхода. Если фамилия найдена, вызывается функция «check_attendance» для проверки посещения занятий и создания отчета в CSV-файле "Отчет.csv". Если фамилия не найдена, выводится соответствующее сообщение.

Таким образом, архитектура программы представляет собой модульную структуру, состоящую из отдельных функций, которые обрабатывают данные, и создают отчет о посещаемости занятий студентами и преподавателями.

Модель программы использует функциональное программирование и включает в себя функции для разбора даты и времени, обработки расписания занятий и данных о входах и выходах, а также проверки посещаемости.

Вывод по второй главе

В данной главе был представлен детальный обзор процесса разработки программы по учету посещаемости студентов и преподавателей. Был описан алгоритм учета посещаемости, а также было произведено функциональное моделирование. Была выбрана система контроля и управления доступом для тестирования разработанного алгоритма. Важным этапом в разработке программы была подготовка данных для ее реализации, которая включала сбор данных расписания с сайта учебного заведения и сбор данных о проходах студентов/преподавателей на основе тестовой базы данных FireBird. И наконец, была представлена архитектура и модель программы, которая была разработана на основе собранных данных. В целом, данная глава дает полное представление о процессе разработки программного продукта, начиная от описания алгоритма и заканчивая реализацией программы.

Глава 3 Тестирование алгоритма учета посещаемости занятий студентами и преподавателями на основе событий системы контроля и управления доступом

3.1 Описание реализованной программы по учету посещаемости студентов и преподавателей

Результатом выполнения данной выпускной квалификационной работы были изучены основные методы учета посещаемости и рассмотрена система контроля и управления доступом APACS 3000.

Также был реализован алгоритм учета посещаемости занятий студентами и преподавателями в программном коде.

Основной функционал реализованной программы предоставляется следующие возможности:

- парсинг данных о расписания занятий и о входе/выходе студентов и преподавателей;
- проверка посещаемости студентов и преподавателей на основе спарсенных данных;
- возможность подавать запрос по конкретному студенту/преподавателю;
- возможность работы с более обширными базами данных.

Для запуска программы необходимо ввести в терминале название файла, содержащий код на языке Python.

Данный файл запускает без передачи каких-либо ключей. Команду запуска можно увидеть на рисунке 24.

Так же на рисунке 24 можно увидеть, что после запуска программы предоставляется ввести фамилию студента/преподавателя, что позволяет получить отчет о его посещаемости занятий в соответствии с данными о проходах. Программа проверяет наличие введенной фамилии в данных входа/выхода, и если она найдена, то генерируется отчет в формате csv. Если

введенная пользователем фамилия не будет найдена, то программа выведет сообщение "Фамилия не найдена в данных входа/выхода". Это помогает пользователю понять, что возможно была допущена ошибка при вводе фамилии, либо данные о проходах для этой фамилии отсутствуют в системе.

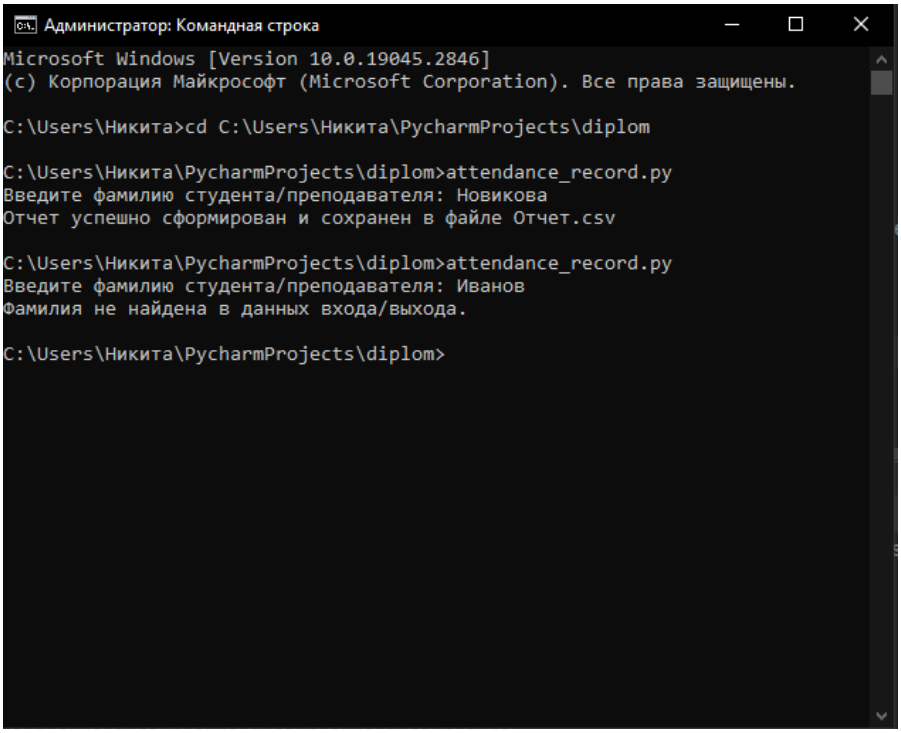
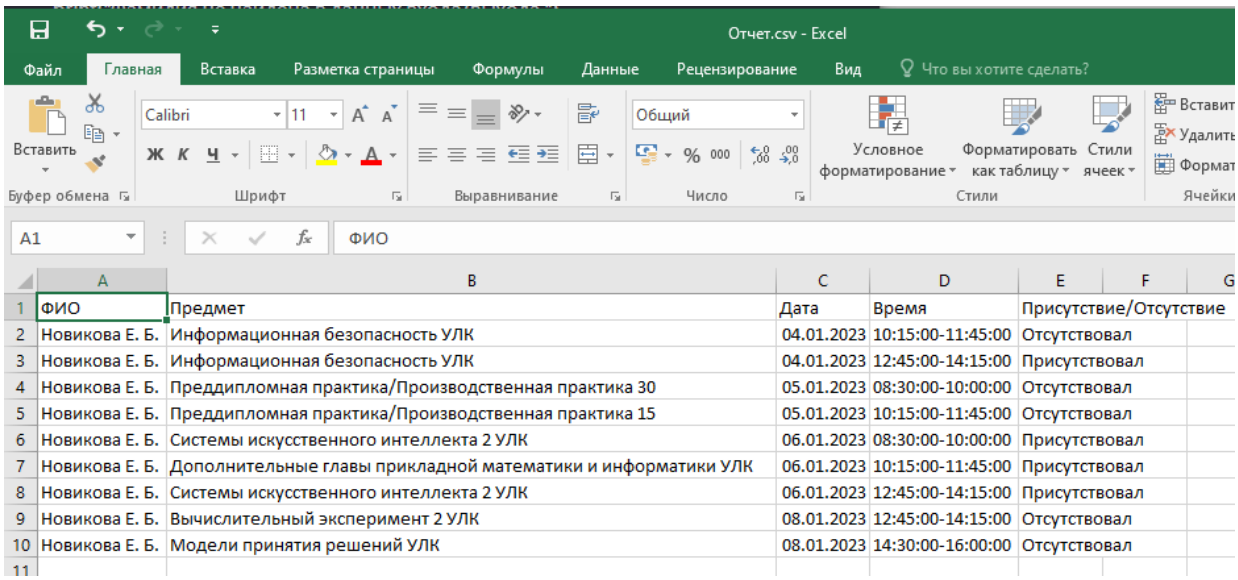


Рисунок 24 – Запуск программы, ее отработка и завершение

Рассмотрим отчет, который сформировала программа. Отчет представляет собой csv-файл с информацией о фамилии, предмете, дате, времени и статуса посещения (присутствовал/отсутствовал) на каждом занятии из расписания. Для просмотра отчета можно открыть файл в любом текстовом редакторе или программе для работы с таблицами, в данном случае файл будет открыт в Excel (рисунок 25).



	A	B	C	D	E	F	G
1	ФИО	Предмет	Дата	Время	Присутствие/Отсутствие		
2	Новикова Е. Б.	Информационная безопасность УЛК	04.01.2023	10:15:00-11:45:00	Отсутствовал		
3	Новикова Е. Б.	Информационная безопасность УЛК	04.01.2023	12:45:00-14:15:00	Присутствовал		
4	Новикова Е. Б.	Преддипломная практика/Производственная практика 30	05.01.2023	08:30:00-10:00:00	Отсутствовал		
5	Новикова Е. Б.	Преддипломная практика/Производственная практика 15	05.01.2023	10:15:00-11:45:00	Отсутствовал		
6	Новикова Е. Б.	Системы искусственного интеллекта 2 УЛК	06.01.2023	08:30:00-10:00:00	Присутствовал		
7	Новикова Е. Б.	Дополнительные главы прикладной математики и информатики УЛК	06.01.2023	10:15:00-11:45:00	Присутствовал		
8	Новикова Е. Б.	Системы искусственного интеллекта 2 УЛК	06.01.2023	12:45:00-14:15:00	Присутствовал		
9	Новикова Е. Б.	Вычислительный эксперимент 2 УЛК	08.01.2023	12:45:00-14:15:00	Отсутствовал		
10	Новикова Е. Б.	Модели принятия решений УЛК	08.01.2023	14:30:00-16:00:00	Отсутствовал		
11							

Рисунок 25 – Содержимое файла Отчет.csv

3.2 Тестирование реализованной программы по учету посещаемости студентов и преподавателей

Теперь необходимо протестировать программу на соответствие требованиям и корректности работы. В качестве тестирования рассмотрим все доступные примеры, чтобы показать, что данная программа работает без нареканий и происходит правильный учет посещаемости занятий студентов и преподавателей.

Рассмотрим первый пример, в котором возьмем данные "Новикова" из таблицы "Вход_Выход" и данные из таблицы "Расписание" за 04.01.2023 число и сравним их с данными "Отчет" (рисунки 26-28).

	A	B	C	D	E
1	ФИО	Предмет	Дата	Время	Присутствие/Отсутствие
2	Новикова Е. Б.	Информационная безопасность УЛК	04.01.2023	10:15:00-11:45:00	Отсутствовал
3	Новикова Е. Б.	Информационная безопасность УЛК	04.01.2023	12:45:00-14:15:00	Присутствовал
4	Новикова Е. Б.	Преддипломная практика/Производственная практика 30	05.01.2023	08:30:00-10:00:00	Отсутствовал
5	Новикова Е. Б.	Преддипломная практика/Производственная практика 15	05.01.2023	10:15:00-11:45:00	Отсутствовал
6	Новикова Е. Б.	Системы искусственного интеллекта 2 УЛК	06.01.2023	08:30:00-10:00:00	Присутствовал
7	Новикова Е. Б.	Дополнительные главы прикладной математики и информатики УЛК	06.01.2023	10:15:00-11:45:00	Присутствовал
8	Новикова Е. Б.	Системы искусственного интеллекта 2 УЛК	06.01.2023	12:45:00-14:15:00	Присутствовал
9	Новикова Е. Б.	Вычислительный эксперимент 2 УЛК	08.01.2023	12:45:00-14:15:00	Отсутствовал
10	Новикова Е. Б.	Модели принятия решений УЛК	08.01.2023	14:30:00-16:00:00	Отсутствовал

Рисунок 26 – Отчет для Новикова Е.Б.

04.01.2023
Нет занятий
Информационная безопасность (Пр)ПМИб_1902бРаченко Т.А.УЛК-408[10:15-11:45]
Информационная безопасность (Лек)ПМИб_1902аПМИб_1902бПМИб_1902вРаченко Т.А.УЛК-418[12:45-14:15]
Нет занятий

Рисунок 27 – Данные из таблицы Расписания

237	03.01.2023 17:54	УЛК - Выход	Новикова Е. Б.
238	04.01.2023 13:50	УЛК - Вход	Новикова Е. Б.
239	04.01.2023 15:30	УЛК - Выход	Новикова Е. Б.
240	04.01.2023 15:44	УЛК - Вход	Новикова Е. Б.
241	04.01.2023 18:14	УЛК - Выход	Новикова Е. Б.
242	05.01.2023 9:09	УЛК - Вход	Новикова Е. Б.
243	05.01.2023 11:15	УЛК - Выход	Новикова Е. Б.

Рисунок 28 – Данные из таблицы Вход_Выход

В данный день было 2 занятия, а именно в 10:15-11:45 и в 12:45-14:15. Вход в УЛК был в 13:50, а это означает, что Новикова не находилась в здании во время проведения занятия и пропустила его в 10:15-11:45. В отчете ей был присвоен статус "Отсутствовал", так как время ее входа не соответствовало времени первого занятия. Но во время второго занятия она находилась в здании и ей был присвоен статус "Присутствовал", так как ее время входа соответствовало времени второго занятия.

Рассмотрим второй пример, в котором возьмем данные за 06.01.2023 число и сравним их с данными "Отчет" (рисунки 29-30).

D
06.01.2023
Системы искусственного интеллекта 2 (Пр)ПМИб_1902бТонких А.П.УЛК-314[08:30-10:00]
Дополнительные главы прикладной математики и информатики (Пр)ПМИб_1902бТырыгина Г.А.УЛК-702[10:15-11:45]
Системы искусственного интеллекта 2 (Лек)ПМИб_1902аПМИб_1902бПМИб_1902вКлимов В.С.УЛК-418[12:45-14:15]
Нет занятий

Рисунок 29 – Данные из таблицы Расписания

249	05.01.2023 18:11	УЛК - Выход	Новикова Е. Б.
250	06.01.2023 8:56	УЛК - Вход	Новикова Е. Б.
251	06.01.2023 11:54	УЛК - Выход	Новикова Е. Б.
252	06.01.2023 12:01	УЛК - Вход	Новикова Е. Б.
253	06.01.2023 12:51	УЛК - Выход	Новикова Е. Б.
254	06.01.2023 14:01	УЛК - Вход	Новикова Е. Б.
255	06.01.2023 16:56	УЛК - Выход	Новикова Е. Б.
256	06.01.2023 16:57	УЛК - Вход	Новикова Е. Б.

Рисунок 30 – Данные из таблицы Вход_Выход

В данном примере на 06.01.2023 года было 3 занятия: первое занятие с 8:30 до 10:00, второе с 10:15 до 11:45 и третье с 12:45 до 14:15. Новикова пришла в УЛК в 8:56, что означает, что она присутствовала в здании во время первого занятии. В отчете ей присвоен статус "Присутствовал".

Теперь рассмотрим второе занятие. Оно начинается в 10:15 и заканчивается в 11:45. Статус в отчете также "Присутствовал", так как в программе есть условие, при котором, если время входа в здание меньше чем начало занятия и во время занятия не выходил, то студент/преподаватель считается присутствовавшим в здании.

Рассмотрим третий пример, в котором возьмем данные за 08.01.2023 число (рисунки 31-32).

08.01.2023
Нет занятий
Нет занятий
Вычислительный эксперимент 2 (Пр)ПМИб_1902бТ ырыгина Г.А.УЛК-314[12:45-14:15]
Модели принятия решений (Пр)ПМИб_1902бТ ырыгина Г.А.УЛК-305[14:30-16:00]

Рисунок 31 – Данные из таблицы Расписания

257	06.01.2023 18:14	УЛК - Выход	Новикова Е. Б.
258	10.01.2023 8:53	УЛК - Вход	Новикова Е. Б.
259	10.01.2023 12:05	УЛК - Выход	Новикова Е. Б.
260	10.01.2023 12:06	УЛК - Вход	Новикова Е. Б.
261	10.01.2023 13:02	УЛК - Выход	Новикова Е. Б.
262	10.01.2023 14:05	УЛК - Вход	Новикова Е. Б.
263	10.01.2023 14:17	УЛК - Выход	Новикова Е. Б.
264	10.01.2023 14:18	УЛК - Вход	Новикова Е. Б.

Рисунок 32 – Данные из таблицы Вход_Выход

В данном примере на 08.01.2023 года было 2 занятия: первое занятие с 12:45 до 14:15, второе с 14:30 до 16:00. Однако, по данным входа/выхода видно, что Новикова не посещала занятия на данное число. В отчете на 08.01.2023 ей присвоен статус "Отсутствовал" на двух занятиях, так как программа проверяет соответствие дат в двух таблицах.

Рассмотрим преимущества и недостатки разработанного алгоритма.

Преимущества разработанного алгоритма:

– алгоритм автоматически обрабатывает данные о расписании занятий и проходах студентов/преподавателей, что значительно экономит время на ручной обработке данных;
– алгоритм учитывает местоположение проведения занятий, что позволяет более точно определять посещаемость студентов/преподавателей;
– алгоритм выводит отчет по конкретному студенту/преподавателю на предмет присутствия/отсутствия в здании во времени каждого занятия, что позволяет быстро выявлять проблемы с посещаемостью и принимать соответствующие меры.

Недостатки разработанного алгоритма:

– алгоритм не учитывает возможные ошибки при регистрации входа/выхода студентов/преподавателей;
– алгоритм может работать некорректно в случае, если формат входных данных не соответствует ожидаемому, что может привести к ошибкам в обработке данных.

После тестирования можно сделать вывод, что программа успешно парсит данные о расписании занятий и о входе/выходе студентов и преподавателей, и на основе полученной информации проводит проверку посещаемости. Кроме того, программа предоставляет возможность получения отчета по конкретному студенту/преподавателю, что обеспечивает удобство в работе с программой. Результаты тестирования подтверждают, что реализованная программа по учету посещаемости является эффективным инструментом для автоматизации процесса учета и контроля посещаемости занятий в учебном заведении.

Вывод по третьей главе

В данной главе была представлена реализованная программа по учету посещаемости студентов и преподавателей, а также описание ее основного функционала. В результате тестирования программы было установлено, что она работает стабильно и выдает правильные значения.

Заключение

Итогом выпускной квалификационной работы является разработанный алгоритм, предназначенный для учета посещаемости студентов и преподавателей на основе системы контроля и управления доступом.

В рамках работы был проведен анализ существующих алгоритмов, обзор системы контроля и управления доступом, включая особенности сбора данных о посещаемости занятий и предварительную обработку этих данных для анализа.

Также был проведен обзор реализованных методов учета посещаемости и поставлена задача на разработку алгоритма учета посещаемости. В качестве результата работы был описан алгоритм учета посещаемости занятий студентами и преподавателями, а также выбран подходящий вычислительный метод для его реализации.

Для реализации программы по учету посещаемости была проведена подготовка данных, включая сбор данных о посещаемости занятий студентами и преподавателями, а также данных о расписании занятий. Были рассмотрены и применены такие существующие алгоритмы, как парсинг, работа с базами данных и алгоритм сравнения дат и времени. Реализация модулей программы на языке Python, позволяющих создавать файл, хранящий в себе отчет о посещаемости выбранного студента или преподавателя.

В заключение, было проведено тестирование реализованной программы по учету посещаемости студентов и преподавателей, которое показало, что программа работает стабильно и выдает правильные значения.

Разработанный алгоритм может быть использован в образовательных организациях для учета посещаемости занятий и повышения эффективности учебного процесса.

Список используемых источников

1. Алгоритмы. Алгоритмизация. Алгоритмические языки [Электронный ресурс]. – Режим доступа: http://book.kbsu.ru/theory/chapter7/1_7.html

2. Новостной портал Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/507268/ – Статья о работе сервера СКУД. 2020.

3. Новостной портал Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/articles/220125/ – Статья о парсинге при помощи библиотеки lxml.2014.

4. Новостной портал Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/otus/articles/596071/ – Статья о Selenium, его настроек и взаимодействий с веб-элементами. 2021.

5. Новостной портал Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/retailrocket/articles/431572/ – Статья о функциональных требованиях. 2018

6. Образовательный блог OTUS JOURNAL [Электронный ресурс]. – Режим доступа: https://otus.ru/journal/algoritmy-i-podprogrammy/ – Статья о алгоритмах и подпрограммах. 2023.

7. Образовательный портал CoderLessons [Электронный ресурс]. – Режим доступа: https://coderlessons.com/tutorials/python-technologies/vyuchit-piton/python-kratkoe-rukovodstvo – Учебник по краткому руководству Python. 2019.

8. Образовательный портал Smartiqa [Электронный ресурс]. – Режим доступа: https://smartiqa.ru/blog/selenium-webdriver-basics – Статья о Selenium WebDriver. 2022.

9. Проблемы реализации биометрической аутентификации: подробное руководство [Электронный ресурс]. – Режим доступа: https://ts2.space/ru/проблемы-реализации-биометрической/ – Статья о биометрической аутентификации, ее преимуществ и недостатков

10. Рыжова В.А Проектирование и исследование комплексных систем безопасности – СПб: НИУ ИТМО, 2012. С. 84-85. – Режим доступа: http://oeps.ifmo.ru/uchebn/UP_KSB.pdf

11. Сайт компании Artonit.ru [Электронный ресурс]. – Режим доступа: http://artonit.ru/index.php?option=com_content&view=article&id=137:2014-05-11-06-52-01&catid=55:tutorial&Itemid=76 – Основные задачи драйвера контроллера. 2014.

12.Сайт компании «ААМ Системз» [Электронный ресурс]. – Режим доступа: https://www.aamsystems.ru/programmnye_kompleksy/programmnyy_kompleks_apacs_3000/#tab3 – Архитектура сервера программного комплекса APACS 3000

13. Сайт компании «ААМ Системз» [Электронный ресурс]. – Режим доступа: https://www.aamsystems.ru/programmnye_kompleksy/programmnyy_kompleks_apacs_3000/#tab4 – Возможности программного комплекса APACS 3000

14. Сайт компании Roistat.Blog [Электронный ресурс]. – Режим доступа: https://roistat.com/rublog/parsing/ – Статья о понятии парсинга и его основных задачах

15. Сайт компании «ПожСистемСтрой» [Электронный ресурс]. – Режим доступа: https://pozhsystems.ru/skud-sistema-kontrolja-i-upravlenija-dostupom-naznachenie-i-primenenie-na-razlichnyh-obektah/ – Описание системы контроля и управлением доступом

16. Сайт компании iBase.ru [Электронный ресурс]: – Режим доступа: http://www.ibase.ru/joins/ – Статья о использование неявных и явных JOIN в Firebird

17. Сайт «Лаборатория линуксоида» [Электронный ресурс]. – Режим доступа: https://younglinux.info/datetime/date – Статья о работе с датами

18. Электронное пособие «Информатика» [Электронный ресурс]. – Режим доступа: http://psk68.ru/files/metod/uchebnik_Informatika/algor.html – Алгоритмы и способы их описания

19. Электронное пособие о языке программирования Python [Электронный ресурс]. – Режим доступа: https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html – Статья о использования конструкций try finally для обработки исключений

20. Яндекс Дзен [Электронный ресурс]. Режим доступа: https://dzen.ru/a/ZEzJD13tNUGS1Ow0 – Статья о работе с библиотекой BeautifulSoup. 2023.

21. FireBird Driver [Электронный ресурс]. – Режим доступа: https://firebird-driver.readthedocs.io/en/latest/getting-started.html – Executing SQL Statements

22. GeeksforGeeks [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/comparing-dates-python/ – Comparing dates in Python

23. Learn Microsoft [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/sql/relational-databases/query-processing-architecture-guide?view=sql-server-ver16 – An article on the query architecture guide

24. Learn Microsoft [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-ver15 – Article about joins (SQL Server)

25. Resources for Developers [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction – Article about Document Object Model