Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Основы алгоритмизации и программирования»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА КАДРОВ НА ПРЕДПРИЯТИИ»**

БГУИР КР 6-05-0611-01 030 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 478101  Ложечка Артём Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента)  Курсовая работа представлена на  проверку \_\_\_.\_\_\_.2025  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2025

**РЕФЕРАТ**

БГУИР КР 6-05-0611-01 030 ПЗ

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УЧЕТА КАДРОВ НА ПРЕДПРИЯТИИ: курсовая работа / А.А. Ложечка – Минск: БГУИР, 2025, – п.з. – 63 с., чертежей (плакатов) – 5 л. формата А4.

Пояснительная записка 63 с., 65 рис., 0 табл., 5 источников, 2 приложения.

**Ключевые слова**: автоматизация, информационная система, клиент, администратор.

*Цель курсовой работы:* оптимизация процесса учета кадров на предприятии посредством разработки программной поддержки.

*Объект исследования* – процесс учета деятельности учета кадров.

*Предмет исследования* – совокупность научной литературы, статистических методов обработки реальных данных, алгоритмов вычисления, способов повышения эффективности объекта исследования.

*Методология проведения работы*: в процессе разработки системы использованы принципы системного подхода, аналитические методы, методы компьютерной обработки экспериментальных данных и моделирование системы с помощью UML-диаграмм, разработана программная и графическая составляющие программного средства.

*Результаты работы:* рассмотрены основные бизнес-процессы предметной области. В результате была разработана системная поддержка, осуществляющая учет кадров на предприятии. Приложение обладает удобным и простым интерфейсом, позволяет обеспечить учет кадров, оптимизировать их работу, упростить процесс управления кадрами.

СОДЕРЖАНИЕ

Введение7

1 Анализ и моделирование предметной области8

1.1 Описание предметной области8

1.2 Описание процесса «Разработки автоматизированной системы учета деятельности фитнес-центра» в виде модели IDEF0 9

1.3 Анализ требований к разрабатываемому программному средству22

1.4 Разработка BPMN-модели регистрации клиентов в системе и её описание.22

1.5 UML-модели представления программного средства и их описание23

2 Проектирование и конструирование программного средства 27

2.1 Постановка задачи27

2.2 Разработка модульной структуры программы 27

2.3 Выбор способа организации данных 30

2.4 Разработка перечня пользовательских функций программы 32

2.5 Разработка схем алгоритмов работы программы23

3 Тестирование и проверка работоспособности программного средства36

4 Инструкция по развертыванию приложения и сквозной тестовый пример40

4.1 Авторизация40

4.2 Модуль администратора41

4.3 Модуль пользователя47

Заключение51

Список использованных источников52

Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат»53

[Приложение Б](https://docs.google.com/document/d/1x_vH5kViJ9Lcw-7kOMglWuKBNDkax_RM/edit#heading=h.44sinio) (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику приложения54

Ведомость документов63

**ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ**

ПО – программное обеспечение, набор программ на компьютере или ином устройстве.

Информационная система (ИС) – система обработки данных совместно с соответствующими организационными ресурсами (человеческими, техническими, финансовыми и т. д.), обеспечивающая и распространяющая информацию.

IDEF0 (Integration Definition for Function Modelling) – метод функционального моделирования и графическая нотация, предназначенные для формализации и описания бизнес-процессов.

BPMN (Business Process Model and Notation) – язык моделирования бизнес-процессов, являющийся промежуточным звеном между формализацией и реализацией бизнес-процесса.

UML (Unified Modelling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

Идентификатор, ID – уникальный признак объекта, отличающий его от других объектов.

Файловая система – способ организации данных на информационных носителях.

Блок-схема алгоритма – графическое представление последовательности шагов, необходимых для выполнения конкретной задачи.

## В современном мире эффективное управление персоналом является одним из ключевых факторов успеха любого предприятия. Кадровый учет играет важную роль в обеспечении стабильной работы организации, так как от качества управления человеческими ресурсами напрямую зависят производительность труда, мотивация сотрудников и общая эффективность бизнеса. Однако ручной учет кадров, особенно на крупных предприятиях, требует значительных временных и трудовых затрат, что делает процесс трудоемким и подверженным ошибкам. В связи с этим разработка автоматизированной системы учета кадров становится актуальной задачей, способной оптимизировать процессы управления персоналом.

## Цель данной курсовой работы – создание информационной системы для учета кадров на предприятии, которая позволит автоматизировать процессы управления персоналом, упростить документооборот и повысить эффективность работы кадровой службы. Для достижения этой цели необходимо выполнить следующие задачи:

## – Провести анализ существующих методов учета кадров и выявить их недостатки.

## – Изучить потребности и требования предприятия к системе учета персонала.

## – Спроектировать и разработать автоматизированную систему учета кадров.

## – Провести тестирование и внедрение разработанной системы.

## Данная работа представляет собой комплексное исследование, охватывающее как теоретические, так и практические аспекты разработки информационной системы учета кадров. В ходе выполнения работы будут использованы методы анализа, проектирования и программирования, с акцентом на современные информационные технологии для создания эффективной и удобной системы.

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

## **Описание предметной области**

Предметная область данной темы – управление персоналом на предприятии. Она охватывает все аспекты деятельности, связанные с учётом сотрудников, их распределением по отделам, управлением должностями, кадровыми изменениями и контролем за выполнением трудовых обязанностей. Внедрение специализированной информационной системы позволит оптимизировать процессы, связанные с подбором, обучением персонала, а также повысить эффективность управления человеческими ресурсами.

Предприятие – это сложная структура, где каждый сотрудник играет важную роль в достижении общих целей. Учёт кадров включает в себя управление данными о сотрудниках, их профессиональных навыках, должностях, отделах, а также контроль за кадровыми изменениями (приём на работу, переводы, увольнения). По статистике, эффективное управление персоналом напрямую влияет на производительность предприятия, а автоматизация этих процессов позволяет сократить издержки и повысить качество работы.

В 2024 году рынок решений для управления персоналом оценивался в более чем 20 миллиардов долларов, с прогнозируемым ежегодным ростом на 10-12%. Целевая аудитория таких систем включает предприятия различных отраслей, от малого бизнеса до крупных корпораций, стремящихся оптимизировать свои кадровые процессы и повысить эффективность управления персоналом.

Действующими лицами деятельности спортивного клуба являются следующие субъекты [1] :

* администратор;
* HR-менеджер(пользователь ИС);
* Рабочий персонал;

Исходя из деятельности предприятия, можно выделить следующие бизнес-процессы:

1. Учёт сотрудников:

* Добавление, редактирование и удаление сотрудников;
* Хранение основных данных сотрудника;

1. Управление кадровыми событиями:

* Оформление приёма на работу;
* Оформление увольнения;
* Учёт отпусков;

1. Поиск, сортировка и фильтрация:

* Поиск сотрудников по ФИО, должности или отделу;
* Фильтрация по выбранному параметру;
* Сортировка по выбранному параметру;
* Поиск по выбранному параметру;

1. Отчётность:

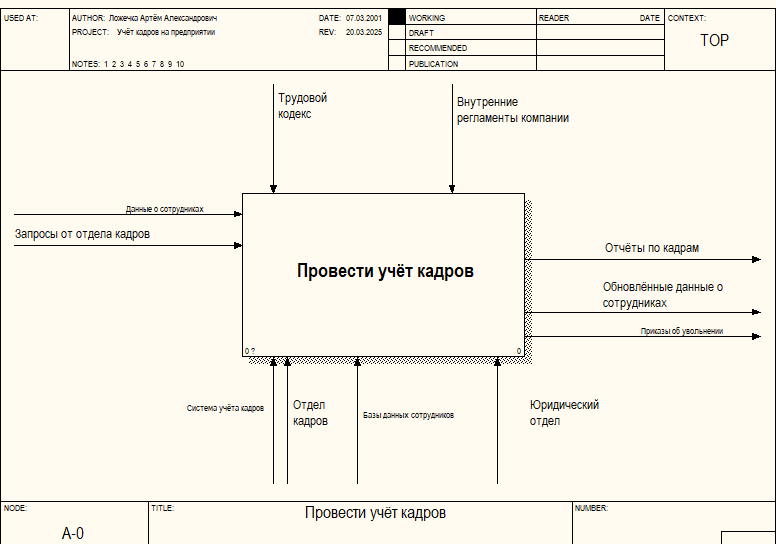
* Вывод списка всех сотрудников;

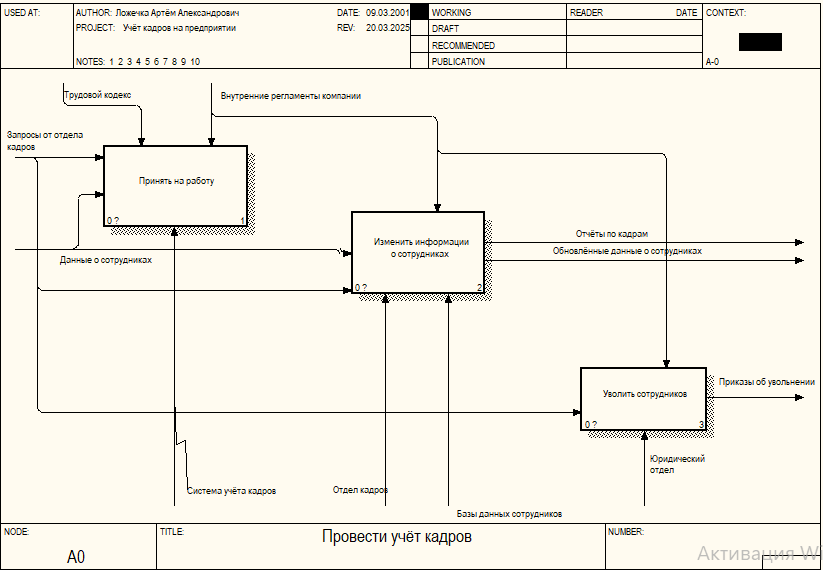
Разработка автоматизированной системы учёта кадров на предприятии включает в себя анализ, проектирование, разработку и внедрение программного обеспечения, а также обучение персонала работе с данной системой, последующую поддержку и сопровождение системы. Это позволяет оптимизировать деятельность предприятия [2].

## **Описание процесса «Разработки автоматизированной системы учета кадров на предприятии» в виде модели IDEF0**

Система учета кадров на предприятии – это целостный комплекс, включающий в себя процессы и инструменты, нацеленные на оптимизацию управления и мониторинга всех сфер деятельности. Внедрение такой системы станет залогом совершенствования работы предприятия, обеспечивая точный учет кадровых потоков, а также эффективное ведение пользовательской базы и управление рабочим персоналом. Это позволит не только повысить качество производства, но и способствует более грамотному распределению ресурсов.

На рисунках 1.2 и 1.3 изображена диаграмма, описывающая систему с помощью методологии IDEF0 [3].

Рисунок 1.1 – Провести учёт кадров

Рисунок 1.2 – Декомпозиция диаграммы верхнего уровня

Входными данными являются данные о сотрудниках и запросы от отдела кадров. На выходе же мы получаем отчёты по кадрам, обновлённые данные о сотрудниках, а также приказы об увольнении. В работе задействован отдел кадров, юридический отдел, базы данных и автоматическая система. В свою очередь будут учитываться трудовой кодекс и внутренние регламенты компании.

Первый подпроцесс занимается принятием сотрудников на работу. Сюда входит обработка данных сотрудников.

Этот подпроцесс можно разбить на более мелкие. Это поможет нам намного подробнее описать его. Декомпозиции представлены на рисунках 1.3, 1.4, 1.5. Рассмотрев их, мы можем понять из каких операций состоит данный подпроцесс.

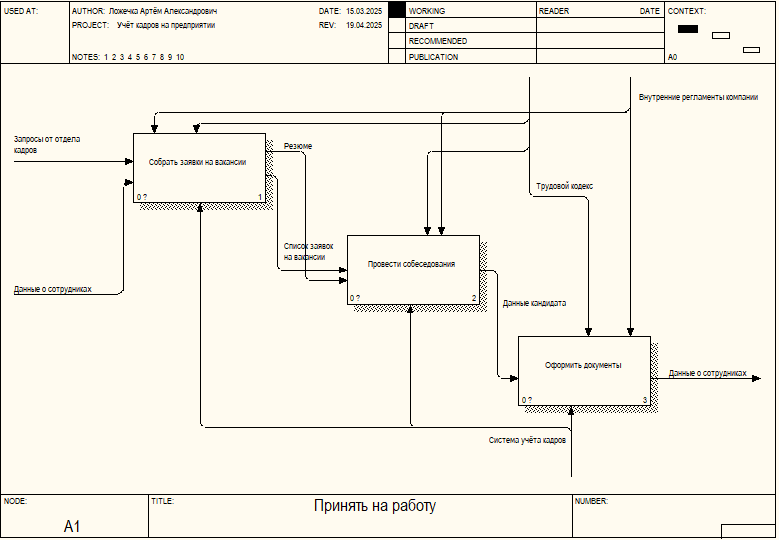


Рисунок 1.3 – «Принять на работу»

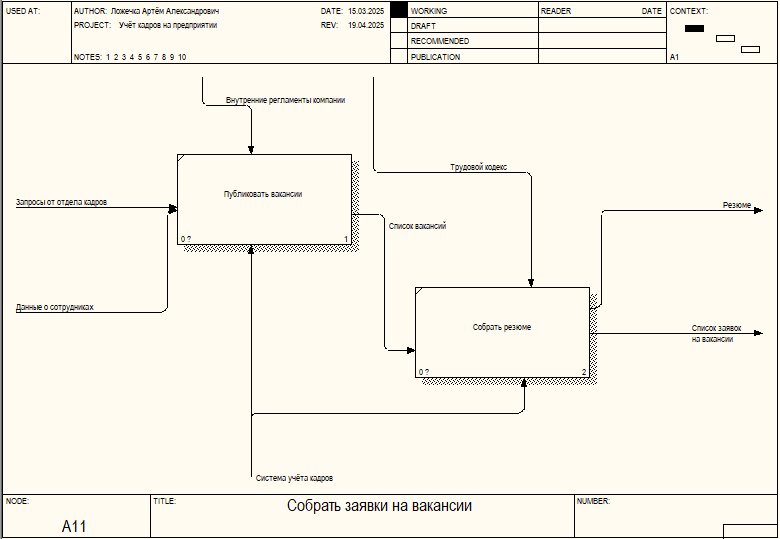


Рисунок 1.4 - Декомпозиция блока «Собрать заявки на вакансии»

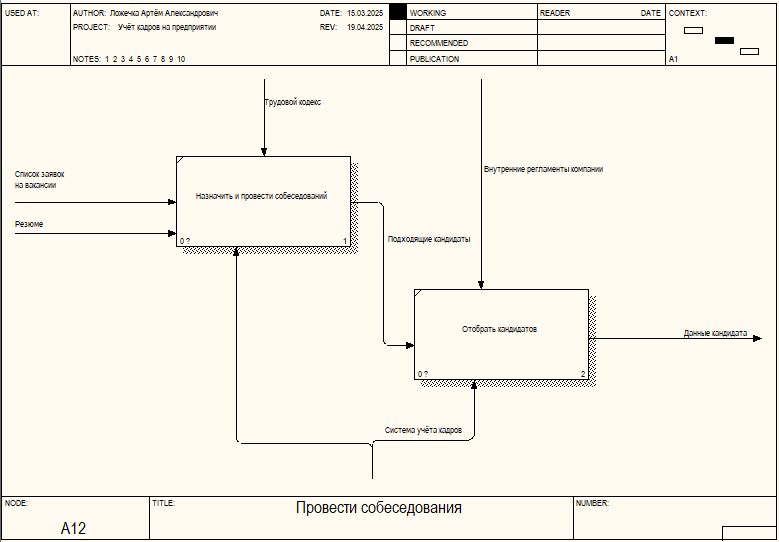


Рисунок 1.5 - Декомпозиция блока «Провести собеседования»

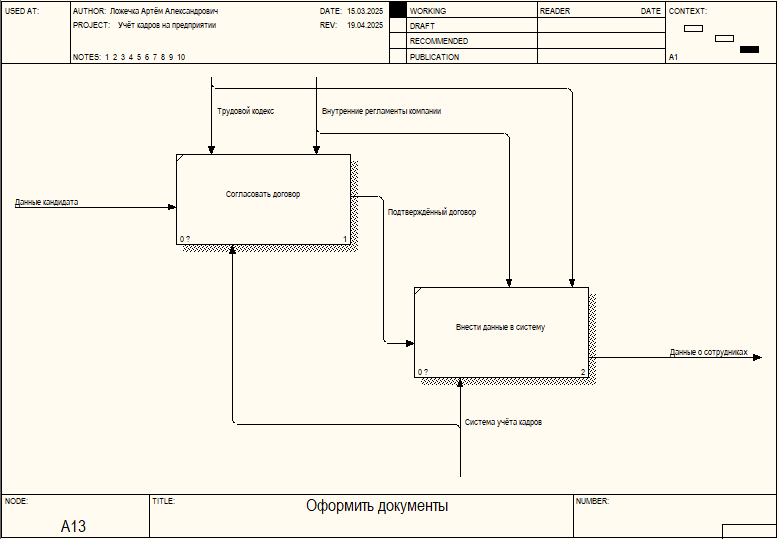


Рисунок 1.6 - Декомпозиция блока «Оформить документы»

Опишем второй подпроцесс «Изменить информацию о сотрудниках».

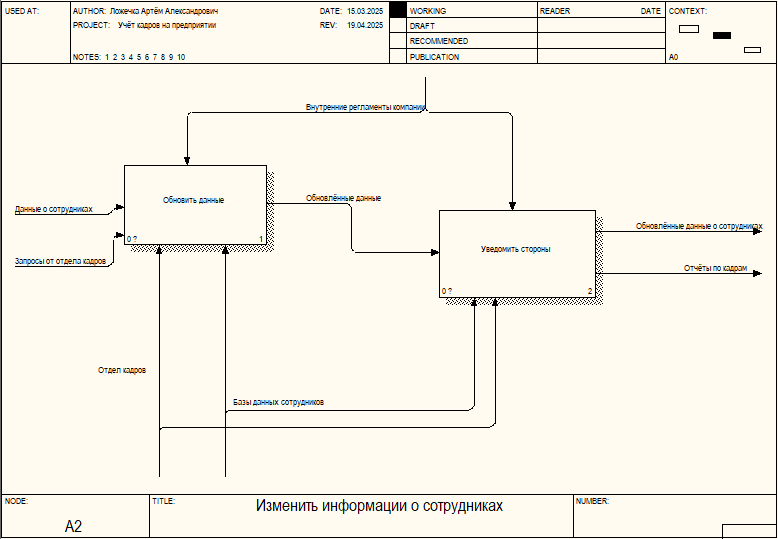


Рисунок 1.7 - Декомпозиция блока «Изменить информацию о сотрудниках»

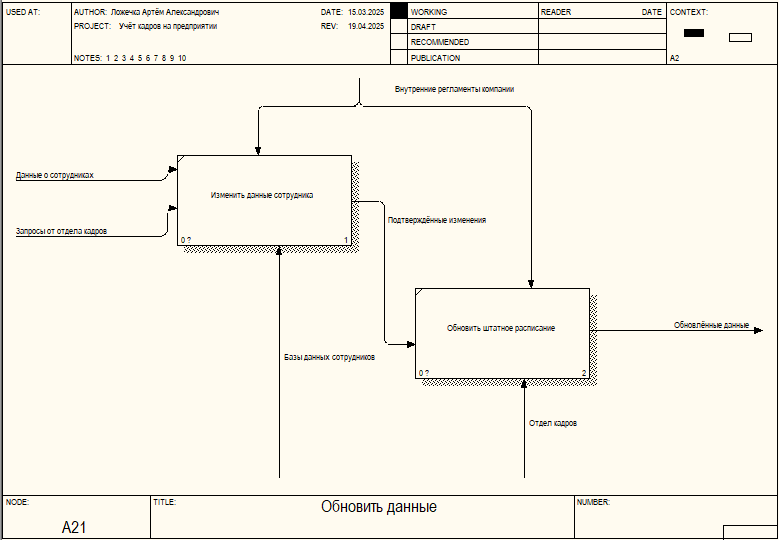


Рисунок 1.8 - Декомпозиция блока «Обновить данные»

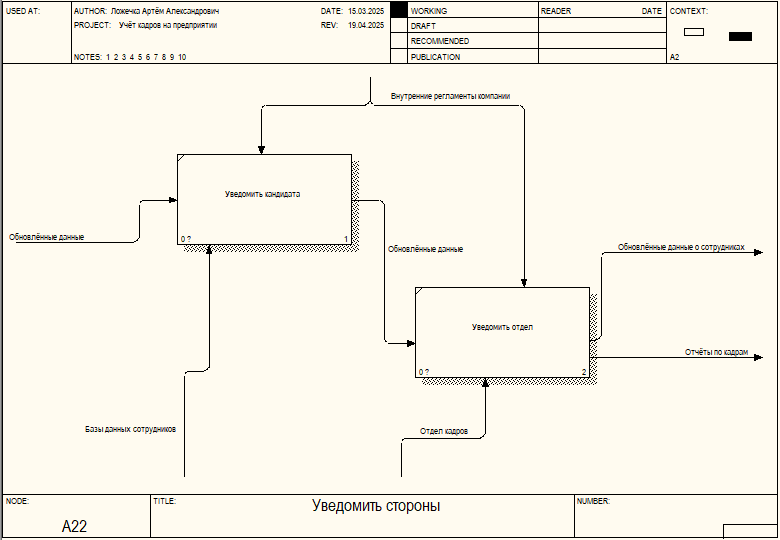


Рисунок 1.9 - Декомпозиция блока «Уведомить стороны»

Также декомпозируем блок «Уволить сотрудника» (см.рис.1.10).

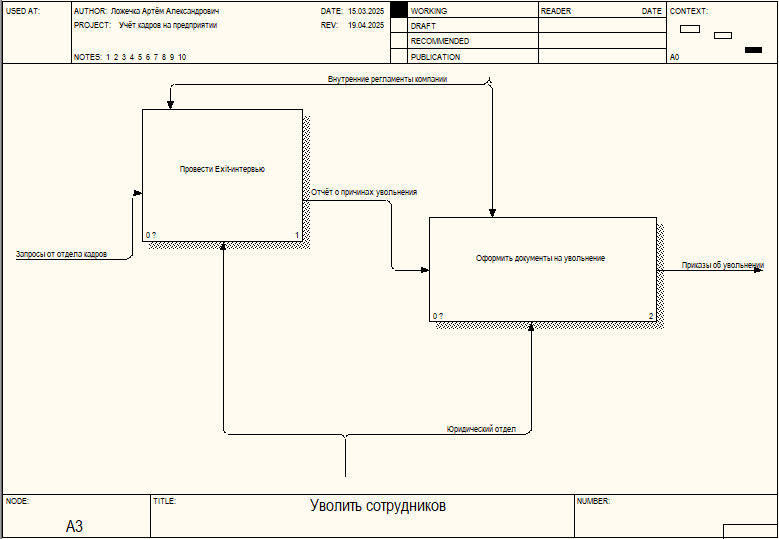


Рисунок 1.10 - Декомпозиция блока «Уволить сотрудников»

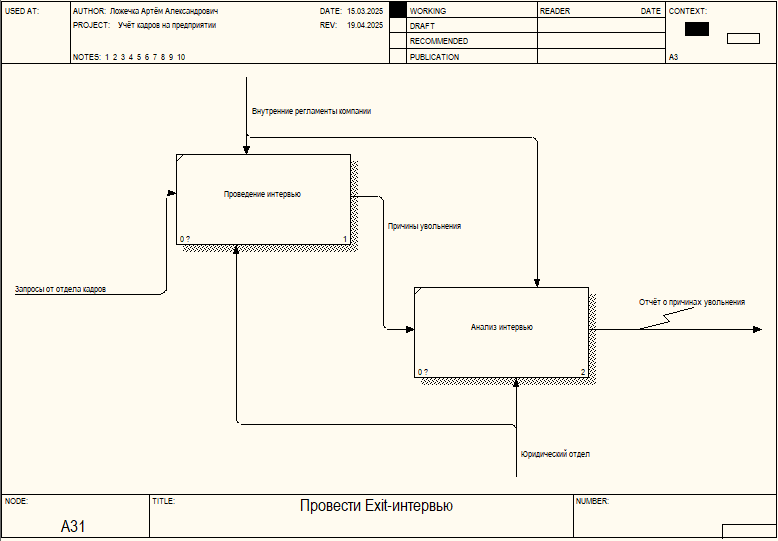
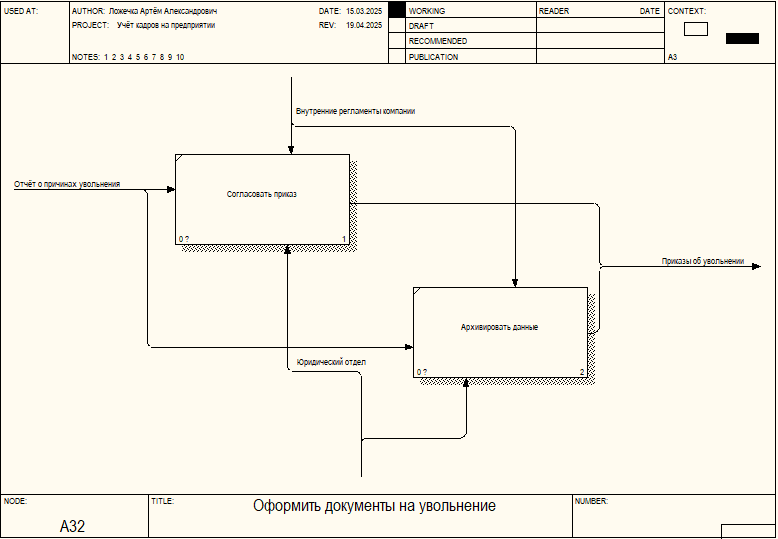


Рисунок 1.11 - Декомпозиция блока «Провести Exit-интервью»

Рисунок 1.12 - Декомпозиция блока «Оформить документы на увольнение»

Третий подпроцесс «Управление правами потребителя» (рис.1.13-1.1) в фитнес-центре можно представить в виде следующего процесса:

1. Получение запроса

* получение информации о запросе на изменение прав потребителя (например, изменение абонемента, доступ к определенным зонам или услугам);
* определение типа запроса (например, временное или постоянное изменение прав).

2. Анализ запроса

* оценка соответствия запроса установленным правилам и политикам фитнес-центра;
* подтверждение возможности выполнения запроса.

3. Обработка запроса

* принятие решения о выполнении запроса;
* регистрация изменений в системе управления правами потребителя.

4. Выполнение запроса

* внесение изменений в доступ потребителя к определенным зонам или услугам;
* уведомление потребителя о выполненных изменениях.

5. Мониторинг изменений

* отслеживание внесенных изменений в права потребителя;
* анализ эффективности выполненных изменений.

6. Отчетность

* формирование отчетов о выполненных изменениях в правах потребителей;
* обратная связь с потребителями о выполненных изменениях.

Этот процесс помогает фитнес-центру эффективно управлять правами потребителей, обеспечивая им доступ к необходимым услугам и зонам, а также поддерживая контроль над изменениями в их правах.

## **1.3 Анализ требований к разрабатываемому программному средству**

Для наглядного представления всех возможностей разработанного программного продукта используется диаграмма вариантов использования (use case-диаграмма) в нотации UML, представленная на рисунке 1.19.

Разрабатываемая система будет использоваться работником отдела кадров и администратором. Следовательно, для работы с данными должны быть предусмотрены два функциональных модуля: модуль администратора и модуль пользователя.

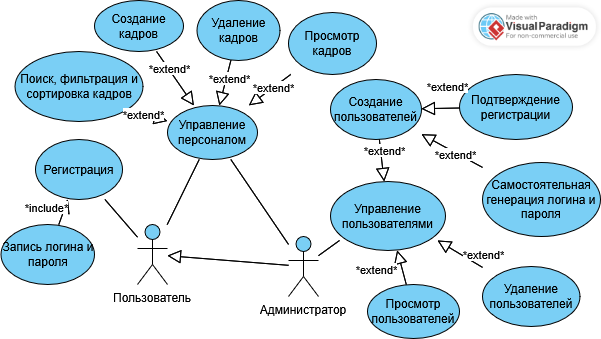


Рисунок 1.13 «Use case-диаграмма»

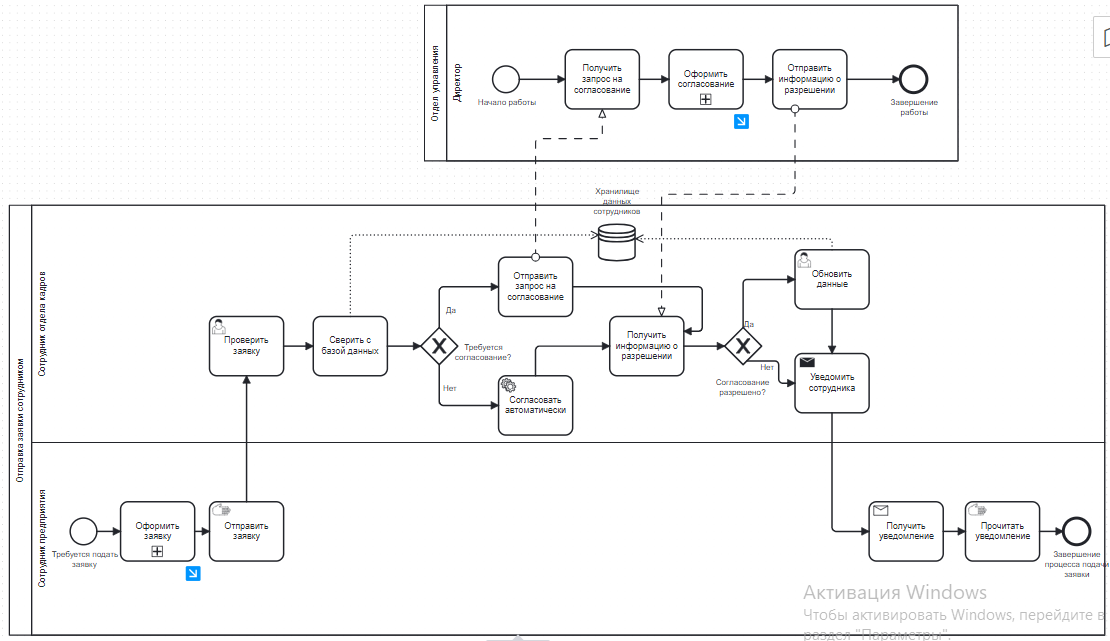
## **1.4 Разработка BPMN-модели изменения данных о сотруднике**

На рисунке 1.20 представлена схема регистрации клиентов в системе.

Здесь используются отдельные пулы для описания процесса регистрации, для клиента и для администратора. Таким образом, мы можем четко определить, кто контролирует какой процесс.

В пуле клиент используются пользовательские задачи. Эти задачи включают в себя: оформление заявки на регистрацию, отправление заявки, проверка результата запроса, повторение запроса или получение данных и регистрация.

Взаимодействие между задачами клиентами и администратора моделируется с использованием потоков сообщений. Эти потоки сообщений инкапсулируют шаги документации, которые должен выполнить администратор, чтобы выполнить пользовательскую задачу. Например, клиент отправляет заявку на рассмотрение регистрации, администратор же получает этот запрос и далее анализирует его на соответствие требованиям. Далее администратор проводит действия для регистрации клиента либо сообщает ему о невозможность его регистрации.

Рис. 1.20 – BPMN-модель регистрации клиентов в системе

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

**Диаграмма последовательности** – это наглядное представление последовательности операций или сообщений между объектами в системе. Она демонстрирует взаимодействие объектов в определенном порядке для достижения конкретной цели. Например, такая диаграмма может иллюстрировать, как пользователь взаимодействует с интерфейсом, отправляет запрос к серверу, который обрабатывает запрос и возвращает ответ. Это важный инструмент для понимания взаимодействия между различными компонентами системы и последовательности шагов, необходимых для достижения определенного результата.

На рисунке 1.21 представлена диаграмма последовательности выполнения действий для регистрации клиентов.

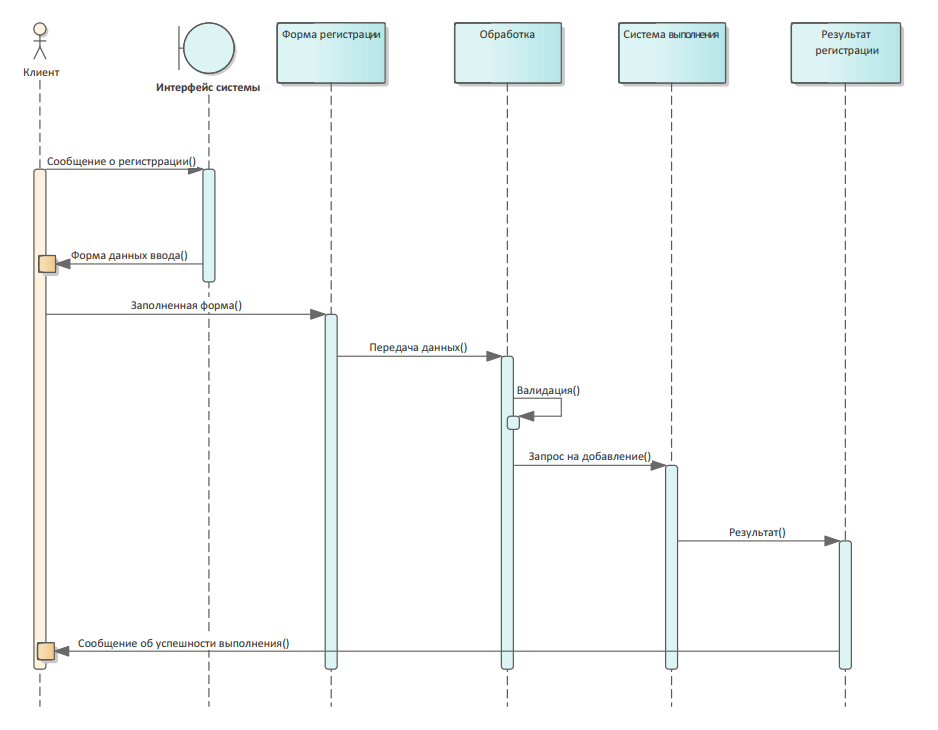


Рис. 1.21 – Диаграмма последовательности выполнения действий для регистрации клиентов

Диаграмма последовательности подробно описывает шаги, совершаемые при регистрации пользователя на сайте или в приложении, начиная от ввода данных пользователем и заканчивая созданием учетной записи в базе данных. Она наглядно представляет взаимодействие между различными компонентами системы и последовательность операций, необходимых для завершения процесса регистрации.

**Диаграмма состояний** иллюстрирует все возможные состояния, в которых может находиться объект или система, и переходы между этими состояниями. Она может отображать различные сценарии работы, такие как состояния включения и выключения устройства, изменения статуса заказа, жизненный цикл продукта и т.д. Этот инструмент помогает представить, как объект или система будут вести себя в различных ситуациях и какие действия будут происходить при переходе между состояниями.

Диаграмма состояний, изображенная на рисунке 1.22, демонстрирует последовательность состояний, в которых будет находиться автоматизированная система учета деятельности спортивного в процессе его работы.

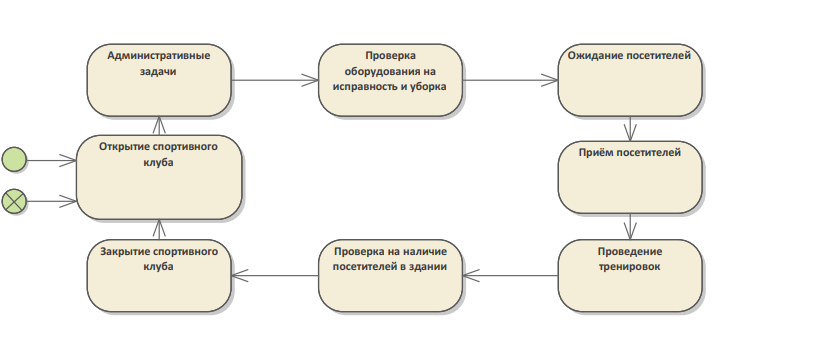


Рис. 1.22 – Диаграмма состояний отражает состояния системы

В диаграмме состояний, показанной на рисунке 1.5.2, можно понять в какой последовательности выполняется деятельность спортивного клуба. После «Открытие спортивного клуба» идёт «Административные задачи». После решения административных задач производится «Проверка оборудования на исправность и уборка», «Ожидание посетителей», «Приём посетителей», «Проведение тренировок» посетителей клуба. Перед закрытием идёт «Проверка на наличие посетителей в здании» и в конце «Закрытие центра».

Диаграмма компонентов предоставляет визуально организацию компонентов системы и их взаимосвязей, что помогает получить общее представление о структуре системы.

Реализация данной информационной системы будет располагаться в четырех компонентах со следующими названиями:

* «Клиент»;
* «Администратор»;
* «Спортивная услуга»;
* «Информационная система».

«Администратор» необходим для управления процессом бронирования косметических услуг. «Клиент» — для осуществления бронирования услуг. «Косметическая услуга» включает работу с данными об услуге. «Информационная система» необходима для организации данных в файл, а также авторизации или регистрации пользователей.

На рисунке 1.23 приведена диаграмма компонентов разрабатываемой информационной системы:

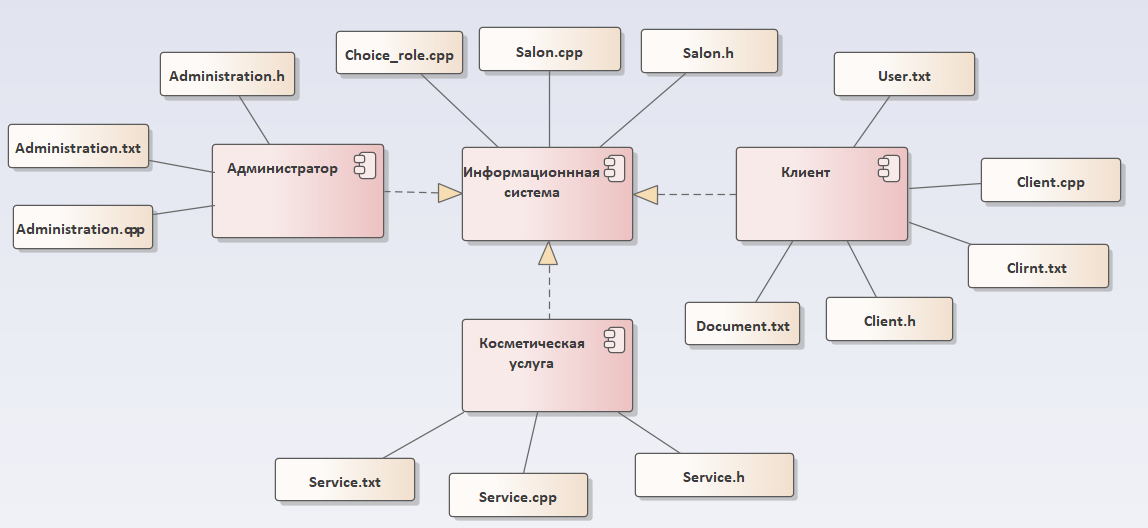


Рисунок 1.23 – Диаграмма компонентов

Таким образом с помощью IDEF0, BPMN, UML-диаграммы вариантов использования, диаграмм последовательности, состояния и компонентов был описан процесс бронирования косметических услуг. Приведенные диаграммы позволяют изучить специфику данного процесса с нескольких сторон. Бронирование косметических услуг подразделяется на несколько этапов, в которых происходит взаимодействие клиента и администратора для достижения главной цели салона — позволить клиенту качественно забронировать косметическую услугу.

# 2 Проектирование и конструирование программного средства

**2.1. Постановка задачи**

По условию задания требуется разработать автоматизированную систему учёта деятельности фитнес-центра. Система должна предусмотреть режимы хранения данных для входа в приложение. Каждый клиент должен будет придумать логин и пароль.

Помимо данных о клиентов, в базе данных хранятся данные о администраторах, такие как логин, пароль и код роли.

Помимо данных о клиентов и администраторах, в базе данных хранятся данные об абонементах, которые будут включать в себя:

* название абонемента;
* дату начала занятий;
* стоимость абонемента;
* количество абонемента на сеанс;
* имя тренера.

Предусмотрим следующие ограничения для информационной системы: для успешного создания записи о модели все информационные параметры должны быть заполнены.

Для удобного и качественного взаимодействия между администратором фитнес-центра и клиентом разрабатываемая информационная система должна обладать следующим функционалом:

* + просмотр всех учетных записей в табличной форме;
  + редактирование учетной записи пользователей;
  + удаление учетной записи.

Пользователь должен иметь возможность просмотра абонементов, фильтрацию, а именно поиск и сортировка абонементов. Это позволит пользователю получить информацию о доступных абонементах,, фильтровать продукты по различным критериям и приобретать абонементы на разные занятия.

Так же для обеспечения безопасности и конфиденциальности данных пользователя, программа должна предоставлять возможность управления заявками на регистрацию, позволяющую администратору блокировать недопустимые учётные записи пользователей.

**2.2 Разработка модульной структуры**

Модульное программирование – это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок.

Unified Modeling Language (UML) [4] – это язык графического описания для объектного моделирования, используемый в разработке программного обеспечения для моделирования бизнес-процессов, системного проектирования и отображения организационных структур. UML позволяет описывать связи между компонентами, модулями или сервисами, программными процессами и многим другим.

В данном проекте использован объектно-ориентированный подход, который предполагает создание собственных типов данных, называемых классами. Этот подход был применен для разработки различных классов в системе. Диаграмма классов приведена на рисунке



Рисунок 2.1 – Диаграмма классов

Диаграмма классов – структурная диаграмма языка моделирования, демонстрирующая общую структуру иерархии классов в системе, их коопераций, атрибутов (полей), методов, интерфейсов, а также взаимосвязей между ними. Рассмотрим классы в проекте более подробно.

1. Класс WaitingData

Класс предназначен для хранения данных, ожидающих подтверждения.

Поля:

* login (string) – Логин;
* password (string) – Пароль.

2. Класс Data

Класс предназначен для работы с данными пользователей.

Поля:

* login (string) – логин;
* saltedHashPassword (string) – хэшированный пароль;
* role (int) – роль пользователя;
* access (int) – уровень доступа.

3. Класс currentDate

Класс хранит текущую дату.

Поля:

* day (int) – день;
* month (int) – месяц;
* year (int) – год.

4. Класс Date

Абстрактный класс для хранения даты, когда был зарегистрирован товар.

Поля:

* day (int) – день;
* month (int) – месяц;
* year (int) – год.

5. Класс Vault

Класс для работы с товарами.

Поля:

* name (string) – название товара;
* amount (int) – количество товара;
* price (int) – цена товара;
* surname (string) – фамилия владельца.

Эти классы и методы были разработаны с использованием UML для создания структурированной иерархии и для обеспечения эффективного взаимодействия между различными компонентами системы.

**2.3 Выбор способа организации хранения данных**

Для оптимизации работы системы, информация в программах хранится в текстовых файлах, где каждая строка разделена на поля с использованием символа перехода на новую строку. При инициализации программы данные из файлов загружаются в оперативную память для последующей обработки.

Для хранения всех данных в программе используется три файла. Ниже будут рассмотрены каждый из файлов более детально.

data.txt (рисунок 2.2) – файл, содержащий данные о всех пользователях зарегистрированных в приложении как администраторов, так и самих клиентов.



Рисунок 2.2 – Файл «user.txt»

waiting\_data.txt (рисунок 2.3) – файл, содержащий данные о всех заявках пользователей на регистрацию в этом приложении. Первая строка содержит логин пользователя, а вторая пароль.

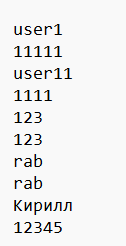


Рисунок 2.3 – Файл «waiting\_data.txt»

vault\_data.txt (рисунок 2.4) – файл, содержащий данные о всех абонементах, которые есть в фитнес-центре. Первое слово в строке содержит название абонемента, второе слово это имя тренера, который проводит эти занятия, третье слово это количество абонементов, четвертое слово это стоимость одного абонемента, пятое слово это день проведения занятий, шестое слово это месяц проведения занятий, седьмое слово это год проведения занятий.

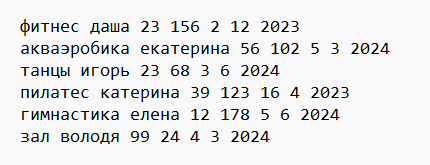


Рисунок 2.4 – Файл «vault\_data.txt»

checkuser1.txt (рисунок 2.5) – файл, содержащий данные о всех билетах купленных пользователем. Первая строка содержит название клуба, третья строка содержит информацию о выбранной секции. Четвертая строка содержит количество купленных абонементов, пятая строка содержит общую стоимость абонементов. Шестая строка содержит дату первого занятия. Седьмая строка содержит информацию о клиенте.

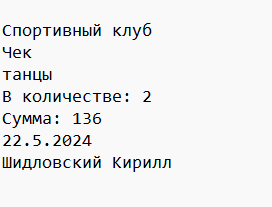


Рисунок 2.5 – Файл «checkuser1.txt»

Использование файлов в программе, значительно улучшит ее в работе. Это избавит пользователя вводить все данные при каждом новом запуске системы.

* 1. **Разработка перечня пользовательских функций программы**

Для разработки программы были использованы следующие функции:

– void creationFirstAdmin() – эта функция используется для создания первого администраторского аккаунта в системе;

–string inputPassword() – функция запрашивает у пользователя ввод пароля и возвращает введенный пароль в виде строки;

– void startMenu() – эта функция отображает начальное меню при запуске программы;

– void showUsers() – функция выводит список пользователей системы;

– void addUser() – данная функция предназначена для добавления нового пользователя в систему;

– void editUsers() – функция предоставляет возможность администратору редактировать информацию о существующих пользователях;

– void deleteUsers() – функция удаляет пользователя из системы;

– void userBase() – эта функция отвечает за управление базой данных пользователей;

– void endProgram() – функция завершает выполнение программы;

– void enter() – функция обрабатывает процесс входа в систему;

– void registration() – функция позволяет пользователям зарегистрироваться в системе;

– int inputCheck(int a, int b) – функция проверяет корректность ввода данных, принимая на вход два значения и возвращая целое число;

– void CreationHash() – функция создает файл с хешом для безопасного хранения паролей, если таковой отсутствует;

– string RandomString() – эта функция создает случайную строку и возвращает ее в виде строки;

– string Crypt(string password) – функция шифрует пароль для обеспечения безопасного хранения;

– string Decrypt(string password) – данная функция дешифрует зашифрованный пароль;

– void showRequests() – функция выводит список заявок на регистрацию пользователей;

– void userRequest() – эта функция обрабатывает заявки на регистрацию пользователей;

– void adminMenu() – функция отображает меню администратора;

– void getCurrentDate(int\* tempDay, int\* tempMonth, int\* tempYear) – данная функция находит текущую дату и записывает ее в переданные аргументы (указатели на переменные дня, месяца и года).

**2.5 Разработка схем алгоритмов программы**

Алгоритм программы [5] представляет собой последовательность шагов или инструкций, которые определяют порядок выполнения конкретной задачи компьютерной программой. Он определяет, какие действия необходимо выполнить и в каком порядке, чтобы достичь желаемого результата. Алгоритм является основой для написания программного кода и помогает программистам организовать задачу перед ее реализацией на компьютере. На рисунке 2.6 представлен алгоритм функции userMenu().

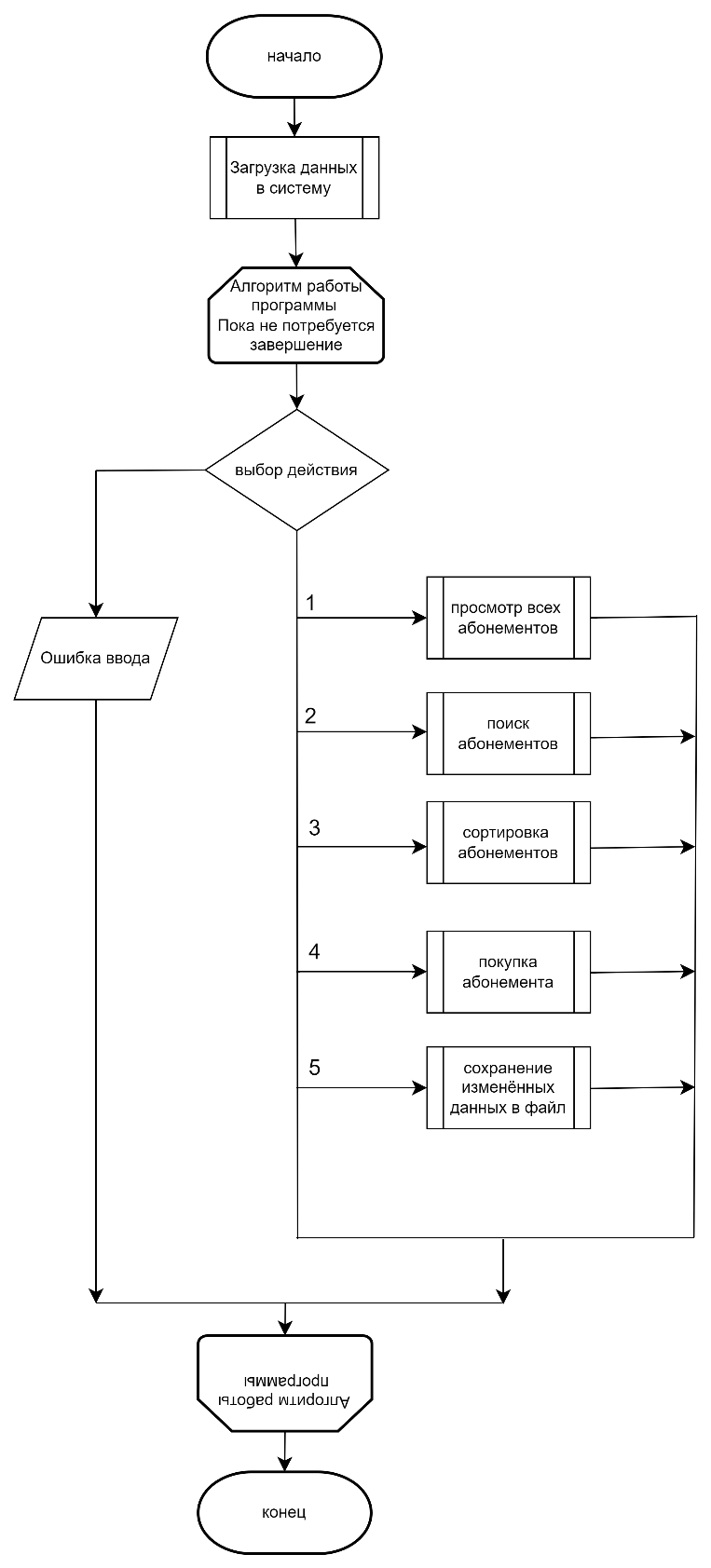


Рисунок 2.6 – Алгоритм функции userMenu()

Сначала в функции userMenu() происходит загрузка данных в систему, позже пользователю показывается меню из 5 пунктов, а именно:просмотреть все абонементы, поиск абонементов, сортировка абонементов, покупка абонемента, выход из программы. В случае ввода не того пункта будет выведено сообщение об ошибке. После успешной работы с данными и их изменения данные сохраняются в файл.

На рисунке 2.7 представлен алгоритм функции addUser(), суть которой заключается в том, добавить новую учётную запись.

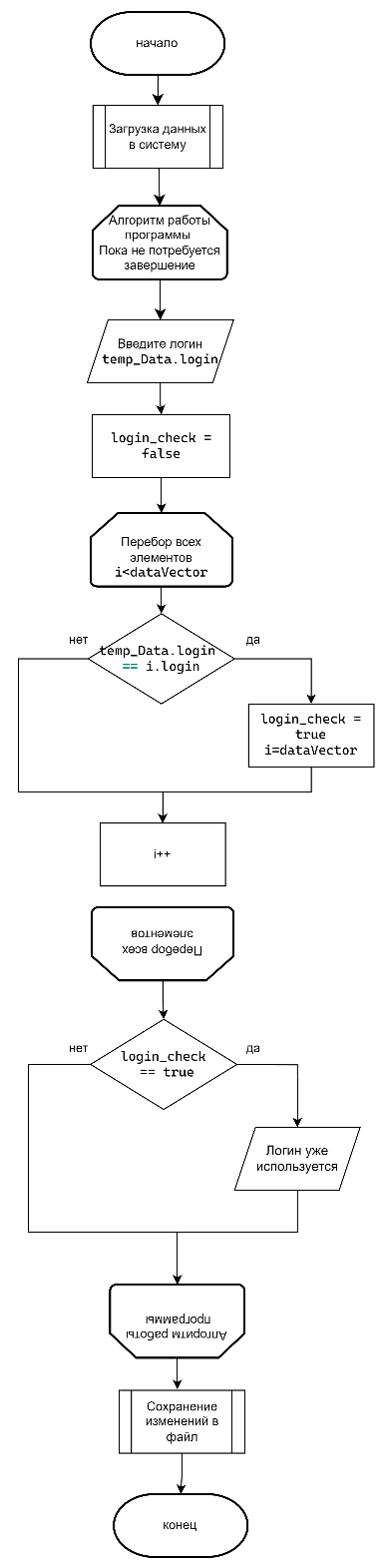


Рисунок 2.7 – Алгоритм функции addUser()

Сначала в функции Sort\_name() происходит загрузка данных в систему. Позже вводится логин и объявляется переменная, которая будет запоминать, существует ли такой логин уже или нет. Дальше идёт проверка есть ли уже такой логин и в случае положительного результата выводится сообщение о том, что такой логин уже используется. После успешного добавления нового пользователя данные сохраняются в файл.

**3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

Проверка данных обеспечивает надежность и безопасность программы, предотвращая возможные ошибки и непредвиденные ситуации. В случае некорректного ввода будет возможность ввести правильные данные. Ниже приведены конкретные примеры вводе некорректных данных.

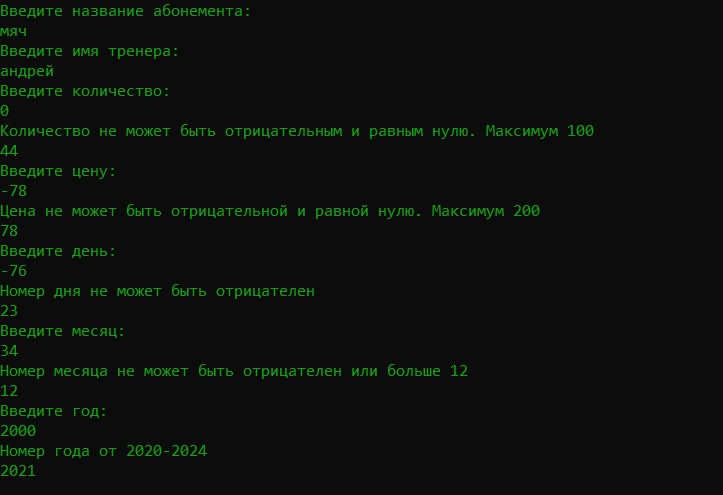


Рисунок 3.1 – Ввод некорректных данных

При попытки удалить записи под номером, которого не существует, программа выведет сообщение о том, что был осуществлён выход за допустимый предел и так же будет выведен допустимый диапазон.

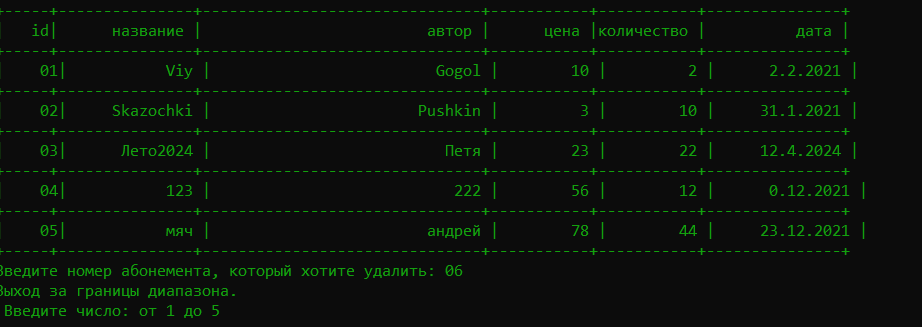


Рисунок 3.2 – Удалить записи под номером, которого не существует

После попытки удаления абонемента пользователю будет отображено сообщение с вопросом, уверен ли он в своем решении, пример представлен на рисунке 3.3. Это позволит убедиться, что удаление абонемента было осознанным шагом и избежать случайных действий.

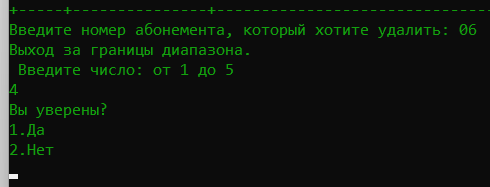


Рисунок 3.3 – Удаление абонемента

При попытке создания новой учетной записи и выборе кода роли необходимо ввести 0 или 1. Система проверяет корректность введенных данных, чтобы убедиться, что выбрано одно из допустимых значений. Если введено неверное значение, отображается сообщение о выходе за допустимый диапазон, пример представлен на рисунке 3.4.

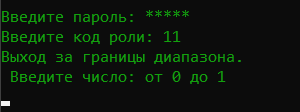


Рисунок 3.4 – Попытка ввода не существующей роли

При попытке создания новой учетной записи система проверяет корректность введенных данных. Если пользователь с таким логином уже существует, отображается сообщение, что данный логин занят, пример представлен на рисунке 3.5.

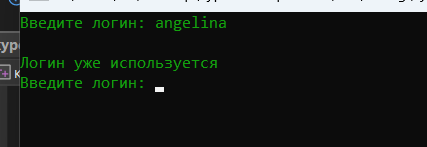


Рисунок 3.5 – Проверка на уже существующий логин

В меню редактирования учетной записи, при вводе недопустимого пункта, выводится сообщение о выходе за допустимый диапазон, а также отображается сам допустимый диапазон, пример представлен на рисунке 3.6.

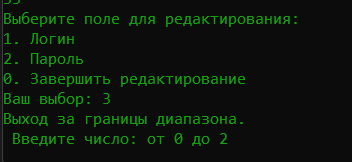


Рисунок 3.6 – Меню редактирования учётной записи

Если пользователю необходимо просмотреть абонементы, а в файле ничего нет, то ему будет выведена сообщение, в которой будет написано, что список пуст, пример представлен на рисунке 3.7.



Рисунок 3.7 – Просмотр пустого списка абонементов

При покупке абонемента у пользователя спрашивают, сколько абонементов он хочет купить, а также, заинтересован ли он в приобретении по специальной цене, пример представлен на рисунке 3.8.

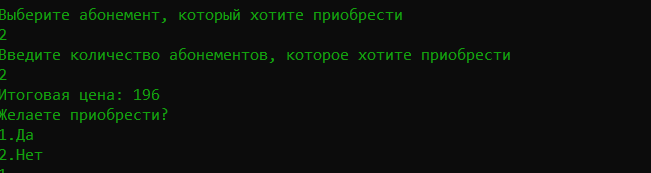


Рисунок 3.8 – Проверка количества абонементов

Таким образом, в этой главе рассмотрены функции системы по обработке неверных данных. Описаны способы и стратегии обработки ошибок, а также методы предотвращения и исправления неправильного ввода, что повышает надежность и стабильность системы в целом.

1. **ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР**
   1. **Авторизация**

В данной части работы представлен пример работы программы. На рисунке 4.1 представлено меню при входе в программу.

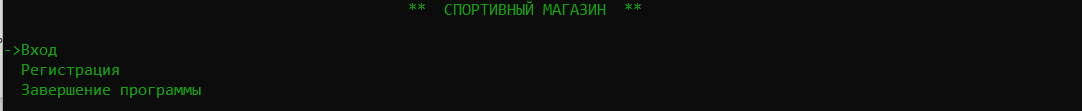


Рисунок 4.1 – Меню при входе в программу

При выборе варианта "Вход" открывается возможность войти в систему. При выборе регистрации можно зарегистрироваться как обычный пользователь или как администратор. Пункт "Выход" позволяет завершить работу в программе.

Непосредственный вход в аккаунт как пользователя представлен на рисунке 4.2.

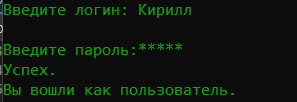


Рисунок 4.2 – Авторизация

Непосредственная регестрация аккаунта представлен на рисунке 4.3.



Рисунок 4.3 – Регистрация пользователя

Функционал системы для разных режимов входа сильно отличается, поэтому последовательно рассмотрим все.

**4.2 Модуль администратора**

При входе в систему под именем администратора перед пользователем появится главное меню администратора, которое включает в себя пункты: пользователи, база данных спортивного клуба и выход.

Меню для работы администратора представлено на рисунке 4.4.

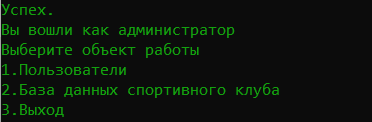


Рисунок 4.4 – Меню администратора

При переходе по первому пункту «Пользователи» система выводит меню с помощью которого можно работать с учётными записями, пример представлен на рисунке 4.5.

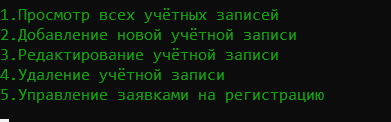


Рисунок 4.5 – Меню управления учетными записями пользователей

При переходе по второму пункту «Работа с данными» система выводит меню, в котором администратор может выбрать между двумя возможностями режима редактирования и режима обработки, пример представлен на рисунке 4.6.

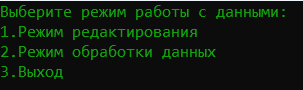


Рисунок 4.6 – Меню работы с данными фильмов

В свою очередь меню работы с данных фильмов состоит из двух пунктов, а именно из режима редактирования и режима обработки.

Режим редактирования подразумевает под собой просмотр всех абонементов. В данных о абонементе находится название абонемента, имя тренера, его стоимость, количесвто абонементов и дата проведения занятий. В режиме редактирования имеется возможность просматривать, добавлять, удалять и редактировать данные об абонементах, пример представлен на рисунке 4.7. В режиме обработки имеется возможность искать абонементы по названию абонемента и по его тренеру, и по ценк , а так же сортировать их по названию, и тренерустоимости, пример представлен на рисунке 4.8.

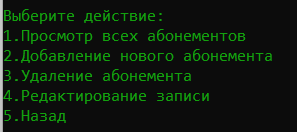


Рисунок 4.7 – Меню режима редактирования

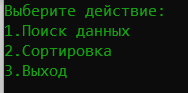


Рисунок 4.9 – Меню режима обработки данных

Просмотр данных осуществляется в виде таблицы, представленной на рисунке 4.10. Таблица предоставляет удобный и систематизированный метод представления информации, что облегчает анализ данных и улучшает их понимание. В ней отображаются данные об абонементах, а именно их ID, название, тренер, цена, дата занятия и количество абонементов.

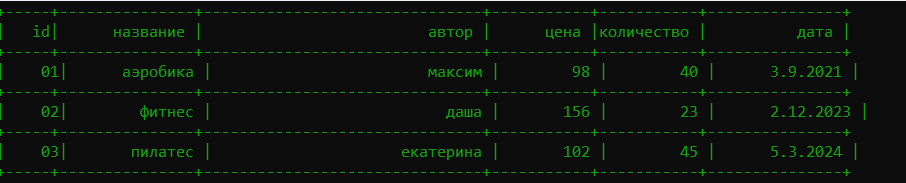


Рисунок 4.10 – Просмотр данных об абонементах

Когда надо добавить новый абонемент, данные о нем должны быть внесены в базу, за выполнение этого отвечает пункт 2 меню «Режим редактирования». При заполнении всех полей система сообщит об успешном завершении операции. Пример успешно завершенной операции добавления приведен на рисунке 4.11.

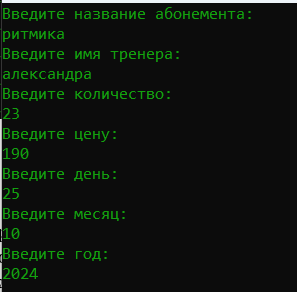
****

Рисунок 4.11 – Добавление абонементов

Перейдя по пункту один «Просмотр всех абонементов», можно просмотреть добавленный абонемент в список всех абонементов в фитнес-центре (рисунок 4.12).

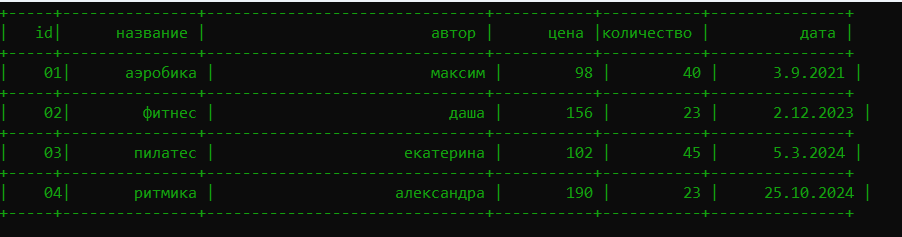


Рисунок 4.12 – Вывод всех абонементов в табличной форме

Удаление абонемента осуществляется путем указания его ID, в соответствии с представлением на рисунке 4.13. Это означает, что для удаления конкретного абонемента необходимо указать его ID, что позволяет точно идентифицировать и удалить нужную запись.

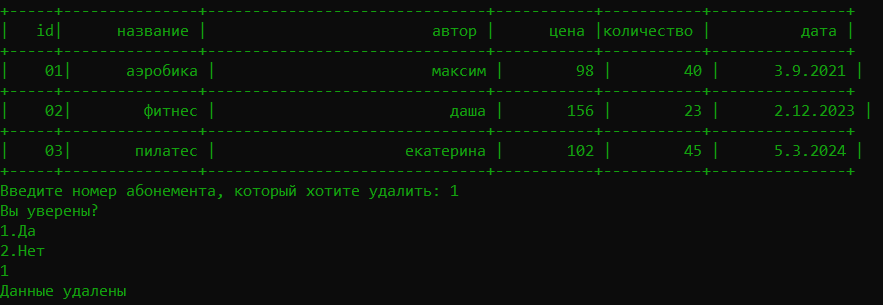


Рисунок 4.13 – Удаление абонемента по его ID

Для редактирования записи необходимо произвести поиск абонемента по его ID. После того как абонемент будет найден, можно внести необходимые изменения в соответствии с требованиями. Такой метод обеспечивает точное и эффективное редактирование данных, поскольку основывается на ID каждой записи.

На рисунке 4.14 представлен процесс редактирования данных абонемента.

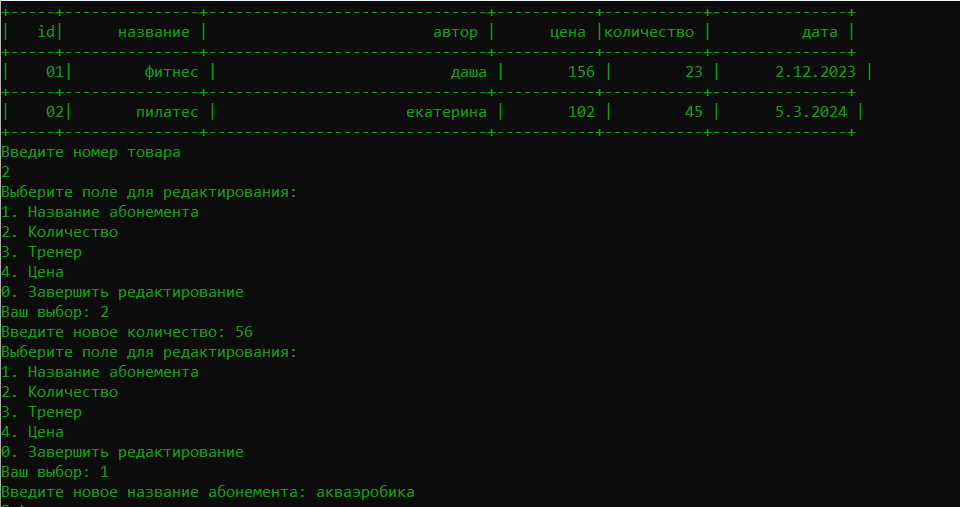


Рисунок 4.14 – Процес редактирования абонемента

На рисунке 4.15 представлена таблица с уже изменённой фамилией пользователя.

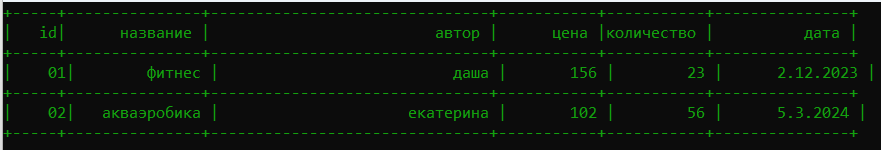


Рисунок 4.15 – Отображение изменений в таблице абонементов

Просмотр учётных записей осуществляется в виде таблицы, представленной на рисунке 4.16. В ней отображаются данные об абонементах, а именно их ID, название, тренер, цена, дата занятия и количество абонементов.

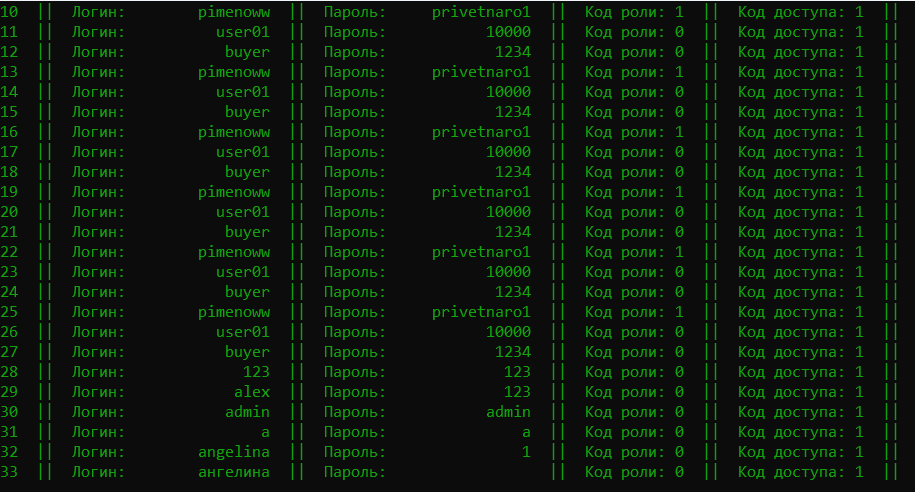


Рисунок 4.16 – Отображение учётных записей в табличной форме

После перехода по пункту "Управление заявками на регистрацию" администратор имеет возможность как одобрить, так и отклонить заявку пользователя. Это означает, что администратор может просматривать заявки, поступившие от пользователей, и принимать решение о допуске или отказе в регистрации. Такой функционал обеспечивает администратору контроль над процессом регистрации пользователей и позволяет поддерживать эффективное управление системой. Процесс одобрения заявок пользователей на регистрацию представлен на рисунке 4.17.

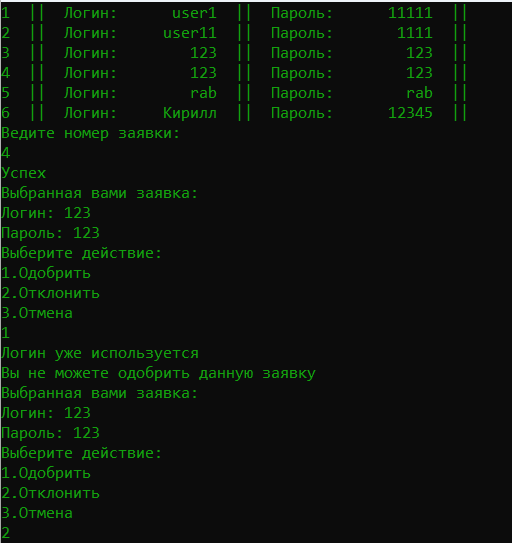


Рисунок 4.17 – Процес одобрения заявок на регистрацию

**4.3 Модуль пользователя**

Меню пользователя позволяет просматривать все абонементы, которые есть, искать по 3-ём параметрам и сортировать данные по 3-ём параметрам, а так же покупать абонемент.

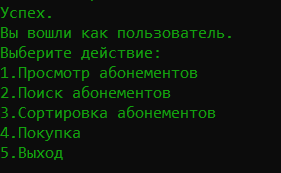


Рисунок 4.18 – Меню пользователя

Для удобной ориентации в списках системы были предусмотрены функции сортировки и поиска данных. Меню сортировки представлено на рисунке 4.19.

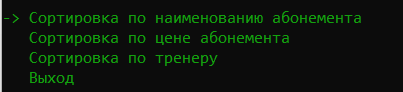


Рисунок 4.19 – Меню сортировок

Для удобства сортировка возможна по трём параметрам, а именно по цене, по тренеру и по названию.

Результат сортировки абонементов по названию представлен на рисунке 4.20.

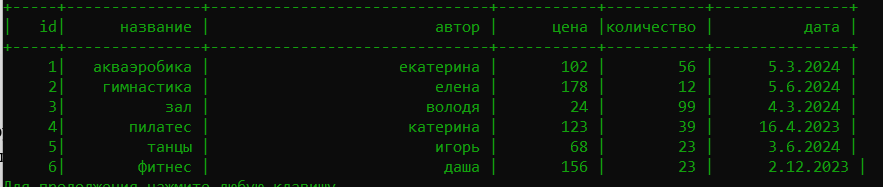


Рисунок 4.20 – Результат сортировки абонементов по названию

Результат сортировки абонементов по цене представлен на рисунке 4.21.

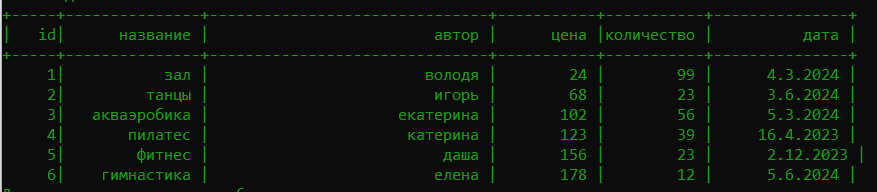


Рисунок 4.21 – Результат сортировки абонементов по цене

Результат сортировки абонементов по тренеру представлен на рисунке 4.22.

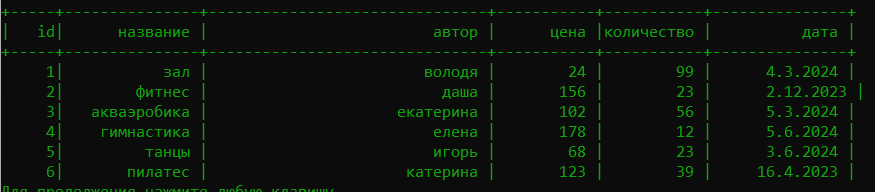


Рисунок 4.22 – Результат сортировки абонементов по тренеру

Поиск осуществляется по нескольким параметрам, а именно по названию, тренеру и цене. Меню поиска представлено на рисунке 4.23. Пользователь может указать тренера, у которого хочет заниматься и система отобразит абонементы под руководством этого тренера.

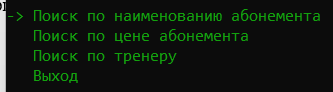


Рисунок 4.23 – Меню поиска абонементов

Результат поиска абонемента по названию представлен на рисунке 4.24.

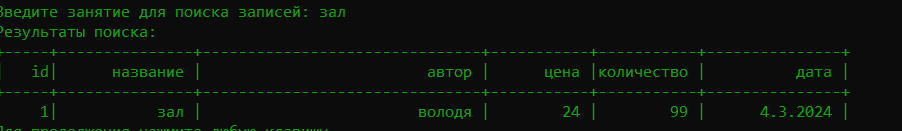


Рисунок 4.24 – Поиск абонементов по названию

Результат поиска абонемента по цене представлен на рисунке 4.25.

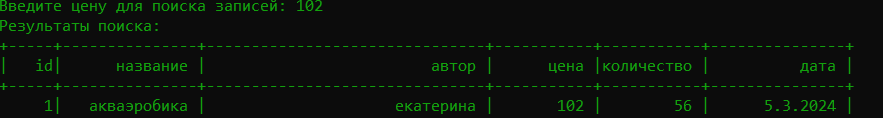


Рисунок 4.25 – Поиск абонементов по цене

Результат поиска абонемента по тренеру представлен на рисунке 4.26.

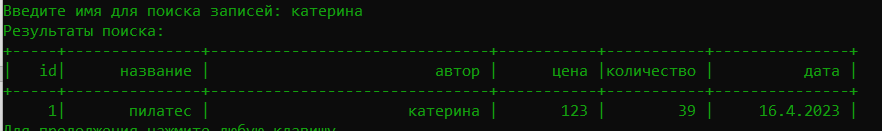


Рисунок 4.26 – Поиск абонементов по тренеру

После того как пользователь выбрал желаемый абонемент из списка доступных в фитнес-центре, он переходит к процессу покупки абонемента. На рисунке 4.27 изображен этап покупки абонемента. Изначально пользователь должен выбрать абонемент из предложенного списка и ввести его номер. Затем пользователь должен ввести количество абонентов, которое хочет приобрести. А после подтвердить покупку и ввести ФИО.

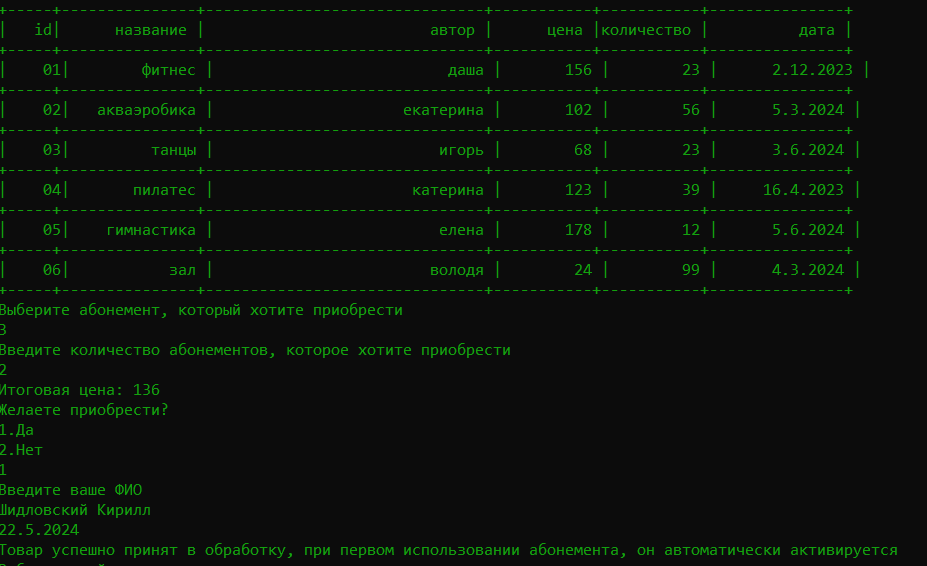


Рисунок 4.27– Процесс покупки абонемента

Таким образом в главе были описаны возможности разработанной системы, использованной с разных сторон пользователем и администратором.

**ЗАКЛЮЧЕНИЕ**

В результате выполнения курсового проекта был тщательно изучен процесс учёта деятельности фитнес-центра. В ходе работы была создана информационная система, оптимизирующая учет и управление деятельностью фитнес-центра.

Система включает возможности работы в различных ролях, таких как администратор и клиент, а также функционал для записи купленных абонементов и хранения заявок на регистрацию в отдельных файлах. Разработанная система обладает ключевыми характеристиками: информативность, эффективность, мобильность и безопасность. Особое внимание уделено безопасному хранению личных данных пользователей.

Эта система не только улучшает процессы управления и учета, но также обеспечивает удобство и простоту использования, делая её доступной для широкого круга пользователей. Программа может быть легко адаптирована для работы на различных устройствах и операционных системах, что обеспечивает её универсальность.

В приведенных разделах был исследован процесс учёта деятельности фитнес-центра. Описана тенденция развития и роль фитнес-центров в современном мире. Проведен анализ процесса учёта деятельности фитнес-центра и этот же процесс предоставлен в спецификации IDEF0. Разработаны диаграммы UML, а именно диаграмма состояния, компонентов приложения, последовательности процессов заказа сладостей. Представлена спецификация вариантов использования для роли пользователя и администратора. Разработаны блок-схемы алгоритмов для функций «добавления учётной записи» и «главного меню пользователя».

Таким образом путём решения следующих задач: провести анализ существующих методов учета деятельности спортивного клуба и выявить их недостатки, изучить потребности и требования спортивного клуба к системе учета, спроектировать и разработать автоматизированную систему учета деятельности спортивного клуба, провести тестирование и внедрение разработанной системы, , поставленная цель — упростить документооборот в спортивном клубе и облегчить учет спортивного инвентаря и прочего оборудования — была успешно достигнута

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] https://cyberleninka.ru/article/n/rol-sportivnogo-kluba-v-populyarizatsii-fizicheskoy-kultury-i-sporta-v-vuze

[2] <https://files.student-it.ru/previewfile/24204/4>

[3] Методология IDEF0 [Электронный ресурс]. – Режим доступа: .. \трпо\ТРПОЛабораторная работа №2.docx

[4] Методическое пособие для выполнения лабораторной роботы «Язык Uml. Диаграммы UML» [Электронный ресурс]. – Режим доступа: .. \трпо\ТРПО - Лабораторные работы №7-8.docx

[5] ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем.

# Приложение А

**(обязательное)**

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

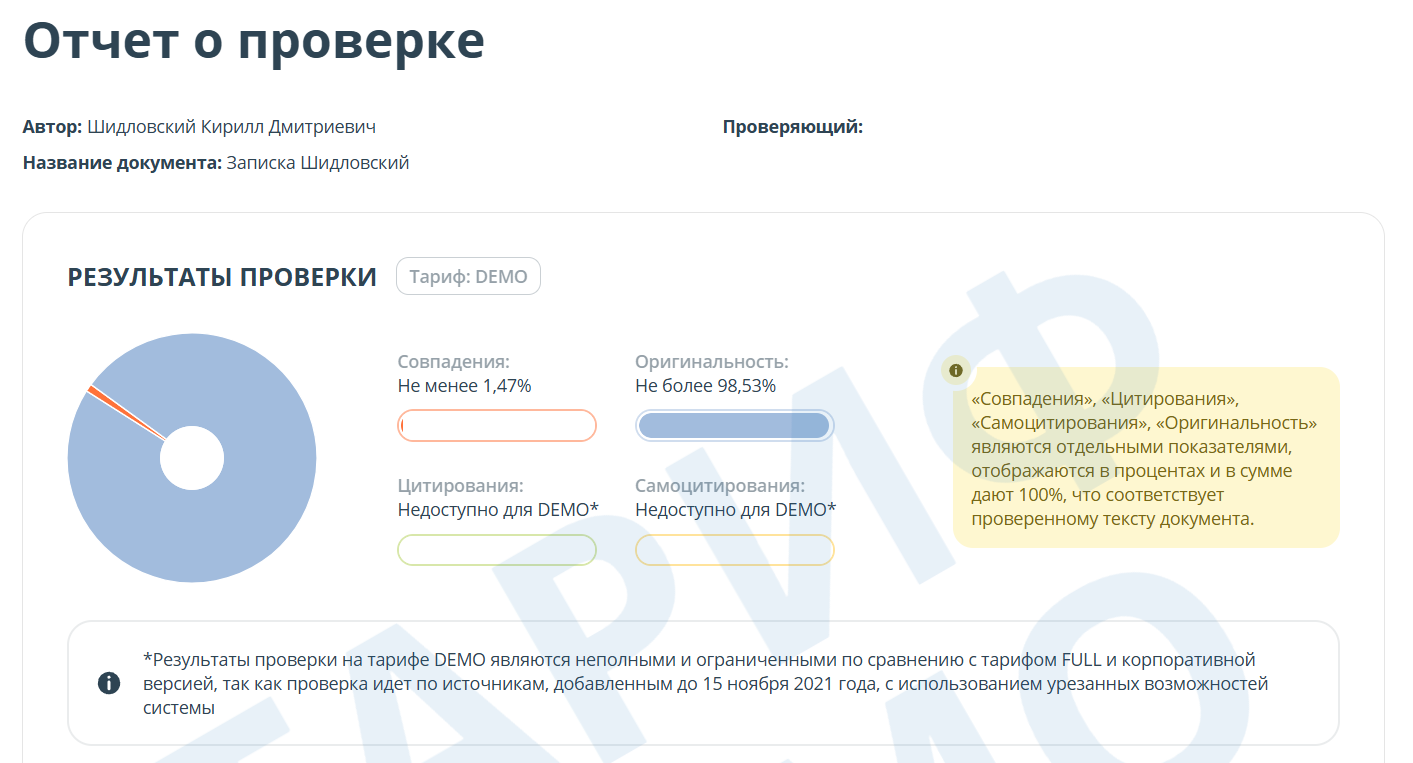
****

Рисунок А.1 – Отчет о проверке на заимствования в системе «Антиплагиат»

# Приложение Б

**(обязательное)**

**Листинг кода алгоритмов, реализующих основную бизнес-логику приложения**

vector <Data> dataVector;

string login1, password;

Data TempData;

namespace console

{

const std::string message{ " \*\* ПРИВЕТСТВУЕМ В АВТОМАТИЗИРОВАННОЙ СИСТЕМЕ УЧЕТА ДЕЯТЕЛЬНОСТИ СПОРТИВНОГО КЛУБА \*\* \n" };

void print(const std::string& text)

{

std::cout << text << std::endl;

}

}

using namespace console;

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(handle, FOREGROUND\_GREEN);

TempData.startMenu();

return 0;

}

void setCursorPosition(int x, int y) {

std::cout << "\033[" << y << ";" << x << "H";

}

void fakeLoading() {

const char animation[] = "|/-\\";

const int numFrames = sizeof(animation) - 1;

const int numDots = 10;

int consoleWidth = 100; // Ширина консоли

int consoleHeight = 65; // Высота консоли

int centerX = consoleWidth / 2;

int centerY = consoleHeight / 2;

int startPosX = centerX - numDots / 2;

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

setCursorPosition(startPosX, centerY);

Продолжение приложения Б

std::cout << "Loading [";

for (int j = 0; j < numDots; ++j) {

if (j < i) {

std::cout << ".";

}

else {

std::cout << " ";

}

}

std::cout << "] " << animation[i % numFrames] << "\r";

std::this\_thread::sleep\_for(std::chrono::milliseconds(200));

}

setCursorPosition(startPosX, centerY);

std::cout << "Loading complete! " << std::endl;

}

void fakeLoading1() {

const char animation[] = "|/-\\";

const int numFrames = sizeof(animation) - 1;

const int numDots = 10;

int consoleWidth = 80; // Ширина консоли

int consoleHeight = 20; // Высота консоли

int centerX = consoleWidth / 2;

int centerY = consoleHeight / 2;

int startPosX = centerX - numDots / 2;

// Вывод ASCII-арт для надписи "BAN"

setCursorPosition(centerX + 12, centerY - 2);

std::cout << "##### ##### # #";

setCursorPosition(centerX + 12, centerY - 1);

std::cout << "# # # # ## #";

setCursorPosition(centerX + 12, centerY);

std::cout << "#### ##### # # #";

setCursorPosition(centerX + 12, centerY + 1);

std::cout << "# # # # # ##";

setCursorPosition(centerX + 12, centerY + 2);

std::cout << "##### # # # #";

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

setCursorPosition(startPosX + 16, centerY + 4);

std::cout << "Loading [";

for (int j = 0; j < numDots; ++j) {

if (j < i) {

std::cout << ".";

}

else {

Продолжение приложения Б

std::cout << " ";

}

}

std::cout << "] " << animation[i % numFrames] << "\r";

std::this\_thread::sleep\_for(std::chrono::milliseconds(200));

}

setCursorPosition(startPosX, centerY + 4);

std::cout << "НАПОМИНАЕМ, НЕ ЗАБЫВАЙТЕ ПРАВИЛЬНО ВВОДИТЬ СВОИ ДАННЫЕ " << std::endl;

}

void Data::startMenu()

{

Sleep(250);

cout << "\t";

print(message);

cout << " "; fakeLoading();

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

CreationHash();

creationFirstAdmin();

//WaitingData temp;

int choice = 0; // Use this variable to track the user's choice

//int amount = 3;

while (true)

{

system("cls");

cout << " \*\* СПОРТИВНЫЙ МАГАЗИН \*\* \n\n";

// Display menu options with arrow key navigation

for (int i = 1; i <= 3; ++i)

{

if (choice == i - 1)

cout << "->";

else

cout << " ";

switch (i)

{

case 1: cout << "Вход\n"; break;

case 2: cout << "Регистрация\n"; break;

case 3: cout << "Завершение программы\n"; break;

}

}

int code = \_getch();

if (code == 224)

Продолжение приложения Б

{

code = \_getch();

if (code == 80) // Down arrow key

choice = (choice + 1) % 3;

else if (code == 72) // Up arrow key

choice = (choice + 2) % 3;

}

else if (code == 13) // Enter key

{

// Process the selected option

switch (choice + 1)

{

case 1: // Вход

enter();

// Implement the login functionality here

continue;

case 2: // Регистрация

registration();

// Implement the registration functionality here

break;

case 3: // Завершение программы

system("cls");

// Save data and perform cleanup

endProgram();

break;

}

}

}

}

void Data::enter()

{

system("cls");

cout << "Введите логин: ";

cin >> login1;

cin.clear();

cin.ignore(32767, '\n');

cout << "\nВведите пароль:";

password = Crypt(inputPassword());

cout << '\n';

int amount = 3;

bool checker = false;

Vault t;

WaitingData temp;

for (auto i : dataVector) //проверка, кто пытается зайти пользователь, администратор + проверяет не заблокирован ли он, если данные неверные выводит соответсвующее сообщение

{

if (login1 == i.login && password + i.salt == i.saltedHashPassword && i.role == 0 && i.access == 1)

Продолжение приложения Б

{

cout << "Успех.\nВы вошли как пользователь.\n";

amount = 1;

t.userMenu();

checker = true;

break;

}

else if (login1 == i.login && password + i.salt == i.saltedHashPassword && i.role == 1)

{

cout << "Успех.\nВы вошли как администратор\n";

temp.adminMenu();

amount = 1;

checker = true;

break;

}

else if (login1 == i.login && password + i.salt == i.saltedHashPassword && i.access == 0)

{

cout << "Извините, но у данного аккаунта нет доступа к базе данных\n";

checker = true;

amount = 1;

break;

}

}

if (checker == false) // если логин или пароль введён неверно выдаёт соответствующее сообщение и уменьшает количество попыток ввода

{

cout << "Неверный логин или пароль\n";

amount--;

if (amount == 0) // если количество попыток ввода на нуле, выводит соответствующее сообщение и показывает изображение

{

cout << "Превышенно количество попыток неверного ввода логина или пароля\n";

fakeLoading1();

amount = 1;

}

else

cout << "Осталось попыток ввода: " << amount << endl;

}

system("pause");

}

void Data::registration()

{

system("cls");

while (true)

Продолжение приложения Б

{

cout << "Введите логин : ";

cin >> login;

cin.clear();

cin.ignore(32767, '\n');

bool loginCheck = false;

for (auto i : dataVector) // проверка, занят ли логин

{

if (login == i.login)

{

loginCheck = true;

break;

}

}

if (loginCheck == true)

{

cout << "Логин уже используется\n";

}

else break;

}

cout << "Введите пароль: ";

password = inputPassword();

fstream waitingData;

waitingData.open("waiting\_data.txt", ios::in | ios::app);

waitingData << login << '\n' << password << '\n';

waitingData.close();

cout << "\nВы зарегистрировались.\nЖдите подтверждения учётной записи\n\n";

}

void Data::endProgram()

{

fstream dataFile;

dataFile.open("data.txt", ios::out | ios::trunc);

for (auto i : dataVector)

{

dataFile << i.login << '\n' << i.saltedHashPassword << '\n' << i.salt << '\n' << i.role << '\n' << i.access << '\n';

}

dataFile.close();

dataVector.clear();

exit(0); // Terminate the program

}

void Data::creationFirstAdmin()

{

cin.clear();

ifstream data;

string specialPassword;

data.open("data.txt", ios::in);

if (!data || data.peek() == EOF)

Продолжение приложения Б

{

data.close();

cout << "Файл с данными логинов и паролей не обнаружен.\nПожалуйста введите специальный пароль для создания первого администратора)\n";

while (true)

{

cin >> specialPassword;

if (specialPassword == "sport")

{

cout << "Успех!!!\nВведите данные первого администратора\nЛогин: ";

cin >> login;

cin.clear();

cin.ignore(32767, '\n');

cout << "\nПароль должен содержать только цифры и латинские строчные символы\nПароль: ";

password = Crypt(inputPassword());

string salt = RandomString();

salt = Crypt(salt);

cout << '\n';

ofstream data;

data.open("data.txt", ios::out);

data << login << '\n' << password + salt << '\n' << salt << '\n' << '1' << '\n' << '1' << '\n';

cout << "Файл с данными успешно создан.\nДля соблюдения установленных норм взаимодействия с программой программа завершит свою работу.\nПожалуйста перезайдите\n";

exit(0);

}

else

{

cout << "Неправильно, для повторного ввода нажмите любую клавишу (esc на выход)\n";

if (\_getch() == 27)

{

exit(0);

}

}

}

}

int count = 0;

string temp;

bool endStruct = false;

while (getline(data, temp))

{

switch (count)

{

case 0:

endStruct = false;

Продолжение приложения Б

TempData.login = temp;

break;

case 1:

TempData.saltedHashPassword = temp;

break;

case 2:

TempData.salt = temp;

break;

case 3:

TempData.role = atoi(temp.c\_str());

break;

case 4:

TempData.access = atoi(temp.c\_str());

endStruct = true;

count = -1;

break;

}

if (endStruct == true)

{

dataVector.push\_back(TempData);

endStruct = false;

}

count++;

}

data.close();

}

string Data::inputPassword()

{

string password1;

while (1)

{

int letter = 0;

while (1)

{

letter = \_getch();

if (letter == 13)

{

break;

}

if (letter == 27)

{

exit(0);

}

if (letter == 8)

{

cout << (char)8 << ' ' << char(8);

if (!password1.empty())

password1.erase(password1.length() - 1);

}

Продолжение приложения Б

else

{

cout << '\*';

password1 += (char)letter;

}

}

return (string)password1;

extern string allLetterAndNumbers;

bool passwordCheckSymbols = true;

for (int i = 0; i < password1.length(); i++)

{

bool password\_check = false;

for (int j = 0; j < allLetterAndNumbers.length(); j++)

{

if (password1[i] == allLetterAndNumbers[j])

{

password\_check = true;

break;

}

}

if (password\_check == false)

{

passwordCheckSymbols = false;

break;

}

}

if (passwordCheckSymbols == false)

{

cout << "\nОшибка ввода";

continue;

}

break;

}

}

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | | Наименование | | Дополнительные сведения | | | |
|  | | | | | Текстовые документы | |  | | | |
|  | | | | |  | |  | | | |
| БГУИР КР 6-05-0611-01 015 ПЗ | | | | | Пояснительная записка | | 59 с. | | | |
|  | | | | |  | |  | | | |
|  | | | | | Графические документы | |  | | | |
|  | | | | |  | |  | | | |
| ГУИР 425751.001 ПД | | | | | Схема алгоритма | | Формат А4 | | | |
|  | | | | | автоматизированной | |  | | | |
|  | | | | | системы учёта | |  | | | |
|  | | | | | деятельности фитнес-центра | |  | | | |
| ГУИР 425751.002 ПЛ | | | | | Описание процесса | | Формат А4 | | | |
|  | | | | | «Система учёта деятельности | |  | | | |
|  | | | | | спортивного комплекса» | |  | | | |
| ГУИР 425751.003 ПЛ | | | | | Цели и задачи | | Формат А4 | | | |
|  | | | | | автоматизированной системы | |  | | | |
|  | | | | | учёта деятельности | |  | | | |
|  | | | | | фитнес-центра | |  | | | |
| ГУИР 425751.004 ПЛ | | | | | Результаты проектирования | | Формат А4 | | | |
|  | | | | | автоматизированной системы | |  | | | |
|  | | | | | учёта деятельности | |  | | | |
|  | | | | | фитнес-центра | |  | | | |
| ГУИР 425751.005 ПЛ | | | | | Модели представления | | Формат А4 | | | |
|  | | | | | автоматизированной системы | |  | | | |
|  | | | | | учёта деятельности | |  | | | |
|  | | | | | фитнес-центра | |  | | | |
|  |  |  |  |  | БГУИР КР 6-05-0611-01 015 Д1 | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Разработка автоматизированной системы учета деятельности фитнес-центра  Ведомость курсовой работы | Лит | | | Лист | Листов |
| Разраб. | | Шидловский |  |  |  | Т |  | 1 | 1 |
| Пров. | | Снопок |  |  | Кафедра ЭИ  гр. 378101 | | | | |
| Т.контр. | | Снопок |  |  |
| Рец. | |  |  |  |
| Н.контр | |  |  |  |
| Утв. | | Ефремов |  |  |
|  | |  |  |  |