Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Технологии разработки программного обеспечения»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Руководитель курсового проекта  Ассистент кафедры ЭИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.И.Полоско |
|  | \_\_\_.\_\_\_\_.2022 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовой работе

на тему:

**«Разработка программы учета продаж театральных билетов»**

БГУИР КР 1-40 05 01-08 028 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 173602  Нехай Никита Александрович  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовая работа представлена на проверку \_\_\_.\_\_\_\_.2022  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2022

**РЕФЕРАТ**

БГУИР КР 1-40 05 01-08 028 ПЗ

**Нехай, Н.А.** Разработка программы учёта продаж театральных билетов/Н.А. Нехай. – Минск: БГУИР, 2022. – 50 с.

Пояснительная записка 50 с., 50 рис., 10 табл., 6 источников, 3 приложения

УЧЁТ продаж театральных билетов, СВЕДЕНИЯ о перформансах, театрах, записях, пользователях, модели *UML*, *IDEF*0, *BPMN*, схемы алгоритмов, программное средство

*Цель* *курсовой работы*: совершенствование процесса учета продаж о театральных билетах. Программа позволяет вносить корректировки в уже имеющиеся данные, создавать новые записи, удалять старые и сохранять отредактированные записи.

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

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

Программный продукт разработан на языке *C* с применением *MS Visual Studio 2019*.

*Область применения результатов*: c помощью разработанного программного средства можно проанализировать учет продаж билетов в театре.

Разработанное программное средство полностью отвечает всем функциональным требованиям, необходимым при учете, создании и реализации представлений и мероприятий в театре.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 1](#_Toc104280231)

[1 АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА 2](#_Toc104280232)

[1.1 Описание предметной области 2](#_Toc104280233)

[1.2 Разработка функциональной модели предметной области 2](#_Toc104280234)

[1.3 Разработка BPMN-модели основного процесса предметной области 8](#_Toc104280235)

[1.4 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 10](#_Toc104280236)

[1.5 UML-модели представления программного обеспечения и их описание 11](#_Toc104280237)

[2 ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 16](#_Toc104280238)

[2.1 Постановка задачи 16](#_Toc104280239)

[2.2 Разработка модульной структуры 18](#_Toc104280240)

[2.3 Выбор способа организации данных 18](#_Toc104280241)

[2.4 Разработка перечня пользовательских функций программы 22](#_Toc104280242)

[2.5 Разработка схем алгоритмов программы 24](#_Toc104280243)

[3 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА 30](#_Toc104280244)

[4 ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР 33](#_Toc104280245)

[4.1 Авторизация 33](#_Toc104280246)

[4.2 Модуль администратора 35](#_Toc104280247)

[4.3 Модуль пользователя 37](#_Toc104280248)

[ЗАКЛЮЧЕНИЕ 41](#_Toc104280249)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 42](#_Toc104280257)

[ПРИЛОЖЕНИЕ А 43](#_Toc104280258)

[ПРИЛОЖЕНИЕ Б 49](#_Toc104280259)

[ПРИЛОЖЕНИЕ В 50](#_Toc104280260)

# **ВВЕДЕНИЕ**

Технологии играют важную роль во всех сферах жизни. Автоматизация жизненных процессов удовлетворяет множеству потребностей человека. Особую ценность для людей имеет время. Целью информационных систем является упрощение работы человека с информацией, и тем самым происходит сокращение времени работы

На данный момент одной из самых прибыльных отраслей предоставления услуг и товаров является индустрия развлечений. Сюда можно отнести не только парки аттракционов и видео-залы с игровыми автоматами, но и популярные как среди молодежи, так и среди взрослого населения театры. Огромный спрос на интересные зрелища заставляет театры расширять диапазон демонстрируемых пьес и спектаклей, ставить какие-то новинки, что ведет к увеличению сложности и интенсивности учета продаваемых на сеансы билетов.

Как же устроена сама информационная система театра? До появления информационных технологий бюрократический аппарат и возможные средства массовой информации, как газеты, радио, листовки, помогали с взаимодействием, распространением всей выходящей информацией из театральных учреждений. Сейчас все эти процессы стали гораздо удобнее, практичнее и компактнее.

Цель данной работы - совершенствование процесса учета продаж о театральных билетах. Программа позволяет вносить корректировки в уже имеющиеся данные, создавать новые записи, удалять старые и сохранять отредактированные записи. Информационная система позволит осуществлять ввод информации о предстоящих спектаклях, составление расписания, хранение данных, быстрый поиск, составление отчетностей, а также выбор и редактирование билетов.

Объектом исследования является управление информацией о спектаклях и билетами театра.

Для достижения цели необходимо решить следующие задачи:

* Изучить работу театра;
* Сформировать на собранной информации объекты, способные реализовать необходимый функционал ИС.
* Описать работу ИС используя инструменты моделирования.
* Разработать алгоритмы работы программного средства.
* Разработать модули структуры программного средства.
* Протестировать работу программного средства.

# **АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

## **Описание предметной области**

Театр предоставляет возможность каждый день ходить на спектакли, показывая по одному спектаклю каждый день, кроме дней премьер и выходных. В такие дни показывают плановый спектакль и премьеру. Соответственно в театре каждый день находится много человек.

Театр предоставляет возможность просмотра билетов на спектакли. Для

получения информации существует приложение театра, на котором размещена информация о расписании спектаклей, описания спектаклей и, библиотека актеров и режиссеров.

Билеты можно приобрести непосредственно в кассах театра, по телефону забронировать билеты, а также в универсальных кассах по всему городу.

Цены на спектакли могу быть любыми более 0 вымышленной валюты. Цены зависят от зоны забронированного места и от спектакля - премьерный или нет. Целью администраторов является контроль учета сведений информации о пользователях, представлениях, театров и записях. То есть деятельность администрации заключается в добавлении новых данных, удалении устаревших, редактирование и просмотр данных.

## **Разработка функциональной модели предметной области**

Рассмотрим информационную систему театра, выполненную с помощью средств моделирования функций IDEF0.

Методология IDEF0 – это совокупность приёмов, правил и процедур, созданных для построения функциональной модели предметной области.

Для начала необходимо сделать контекстную модель информационной системы. Контекстная диаграмма-самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками. Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой (см. рисунок 1.1).

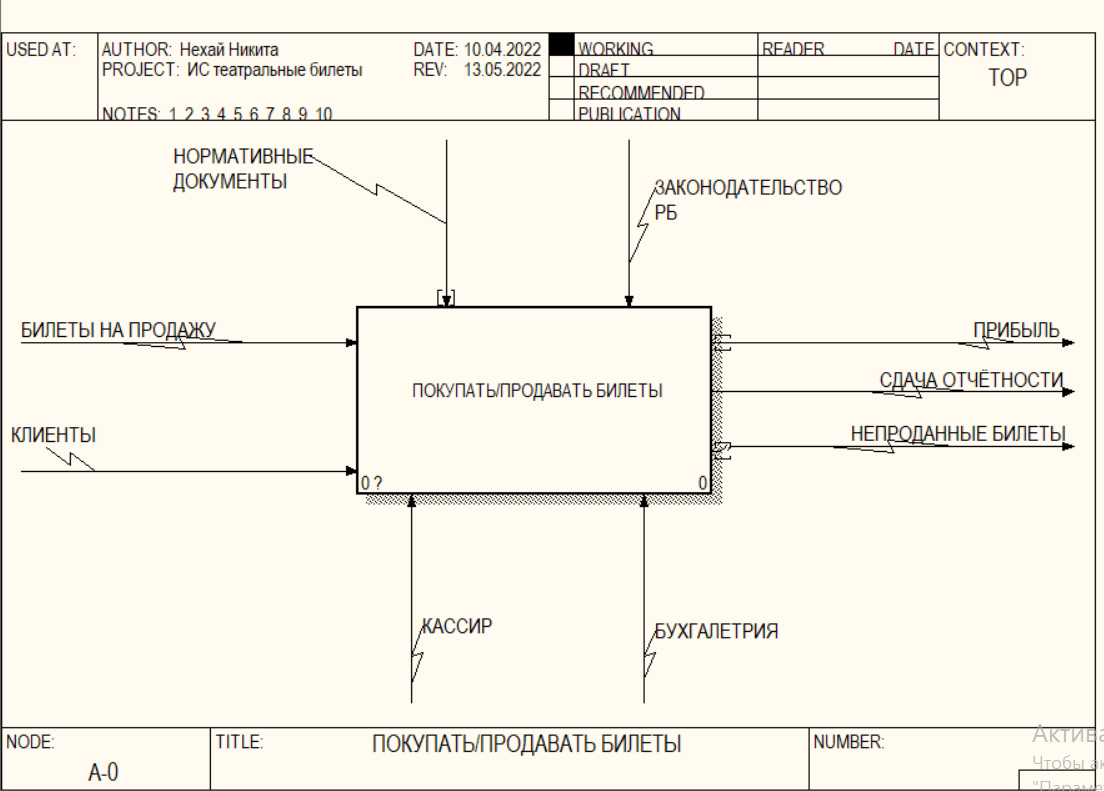


Рисунок 1.1 – Контекстная диаграмма информационной системы

Контекстная диаграмма содержит следующие параметры:

* Вход (билеты на продажу, клиенты);
* Механизм (кассир и бухгалтерия);
* Управление (законодательство РБ и нормативные документы);
* Выход (прибыль, сдача отчётности и непроданные билеты).

Нотация IDEF0 поддерживает последовательную декомпозицию процесса до требуемого уровня. Дочерняя диаграмма, охватывает ту же область, что и родительская, но описывает ее более подробно. При декомпозиции стрелки родительского процесса переносятся на дочернюю диаграмму в виде граничных стрелок. Проведем декомпозицию контекстной диаграммы, описав последовательно осуществление покупки и продажи билетов (см. рисунок 1.2).

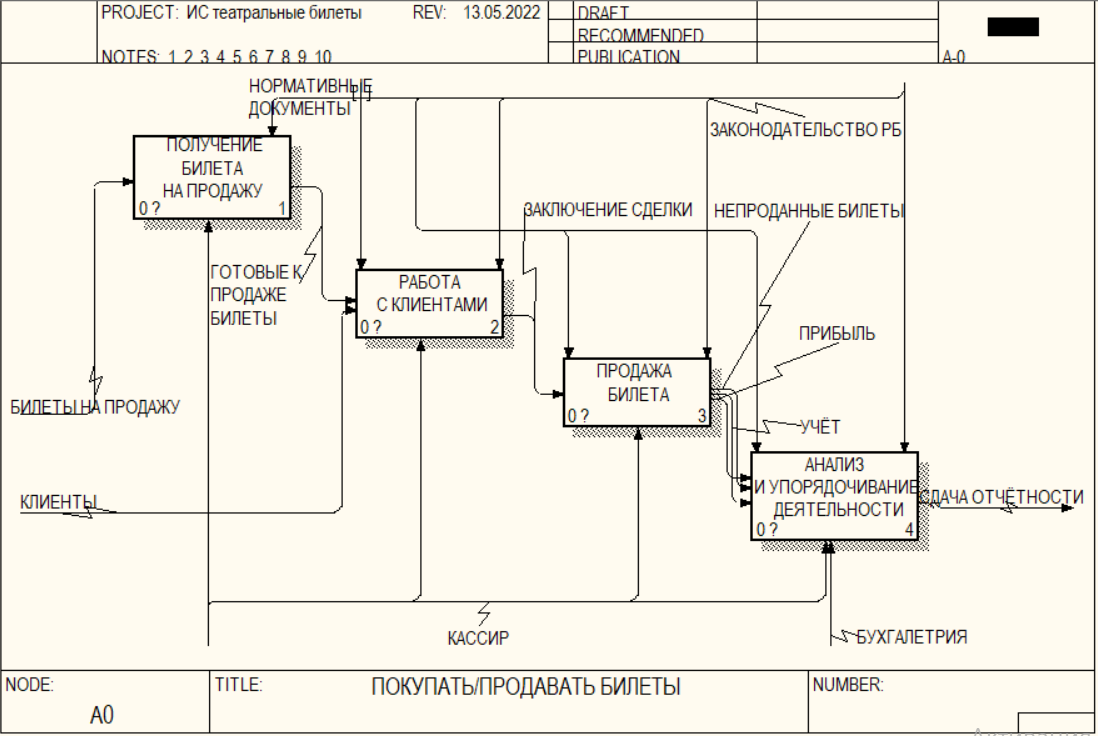


Рисунок 1.2 – Диаграмма декомпозиции «Покупать/продавать билеты»

На выходе получили диаграмму, состоящую из 4-ёх процессов:

1. Получение билета на продажу. Входные данные получаем из входных потоков.
2. Работа с клиентами. В базе данных театра хранится вся необходимая информация о билетах, представлениях, театрах и пользователях. Ввод осуществляет работник администрации.
3. Продажа билета. Происходит продажа билета клиента.
4. Анализ и упорядочивание деятельности. Происходит обновление информации, создание отчётов и статистики.

Закончив декомпозицию контекстной диаграммы, перейдем к декомпозиции диаграммы следующего уровня. Декомпозируем последовательно все блоки полученной диаграммы.

Первым шагом декомпозируем процесс «Получение билета на продажу». Первым этапом необходимо принять и обработать информацию о перформансах и доступных театрах. После формируется запись и формируется полноценный билет (см. рисунок 1.3).

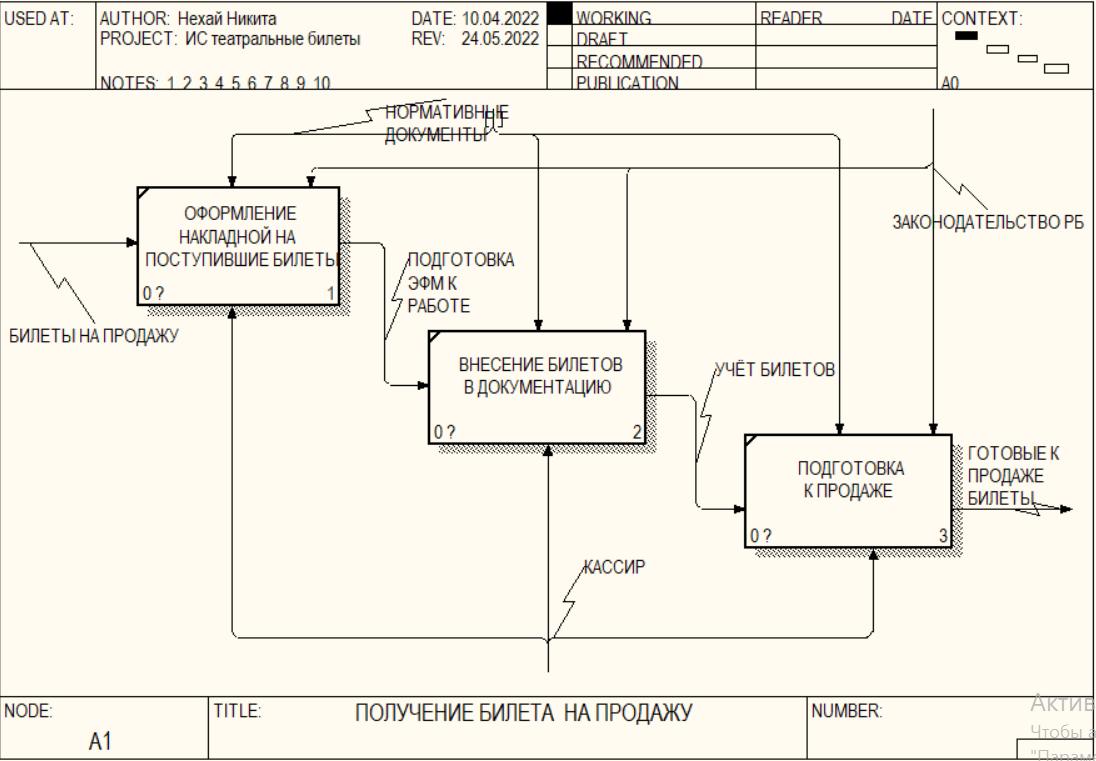


Рисунок 1.3 – Диаграмма декомпозиции процесса «Получение билета на продажу»

Вторым этапом будем декомпозировать процесс «Работа с клиентами». Получив готовые билеты и составив афишу. Театр готов к продаже билетов и работе с клиентами. Этот процесс взаимодействия постоянно происходит с актуальной информацией из БД. Выбрав нужное мероприятие и билеты происходит передача данных в БД. После работы с финансовыми переводами. Клиент информируется по e-mail или по указанному телефону (см. рисунок 1.4).

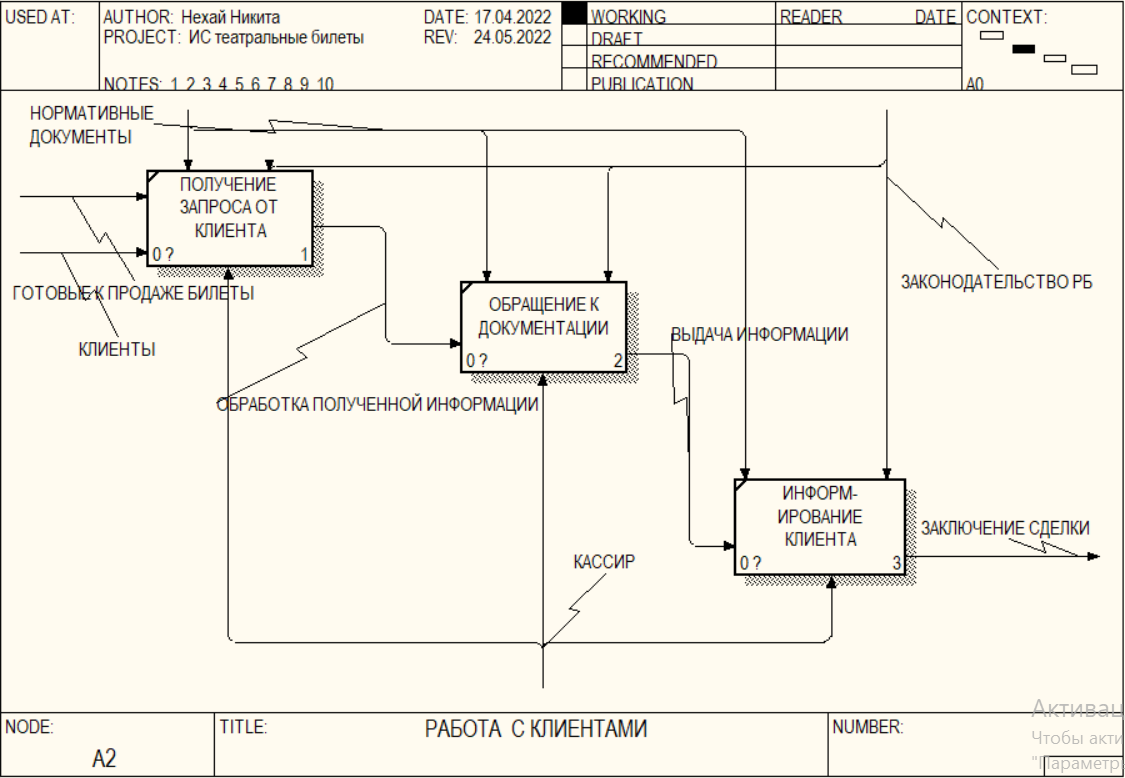


Рисунок 1.4 – Диаграмма декомпозиции процесса «Работа с клиентами»

Чтобы полностью раскрыть процесс «Продажа билета», сделаем его декомпозицию – это третий этап. После получения нужной информации от клиента из БД. Формируется временное бронирование до подтверждения транзакции. Совершив транзакцию и получив успешный перевод происходит выкуп билета. При другом исходе, временное бронирование снимается и билет вновь становится доступным к покупке, пройдя через определение «непроданного билета». Вся полученная информация от пользователя и о транзакциях передаётся далее (см. рисунок 1.5).

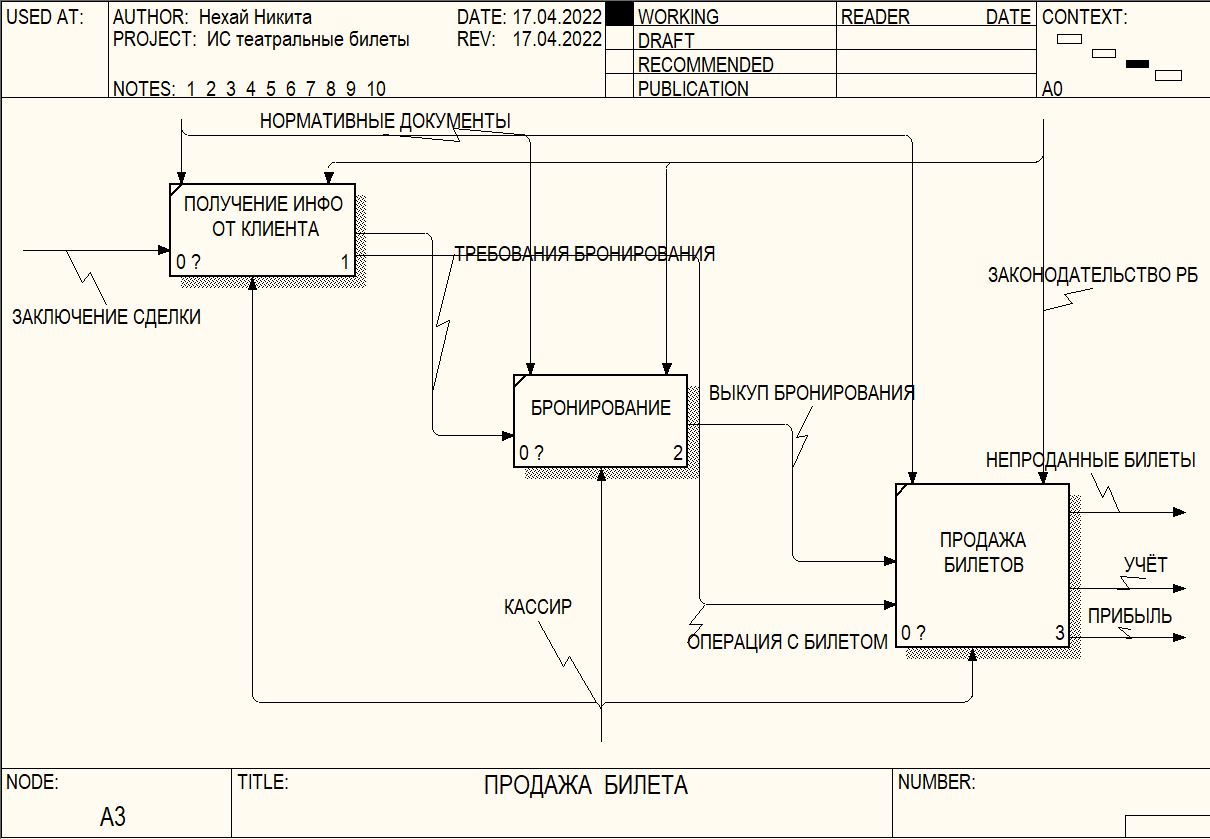


Рисунок 1.5 – Диаграмма декомпозиции «Продажа билета»

Четвертый шаг. На следующем рисунке 1.6 изображена последняя диаграмма «Анализ и упорядочивание деятельности». Осуществляется полная обработка всей информации, с которой происходило взаимодействие. Формируется отчётность, накладная и статистика. Совершается подсчёт прибыли и её разделение. Составляются сопутствующие документы (см. рисунок 1.6).

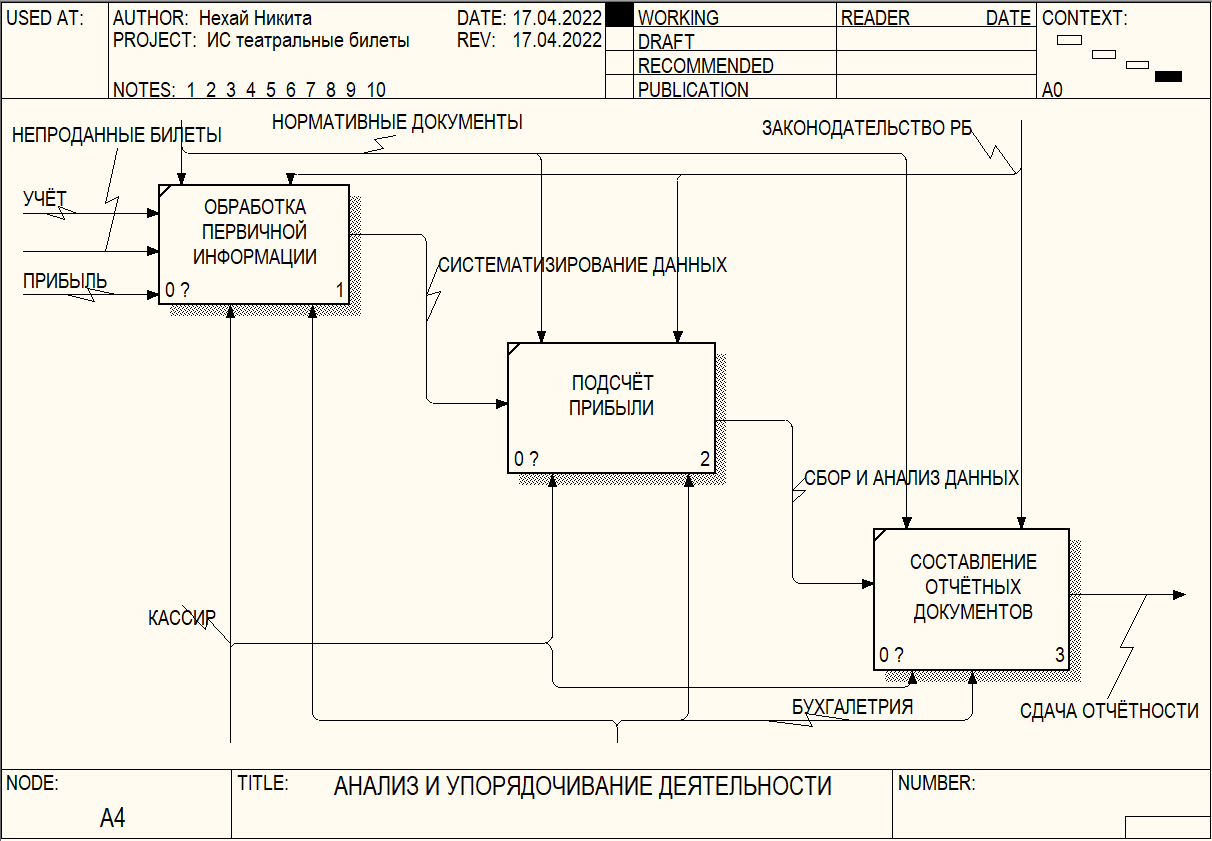


Рисунок 1.6 – Диаграмма декомпозиции процесса «Анализ и упорядочивание деятельности»

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

Business Process Model and Notation (нотация моделирования бизнес-процессов) — это программа, фокусирующая на разработке диаграмм или цепей бизнес-процессов. Также, это система условных обозначений, которая отображает бизнес-процессы с помощью блок-схем. BPMN диаграмма показывает в какой последовательности совершаются рабочие действия и перемещаются потоки информации.

Следующая часть работы заключается в построении контекстной диаграммы, которая изображены на рисунке 1.9 Построение диаграммы начинаем с нанесения на диаграмму дорожек, событий «начало» и «конец». После чего можем заполнить диаграмму действиями. Для создания развилок используем логические операторы и необходимые объекты.

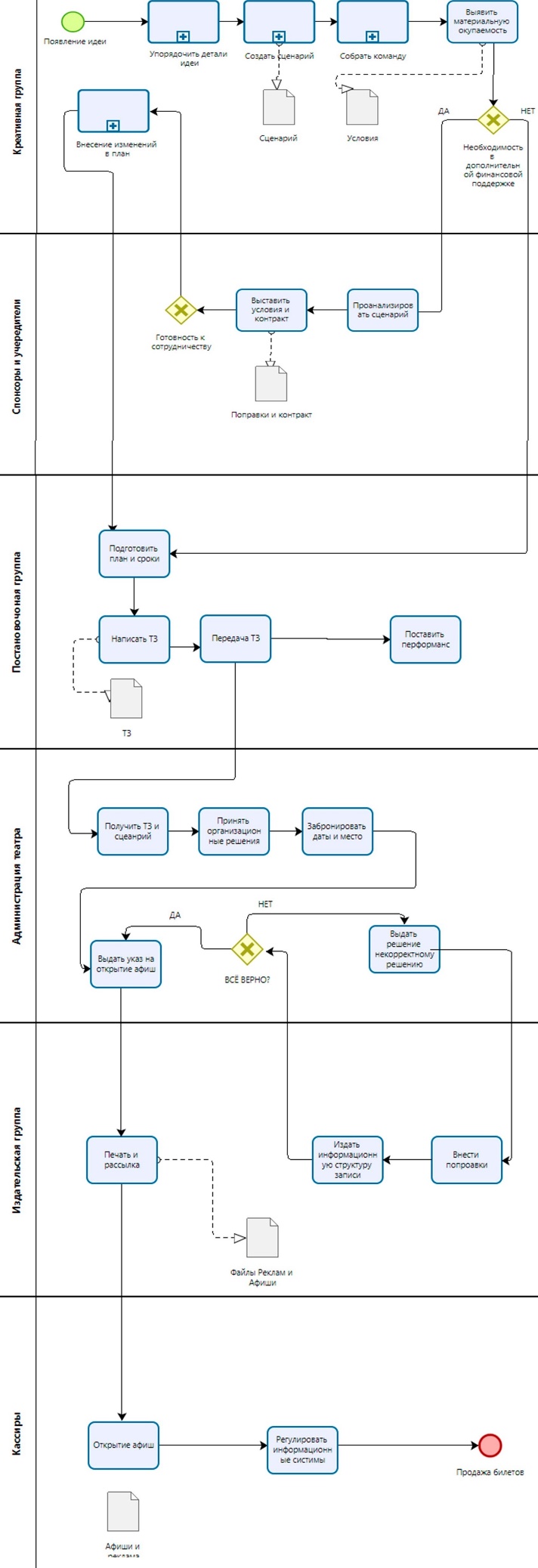


Рисунок 1.9 – BPMN-модель процесса создания записи на перформанс театром

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

Разрабатываемое в ходе курсовой работы программное средство должно соответствовать определенным требованиям. Для удобства понимания кода, избежание логических ошибок и легкой ориентации в нем, следует учесть ряд правил.

Все переменные и константы должны иметь [осмысленные](http://ci-plus-plus-snachala.ru/?page_id=228) имена в рамках тематики варианта курсовой работы. Переменным рекомендуется присваивать имена, состоящие из букв нижнего регистра; для формирования составного имени используется нижнее подчеркивание. Константам рекомендуется присваивать имена, состоящие из букв верхнего регистра.

Имена функций должны быть осмысленными, начинаться с буквы нижнего регистра, строиться по принципу «глагол плюс существительное».

Не допускается использование оператора прерывания «goto».

Код необходимо комментировать (как минимум в части объявления структур, массивов/векторов, прототипов функций, нетривиальной логики).

Код не должен дублироваться – для этого существуют функции.

Одна функция решает только одну задачу (например, не допускается в одной функции считывать данные из файла и выводить их на консоль – это две разные функции). При этом внутри функции возможен вызов других функций.

Оператор сотовой связи хранит в себе: ФИО абонента, номер телефона, год подключения, наименование текущего тарифного плана.

Индивидуальное задание: вывести список и подсчитать общее количество абонентов, подключенных с xxxx года (год вводится с клавиатуры).

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

Для обеспечение максимальной эффективности от автоматизации данного процесса разрабатываемое программное средство должно обладать следующей функциональностью:

* регистрация;
* авторизация;
* добавление учетных записей;
* редактирование учетных записей;
* удаление учетных записей;
* отображение учетных записей;
* отображение данных;
* редактирование данных;
* удаление данных;
* добавление данных;
* сортировка данных по разным критериям;
* фильтрация данных по разным критериям;
* вывод данных;
* шифрование данных;
* обеспечение корректного ввода данных;
* вывод сообщений для пользователя.

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

UML Diagrams расшифровывается как Unified Modeling Language. Это стандарт, который в основном используется для создания объектно-ориентированных, значимых моделей документации для любой программной системы, представленной в реальном мире. Это дает нам возможность разрабатывать богатые модели, которые описывают работу любых программно-аппаратных систем.

Он является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению.

UML служит отличным способом создания профессиональной документации, которая является необходимой частью любой разработки проекта. UML является неотъемлемой частью создания объектно-ориентированного проектирования систем. Он предоставляет вам средства для создания мощных моделей и конструкций для рациональных систем, которые можно понять без особых трудностей.

Моделирование необходимо для понимания системы. Обычно при этом единственной модели никогда не бывает достаточно. Поэтому приходиться разрабатывать большое количество взаимосвязанных моделей.

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Диаграмма вариантов использования представляет совокупность прецедентов и актеров. Они описывают систему с точки зрения возможностей ее использования. Наиболее часто данный вид диаграмм используют при организации и моделировании поведения системы. Она позволяет создать список операций, которые должна выполнять система. Часто эти диаграммы называют диаграммами функций так как на их основе определяется список требований к системе и множество, выполняемых системой функций. На диаграмме отображаются объекты предметной области с выполняемыми ими задачами (см. рисунок 1.10).

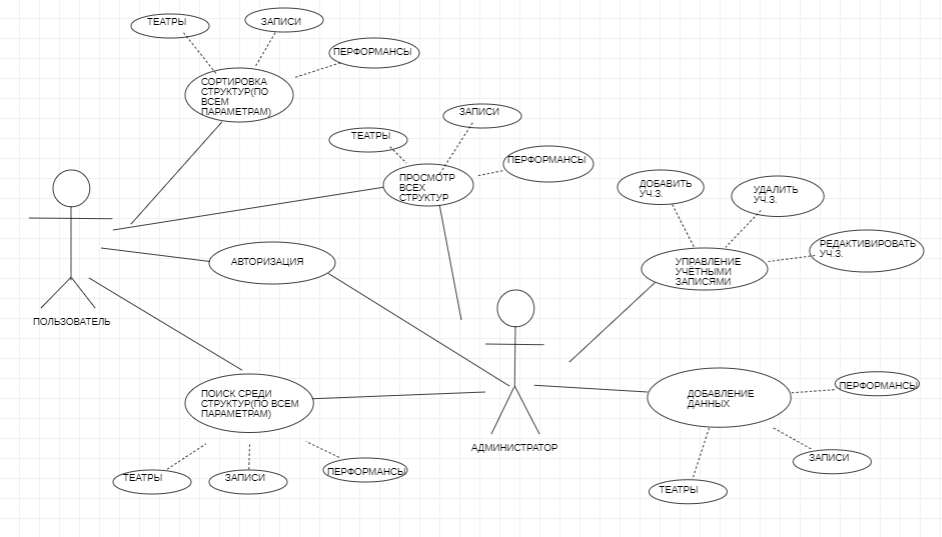
****

Рисунок 1.10– Диаграмма вариантов использования

Показать разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

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

Диаграммы компонентов следует применять, когда система разделяется на компоненты и надо показать их взаимоотношения посредством интерфейсов или схему компонентов в низкоуровневой структуре системы.

Именно данный тип диаграмм позволяет не потеряться в обилии модулей и связей между ними (см. рисунок 1.11).

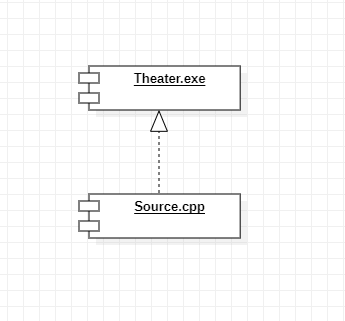


Рисунок 1.11 – Диаграмма компонентов

Каждый объект системы, обладающий определенным поведением, может находится в определенных состояниях, переходить из состояния в состояние, совершая при этом определенные действия в процессе реализации сценария поведения объекта.

Диаграмма состояний позволяет описывать поведение системы. В объектно-ориентированном подходе разрабатывается диаграмма состояний единственного класса, демонстрирующая поведение одного объекта в течение его жизни. Состояние на диаграмме является более абстрактным понятием, чем состояние объекта (последнее есть комбинация всех данных из полей объекта). Диаграмма позволяет проектировать различные способы реакции на события (см. рисунок 1.12).

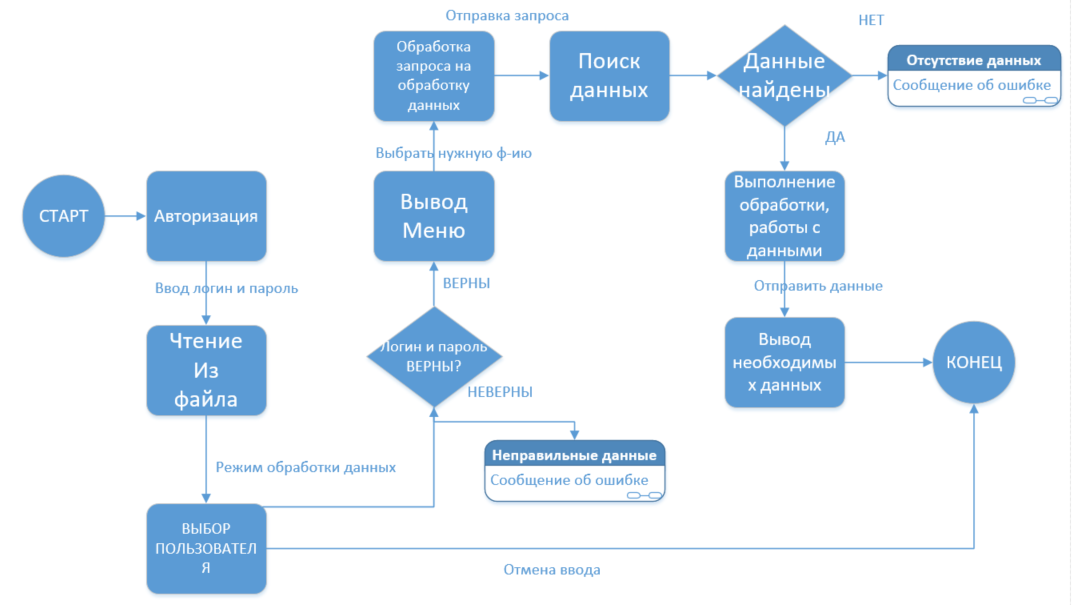


Рисунок 1.12 – Диаграмма состояния

Диаграмма развёртывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения - маршруты передачи информации между аппаратными узлами. Диаграмма связана с диаграммой компонентов так как в узле размещается один либо несколько компонентов (см. рисунок 1.13).

В данной диаграмме представлены файлы, которые в свою очередь образуют программу и зависят друг от друга):

* Theatre.exe – сама программа;
* user.txt – файл, в котором хранится информация о всех обычных пользователях;
* admin.txt – файл, в котором хранится информация о всех админах;

performance.txt – файл, в котором хранится информация о всех возможных представлениях.

* record.txt – файл, в котором хранится информация о всех записях в театрах.
* theatre.txt – файл, в котором хранится информация о всех работающих театрах.

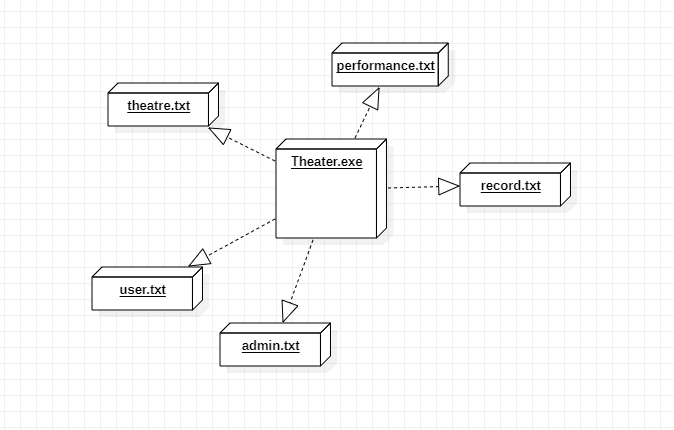


Рисунок 1.13 – Диаграмма развертывания

На UML-диаграмме последовательностей (рис. 1.14) в общих чертах изображена основная суть разрабатываемого программного средства. Пользователь вводит данные (пароль и логин). В случае, если данные введены верно, пользователь получает доступ к определенному набору действий, определенных разработчиком. В противном же случае, пользователь будет уведомлен о том, что данные были введены неверно.

Диаграмма последовательности оперирует объектами и сообщениями.

На данном типе диаграмм в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов. На них отражается последовательность передачи сообщений между объектами и главный акцент уделяется последовательности приема и передачи сообщений.

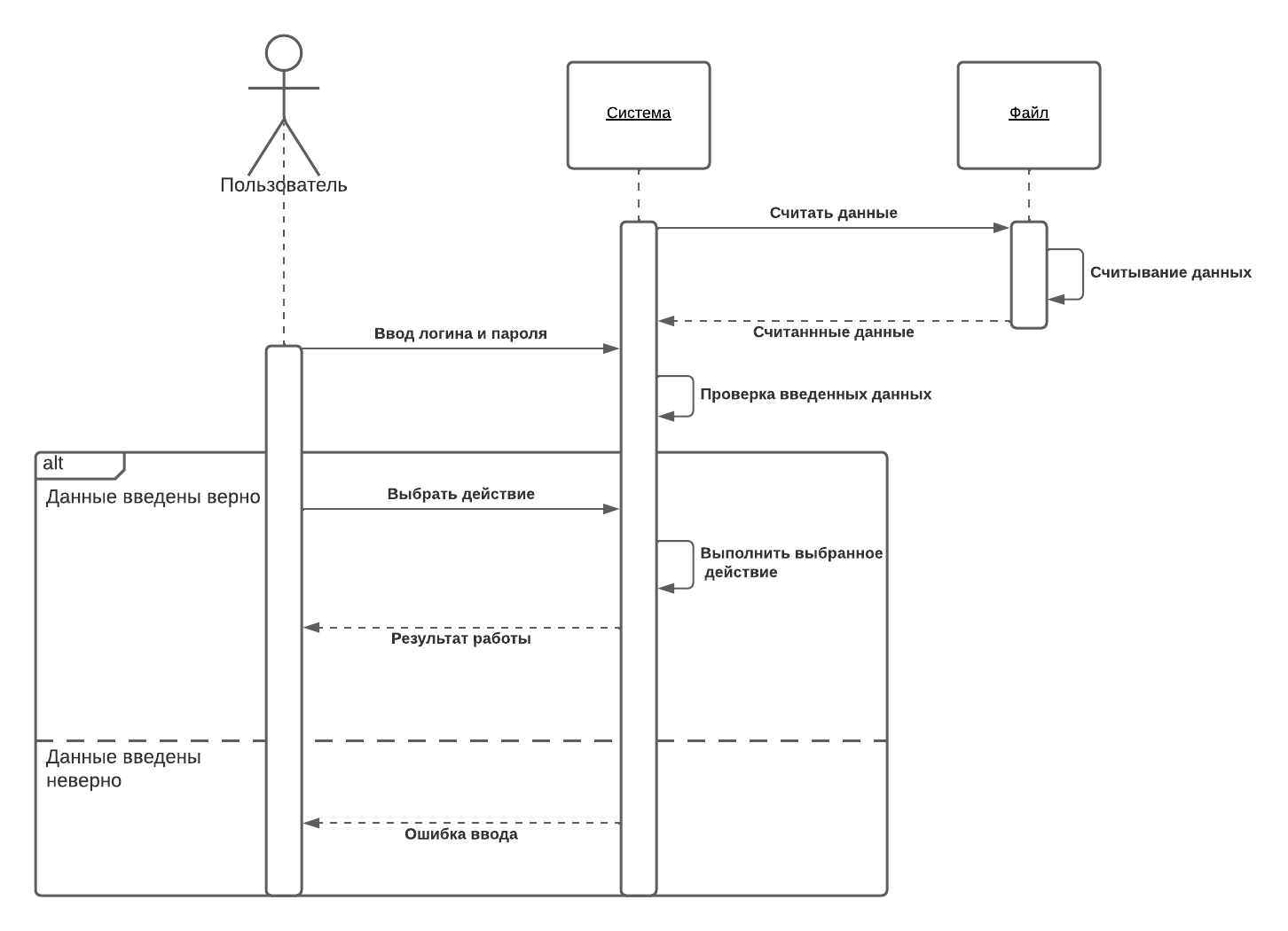


Рисунок 1.14 – Диаграмма последовательности

# **ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

# 

## **Постановка задачи**

По условию задания требуется разработать информационную систему для автоматизации учёта сведений о театральных билетах, представления и театрах. Система должна предусматривать режимы ведения данных о пользователях, о представления, театрах и записях, отражающих все нужные сведения. В зависимости от предоставляемого уровня доступа в программе предусмотрены ограничения на выполнение некоторых функций. То есть пользователь не должен иметь прав, таких, какие имеет администратор. Пользователь и администратор обязаны иметь достаточный удобный и практичный функционал для комфортной работы в программе.

В ходе разработки автоматизированной системы нужно реализовать

следующие возможности для пользователя:

* авторизация пользователя, которая включает в себя регистрацию пользователя (производится админом), вход в систему при наличии данных о регистрации;
* просмотр всех данных;
* поиск среди театров, представлений и записей;
* поиск точных объектов;
* фильтрация;
* сортировка.

В ходе разработки автоматизированной системы нужно реализовать следующие методы для администратора:

* авторизация администратора;
* добавление данных;
* просмотр информации;
* редактирование данных;
* поиск данных;
* добавление нового пользователя;
* удаление учетной записи пользователя;
* сортировка данных;
* фильтрация данных.

Регистрация происходит путем ввода личных данных в программу, в данный момент производится запись информации о пользователе и его логин и пароль заносятся в файл.

Авторизация происходит путем ввода личного логина и пароля, при некорректной введенной информации программа выдаст ошибку и попросит ввести логин и пароль снова.

Для успешного создания записи о модели все информационные параметры должны быть заполнены, так что предусмотрим следующие ограничения для информационной системы:

Модуль администраторавключает следующие подмодули:

1. Управление учетными записями пользователей:
   * просмотр всех учетных записей;
   * добавление новой учетной записи;
   * удаление учетной записи.
2. Работа с данными:
3. режим редактирования:

* просмотр всех данных;
* добавление новой записи;
* удаление записи;
* редактирование записи (по различным параметрам);

1. режим обработки информации:

* поиск данных (по различным параметрам);
* сортировка данных (по различным параметрам);
* поиск записи с наилучшим рейтингом;

Модуль пациента(пользователя) включает в себя подмодуль работы с данными со следующими функциональными возможностями:

* просмотр всех данных о театрах, представлениях и записях;
* поиск данных (по различным параметрам);
* сортировка данных (по различным параметрам);
* поиск записи с наилучшим рейтингом;

Для реализации перечисленных модулей/подмодулей создано меню с соответствующими пунктами.

## **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. Для разработки системы был выбран метод восходящей разработки. Разработка модульной структуры программы подразумевает графическое представление структуры программы с указанием модулей, подмодулей и их функциональных возможностей. Модульная структура предоставлена в ПРИЛОЖЕНИЕ В.

## **Выбор способа организации данных**

В ходе проектирования приложения в качестве способа организации входных данных были выбраны структуры. Выбор обусловлен лёгкой программной функциональностью. Описание используемых структур представлено ниже.

Таблица 2.3 – Описание структуры Theatre

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| t\_code | long int | Идентификационный номер театра |
| t\_name | char[30] | Название театра |
| location | struct Adress | Адрес театра |
| phone | long int | Номер доступного телефона театра |
| t\_place | int | Количество мест в актовом зале театр |

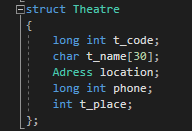


Рисунок 2.3 – Структура Theatre

Таблица 2.4 – Описание структуры Performance

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| p\_code | long int | Идентификационный номер перформанса |
| p\_name | char | Название перформанса |
| rating | int | Рейтинг перформанса |
| actor | struct FIO | ФИО главного актёра |
| cost | int | Цена за перформанс |
| author | char[30] | Автор перформанса |
| genre | char[30] | Жанр перформанса |

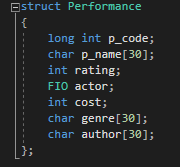


Рисунок 2.4 – Структура Performance

Таблица 2.5 – Описание структуры Record

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| t\_code | long int | Идентификационный номер театра |
| p\_code | long int | Идентификационный номер перформанса |
| season | char[30] | Сезон |
| \_time | struct time | Дата(часы.минуты) представления |
| \_date | struct date | Дата(день.месяц.год) представления |
| tickets | int | Количество билетов |

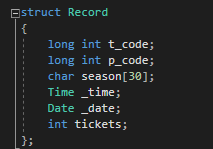


Рисунок 2.5 – Структура Record

Таблица 2.6 – Описание структуры Adress

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| street | char[30] | Улица |
| home | int | Номер дома |

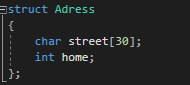


Рисунок 2.6 – Структура Adress

Таблица 2.7 – Описание структуры FIO

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| name | char[30] | Имя |
| s\_name | char[30] | Фамилия |
| l\_name | char[30] | Отчество |

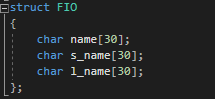


Рисунок 2.7 – Структура FIO

Таблица 2.8 – Описание структуры Date

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| day | int | День |
| month | int | Месяц |
| year | int | Год |

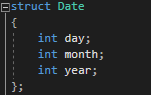


Рисунок 2.8 – Структура Date

Таблица 2.9 – Описание структуры Time

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| hour | int | Часы |
| min | int | Минуты |

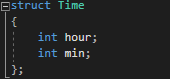


Рисунок 2.9 – Структура Time

Таблица 2.10 – Описание структуры Log

|  |  |  |
| --- | --- | --- |
| Название поля | Тип данных | Назначение поля |
| logIn | char[40] | Логин |
| passWord | char[40] | Пароль |

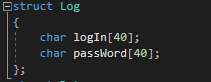


Рисунок 2.10 – Структура Log

## **Разработка перечня пользовательских функций программы**

Для реализации программы были использованы следующие функции (далее будут указаны прототипы функций):

Функции, отвечающие за правильность ввода данных (символы, числа, коды данных структур, длина строки). Проверка на корректность ввода (возвращение «1», если введено корректно и «-1», если нет):

// Проверки на ввод

bool CheckLetters(char[]);//

bool CheckNumbers(char[]);//

bool CheckTheatreCode(Theatre\* arr, int n, int code); //

bool CheckPerformanceCode(Performance\* arr, int n, int code); //

bool CheckLength(char str[]) ; //

Функции, отвечающие за запись в файл и получение информации из файла («set»– запись, «get»– чтение):

// Получение всей информации из файлов

int GetFileLog(Log\*\*, char[]);//

int GetFileRecord(Record\*\*);//

int GetFileTheatre(Theatre\*\*);//

int GetFilePerformance(Performance\*\*);//

// Запись информации в файл

void SetFileLog(Log\*, int, char[]);//

void SetFileRecord(Record\*, int);//

void SetFileTheatre(Theatre\*, int);//

void SetFilePerformance(Performance\*, int);//

Функции, которые являются меню, которые видит администратор или пользователь. Помимо главного меню, которое отображается после авторизации, есть меню фильтрации, сортировки, поиска и так далее, в данном случае нам надо выбрать пункт, по которому информация будет отсортирована, отфильтрована и так далее:

// Меню ф-ий

void MenuAdmin();//

void MenuUserControl();

void MenuView();//

void MenuAdd();//

void MenuEdit();//

void MenuDelete();//

void MenuSearch();//

void MenuSort();//

void MenuFiltration();//

// Авторизация и первый запуск

bool LogIn(char[]);//

void UserMenu();

void MenuLogIn();//

void Srart();//

Функции, отвечающие за просмотр (View), редактирование (Edit/Compare), удаление (Del), добавление (Add/Write), фильтрацию (Filter), сортировку (Sort) и поиск (Search) данных:

// Ф-ии работы с данными: добавление, просмотр, удаление, редактивирование

void AddTheatre();//

void AddRecord();//

void AddPerformance();//

void ViewTheatre(Theatre\* arr, int n);//

void ViewRecord(Record\* arr, int n);//

void ViewPerformance(Performance\* arr, int n);//

void DeletePerformance();//

void DeleteTheatre();//

void DeleteRecord();//

void EditPerformance();//

void EditTheatre();//

void EditRecord();//

// Ф-ии работы с временем: Чтение, запись, сравнение времени

int Compare\_time(Time a, Time b);//

Time ReadTime();//

void WriteTime(Time a);//

int Compare\_date(Date a, Date b);//

Date ReadDate();//

void WriteDate(Date a);//

// Ф-ии поиска информации

void SearchRecordSes();//

void SearchTheatre();

void SearchPerformance();

void SearchPerformanceActor();

void SearchPerformanceGenre();

// Ф-ии сортировки по алфавиту, рейтингу, цене

void SortAbc();//

void SortRating();//

void SortCost();//

// Ф-ии фильтрации по времени и рейтингу

void FilterCost();//

void FilterRating();

// Ф-ия инд. задания

void Task();//

// Ф-ии добавление, удаления и просмотра пользователей

Log AddUser(Log\* arr, int n);//

void DelUser(Log\* arr, int n);//

void ViewUser(Log\* arr, int n);//

## **Разработка схем алгоритмов программы**

Алгоритм в программировании — это набор последовательных инструкций, которые описывают порядок поведения программы для достижения нужной цели.

По сути, само «программирование программы» — это и есть запись последовательных алгоритмов, а значит, определенных действий при помощи кода, которые пишутся для получения нужного результата работы самой программы. Поэтому можно смело заявить, что любая программа — это определенные алгоритмы, плюс определенные структуры данных.

Алгоритм работы программы представлен на рисунке 2.11.



Рисунок 2.11 – Схема алгоритма работы всей программы

На рисунке 2.3.2 представлен алгоритм работы функции фильтрации записей в алфавитном порядке. Чтобы произвести фильтрацию, необходимо получить количество(длину) записей. В данной функции подразумевается, что до её использования произошла проверка на наличие записей.

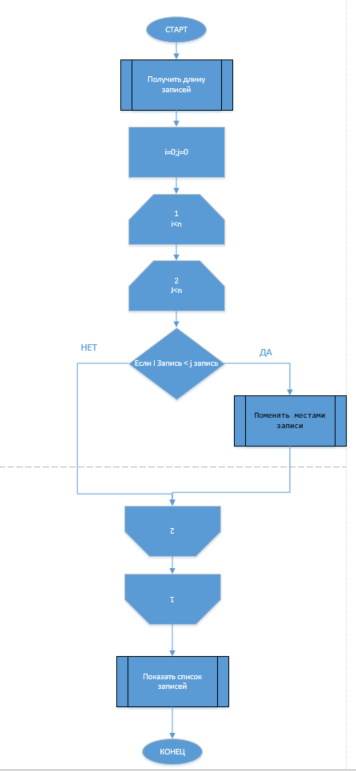


Рисунок 2.12 – Алгоритм работы функции фильтрации записей в алфавитном порядке

На рисунке 2.13 представлен алгоритм работы функции поиска. В начале необходимо выбрать параметр поиска.

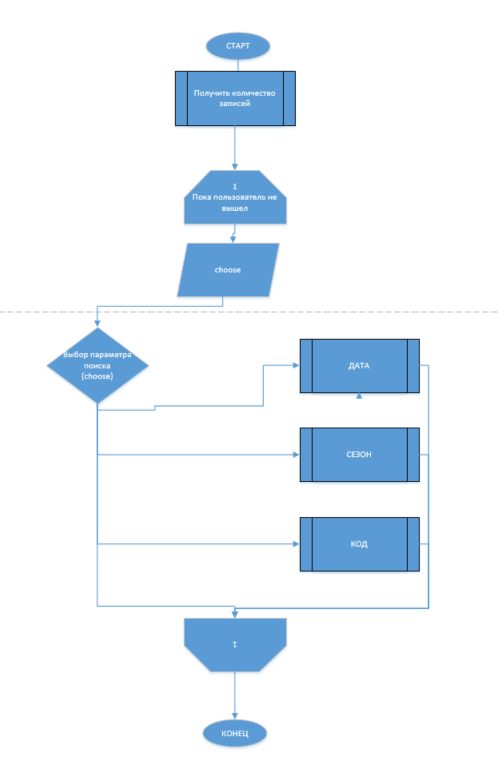


Рисунок 2.13 – Схема алгоритма функции поиска записи

После ввода значения параметра, программа выводит на экран найденные записи в случае их наличия, если же записи с данным параметром отсутствуют, пользователь получает сообщение о произошедшей ошибке. Функция поиска записи по сезону (см. рисунки 2.14-2.15).

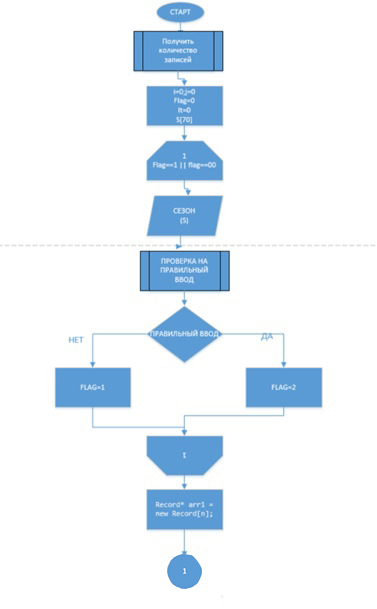


Рисунок 2.14 – Схема алгоритма функции поиска записи по сезону (1)

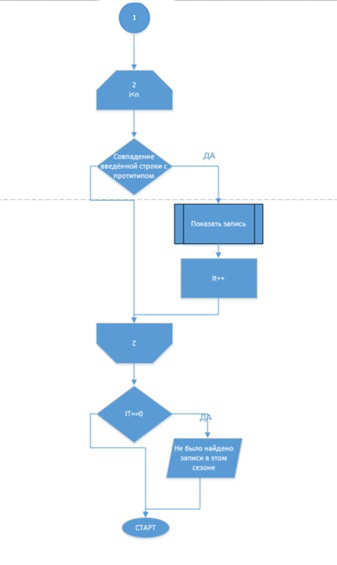


Рисунок 2.15 – Схема алгоритма функции поиска записи по сезону (2)

# **ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА**

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

* введенные пользователем данные не соответствуют формату поля (например, символы в числовом поле, некорректная дата);
* введенные пользователем данные нелогичны (например, год представления больше 2021);
* ничего не найдено по результатам поиска.
* удаление пустых данных
* редактирование пустых данных.

При вводе некорректных данных при выборе во всех меню программы, программа оповещает о корректности неверных данных и предлагает осуществить повторный ввод.

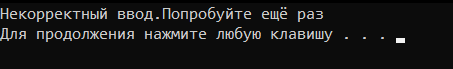
****

Рисунок 3.1 – Главное меню программы

При вводе некорректных данных при авторизации пользователю даётся возможность повторять входы до ввода корректных данных логина и пароля.

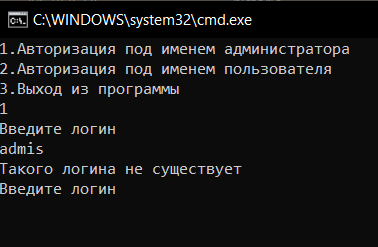
****

Рисунок 3.2 – Меню авторизации

Если пользователю необходимо просмотреть, редактировать, удалить, отсортировать записи, а в файле ничего нет, то при нажатии соответствующего пункта меню, пользователь будет оставаться в этом меню и будет предупрежден о том, что соответствующий файл пуст (см. рисунок 3.3).

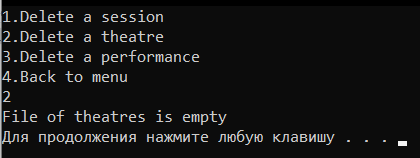
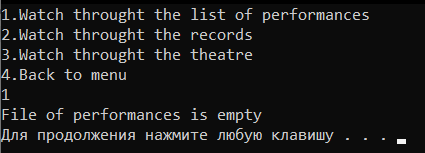
C:\Users\Леново\Documents\ShareX\Screenshots\2022-05\cmd_BXSArLRXoZ.png

Рисунок 3.3 – Работа с пустыми файлами.

Аналогичная ситуация происходит при поиске пациента или доктора, когда искомого параметра нет в файле.

Если администратор хочет создать запись без созданных перформансов и доступных театров ему это не получиться сделать и будет выдано следующее предупреждение (см. рисунок 3.4).

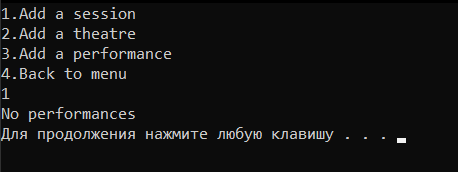


Рисунок 3.4 – Работа с пустыми данными о перформансах.

При вводе некорректных данных даты. Пользователь будет оповещён о некорректности ввода и ему будет предложена верная форма для заполнения ввода (см. рисунки 3.5 и 3.6).

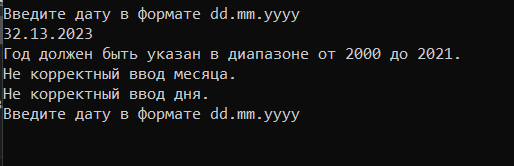


Рисунок 3.5 – Работа с некорректной датой.

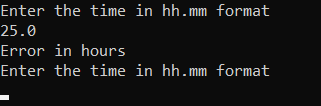


Рисунок 3.6 – Работа с некорректным временем.

При вводе чисел вместо символов, когда запрос стоит о вводе имени или названии улицы, то пока пользователь не введёт верный формат текста, он не выйдет из режима (см. рисунок 3.7).

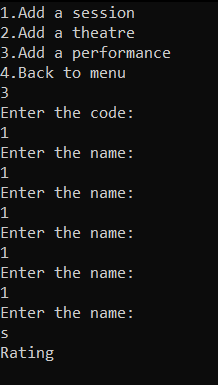


Рисунок 3.7 – Работа с некорректным форматом.

# **ИНСТРУКЦИЯ ПО РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ И СКВОЗНОЙ ТЕСТОВЫЙ ПРИМЕР**

## **Авторизация**

После запуска программы в консоли отображается главное меню, в котором пользователю предлагается осуществить 3 действия: вход под администратором, вход под пользователем или же завершение работы приложения (рисунок 4.1).

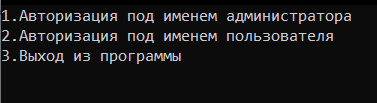


Рисунок 4.1 – Меню первого уровня

Не зависимо от чьего имени будет осуществляться вход, необходимо ввести логин и пароль (рисунок 4.2). В целях безопасности пароль маскируется звёздочками.

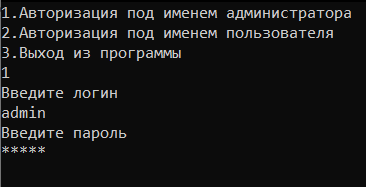


Рисунок 4.2 – Авторизация пользователя

При вводе неверного логина, пользователю выводится фраза об отсутствии такого логина, пользователю даётся три попытки на ввод существующего логина (рисунок 4.3).

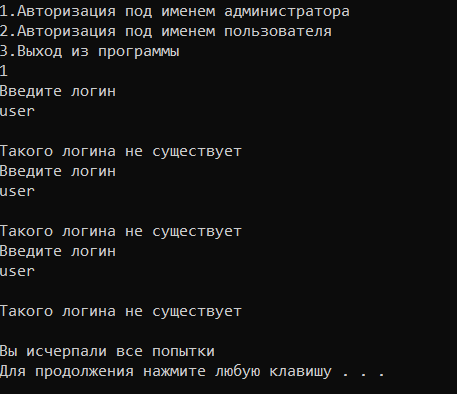


Рисунок 4.3 – Ввод несуществующего логина

При вводе неверного пароля, пользователю даются дополнительные попытки ввода, которых также всего 3. Авторизация будет осуществляться до тех пор, пока пользователь не исчерпает все попытки (рисунок 4.4) или пока не введет верный пароль.

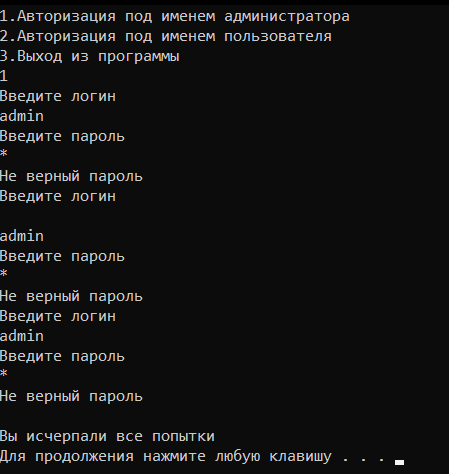


Рисунок 4.4 – Ошибка ввода пароля

## **4.2 Модуль администратора**

После успешной авторизации от имени администратора (рисунок 4.5) нам выводится меню с возможностью выбора пунктов, часть из которых не доступна обычному пользователю (рисунок 4.6).

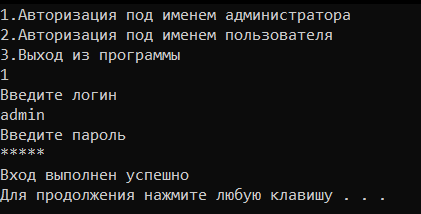


Рисунок 4.5 – Успешная авторизация

В отличие от обычного пользователя, администратору доступны такие функции:

* добавление новой записи;
* редактирование записей;
* удаление записей;
* меню управления пользователями.

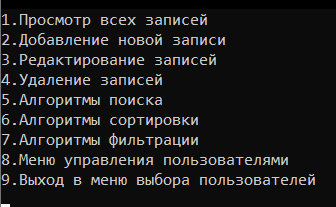


Рисунок 4.6 – Меню администратора

Меню управления пользователями (рисунок 4.7) позволяет администратору просматривать все аккаунты, а также регистрировать новые и удалять определённые ранее.

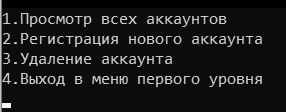


Рисунок 4.7 – Меню управления пользователями

Пункт «Добавление новой записи» дает возможность администратору добавить новый театр, перформанс и запись-билет (рисунок 4.8).

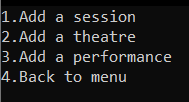


Рисунок 4.8 – Меню добавления данных

Редактирование данных позволяет администратору вносить изменения в данные о перформансах, театрах и билетах (рисунок 4.9).

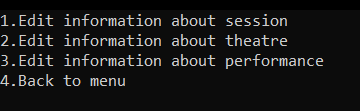


Рисунок 4.9 – Меню редактирования данных

Пункт «Удаление данных» дает возможность удалить сведение о каком-либо театре, перформансе или же виде билета (рисунок 4.10).

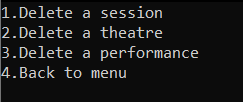


Рисунок 4.10 – Меню удаления данных

## **4.3 Модуль пользователя**

После успешной авторизации от имени пользователя, выводится меню возможных операций (рисунок 4.11)

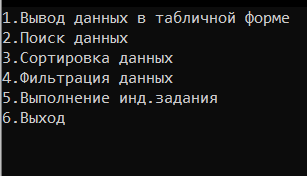


Рисунок 4.11 – Меню пользователя

Единственный пункт, недоступный для администратора, –это «Выполнение индивидуального задания», он подразумевает составление рейтинга о самых популярных перформансах и наиболее посещаемых театрах в табличной форме (рисунок 4.12).

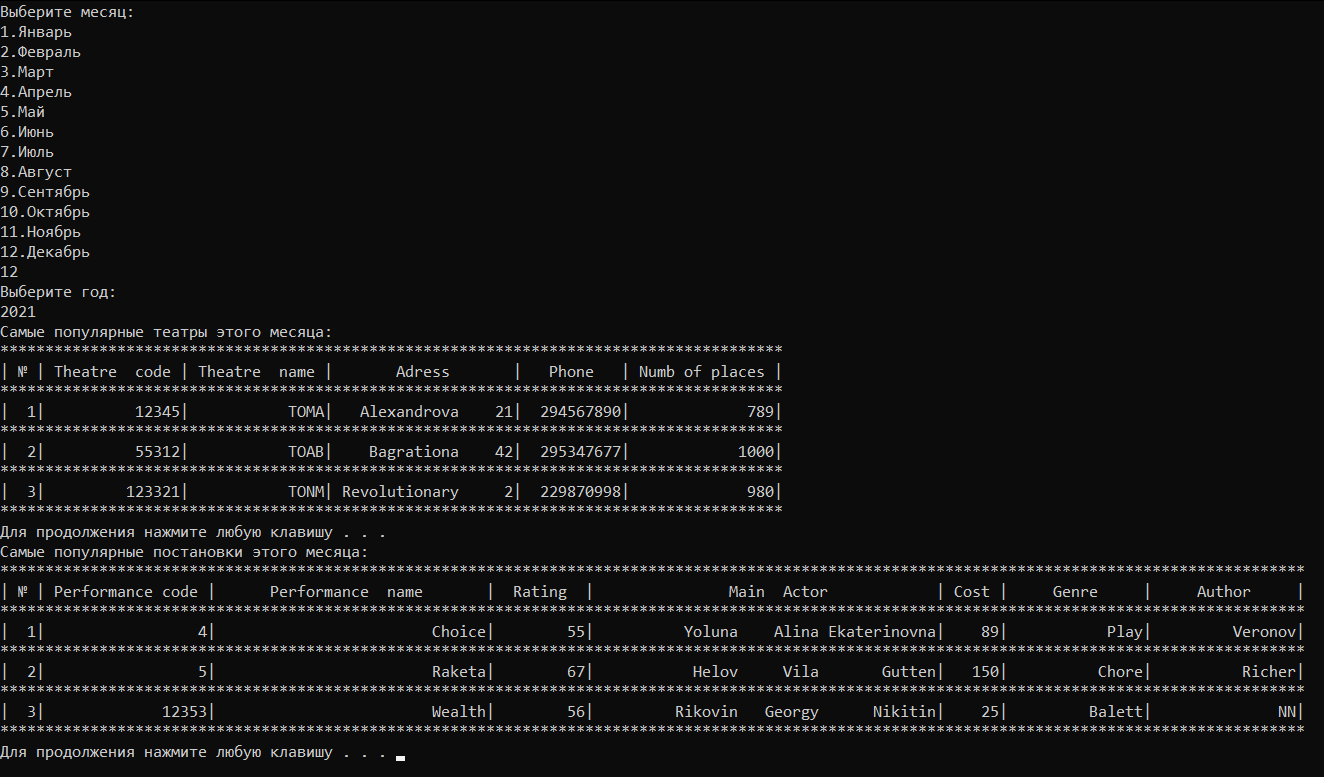


Рисунок 4.12 – Выполнение индивидуального задания

Вывод данных в табличной форме включает в себя вывод данных о театрах, записях, перформансах (рисунок 4.13).

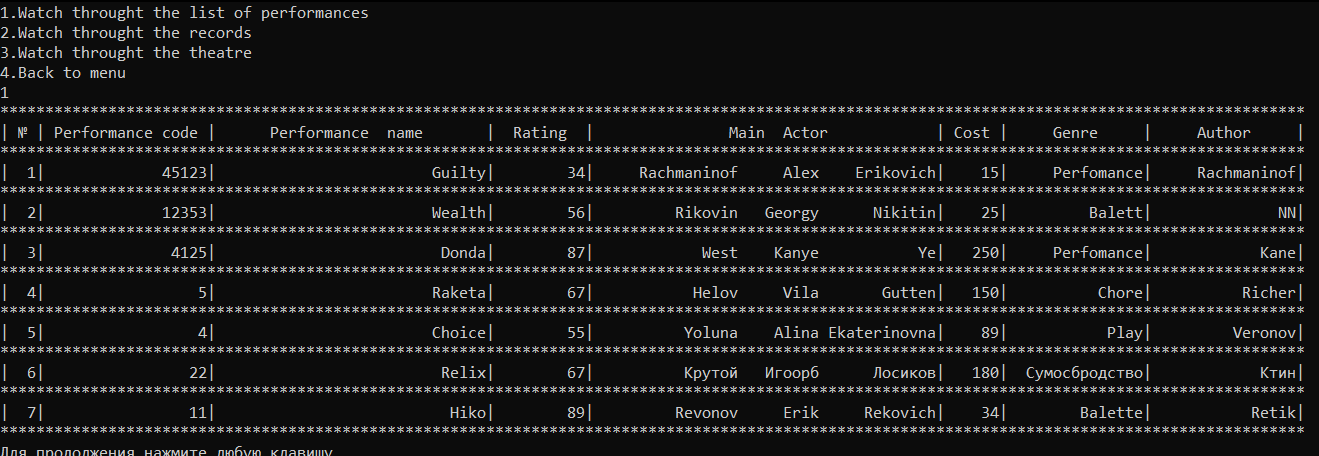


Рисунок 4.13 – Пример вывода данных перформансов

Пункт поиск данных позволяет путем ввода критерия поиска найти нужный перформанс или же запись-билет.

Для примера рассмотрим поиск театра по названию. Выбирается нужный критерий и вводится корректное имя (рисунок 4.14).

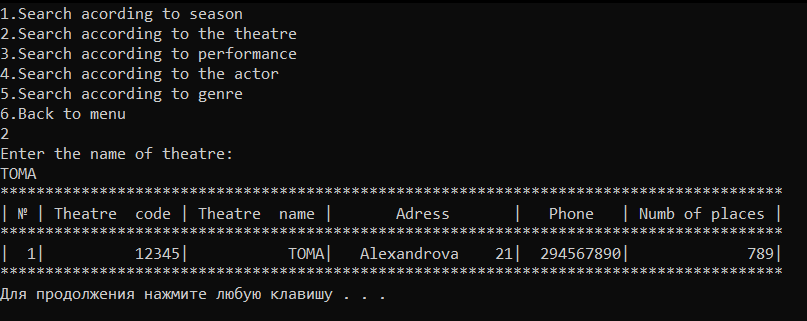


Рисунок 4.14 – Поиск театра по имени

Следующие две возможности тесно связаны друг с другом - это сортировка и фильтрация перформансов по параметрам.

Для осуществления сортировки нам необходимо выбрать, по какому критерию следует осуществить сортировку (рисунок 4.19). Выберем пункт «Sort performance according to an aphabet». Он подразумевает сортировку по алфавиту (рисунок 4.15).

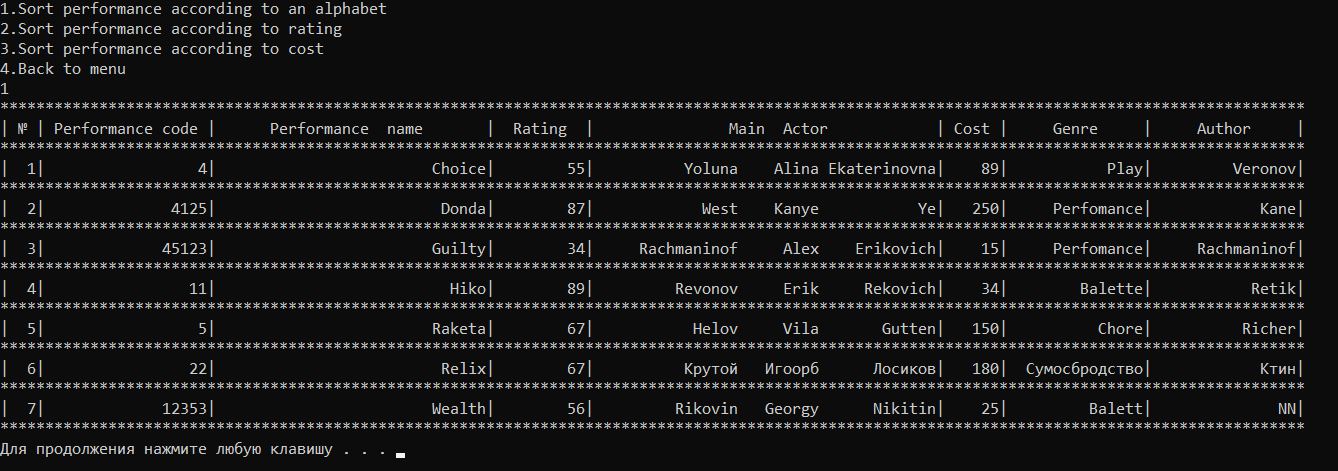


Рисунок 4.15 – Отсортированные по алфавиту перформансы

Пункт «фильтрация» позволяет пользователю отобрать данные в зависимости от границ поиска и параметра (рисунок 4.16).

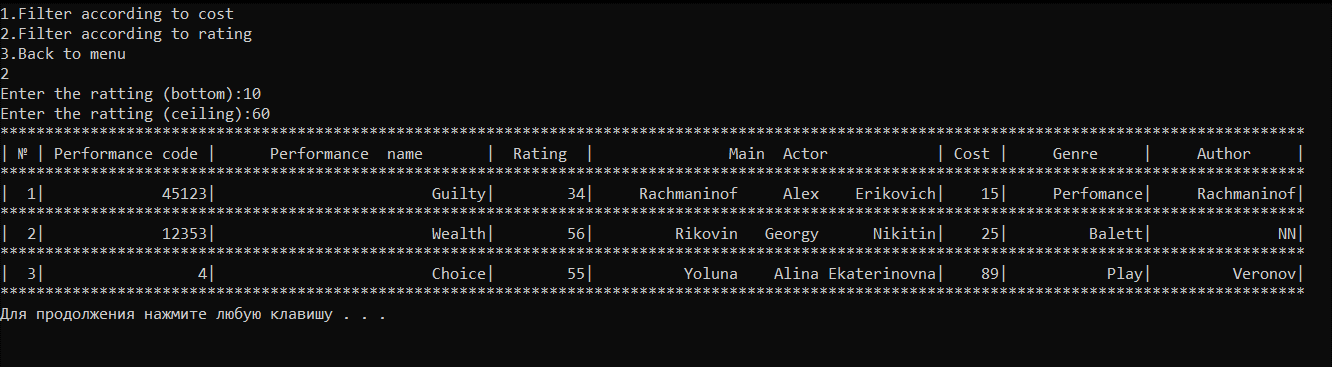


Рисунок 4.16 – Фильтрация перформансов по рейтингу в заданном диапазоне

# **ЗАКЛЮЧЕНИЕ**

При выполнении курсовой работы был изучен процесс учета продажи и обработки театральных билетов. Было установлено, что правильно продуманная и написанная программа позволяет уменьшить трудоемкость выполнения этой операции вручную.

Были созданы пользовательские функции приложений: функции поиска, сортировки, получения данных из файла, авторизации, редактирования, проверки данных и так далее.

Были тщательно продуманы и написаны следующие алгоритмы функций: главного меню, фильтрации, поиска по параметру, сортировок по параметру.

Были рассмотрены выходящие из порядка ситуации и была реализована их обработка с помощью функций проверок (на основе языка С).

Цель, заявленная во введении этой работы: усовершенствование процесса учёта продажи театральных билетов – была успешно достигнута.

Мой итог данной работы – функциональное приложение, которое четко автоматизирует учет театральных билетов. В этом приложении присутствуют конструкции объектно-ориентированного программирования, стандартные и пользовательские функции.

Несомненно, данное приложение можно усовершенствовать, добавив больше информации и более широкий функционал, что могло бы позволить сделать спектр решаемых задач глубже. Также требует отметить, что с данной программой комфортно может работать как клиент-пользователь, так и рабочий с правами администратора.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Методология IDEF0 [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №2.docx](file:///C:\Users\Леново\Desktop\Универ\трпо\ТРПО%20Лабораторная%20работа%20№2.docx)
2. Методическое пособие для выполнения лабораторной роботы «Язык Uml. Диаграммы UML» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО - Лабораторные работы №7-8.doc](file:///C:\Users\Леново\Desktop\Универ\трпо\ТРПО%20-%20Лабораторные%20работы%20№7-8.doc)
3. Методическое пособие для выполнение лабораторной работы «Моделирование бизнес-процессов на основе нотации BPMN в BizagiProcess Modeler» [Электронный ресурс]. – Режим доступа: [..\трпо\ТРПО Лабораторная работа №4 (2).docx](file:///C:\Users\Леново\Desktop\Универ\трпо\ТРПО%20Лабораторная%20работа%20№4%20(2).docx)
4. Афиша представлений Большого Театра Минск [Электронный ресурс]. – Режим доступа: https://bolshoibelarus.by/rus/afisha-ru/a/-.html
5. Пособие по проектированию ИС театра [Электронный ресурс]. – Режим доступа:  [https://www.npmaap.ru/catfile/possob/pstaetr.html](https://beltelecom.by/private/telephony/installation-and-use/rules)
6. Система антиплагиат [Электронный ресурс]. – Режим доступа:  https://textovod.com/unique

# **ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Листинг кода**

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "RUS");

//Srart();

MenuLogIn();

return 0;

}

void UserMenu()

{

bool work = true;

while (work)

{

system("cls");

printf("1.Вывод данных в табличной форме\n");

printf("2.Поиск данных\n");

printf("3.Сортировка данных\n");

printf("4.Фильтрация данных\n");

printf("5.Выполнение инд.задания\n");

printf("6.Выход\n");

char c;

scanf(" %c", &c);

switch (c)

{

case '1':

{

MenuView();

break;

}

case '2':

{

MenuSearch();

break;

}

case '3':

{

MenuSort();

break;

}

case '4':

{

MenuFiltration();

break;

}

case '5':

{

Task();

break;

}

case '6':

{

work = false;

break;

}

default:

{

system("cls");

printf("Некорректный ввод\n");

system("pause");

break;

}

}

}

}

void MenuLogIn()

{

bool work = true;

while (work)

{

system("cls");

printf("1.Авторизация под именем администратора\n");

printf("2.Авторизация под именем пользователя\n");

printf("3.Выход из программы\n");

char c;

scanf(" %c", &c);

switch (c)

{

case '1':

{

char s[70] = "admin.txt";

if (LogIn(s))

{

MenuAdmin();

}

}

break;

case '2':

{

char s[70] = "user.txt";

if (LogIn(s))

{

UserMenu();

}

}

break;

case '3':

{

work = false;

}

break;

default:

{

system("cls");

printf("Некорректный ввод.Попробуйте ещё раз\n");

system("pause");

break;

}

}

}

}

void Task()

{

system("cls");

Record\* arr = NULL;

int n = GetFileRecord(&arr);

if (n == 0)

{

printf("Not enought data\n");

system("pause");

return;

}

char s[70];

int month = 0;

while (month < 1 || month >12)

{

printf("Выберите месяц:\n");

printf("1.Январь\n");

printf("2.Февраль\n");

printf("3.Март\n");

printf("4.Апрель\n");

printf("5.Май\n");

printf("6.Июнь\n");

printf("7.Июль\n");

printf("8.Август\n");

printf("9.Сентябрь\n");

printf("10.Октябрь\n");

printf("11.Ноябрь\n");

printf("12.Декабрь\n");

scanf(" %s", s);

if (CheckNumbers(s))

{

month = atoi(s);

}

}

int year = 0;

while (year < 2000 || year>2023)

{

printf("Выберите год:\n");

scanf(" %s", s);

if (CheckNumbers(s))

{

year = atoi(s);

}

}

Record\* rec = (Record\*)malloc(sizeof(Record) \* n);

int nRec = 0;

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

{

if (arr[i].\_date.year == year && arr[i].\_date.month == month)

{

rec[nRec] = arr[i];

nRec++;

}

}

if (nRec == 0)

{

printf("В этом месяце постановок нет\n");

free(arr);

free(rec);

system("pause");

return;

}

Theatre\* theatres = NULL;

int nTh = GetFileTheatre(&theatres);

if (nTh == 0)

{

printf("Недостаточно данных\n");

system("pause");

free(arr);

free(rec);

return;

}

Performance\* perf = NULL;

int nPer = GetFilePerformance(&perf);

if (nPer == 0)

{

printf("Недостаточно данных\n");

system("pause");

free(arr);

free(rec);

free(theatres);

return;

}

int\* indexesT = (int\*)malloc(sizeof(int) \* nTh);

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

{

indexesT[i] = 0;

}

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

{

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

{

if (rec[i].t\_code == theatres[j].t\_code)

{

indexesT[j]++;

}

}

}

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

{

for (int j = i; j < nTh; j++)

{

if (indexesT[i] < indexesT[j])

{

Theatre temp = theatres[i];

theatres[i] = theatres[j];

theatres[j] = temp;

int t = indexesT[i];

indexesT[i] = indexesT[j];

indexesT[j] = t;

}

}

}

int x = 0;

while (x < nTh && indexesT[x] != 0)

{

x++;

}

if (x != 0)

{

printf("Самые популярные театры этого месяца:\n");

ViewTheatre(theatres, x);

}

else

{

printf("Театров в этом месяце нет\n");

}

free(theatres);

free(indexesT);

indexesT = (int\*)malloc(sizeof(int) \* nPer);

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

{

indexesT[i] = 0;

}

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

{

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

{

if (rec[i].p\_code == perf[j].p\_code)

{

indexesT[j]++;

}

}

}

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

{

for (int j = i; j < nPer; j++)

{

if (indexesT[i] < indexesT[j])

{

Performance temp = perf[i];

perf[i] = perf[j];

perf[j] = temp;

int t = indexesT[i];

indexesT[i] = indexesT[j];

indexesT[j] = t;

}

}

}

x = 0;

while (x < nPer && indexesT[x] != 0)

{

x++;

}

if (x != 0)

{

printf("Самые популярные постановки этого месяца:\n");

ViewPerformance(perf, x);

}

else

{

printf("Постановок в этом месяце нет\n");

}

system("pause");

free(perf);

free(arr);

free(rec);

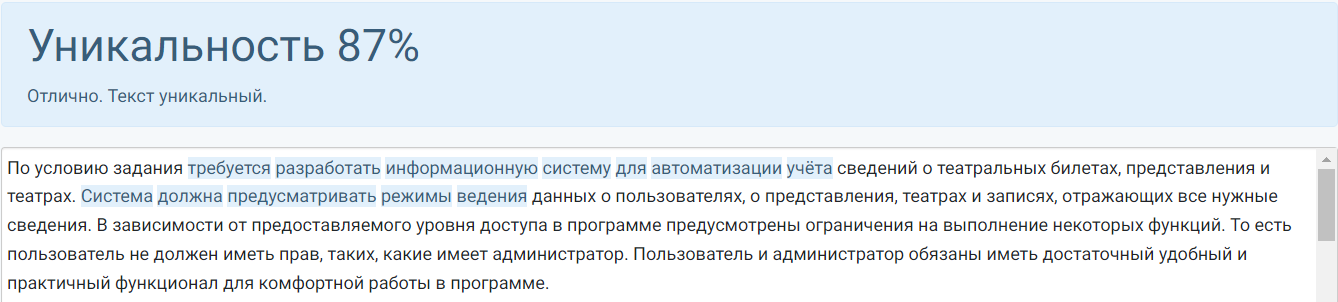
free(indexesT);

}

# **ПРИЛОЖЕНИЕ Б**

**(обязательное)**

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



# **ПРИЛОЖЕНИЕ В**

