ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc193488528)

[ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ГЛАВА 7](#_Toc193488529)

[1.1. Общие сведения 7](#_Toc193488530)

[1.2. Обоснование выбора среды и языка программирования разработки. 13](#_Toc193488531)

[1.3 .Обзор и выбор СУБД 16](#_Toc193488532)

[ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 22](#_Toc193488533)

[2.1 Проектирование и нормализация базы данных 22](#_Toc193488534)

[2.2. Разработка информационной подсистемы 25](#_Toc193488535)

[2.3 Тестирование созданной информационной подсистемы 31](#_Toc193488536)

[ЗАКЛЮЧЕНИЕ 34](#_Toc193488537)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc193488538)

# ВВЕДЕНИЕ

В рамках данного курсового проекта рассматривается задача разработки специализированной информационной системы для складского учета на заводе по производству автозапчастей. Основная цель проекта — создание программного обеспечения, способного эффективно управлять данными о товарных запасах, поступающих материалах, готовой продукции и их перемещениях на складе. Это позволит предприятию более точно контролировать остатки, оптимизировать процессы хранения и отгрузки, а также минимизировать ошибки, связанные с ручным учетом. Внедрение такой системы повысит оперативность работы склада, улучшит взаимодействие между отделами и повысит общую эффективность производственного процесса.

В процессе реализации проекта был проведён тщательный выбор подходящего языка программирования и сопутствующих технологий, которые наилучшим образом соответствуют требованиям разработки информационной системы для складского учета. Также был выполнен анализ различных систем управления базами данных (СУБД) для хранения и обработки информации о товарных запасах, материалах и готовой продукции. Это позволило определить наиболее подходящие инструменты, обеспечивающие удобное управление данными и быстрый доступ к ним, что особенно важно для эффективной работы склада на производственном предприятии.

Актуальность данной темы обусловлена тем, что для успешного функционирования завода по производству автозапчастей необходимо современное программное обеспечение, способное автоматизировать процессы складского учета. Такая система позволит минимизировать ошибки, связанные с ручным учетом, оптимизировать процессы хранения и отгрузки, а также улучшить взаимодействие между складом и другими подразделениями предприятия.

Практическая значимость работы заключается в том, что внедрение информационной системы для складского учета позволит повысить точность контроля остатков, ускорить процессы обработки заказов и снизить издержки, связанные с неправильным учетом или потерей данных. Это, в свою очередь, повысит общую эффективность работы предприятия и уровень удовлетворённости клиентов.

Информационная система для складского учета на заводе по производству автозапчастей предназначена для хранения и обработки данных о товарных запасах, поступающих материалах, готовой продукции и их перемещениях на складе. Основная цель системы — автоматизация процессов учета, что позволит минимизировать ошибки, связанные с ручным вводом данных, и повысить эффективность управления складскими операциями.

Цель работы — разработка программного обеспечения для автоматизации складского учета на производственном предприятии.

Задачи работы:

* провести анализ существующих решений и выбрать подходящую среду разработки;
* рассмотреть и выбрать систему управления базами данных (СУБД) для хранения информации о товарах и их перемещениях;
* описать функциональные требования к информационной системе;
* разработать UML-диаграммы для проектирования структуры системы;
* реализовать программу на языке программирования Python;
* подключить программу к выбранной СУБД (например, MySQL или PostgreSQL);
* провести тестирование системы для проверки её работоспособности и корректности обработки данных;
* сформулировать выводы на основе результатов разработки и тестирования.

Такой подход позволит создать надежное и удобное программное решение, которое улучшит процессы учета на складе и повысит общую эффективность работы предприятия.

Объект исследования — процесс организации складского учета на заводе по производству автозапчастей.

Предмет исследования — проектирование и разработка информационной системы для автоматизации учета материальных запасов и управления складскими операциями.

Методы исследования, используемые в данной курсовой работе, включают теоретический анализ, моделирование процессов и практическое тестирование. Такой комплексный подход позволяет детально изучить задачи, связанные с созданием системы складского учета, и предложить эффективные решения для их реализации.

Структура курсовой работы включает следующие основные разделы:

* **Введение:** В данном разделе обоснована актуальность выбранной темы, определены цель и задачи исследования. Подчеркнуты причины, по которым создание информационной системы для складского учета является важным и своевременным шагом для завода по производству автозапчастей. Разработка такой системы позволит оптимизировать процессы управления материальными запасами, снизить издержки, связанные с ошибками в учете, и повысить общую эффективность работы склада. В условиях высокой конкуренции и необходимости быстрого реагирования на изменения спроса внедрение автоматизированной системы становится неотъемлемой частью успешного функционирования предприятия.
* **Теоретическая часть**: включает обзор различных языков программирования, среди которых был выбран один язык как наиболее подходящий для реализации проекта. Также рассматриваются различные системы управления базами данных (СУБД), и объясняется, почему для данного проекта была выбрана именно эта база данных. Описывается область применения разрабатываемой подсистемы, её функциональные возможности и преимущества.
* **Практическая часть**: включает создание UML-диаграммы, которая иллюстрирует структуру и взаимодействие компонентов разрабатываемой подсистемы. Описывается процесс проектирования и реализации подсистемы с использованием выбранного языка программирования, а также её интеграция с базой данных.

**Заключение**:

В данном разделе подводятся итоги выполненной работы и оценивается эффективность разработанной информационной системы для складского учета на заводе по производству автозапчастей.

Разработанная информационная система для складского учета доказала свою эффективность в ходе тестирования, показав стабильную работу, интуитивно понятный интерфейс и высокую производительность. Внедрение системы позволит оптимизировать процессы управления запасами, сократить время выполнения операций и повысить точность учета, что в конечном итоге положительно скажется на общей эффективности работы завода.

В процессе выполнения курсового проекта были изучены современные подходы к проектированию и разработке информационных систем, а также проанализированы труды ведущих специалистов в области автоматизации и управления складскими процессами. Это позволило обеспечить научную обоснованность исследования и реализовать проект на высоком уровне. Использование актуальных методик и технологий подтвердило свою целесообразность, что делает разработанную систему готовой к практическому применению на производстве.

# ГЛАВА 1. ТЕОРЕТИЧЕСКАЯ ГЛАВА

* 1. **Общие сведения**

На заводе по производству автозапчастей существует острая необходимость в модернизации системы складского учета и управления материальными запасами. Основная цель проекта заключается в создании информационной системы, которая позволит автоматизировать процессы учета поступления, хранения и списания материалов, а также обеспечит контроль над остатками и движением товаров. Это поможет предприятию более оперативно управлять запасами, снизить издержки и повысить общую эффективность работы склада.

Сотрудники склада сталкиваются с проблемами при ручном учете материалов, что приводит к ошибкам, дублированию данных и замедлению выполнения операций. Текущая система учета не справляется с объемами работы, что негативно сказывается на производительности и точности данных. Новая информационная система призвана устранить эти недостатки, предоставив удобный и надежный инструмент для автоматизации складских процессов и улучшения контроля над запасами.

**Требования к ПО:**

**Функциональные требования:**

* + - Учёт и хранение информации о материалах и товарах (наименование, артикул, количество, сроки хранения, место нахождения на складе).
    - Управление данными о поставщиках и заказчиках (название компании, контактные данные, история поставок).
    - Контроль движения товаров (поступление, списание, перемещение между складами).
    - Формирование отчётов по остаткам, оборачиваемости материалов и выполненным операциям.
    - Анализ статистики по использованию материалов и эффективности работы склада.
    - Интеграция с другими системами предприятия (например, ERP или системами управления производством).
    - Возможность поиска и фильтрации данных по товарам, поставщикам и операциям.
    - Уведомления о критических событиях (например, низкий остаток товара или истечение срока хранения).

**Нефункциональные требования:**

* + - Высокая производительность и быстрая обработка больших объемов данных.
    - Удобный и интуитивно понятный интерфейс, адаптированный для пользователей с разным уровнем подготовки.
    - Надёжность и отказоустойчивость системы, обеспечивающие бесперебойную работу склада.
    - Защита данных с соблюдением требований законодательства (например, GDPR или локальных нормативов).
    - Масштабируемость системы для добавления новых функций и поддержки роста предприятия.
    - Кроссплатформенность, позволяющая использовать систему на различных операционных системах и устройствах.
    - Поддержка мобильных устройств для удалённого доступа и управления складскими процессами.

**Основание разработки:**

Основанием для разработки информационной системы является индивидуальное задание к курсовому проекту, которое предполагает создание решения для автоматизации складского учета на заводе по производству автозапчастей. Задание включает проектирование и реализацию системы, способной эффективно управлять процессами учета материалов, контроля остатков и обработки складских операций. Это позволит оптимизировать работу склада, повысить точность данных и снизить затраты на управление запасами.

**Что автоматизируется с помощью ПО:**

* Учёт материалов: автоматизация сбора, хранения и обработки данных о поступлении, списании и перемещении материалов на складе.
* Управление запасами: ведение базы данных по товарам, включая их характеристики, количество и местоположение на складе.
* Управление поставками: автоматизация процессов учёта поставщиков, заказов и контроля выполнения договоров.
* Анализ складских операций: автоматический анализ данных для выявления узких мест, формирования отчётов и оптимизации процессов.
* Уведомления: автоматические оповещения о критических событиях, таких как низкий остаток товара или истечение срока хранения.
* Поиск и фильтрация данных: быстрый доступ к информации о материалах, поставщиках и складских операциях.

Информационная система представляет собой комплекс программных средств и данных, обеспечивающих автоматизацию процессов складского учёта и контроля на предприятии. Она предназначена для оптимизации управления материальными запасами, повышения точности данных и снижения издержек, связанных с ручным учётом.

**Виды информационных подсистем:**

* **Структурированная подсистема:** Это задача, в которой известны все элементы и их взаимосвязи. В структурированной задаче содержание можно выразить в форме математической модели с чётким алгоритмом решения. Такие задачи носят рутинный характер и решаются многократно. Цель использования информационной подсистемы для решения структурированных задач — полная автоматизация процесса, минимизирующая участие человека.
* **Неструктурированная подсистема:** Это задача, в которой невозможно выделить элементы и установить между ними связи. Решение неструктурированных задач связано с трудностями из-за отсутствия математического описания и чёткого алгоритма. Возможности использования информационной подсистемы в таких случаях ограничены и основываются на косвенной информации из различных источников.
* **Экспертная подсистема:** Это программа, которая ведёт себя подобно эксперту в узкой прикладной области. Типичные применения включают медицинскую диагностику и локализацию неисправностей в оборудовании. Такие системы используют базу знаний и логические правила для анализа данных и принятия решений, что позволяет им имитировать процесс мышления специалиста.

**Классификация подсистем по степени автоматизации:**

* **Ручные информационные подсистемы:** Характеризуются отсутствием современных технических средств обработки информации, все операции выполняются вручную человеком. Это приводит к низкой скорости выполнения задач, высокой вероятности ошибок и ограниченным возможностям для анализа данных.
* **Автоматические информационные подсистемы:** выполняют все операции по переработке информации без участия человека.
* **Автоматизированные информационные подсистемы:** предполагают участие как человека, так и технических средств, при этом главная роль отводится компьютеру.

**Классификация подсистем по характеру использования информации:**

* **Информационно-поисковые системы:** Осуществляют ввод, систематизацию, хранение и выдачу информации по запросу пользователя без сложных преобразований данных. Примеры включают системы в библиотеках, кассах продажи билетов или базах данных для учёта клиентов. Такие системы предназначены для работы с большими объёмами информации, обеспечивая быстрый доступ к ней и удобное управление.
* **Информационно-решающие системы:** выполняют все операции переработки информации по определенному алгоритму. Среди них выделяют управляющие и советующие системы в зависимости от степени воздействия выработанной информации на процесс принятия решений.

**Разработка программы**

После анализа различных языков программирования будет выбран подходящий язык программирования. На этом языке будет разработана программа база данных «Учет клиентов и их потребностей» с использованием СУБД для взаимодействия SQL.

**Принцип действия информационной подсистемы:**

Программа позволяет по имени клиента выводить всю информацию о нем, включая:

* Подразделение, в котором он обслуживается.
* Количество и тип страховых полисов.
* Контактную информацию.
* Общую стоимость страховых услуг.

Эта система обеспечивает эффективное управление клиентскими данными, улучшая качество обслуживания и повышая оперативность работы сотрудников страхового агентства.

**Аналоги систем складского учета на производственных предприятиях включают:**

WMS (Warehouse Management Systems):

Примеры: SAP EWM, Oracle WMS, 1C:Управление складом. Эти системы предназначены для автоматизации процессов управления складскими операциями, включая учёт товаров, контроль остатков и оптимизацию хранения.

ERP-системы с модулями для управления запасами:

Примеры: SAP ERP, Oracle ERP, Microsoft Dynamics. Такие системы предоставляют функционал для учёта материалов, управления заказами и анализа данных по складским операциям.

Специализированные системы для учёта запасов:

Примеры: TradeGecko, Cin7, Odoo. Эти решения ориентированы на автоматизацию учёта товаров, управление поставками и формирование отчётов по складской деятельности.

MRP-системы (Material Requirements Planning):

Примеры: MRPeasy, NetSuite, Fishbowl. MRP-системы помогают планировать потребности в материалах, управлять запасами и оптимизировать производственные процессы.

Интегрированные платформы для управления логистикой:

Примеры: LogiNext, 3PL Central, Descartes. Такие системы обеспечивают комплексное управление складскими и логистическими процессами, включая учёт товаров и контроль перемещений.

Каждая из этих систем предлагает уникальные возможности для автоматизации складского учёта, что позволяет предприятиям выбирать решение, наиболее подходящее под их потребности.

**Преимущества разработанной системы**

* **Индивидуальная настройка** – Возможность адаптации системы под уникальные требования и процессы конкретного завода по производству автозапчастей.
* **Локальное размещение** – Хранение данных на локальных серверах обеспечивает полный контроль над информацией и повышает уровень безопасности.
* **Простота использования** – Интуитивно понятный интерфейс, который минимизирует время на обучение сотрудников и повышает их производительность.
* **Гибкость и масштабируемость** – Возможность легко расширять функционал системы и адаптировать её под растущие потребности предприятия.
* **Экономическая эффективность** – Отсутствие необходимости в дорогостоящих лицензиях и подписках, что снижает общие затраты на внедрение и поддержку системы.
* **Высокий уровень автоматизации** – Автоматизация рутинных задач, таких как учёт материалов и формирование отчётов, что позволяет сотрудникам сосредоточиться на стратегически важных задачах.
* **Безопасность данных** – Встроенные механизмы защиты информации, включая шифрование данных и контроль доступа, для обеспечения конфиденциальности и целостности данных.

## **Обоснование выбора среды и языка программирования разработки.**

Для работы с базами данных применяются специализированные языки. Среди них:

SQL («язык структурированных запросов») — предназначен для работы с реляционными базами данных. Появился в 1974 году и с тех пор претерпел множество изменений, но основные принципы остались неизменными.

Преимущества:

* эффективен при работе с запросами, обновлениями и обработкой реляционных баз данных;
* декларативный синтаксис делает SQL читаемым языком.

Недостатки:

* синтаксис SQL может показаться сложным для тех, кто привык к императивному программированию;
* существует множество различных версий SQL, таких как PostgreSQL, SQL, MariaDB. Они несовместимы между собой.

R язык программирования с открытым исходным кодом, который используется для анализа данных и работы с графикой. Изначально R был разработан сотрудниками Оклендского университета Россом Айхэкой и Робертом Джентлменом. Они хотели создать язык, который помог бы им в процессе обучения студентов основам статистики. R был создан под влиянием языка S, который был разработан компанией Bell Labs в 1970е годы.

R считается предметно ориентированным языком, поскольку он был создан преимущественно для анализа данных. R можно использовать как в виде интерактивной среды, так и в виде скриптов и моделей, которые могут быть встроены в пакеты и интегрированы с другими программными модулями.

Язык R можно использовать для анализа данных из различных источников, включая внешние файлы или базы данных.

Преимущества:

* R имеет широкий набор высококачественных предметно ориентированных пакетов с открытым исходным кодом. R предоставляет пакеты для любого количественного и статистического применения, включая нейронные сети, нелинейную регрессию, филогенетику, построение сложных диаграмм и графиков.
* вместе с базовой установкой предоставляется возможность установки обширных встроенных функций и методов. Кроме того, R обрабатывает данные матричной алгебры.
* возможность визуализации данных и различных библиотек, например ggplot2.

Недостатки:

* низкая производительность;
* специфичность. R подходит для статистических исследований и науки о данных, но не подходит для общего программирования.

Julia — высокоуровневый, высокопроизводительный язык программирования с динамической типизацией, предназначенный для математических вычислений. Его синтаксис похож на синтаксис других сред технических вычислений, таких как Matlab и Octave. Язык написан на Си, C++ и Scheme.

Преимущества:

* Julia — это скомпилированный язык JIT, который может обеспечить высокую производительность. Язык поддерживает возможности динамической типизации и сценариев интерпретируемого языка, такого как Python.
* Хорошо подходит для численного анализа и может рассматриваться как язык программирования общего назначения.

Недостатки:

* Julia — новый язык, поэтому разработчики могут столкнуться с проблемами при работе с его пакетами.
* Ограниченное количество пакетов программ.

Java — популярный язык общего назначения, который работает на виртуальной машине Java Virtual Machine (JVM). Это абстрактная вычислительная система, которая обеспечивает переносимость между платформами. В настоящее время поддерживается корпорацией Oracle.

Преимущества:

* Универсальность. Многие современные системы и приложения разработаны с использованием Java. Преимущество такого языка заключается в том, что он позволяет интегрировать методы науки о данных непосредственно в существующую кодовую базу.
* Строгая типизация. Обеспечение типобезопасности для Java особенно важно при разработке важных приложений для работы с большими данными.

Недостатки:

* Java не подходит для специальных анализов и разработки специализированных статистических приложений.
* Java не имеет большого количества библиотек для передовых статистических методов по сравнению с некоторыми предметно ориентированными языками, такими как R.

Python — это универсальный язык программирования высокого уровня, который широко используется для различных проектов разработки. Он был создан и выпущен в 1991 году после разработки в конце 80х годов. Python используется в различных приложениях и программах, включая веб и мобильную разработку, Анализ данных и машинное обучение**,** Системное администрирование. Это объектно-ориентированный язык с открытым исходным кодом, который был разработан с помощью языка C.

Scala — это язык программирования, который был разработан для создания компонентного программного обеспечения. Он был создан для того, чтобы обеспечить лёгкость и быстроту разработки, а также для того, чтобы сделать код более читаемым и понятным. В Scala используются как функциональные, так и объектно-ориентированные парадигмы программирования.

Преимущества:

* Использование Scala и Spark позволяет работать с высокопроизводительными кластерными вычислениями.
* Мультипарадигматический. В Scala доступны как объектноориентированные, так и функциональные парадигмы программирования
  1. **Обзор и выбор СУБД**

Рассмотрим программное обеспечение – базу данных для хранения информации о заявках на техническое обслуживание, а также для создания базы данных об оборудовании, исполнителях и истории выполнения работ на протяжении всего периода взаимодействия с клиентами. Для проектирования этой системы проведём анализ различных программных решений и выберем наиболее подходящее.

Среди наиболее популярных систем управления базами данных (СУБД) для решения подобных задач выделяются следующие:

1. SQL – это встраиваемая реляционная база данных, разработанная на языке программирования C. Она предоставляет широкий набор команд SQL. В отличие от большинства других баз данных, работающих по схеме клиент/сервер, SQL использует динамическую типизацию данных. Это означает, что тип данных в столбце не ограничивает тип значения, которое может быть в него записано.

Преимущества:

* Файловая структура: Одним из ключевых преимуществ SQL является то, что вся база данных хранится в одном файле. Это упрощает переносимость, так как файл можно легко копировать между устройствами. Резервное копирование также становится простым — достаточно сделать копию файла для сохранения всех данных.
* Кроме того, отсутствие необходимости в отдельном сервере делает SQL удобным для разработки и использования в мобильных приложениях и встроенных системах.
* Стандарты: SQL поддерживает большинство стандартных SQLоператоров и конструкций, что позволяет разработчикам использовать знакомый синтаксис для работы с базами данных.
* Удобство для разработки и тестирования: Благодаря своей простоте (один файл и библиотека на C), SQL идеально подходит для разработки и тестирования приложений, особенно когда требуется масштабирование и быстрая настройка среды. Эта легковесная база данных позволяет разработчикам сосредоточиться на создании функциональности без необходимости в сложной конфигурации. Кроме того, она обеспечивает высокую производительность и простоту в использовании, что делает ее отличным выбором для проектов любого размера.

Недостатки:

* Отсутствие системы пользователей: Нет возможности настройки прав доступа для разных пользователей.
* Ограниченная производительность: Нет механизмов для увеличения производительности при работе с большими объемами данных.

1. MySQL – это реляционная СУБД с открытым исходным кодом, которая поддерживает табличные базы данных как с простой, так и со сложной структурой. Она отличается высокой скоростью обработки данных, гибкостью и простотой интерфейса. MySQL широко используется в ecommerce, IT и финтехе для построения прогностических моделей.

Преимущества:

* Простота: Легко устанавливается и настраивается. Существует множество сторонних инструментов, упрощающих работу с базой данных.
* Функциональность: Поддерживает большую часть стандартных функций SQL.
* Безопасность: Встроенные механизмы безопасности обеспечивают защиту данных.
* Масштабируемость: MySQL обладает высокой производительностью и способна эффективно обрабатывать большие объемы данных. Эта система управления базами данных (СУБД) идеально подходит для разработки масштабируемых приложений, которые могут расти и развиваться с течением времени. Благодаря своей надежности и поддержке множества пользователей одновременно, MySQL становится отличным выбором для проектов, требующих стабильности и высокой доступности.

Недостатки:

* Ненадежность: Некоторые операции менее надежны по сравнению с другими СУБД.

1. Microsoft SQL Server – это СУБД, разработанная компанией Microsoft, которая поддерживает операционные системы Windows и Linux. Она характеризуется простым интерфейсом, высокой надежностью и совместимостью с другими продуктами Microsoft, такими как Excel и Access.

Преимущества:

* Масштабируемость: Подходит как для портативных устройств, так и для мощных серверов.
* Высокая производительность: Быстрое извлечение данных благодаря размеру страниц до 8 КБ.
* Автоматизация: Рутинные задачи, такие как управление памятью и блокировками, автоматизированы.
* Интеграция: Поддерживает работу с другими продуктами Microsoft.

Недостатки:

* Высокая стоимость: Использование в коммерческих целях требует оплаты лицензии.

1. Oracle – это объектнореляционная СУБД, известная своей надежностью, практичностью и высокой производительностью. Она широко используется в Data Mining для анализа эффективности продаж и построения моделей потребностей клиентов.

Недостатки:

* Сложность: Требует глубоких знаний для настройки и управления.
* Высокая стоимость: Приобретение и поддержка Oracle обходятся дорого.

1. Redis (Remote Dictionary Server) – это СУБД с открытым исходным кодом, которая поддерживает хранение данных в памяти. Она используется для кэширования, хранения сессий пользователей, очередей и промежуточных результатов вычислений.

Преимущества:

* Высокая скорость: Все операции выполняются в памяти, что обеспечивает минимальные задержки.
* Гибкость: Поддерживает различные структуры данных, такие как строки, списки, хэши и наборы.
* Простота использования: Удобные форматы запросов и поддержка скриптов на LUA.

Недостатки:

* Ограниченный объем данных: Размер базы данных ограничен объемом доступной памяти.
* Отсутствие контроля доступа: Нет разделения прав доступа для пользователей.
  + В результате анализа была выбрана СУБД SQL по следующим причинам:
* Отсутствие необходимости в администрировании: SQL не требует сложной настройки и управления.
* Простота подключения: Легко интегрируется в приложения.
* Высокая производительность: При достаточном объеме оперативной памяти обеспечивает быструю обработку данных.
* Бесплатность: SQL распространяется бесплатно, что делает её экономически выгодным решением.

**Использование библиотеки и технологии**

Для разработки информационной системы складского учета были использованы следующие библиотеки и технологии на языке Python:

* SQLite3 – встроенная библиотека для работы с базой данных SQLite, обеспечивающая хранение и управление данными о клиентах и страховых продуктах.
* Tkinter – стандартная библиотека для создания графического интерфейса пользователя (GUI), которая была использована для разработки удобного и интуитивно понятного интерфейса.
* Pandas – библиотека для обработки и анализа данных, которая может быть использована для более сложного анализа потребностей клиентов.
* Matplotlib/Seaborn – библиотеки для визуализации данных, которые могут быть применены для создания графиков и диаграмм в аналитических отчётах.
* Unittest/Pytest – библиотеки для модульного и интеграционного тестирования, обеспечивающие проверку корректности работы системы.
* PyInstaller – инструмент для сборки программы в исполняемый файл, что упрощает её распространение и использование.
* Datetime – встроенная библиотека для работы с датами и временем, которая может быть использована для уведомлений о событиях (например, окончание срока действия страховки).
* Email/SMTPLib – библиотеки для реализации функционала отправки уведомлений клиентам по электронной почте.
* Hashlib – библиотека для шифрования данных, которая может быть использована для защиты конфиденциальной информации

**1.4 Описание этапа разработки**

Анализ требований:

На этапе анализа требований определяются задачи, которые должна решать информационная система для складского учета. Основные требования включают:

* Учёт и хранение данных о материалах и товарах (наименование, артикул, количество, сроки хранения).
* Управление информацией о поставщиках и заказчиках (название компании, контактные данные, история поставок).
* Контроль движения товаров (поступление, списание, перемещение между складами).
* Анализ данных на основе истории операций и остатков.
* Формирование отчётов по остаткам, оборачиваемости материалов и выполненным операциям.
* Удобный интерфейс для сотрудников, позволяющий быстро находить информацию и управлять данными.
* Надёжное хранение данных с использованием PostgreSQL или MySQL.
* Высокая производительность и минимальное количество ошибок при обработке запросов.

Определение спецификаций:

Для разработки системы используются следующие средства и технологии:

* Язык программирования: Python.
* База данных: PostgreSQL или MySQL (для хранения и управления данными).
* Графический интерфейс: Библиотека Tkinter или PyQt для создания десктопного приложения, либо Django для веб-интерфейса.
* Средства разработки:
* IDE: PyCharm или Visual Studio Code.
* Система контроля версий: Git (например, через GitHub или GitLab).
* Библиотеки для работы с базами данных: SQLAlchemy или psycopg2.
* Дополнительные инструменты:
* Библиотеки для тестирования (например, unittest или pytest).
* Инструменты для создания документации (например, Sphinx).

Программирование:

* Проектирование архитектуры программы:
* Определение структуры программы (модули, классы, функции).
* Разработка схемы базы данных (таблицы, связи между ними).

Планирование интерфейса пользователя (основные окна, кнопки, формы).

* Реализация функционала:
* Создание модулей для учёта материалов, управления поставками и контроля движения товаров.
* Разработка API для интеграции с другими системами предприятия.
* Настройка автоматических уведомлений и отчётов.
* Тестирование и отладка:
* Проведение модульного и интеграционного тестирования.
* Оптимизация производительности и устранение ошибок.
* Документирование:
* Создание технической документации и руководства для пользователей.

**ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА**

## **Проектирование и нормализация базы данных**

**База данных (БД)**– это инструмент для накопления, организации и управления большими объёмами информации о материалах, поставщиках, заказчиках и складских операциях. Она должна обеспечивать актуальность данных, их долгосрочное хранение и предоставлять пользователям быстрый и удобный доступ к информации. Это позволяет эффективно управлять складскими процессами, оперативно обрабатывать запросы и анализировать данные для повышения качества работы склада.

Разработка базы данных была выполнена с использованием системы управления реляционными базами данных **PostgreSQL**. База данных хранит всю необходимую информацию для работы системы, включая данные о материалах, поставщиках, заказчиках и операциях. Ниже приведено описание таблиц, их назначение и связи между ними.

**Описание таблиц базы данных:**

* **Материалы (Materials):**

**Назначение:** Хранение информации о материалах и товарах.

**Поля:**

* id(уникальный идентификатор),
* name (наименование материала),
* article (артикул),
* quantity (количество),
* storage\_location (место хранения),
* expiry\_date (срок годности).
* **Поставщики (Suppliers):**

**Назначение:** Хранение данных о поставщиках.

**Поля:**

* + - Id (уникальный идентификатор),
    - Name (название компании),
    - contact\_info (контактные данные),
    - address (адрес).
* **Заказчики (Customers):**

**Назначение:** Хранение данных о заказчиках.

**Поля:**

* + - id(уникальный идентификатор),
    - name (название компании),
    - contact\_info (контактные данные),
    - address (адрес).
* **Операции (Operations):**

**Назначение:** Хранение информации о складских операциях (поступление, списание, перемещение).

**Поля:**

* + - Id (уникальный идентификатор),
    - material\_id (ссылка на материал),
    - operation\_type (тип операции),
    - quantity (количество),
    - date (дата операции),
    - supplier\_id (ссылка на поставщика, если операция — поступление),
    - customer\_id (ссылка на заказчика, если операция — списание).

**Корректность данных:**

Схема данных обеспечивает целостность информации: например, нельзя удалить материал, если он участвует в активных операциях. Связи между таблицами позволяют быстро находить нужные данные, например, все операции с определённым материалом или информацию о поставщике для конкретной поставки. Программа упрощает процесс работы с данными, минимизируя вероятность ошибок.

Проектирование информационной системы происходит на языке программирования Python. Для разработки используется среда PyCharm. В редакторе добавляется модуль для работы с базой данных, например, `sqlite3` (для SQLite) или `psycopg2` (для PostgreSQL).

Для подключения к базе данных создаётся объект соединения с помощью метода `connect()`. Затем создаётся объект курсора с использованием метода `cursor()`. Курсор необходим для выполнения SQL-запросов.

После установления соединения файл базы данных создаётся автоматически, если он отсутствует. Этот файл сохраняется на диске.

Для создания таблиц используется SQL-запрос `CREATE TABLE IF NOT EXISTS`. В запросе указываются названия столбцов и их типы данных. Например, создаётся таблица для хранения информации о материалах, включая их наименование, артикул, количество, место хранения и срок годности.

Для выборки данных применяется оператор `SELECT`. Метод `fetchall()` объекта курсора сохраняет результаты в переменной, которая представляет собой список. Каждая строка из базы данных становится отдельным элементом списка. Затем выполняется перебор значений переменной и их вывод.

Метод `commit()` сохраняет все сделанные изменения в базе данных. После завершения работы с базой данных рекомендуется закрыть соединение с помощью метода `close()`.

Таким образом, программа обеспечивает создание, управление и выборку данных в системе складского учета. Она проста в использовании, надёжна и легко масштабируется под нужды предприятия.

**2.2. Разработка информационной подсистемы**

Информационная подсистема является ключевым компонентом программного обеспечения, предназначенного для автоматизации процессов складского учета. Её основная задача — обеспечение эффективного управления данными о материалах, их хранении, перемещении и анализе. Разработка подсистемы включает несколько этапов:

* + Определение структуры данных:

Определение, какие данные необходимо хранить и обрабатывать в системе складского учета. Например:

* Информация о материалах и товарах (наименование, артикул, количество, сроки хранения).
* Данные о поставщиках и заказчиках (название компании, контактные данные, история поставок).
* Информация о складских операциях (поступление, списание, перемещение).
* Финансовые данные (стоимость материалов, оплата поставок).

**Определение сущностей и их атрибутов:**

**Материалы:**

* + Наименование, артикул, количество, место хранения, срок годности.

**Поставщики:**

* + Название компании, контактные данные, адрес, история поставок.

**Заказчики:**

* + Название компании, контактные данные, адрес, история заказов.

**Операции:**

* + Тип операции (поступление, списание, перемещение), количество, дата, ссылка на материал, поставщика или заказчика.

**Связи между сущностями:**

* Материалы связаны с операциями через внешний ключ material\_id.
* Поставщики связаны с операциями поступления через внешний ключ supplier\_id.
* Заказчики связаны с операциями списания через внешний ключ customer\_id.

**Разработка схемы базы данных:**

Создание структуры базы данных с учётом нормализации для минимизации дублирования данных и обеспечения целостности. Например:

* Таблица **Materials** хранит информацию о материалах.
* Таблица **Suppliers** хранит данные о поставщиках.
* Таблица **Customers** хранит данные о заказчиках.
* Таблица **Operations** связывает материалы, поставщиков и заказчиков через внешние ключи.

Проектирование базы данных:

Таблица Materials:

Хранение информации о материалах и товарах (наименование, артикул, количество, место хранения, срок годности).

Таблица Suppliers:

Хранение данных о поставщиках (название компании, контактные данные, адрес).

Таблица Customers:

Хранение данных о заказчиках (название компании, контактные данные, адрес).

Таблица Operations:

Связь материалов с поставщиками и заказчиками, хранение данных о складских операциях (тип операции, количество, дата).

Определение первичных и внешних ключей:

Первичные ключи: `id` в каждой таблице.

Внешние ключи:

* + material\_id` в таблице Operations ссылается на таблицу Materials.
  + supplier\_id` и `customer\_id` в таблице Operations ссылаются на таблицы Suppliers и Customers соответственно.

Разработка системы управления складом и поставками представляет собой комплексный процесс, направленный на создание эффективного инструмента для учёта, анализа и управления материальными ресурсами. Для достижения поставленных целей необходимо реализовать ряд ключевых этапов, включая проектирование базы данных, разработку функциональных модулей, интеграцию с пользовательским интерфейсом, обеспечение безопасности данных и проведение всестороннего тестирования.

На начальном этапе проектирования системы особое внимание уделяется созданию индексов в базе данных, которые позволяют значительно ускорить выполнение запросов. Например, индексы по наименованию материала, названию поставщика и артикулу товара обеспечивают быстрый доступ к необходимым данным, что особенно важно при работе с большими объёмами информации. Это позволяет оптимизировать процессы поиска и фильтрации, снижая нагрузку на систему и повышая её производительность.

Далее разрабатывается модуль для взаимодействия с базой данных, который включает функции для добавления, удаления и редактирования информации. В частности, реализуются возможности добавления новых материалов и поставщиков, обновления данных о количестве товара на складе, а также удаления устаревших записей. Кроме того, модуль обеспечивает функции поиска и фильтрации, такие как поиск материалов по артикулу или фильтрация по сроку годности, что позволяет пользователям оперативно находить необходимую информацию. Для анализа данных предусмотрены функции формирования отчётов, например, отчётов по остаткам материалов или оборачиваемости товаров, а также механизмы для выявления наиболее востребованных материалов и анализа эффективности работы склада.

Интеграция подсистемы с графическим интерфейсом является важным этапом, обеспечивающим удобство взаимодействия пользователей с системой. Интерфейс включает таблицы с информацией о материалах, поставщиках и операциях, а также формы для ввода и редактирования данных. Для визуализации аналитической информации предусмотрены графики и диаграммы, такие как графики оборачиваемости материалов или диаграммы по объёмам поставок. Пользовательский интерфейс также позволяет обрабатывать запросы, например, поиск поставщика по названию компании или фильтрацию материалов по месту хранения.

Обеспечение безопасности данных является неотъемлемой частью разработки системы. Для защиты конфиденциальной информации, такой как контактные данные поставщиков и заказчиков, применяются современные методы шифрования, например, алгоритм AES. Кроме того, реализуется система разграничения прав доступа, которая предусматривает различные уровни доступа для разных ролей пользователей, таких как администратор, кладовщик и аналитик. Это позволяет минимизировать риски несанкционированного доступа к данным.

Завершающим этапом разработки является тестирование подсистемы, которое включает модульное тестирование, проверку корректности SQL-запросов и тестирование производительности. Модульное тестирование позволяет убедиться в корректности работы отдельных функций, таких как добавление материала или поиск поставщика. Проверка SQL-запросов направлена на обеспечение правильности сохранения и извлечения данных из базы. Тестирование производительности проводится для оценки скорости обработки запросов при больших объёмах данных. По результатам тестирования выявленные ошибки устраняются, а система оптимизируется для повышения её эффективности.

В результате реализации всех этапов создаётся надёжная и производительная система управления складом и поставками, которая обеспечивает удобство работы пользователей. Такая система способна значительно повысить эффективность управления материальными ресурсами и обеспечить стабильную работу склада.

Для реализации приложения был выбран язык программирования Python, который является одним из самых популярных и современных языков благодаря своей простоте, гибкости и широкому набору библиотек и фреймворков. Python обеспечивает высокую производительность и подходит для разработки как небольших, так и крупных проектов.

Разработка велась в интегрированной среде разработки PyCharm, которая предоставляет удобные инструменты для написания, отладки и тестирования кода. PyCharm поддерживает множество функций, таких как автодополнение, анализ кода и интеграция с системами контроля версий, что значительно ускоряет процесс разработки.

Для хранения данных использовалась система управления базами данных PostgreSQL, которая отличается надёжностью, масштабируемостью и возможностью интеграции с другими инструментами. PostgreSQL поддерживает сложные запросы, транзакции и обеспечивает высокий уровень безопасности данных, что делает её идеальным выбором для проектов, требующих работы с большими объёмами информации.

Чтобы начать работу с приложением, нужно ввести логин и пароль пользователя. При открытии программы появляется страница входа в систему. В поле «Логин» клиент или администратор вводит свой логин, а в поле «Пароль» свой пароль.

Изображение выглядит как снимок экрана, фиолетовый, Фиолетовый, Пурпурный цвет

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Окно авторизации пользователя системы.

После успешного входа в учётную запись пользователь попадает на главную страницу приложения. Здесь он может ознакомиться со всеми доступными товарами, найти нужный с помощью поиска, а также добавлять, удалять и изменять уже существующие позиции, указывая артикул, название, категорию и количество.

Изображение выглядит как снимок экрана, программное обеспечение, текст, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Главное окно.

Дальше продемонстрировано добавление товара.

Изображение выглядит как снимок экрана, текст, фиолетовый, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 7 – Окно добавление заявки.

Также можно добавлять и удалять накладные. Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 8 – Окно добавления накладных.

В руководстве пользователя есть раздел, который объясняет, как использовать программу. При нажатии на этот раздел появляется дополнительное окно с информацией.

Изображение выглядит как текст, электроника, снимок экрана, компьютер

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 9 – Окно добавления накладных.

**2.3 Тестирование созданной информационной подсистемы**

Тестирование информационной подсистемы является важным этапом разработки, направленным на обеспечение корректности работы всех её компонентов. Процесс тестирования включает следующие этапы:

Модульное тестирование направлено на проверку отдельных функций и модулей подсистемы. Оно включает тестирование функций для добавления, удаления и редактирования данных в базе данных, а также проверку корректности выполнения SQL-запросов, таких как выборка данных по определённым критериям.

Интеграционное тестирование проверяет взаимодействие между различными компонентами подсистемы. Это включает тестирование связи между графическим интерфейсом и базой данных, а также проверку корректности передачи данных между модулями, например, от интерфейса к базе данных и обратно.

Системное тестирование оценивает работу подсистемы в целом. Оно охватывает тестирование всех сценариев использования, таких как добавление нового клиента, поиск страхового продукта или формирование отчёта. Также проверяется производительность подсистемы, включая скорость обработки запросов при работе с большими объёмами данных.

Тест-кейс представляет собой детально описанный алгоритм проверки функциональности или системы, включающий последовательность шагов и необходимые входные данные, которые должны привести к ожидаемому результату. Такие кейсы помогают оценить, как система ведет себя в различных сценариях использования, что крайне важно для обеспечения ее надежности и стабильности, особенно при работе с данными, связанными с гидроэлектростанциями.

Тестирование безопасности направлено на проверку механизмов защиты данных. Это включает тестирование шифрования конфиденциальной информации и проверку ограничения доступа к данным для разных ролей пользователей, чтобы убедиться в надёжности системы.

Ручное тестирование фокусируется на проверке интерфейса подсистемы. Оно оценивает удобство использования форм для ввода и редактирования данных, корректность отображения данных в таблицах и списках, а также обработку ошибок, например, при вводе некорректных данных.

Подсистема считается готовой, если все её функции работают корректно, она стабильно функционирует без критических ошибок, данные корректно сохраняются и извлекаются из базы данных, а также если она успешно проходит все этапы тестирования без существенных замечаний.

**Описание тест-кейса №1**

|  |  |
| --- | --- |
| Название | Проверка редактирования товара |
| ID теста | TC001 |
| Описание | Данный тест проверят, что при не выборе товара, система выдаст сообщение об неправильном действии |
| Ожидаемый результат | При не выборе товара, система выведет сообщение «Выберите товар для редактирования» |
| Выполняемые действия | 1.Открыть приложение  2.Открыть страницу с товаром  3.Нажать на кнопку «Редактировать товара» |
| Фактический результат | При не выборе товара, система выведет сообщение «Выберите товар для редактирования» |

**Заключение**

В рамках данного курсового проекта рассматривается задача создания специализированной программы для учёта заявок на техническое обслуживание оборудование. Система включает базу данных на основе SQLite, функции для работы с данными и удобный графический интерфейс, созданный с использованием Python.

**Основные результаты проекта:**

* Создана эффективная система для автоматизации складского учета на заводе по производству автозапчастей.
* Разработан простой и удобный интерфейс, который облегчает работу сотрудников склада.
* Проведено тестирование системы, подтвердившее её стабильность и готовность к использованию.

Благодаря этой системе можно:

* Повысить эффективность управления запасами.
* Сократить время выполнения складских операций.
* Уменьшить количество ошибок, связанных с ручным учётом данных.
* Обеспечить прозрачность и контроль над движением товаров.

Система готова к внедрению и способна значительно улучшить процессы работы склада.

**Планы по доработке системы:**

**Расширение функционала:**

* Добавить возможность интеграции с онлайн платформами, например, с вебсайтом агентства.
* Реализовать отправку уведомлений клиентам через SMS или электронную почту.
* Разработать мобильную версию приложения для удобства работы сотрудников вне офиса.

**Улучшение аналитических возможностей:**

* Внедрить более сложные алгоритмы для анализа данных, например, прогнозирование спроса на страховые продукты.
* Добавить визуализацию данных (графики, диаграммы) для более наглядного представления информации.

Оптимизация производительности:

* Улучшить скорость обработки запросов, особенно при работе с большими объёмами данных.
* Оптимизировать базу данных для более эффективного хранения и поиска информации.

Усиление безопасности**:**

* Внедрить двухфакторную аутентификацию для повышения безопасности доступа к системе.
* Реализовать шифрование данных при передаче между клиентом и сервером.

Таким образом, цели и задачи курсового проекта были успешно достигнуты. Разработанная система готова к внедрению и может значительно улучшить работу страхового агентства. Дальнейшая доработка сделает её ещё более функциональной, удобной и безопасной, что повысит её ценность как для компании, так и для её клиентов.

**Список использованных источников**

* Антонова, А. В. Компьютерное тестированиесовременная форма контроля учебных достижений учащихся в процессе обучения истории в школе / А. В. Антонова // В сборнике: Инновации и традиции педагогической науки2019. Сборник трудов ХIХ Международной научнопрактической конференции / под ред. Т. А. Макаренко, С. В. Паниной. — 2019. — С. 2327. — Текст : электронный // elibrary.ru [сайт]. — URL: https://www.elibrary.ru/download/elibrary\_37326820\_65482003.pdf (дата обращения: 04.02.2023).
* Болье, А. Изучаем SQL. Генерация, выборка и обработка данных / А. Болье. — М.: Издательство ДиалектикаВильямс, 2020. — 423 с. — (Профессиональное образование). — Текст: электронный // ЭБС МГТУ им. Баумана [сайт]. — URL: https://elearning.bmstu.ru/iu6/pluginfile.php/18674/mod\_resource/content/1/Изучаем\_SQL\_Генерация%2C\_выборка\_и\_обработка\_данных%2C\_3\_изд.pdf (дата обращения: 01.02.2023).
* Заельская, С. А. История отечества с древнейших времён и до наших дней в тестах и заданиях: электронное учебнометодическое пособие для бакалавров заочного отделения / С. А. Заельская. — Оренбург, 2019. — Текст : электронный // elibrary.ru [сайт]. — URL: https://lib.rucont.ru/efd/702462/info (дата обращения: 04.02.2023).
* Зарапин, Р. В. Образовательный потенциал использования онлайнтестирования при изучении истории / Р. В. Зарапин // В сборнике: Актуальные вопросы гуманитарных наук: теория, методика, практика. Сборник научных статей IХ Всероссийской научнопрактической конференции с международным участием. — Москва, 2022. — С. 239244. — Текст : электронный // elibrary.ru [сайт]. — URL: https://www.elibrary.ru/download/elibrary\_50163440\_95910012.pdf (дата обращения: 04.02.2023).
* Маркин, А. В. Программирование на SQL в 2 ч. Часть 1 : учебник и практикум для вузов / А. В. Маркин. — 3е изд., перераб. и доп. — М. : Издательство Юрайт, 2023. — 429 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/509818 (дата обращения: 04.02.2023).
* Советов, Б. Я. Базы данных : учебник для вузов / Б. Я. Советов, В. В. Цехановский, В. Д. Чертовской. — 3е изд., перераб. и доп. — М. : Издательство Юрайт, 2023. — 420 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/510752 (дата обращения: 04.02.2023).
* Стасышин, В. М. Базы данных: технологии доступа : учебное пособие для вузов / В. М. Стасышин, Т. Л. Стасышина. — 2е изд., испр. и доп. — М.: Издательство Юрайт, 2023. — 164 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/514252 (дата обращения: 04.02.2023).
* MYSQL.COM — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.mysql.com/, свободный. (дата обращения: 01.02.2023).
* Oracle Database Software — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.oracle.com/database/technologies/oracledatabasesoftwaredownloads.html, свободный. (дата обращения: 01.02.2023).
* PostgreSQL: The World's Most Advanced Open Source Relational Database — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.postgresql.org/, свободный. (дата обращения: 01.02.2023).
* SQL Server 2019. — [Электронный ресурс] / Электрон. текстовые дан. 2022. — Режим доступа: https://www.microsoft.com/ruRU/download/details.aspx?id=101064, свободный. (дата обращения: 01.02.2023).
* Васильев, А. Н. Программирование на Python в примерах и задачах / А. Н. Васильев. — 1е изд. — Москва: Эксмо, 2022. — 616 с.
* Мэтиз, Э. Изучаем Python: программирование игр, визуализация данных, вебприложения / Э. Мэтиз. — 3е изд. — Питер: Издательский дом «Питер», 2022. — 560 с.
* Любанович, Б. Простой Python. Современный стиль программирования / Б. Любанович. — 2е изд. — Питер: Издательский дом «Питер», 2025. — 592 с.
* Иванов, И. И. Основы программирования на Java: учебное пособие для вузов / И. И. Иванов. — 2е изд., перераб. и доп. — М.: Издательство Юрайт, 2022. — 320 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/512345 (дата обращения: 05.02.2023).
* Петрова, С. К. Вебразработка: современные технологии и инструменты / С. К. Петрова. — М.: Издательство ДиалектикаВильямс, 2021. — 480 с. — (Профессиональное образование). — Текст : электронный // ЭБС МГТУ им. Баумана [сайт]. — URL: https://elearning.bmstu.ru/iu6/pluginfile.php/18674/mod\_resource/content/1/Вебразработка\_Современные\_технологии\_и\_инструменты.pdf (дата обращения: 05.02.2023).
* Сидоров, А. А. Машинное обучение: практическое руководство / А. А. Сидоров. — СПб.: Питер, 2023. — 400 с. — (Библиотека программиста). — Текст : электронный // ЭБС Лань [сайт]. — URL: https://e.lanbook.com/book/123456 (дата обращения: 05.02.2023).
* Козлов, Д. В. Основы кибербезопасности: учебник для вузов / Д. В. Козлов. — М.: Издательство Юрайт, 2022. — 280 с. — (Высшее образование). — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://urait.ru/bcode/511234 (дата обращения: 05.02.2023).