

МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ  
Кировское областное государственное профессиональное образовательное  
бюджетное учреждение  
"Слободской колледж педагогики и социальных отношений"

**КУРСОВОЙ ПРОЕКТ**

по профессиональному модулю ПМ.01. "Разработка программных модулей"

на тему:

**«Разработка программного модуля для учёта программного обеспечения  
в организации»**

Труфакин Сергей  
Васильевич

Специальность 09.02.07 -

Информационные  
системы и  
программирование

Курс 21П-1

Форма обучения: очная  
Руководитель:

Калинин Арсений  
Олегович

Дата защиты курсового проекта:

Оценка за защиту курсового проекта:

Председатель ПЦК:

Слободской

2024

**ОГЛАВЛЕНИЕ**

1. Введение -----	3
2. Анализ предметной области -----	5
3. Разработка технического -----	7
4. Описание алгоритмов и функционирования -----	10
5. Тестирование программного модуля -----	15
6. Руководство пользователя -----	17
7. Заключение-----	21
8. Список литературы-----	22
9. Приложение-----	23

## ВВЕДЕНИЕ

Учет программного обеспечения (ПО) в организации представляет собой важный элемент управления информационными ресурсами, который включает в себя контроль за использованием, лицензированием и обновлением программных продуктов. В условиях быстрого развития технологий и увеличения числа программных решений, задача эффективного учета ПО становится особенно актуальной. Неправильное управление лицензиями может привести к юридическим последствиям и финансовым потерям, поэтому необходимо разработать систему, которая позволит автоматизировать процессы учета и мониторинга программного обеспечения.

Современные организации используют множество программных решений для оптимизации своих бизнес-процессов. Учет ПО не только помогает избежать штрафов за нарушение лицензионных соглашений, но и способствует более эффективному управлению ресурсами. В связи с этим, создание системы для учета и управления лицензиями на ПО становится насущной необходимостью.

**Цель курсового проекта** – создание программного обеспечения для учета и управления лицензиями на ПО в организации.

### **Задачи исследования:**

- Описать предметную область учета ПО.
- Разработать техническое задание на создание программного продукта.
- Описать архитектуру системы.
- Описать алгоритмы и функционирование системы.
- Провести тестирование и опытную эксплуатацию.
- Разработать руководство пользователя.

### **Объект и предмет исследования**

**Объект исследования** – процесс учета программного обеспечения в организации.

**Предмет исследования** – разработка программной системы для учета и управления лицензиями на ПО.

Методы исследования включают системный анализ и функциональное моделирование. Информационную систему исследования составляют официальные нормативно-правовые источники, а также данные об использовании современных информационных систем.

Структура работы состоит из введения, трех глав, заключения, списка используемой литературы и приложений. Каждая глава будет подробно освещать поставленные задачи и предлагать решения для эффективного учета программного обеспечения в организации.

Таким образом, данное введение подчеркивает важность учета программного обеспечения в современных организациях и определяет основные направления исследования, которые будут рассмотрены в ходе выполнения курсового проекта.

## АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Учет программного обеспечения (ПО) – это процесс систематизации и контроля использования программных продуктов в организации с целью обеспечения их эффективного функционирования, соблюдения лицензионных соглашений и защиты интеллектуальной собственности. Учет ПО включает в себя регистрацию, классификацию, мониторинг и анализ программных ресурсов, что позволяет оптимизировать затраты и минимизировать риски, связанные с использованием нелегального или устаревшего ПО.

### **Механизмы учета программного обеспечения и их классификация**

Существует несколько методов учета программного обеспечения в организациях. Классификация механизмов учета может осуществляться по нескольким критериям:

#### **1. По типу ПО**

- **Коммерческое ПО:** требует приобретения лицензии. Это ПО часто включает в себя техническую поддержку и обновления, что делает его более надежным, но и более затратным.
- **Бесплатное ПО:** доступно для использования без оплаты, но может иметь ограничения, такие как отсутствие технической поддержки или функциональные ограничения.
- **Открытое ПО:** предоставляет пользователям возможность изменять и распространять программный код. Это решение часто используется для снижения затрат, однако требует наличия специалистов, способных адаптировать и поддерживать данное ПО.

#### **2. По способу учета**

- **Автоматизированные системы учета:** внедрение специализированных программных решений для автоматизации учета ПО. Эти системы позволяют вести реестр лицензий, отслеживать установки и обновления, а также генерировать отчеты для анализа. Преимущества таких

систем включают скорость обработки данных и уменьшение количества ошибок.

- Ручной учет: использование таблиц и документов для ведения учета программного обеспечения. Этот метод менее эффективен и подвержен ошибкам, но может быть полезен для небольших организаций. Важно отметить, что ручной учет требует регулярного обновления информации и может быть трудоемким.

### 3. По надежности

- Проверенные системы учета: решения, которые зарекомендовали себя на рынке и имеют положительные отзывы. Эти системы часто предлагают широкий спектр функций и интеграцию с другими корпоративными системами.

- Новые решения: системы, которые только начинают использоваться и могут требовать дополнительного тестирования. Несмотря на потенциальные преимущества, такие решения могут быть менее стабильными.

### 4. По рентабельности

- Доступные системы учета: решения с низкой стоимостью, подходящие для малых и средних организаций. Они могут предоставить базовые функции, необходимые для учета ПО.

- Дорогие системы учета: высокофункциональные решения, которые могут потребовать значительных инвестиций, но предлагают расширенные возможности, такие как интеграция с другими системами управления активами и аналитика.

### Внедрение учета программного обеспечения

Для эффективного учета ПО в организации важно создать систему, которая будет включать следующие элементы:

#### 1. Регистрация ПО

- Все используемое программное обеспечение должно быть зарегистрировано в реестре с указанием его типа, версии, даты приобретения

и срока действия лицензии. Это позволит избежать проблем с нарушением лицензионных соглашений.

## 2. Мониторинг использования

- Регулярный мониторинг использования ПО позволяет выявить несанкционированные установки и лицензии, а также определить необходимость обновлений или замены устаревшего ПО. Такой подход помогает поддерживать актуальность программного обеспечения и снизить риски, связанные с использованием устаревших версий.

## 3. Анализ затрат

- Оценка затрат на приобретение и поддержку ПО поможет оптимизировать бюджет и избежать лишних расходов. Важно проводить регулярный анализ, чтобы выявлять возможности для экономии и оптимизации расходов на ПО.

### Примеры систем учета программного обеспечения

Существуют различные программы и решения для учета программного обеспечения, которые помогают организациям управлять своими активами:

- **FlexNet Manager** – мощное решение для управления лицензиями и оптимизации использования ПО. Оно предоставляет возможность централизованного управления лицензиями и анализа использования ПО.
- **Lansweeper** – инструмент для автоматического обнаружения и учета ПО в сети. Он позволяет быстро идентифицировать установленное ПО и его версии, что упрощает процесс учета.
- **Asset Panda** – облачная платформа для учета активов, включая программное обеспечение, с возможностью создания отчетов и анализа данных. Она обеспечивает гибкость и доступность данных в любое время и из любого места.

## РАЗРАБОТКА ТЕХНИЧЕСКОГО ЗАДАНИЯ

Наименование программы – «Учет программного обеспечения в организации». Программа предназначена для систематизации и контроля использования программных продуктов в организации с целью оптимизации затрат и соблюдения лицензионных требований.

Разработка программы осуществляется на основании учебного плана и перечня тем, утвержденных на заседании предметно-цикловой комиссии информатики и программирования, в соответствии с ГОСТ [] - "Программное обеспечение. Общие требования к документации на программное обеспечение".

### Функциональное назначение программы

Функциональное назначение программы заключается в создании и поддержании реестра программного обеспечения, а также в обеспечении возможности выполнения следующих функций:

- Регистрация программного обеспечения с указанием лицензий и версий.
- Мониторинг использования ПО для выявления несанкционированных установок.

### Организационно-технические мероприятия

Надежное функционирование программы должно быть обеспечено выполнением заказчиком следующих организационно-технических мероприятий:

- Организация бесперебойного питания технических средств.
- Использование лицензионного программного обеспечения в соответствии с ГОСТ Р 50779.42-99 - "Программное обеспечение. Условия лицензирования".
- Обеспечение защиты от вредоносного ПО, наличие антивирусной программы.
- Соблюдение правил эксплуатации технических средств.



### Время восстановления и отказоустойчивость

Время восстановления после отказа, вызванного сбоем электропитания или другими внешними факторами, не должно превышать 5 минут при соблюдении условий эксплуатации. Время восстановления после фатального сбоя операционной системы должно соответствовать времени, необходимому для устранения неисправностей.

Отказы программы могут возникать из-за некорректных действий оператора. Для минимизации таких рисков необходимо ограничить пользователю административные привилегии.

### Климатические условия эксплуатации

Климатические условия эксплуатации должны соответствовать требованиям, предъявляемым к техническим средствам, согласно ГОСТ 15150-69 - "Условия хранения и эксплуатации".

### Технические требования

В состав технических средств должен входить IBM-совместимый персональный компьютер (ПЭВМ), включающий:

- Процессор с тактовой частотой не менее 1 ГГц.
- Оперативную память объемом не менее 512 Мб.
- Жесткий диск с не менее 500 Мб свободного места.
- Монитор с разрешением экрана не менее 1024\*768.
- Оптический привод.
- Компьютерную мышь и клавиатуру.

Исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки следует использовать Microsoft Visual Studio 2022. Системные программные средства должны быть представлены лицензионной локализованной версией операционной системы Windows 7/8/10/11.

Программное обеспечение поставляется на CD-диске. Упаковка программного изделия должна осуществляться в упаковочную тару предприятия-изготовителя компакт-диска. Требования к транспортировке и

хранению должны соответствовать условиям эксплуатации носителей, на которых находится программный продукт.

### Пользовательский интерфейс

Программа должна обеспечивать взаимодействие с пользователем через графический пользовательский интерфейс.

### Документация

Предварительный состав программной документации включает:

- Техническое задание.
- Руководство оператора.

### Этапы разработки

Разработка должна быть проведена в следующие стадии и этапы:

1. Анализ требований:
  - На этой стадии формулируются цели и задачи проекта, создаётся основа для дальнейшего проектирования.
2. Проектирование:
  - Разработка программной документации, включая техническое задание.
  - Определение и уточнение требований к техническим средствам и программе.
  - Разработка алгоритма программы.
  - Кодирование, где алгоритмы реализуются в среде программирования.
  - Тестирование и отладка, включающие проверку работоспособности программы и исправление ошибок.

Приемо-сдаточные испытания должны проводиться с использованием технических средств. Приемка программы включает проверку её работоспособности с вводом реальных или демонстрационных данных. В случае успешного прохождения испытаний программа вводится в эксплуатацию. При наличии критических ошибок программа отправляется на доработку.

В данном документе описано техническое задание, содержащее информацию о программном продукте, его функциональных возможностях и требованиях к его разработке и эксплуатации.

## ОПИСАНИЕ АЛГОРИТМОВ И ФУНКЦИОНИРОВАНИЯ СИСТЕМЫ

Программа «Учет программного обеспечения» включает в себя несколько ключевых модулей:

### 4.1. Примеры схемы

Схема выполнения программы приведен схематично на рисунке 1 в нем отражается вся функциональная составляющая программы и ее основные функции в упрощенном виде (Рис. 1).

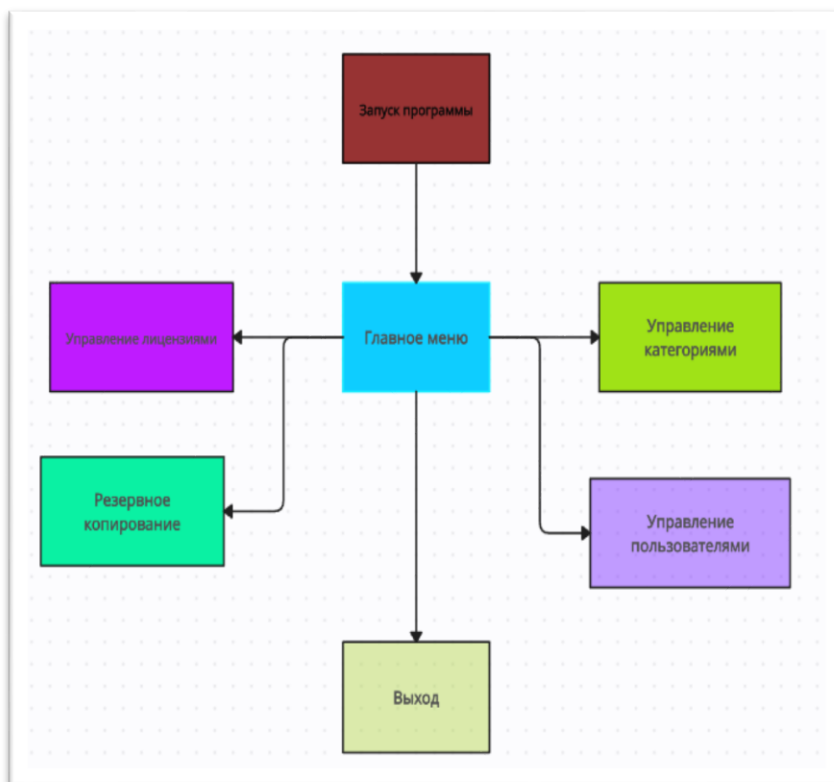


Рисунок 1 - схема выполнения программы

При запуске программы происходит отображение главной формы (Рис 2), на которой пользователю предлагается войти в систему

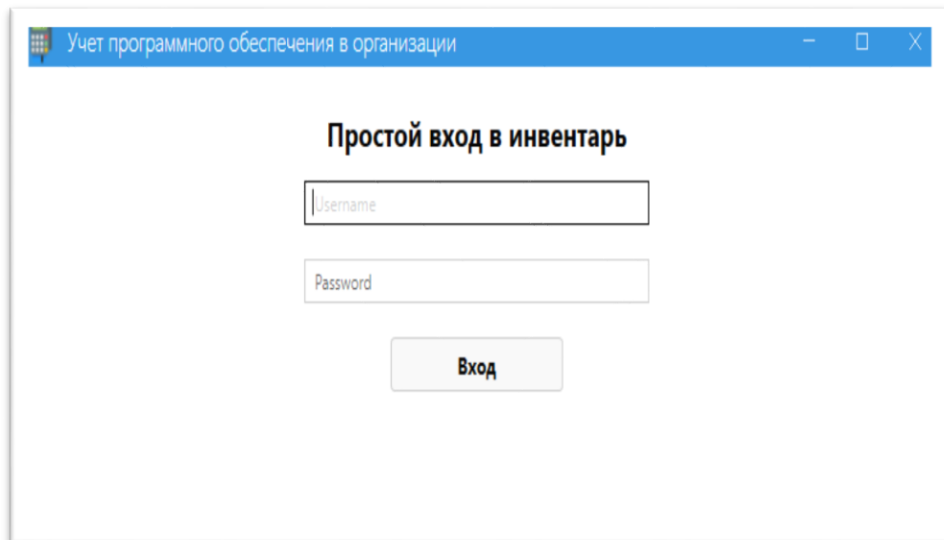


Рисунок 2 – Окно авторизации

После заполнения полей «логин и пароль» при нажатии на кнопку Вход откроется главное (Рис. 3).

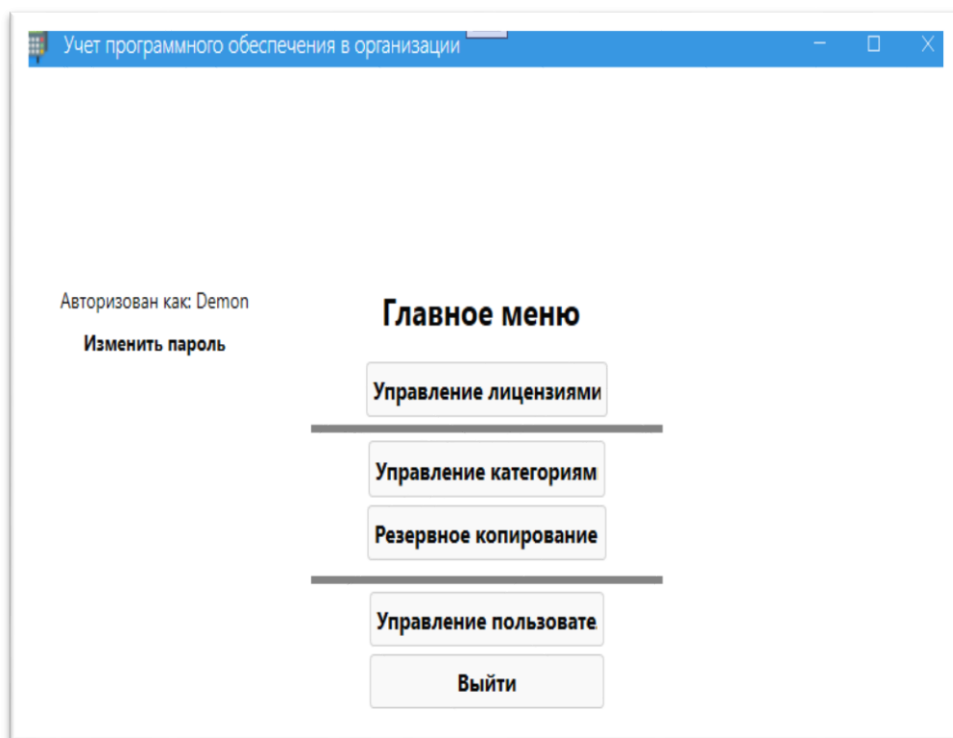


Рисунок 3 – Главное меню

Кнопка «Управление лицензиями» переносит на новое окно где есть приложения у которые есть лицензионный ключ (Рис. 4).

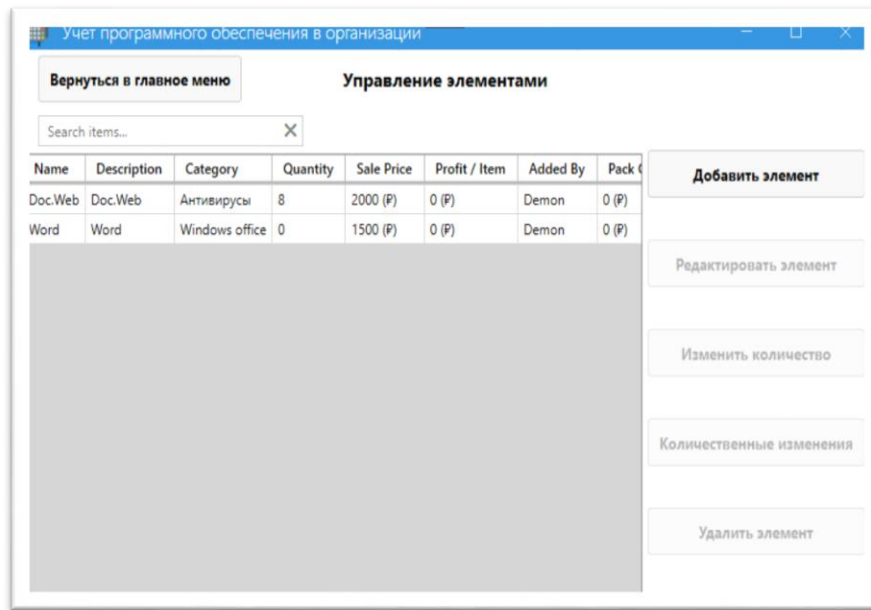


Рисунок 4 – Управление лицензиями

Так же реализованы кнопки как «Добавить элемент», «Редактировать элемент», «Изменить количества элемента», «Удаление элемента» (Рис. 5).

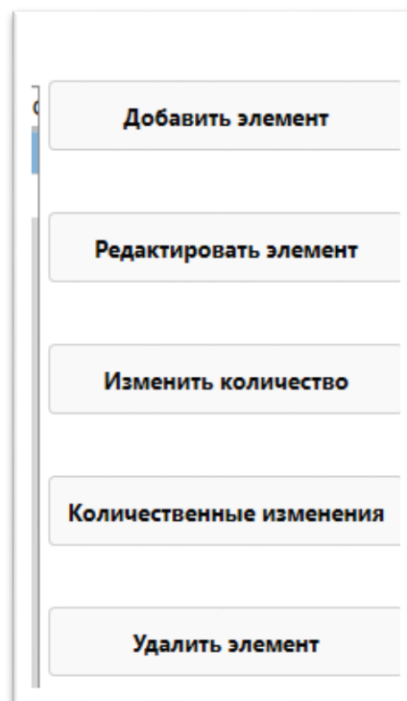


Рисунок 5 – Управление элементами через кнопки

Далее идет кнопка «Резервное копирование» - оно сохраняет файл с SIDB file (Рис. 6).

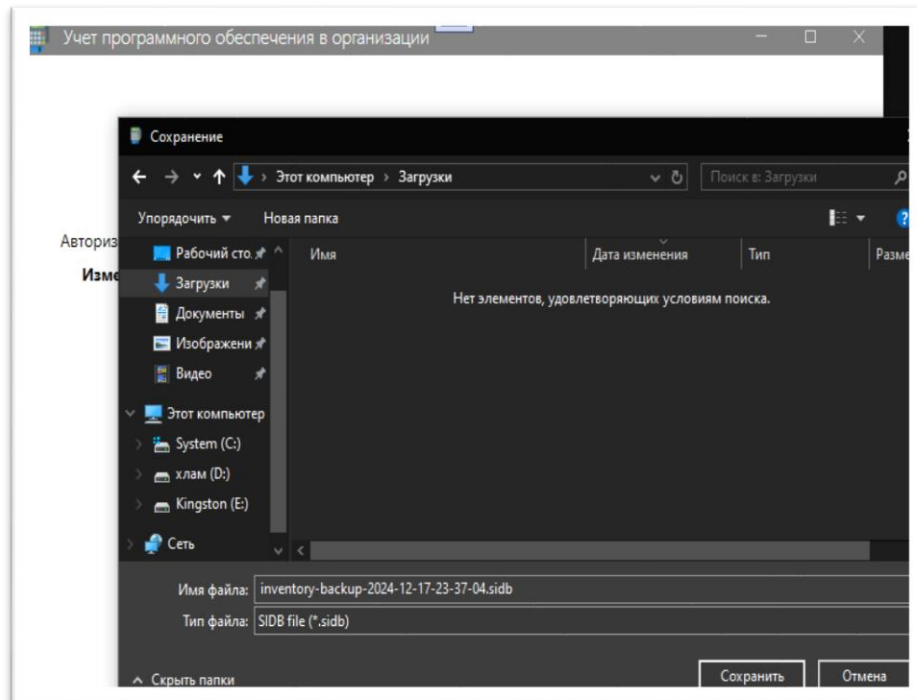


Рисунок 6 – Резервное копирование

Следующая кнопка это – «Управление пользователями» в это разделе можно добавить нового пользователя и настроить его доступ к программе(Рис. 7).

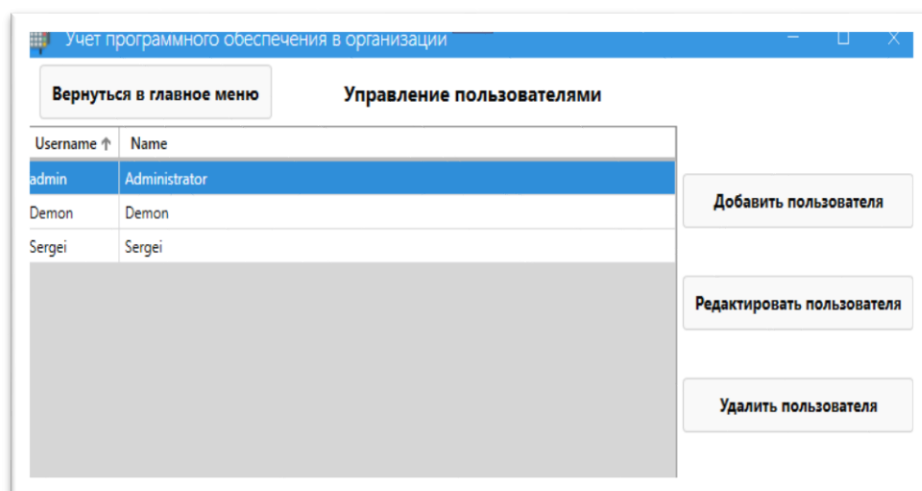


Рисунок 7 – Управление пользователями

Вывод: таким образом, программа «Учет программного обеспечения» предлагает пользователям интуитивно понятный интерфейс и набор функций,

позволяющих эффективно управлять лицензиями, пользователями и данными. Система обеспечивает надежность, безопасность и простоту в использовании, что делает ее незаменимым инструментом для организаций, стремящихся оптимизировать учет программного обеспечения.



## ТЕСТИРОВАНИЕ ПРОГРАММНОГО МОДУЛЯ

Для проведения тестирования программы мною было произведено базовое тестирование во время разработки программы. При тестировании был выявлен ряд ошибок, которые возникли в ходе выполнения программы.

### 5.1. Методы тестирования

- Попробовать войти в систему не заполняя одно из значений

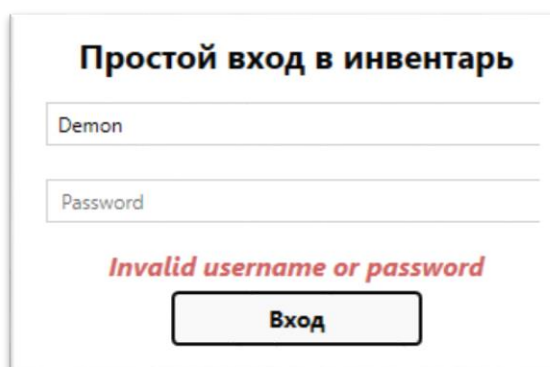


Рисунок 1 – Ошибка входа в систему

Ожидаемый результат: Ошибка о некорректных данных.

Полученный результат: Ошибка о некорректных данных (Рис. 1).

Решение проблемы: при входе в систему использовать правильный логин и пароль.

- Попробовать добавить пользователя без заполнения полей «логин» «пароль»

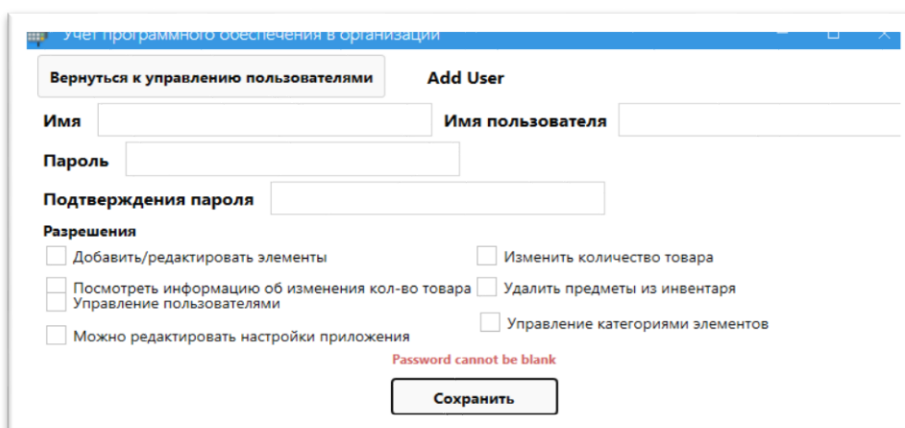


Рисунок 2 – Добавление пользователя

Ожидаемый результат: Ошибка о некорректных данных.

Полученный результат: Ошибка о некорректных данных (Рис. 2).

Решение проблемы: заполнить все данные

- Попробовать сделать резервное копировании.

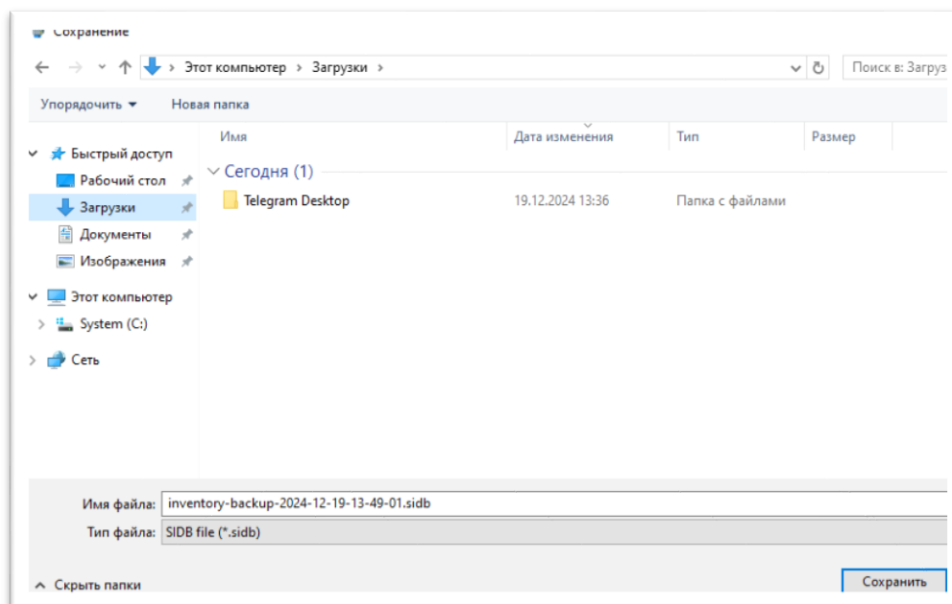


Рисунок 3 – резервное копирование

Ожидаемый результат: Успешно.

Полученный результат: Успешно (Рис. 3).

- Попробовать добавить пользователя и дать ему имя

Менеджер

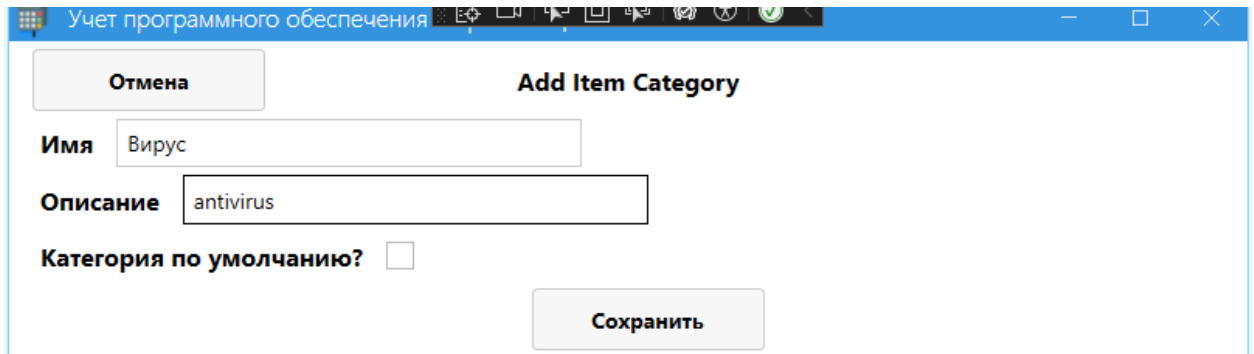
<a href="#">Вернуться к управлению пользователями</a>		<b>Add User</b>	
<b>Имя</b>	<input type="text" value="Менеджер"/>	<b>Имя пользователя</b>	<input type="text" value="Менеджер"/>
<b>Пароль</b>	<input type="password" value="•"/>		
<b>Подтверждения пароля</b>	<input type="password" value="•"/>		

Рисунок – 4 Добавление пользователя

Ожидаемый результат: Успешно.

Полученный результат: Успешно (Рис. 4).

- Попробовать добавить категорию.



Учет программного обеспечения

Отмена

Add Item Category

Имя:

Описание:

Категория по умолчанию? ☐

Сохранить

Рисунок 5- Добавление категории

Ожидаемый результат: Успешно.

Полученный результат: Успешно (Рис. 4).

## 5.2. Результаты тестирования

Результаты тестирования показали, что программа работает корректно. В процессе тестирования было выявлено 15 ошибок, из которых 12 были исправлены до окончательной версии программы. Время, затраченное на исправление ошибок, составило в среднем 2 часа на каждую ошибку.

## РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

### Введение

Данное руководство предназначено для специалистов по информационным технологиям и менеджеров по инвентаризации в организации. Оно описывает процесс учета программного обеспечения (ПО), включая его регистрацию, лицензирование, обновление и аудит. Эффективный учет ПО позволяет оптимизировать расходы, обеспечить соблюдение лицензионных соглашений и повысить безопасность информационных систем.

### Установка

#### Системные требования:

- Операционная система Windows (минимальная версия Windows 7).
- Процессор с тактовой частотой не менее 1 ГГц.
- Оперативную память объемом не менее 512 Мб.
- Жесткий диск с не менее 500 Мб свободного места.
- Монитор с разрешением экрана не менее 1024\*768.
- Оптический привод.
- Компьютерную мышь и клавиатуру.

#### Процесс установки:

1. Скачайте инсталляционный файл с usb-носителя.
2. Запустите инсталлятор и следуйте инструкциям на экране.
3. После завершения установки программа автоматически откроется.

#### Интерфейс программы:

При запуске программы мы увидим окно авторизации (Рис. 1).

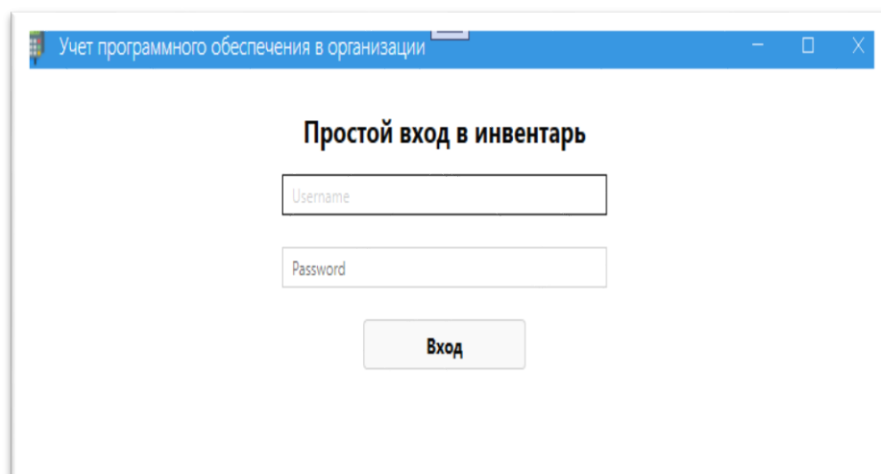


Рисунок 1 – Вход

После авторизации мы увидим главное меню в котором мы можем управлять лицензиями, управлять категориями, делать резервное копирование, управлять пользователями (Рис. 2).

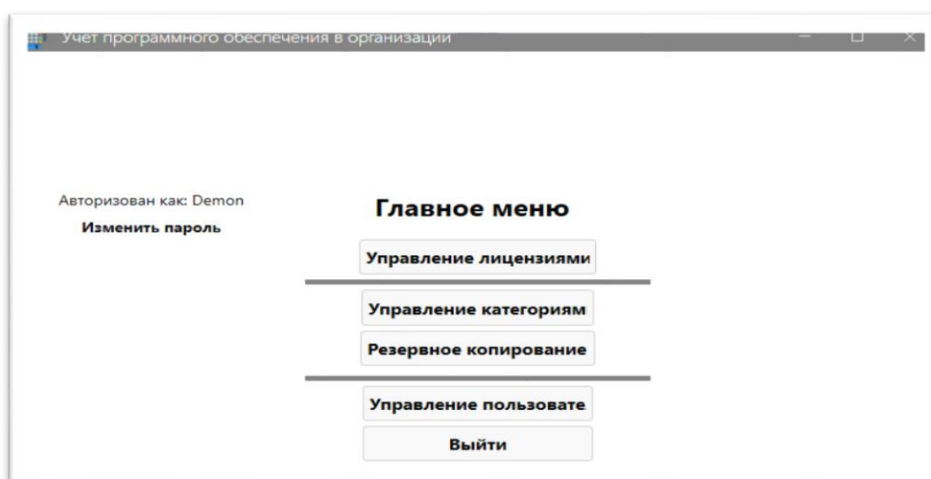


Рисунок 2 – Главное меню

При нажатии на кнопку «Управление лицензиями» откроется новое окно со всеми программными, на которое есть лицензия, там можно как добавить программу, которую купили, так и изменять ключ лицензии, который вы купили. Так же удалять программу, которой больше не пользуетесь или не купили лицензионный ключ (Рис. 3).

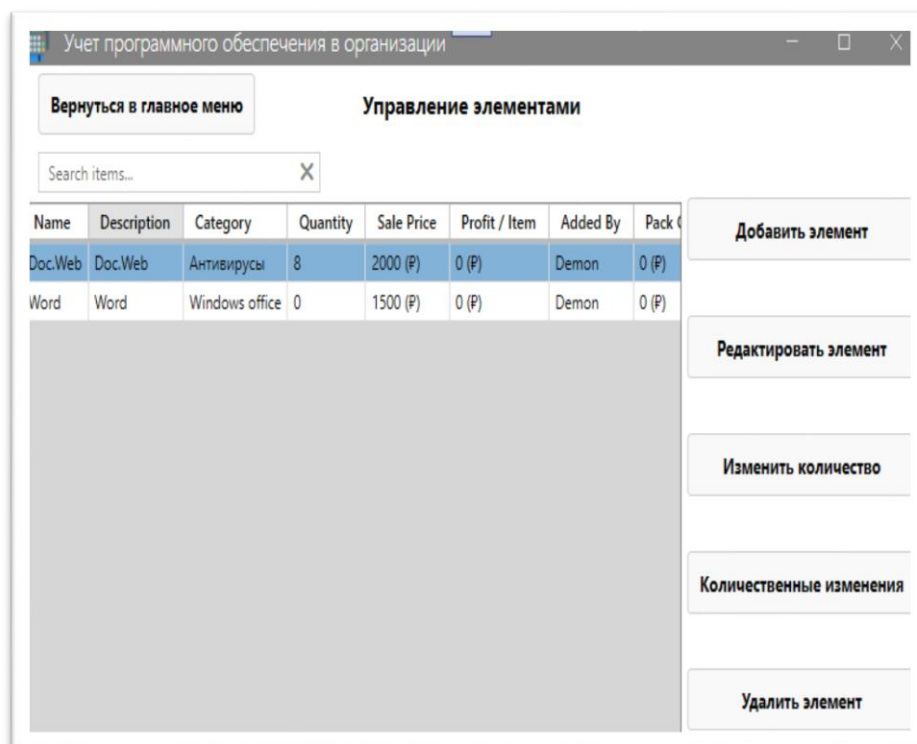


Рисунок 3 – Управление элементами

При нажатии на кнопку «Управление категориями» - то откроется окно где можно вписать название программы которую вы купили. Так же можно категорию изменять и удалять (Рис. 4).

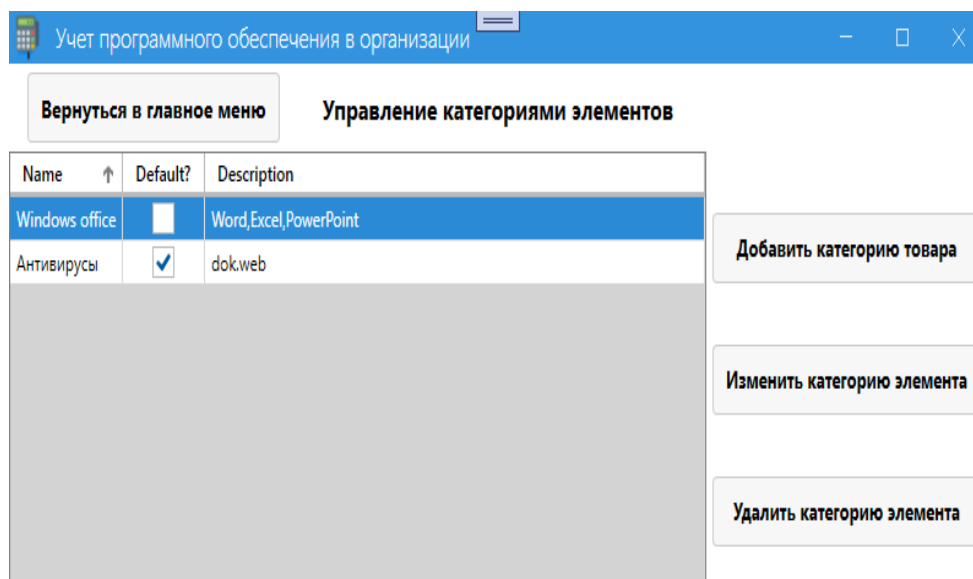


Рисунок 4 – Управление категориями

Далее идет кнопка «Резервное копирование» - оно сохраняет файл с SIDB file (Рис.5).

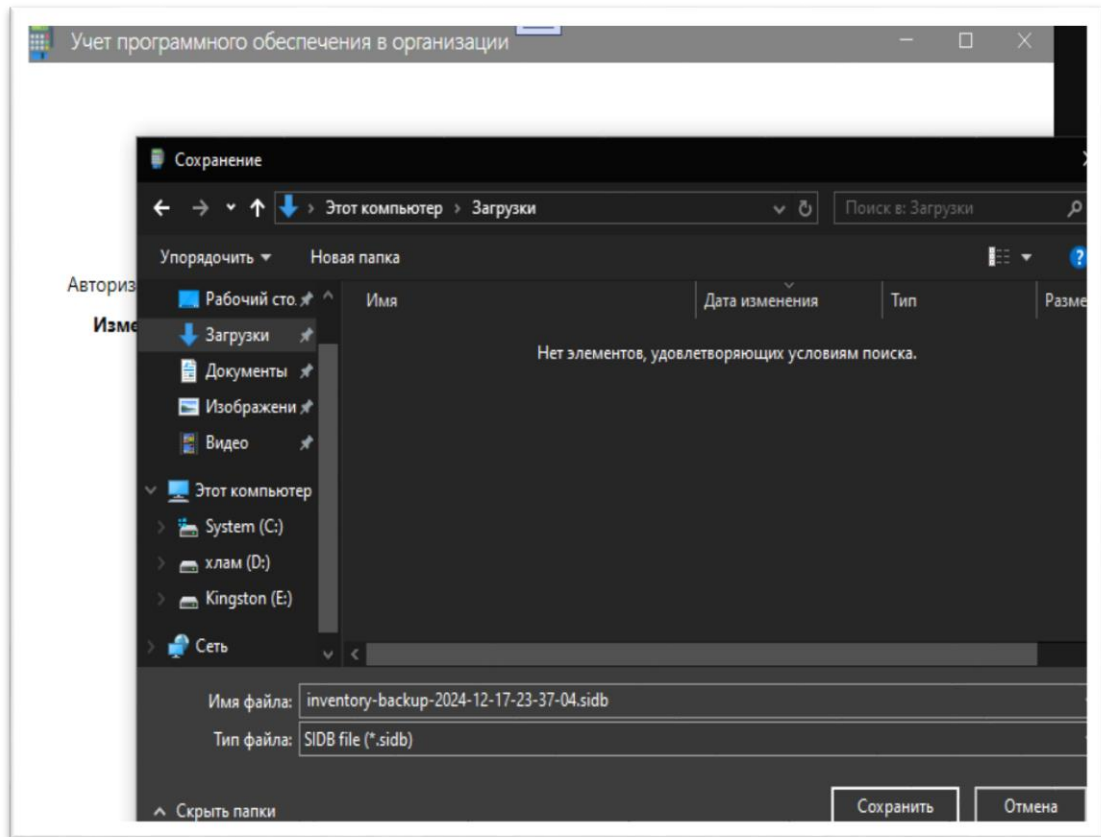


Рисунок 5 - Резервное копирование

Следующая кнопка это – «Управление пользователями» в это разделе можно добавить нового пользователя и настроить его доступ к программе (Рис. 6).

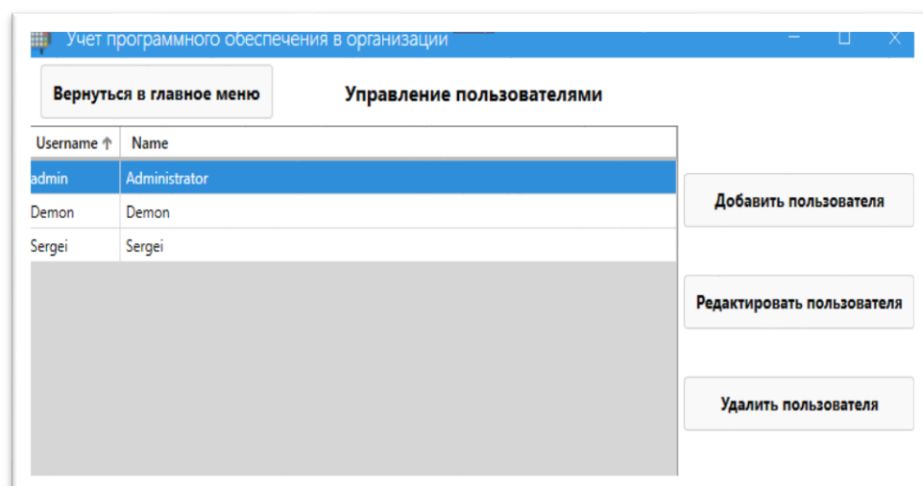


Рисунок 6 – Управление пользователями

## Заключение

В ходе выполнения курсового проекта нами была разработана программа для учета программного обеспечения в организации. В процессе работы были изучены основные требования к учету программного обеспечения, разработаны функциональные требования к системе, а также реализован программный модуль, позволяющий автоматизировать данный процесс.

Разработанная программа решает ряд важных задач:

- Автоматизирует учет установленных программных продуктов и лицензий.
- Обеспечивает контроль за сроками действия лицензий и предупреждает о необходимости их продления.
- Позволяет анализировать использование программного обеспечения и выявлять несанкционированные установки.
- Предоставляет удобный интерфейс для управления программными активами организации.

В процессе разработки был проведен анализ предметной области, составлено техническое задание, спроектирована архитектура системы и реализованы основные функциональные модули.

Программа может быть успешно внедрена в организации для ведения учета программного обеспечения, что позволит оптимизировать процессы лицензирования, сократить расходы и минимизировать риски, связанные с использованием нелегального или устаревшего ПО.

Таким образом, курсовой проект позволил приобрести ценные знания и практические навыки в области разработки программного обеспечения, начиная от проектирования системы и заканчивая ее тестированием и внедрением. Полученные результаты могут быть использованы для дальнейшего развития проекта, его интеграции с другими информационными системами и повышения эффективности управления программными ресурсами организации.



## СПИСОК ЛИТЕРАТУРЫ

1. ГОСТ 15150-69. Условия хранения и эксплуатации. [Электронный ресурс] URL: <http://docs.cntd.ru/document/1200007726> (дата обращения: 19.12.2024).
2. ГОСТ 28147-89. Системы защиты информации. Алгоритмы криптографической защиты. [Электронный ресурс] URL: <http://docs.cntd.ru/document/1200008236> (дата обращения: 19.12.2024).
3. ГОСТ Р 51141-98. Программное обеспечение. Общие требования к документации на программное обеспечение. [Электронный ресурс] URL: <http://docs.cntd.ru/document/1200009374> (дата обращения: 19.12.2024).
4. ГОСТ Р 50779.42-99. Программное обеспечение. Условия лицензирования. [Электронный ресурс] URL: <http://docs.cntd.ru/document/1200020300> (дата обращения: 19.12.2024).
5. ГОСТ Р ИСО/МЭК 25010-2011. Системы и программное обеспечение. Модели качества. [Электронный ресурс] URL: <http://docs.cntd.ru/document/1200091923> (дата обращения: 19.12.2024).
6. ГОСТ Р 56303-2015. Информационные технологии. Системы управления активами. [Электронный ресурс] URL: <http://docs.cntd.ru/document/1200120486> (дата обращения: 19.12.2024).
7. Баранов В. И. Учет программного обеспечения: современные подходы. – М.: Инфра-М, 2019. – 230 с. [Электронный ресурс] URL: <https://www.labirint.ru/books/> (дата обращения: 19.12.2024).
8. Григорьев С. Н. Информационные технологии в учете и аудите. – СПб.: Питер, 2020. – 240 с. [Электронный ресурс] URL: <https://www.piter.com/> (дата обращения: 19.12.2024).
9. Иванов С. В. Учет и аудит программного обеспечения. – Новосибирск: Сибирское университетское издательство, 2018. – 200 с.

10. Костюков А. В. Управление программным обеспечением в организации. – М.: Инфра-М, 2021. – 320 с. [Электронный ресурс] URL: <https://www.labirint.ru/> (дата обращения: 19.12.2024).
11. Кузнецов Д. В. Автоматизация процессов учета в организации. – Казань: Казанский университет, 2019. – 220 с.
12. Лебедев А. Г. Инвентаризация программного обеспечения. – М.: Альпина Паблишер, 2021. – 250 с. [Электронный ресурс] URL: <https://www.alpinabook.ru/> (дата обращения: 19.12.2024).
13. Михайлов А. В. Управление программными активами: методология и практика. – М.: Юрайт, 2021. – 320 с. [Электронный ресурс] URL: <https://urait.ru/> (дата обращения: 19.12.2024).
14. Петрова И. С. Автоматизация учета программного обеспечения. – СПб.: Питер, 2020. – 240 с. [Электронный ресурс] URL: <https://www.piter.com/> (дата обращения: 19.12.2024).
15. Романов И. П. Аудит информационных систем. – М.: РГГУ, 2020. – 270 с. [Электронный ресурс] URL: <https://www.rsuh.ru/> (дата обращения: 19.12.2024).
16. Сидоров В. Н. Лицензирование программного обеспечения: практика и рекомендации. – Екатеринбург: УралГТУ, 2019. – 180 с. [Электронный ресурс] URL: <https://urfu.ru/> (дата обращения: 19.12.2024).
17. Смирнова Е. А. Управление ИТ-активами: современные подходы. – М.: КНОРУС, 2021. – 280 с. [Электронный ресурс] URL: <https://www.knorus.ru/> (дата обращения: 19.12.2024).
18. Тихонов М. А. Лицензирование программного обеспечения: теория и практика. – СПб.: БХВ-Петербург, 2020. – 150 с. [Электронный ресурс] URL: <https://bhv.ru/> (дата обращения: 19.12.2024).
19. Федоров И. Ю. Системы управления активами: от теории к практике. – М.: Дело, 2019. – 300 с. [Электронный ресурс] URL: <https://www.delo.ru/> (дата обращения: 19.12.2024).

20. Чернов А. С. Эффективное управление ИТ-ресурсами. – Екатеринбург: УралГТУ, 2021. – 190 с. [Электронный ресурс] URL: <https://urfu.ru/> (дата обращения: 19.12.2024).

21. Шевченко А. П. Информационные технологии в управлении активами. – М.: Эксмо, 2020. – 350 с. [Электронный ресурс] URL: <https://eksmo.ru/> (дата обращения: 19.12.2024).

# Приложение

```

using SimpleInventory.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace SimpleInventory.Views
{
    /// <summary>
    /// Interaction logic for ChangePassword.xaml
    /// </summary>
    public partial class ChangePassword : UserControl
    {
        public ChangePassword()
        {
            InitializeComponent();
        }
        private void PasswordInput_PasswordChanged(object sender, RoutedEventArgs e)
        {
            var dataContext = DataContext as ChangePasswordViewModel;
            if (dataContext != null)
            {
                dataContext.Password = PasswordInput.SecurePassword;
            }
        }

        private void ConfirmPasswordInput_PasswordChanged(object sender, RoutedEventArgs e)
        {
            var dataContext = DataContext as ChangePasswordViewModel;

```

```

        if (dataContext != null)
        {
            dataContext.ConfirmPassword = ConfirmPasswordInput.SecurePassword;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;

namespace SimpleInventory.Views
{
    /// <summary>
    /// Interaction logic for AddItem.xaml
    /// </summary>
    public partial class CreateOrEditItem : UserControl
    {
        public CreateOrEditItem()
        {
            InitializeComponent();
            Loaded += CreateOrEditItem_Loaded;
        }

        private void CreateOrEditItem_Loaded(object sender, RoutedEventArgs e)
        {
            Keyboard.Focus(NameTextBox);
            Loaded -= CreateOrEditItem_Loaded;
        }
    }
}

```

```

using PdfSharp.Drawing;
using PdfSharp.Pdf;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;
using System.Drawing.Imaging;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
using SimpleInventory.Models;

namespace SimpleInventory.Helpers
{
    // class for generating a PDF of X number of barcodes via PDFSharp and BarcodeLib
    class BarcodePDFGenerator
    {
        public BarcodePDFGenerator()
        {
            IsDryRun = false;
            BarcodeType = BarcodeLib.TYPE.CODE128;
            PageSize = PdfSharp.PageSize.A4;
            NumberOfPages = 1;
        }

        private BitmapSource ConvertImageToBitmapImage(Image img)
        {
            using (var memory = new MemoryStream())
            {
                img.Save(memory, ImageFormat.Jpeg);
                memory.Position = 0;

                var bitmapImage = new BitmapImage();
                bitmapImage.BeginInit();
                bitmapImage.StreamSource = memory;
                bitmapImage.CacheOption = BitmapCacheOption.OnLoad;
                bitmapImage.EndInit();

                return bitmapImage;
            }
        }
    }
}

```

```

    }

    private Bitmap ResizeImage(Image image, int width, int height, int resolution = 96)
    {
        var destRect = new Rectangle(0, 0, width, height);
        var destImage = new Bitmap(width, height);
        using (var graphics = Graphics.FromImage(destImage))
        {
            graphics.CompositingMode =
System.Drawing.Drawing2D.CompositingMode.SourceCopy;
            graphics.CompositingQuality =
System.Drawing.Drawing2D.CompositingQuality.HighQuality;
            graphics.InterpolationMode =
System.Drawing.Drawing2D.InterpolationMode.HighQualityBicubic;
            graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.HighQuality;
            graphics.PixelOffsetMode =
System.Drawing.Drawing2D.PixelOffsetMode.HighQuality;
            using (var wrapMode = new ImageAttributes())
            {
                wrapMode.SetWrapMode(System.Drawing.Drawing2D.WrapMode.TileFlipXY);
                graphics.DrawImage(image, destRect, 0, 0, image.Width, image.Height,
GraphicsUnit.Pixel, wrapMode);
            }
        }
        destImage.SetResolution(resolution, resolution);
        return destImage;
    }

    /// <summary>
    /// Defaults to false.
    /// If true, does not save anything to disk on barcode generate or update the database.
    /// Use for figuring out how many barcodes will be generated ahead of time.
    /// </summary>
    public bool IsDryRun { get; set; }

    /// <summary>
    /// Defaults to BarcodeLib.TYPE.CODE128
    /// </summary>
    public BarcodeLib.TYPE BarcodeType { get; set; }

    /// <summary>
    /// Defaults to PdfSharp.PageSize.A4

```



```

/// </summary>
public PdfSharp.PageSize PageSize { get; set; }

/// <summary>
/// Defaults to 1
/// </summary>
public int NumberOfPages { get; set; }

/// <summary>
/// Generates a barcode PDF and returns the number of barcodes generated
/// </summary>
/// <param name="outputPath"></param>
/// <param name="numberOfPages"></param>
/// <returns>The number of barcodes generated</returns>
public int GenerateBarcodes(string outputPath)
{
    if (NumberOfPages > 0)
    {
        PdfDocument document = new PdfDocument();
        document.Info.Title = "Inventory Barcodes";
        long barcodeToUse = GeneratedBarcode.GetLatestBarcodeNumber() + 1;
        var barcodesGenerated = new List<long>();
        for (int i = 0; i < NumberOfPages; i++)
        {
            PdfPage page = document.AddPage();
            page.Size = PageSize;

            XGraphics gfx = XGraphics.FromPdfPage(page);
            XFont font = new XFont("Verdana", 20, XFontStyle.Bold);
            XUnit yCoord = XUnit.FromInch(1); // pixels
            gfx.DrawString("Inventory Barcodes", font, XBrushes.Black,
                new XRect(0, yCoord, page.Width, page.Height), XStringFormats.TopCenter);

            yCoord += XUnit.FromInch(0.7);

            // Generate a barcode
            var barcodeCreator = new BarcodeLib.Barcode();
            barcodeCreator.ImageFormat = ImageFormat.Jpeg;
            barcodeCreator.IncludeLabel = false;
            //barcodeCreator.IncludeLabel = true;
            //barcodeCreator.LabelPosition = BarcodeLib.LabelPositions.BOTTOMCENTER;
            barcodeCreator.Alignment = BarcodeLib.AlignmentPositions.CENTER;

```

```

bool isPageFull = false;
XUnit imageHeight = XUnit.FromPoint(60);
while (!isPageFull)
{
    var isWidthFull = false;
    XUnit xCoord = XUnit.FromInch(1);
    while (!isWidthFull)
    {
        var image = barcodeCreator.Encode(BarcodeType, barcodeToUse.ToString());
        if (image != null)
        {
            // make sure images are a good size based on DPI
            // TODO: There has got to be a better way to make things fairly consistent across
computers

            // with different DPI. This is ridiculous. I love WPF most of the time with its
DPI

            // help, but in this case.....ugh. Images come out a little blurry this way
            // on computers with a non-192 DPI, but scanners will probably be OK.
            double ratioTo192 = (192 / image.VerticalResolution);
            int resizeHeight = (int)(image.Height / ratioTo192);
            int resizeWidth = (int)(image.Width / ratioTo192);
            image      =      ResizeImage(image,      resizeWidth,      resizeHeight,
(int)image.VerticalResolution);

            // ok, now we can draw.
            XImage      pdfImage      =
XImage.FromBitmapSource(ConvertImageToBitmapImage(image));
            gfx.DrawImage(pdfImage, xCoord, yCoord);
            // now draw label
            XFont barcodeFont = new XFont("Verdana", 16, XFontStyle.Bold);
            // + 2 on the y coordinate there just to give it a teensy bit of space
            gfx.DrawString(barcodeToUse.ToString(), barcodeFont, XBrushes.Black,
                new XRect(xCoord, yCoord + pdfImage.PointHeight + 2,
pdfImage.PointWidth, 150), XStringFormats.TopCenter);
            //
            xCoord += XUnit.FromPoint(pdfImage.PointWidth);
            imageHeight = XUnit.FromPoint(pdfImage.PointHeight);
            //var blah = XUnit.FromPoint(image.Width);
            XUnit spaceBetweenBarcodes = XUnit.FromInch(0.75);
            if (xCoord + XUnit.FromPoint(pdfImage.PointWidth) +
spaceBetweenBarcodes > page.Width - XUnit.FromInch(1))
            {

```

```

        isWidthFull = true;
    }
    barcodesGenerated.Add(barcodeToUse);
    barcodeToUse++;
    xCoord += spaceBetweenBarcodes;
}
else
{
    // failure case
    isWidthFull = true;
    isPageFull = true;
    break;
}
}
yCoord += imageHeight;
yCoord += XUnit.FromInch(0.7);
if (yCoord + imageHeight > page.Height - XUnit.FromInch(1))
{
    isPageFull = true;
}
}
}
if (!IsDryRun)
{
    // save the fact that we generated barcodes
    GeneratedBarcode.AddGeneratedCodes(barcodesGenerated, DateTime.Now, 1);
    // save the document and start the process for viewing the pdf
    document.Save(outputPath);
    Process.Start(outputPath);
}
return barcodesGenerated.Count;
}
return 0;
}
}
}

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

```

```

using System.Data.SQLite;
using System.Security.Cryptography;
using SimpleInventory.Models;

namespace SimpleInventory.Helpers
{
    class DatabaseHelper
    {
        private const string _directory = "data";
        private const string _fileName = "inventory.sldb";

        private string GetFilePath()
        {
            return _directory + "/" + _fileName;
        }

        public string GetDatabaseFilePath()
        {
            return GetFilePath();
        }

        private bool DoesDatabaseExist()
        {
            return File.Exists(GetFilePath());
        }

        private SQLiteConnection GetDatabaseConnectionWithoutMigrating()
        {
            var conn = new SQLiteConnection("data source=" + GetFilePath());
            conn.Open();
            return conn;
        }

        public SQLiteConnection GetDatabaseConnection()
        {
            if (!DoesDatabaseExist())
            {
                CreateDatabase();
            }
            var conn = GetDatabaseConnectionWithoutMigrating();
            using (var command = new SQLiteCommand(conn))
            {

```

```

        command.CommandText = "PRAGMA foreign_keys = 1";
        command.ExecuteNonQuery();
        PerformMigrationsAsNecessary(command);
    }
    return conn;
}

/// <summary>
/// Returns a command with an open db connection and foreign keys turned on
/// </summary>
/// <param name="conn"></param>
/// <returns></returns>
public SQLiteCommand GetSQLiteCommand(SQLiteConnection conn)
{
    return new SQLiteCommand(conn);
}

public bool ReadBool(SQLiteDataReader reader, string columnName)
{
    int ordinal = reader.GetOrdinal(columnName);
    return reader.IsDBNull(ordinal) ? false : reader.GetBoolean(ordinal);
}

public bool ReadBool(SQLiteDataReader reader, int columnNumber)
{
    return reader.IsDBNull(columnNumber) ? false : reader.GetBoolean(columnNumber);
}

public int ReadInt(SQLiteDataReader reader, string columnName)
{
    int ordinal = reader.GetOrdinal(columnName);
    return reader.IsDBNull(ordinal) ? 0 : reader.GetInt32(ordinal);
}

public int ReadInt(SQLiteDataReader reader, int columnNumber)
{
    return reader.IsDBNull(columnNumber) ? 0 : reader.GetInt32(columnNumber);
}

public long ReadLong(SQLiteDataReader reader, string columnName)
{
    int ordinal = reader.GetOrdinal(columnName);

```

```

        return reader.IsDBNull(ordinal) ? 0 : reader.GetInt64(ordinal);
    }

```

```

public long ReadLong(SQLiteDataReader reader, int columnNumber)
{
    return reader.IsDBNull(columnNumber) ? 0 : reader.GetInt64(columnNumber);
}

```

```

public string ReadString(SQLiteDataReader reader, string columnName)
{
    int ordinal = reader.GetOrdinal(columnName);
    return reader.IsDBNull(ordinal) ? "" : reader.GetString(ordinal);
}

```

```

public string ReadString(SQLiteDataReader reader, int columnNumber)
{
    return reader.IsDBNull(columnNumber) ? "" : reader.GetString(columnNumber);
}

```

```

public decimal ReadDecimal(SQLiteDataReader reader, string columnName)
{
    int ordinal = reader.GetOrdinal(columnName);
    return reader.IsDBNull(ordinal) ? 0m : reader.GetDecimal(ordinal);
}

```

```

public decimal ReadDecimal(SQLiteDataReader reader, int columnNumber)
{
    return reader.IsDBNull(columnNumber) ? 0m : reader.GetDecimal(columnNumber);
}

```

```

private void PerformMigrationsAsNecessary(SQLiteCommand command)
{
    // uncomment when you need some migrations
    command.CommandText = "PRAGMA user_version";
    using (var reader = command.ExecuteReader())
    {
        if (reader.Read())
        {
            var userVersion = reader.GetInt32(0); // initial version is 0
            reader.Close(); // have to close it now otherwise we can't execute commands
            switch (userVersion + 1)
            {

```

case 1:

```
// create QuantityAdjustments table
string createQuantityAdjustmentsTable = "CREATE TABLE QuantityAdjustments (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "AmountChanged TEXT," +
    "DateTimeChanged TEXT," +
    "InventoryItemID INTEGER REFERENCES InventoryItems(ID)," +
    "AdjustedByUserID INTEGER REFERENCES Users(ID))";
command.CommandText = createQuantityAdjustmentsTable;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 1;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 2; // weeee
```

case 2:

```
// add IsDrink column
string addIsDrinkColumn = "" +
    "ALTER TABLE InventoryItems " +
    "ADD COLUMN IsDrink INTEGER DEFAULT 0;";
command.CommandText = addIsDrinkColumn;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 2;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 3;
```

case 3:

```
// add ItemTypes table
command.CommandText = "PRAGMA foreign_keys = 0;";
command.ExecuteNonQuery();
command.CommandText = "BEGIN TRANSACTION;";
command.ExecuteNonQuery();
string addItemTypesTable = "" +
    "CREATE TABLE ItemTypes (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "Name TEXT," +
    "Description TEXT)";
command.CommandText = addItemTypesTable;
command.ExecuteNonQuery();
string addInitialItemTypes = "" +
```

```

"INSERT INTO ItemTypes (Name, Description) VALUES ('School supplies\',"
"Pencils, pens, etc.\")";

command.CommandText = addInitialItemTypes;
command.ExecuteNonQuery();
addInitialItemTypes = "" +
    "INSERT INTO ItemTypes (Name, Description) VALUES ('Drinks\',"
    "Water, milk, etc.\")";

command.CommandText = addInitialItemTypes;
command.ExecuteNonQuery();
addInitialItemTypes = "" +
    "INSERT INTO ItemTypes (Name, Description) VALUES ('Meal tickets\',"
    "Tickets for student meals\")";

command.CommandText = addInitialItemTypes;
command.ExecuteNonQuery();
// Ugh, to change IsDrink to ItemTypeID with a FK, we have to recreate
// the entire table. :( :(
// to do so, we create a new table, copy over the data, drop the old table,
// and rename the new table to the new table
string recreateInventoryItemTable = "CREATE TABLE New_InventoryItems (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "Name TEXT," +
    "Description TEXT," +
    "PicturePath TEXT," +
    "Cost TEXT," +
    "CostCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "ProfitPerItem TEXT," +
    "ProfitPerItemCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "Quantity INTEGER," +
    "BarcodeNumber TEXT," +
    "WasDeleted INTEGER DEFAULT 0," +
    "CreatedByUserID INTEGER REFERENCES Users(ID)," +
    "ItemTypeID INTEGER REFERENCES ItemTypes(ID))";
command.CommandText = recreateInventoryItemTable;
command.ExecuteNonQuery();
string moveInventoryData = "" +
    "INSERT INTO New_InventoryItems (Name, Description, PicturePath, " +
    "Cost, CostCurrencyID, ProfitPerItem, ProfitPerItemCurrencyID, " +
    "Quantity, BarcodeNumber, WasDeleted, CreatedByUserID, ItemTypeID) " +
    "SELECT Name, Description, PicturePath, " +
    "Cost, CostCurrencyID, ProfitPerItem, ProfitPerItemCurrencyID, " +
    "Quantity, BarcodeNumber, WasDeleted, CreatedByUserID, 1 " +
    "FROM InventoryItems " +

```



```

"ORDER BY ID;";
command.CommandText = moveInventoryData;
command.ExecuteNonQuery();
string removeOldTable = "" +
    "DROP TABLE InventoryItems;";
command.CommandText = removeOldTable;
command.ExecuteNonQuery();
string renameNewTable = "" +
    "ALTER TABLE New_InventoryItems RENAME TO InventoryItems;";
command.CommandText = renameNewTable;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 3;";
command.ExecuteNonQuery();
command.Parameters.Clear();
command.CommandText = "COMMIT TRANSACTION;";
command.ExecuteNonQuery();
command.CommandText = "PRAGMA foreign_keys = 1";
command.ExecuteNonQuery();
// cleanup
command.CommandText = "VACUUM;";
command.ExecuteNonQuery();
goto case 4;
case 4:
    // add IsDefault column to ItemTypes
    string addIsDefaultColumn = "" +
        "ALTER TABLE ItemTypes " +
        "ADD COLUMN IsDefault INTEGER DEFAULT 0;";
    command.CommandText = addIsDefaultColumn;
    command.ExecuteNonQuery();
    // set default item type to school supplies
    command.CommandText = " UPDATE ItemTypes SET IsDefault = 1 WHERE ID = 1";
    command.ExecuteNonQuery();
    // bump user_version
    command.CommandText = "PRAGMA user_version = 4;";
    command.ExecuteNonQuery();
    command.Parameters.Clear();
    goto case 5;
case 5:
    command.CommandText = "PRAGMA foreign_keys = 0";
    command.ExecuteNonQuery();
    // oh bother, AmountChanged is the wrong column type. Should be int, is text. -_-

```

```

// Gotta recreate THAT table too...
string recreateQuantityAdjustmentsTable = "CREATE TABLE
New_QuantityAdjustments (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "AmountChanged INTEGER," +
    "DateTimeChanged TEXT," +
    "InventoryItemID INTEGER REFERENCES InventoryItems(ID)," +
    "AdjustedByUserID INTEGER REFERENCES Users(ID));";
command.CommandText = recreateQuantityAdjustmentsTable;
command.ExecuteNonQuery();
string moveQuantityAdjustmentData = "" +
    "INSERT INTO New_QuantityAdjustments (AmountChanged,
DateTimeChanged, InventoryItemID, AdjustedByUserID) " +
    "SELECT AmountChanged, DateTimeChanged, InventoryItemID,
AdjustedByUserID " +
    "FROM QuantityAdjustments " +
    "ORDER BY ID;";
command.CommandText = moveQuantityAdjustmentData;
command.ExecuteNonQuery();
string removeOldQuantityAdjustmentTable = "" +
    "DROP TABLE QuantityAdjustments;";
command.CommandText = removeOldQuantityAdjustmentTable;
command.ExecuteNonQuery();
string renameNewQuantityAdjustmentsTable = "" +
    "ALTER TABLE New_QuantityAdjustments RENAME TO
QuantityAdjustments;";
command.CommandText = renameNewQuantityAdjustmentsTable;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 5;";
command.ExecuteNonQuery();
command.Parameters.Clear();
command.CommandText = "PRAGMA foreign_keys = 1";
command.ExecuteNonQuery();
goto case 6;
case 6:
    // add user permissions table
    string addUserPermissionsTable = "" +
        "CREATE TABLE UserPermissions (" +
        "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
        "CanAddEditItems INTEGER," +
        "CanAdjustItemQuantity INTEGER," +

```

```

"CanViewDetailedItemQuantityAdjustments INTEGER," +
"CanScanItems INTEGER," +
"CanGenerateBarcodes INTEGER," +
"CanViewReports INTEGER," +
"CanViewDetailedItemSoldInfo INTEGER," +
"CanSaveReportsToPDF INTEGER," +
"CanDeleteItemsFromInventory INTEGER," +
"CanManageItemCategories INTEGER," +
"UserID INTEGER REFERENCES Users(ID))";
command.CommandText = addUserPermissionsTable;
command.ExecuteNonQuery();
// add default user permission for default user
string addDefaultPermissions = "" +
"INSERT INTO UserPermissions (CanAddEditItems, CanAdjustItemQuantity, " +
"CanViewDetailedItemQuantityAdjustments, CanScanItems, CanGenerateBarcodes,
CanViewReports," +
"CanViewDetailedItemSoldInfo, CanSaveReportsToPDF,
CanDeleteItemsFromInventory, CanManageItemCategories," +
"UserID) " +
"VALUES (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)";
command.CommandText = addDefaultPermissions;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 6;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 7;
case 7:
// add CanManageUsers column to UserPermissions
string addCanManageUsersColumn = "" +
"ALTER TABLE UserPermissions " +
"ADD COLUMN CanManageUsers INTEGER DEFAULT 1;";
command.CommandText = addCanManageUsersColumn;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 7;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 8;
case 8:
// add WasDeleted column to Users
string addWasDeletedUsersColumn = "" +

```

```

"ALTER TABLE Users " +
"ADD COLUMN WasDeleted INTEGER DEFAULT 0;";
command.CommandText = addWasDeletedUsersColumn;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 8;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 9;
case 9:
// add CanDeleteItemsSold column to UserPermissions
string addCanDeleteItemsSold = "" +
"ALTER TABLE UserPermissions " +
"ADD COLUMN CanDeleteItemsSold INTEGER DEFAULT 1;";
command.CommandText = addCanDeleteItemsSold;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 9;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 10;
case 10:
// add Explanation column to QuantityAdjustments
string addExplanationColumn = "" +
"ALTER TABLE QuantityAdjustments " +
"ADD COLUMN Explanation TEXT DEFAULT ";";
command.CommandText = addExplanationColumn;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 10;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 11;
case 11:
// add CanManageUsers column to UserPermissions
string addCanViewManageInventoryQuantityColumn = "" +
"ALTER TABLE UserPermissions " +
"ADD COLUMN CanViewManageInventoryQuantity INTEGER DEFAULT
1;";

command.CommandText = addCanViewManageInventoryQuantityColumn;
command.ExecuteNonQuery();
// bump user_version

```

```

command.CommandText = "PRAGMA user_version = 11;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 12;
case 12:
    // add WasAdjustedForStockPurchase column to QuantityAdjustments
    string addWasAdjustedForStockPurchase = "" +
        "ALTER TABLE QuantityAdjustments " +
        "ADD COLUMN WasAdjustedForStockPurchase INTEGER DEFAULT 0;";
    command.CommandText = addWasAdjustedForStockPurchase;
    command.ExecuteNonQuery();
    // bump user_version
    command.CommandText = "PRAGMA user_version = 12;";
    command.ExecuteNonQuery();
    command.Parameters.Clear();
    goto case 13;
case 13:
    // add WasAdjustedForStockPurchase column to QuantityAdjustments
    string addPurchaseCostAndItemsPerPurchase = "" +
        "ALTER TABLE InventoryItems " +
        "ADD COLUMN ItemPurchaseCost TEXT DEFAULT '0'; " +
        "ALTER TABLE InventoryItems " +
        "ADD COLUMN ItemPurchaseCostCurrencyID INTEGER DEFAULT 0;" +
        "ALTER TABLE InventoryItems " +
        "ADD COLUMN ItemsPerPurchase INTEGER DEFAULT 0;";
    command.CommandText = addPurchaseCostAndItemsPerPurchase;
    command.ExecuteNonQuery();
    // bump user_version
    command.CommandText = "PRAGMA user_version = 13;";
    command.ExecuteNonQuery();
    command.Parameters.Clear();
    goto case 14;
case 14:
    // add CanEditAppSettings column to UserPermissions
    string addCanEditAppSettings = "" +
        "ALTER TABLE UserPermissions " +
        "ADD COLUMN CanEditAppSettings INTEGER DEFAULT 0;";
    command.CommandText = addCanEditAppSettings;
    command.ExecuteNonQuery();
    // bump user_version
    command.CommandText = "PRAGMA user_version = 14;";
    command.ExecuteNonQuery();

```

```

command.Parameters.Clear();
goto case 15;
case 15:
    // add CanManageCurrencies column to UserPermissions
    string addCanEditCurrencies = "" +
        "ALTER TABLE UserPermissions " +
        "ADD COLUMN CanManageCurrencies INTEGER DEFAULT 0;";
    command.CommandText = addCanEditCurrencies;
    command.ExecuteNonQuery();
    // bump user_version
    command.CommandText = "PRAGMA user_version = 15;";
    command.ExecuteNonQuery();
    command.Parameters.Clear();
    goto case 16;
case 16:
    string addPurchases = "" +
        "CREATE TABLE Purchases (" +
        "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
        "DateTimePurchased TEXT," +
        "TotalCost TEXT," +
        "Name TEXT," +
        "Phone TEXT," +
        "Email TEXT," +
        "UserID INTEGER REFERENCES Users(ID))";
    command.CommandText = addPurchases;
    command.ExecuteNonQuery();
    string addPurchasedItems = "" +
        "CREATE TABLE PurchasedItems (" +
        "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
        "Quantity INTEGER," +
        "Name TEXT," +
        "Type TEXT," +
        "Cost TEXT," +
        "CostCurrencySymbol TEXT," +
        "CostCurrencyConversionRate TEXT," +
        "Profit TEXT," +
        "ProfitCurrencySymbol TEXT," +
        "ProfitCurrencyConversionRate TEXT," +
        "PurchaseID INTEGER REFERENCES Purchases(ID))";
    command.CommandText = addPurchasedItems;
    command.ExecuteNonQuery();
    // bump user_version

```

```

command.CommandText = "PRAGMA user_version = 16;";
command.ExecuteNonQuery();
command.Parameters.Clear();
goto case 17;
case 17:
string addPurchaseColumns = "" +
    "ALTER TABLE Purchases " +
    "ADD COLUMN CostCurrencySymbol TEXT DEFAULT '$'; " +
    "ALTER TABLE Purchases " +
    "ADD COLUMN CostCurrencyConversionRate TEXT DEFAULT '1';";
command.CommandText = addPurchaseColumns;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 17;";
command.ExecuteNonQuery();
command.Parameters.Clear();
break;
case 18:
string addMorePurchaseColumns = "" +
    "ALTER TABLE PurchasedItems " +
    "ADD COLUMN InventoryItemID INTEGER REFERENCES InventoryItems(ID); " +
    "ALTER TABLE Purchases " +
    "ADD COLUMN ChangeCurrencySymbol TEXT DEFAULT '$'; " +
    "ALTER TABLE Purchases " +
    "ADD COLUMN ChangeCurrencyConversionRate TEXT DEFAULT '1';";
command.CommandText = addMorePurchaseColumns;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 18;";
command.ExecuteNonQuery();
command.Parameters.Clear();
break;
case 19:
string addPurchaseMethodColumn = "" +
    "ALTER TABLE Purchases " +
    "ADD COLUMN PurchaseMethod INTEGER DEFAULT 1; " +
    "ALTER TABLE ItemsSoldInfo " +
    "ADD COLUMN PurchaseMethod INTEGER DEFAULT 1; ";
command.CommandText = addPurchaseMethodColumn;
command.ExecuteNonQuery();
// bump user_version
command.CommandText = "PRAGMA user_version = 19;";

```

```

        command.ExecuteNonQuery();
        command.Parameters.Clear();
        break;
    }
}
else
{
    reader.Close();
}
}
}

private void CreateDatabase()
{
    // create directory (if needed) and sqlite file
    if (!Directory.Exists(_directory))
    {
        Directory.CreateDirectory(_directory);
    }
    SQLiteConnection.CreateFile(GetFilePath());
    // now open and create the database
    using (var conn = GetDatabaseConnectionWithoutMigrating())
    {
        using (var command = GetSQLiteCommand(conn))
        {
            string createUsersTable = "CREATE TABLE Users (" +
                "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
                "Name TEXT," +
                "Username TEXT," +
                "PasswordHash TEXT)";
            command.CommandText = createUsersTable;
            command.ExecuteNonQuery();

            string createCurrenciesTable = "CREATE TABLE Currencies (" +
                "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
                "Name TEXT," +
                "Abbreviation TEXT," +
                "Symbol TEXT," +
                "ConversionRateToUSD TEXT," +
                "IsDefaultCurrency INTEGER DEFAULT 0)";
            command.CommandText = createCurrenciesTable;
            command.ExecuteNonQuery();

```



```

string createInventoryItemTable = "CREATE TABLE InventoryItems (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "Name TEXT," +
    "Description TEXT," +
    "PicturePath TEXT," +
    "Cost TEXT," +
    "CostCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "ProfitPerItem TEXT," +
    "ProfitPerItemCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "Quantity INTEGER," +
    "BarcodeNumber TEXT," +
    "WasDeleted INTEGER DEFAULT 0," +
    "CreatedByUserID INTEGER REFERENCES Users(ID))";
command.CommandText = createInventoryItemTable;
command.ExecuteNonQuery();

string createItemSoldInfoTable = "CREATE TABLE ItemsSoldInfo (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "DateTimeSold TEXT," +
    "QuantitySold INTEGER DEFAULT 1," +
    "Cost TEXT," +
    "CostCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "Paid TEXT," +
    "PaidCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "Change TEXT," +
    "ChangeCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "ProfitPerItem TEXT," +
    "ProfitPerItemCurrencyID INTEGER REFERENCES Currencies(ID)," +
    "InventoryItemID INTEGER REFERENCES InventoryItems(ID)," +
    "SoldByUserID INTEGER REFERENCES Users(ID) )";
command.CommandText = createItemSoldInfoTable;
command.ExecuteNonQuery();

string createGeneratedBarcodesTable = "CREATE TABLE GeneratedBarcodes (" +
    "ID INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT," +
    "Number INTEGER," +
    "DateTimeGenerated TEXT," +
    "GeneratedByUserID INTEGER REFERENCES Users(ID) )";
command.CommandText = createGeneratedBarcodesTable;
command.ExecuteNonQuery();

```

```

// add initial data
// add default user
string addInitialUser = "" +
    "INSERT INTO Users (Name, Username, PasswordHash) VALUES (@name,
@username, @passwordHash)";
    command.CommandText = addInitialUser;
    command.Parameters.Clear();
    command.Parameters.AddWithValue("@name", "Administrator");
    command.Parameters.AddWithValue("@username", "admin");
    command.Parameters.AddWithValue("@passwordHash",
User.HashPassword("changeme"));
    command.ExecuteNonQuery();

// add default currencies
string addCurrency = "" +
    "INSERT INTO Currencies (Name, Abbreviation, Symbol, ConversionRateToUSD,
IsDefaultCurrency) " +
    "VALUES (@name, @abbreviation, @symbol, @conversion, @isDefault)";
    command.CommandText = addCurrency;
    command.Parameters.Clear();
    command.Parameters.AddWithValue("@name", "US Dollar");
    command.Parameters.AddWithValue("@abbreviation", "USD");
    command.Parameters.AddWithValue("@symbol", "$");
    command.Parameters.AddWithValue("@conversion", "1.0");
    command.Parameters.AddWithValue("@isDefault", false);
    command.ExecuteNonQuery();
    command.Parameters.Clear();
    command.Parameters.AddWithValue("@name", "Cambodian Riel");
    command.Parameters.AddWithValue("@abbreviation", "KHR");
    command.Parameters.AddWithValue("@symbol", "៛");
    command.Parameters.AddWithValue("@conversion", "4050");
    command.Parameters.AddWithValue("@isDefault", true);
    command.ExecuteNonQuery();

    command.CommandText = "PRAGMA user_version = 0";
    command.Parameters.Clear();
    command.ExecuteNonQuery();

// close the connection
conn.Close();
}
}

```

```

    }
}
} using ClosedXML.Excel;
using SimpleInventory.Models;
using System;
using System.Collections.Generic;
using System.Diagnostics;

namespace SimpleInventory.Helpers
{
    class StockInfoExcelGenerator
    {

        public void ExportStockInfo(List<DetailedStockReportInfo> items, DateTime startDate, DateTime
endDate, string path)
        {
            items.Sort((a, b) => (a.Item.Name + a.Item.Description).ToLower().CompareTo((b.Item.Name +
b.Item.Description).ToLower()));
            var startDateString = startDate.ToString(Utilities.DateTimeToFriendlyFullDateTimeStringFormat());
            var endDateString = endDate.ToString(Utilities.DateTimeToFriendlyFullDateTimeStringFormat());
            using (var workbook = new XLWorkbook())
            {
                var worksheet = workbook.Worksheets.Add("Stock Info");
                worksheet.Cell("A1").Value = "SimpleInventory -- Stock Info Report for Sold Items";
                worksheet.Cell("A1").Style.Font.Bold = true;
                worksheet.Cell("A2").Value = startDateString + " - " + endDateString;

                // table headers
                worksheet.Cell("A4").SetValue("Name").Style.Font.SetBold(true);
                worksheet.Cell("B4").SetValue("Description").Style.Font.SetBold(true);
                worksheet.Cell("C4").SetValue("Beginning Stock (Computer)").Style.Font.SetBold(true);
                worksheet.Cell("D4").SetValue("Ending Stock (Computer)").Style.Font.SetBold(true);
                worksheet.Cell("E4").SetValue("Ending Stock (Manual Entry)").Style.Font.SetBold(true);
                worksheet.Cell("F4").SetValue("Computer Difference").Style.Font.SetBold(true);
                worksheet.Cell("G4").SetValue("Manual Difference").Style.Font.SetBold(true);
                worksheet.Cell("H4").SetValue("Stock Difference").Style.Font.SetBold(true);
                worksheet.Cell("I4").SetValue("Item Cost").Style.Font.SetBold(true);
                worksheet.Cell("J4").SetValue("Cost Difference (Missing Items)").Style.Font.SetBold(true);
                worksheet.Cell("K4").SetValue("Cost Difference (Extra Items)").Style.Font.SetBold(true);

                // start exporting data
                var currentCell = worksheet.Cell("A5");

```

```

var lastRow = currentCell.WorksheetRow();
IXLCell firstCellWithData = null;
// TODO: adjust formulas with string.Format() rather than string concat
foreach (DetailedStockReportInfo item in items)
{
    lastRow = currentCell.WorksheetRow();
    if (firstCellWithData == null)
    {
        firstCellWithData = currentCell;
    }
    currentCell.Value = item.Item.Name;
    currentCell.CellRight(1).Value = item.Item.Description;
    currentCell.CellRight(2).Value = item.StartStockWithPurchaseStockIncrease;
    currentCell.CellRight(3).Value = item.EndStock; // computer
    currentCell.CellRight(4).Value = ""; // manual entry
    currentCell.CellRight(4).AddConditionalFormat()
        .WhenEquals("\\"")
        .Fill.SetBackgroundColor(XLColor.Yellow); // if data not entered, highlight that
work needs to happen!!

    currentCell.CellRight(5).FormulaA1 = "=SUM(-" +
currentCell.CellRight(2).Address.ToStringFixed() + ","
    + currentCell.CellRight(3).Address.ToStringFixed() + ")"; // computer diff
    currentCell.CellRight(6).FormulaA1 = "=IF(" +
currentCell.CellRight(3).Address.ToStringFixed() + "<\"\", \"-\", \"
    + \"SUM(-\" + currentCell.CellRight(2).Address.ToStringFixed() + \",\"
    + currentCell.CellRight(4).Address.ToStringFixed() + \"))"; // manual diff

    currentCell.CellRight(5).AddConditionalFormat()
        .WhenNotEquals("=" + currentCell.CellRight(6).Address.ToStringFixed())
        .Fill.SetBackgroundColor(XLColor.LightPink);
    currentCell.CellRight(6).AddConditionalFormat()
        .WhenNotEquals("=" + currentCell.CellRight(5).Address.ToStringFixed())
        .Fill.SetBackgroundColor(XLColor.LightPink);
    currentCell.CellRight(7).SetFormulaA1("=ABS(SUM(" +
currentCell.CellRight(5).Address.ToStringFixed() + ", -"
    +
        currentCell.CellRight(6).Address.ToStringFixed()
    +
    "))").AddConditionalFormat()
        .WhenNotEquals("0")
        .Fill.SetBackgroundColor(XLColor.LightPink); // stock difference
    currentCell.CellRight(8).Value = item.Item.Cost; // item cost
    // first sum column is items that have less in real life than in the computer

```

```

string formula = string.Format("=IF({0}<>\"\", IF({1}>{2},{3}*{4},\"\"), \"\")",
    currentCell.CellRight(4).Address.ToStringFixed(), // ending stock manual entry
    currentCell.CellRight(3).Address.ToStringFixed(), // ending stock computer
    currentCell.CellRight(4).Address.ToStringFixed(), // ending stock manual entry
    currentCell.CellRight(7).Address.ToStringFixed(), // stock diff
    currentCell.CellRight(8).Address.ToStringFixed() // item cost
);
currentCell.CellRight(9).SetFormulaA1(formula); // cost difference for missing items
// second sum column is items that have more in real life than in the computer
formula = string.Format("=IF({0}<>\"\", IF({1}<{2},{3}*{4},\"\"), \"\")",
    currentCell.CellRight(4).Address.ToStringFixed(), // ending stock manual entry
    currentCell.CellRight(3).Address.ToStringFixed(), // ending stock computer
    currentCell.CellRight(4).Address.ToStringFixed(), // ending stock manual entry
    currentCell.CellRight(7).Address.ToStringFixed(), // stock diff
    currentCell.CellRight(8).Address.ToStringFixed() // item cost
);
currentCell.CellRight(10).SetFormulaA1(formula); // cost difference for extra items
// if item count is equal, doesn't add to either column

if (currentCell.WorksheetRow().RowNumber() % 2 == 0)
{
    currentCell.WorksheetRow().Style.Fill.BackgroundColor = XLColor.LightGray;
}

// if you add more data columns make sure to adjust print area!!!

// go to next row
currentCell = currentCell.CellBelow();
}
// add cost discrepancy
if (items.Count > 0)
{
    currentCell.CellRight(8).SetValue("Cost Discrepancy").Style.Font.SetBold(true);
    currentCell.CellRight(9).SetFormulaA1("=SUM("
firstCellWithData.CellRight(9).Address.ToStringFixed()
        + ":" + currentCell.CellAbove(1).CellRight(9).Address.ToStringFixed()
    ").Style.Font.SetBold(true);
    currentCell.CellRight(10).SetFormulaA1("=SUM("
firstCellWithData.CellRight(10).Address.ToStringFixed()
        + ":" + currentCell.CellAbove(1).CellRight(10).Address.ToStringFixed()
    ").Style.Font.SetBold(true);
}

```

```

        /// auto fit width
        worksheet.Columns().AdjustToContents(4, 4, 10, 25);
        // set print area
        worksheet.PageSetup.PrintAreas.Clear();
        var firstCellForPrinting = worksheet.Cell("A1");
        var lastCellForPrinting = items.Count > 0 ? currentCell.CellRight(10) :
worksheet.Cell("J4");
        worksheet.PageSetup.PrintAreas.Add(firstCellForPrinting.Address.ToStringRelative() +
":" + lastCellForPrinting.Address.ToStringRelative());
        worksheet.PageSetup.SetRowsToRepeatAtTop("4:4");
        worksheet.PageSetup.PagesWide = 1;
        worksheet.PageSetup.PageOrientation = XLPPageOrientation.Landscape;
        workbook.SaveAs(path);
        Process.Start(path);
    }
}
}
}
using SimpleInventory.Models;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Runtime.InteropServices;
using System.Security;
using System.Text;
using System.Threading.Tasks;

namespace SimpleInventory.Helpers
{
    class Utilities
    {
        public static bool InDesignMode()
        {
            return LicenseManager.UsageMode == LicenseUsageMode.Designtime;
        }

        public static string DateTimeToStringFormat()
        {
            return "yyyy-MM-dd HH:mm:ss"; // 24 hr time (0-23)
        }
    }
}

```

```

public static string DateTimeToDateOnlyStringFormat()
{
    return "yyyy-MM-dd";
}

```

```

public static string DateTimeToFriendlyFullDateTimeStringFormat()
{
    return "dddd, d MMMM, yyyy 'at' h:mm:ss tt";
}

```

```

public static string DateTimeToFriendlyJustDateStringFormat()
{
    return "dddd, d MMMM, yyyy";
}

```

```

public static decimal ConvertAmount(decimal amount, Currency initialCurrency, Currency toCurrency)
{
    if (initialCurrency.ID == toCurrency.ID)
    {
        return amount;
    }
    else
    {
        // / = convert to USD, then convert to other currency
        return amount / initialCurrency.ConversionRateToUSD * toCurrency.ConversionRateToUSD;
    }
}

```

```

public static decimal ConvertAmountWithRates(decimal amount, decimal initialCurrencyConversion,
decimal toCurrencyConversion)
{
    // / = convert to USD, then convert to other currency
    return amount / initialCurrencyConversion * toCurrencyConversion;
}

```

```

public static Currency CurrencyForOrder(List<ItemSoldInfo> items)
{
    Currency currency = null;
    foreach (var item in items)
    {
        if (currency == null)
        {

```

```

        currency = item.CostCurrency;
    }
    else if (item.CostCurrency != null)
    {
        if (currency.ID != item.CostCurrency.ID)
        {
            return null; // not all currencies are the same
        }
    }
}
return currency;
}

```

// <https://stackoverflow.com/a/819705>

// I don't care that much about this string being in RAM for a short time. :)

```

public static string SecureStringToString(SecureString value)
{
    IntPtr valuePtr = IntPtr.Zero;
    try
    {
        valuePtr = Marshal.SecureStringToGlobalAllocUnicode(value);
        return Marshal.PtrToStringUni(valuePtr);
    }
    catch { }
    finally
    {
        Marshal.ZeroFreeGlobalAllocUnicode(valuePtr);
    }
    return "";
}
}
}

```



**CD-накопитель с программой**