Оглавление

[ВВЕДЕНИЕ 2](#_Toc166296072)

[ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЩАЯ ПОСТАНОВКА ЗАДАЧ 4](#_Toc166296073)

[1.1 Историю развития учета лекарств в аптеках 4](#_Toc166296074)

[1.2Обзор существующих методов учета лекарств в аптеках 5](#_Toc166296075)

[1.3Требования к информационной системе учета лекарств в аптеке 9](#_Toc166296076)

[1.4 Постановка задач 13](#_Toc166296077)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ 16](#_Toc166296078)

[2.1 Концептуальное проектирование программы 16](#_Toc166296079)

[2.2 Структура программного продукта 16](#_Toc166296080)

[2.3 Руководство пользователя 17](#_Toc166296081)

[2.4Руководство программиста 27](#_Toc166296082)

[ЗАКЛЮЧЕНИЕ 29](#_Toc166296083)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 30](#_Toc166296084)

[ПРИЛОЖЕНИЕ 31](#_Toc166296085)

# ВВЕДЕНИЕ

В наше время аптеки являются ключевым элементом здравоохранения, обеспечивая население необходимыми лекарственными средствами. Однако, несмотря на значимость их функции, многие аптеки сталкиваются с проблемами в учете и управлении запасами лекарств. Неэффективные методы учета и отсутствие автоматизации процессов могут привести к ошибкам в поставках, излишкам или недостатку лекарств, что в конечном итоге может негативно сказаться на обслуживании пациентов. В связи с этим возникает актуальная потребность в разработке информационной системы учета лекарств, которая позволила бы оптимизировать процессы управления запасами и повысить качество обслуживания.

Целью данной работы является разработка информационной системы учета лекарств в аптеке с целью повышения эффективности управления запасами и обеспечения более качественного обслуживания пациентов.

Для реализации поставленной цели, необходимо решить следующие задачи:

1. Анализ требований и составление технического задания.
2. Проектирование системы для хранения информации о лекарствах, покупках, редактирование данных в системе хранения.
3. Разработка интерфейса для удобного редактирования и добавления данных.
4. Разработка пользовательского интерфейса для удобного ввода и отображения данных.
5. Реализация функционала продаж, вывода количества позиций и прочих операций.
6. Обеспечение безопасности данных.
7. Тестирование системы на соответствие требованиям и корректировка ошибок.

Решение данных задач позволит создать информационную систему, которая не только улучшит работу аптеки, но и способствует повышению качества обслуживания клиентов и оптимизации бизнес-процессов в сфере здравоохранения.

Предметом исследования является процесс разработки информационной системы учета лекарств в аптеке. Это включает в себя анализ существующих методов учета лекарств, определение требований к информационной системе, разработку структуры хранения данных, реализацию функций системы, ее внедрение и эксплуатацию.

Объектом исследования выступают аптеки и их управление запасами лекарств. Это включает в себя процессы закупки, хранения, отслеживания сроков годности, продажи и учета лекарственных препаратов. Исследование направлено на разработку информационной системы, которая поможет оптимизировать и автоматизировать эти процессы, улучшая качество обслуживания пациентов и эффективность работы аптек.

Данное исследование основывается на принципах информационных систем, технологиях хранения данных, методах анализа и проектирования систем, а также теории управления запасами. Эти теоретические основы позволяют разработать информационную систему, которая эффективно учитывает и управляет лекарственными средствами в аптеке, улучшая тем самым качество обслуживания пациентов и оптимизируя работу аптеки в целом.

# ГЛАВА 1. ОБЗОР ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЩАЯ ПОСТАНОВКА ЗАДАЧ

## **1.1 Историю развития учета лекарств в аптеках**

История развития учета лекарств в аптеках насчитывает множество важных этапов, которые отразились на современной системе учета. Давайте вместе рассмотрим основные моменты этого процесса.

В древности и в средние века учет лекарств велся в основном вручную. Аптекари вели записи о поступлении и продаже лекарственных средств, используя простые документы, такие как журналы и книги учета. Этот процесс был трудоемким и подвержен ошибкам.

С развитием научных знаний и расцветом медицины в 17-18 веках возникла потребность в более систематическом учете лекарств. В этот период появились первые формы организации учета, такие как карточки учета и журналы прихода и расхода.

В XIX веке с ростом населения и увеличением объемов торговли аптеки сталкивались с растущей сложностью учета лекарств. Это привело к необходимости создания более эффективных систем учета и контроля за запасами.

Во второй половине XX века с развитием компьютерных технологий начался процесс автоматизации учета лекарств. Появились первые программы и информационные системы, предназначенные для ведения учета и управления аптечными запасами.

В настоящее время современные информационные технологии играют ключевую роль в учете лекарств в аптеках. Автоматизированные системы позволяют вести учет в реальном времени, отслеживать запасы, контролировать сроки годности и формировать отчеты с высокой точностью и скоростью.

Таким образом, история развития учета лекарств в аптеках отражает постепенное совершенствование системы учета от простых ручных методов до современных автоматизированных систем, что способствует более эффективному управлению аптекой и обеспечению качественного обслуживания пациентов

**1.2Обзор существующих методов учета лекарств в аптеках**

Традиционные методы учета лекарств включают использование карточек учета, журналов прихода и расхода, а также другие подходы, применяемые в аптечной практике. Карточки учета являются основным инструментом для записи информации о каждом лекарственном препарате. Они содержат данные о названии лекарства, дозировке, сроке годности, поставках и продажах. Журналы прихода и расхода представляют собой документы, в которых фиксируются операции по поступлению и реализации лекарств. Каждый приход и расход лекарств регистрируется с указанием количества, даты и других сопутствующих данных. Помимо этого, существуют и другие традиционные методы учета, такие как использование инвентарных таблиц и учетных книг.

Преимущества традиционных методов учета:

Простота и доступность: Одним из основных преимуществ традиционных методов учета является их простота в использовании и доступность для широкого круга пользователей. Карточки учета и журналы прихода и расхода легко вводятся и используются аптекарями без необходимости специальной подготовки.

Надежность: Традиционные методы учета характеризуются высокой степенью надежности. При правильном использовании и хранении записей на бумажных носителях они могут служить долгое время без потери информации.

Привычность: Многие аптекари имеют опыт работы с традиционными методами учета и привыкли к ним. Это делает переход на новые технологии более сложным процессом и может вызывать сопротивление со стороны персонала.

Недостатки традиционных методов учета:

Ручной труд: Один из основных недостатков традиционных методов учета заключается в необходимости ручного ввода и обработки данных. Это может быть трудоемким процессом, требующим много времени и увеличивающим вероятность ошибок.

Ограниченная автоматизация: Традиционные методы учета не обеспечивают полной автоматизации процесса учета, что может затруднить анализ и контроль запасов лекарств. Например, они могут не предоставлять возможности автоматического уведомления о просроченных сроках годности или низких запасах.

Уязвимость к ошибкам: Ручной ввод данных в традиционных методах учета увеличивает вероятность ошибок, таких как опечатки, неверный расчет количества или неправильное заполнение документов. Это может привести к неправильному учету запасов и потере денежных средств.

Ограниченная аналитика и отчетность: Традиционные методы учета часто не предоставляют возможности для подробного анализа данных и формирования разнообразных отчетов, что затрудняет принятие обоснованных решений по управлению аптекой.

Ограниченная масштабируемость: При росте объема работы традиционные методы учета могут стать неэффективными и трудно масштабируемыми. Их использование может ограничивать развитие аптеки и усложнять управление большими объемами данных.

Современные информационные технологии предлагают более эффективные и автоматизированные способы учета лекарств в аптеках. Автоматизированные системы учета лекарств позволяют аптекарям автоматизировать процессы учета, отслеживания запасов и оформления заказов. Специализированные программы и приложения для учета лекарств предоставляют возможность удобного ведения учета на компьютере или мобильном устройстве. Эти технологии обеспечивают более точный и быстрый учет, а также позволяют оптимизировать процессы управления аптечными запасами. Они также обеспечивают возможность автоматической генерации отчетов, анализа данных и прогнозирования спроса.

Преимущества современных методов учета:

- Автоматизация процессов: Современные информационные системы позволяют автоматизировать множество операций, связанных с учетом лекарств. Это включает в себя автоматическое отслеживание поступления и реализации лекарств, контроль за сроками годности и формирование отчетов.

- Точность данных: Благодаря использованию современных методов учета, данные о наличии лекарств становятся более точными и достоверными. Автоматический сбор и обработка информации уменьшают риск ошибок

- Увеличение скорости работы: Современные системы учета позволяют значительно ускорить выполнение операций по учету лекарств. Это особенно важно в условиях высокой загруженности аптеки, когда каждая минута имеет значение.

- Лучшая аналитика и отчетность: Современные системы предоставляют более широкие возможности для анализа данных и формирования отчетов. Аптекари могут получать подробную информацию о продажах, остатках, динамике спроса и других показателях, что помогает принимать решения по управлению аптекой.

- Гибкость и настраиваемость: Современные системы учета обычно имеют гибкую конфигурацию, которая позволяет адаптировать их под индивидуальные потребности аптеки. Структура систем позволяет выбирать необходимый функционал и настраивать его согласно требованиям.

Недостатки современных методов учета:

- Высокие затраты на внедрение и обслуживание: Внедрение современных систем учета часто требует значительных финансовых вложений на закупку программного обеспечения, обучение персонала и поддержку системы.

- Необходимость обучения персонала: Использование современных систем учета требует обучения персонала, что может занять некоторое время и потребовать дополнительных ресурсов.

- Риск технических сбоев: Как и любая другая технология, современные системы учета подвержены риску технических сбоев, которые могут привести к потере данных и нарушению работы аптеки.

- Необходимость постоянного обновления: Современные системы требуют постоянного обновления и поддержки для обеспечения их корректной работы и соответствия современным требованиям и стандартам.

- Зависимость от ИТ-инфраструктуры: Эффективная работа современных систем учета зависит от качества и надежности информационно-технической инфраструктуры, что может быть проблемой в случае технических неполадок или отсутствия доступа к интернету.

Сравнение современных и традиционных методов для учета лекарств является важным пунктом при выборе оптимальной системы учета для аптеки. Рассмотрим основные различия и сходства между этими методами учета данных о товаре

1.Точность и надежность:

Современные методы учета, такие как автоматизированные системы и специализированные программы, обычно обеспечивают более высокую точность и надежность учета по сравнению с традиционными методами. Например, автоматизированные системы могут автоматически отслеживать запасы лекарств и предупреждать об их необходимости пополнения, что уменьшает риск ошибок и недостачи товара на полках. Также современные программы часто имеют функции проверки данных и контроля за сроками годности, что повышает качество учета и безопасность лекарственных средств.

2.Эффективность и скорость:

Современные методы учета обычно более эффективны и быстры в использовании, поскольку они автоматизированы и обеспечивают возможность быстрого доступа к информации. Например, автоматизированные системы могут проводить анализ данных и формировать отчеты за короткое время, что позволяет аптекарям быстро принимать решения и управлять запасами лекарств. В то время как традиционные методы могут потребовать больше времени на обработку данных и ведение учета вручную.

3.Сложность и стоимость:

В то время как современные методы учета могут предоставлять больше функциональности и удобства, их внедрение и обслуживание часто требует дополнительных затрат на приобретение программного обеспечения и обучение персонала. Например, для внедрения автоматизированной системы учета может потребоваться инвестировать в закупку и настройку программного обеспечения, обучение персонала и поддержку системы. Однако в долгосрочной перспективе современные методы учета могут сэкономить средства за счет оптимизации процессов управления запасами и уменьшения риска ошибок.

4.Гибкость и настраиваемость:

Современные методы учета обычно предлагают большую гибкость и возможность настройки под конкретные потребности аптеки. Например, они позволяют создавать индивидуальные отчеты, настраивать права доступа к данным и интегрировать систему учета с другими информационными системами. Таким образом, современные методы учета могут быть более адаптивными и способными адаптироваться к изменяющимся потребностям аптеки. В то время как традиционные методы могут быть менее гибкими и иметь ограниченные возможности настройки.

**1.3Требования к информационной системе учета лекарств в аптеке**

Современная медицина тесно связана с аптеками, которые являются незаменимым звеном в обеспечении населения необходимыми лекарственными средствами. Каждый день тысячи людей обращаются в аптеки в поисках медикаментов, способных облегчить их состояние и улучшить качество жизни. Однако, с ростом спроса и разнообразием лекарственных препаратов возникают сложности в их учете, контроле и управлении. Именно в этой сфере проявляется острая необходимость в эффективной информационной системе учета лекарств в аптеке.

Подобная информационная система представляет собой не просто программный продукт, а сложную систему, способную обеспечить полную автоматизацию процессов учета лекарственных средств, от их приобретения до реализации. Но прежде чем перейти к рассмотрению требований к такой системе, важно понять контекст, в котором она функционирует, и проблемы, с которыми сталкиваются аптеки в повседневной деятельности.

Сегодняшняя реальность характеризуется быстрым развитием фармацевтической индустрии, постоянным появлением новых лекарственных препаратов и изменением требований к их учету и хранению. Это создает сложности для аптек в поддержании актуальности информации о наличии лекарств на складе, контроле сроков годности и обеспечении качественного обслуживания пациентов. В такой динамичной среде традиционные методы учета и контроля становятся недостаточно эффективными, что выдвигает на первый план потребность в цифровых решениях.

Важным аспектом является также обеспечение безопасности пациентов. Неправильное управление лекарствами может привести к серьезным последствиям для здоровья людей, вплоть до возникновения нежелательных реакций или осложнений заболевания. Поэтому информационная система учета лекарств должна обеспечивать надежную защиту данных, предотвращая возможность ошибок и исключая риски, связанные с неправильным использованием медикаментов.

Кроме того, следует учитывать тенденции цифровизации и автоматизации процессов в здравоохранении. С каждым годом внедрение информационных технологий в медицинскую сферу становится все более актуальным и необходимым. Информационные системы учета лекарств в аптеке не только оптимизируют работу персонала и сокращают временные затраты на рутинные процессы, но и повышают качество обслуживания пациентов, предоставляя им доступ к актуальной информации о лекарствах и их характеристиках.

Таким образом, разработка информационной системы учета лекарств в аптеке — это необходимый шаг в направлении современного и эффективного управления лекарственными ресурсами. В дальнейшем мы рассмотрим основные требования к такой системе, учитывая ее функциональные и нефункциональные аспекты, а также специфику работы аптек.

Функциональные требования

Регистрация и аутентификация пользователей: Система должна предоставлять возможность регистрации новых пользователей и аутентификации уже зарегистрированных. Это обеспечит безопасность данных и контроль доступа к информации.

Учет лекарственных препаратов: Система должна иметь функционал для регистрации поступления лекарственных препаратов в аптеку, включая информацию о наименовании, количестве, дате поступления и сроке годности. Кроме того, необходимо учитывать их категории, дозировки и другие характеристики.

Система отслеживания лекарств: Важным функционалом системы является возможность отслеживания перемещения лекарственных препаратов внутри аптеки, начиная от их поступления на склад и заканчивая продажей или списанием. Это позволит поддерживать актуальные данные о наличии лекарств на складе.

Оформление заказов от покупателей: Информационная система должна обеспечивать возможность оформления заказов от покупателей как в онлайн, так и в офлайн формате. Это включает в себя возможность выбора лекарств, добавление их в корзину, оформление заказа с указанием контактных данных покупателя и способа доставки.

Генерация отчетов: Система должна иметь функционал для генерации различных отчетов, включающих информацию о запасах лекарств, обороте и прибыли. Это позволит аптеке анализировать свою деятельность, выявлять тенденции и принимать обоснованные управленческие решения.

Нефункциональные требования

Производительность: Система должна обеспечивать высокую производительность даже при больших объемах данных и высокой нагрузке. Это гарантирует быстрый доступ к информации и минимизацию времени ожидания для пользователей.

Безопасность: Безопасность данных является приоритетом при разработке информационной системы. Система должна обеспечивать защиту информации от несанкционированного доступа, в том числе с использованием шифрования данных и механизмов аутентификации.

Удобство использования: Интерфейс системы должен быть интуитивно понятным и удобным для использования даже для пользователей без специального обучения. Это поможет снизить вероятность ошибок при работе с системой и повысить эффективность работы персонала аптеки.

Надежность: Система должна быть надежной и стабильной, чтобы минимизировать риск возникновения сбоев или потери данных. Регулярное резервное копирование данных и мониторинг состояния системы помогут обеспечить ее надежную работу.

Масштабируемость: Информационная система должна быть готова к масштабированию, чтобы адаптироваться к изменяющимся потребностям и объемам работы аптеки. Это включает в себя возможность добавления новых функций и расширение ее возможностей без серьезных изменений в архитектуре системы.

Разработка информационной системы учета лекарств в аптеке представляет собой важный этап в совершенствовании фармацевтической индустрии и повышении качества медицинского обслуживания. В современном мире, где здоровье населения играет ключевую роль, эффективное управление и контроль за лекарственными препаратами становятся необходимостью, на которую нельзя пренебрегать.

В ходе нашего исследования мы выявили множество требований к информационной системе учета лекарств, которые необходимо учесть при ее разработке и внедрении. Основные функциональные требования включают в себя регистрацию и аутентификацию пользователей, учет лекарственных препаратов, отслеживание движения товара, оформление заказов от покупателей и генерацию отчетов. Эти функции позволят не только эффективно управлять ассортиментом лекарств, но и оптимизировать работу персонала аптеки и обеспечить качественное обслуживание клиентов.

Однако, помимо функциональных требований, важно уделить внимание и нефункциональным аспектам, таким как производительность, безопасность, удобство использования, надежность и масштабируемость системы. Именно сочетание этих факторов позволит создать информационную систему, которая будет эффективно поддерживать работу аптеки и повышать ее конкурентоспособность на рынке.

В заключение, следует отметить, что информационная система учета лекарств в аптеке — это не только технологическое решение, но и инструмент, способствующий заботе о здоровье и благополучии пациентов. Ее внедрение позволит существенно улучшить процессы управления лекарственными ресурсами, повысить эффективность работы аптеки и обеспечить высокий уровень обслуживания пациентов. Таким образом, разработка и внедрение информационной системы учета лекарств в аптеке являются важным шагом в направлении современного и качественного здравоохранения.

1. Начало формы

## **1.4 Постановка задач**

При разработке информационной системы учета лекарств в аптеке необходимо провести комплексный анализ требований и определить набор задач, которые обеспечат эффективное функционирование системы, удовлетворяя потребности аптеки и ее клиентов. Ниже приведены ключевые задачи, которые должны быть решены в процессе разработки:

Анализ потребностей аптеки: Первоочередной задачей является проведение анализа потребностей конкретной аптеки. Это включает в себя изучение особенностей ее деятельности, ассортимента лекарств, численности персонала, объемов продаж и прочих факторов, которые могут повлиять на требования к информационной системе.

Определение функциональных требований: На основе анализа потребностей аптеки определяются функциональные требования к информационной системе. Это включает в себя функции регистрации и аутентификации пользователей, учет поступления и реализации лекарств, генерацию отчетов о запасах, обороте и прибыли, а также другие функции, необходимые для эффективного управления аптекой.

Установление нефункциональных требований: Помимо функциональных требований, необходимо учесть нефункциональные аспекты, такие как производительность, безопасность, удобство использования и надежность системы. Это включает в себя обеспечение высокой производительности при работе с большим объемом данных, защиту конфиденциальности данных пациентов, интуитивно понятный интерфейс пользователя и надежную работу системы даже в условиях высокой нагрузки.

Разработка архитектуры системы: На основе вышеуказанных требований разрабатывается архитектура информационной системы. Это включает в себя выбор подходящих технологий и инструментов разработки, определение структуры базы данных, а также разработку алгоритмов обработки данных и логики работы системы.

Планирование и распределение задач: После определения всех требований и разработки архитектуры системы происходит планирование и распределение задач между участниками команды разработки. Это включает в себя определение сроков выполнения работ, оценку ресурсов и выделение приоритетов.

Разработка и тестирование: Непосредственно после планирования начинается разработка информационной системы. Параллельно с разработкой проводится тестирование, чтобы обеспечить высокое качество и надежность системы.

Внедрение и поддержка: После завершения разработки и успешного прохождения тестирования происходит внедрение системы в аптеку. После внедрения осуществляется поддержка и обновление системы, чтобы обеспечить ее бесперебойную работу и соответствие требованиям аптеки в долгосрочной перспективе.

Каждая из перечисленных выше задач играет важную роль в процессе разработки информационной системы учета лекарств в аптеке и направлена на обеспечение ее эффективного функционирования и соответствия потребностям заказчика.

# ГЛАВА 2. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## **2.1 Концептуальное проектирование программы**

Концептуальное проектирование представляет собой начальную стадию жизненного цикла проекта, на которой зарождаются основные идеи и решения по реализации проекта, оказывающие максимальное влияние на проект в целом и позволяющие своевременно оценить его эффективность и риски. В приложении определено:

1. Модель данных, которая отображает основные сущности аптеки (лекарства, сортировка, поиск лекарств и т.д.)

2. Разработка формы пользовательского интерфейса для различных операций, таких как добавление данных, удаление данных и т.д. Элементы управления находятся на формах, чтобы пользователь мог легко взаимодействовать с приложением.

## **2.2 Структура программного продукта**

Структура программного продукта информационной системе учета лекарств в аптеке на Windows Forms C#:

Формы приложения:

1. ScreenUsesr. Основная форма гостевого аккаунта, содержащая элементы управления для переключения между другими формами, информацию о приложении.
2. Balans. Форма для отображения, пополнения баланса
3. AbautIconUs. Форма информационного характера для отображения информации конкретного лекарства в гостевом аккаунте, находящегося в списке лекарств
4. ScreenAdmin. Основная форма аккаунта администратора, содержащая элементы управления для переключения между другими формами, информацию о приложении.
5. AbautIcon. Форма информационного характера для отображения информации конкретного лекарства в аккаунте администратора, находящегося в списке лекарств.
6. AddUnits. Форма для редактирования количества единиц товара
7. ScreenAdd. Форма для добавления лекарств и данных о них в список.

Основные функции приложения:

1. вывод меню для работы со списком;
2. добавление нового лекарства в начало списка.
3. добавление нового лекарства в конец
4. добавление нового лекарства, отсортированного по названию в алфавитном порядке.
5. добавление нового лекарства перед указанным.
6. добавление нового лекарства после указанного.
7. вывод всего списка лекарств;
8. поиск лекарства с максимальным количеством экземпляров в аптеке.
9. по запросу выдаются сведения о наличии лекарств в аптеке, упорядоченные по стране-производителю.
10. вывод лекарств, сгруппированных по типам.

## **2.3 Руководство пользователя**

Руководство пользователю по использованию приложенияинформационной системы учета лекарств в аптеке предназначено для ознакомленияпользователя с функциональными возможностями приложения, поможетнаучиться использовать его.

Руководство пользователя обеспечивает полную информативность по структуре интерфейса приложения, описывает все реализованные функции.

Инструкция пользователя

Запуск приложения

Чтобы запустить приложение, нужно дважды кликнуть левой кнопкой

мыши (ЛКМ) по исполняемому файлу — файл с расширением .exe. Данное действие показано на рисунке 2.

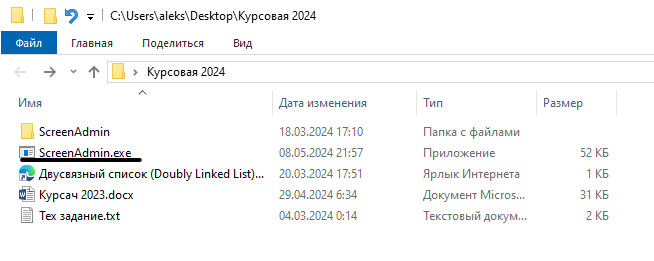


Рис. 2. Запуск приложения

Главное окно пользователя

После нажатия на .exe файл откроется главное окно пользователя (рис. 3). Это окно делится на header (показывает название приложение и баланс рис.4), side bar (боковое меню, где находятся все функции рис. 5), main (меню где отображаются все лекарства рис.6).

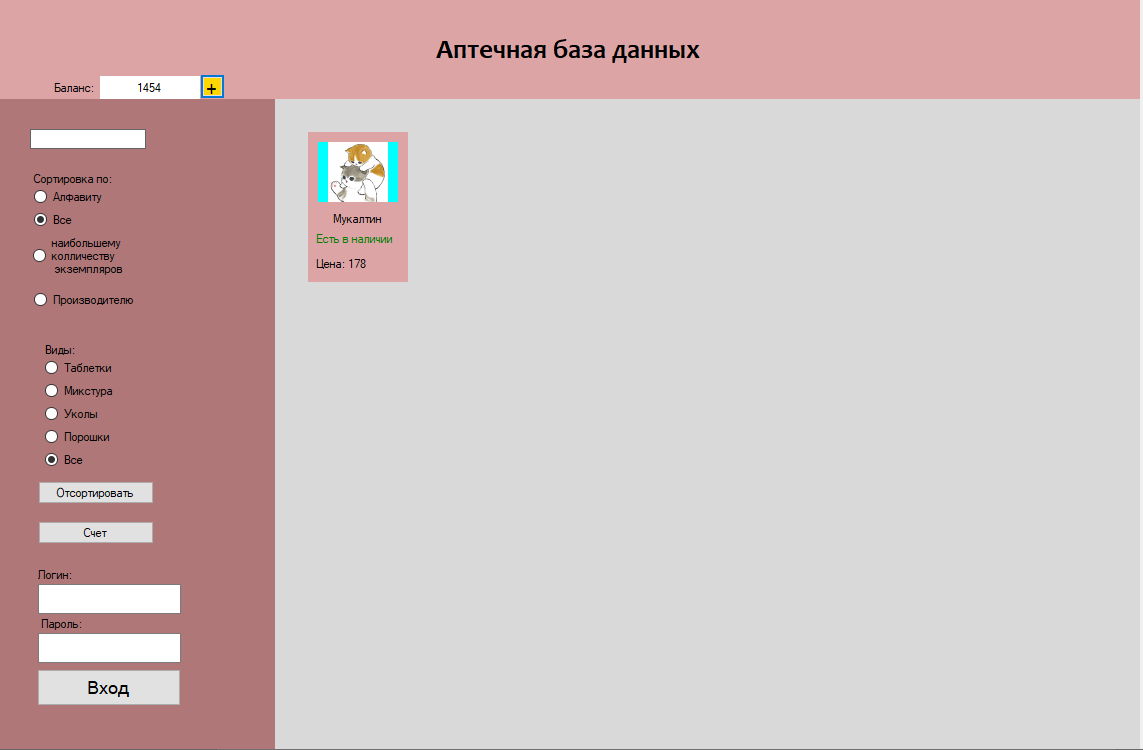


Рис.3. Главное окно пользователя

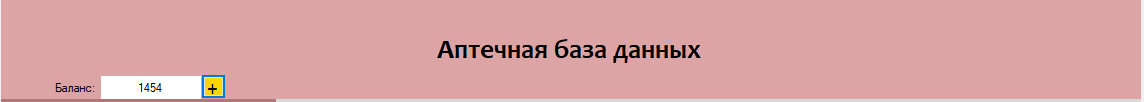
****

Рис.4. Верхний блок

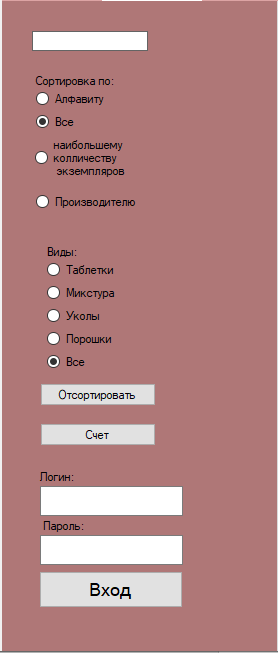


Рис.5. Боковое меню



Рис.6. Блок, содержащий контент

В main находятся иконки всех лекарств, если на иконку кликнуть левой кнопкой мыши(ЛКМ), то откроется окно с полной информацией о лекарстве (рис.7)

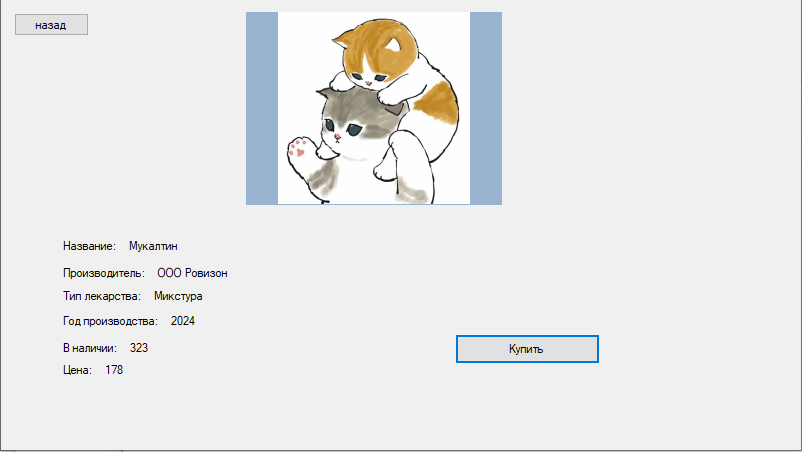


Рис.7. Окно с информацией о лекарстве

При нажатии ЛКМ на кнопку «Купить» будет производится покупка лекарства, при условии, что на балансе достаточно средств для оплаты в противном случае будет выводится надпись: «Недостаточно средств», если экземпляров лекарства будет 0, то при попытке купить товар будет выводится «Товар закончился», а на иконке будет надпись: «Нет в наличии». Если нажать на кнопку «Назад», то вы вернетесь на главный экран пользователя.

В side bar находится поиск лекарств, при вводе в поиск названия лекарства, то при его наличии они буду показаны в списке под поиском(рис.8), в противном случае под поиском ничего не будет указано

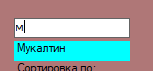


Рис.8. Поиск

Ниже поиска находится сортировка лекарств(рис.9), сортировка происходит после нажатия кнопки «Сортировать»

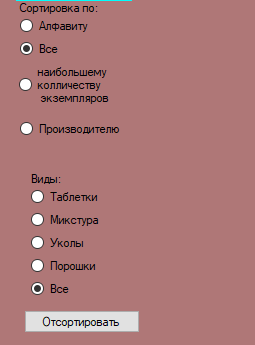


Рис.9. Панель сортировки

При вводе правильного логина и пароля(рис.10) вы попадете на аккаунт администратора, где появятся дополнительные функции.

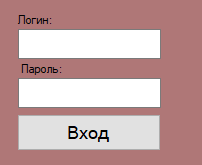


Рис.10. Форма входа в аккаунт администратора

Главный экран администратора

После ввода корректного логина и пароля откроется главное окно администратора (рис. 11). Это окно делится на header (показывает название приложения рис.12), side bar (боковое меню, где находятся все функции рис.13), main (меню где отображаются все лекарства рис.6)

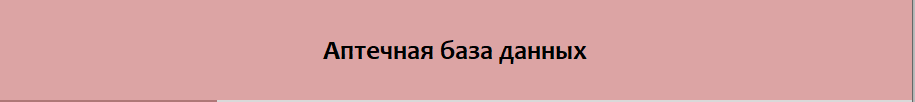


Рис.12. Верхний блок администратора

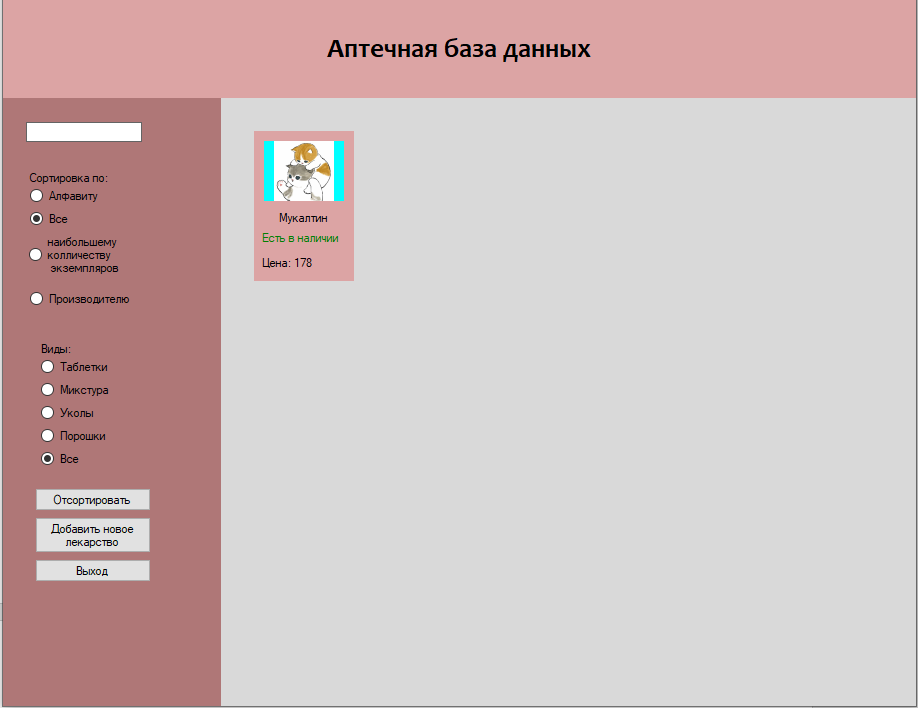


Рис.11. Главное окно администратора

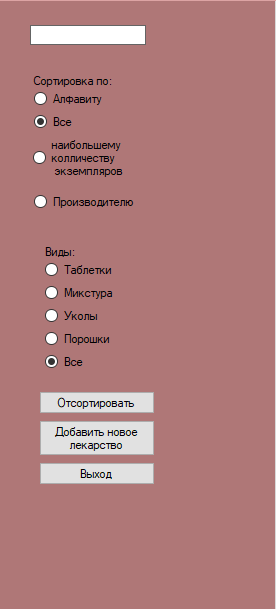


Рис.13. Боковое меню администратора

В main находятся иконки всех лекарств, если на иконку кликнуть левой кнопкой мыши(ЛКМ), то откроется окно с полной информацией о лекарстве(рис.14)

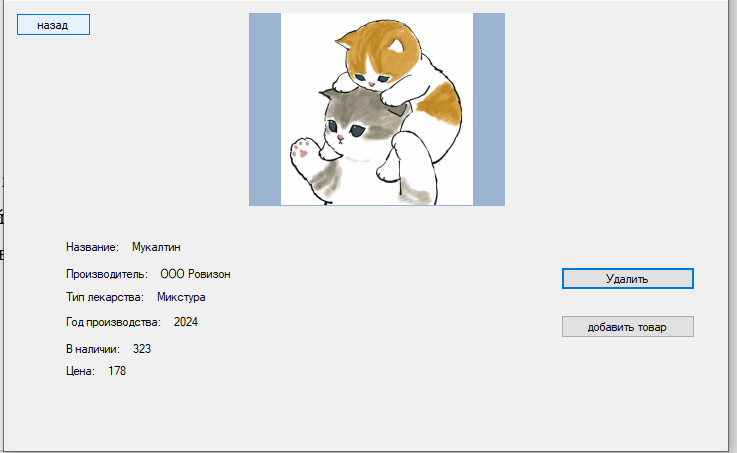


Рис.14. Информационное окно иконки администратора

В данном окне присутствует возможность удаления лекарства из списка всех лекарств (для этого необходимо нажать на кнопку «Удалить»), так же можно изменить количество экземпляров лекарства (для этого необходимо нажать на кнопку «Добавить товар»). Если нажать на кнопку «Назад», то вы вернетесь на главный экран администратора.

В side bar (рис. 14) поиск и сортировка абсолютно такие же, как и в side bar пользователя. В side bar администратора появляется две новых кнопки – «Добавить новое лекарство» и «Выход».

При нажатии кнопки «Добавить новое лекарство», осуществляется переход на окно, в котором происходит создание иконки лекарства (заполнение данных иконки лекарства рис. 15)

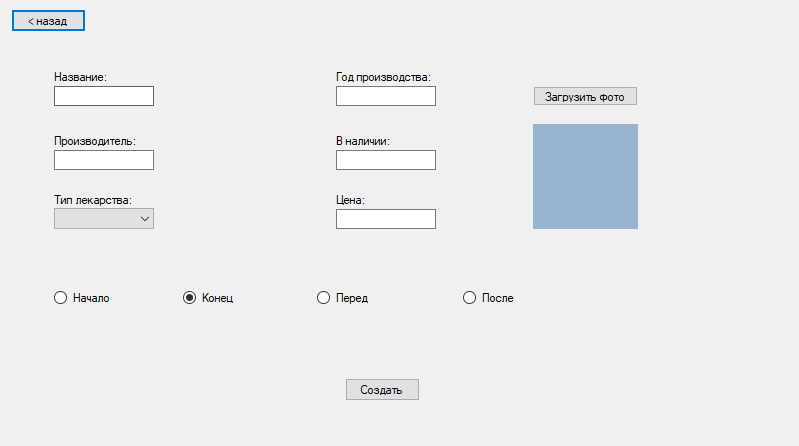


Рис. 15. Окно заполнения данных иконки лекарства

Поля ввода «Названия» и «Производитель», можно вводить с пробелами без ограничений по символам, если какое-то из полей не заполнено, то иконку создать не удастся. Чтобы выбрать тип лекарства необходимо нажать на стрелку под надписью: «Тип лекарства» и выбрать необходимый тип (рис. 16), если ничего не выбрать, то иконка не создаться. Поля ввода «Год производства», «В наличии», «Цена» принимают исключительно положительные цифры, если введенное число таким не является, то иконка не создаться.

При нажатии кнопки «Загрузить фото» появляется окно в котором можно выбрать необходимое фото (рис. 17). Выбранное фото будет отображаться под кнопкой загрузки фото (рис. 18), если фотография не будет выбрана, то при создании иконки лекарства будет выбрана фотография по дефолту.

Так же есть возможность выбирать местонахождения иконки в списке всех лекарств, для того чтобы воспользоваться этой функцией необходимо выбрать пункт и нажать на него (рис. 19). Пункт «Начало» вставляет иконку в начало списка, пункт «Конец» вставляет иконку в конец списка. При нажатии пункта «Перед» появляется поле ввода, в которое необходимо ввести название иконки, перед которой необходимо вставить создаваемую иконку (рис. 20). При нажатии пункта «После» появляется поле ввода, в которое необходимо ввести название иконки, после которой необходимо вставить создаваемую иконку (рис. 21)

При нажатии кнопки «Создать» создаться иконка лекарства и произойдет переход на главный экран администратора, при условии, что вы заполнили все данные корректно.

При нажатии кнопки «Назад» произойдет возврат на главный экран администратора, а все введенные данные удалятся.

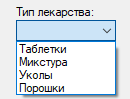


Рис. 16. Раскрытый список типов лекарств

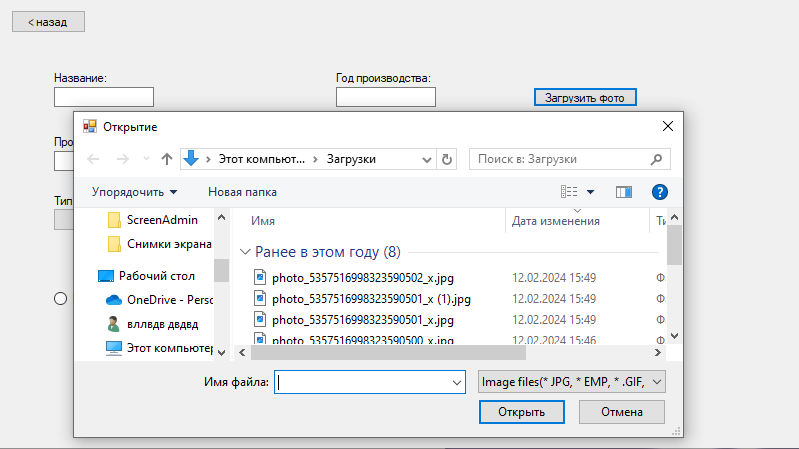


Рис. 17. Окно выбора загружаемой фотографии



Рис. 18. Отображение фотографии после ее выбора

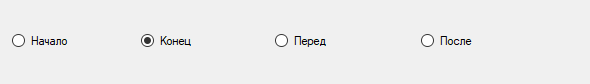


Рис. 19. Пункты выбора местоположения иконки лекарства в списке

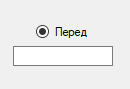


Рис. 20. Пункт вставки создаваемой иконки

перед какой-либо иконки в списке

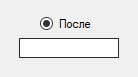


Рис. 21. Пункт вставки создаваемой иконки

после какой-либо иконки в списке

Если в side bar (рис. 13) главного экрана администратора нажать на кнопку «Выход», то вы выполните выход с аккаунта администратора и попадете на аккаунт пользователя.

## **2.4Руководство программиста**

Данная информационная система разработана для хранения текущей информации о лекарствах в аптеке.

Минимальные технические требования к компьютерной технике:

1. Процессор с системой охлаждения, типоразмер сокета «Intel Socket 775», встроенный видеоадаптер, активное охлаждение;
2. Системная (материнская) плата, типоразмер сокета «Intel Socket 775», гигабитный сетевой адаптер, USB 2.0, HDMI, звуковой выход (стерео), звуковой вход для микрофона;
3. Модуль оперативной памяти (ОЗУ) 4GB (2x2GB Dual Chanel) DDR3;
4. Твердотельный диск SSD 256GB;
5. Блок питания с системой охлаждения 450W, активное охлаждение;
6. Корпус системного блока microATX;
7. монитор Размер 23.8", тип матрицы IPS, цифровой интерфейс HDMI, разрешение 1920×1080, без акустической системы;
8. Клавиатура проводная, USB, русская (латинская) раскладка;
9. манипулятор типа «мышь» Проводная, USB, лазерная (оптическая);

Программа должна обеспечивать:

1. вывод меню для работы со списком;
2. добавление нового лекарства в начало списка.
3. добавление нового лекарства в конец списка.
4. добавление нового лекарства, отсортированного по названию в алфавитном порядке.
5. добавление нового лекарства перед указанным.
6. добавление нового лекарства после указанного.
7. удаление данных о лекарствах (при осуществлении покупки);
8. вывод всего списка лекарств;
9. поиск лекарства с максимальным количеством экземпляров в аптеке.
10. по запросу выдаются сведения о наличии лекарств в аптеке, упорядоченные по стране-производителю.
11. вывод лекарств, сгруппированных по типам.

Программа должна обеспечивать диалог с помощью меню и контроль ошибок при вводе.

Во время разработки приложения были использованы:

1. Интегрированная среда разработки Microsoft Visual Studio 2019;
2. Объектно-ориентированный язык программирования C#;
3. Библиотеки: using System, using System.Collections.Generic, using System.Data, using System.Drawing, using System.Linq, using System.Windows.Forms, using System.Threading, using System.IO, using System.Linq, using System.IO;
4. Windows Forms C#;

Листинг кода из файлов с расширением .cs представлен в приложении.

# 

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы разработана информационная

система для аптеки, используя язык программирования C# и

функционал Windows Forms для создания пользовательского интерфейса.

Пользовательский интерфейс разработан с использованием Windows

Forms, обеспечивая удобную навигацию и интуитивно понятные элементы

управления. Интерфейс позволяет легко взаимодействовать с системой,

добавлять, редактировать и удалять данные.

Текстовой файл выбран в качестве хранилища данных, благодаря

своей простоте использования.

Основные преимущества разработанной системы включают:

1. Удобный и интуитивно понятный пользовательский интерфейс, обеспечивающий простоту и эффективность работы с системой.
2. Централизованное хранение и управление информацией о лекарствах облегчающее контроль и доступ к данным.
3. Присутствует интерфейс

В ходе работы также выявлены некоторые возможные области для

дальнейшего улучшения системы. Некоторые из них включают:

1. Расширение функциональности системы
2. Улучшение интерфейса, с учетом современных требований дизайна и удобства использования.
3. Использование более мощной и масштабируемой базы данных, такой как Microsoft SQL Server, для поддержки роста аптеки и увеличения производительности системы.

Цель работы была успешно достигнута — разработана функциональная информационная система, позволяющая управлять текущей информацией лекарств в аптеке и создавать их.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**Научная и учебно-методическая литература**

1. Шилдт, Г. C# 4.0: полное руководство. - Москва: ООО «И. Вильямс», 127055, 2015. - 1056 с.

**Интернет-ресурсы**

1. «Что такое C#: плюсы и минусы языка» (<https://gb.ru/blog/chto-takoe-c/>).
2. «История фармации» ([История фармации (kurskmed.com)](https://kurskmed.com/upload/departments/library/files/bibliography/2019/Istoriya_farmatsii_2019.pdf))
3. «Исторические этапы эволюции систем обеспечения и контроля качества лекарственных средств» ([Исторические этапы эволюции систем обеспечения и контроля качества лекарственных средств (cyberleninka.ru)](https://cyberleninka.ru/article/n/istoricheskie-etapy-evolyutsii-sistem-obespecheniya-i-kontrolya-kachestva-lekarstvennyh-sredstv/viewer))
4. «Информационные технологии в аптеках» ([Информационные технологии в аптеках: история, настоящее и будущее... (itapteka.ru)](https://itapteka.ru/news/novosti/article/avtomatizatsiya-aptek/informatsionnye-tekhnologii-v-aptekakh-istoriya-nastoyashchee-i-budushchee-/))
5. «Информационные технологии и их применение в современной системе здравоохранения» ([Информационные технологии и их применение в современной системе здравоохранения (cyberleninka.ru)](https://cyberleninka.ru/article/n/informatsionnye-tehnologii-i-ih-primenenie-v-sovremennoy-sisteme-zdravoohraneniya/viewer))
6. «Что такое C#: плюсы и минусы языка» (<https://gb.ru/blog/chto-takoe-c/>).

# ПРИЛОЖЕНИЕ А

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Threading;

using System.Windows.Forms;

namespace ScreenAdmin

{

public class ScreenUsesr:ScreenAdmin

{

private System.Windows.Forms.TextBox login;

private System.Windows.Forms.TextBox password;

private System.Windows.Forms.Label labelLogin;

private System.Windows.Forms.Label labelPassword;

private Button linkChet;

private Label label5;

private Label chet;

private Button button1;

private Panel searchElementUs;

private Button SignIn;

private void InitializeComponent()

{

this.password = new System.Windows.Forms.TextBox();

this.login = new System.Windows.Forms.TextBox();

this.labelPassword = new System.Windows.Forms.Label();

this.labelLogin = new System.Windows.Forms.Label();

this.SignIn = new System.Windows.Forms.Button();

this.linkChet = new System.Windows.Forms.Button();

this.label5 = new System.Windows.Forms.Label();

this.chet = new System.Windows.Forms.Label();

this.button1 = new System.Windows.Forms.Button();

this.searchElementUs = new System.Windows.Forms.Panel();

this.sideBar.SuspendLayout();

this.panel2.SuspendLayout();

this.panel1.SuspendLayout();

this.head.SuspendLayout();

this.conteiner.SuspendLayout();

this.SuspendLayout();

//

// sideBar

//

this.sideBar.Controls.Add(this.searchElementUs);

this.sideBar.Controls.Add(this.linkChet);

this.sideBar.Controls.Add(this.SignIn);

this.sideBar.Controls.Add(this.labelLogin);

this.sideBar.Controls.Add(this.labelPassword);

this.sideBar.Controls.Add(this.login);

this.sideBar.Controls.Add(this.password);

this.sideBar.MinimumSize = new System.Drawing.Size(250, 0);

this.sideBar.Size = new System.Drawing.Size(275, 789);

this.sideBar.Controls.SetChildIndex(this.searchBox, 0);

this.sideBar.Controls.SetChildIndex(this.password, 0);

this.sideBar.Controls.SetChildIndex(this.login, 0);

this.sideBar.Controls.SetChildIndex(this.labelPassword, 0);

this.sideBar.Controls.SetChildIndex(this.labelLogin, 0);

this.sideBar.Controls.SetChildIndex(this.SignIn, 0);

this.sideBar.Controls.SetChildIndex(this.panel1, 0);

this.sideBar.Controls.SetChildIndex(this.panel2, 0);

this.sideBar.Controls.SetChildIndex(this.addNew, 0);

this.sideBar.Controls.SetChildIndex(this.sortirovka, 0);

this.sideBar.Controls.SetChildIndex(this.Exit, 0);

this.sideBar.Controls.SetChildIndex(this.linkChet, 0);

this.sideBar.Controls.SetChildIndex(this.searchElementUs, 0);

//

// searchBox

//

this.searchBox.Location = new System.Drawing.Point(30, 30);

this.searchBox.TextChanged += new System.EventHandler(this.Search);

//

// Exit

//

this.Exit.Location = new System.Drawing.Point(18, 641);

this.Exit.Size = new System.Drawing.Size(182, 32);

this.Exit.Text = "Вход";

this.Exit.Visible = false;

//

// sortirovka

//

this.sortirovka.Location = new System.Drawing.Point(38, 382);

//

// addNew

//

this.addNew.Location = new System.Drawing.Point(18, 637);

this.addNew.Size = new System.Drawing.Size(182, 36);

this.addNew.Visible = false;

//

// head

//

this.head.Controls.Add(this.label5);

this.head.Controls.Add(this.chet);

this.head.Controls.Add(this.button1);

this.head.Size = new System.Drawing.Size(1140, 100);

this.head.Controls.SetChildIndex(this.header, 0);

this.head.Controls.SetChildIndex(this.burger, 0);

this.head.Controls.SetChildIndex(this.button1, 0);

this.head.Controls.SetChildIndex(this.chet, 0);

this.head.Controls.SetChildIndex(this.label5, 0);

//

// burger

//

this.burger.Location = new System.Drawing.Point(1065, 25);

//

// header

//

this.header.Location = new System.Drawing.Point(430, 33);

//

// conteiner

//

this.conteiner.Location = new System.Drawing.Point(147, 0);

this.conteiner.Size = new System.Drawing.Size(1140, 789);

//

// password

//

this.password.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.password.Location = new System.Drawing.Point(38, 534);

this.password.Name = "password";

this.password.PasswordChar = 'x';

this.password.Size = new System.Drawing.Size(143, 30);

this.password.TabIndex = 21;

//

// login

//

this.login.Font = new System.Drawing.Font("Microsoft Sans Serif", 15F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.login.Location = new System.Drawing.Point(38, 485);

this.login.Name = "login";

this.login.Size = new System.Drawing.Size(143, 30);

this.login.TabIndex = 22;

//

// labelPassword

//

this.labelPassword.AutoSize = true;

this.labelPassword.Location = new System.Drawing.Point(38, 518);

this.labelPassword.Name = "labelPassword";

this.labelPassword.Size = new System.Drawing.Size(48, 13);

this.labelPassword.TabIndex = 23;

this.labelPassword.Text = "Пароль:";

//

// labelLogin

//

this.labelLogin.AutoSize = true;

this.labelLogin.Location = new System.Drawing.Point(35, 469);

this.labelLogin.Name = "labelLogin";

this.labelLogin.Size = new System.Drawing.Size(41, 13);

this.labelLogin.TabIndex = 24;

this.labelLogin.Tag = "";

this.labelLogin.Text = "Логин:";

//

// SignIn

//

this.SignIn.Font = new System.Drawing.Font("Microsoft Sans Serif", 13F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.SignIn.Location = new System.Drawing.Point(37, 570);

this.SignIn.Name = "SignIn";

this.SignIn.Size = new System.Drawing.Size(144, 37);

this.SignIn.TabIndex = 26;

this.SignIn.Text = "Вход";

this.SignIn.UseVisualStyleBackColor = true;

this.SignIn.Click += new System.EventHandler(this.SignIn\_Click);

//

// linkChet

//

this.linkChet.Location = new System.Drawing.Point(38, 422);

this.linkChet.Name = "linkChet";

this.linkChet.Size = new System.Drawing.Size(116, 23);

this.linkChet.TabIndex = 28;

this.linkChet.Text = "Счет";

this.linkChet.UseVisualStyleBackColor = true;

this.linkChet.Click += new System.EventHandler(this.chet\_Click);

//

// label5

//

this.label5.Location = new System.Drawing.Point(50, 77);

this.label5.Margin = new System.Windows.Forms.Padding(50, 0, 0, 0);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(50, 23);

this.label5.TabIndex = 7;

this.label5.Text = "Баланс:";

this.label5.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

//

// chet

//

this.chet.BackColor = System.Drawing.Color.White;

this.chet.Location = new System.Drawing.Point(100, 77);

this.chet.Margin = new System.Windows.Forms.Padding(50, 0, 0, 0);

this.chet.Name = "chet";

this.chet.Size = new System.Drawing.Size(100, 23);

this.chet.TabIndex = 6;

this.chet.Text = "0";

this.chet.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

//

// button1

//

this.button1.BackColor = System.Drawing.Color.Gold;

this.button1.FlatAppearance.BorderColor = System.Drawing.Color.Green;

this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 13F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.button1.Location = new System.Drawing.Point(200, 75);

this.button1.Margin = new System.Windows.Forms.Padding(50, 0, 0, 0);

this.button1.Name = "button1";

this.button1.Size = new System.Drawing.Size(25, 25);

this.button1.TabIndex = 5;

this.button1.Text = "+";

this.button1.UseVisualStyleBackColor = false;

this.button1.Click += new System.EventHandler(this.chet\_Click);

//

// searchElementUs

//

this.searchElementUs.Location = new System.Drawing.Point(30, 53);

this.searchElementUs.Name = "searchElementUs";

this.searchElementUs.Size = new System.Drawing.Size(116, 10);

this.searchElementUs.TabIndex = 1;

//

// ScreenUsesr

//

this.AutoScaleDimensions = new System.Drawing.SizeF(96F, 96F);

this.ClientSize = new System.Drawing.Size(1433, 750);

this.Name = "ScreenUsesr";

this.Text = "Form";

this.Load += new System.EventHandler(this.BalansRead);

this.sideBar.ResumeLayout(false);

this.sideBar.PerformLayout();

this.panel2.ResumeLayout(false);

this.panel2.PerformLayout();

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

this.head.ResumeLayout(false);

this.head.PerformLayout();

this.conteiner.ResumeLayout(false);

this.ResumeLayout(false);

}

ScreenAdmin a;

public Label Chet { get => chet; set => chet = value; }

public ScreenUsesr()

{

InitializeComponent();

}

public ScreenUsesr(AbautIconUs owner)

{

InitializeComponent();

this.pills.Checked = owner.Pills;

this.potion.Checked = owner.Potion;

this.powders.Checked = owner.Powders;

this.injections.Checked = owner.Injections;

this.allType.Checked = owner.AllType;

this.alphabetically.Checked = owner.Alphabetically;

this.asItIs.Checked = owner.AsItIs;

this.byManufacturer.Checked = owner.ByManufacturer;

this.greatestNumber.Checked = owner.GreatestNumber;

this.chet.Text = Convert.ToString(owner.Balans);

//iconList.Remove(Delete(iconList, owner.Name1));

}

public ScreenUsesr(ScreenAdd a, AbautIconUs owner)

{

InitializeComponent();

this.pills.Checked = owner.Pills;

this.potion.Checked = owner.Potion;

this.powders.Checked = owner.Powders;

this.injections.Checked = owner.Injections;

this.allType.Checked = owner.AllType;

this.alphabetically.Checked = owner.Alphabetically;

this.asItIs.Checked = owner.AsItIs;

this.byManufacturer.Checked = owner.ByManufacturer;

this.greatestNumber.Checked = owner.GreatestNumber;

this.chet.Text = Convert.ToString(owner.Balans);

for (int i = 0; i < iconList.Count; i++)

{

if (iconList.ElementAt(i).name1 == a.name1)

{

iconList.ElementAt(i).inStock1 = a.inStock1;

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Add.txt"));

}

}

}

protected override void CreateIcon(ScreenAdd a, int x, int y)

{

icon = new Panel();

this.nalLabel = new Label();

this.photo = new PictureBox();

this.icon.SuspendLayout();

this.main.Controls.Add(this.icon);

this.icon.Size = new Size(100, 150);

this.icon.BackColor = Color.FromArgb(220, 164, 164);

this.icon.Location = new Point(x, 0);

this.icon.Controls.Add(this.photo);

photo.Size = new Size(80, 60);

photo.Location = new Point(10, 10);

photo.BackColor = Color.Aqua;

photo.SizeMode = PictureBoxSizeMode.Zoom;

photo.Image = new Bitmap($@"{Directory.GetCurrentDirectory()}\Resources\{a.namePhoto}");

nameIcon = new Label();

this.icon.Controls.Add(this.nameIcon);

this.nameIcon.Location = new Point(0, 75);

this.nameIcon.Width = 100;

this.nameIcon.TextAlign = ContentAlignment.MiddleCenter;

nameIcon.Text = a.name1;

nalLabel.Text = a.inStock1 > 0 ? "Есть в наличии" : "Нет в наличии";

this.icon.Controls.Add(this.nalLabel);

nalLabel.Location = new Point(5, 100);

if (nalLabel.Text == "Нет в наличии")

{

this.nalLabel.ForeColor = Color.Red;

this.icon.BackColor = Color.Beige;

}

else

{

this.nalLabel.ForeColor = Color.Green;

this.icon.BackColor = Color.FromArgb(220, 164, 164);

}

int coast = a.coast1;

this.coast = new Label();

this.coast.Text = $"Цена: {coast}";

this.coast.Location = new Point(5, 125);

this.icon.Controls.Add(this.coast);

icon.Click += new System.EventHandler(Icon\_Click);

nameIcon.Click += new System.EventHandler(Icon\_Click);

this.coast.Click += new System.EventHandler(Icon\_Click);

this.nalLabel.Click += new System.EventHandler(Icon\_Click);

photo.Click += new EventHandler(Icon\_Click);

void Icon\_Click(object sender, EventArgs e)

{

AbautIconUs abautIcon = new AbautIconUs(a, this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(abautIcon);

}

}

}

protected override void Search(object sender, EventArgs e)

{

this.y = 0;

this.searchElementUs.Controls.Clear();

LinkedList<ScreenAdd> list = new LinkedList<ScreenAdd>();

string searchIcon = searchBox.Text;

for (int i = 0; i < iconList.Count; i++)//обновляет провереямые элементы

{

int start = 0;

int end = searchIcon.Length;

string iconName = "";

Char[] el = iconList.ElementAt(i).name1.ToCharArray();

if (searchBox.Text == "") break;

for (int s = 0; s < el.Length - searchIcon.Length + 1; s++)//проверяет на соответствие

{

for (int z = start; z < end; z++)

{

iconName = iconName + Convert.ToString(el[z]);

}

if (String.Compare(iconName.ToLower(), searchIcon.ToLower()) == 0)

{

list.AddLast(iconList.ElementAt(i));

break;

}

start++;

end++;

iconName = "";

}

}

list.AddLast(new ScreenAdd());

for (int i = 0; i < list.Count; i++)

{

this.searchElementUs.Size = new Size(116, this.y);

CreateIconSearch(list.ElementAt(i), this.y);

this.y = y + 20;

}

void CreateIconSearch(ScreenAdd a, int y)

{

IconSearch = new Panel();

this.IconSearch.SuspendLayout();

this.searchElementUs.Controls.Add(this.IconSearch);

this.IconSearch.Size = new Size(116, 20);

this.IconSearch.BackColor = Color.Aqua;

this.IconSearch.Location = new Point(0, y);

nameIcon = new Label();

this.IconSearch.Controls.Add(this.nameIcon);

this.nameIcon.Location = new Point(0, 0);

nameIcon.Text = a.name1;

IconSearch.Click += new System.EventHandler(Icon\_Click);

nameIcon.Click += new System.EventHandler(Icon\_Click);

void Icon\_Click(object s, EventArgs w)

{

AbautIconUs abautIconUs = new AbautIconUs(a, this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(abautIconUs);

}

}

}

}

private void SignIn\_Click(object sender, EventArgs e)

{

a = new ScreenAdmin();

if (String.Compare(password.Text, "215") == 0 & String.Compare(login.Text, "TTIiP") == 0)

{

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(a);

}

}

else if(String.Compare(password.Text, "") == 0 || String.Compare(login.Text, "") == 0) MessageBox.Show($"Введите пароль и логин!");

else MessageBox.Show($"Пароль или логин неправильные!");

}

private void chet\_Click(object sender, EventArgs e)

{

Balans balans = new Balans(this);

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Balans.txt"));

balans.ShowDialog();

}

private void BalansRead(object sender, EventArgs e)

{

using (StreamReader sr = new StreamReader($@"{Directory.GetCurrentDirectory()}\txt\Balans.txt"))

{

chet.Text = sr.ReadLine();

}

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

using System.Threading;

using System.IO;

namespace ScreenAdmin

{

public partial class ScreenAdmin : Form

{

protected Thread th;

public static LinkedList<ScreenAdd> iconList = new LinkedList<ScreenAdd>();

public static LinkedList<ScreenAdd> iconPills;

public static LinkedList<ScreenAdd> iconPotion;

public static LinkedList<ScreenAdd> iconInjections;

public static LinkedList<ScreenAdd> iconPowders;

public static LinkedList<ScreenAdd> All;

public static LinkedList<ScreenAdd> sortAlphabetically;

public static LinkedList<ScreenAdd> sortAsItIs;

public static LinkedList<ScreenAdd> sortGreatestNumber;

public static LinkedList<ScreenAdd> sortByManufacturer;

public static LinkedList<ScreenAdd> sort;

ScreenAdd iconDannie;

static int power;

protected int y;

private LinkedList<ScreenAdd> SortAlhabet(bool alphabetically, bool asItIs, bool greatestNumber, bool byManufacturer)

{

LinkedList<ScreenAdd> a = iconList;

if (alphabetically == true)

{

a = new LinkedList<ScreenAdd>(a.OrderBy(x => x.name1));

}

else if (greatestNumber == true)

{

a = new LinkedList<ScreenAdd>(a.OrderBy(x => x.inStock1));//оформит сортировку пузырьком или ставками

return a;

}

else if (byManufacturer == true)

{

a = new LinkedList<ScreenAdd>(a.OrderBy(x => x.producer1));

}

return a;

}

private static LinkedList<ScreenAdd> Filter(LinkedList<ScreenAdd> a, string type)

{

LinkedList<ScreenAdd> iconType = new LinkedList<ScreenAdd>();

for (int i = 0; i < a.Count; i++)

{

if (type == a.ElementAt(i).typeMedication1) iconType.AddLast(a.ElementAt(i));

}

return iconType;

}

public ScreenAdmin()

{

InitializeComponent();

}

public ScreenAdmin(ScreenAdd owner, string placeAdd, string beforeAfter)

{

InitializeComponent();

iconDannie = new ScreenAdd(owner.name.Text, owner.producer.Text, owner.typeMedication.Text, Convert.ToInt32(owner.yearProduction.Text)

, Convert.ToInt32(owner.inStock.Text), Convert.ToInt32(owner.coast.Text),owner.openFile.SafeFileName);

if (placeAdd == "После")

{

iconList.AddAfter(AddBeforeAfter(owner.name.Text, iconList, beforeAfter), iconDannie);

}

else if (placeAdd == "Перед")

{

iconList.AddBefore(AddBeforeAfter(owner.name.Text, iconList, beforeAfter), iconDannie);

}

else if (placeAdd == "Начало")

{

iconList.AddFirst(iconDannie);

}

else if (placeAdd == "Конец")

{

iconList.AddLast(iconDannie);

}

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Add.txt"));

this.pills.Checked = owner.Pills;

this.potion.Checked = owner.Potion;

this.powders.Checked = owner.Powders;

this.injections.Checked = owner.Injections;

this.allType.Checked = owner.AllType;

this.alphabetically.Checked = owner.Alphabetically;

this.asItIs.Checked = owner.AsItIs;

this.byManufacturer.Checked = owner.ByManufacturer;

this.greatestNumber.Checked = owner.GreatestNumber;

}

private LinkedListNode<ScreenAdd> AddBeforeAfter(string name, LinkedList<ScreenAdd> a, string el)

{

LinkedListNode<ScreenAdd> n;

for (int i = 0; i < a.Count; i++)

{

if (a.ElementAt(i).name1 == el) return n = a.Find(a.ElementAt(i));

}

return null;

}

public ScreenAdmin(string name, AbautIcon owner)

{

InitializeComponent();

iconList.Remove(Delete(iconList, name));

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Add.txt"));

this.pills.Checked = owner.Pills;

this.potion.Checked = owner.Potion;

this.powders.Checked = owner.Powders;

this.injections.Checked = owner.Injections;

this.allType.Checked = owner.AllType;

this.alphabetically.Checked = owner.Alphabetically;

this.asItIs.Checked = owner.AsItIs;

this.byManufacturer.Checked = owner.ByManufacturer;

this.greatestNumber.Checked = owner.GreatestNumber;

}

public ScreenAdmin(ScreenAdd a, AbautIcon owner)

{

InitializeComponent();

this.pills.Checked = owner.Pills;

this.potion.Checked = owner.Potion;

this.powders.Checked = owner.Powders;

this.injections.Checked = owner.Injections;

this.allType.Checked = owner.AllType;

this.alphabetically.Checked = owner.Alphabetically;

this.asItIs.Checked = owner.AsItIs;

this.byManufacturer.Checked = owner.ByManufacturer;

this.greatestNumber.Checked = owner.GreatestNumber;

for (int i = 0; i < iconList.Count; i++)

{

if (iconList.ElementAt(i).name1 == a.name1)

{

iconList.ElementAt(i).inStock1 = a.inStock1;

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Add.txt"));

}

}

}

public ScreenAdmin(ScreenAdmin owner)

{

InitializeComponent();

this.pills.Checked = owner.Pills.Checked;

this.potion.Checked = owner.Potion.Checked;

this.powders.Checked = owner.Powders.Checked;

this.injections.Checked = owner.Injections.Checked;

this.allType.Checked = owner.AllType.Checked;

this.alphabetically.Checked = owner.Alphabetically.Checked;

this.asItIs.Checked = owner.AsItIs.Checked;

this.byManufacturer.Checked = owner.ByManufacturer.Checked;

this.greatestNumber.Checked = owner.GreatestNumber.Checked;

}

public ScreenAdmin(ScreenAdd owner)

{

InitializeComponent();

this.pills.Checked = owner.Pills;

this.potion.Checked = owner.Potion;

this.powders.Checked = owner.Powders;

this.injections.Checked = owner.Injections;

this.allType.Checked = owner.AllType;

this.alphabetically.Checked = owner.Alphabetically;

this.asItIs.Checked = owner.AsItIs;

this.byManufacturer.Checked = owner.ByManufacturer;

this.greatestNumber.Checked = owner.GreatestNumber;

}

protected ScreenAdd Delete(LinkedList<ScreenAdd> a, string name)

{

for (int i = 0; i < a.Count; i++)

{

if (a.ElementAt(i).name1 == name)

{

return a.ElementAt(i);

}

}

return null;

}

public void Form1\_Load(object sender, EventArgs e)

{

this.main.Padding = new Padding(30);

this.sideBar.Size = new Size(0, 0);

ScreenAdmin\_SizeChanged(this, EventArgs.Empty);

LinkedList<ScreenAdd> f = new LinkedList<ScreenAdd>();

using (StreamReader sr = new StreamReader($@"{Directory.GetCurrentDirectory()}\txt\Add.txt"))

{

string line;

while ((line = sr.ReadLine()) != null)

{

f.AddLast(SplAdd(line));

}

}

iconList = f;

sort = SortAlhabet(alphabetically.Checked, asItIs.Checked, greatestNumber.Checked, byManufacturer.Checked);

int x = 50;

int z = 150;

if (this.pills.Checked == true)

{

iconPills = Filter(sort, "Таблетки");

for (int i = 0; i < iconPills.Count; i++)

{

CreateIcon(iconPills.ElementAt(i), x, 50);

x += z;

}

}

else if (powders.Checked == true)

{

iconPowders = Filter(sort, "Порошки");

for (int i = 0; i < iconPowders.Count; i++)

{

CreateIcon(iconPowders.ElementAt(i), x, 50);

x += z;

}

}

else if (allType.Checked == true)

{

All = sort;

for (int i = 0; i < All.Count; i++)

{

CreateIcon(All.ElementAt(i), x, 50);

x += z;

}

}

else if (injections.Checked == true)

{

iconInjections = Filter(sort, "Уколы");

for (int i = 0; i < iconInjections.Count; i++)

{

CreateIcon(iconInjections.ElementAt(i), x, 50);

x += z;

}

}

else if (potion.Checked == true)

{

iconPotion = Filter(iconList, "Микстура");

for (int i = 0; i < iconPotion.Count; i++)

{

CreateIcon(iconPotion.ElementAt(i), x, 50);

x += z;

}

}

this.main.Controls.Add(this.planX);

}

private void ScreenAdmin\_SizeChanged(object sender, EventArgs e)

{

header.Location = new Point((head.Width - header.Width) / 2, (head.Height - header.Height) / 2);

this.conteiner.Size = new Size(this.Width, this.Height);

this.sideBar.Location = new Point(0, 0);

main.Location = new Point(0, this.conteiner.Height - (this.conteiner.Height - this.head.Height));

main.Size = new Size(conteiner.Width, this.conteiner.Height - head.Height -39);

if (power == 1) sideBar.Size = new Size(250, this.Height);

burger.Location = new Point(head.Width - 75, (head.Height - burger.Height) / 2);

this.conteiner.Location = new Point((this.Width - this.conteiner.Width - 15) / 2, 0);

this.planX.Size = new Size(main.Width - 100,50);

Width768();

Width425X768();

void Width425X768()

{

if (this.conteiner.Width <= 768 & this.conteiner.Width >= 425)

{

burger.Visible = true;

}

}

void Width768()

{

if (this.conteiner.Width > 768)

{

burger.Visible = false;

int onePesentWidth = this.conteiner.Width / 100;

int onePesentHeight = this.conteiner.Height / 100;

this.sideBar.Size = new Size(onePesentWidth \* 25, this.conteiner.Height);

this.sideBar.Location = new Point(0, this.conteiner.Height - (this.conteiner.Height - this.head.Height));

this.main.Size = new Size(this.conteiner.Width - this.sideBar.Width, this.conteiner.Height - head.Height);

this.main.Location = new Point(this.conteiner.Width - main.Width, this.conteiner.Height - main.Height);

this.planX.Size = new Size(main.Width - 100, 70);

}

}

}

private void burger\_Click(object sender, EventArgs e)

{

if (power == 0)

{

this.sideBar.Location = new Point(0, 0);

sideBar.Size = new Size(250, this.Height);

sideBar.BringToFront();

power++;

}

else

{

sideBar.Size = new Size(0, 0);

sideBar.SendToBack();

power--;

}

}

private void button1\_Click(object sender, EventArgs e)

{

ScreenAdd screenAdd = new ScreenAdd();

screenAdd.Show();

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(new ScreenAdd(this));

}

}

protected virtual void CreateIcon(ScreenAdd a, int x, int y)

{

icon = new Panel();

this.nalLabel = new Label();

this.photo = new PictureBox();

this.icon.SuspendLayout();

this.main.Controls.Add(this.icon);

this.icon.Size = new Size(100, 150);

this.icon.BackColor = Color.FromArgb(220, 164, 164);

this.icon.Location = new Point(x, 0);

this.icon.Controls.Add(this.photo);

photo.Size = new Size(80,60);

photo.Location = new Point(10,10);

photo.BackColor = Color.Aqua;

photo.SizeMode = PictureBoxSizeMode.Zoom;

photo.Image = new Bitmap($@"{Directory.GetCurrentDirectory()}\Resources\{a.namePhoto}");

nameIcon = new Label();

this.icon.Controls.Add(this.nameIcon);

this.nameIcon.Location = new Point(0, 75);

this.nameIcon.Width = 100;

this.nameIcon.TextAlign = ContentAlignment.MiddleCenter;

nameIcon.Text = a.name1;

nalLabel.Text = a.inStock1 > 0 ? "Есть в наличии" : "Нет в наличии";

this.icon.Controls.Add(this.nalLabel);

nalLabel.Location = new Point(5, 100);

if (nalLabel.Text == "Нет в наличии")

{

this.nalLabel.ForeColor = Color.Red;

this.icon.BackColor = Color.Beige;

}

else

{

this.nalLabel.ForeColor = Color.Green;

this.icon.BackColor = Color.FromArgb(220, 164, 164);

}

int coast = a.coast1;

this.coast = new Label();

this.coast.Text = $"Цена: {coast}";

this.coast.Location = new Point(5, 125);

this.icon.Controls.Add(this.coast);

icon.Click += new System.EventHandler(Icon\_Click);

nameIcon.Click += new System.EventHandler(Icon\_Click);

this.coast.Click += new System.EventHandler(Icon\_Click);

this.nalLabel.Click += new System.EventHandler(Icon\_Click);

photo.Click += new EventHandler(Icon\_Click);

void Icon\_Click(object sender, EventArgs e)

{

AbautIcon abautIcon = new AbautIcon(a, this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(abautIcon);

}

}

}

protected void ButtonSort\_Click(object sender, EventArgs e)

{

this.main.Controls.Clear();

Form1\_Load(this, EventArgs.Empty);

}

protected virtual void Search(object sender, EventArgs e)

{

this.y = 0;

this.searchElement.Controls.Clear();

LinkedList<ScreenAdd> list = new LinkedList<ScreenAdd>();

string searchIcon = searchBox.Text;

for (int i = 0; i < iconList.Count; i++)//обновляет провереямые элементы

{

int start = 0;

int end = searchIcon.Length;

string iconName = "";

Char[] el = iconList.ElementAt(i).name1.ToCharArray();

if (searchBox.Text == "") break;

for (int s = 0; s < el.Length - searchIcon.Length + 1; s++)//проверяет на соответствие

{

for (int z = start; z < end; z++)

{

iconName = iconName + Convert.ToString(el[z]);

}

if (String.Compare(iconName.ToLower(), searchIcon.ToLower()) == 0)

{

list.AddLast(iconList.ElementAt(i));

break;

}

start++;

end++;

iconName = "";

}

}

list.AddLast(new ScreenAdd());

for (int i = 0; i < list.Count; i++)

{

this.searchElement.Size = new Size(116, this.y);

CreateIconSearch(list.ElementAt(i), this.y);

this.y = y + 20;

}

void CreateIconSearch(ScreenAdd a, int y)

{

iconSearch = new Panel();

this.iconSearch.SuspendLayout();

this.searchElement.Controls.Add(this.iconSearch);

this.iconSearch.Size = new Size(116, 20);

this.iconSearch.BackColor = Color.Aqua;

this.iconSearch.Location = new Point(0, y);

nameIcon = new Label();

this.iconSearch.Controls.Add(this.nameIcon);

this.nameIcon.Location = new Point(0, 0);

nameIcon.Text = a.name1;

iconSearch.Click += new System.EventHandler(Icon\_Click);

nameIcon.Click += new System.EventHandler(Icon\_Click);

void Icon\_Click(object s, EventArgs w)

{

AbautIcon abautIcon = new AbautIcon(a, this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(abautIcon);

}

}

}

}

protected void Exit\_Click(object sender, EventArgs e)

{

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

void Open(object obj)

{

Application.Run(new ScreenUsesr());

}

}

protected virtual void Save(StreamWriter z)

{

using ( z)

{

foreach (ScreenAdd el in iconList)

{

z.WriteLine($"{el.name1};{el.producer1};{el.typeMedication1};{el.yearProduction1};{el.inStock1};{el.coast1};{el.namePhoto}");

}

}

}

protected virtual ScreenAdd SplAdd(string spl)

{

string[] e = spl.Split(Convert.ToChar(";"));

return new ScreenAdd(e[0], e[1], e[2], Convert.ToInt32(e[3]), Convert.ToInt32(e[4]), Convert.ToInt32(e[5]), e[6]);

}

}

}

using System;

using System.Drawing;

using System.Linq;

using System.Threading;

using System.Windows.Forms;

using System.IO;

namespace ScreenAdmin

{

public partial class ScreenAdd : Form

{

public string name1;

public string producer1;

public string typeMedication1;

public int yearProduction1;

public int inStock1;

public int coast1;

public string namePhoto;

bool pills;

bool potion;

bool powders;

bool injections;

bool allType;

bool alphabetically;

bool asItIs;

bool byManufacturer;

bool greatestNumber;

public OpenFileDialog openFile;

ScreenAdmin screenAdmin;

Thread th;

string nal;

public bool Pills { get => pills; set => pills = value; }

public bool Potion { get => potion; set => potion = value; }

public bool Powders { get => powders; set => powders = value; }

public bool Injections { get => injections; set => injections = value; }

public bool AllType { get => allType; set => allType = value; }

public bool Alphabetically { get => alphabetically; set => alphabetically = value; }

public bool AsItIs { get => asItIs; set => asItIs = value; }

public bool ByManufacturer { get => byManufacturer; set => byManufacturer = value; }

public bool GreatestNumber { get => greatestNumber; set => greatestNumber = value; }

public string Nal { get => nal; set => nal = value; }

public ScreenAdd()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

screenAdmin = new ScreenAdmin(this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

}

private void Open(object obj)

{

Application.Run(screenAdmin);

}

public ScreenAdd(ScreenAdmin z)

{

InitializeComponent();

this.pills = z.Pills.Checked;

this.potion = z.Potion.Checked;

this.powders = z.Powders.Checked;

this.injections = z.Injections.Checked;

this.allType = z.AllType.Checked;

this.byManufacturer = z.ByManufacturer.Checked;

this.alphabetically = z.Alphabetically.Checked;

this.asItIs = z.AsItIs.Checked;

this.greatestNumber = z.GreatestNumber.Checked;

}

public ScreenAdd(string name, string producer, string typeMedication, int yearProduction, int inStock, int coast, string namePhoto)

{

this.name1 = name;

this.producer1 = producer;

this.typeMedication1 = typeMedication;

this.yearProduction1 = yearProduction;

this.inStock1 = inStock;

this.coast1 = coast;

this.namePhoto = namePhoto;

}

public void OpenAdmin()

{

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

}

public int Chek()

{

int a = 0;

char[] chek;

string[] pols = new string[] { typeMedication.Text, yearProduction.Text, inStock.Text, coast.Text };

string[] number = new string[] { yearProduction.Text, inStock.Text, coast.Text };

for (int i = 0; i < pols.Length; i++)

{

try

{

chek = pols[i].ToArray();

for (int z = 0; z < chek.Length; z++)

{

string el = Convert.ToString(chek[z]);

if (String.Compare(el, " ") == 0 || String.Compare(el, null) == 0) return 0;

}

if (i <= 2) a = Convert.ToInt32(number[i]) <= 0 ? a = Convert.ToInt32("") : a = Convert.ToInt32(number[i]);

}

catch (FormatException)

{

return 1;

}

}

return -1;

}

private void button3\_Click(object sender, EventArgs e)

{

try{

photo.Image.Save($@"{Directory.GetCurrentDirectory()}\Resources\{openFile.SafeFileName}");

if (Chek() == 0)

{

MessageBox.Show($"Вы незаполнили все поля \n или в них есть пробелы");

}

else if (Chek() == 1)

{

MessageBox.Show($"В местах , где необходимо написать цифры\nнеизвестные символы или пробелы");

}

else if (before.Checked == true)

{

screenAdmin = new ScreenAdmin(this, placeAdd(), textBoxBefore.Text);

OpenAdmin();

}

else if (after.Checked == true)

{

screenAdmin = new ScreenAdmin(this, placeAdd(), textBoxAfter.Text);

OpenAdmin();

}

else

{

screenAdmin = new ScreenAdmin(this, placeAdd(), "");

OpenAdmin();

}

}

catch

{

MessageBox.Show("Вы не добавили фотографию");

}

}

public string placeAdd()

{

if (start.Checked == true)

{

return start.Text;

}

else if (before.Checked == true)

{

textBoxBefore.Visible = true;

return before.Text;

}

else if (after.Checked == true)

{

textBoxAfter.Visible = true;

return after.Text;

}

return end.Text;

}

private void ScreenAdd\_Load(object sender, EventArgs e)

{

end.Checked = true;

}

private void before\_Click(object sender, EventArgs e)

{

placeAdd();

if (after.Checked == false)

{

textBoxAfter.Text = "";

textBoxAfter.Visible = false;

}

if (before.Checked == false)

{

textBoxBefore.Text = "";

textBoxBefore.Visible = false;

}

}

private void addPhoto\_Click(object sender, EventArgs e)

{

openFile = new OpenFileDialog();

openFile.Filter = "Image files(\* JPG, \* EMP, \* .GIF, \* .PNG) | \* .JPG; \*.EMP;\*.GIF;\*.PNG | All files(\*.\*) | \*.\*";

if (openFile.ShowDialog() == DialogResult.OK)

try

{

this.photo.Image = new Bitmap(openFile.FileName);

}

catch

{

MessageBox.Show("Фото неверного формата");

}

}

}

}

using System;

using System.IO;

using System.Windows.Forms;

namespace ScreenAdmin

{

public partial class Balans : Form

{

ScreenUsesr screenUsesr;

public Balans(ScreenUsesr a)

{

InitializeComponent();

this.screenUsesr = a;

}

private void deposit\_Click(object sender, EventArgs e)

{

if(-1 == Chek(textBoxDeposit.Text))

{

this.chet.Text = Convert.ToString(Convert.ToInt32(textBoxDeposit.Text) + Convert.ToInt32(this.chet.Text));

screenUsesr.Chet.Text = this.chet.Text;

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Balans.txt"));

textBoxDeposit.Text = "0";

}

else if (0 == Chek(textBoxDeposit.Text))

{

MessageBox.Show("Вы ничего не ввели!");

}

else if (1 == Chek(textBoxDeposit.Text))

{

MessageBox.Show("Введите число!");

textBoxDeposit.Text = "0";

}

}

public int Chek(string txt)

{

if (String.Compare(txt,"") == 0) return 0;

try

{

int a = Convert.ToInt32(txt) == 0 ? a = Convert.ToInt32("yg") : a = Convert.ToInt32(txt);

}

catch (FormatException)

{

return 1;

}

finally

{

}

return -1;

}

private void withdraw\_Click(object sender, EventArgs e)

{

if (-1 == Chek(textBoxWithdraw.Text))

{

if ( Convert.ToInt32(chet.Text) - Convert.ToInt32(textBoxWithdraw.Text) >= 0)

{

this.chet.Text = Convert.ToString(Convert.ToInt32(this.chet.Text) - Convert.ToInt32(textBoxWithdraw.Text));

screenUsesr.Chet.Text = this.chet.Text;

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\Balans.txt"));

textBoxWithdraw.Text = "0";

} else MessageBox.Show("Недостаточно средств!");

}

else if (0 == Chek(textBoxWithdraw.Text))

{

MessageBox.Show("Вы ничего не ввели!");

}

else if (1 == Chek(textBoxWithdraw.Text))

{

MessageBox.Show("Введите число!");

textBoxWithdraw.Text = "0";

}

}

private void Balans\_Load(object sender, EventArgs e)

{

this.chet.Text = screenUsesr.Chet.Text;

}

protected void Save(StreamWriter z)

{

using (z)

{

z.WriteLine($"{chet.Text}");

}

}

}

}

using System;

using System.Windows.Forms;

namespace ScreenAdmin

{

public partial class AddUnits : Form

{

AbautIcon a;

public AddUnits()

{

InitializeComponent();

}

public AddUnits(AbautIcon owner)

{

InitializeComponent();

a = owner;

}

private void done\_Click(object sender, EventArgs e)

{

if (0< Convert.ToInt32(textBox1.Text))

{

try

{

int chek = Convert.ToInt32(textBox1.Text);

a.InStock.Text = textBox1.Text;

this.Close();

}

catch

{

MessageBox.Show("Не является числом");

}

}

else MessageBox.Show("Колличество товара не может быть отрицательным");

}

}

}

using System;

using System.IO;

using System.Threading;

using System.Windows.Forms;

namespace ScreenAdmin

{

public class AbautIconUs:AbautIcon

{

private System.Windows.Forms.Button Buy;

private void InitializeComponent()

{

this.Buy = new System.Windows.Forms.Button();

this.back1 = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// addUnits

//

this.addUnits.Location = new System.Drawing.Point(186, 415);

this.addUnits.Visible = false;

//

// back

//

this.back.Location = new System.Drawing.Point(326, 415);

this.back.Visible = false;

//

// delete

//

this.delete.Location = new System.Drawing.Point(46, 415);

this.delete.Visible = false;

//

// Buy

//

this.Buy.Location = new System.Drawing.Point(454, 334);

this.Buy.Name = "Buy";

this.Buy.Size = new System.Drawing.Size(145, 30);

this.Buy.TabIndex = 14;

this.Buy.Text = "Купить";

this.Buy.UseVisualStyleBackColor = true;

this.Buy.Click += new System.EventHandler(this.Buy\_Click);

//

// back1

//

this.back1.Location = new System.Drawing.Point(13, 13);

this.back1.Name = "back1";

this.back1.Size = new System.Drawing.Size(75, 23);

this.back1.TabIndex = 15;

this.back1.Text = "назад";

this.back1.UseVisualStyleBackColor = true;

this.back1.Click += new System.EventHandler(this.back\_Click);

//

// AbautIconUs

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.back1);

this.Controls.Add(this.Buy);

this.Name = "AbautIconUs";

this.Controls.SetChildIndex(this.delete, 0);

this.Controls.SetChildIndex(this.addUnits, 0);

this.Controls.SetChildIndex(this.back, 0);

this.Controls.SetChildIndex(this.Buy, 0);

this.Controls.SetChildIndex(this.back1, 0);

this.ResumeLayout(false);

this.PerformLayout();

}

ScreenUsesr screenUsesr;

private Button back1;

Thread th;

int balans;

public int Balans { get => balans; set => balans = value; }

public AbautIconUs(ScreenAdd a, ScreenUsesr z)

{

InitializeComponent();

this.screenUsesr = z;

this.pills = z.Pills.Checked;

this.potion = z.Potion.Checked;

this.powders = z.Powders.Checked;

this.injections = z.Injections.Checked;

this.allType = z.AllType.Checked;

this.alphabetically = z.Alphabetically.Checked;

this.asItIs = z.AsItIs.Checked;

this.byManufacturer = z.ByManufacturer.Checked;

this.greatestNumber = z.GreatestNumber.Checked;

this.balans = Convert.ToInt32(z.Chet.Text);

this.sc = a;

this.Name1 = a.name1;

this.producer1 = a.producer1;

this.typeMedication1 = a.typeMedication1;

this.yearProduction1 = a.yearProduction1;

this.inStock1 = a.inStock1;

this.coast1 = a.coast1;

this.namePhoto = a.namePhoto;

}

private void Buy\_Click(object sender, EventArgs e)

{

if (balans >= coast1 & inStock1 > 0)

{

this.inStock1--;

InStock.Text = Convert.ToString(inStock1);

balans -= coast1;

Save(new StreamWriter($@"{Directory.GetCurrentDirectory()}\txt\\Balans.txt"));

}

else if (inStock1 <= 0)

{

MessageBox.Show("Товар закончился!");

}

else MessageBox.Show("недостаточно средств!");

}

protected override void back\_Click(object sender, EventArgs e)

{

sc = new ScreenAdd(Name1, producer1, typeMedication1, yearProduction1,inStock1,coast1, namePhoto);

screenUsesr = new ScreenUsesr(sc, this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

}

private void Open()

{

Application.Run(screenUsesr);

}

protected void Save(StreamWriter z)

{

using (z)

{

z.WriteLine($"{balans}");

}

}

}

}

using System;

using System.Drawing;

using System.IO;

using System.Threading;

using System.Windows.Forms;

namespace ScreenAdmin

{

public partial class AbautIcon : Form

{

ScreenAdmin screenAdmin;

Thread th;

private string name1;

protected string producer1;

protected string typeMedication1;

protected int yearProduction1;

protected int inStock1;

protected int coast1;

protected string namePhoto;

protected ScreenAdd sc;

AddUnits addUnit;

ScreenAdmin owner;

protected bool pills;

protected bool potion;

protected bool powders;

protected bool injections;

protected bool allType;

protected bool alphabetically;

protected bool asItIs;

protected bool byManufacturer;

protected bool greatestNumber;

public int InStock1

{

get => inStock1;

set

{

if (value < 0) inStock1 = 0;

else inStock1 = value;

}

}

public bool Pills { get => pills; set => pills = value; }

public bool Potion { get => potion; set => potion = value; }

public bool Powders { get => powders; set => powders = value; }

public bool Injections { get => injections; set => injections = value; }

public bool AllType { get => allType; set => allType = value; }

public bool Alphabetically { get => alphabetically; set => alphabetically = value; }

public bool AsItIs { get => asItIs; set => asItIs = value; }

public bool ByManufacturer { get => byManufacturer; set => byManufacturer = value; }

public bool GreatestNumber { get => greatestNumber; set => greatestNumber = value; }

public string Name1 { get => name1; set => name1 = value; }

public string NamePhoto { get => namePhoto; set => namePhoto = value; }

public AbautIcon()

{

InitializeComponent();

}

public AbautIcon(ScreenAdd a,ScreenAdmin z)

{

InitializeComponent();

this.owner = z;

this.pills = z.Pills.Checked;

this.potion = z.Potion.Checked;

this.powders = z.Powders.Checked;

this.injections = z.Injections.Checked;

this.allType = z.AllType.Checked;

this.alphabetically = z.Alphabetically.Checked;

this.asItIs = z.AsItIs.Checked;

this.byManufacturer = z.ByManufacturer.Checked;

this.greatestNumber = z.GreatestNumber.Checked;

this.sc = a;

this.name1 = a.name1;

this.producer1 = a.producer1;

this.typeMedication1 = a.typeMedication1;

this.yearProduction1 = a.yearProduction1;

this.inStock1 = a.inStock1;

this.coast1 = a.coast1;

this.namePhoto = a.namePhoto;

}

protected virtual void back\_Click(object sender, EventArgs e)

{

sc = new ScreenAdd(name1, producer1, typeMedication1, yearProduction1,Convert.ToInt32(inStock.Text),coast1,namePhoto);

screenAdmin = new ScreenAdmin(sc,this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

}

protected virtual void Open()

{

Application.Run(screenAdmin);

}

protected void AbautIcon\_Load(object sender, EventArgs e)

{

name.Text = name1;

manufacturer.Text = producer1;

typeMedicine.Text = typeMedication1;

productionYear.Text = Convert.ToString(yearProduction1);

inStock.Text = Convert.ToString(this.inStock1);

coast.Text = Convert.ToString(coast1);

photo.Image = new Bitmap($@"{Directory.GetCurrentDirectory()}\Resources\{namePhoto}");

}

private void delete\_Click(object sender, EventArgs e)

{

screenAdmin = new ScreenAdmin(name1,this);

this.Close();

th = new Thread(Open);

th.SetApartmentState(ApartmentState.STA);

th.Start();

}

private void addUnits\_Click(object sender, EventArgs e)

{

new AddUnits(this).ShowDialog();

}

}

}