|  |  |
| --- | --- |
| **КГЭУ** | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  **Федеральное государственное бюджетное образовательное**  **учреждение высшего образования**  **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»** |

Институт: Институт цифровых технологий и экономики

Кафедра: Информационные технологии и интеллектуальные системы

**О Т Ч Е Т**

**По учебной практике**

**(ознакомительной)**

Сапожникова Ильи Алексеевича

обучающегося в группе ТРП-3-22

по образовательной программе направления подготовки

09.03.01 Информатика и вычислительная техника

направленность (профиль) программы

Технологии разработки программного обеспечения

ОТЧЕТ ПРОВЕРИЛ

Руководитель практики:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_Соложенцева Р.С.\_\_\_\_\_

«15» июня 2024 г.

Руководитель практики от Профильной организации:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Натальсон А. В.\_\_\_\_\_\_\_\_

ОЦЕНКА при защите отчета:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«15» июня 2024 г.

Казань, 2024 г.

Оглавление

[Введение 3](#_Toc168548583)

[1. Теоретические основания разработки программного обеспечения.4](#_Toc168548584)

[1.1. Сравнительный обзор инструментов языка программирования python……… 4](#_Toc168548585)

[1.2. Выбор технологии реализации программного обеспечения. 6](#_Toc168548586)

[2. Практическая реализация программного обеспечения 11](#_Toc168548587)

[2.1. Проектирование алгоритма программного обеспечения средствами языка программирования python. 11](#_Toc168548588)

[2.2. Реализация программного обеспечения «Заметки». Обоснование его применения в профессиональной деятельности 14](#_Toc168548589)

[Заключение 20](#_Toc168548590)

[Список использованных источников 21](#_Toc168548591)

[Приложение. Реализация программного обеспечения «Заметки» 22](#_Toc168548592)

# Введение

Учебная практика (ознакомительная) направлена на закрепление и углубление теоретической подготовки обучающегося, приобретение им практических навыков и компетенций, а также опыта самостоятельной работы в выбранной профессиональной деятельности.

Целью освоение учебной практики является получение первичных профессиональных умений и навыков.

Прохождение учебной практики предполагает выполнение следующих задач:

− закрепление теоретических знаний и умений, приобретенных в предшествующий период теоретического обучения;

− овладение профессиональными навыками решения практических задач;

− развитие первичных профессиональных умений по направлению и

профилю подготовки;

# Теоретические основания разработки программного обеспечения.

## Сравнительный обзор инструментов языка программирования python

**Pycharm**

PyCharm - это мощная интегрированная среда разработки (IDE), специально разработанная для программистов на языке Python. Она предоставляет все необходимые инструменты для написания, редактирования, отладки и тестирования кода в одном месте. PyCharm предлагает продвинутые функции, которые значительно упрощают процесс разработки программного обеспечения.

IDE предлагает широкие возможности для разработчика:

* Поддержка только необходимых плагинов и настроек для оптимальной работы;
* Поддержка различных языков программирования, таких как Python, JavaScript, CoffeeScript, TypeScript и другие;
* Возможность разработки веб-приложений с использованием Django, Flask и других фреймворков;
* Возможность интеграции с системами контроля версий, такими как Git, Mercurial, SVN.

Интегрированная среда разработки предоставляет множество функций, упрощающих написание кода, таких как: интеллектуальное автодополнение, статический анализ кода, рефакторинг, навигация по коду и интегрированный отладчик. PyCharm также поддерживает виртуальные среды, что позволяет изолировать проекты друг от друга и управлять зависимостями. Кроме того, IDE позволяет настроить внешний вид и поведение интерфейса под свои предпочтения.

PyCharm предлагает инструменты для тестирования, такие как интеграция с популярными библиотеками тестирования (pytest, unittest), и средства для быстрого выполнения и проверки тестов. Это позволяет разработчикам уверенно писать качественный код и легко находить и исправлять ошибки.

**Sublime Text**

Sublime Text - это легкий, но мощный текстовый редактор, который разработан для быстрой и удобной работы с кодом. Хотя это не полноценная интегрированная среда разработки, он предлагает множество функций, которые делают его популярным среди программистов для написания и редактирования кода.

Редактор предлагает широкие возможности для разработчика:

* Минималистичный и настраиваемый интерфейс, который позволяет сосредоточиться на коде;
* Поддержка множества языков программирования через плагины и пакеты;
* Возможность расширения функциональности с помощью пакетов, устанавливаемых через Package Control;
* Быстрая навигация по коду и проектам с помощью командной палитры и мини-карты кода.

Sublime Text предоставляет множество функций, упрощающих написание кода, таких как: множественное редактирование (мультикурсор), мощная система сниппетов, синтаксическое подсвечивание и сворачивание кода. Редактор также поддерживает Goto Anything, позволяющую мгновенно перемещаться к файлам, символам или строкам в проекте.

Кроме того, Sublime Text позволяет пользователям персонализировать свои рабочие пространства и настройки редактора, чтобы максимально соответствовать их потребностям. Несмотря на то, что в Sublime Text отсутствует встроенный отладчик и другие инструменты полноценной IDE, его легкость и гибкость делают его отличным выбором для быстрого редактирования и прототипирования кода.

**Visual Studio**

Visual Studio - это мощное средство разработчика, которое можно использовать для выполнения всего цикла разработки в одном месте. Это комплексная интегрированная среда разработки (IDE), которую можно использовать для записи, редактирования, отладки и сборки кода, а затем развертывания приложения. Помимо редактирования и отладки кода Visual Studio включает компиляторы, средства завершения кода, управление версиями, расширения и многое другое, чтобы улучшить каждый этап процесса разработки программного обеспечения.

IDE предлагает широкие возможности для разработчика:

- Предоставляет возможность установить только то, что нужно разработчику;

- поддержка нескольких языков программирования, такие как c#, c++, Python, JavaScript и т.д.;

- Возможность создавать приложения для любой платформы (кроссплатформенность);

- Возможность совместной работы над кодом;

Интегрированная среда разработки предоставляет множество функций, упрощающих написание кода, такие как: лампочки, предлагающие действия для улучшения кода, структурирование, позволяющее свернуть или развернуть большую часть кода, обозреватель решений, который включает код, упорядоченный по файлам, что позволяет легко ориентироваться в написанном коде, также Visual Studio позволяет персонализировать среду разработки, что позволяет пользователю отредактировать внешний вид IDE как ему угодно.

Visual Studio предоставляет возможность сразу создавать сборки приложения и тестировать их, также в нем присутствует отладчик, что упрощает написание кода.

## Выбор технологии реализации программного обеспечения.

Python - это мощный, интерпретируемый, объектно-ориентированный язык программирования с открытым исходным кодом, который отличается простотой в изучении и использовании. Он широко используется для разработки разнообразных приложений и занимает высокие позиции в рейтингах популярности языков программирования.

**Достоинства языка для разработки:**

**Простота и читаемость кода.** Python известен своей лаконичностью и понятностью. Синтаксис языка позволяет писать четкий и легко читаемый код, что способствует повышению продуктивности разработчиков. Это особенно важно в больших командах, где разработчики должны понимать и поддерживать код друг друга.

**Высокий уровень абстракции.** Python позволяет быстро и эффективно решать сложные задачи за счет высокого уровня абстракции. Встроенные высокоуровневые структуры данных и мощные библиотеки позволяют сосредоточиться на логике приложения, а не на деталях реализации.

**Большая стандартная библиотека.** Python имеет обширную стандартную библиотеку, которая включает множество модулей и пакетов для выполнения различных задач, таких как работа с файлами, сетевое программирование, обработка текста, управление процессами и многое другое.

**Кроссплатформенность.** Программы, написанные на Python, могут быть выполнены на различных платформах без изменений в коде. Это упрощает разработку и тестирование приложений для разных операционных систем.

**Поддержка различных парадигм программирования.** Python поддерживает несколько парадигм программирования, включая объектно-ориентированное, процедурное и функциональное программирование. Это дает разработчикам гибкость в выборе подхода к решению задач.

**Активное сообщество и богатая экосистема.** Python имеет большое и активное сообщество, что способствует быстрому решению проблем и развитию языка. Существуют тысячи библиотек и фреймворков, которые можно использовать для разработки приложений в самых разных областях, от веб-разработки до анализа данных и машинного обучения.

**Недостатки языка для разработки:**

**Скорость выполнения.** Поскольку Python является интерпретируемым языком, он может быть медленнее по сравнению с компилируемыми языками, такими как C или C++. Это может быть критично для задач, требующих высокой производительности, таких как численные вычисления или игры.

**Потребление памяти.** Из-за высокой абстракции и динамической типизации Python может потреблять больше памяти по сравнению с языками низкого уровня. Это может быть важным фактором для приложений с ограниченными ресурсами.

**Проблемы с многопоточностью.** Встроенный механизм Global Interpreter Lock (GIL) ограничивает выполнение нескольких потоков одновременно в одном процессе Python, что может быть препятствием для многопоточных программ. Однако, это можно обойти с помощью многопроцессорности или использования асинхронных библиотек.

**Зависимость от интерпретатора.** Программы на Python требуют наличия интерпретатора для выполнения, что может вызывать сложности при развертывании приложений на платформах, где Python не установлен по умолчанию.

**Безопасность кода.** Как и в случае с другими интерпретируемыми языками, исходный код на Python легко читаем, что может представлять угрозу безопасности. Код может быть декомпилирован, и злоумышленники могут изучить его для поиска уязвимостей.

В целом, Python является отличным выбором для разработки множества типов приложений благодаря своей простоте, универсальности и мощным встроенным инструментам.

**Сферы применения языка программирования Python**

* Веб-разработка

Python широко используется для создания веб-приложений. Фреймворки, такие как Django и Flask, предоставляют мощные инструменты и библиотеки для разработки серверной логики, управления базами данных и создания интерфейсов. Django, в частности, позволяет быстро разрабатывать сложные и масштабируемые веб-приложения благодаря своему высокоуровневому подходу и широкому набору встроенных функций.

* Анализ данных и машинное обучение

Python стал стандартом в области анализа данных и машинного обучения благодаря своей простоте и обширной экосистеме библиотек. Библиотеки, такие как Pandas, NumPy, SciPy, Matplotlib и Seaborn, позволяют легко обрабатывать, анализировать и визуализировать данные. Фреймворки машинного обучения, такие как TensorFlow, Keras и Scikit-learn, предоставляют мощные инструменты для создания, обучения и развертывания моделей машинного обучения.

* Научные вычисления

Python популярен среди ученых и инженеров для выполнения научных вычислений. Его библиотеки SciPy и NumPy предоставляют высокопроизводительные математические и численные инструменты. Библиотека SymPy позволяет выполнять символические вычисления, что полезно для решения математических задач и моделирования физических процессов.

* Автоматизация и скриптинг

Python часто используется для автоматизации рутинных задач и написания скриптов. Его простота и мощные библиотеки позволяют автоматизировать различные процессы, такие как обработка файлов, управление системами, автоматизация тестирования и взаимодействие с веб-сайтами. Модули, такие как Selenium и BeautifulSoup, позволяют автоматизировать взаимодействие с веб-страницами и парсинг данных.

* Искусственный интеллект и нейросети

Python является ведущим языком для разработки приложений искусственного интеллекта и нейросетей. Библиотеки TensorFlow, Keras и PyTorch предоставляют мощные инструменты для создания и обучения глубоких нейронных сетей. Эти инструменты широко используются для задач компьютерного зрения, обработки естественного языка и других приложений ИИ.

* Разработка игр

Python также используется для разработки игр, особенно для прототипирования и создания небольших проектов. Библиотека Pygame предоставляет инструменты для создания 2D-игр, а фреймворки, такие как Panda3D и Godot (с поддержкой Python), позволяют разрабатывать более сложные игровые приложения.

* Интернет вещей (IoT)

Python часто используется для разработки приложений Интернета вещей. Его простота и кроссплатформенность делают его идеальным выбором для работы с микроконтроллерами и одноплатными компьютерами, такими как Raspberry Pi. Библиотеки, такие как MicroPython и CircuitPython, облегчают разработку встроенных систем и управление аппаратными устройствами.

* Образование

Python широко используется в образовательных учреждениях для обучения программированию и компьютерным наукам. Его простой и понятный синтаксис делает его идеальным для начинающих программистов. Учебные курсы и учебники по Python помогают студентам быстро освоить основы программирования и перейти к более сложным темам.

* Примеры библиотек и фреймворков:
* **Django** и **Flask** для веб-разработки.
* **Pandas**, **NumPy**, **SciPy** для анализа данных и научных вычислений.
* **TensorFlow**, **Keras**, **PyTorch** для машинного обучения и нейросетей.
* **Selenium**, **BeautifulSoup** для автоматизации и скриптинга.
* **Pygame**, **Panda3D** для разработки игр.
* **MicroPython**, **CircuitPython** для IoT-приложений.

Python продолжает развиваться и привлекать внимание разработчиков благодаря своей универсальности и широкому спектру применения.

# Практическая реализация программного обеспечения

## Проектирование алгоритма программного обеспечения средствами языка программирования python.

При проектировании программного обеспечения, предназначенного для ведения записей, предполагается реализация функций, позволяющих пользователям создавать заметки, которые будут фильтроваться по дате создания.

Для разработки данного приложения используется язык программирования python, известный своей простатой и высокой скоростью разработки. Для создания графического интерфейса приложения используются библиотеки tkinter и customtkinter, обеспечивающие возможности для создания интуитивно понятного, простого и приятного глазу интерфейса. Так же я использую базу данных, СУБД – SQL3Lite, не требующие отдельного сервера или настройки.

**Функционал заметок.**

Функционал заметок создается с помощью библиотеки tkinter. С помощью нее создается окно, в котором мы реализуем функционал программы, используя графический интерфейс библиотеки customtkinter, синхронизировав все записи с базой данных.

db\_start();

Инициализирует базу данных и создает таблицу notes, если она еще не существует. Добавляет необходимые столбцы, если они отсутствуют.

initialize\_created\_at();

Обновляет записи в таблице notes, добавляя текущую дату и время в столбец created\_at, если он пуст.

save\_note();

Сохраняет новую заметку в базу данных. Заголовком становится первая строка заметки в верхнем регистре, содержимым — остальные строки. Также сохраняет дату и время создания заметки. Пользователь вводит текст в поле ввода и нажимает клавишу "Tab" или кнопку для сохранения заметки.

capitalize\_first\_letter();

Капитализирует первую букву первой строки текста в поле ввода, если она в нижнем регистре. Пользователь вводит текст в поле ввода, и функция автоматически корректирует первую букву после каждого нажатия клавиши.

delete\_note\_from\_edit(note\_id, edit\_window);

Удаляет выбранную заметку из базы данных и обновляет список заметок. Закрывает окно редактирования. Пользователь открывает заметку для редактирования и нажимает кнопку "Удалить".

edit\_note();

Открывает окно для редактирования выбранной заметки. Позволяет изменять заголовок и содержимое заметки, дважды щелкая на заметку в списке для редактирования.

save\_edited\_note(edit\_window, note\_id, new\_title, new\_content);

Сохраняет изменения в заметке и обновляет список заметок. Закрывает окно редактирования. Пользователь изменяет заголовок и/или содержимое заметки в окне редактирования и нажимает кнопку "Сохранить".

update\_notes\_list();

Обновляет отображение списка заметок, группируя их по дате создания. Эта функция вызывается автоматически при изменении заметок для обновления списка.

clear\_notes();

Удаляет все заметки из базы данных и обновляет список заметок. Пользователь нажимает кнопку "Очистить список" для удаления всех заметок.

on\_note\_double\_click(event);

Обрабатывает двойной щелчок на заметке, вызывая функцию редактирования. Пользователь дважды щелкает на заметку для открытия окна редактирования.

add\_note\_with\_tab(event);

Добавляет новую заметку при нажатии клавиши "Tab".

Также в файле описываются все кнопки, размеры открывающихся окон и оформлениие.

**База данных.**

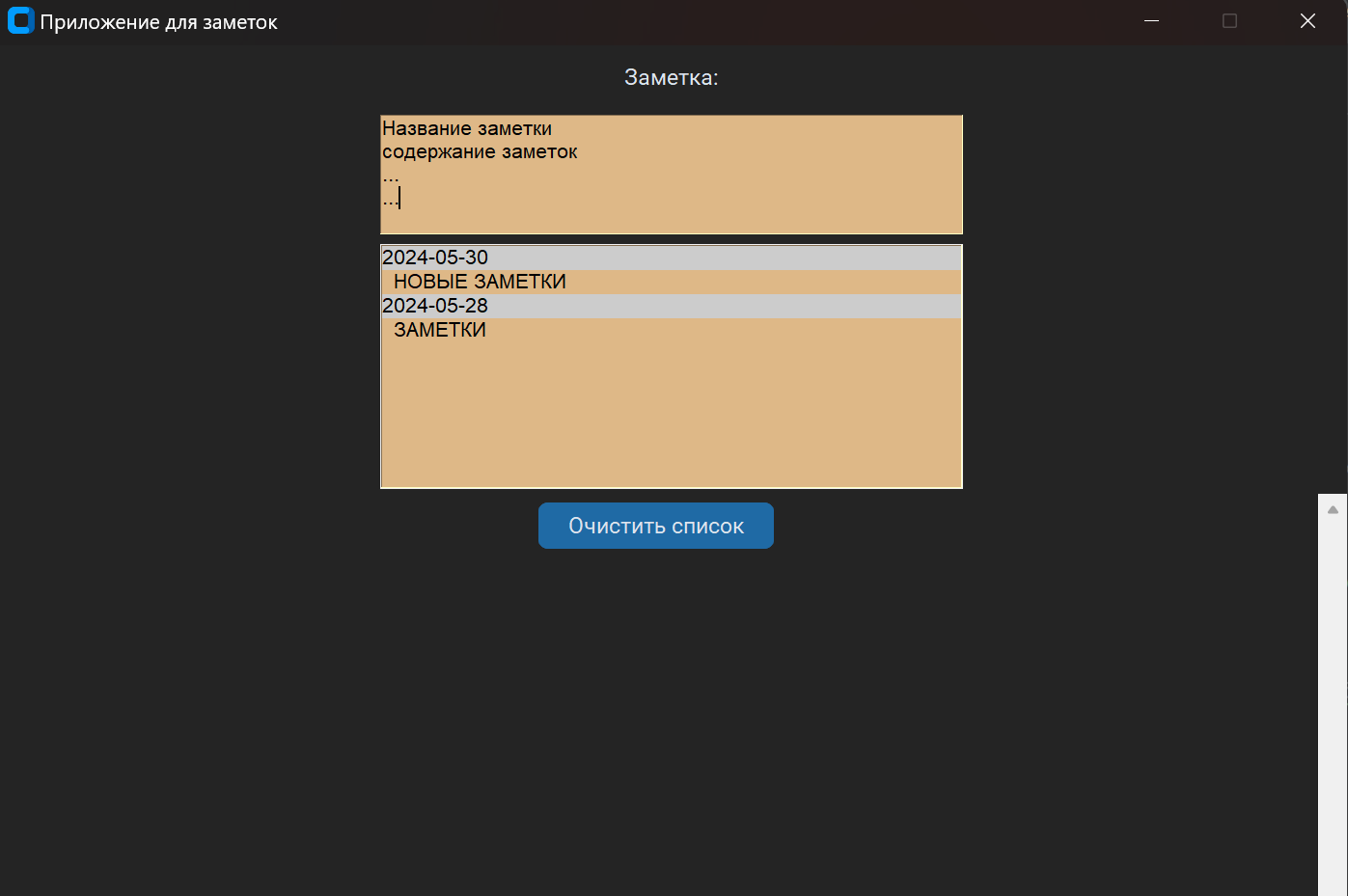
Использование базы данных в приложении для заметок имеет несколько ключевых целей и преимуществ:

1. Хранение данных:
   * База данных служит для долговременного хранения заметок, чтобы они сохранялись между сеансами работы приложения.
   * В случае завершения работы приложения или перезагрузки компьютера данные остаются доступными для последующего использования.
2. Организация данных:
   * База данных позволяет структурировать данные в таблицы с четко определенными столбцами и типами данных, что упрощает управление и обработку информации.
3. Поиск и фильтрация:
   * База данных предоставляет мощные возможности для поиска и фильтрации данных. Это позволяет быстро находить нужные заметки по заголовку, содержимому или дате создания.
4. Целостность данных:
   * Использование базы данных обеспечивает целостность и согласованность данных за счет транзакций, которые позволяют безопасно выполнять несколько операций одновременно.
5. Масштабируемость:
   * База данных легко масштабируется для хранения большого объема данных. Это важно, если количество заметок возрастает со временем.

## Реализация программного обеспечения «Заметки». Обоснование его применения в профессиональной деятельности

Реализация алгоритма, приведенного в пункте 2.1.

Для создания заметок необходимо в первой строчке ввести название заметки, где первая буква будет автоматически в первом регистре, как в заметках на телефоне. Последующие строчки – содержание заметки. Также если попробовать добавить пустую заметку, приложение никак не отреагирует. (рис. 1)

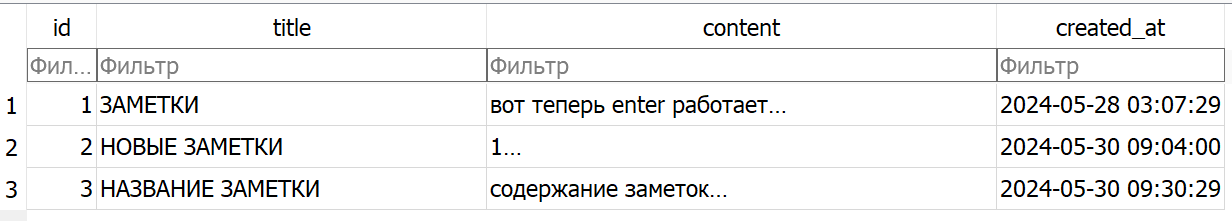


Главная страница (рис. 1)

Каждая заметка отсортирована по дате создания, для удобства пользователей (рис. 2). Все эти данные также передаются в базу данных (рис.3)

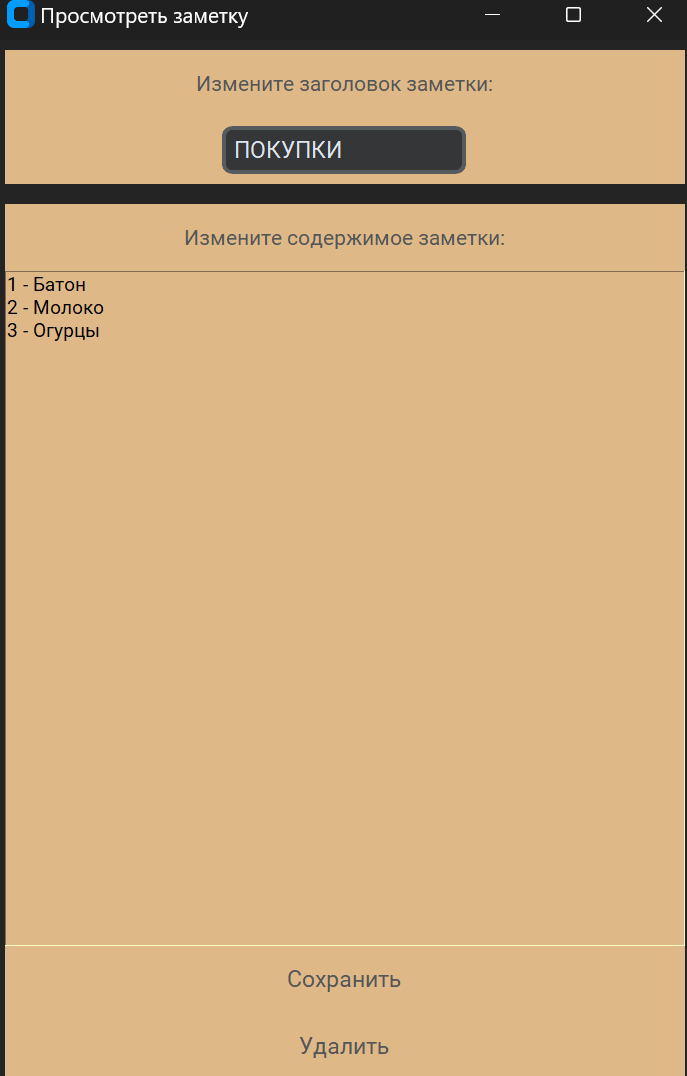


Список существующих заметок (рис. 2)



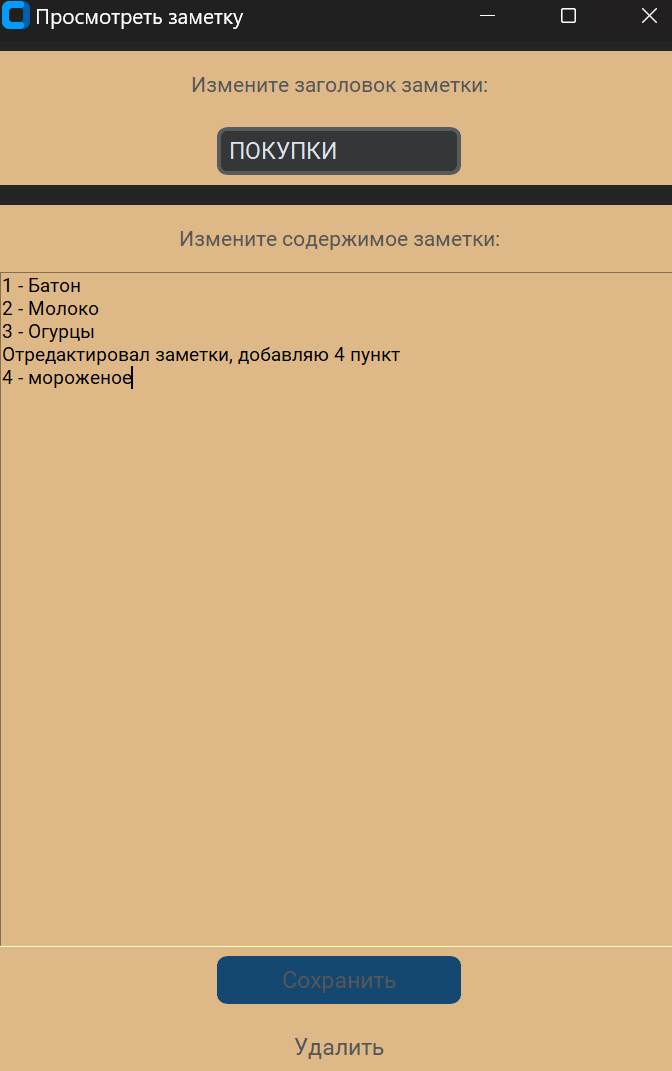
База данных (рис. 3)

Для того чтобы просмотреть заметки, нужно дважды нажать левой кнопкой мыши на нужное нам название в списке (рис. 4)



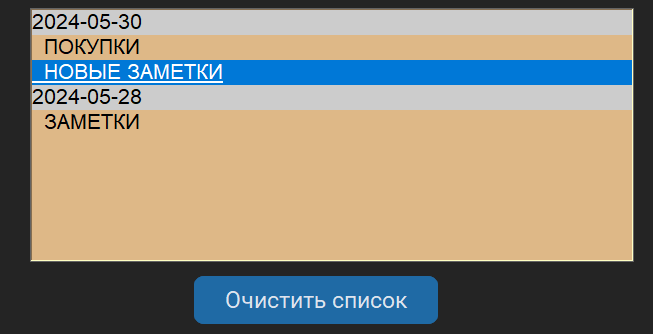
Окно просмотра заметок (рис. 4)

В том же окне просмотра заметок мы можем ее удалить, либо редактировать. Для редактирования названия нужно изменить текст в верхнем окне, а для редактирования содержания в окне ниже. Для того чтобы сохранить изменения в нижней части окна есть кнопка «Сохранить», для удаления «Удалить» (рис. 5)

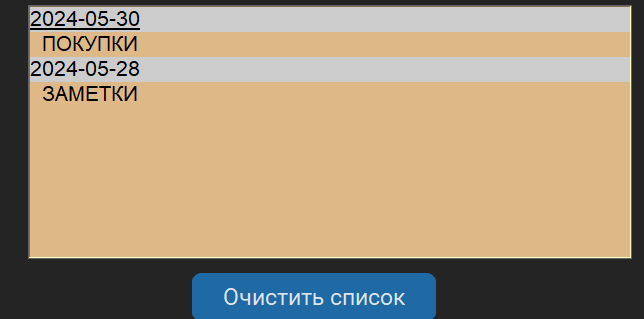


Окно редактирования (рис. 5)

Так же можно удалять заметки не открывая окно редактирования, нужно выделить название из списка и нажать кнопку «delete» на клавиатуре (рис. 6, рис. 7).

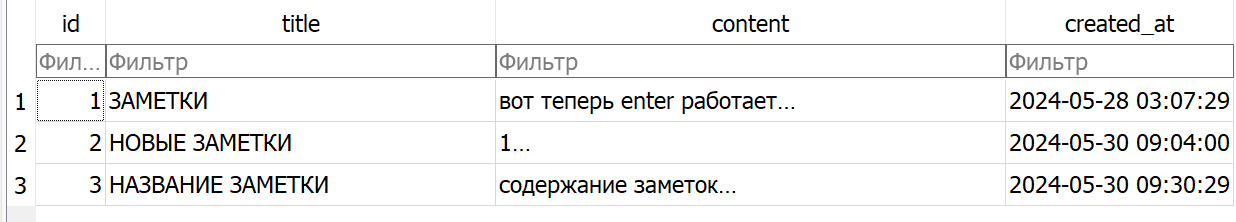


Выделил название (рис. 6)



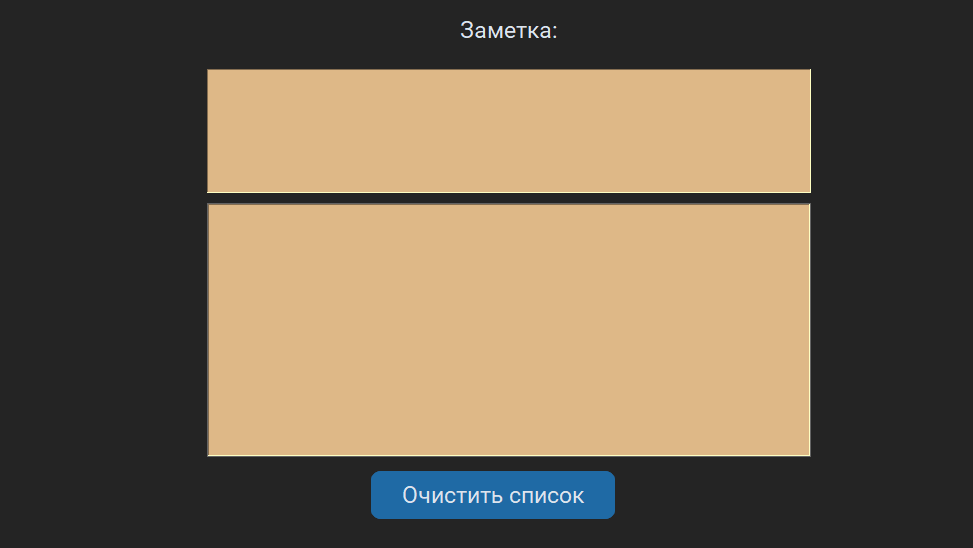
Удаление одной заметки (рис. 7)

Эта заметка удалилась так же и в базе данных (рис. 8).

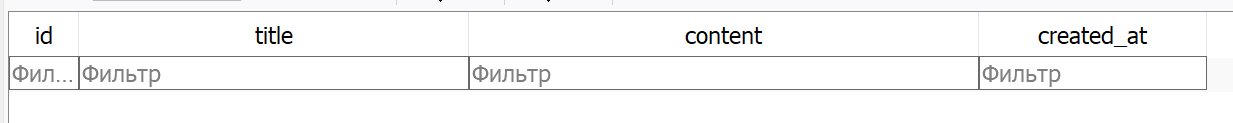


Обновленная база данных (рис. 8)

Если нужно полностью избавиться от всех заметок, в нижней части окна есть кнопка «Очистить список», она полностью удалит все записки из приложения и базы данных (рис. 9, рис. 10).



Использование кнопки «Очистить список» (рис. 9)



Очищенная база данных (рис. 10)

Приложение "Заметки" - это инструмент, который помогает в организации и управлении вашими записями и идеями. Оно предоставляет возможность создавать, редактировать и удалять заметки, а также просматривать их и сохранять в базе данных для последующего доступа. Это приложение может быть полезным инструментом для организации информации, планирования задач, ведения ежедневных журналов или просто для фиксации важных мыслей и заметок.

Нерациональное управление информацией и отсутствие систематизации данных могут привести к потере важных идей или информации. Приложение "Заметки" помогает вам избежать этого, обеспечивая структурированное хранение и легкий доступ к вашим записям. Оно позволяет вам организовывать свои мысли, планировать свои действия и сохранять все необходимые данные в одном месте.

Вот несколько причин, по которым стоит использовать приложение "Заметки":

1. Управление информацией: Основное преимущество приложения "Заметки" заключается в его способности организовывать вашу информацию так, чтобы она была легко доступна и понятна. Вы можете находить нужную информацию по датам или ключевым названиям.
2. Эффективное планирование: Приложение "Заметки" помогает вам планировать свои задачи, проекты и важные события, записывая их в заметки. Это позволяет вам быть более организованным и эффективным в управлении своим временем, и ресурсами.
3. Ведение ежедневника: Вы можете использовать приложение "Заметки" для ведения ежедневного журнала или дневника, записывая свои мысли, чувства, достижения и цели. Это помогает вам отслеживать свой прогресс, анализировать свои мысли и чувства, а также делать выводы из прошлого опыта.

Использование приложения "Заметки" помогает вам быть более организованным, продуктивным и эффективным в управлении вашей информацией, и ресурсами. Оно обеспечивает структурированное хранение вашей информации, легкий доступ к ней и удобные инструменты для работы с ней. Таким образом, приложение "Заметки" может стать незаменимым помощником в вашей повседневной жизни и работе.

# Заключение

В ходе учебной практики был освоен язык программирования Python, и разработано программное обеспечение для решения задачи учета и контроля питания под названием "Заметки". Это приложение позволяет пользователям организовывать свои записи, идеи и планы, обеспечивая структурированное хранение информации и легкий доступ к ней.

Задание выполнено в полном объеме, и были использованы современные информационные технологии и программные средства Python. При изложении материала были соблюдены основные принципы оформления технической документации, а для защиты отчета был подготовлен доклад, объясняющий применение приложения "Заметки" в профессиональной деятельности.

Разработанное приложение "Заметки" предоставляет удобный интерфейс пользователя, реализованный с использованием библиотеки tkinter, и включает следующие основные функции:

* Создание, редактирование и удаление заметок.
* Хранение и предоставление истории изменений заметок.
* Использование базы данных для быстрого доступа и хранения информации

Разработка приложения "Заметки" позволила закрепить на практике полученные теоретические знания по языку программирования Python, а также продемонстрировать навыки проектирования и реализации информационной системы с использованием современных технологий. Применение приложения "Заметки" может быть полезным в различных сферах профессиональной деятельности, включая организацию работы, планирование задач, ведение дневников и журналов, а также для обмена информацией и совместной работы над проектами.

# Список использованных источников

1. Филимонова, Е.В. Информационные технологии в профессиональной деятельности: учебник / Филимонова Е.В. – Москва: КноРус, 2019. – 482 с. — ISBN 978-5-406-06532-7.
2. Рочев К.В. Информационные технологии. Анализ и проектирование информационных систем. Учебное пособие. СПб.: Лань. – 2019.
3. Шукис С.К., Шукис И.К. Python. Быстрый старт. - М.: ДМК Пресс, 2023. - 288 с.
4. Саммерфилд М. Программирование на Python 3. Подробное руководство. - СПб.: Символ-Плюс, 2021. - 608 с.
5. Джонс Дж.Ф. Tkinter для Python. Практическое руководство. - СПб.: Питер, 2021. - 320 с.
6. Овсянников В.А. Язык программирования Python. Практикум. - М.: РИОР, 2023. - 272 с.
7. Миронов Д.Ф. Python для начинающих. Практический курс по программированию. - М.: ДМК Пресс, 2020. - 256 с.

# Приложение. Реализация программного обеспечения «Заметки»

import tkinter as tk

import customtkinter

import sqlite3

import datetime

# Функция для инициализации базы данных

def db\_start():

global conn, cur

conn = sqlite3.connect('notes.db') # Подключаемся к базе данных 'notes.db'

cur = conn.cursor()

# Создаем таблицу 'notes', если она не существует

cur.execute("CREATE TABLE IF NOT EXISTS notes (id INTEGER PRIMARY KEY, title TEXT, content TEXT, created\_at TEXT)")

conn.commit()

# Проверяем, существуют ли необходимые столбцы и добавляем их, если нет

cur.execute("PRAGMA table\_info(notes)")

columns = [column[1] for column in cur.fetchall()]

if 'title' not in columns:

cur.execute("ALTER TABLE notes ADD COLUMN title TEXT")

conn.commit()

if 'content' not in columns:

cur.execute("ALTER TABLE notes ADD COLUMN content TEXT")

conn.commit()

if 'created\_at' not in columns:

cur.execute("ALTER TABLE notes ADD COLUMN created\_at TEXT")

conn.commit()

# Функция для инициализации столбца created\_at, если он пуст

def initialize\_created\_at():

cur.execute("SELECT id, created\_at FROM notes")

notes = cur.fetchall()

for note in notes:

note\_id, created\_at = note

if created\_at is None:

cur.execute("UPDATE notes SET created\_at = ? WHERE id = ?",

(datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S"), note\_id))

conn.commit()

# Функция для сохранения новой заметки

def save\_note(event=None):

note\_text = note\_entry.get("1.0", tk.END).strip() # Получаем текст из текстового поля

if not note\_text: # Проверяем, если текст заметки пустой, не сохраняем заметку

return

note\_lines = note\_text.split("\n")

if not note\_lines:

return

title = note\_lines[0].upper() # Заголовок - первая строка, преобразованная в верхний регистр

content = "\n".join(note\_lines[1:]) # Содержание - все остальные строки

created\_at = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S") # Время создания заметки

cur.execute("INSERT INTO notes (title, content, created\_at) VALUES (?, ?, ?)", (title, content, created\_at))

conn.commit()

update\_notes\_list()

note\_entry.delete("1.0", tk.END) # Очищаем текстовое поле

# Функция для капитализации первой буквы первой строки

def capitalize\_first\_letter(event):

first\_line = note\_entry.get("1.0", "1.end")

if first\_line and first\_line[0].islower():

note\_entry.delete("1.0")

note\_entry.insert("1.0", first\_line.capitalize())

# Функция для удаления заметки из режима редактирования

def delete\_note\_from\_edit(note\_id, edit\_window):

cur.execute("DELETE FROM notes WHERE id=?", (note\_id,))

conn.commit()

update\_notes\_list()

edit\_window.destroy()

# Функция для удаления выбранной заметки

def delete\_selected\_note():

index = notes\_list.curselection()

if index:

selected\_note = notes\_list.get(index)

if not selected\_note[0].isdigit(): # Проверяем, что это не дата

cur.execute("SELECT id FROM notes WHERE title=?", (selected\_note.strip(),))

note = cur.fetchone()

if note:

note\_id = note[0]

cur.execute("DELETE FROM notes WHERE id=?", (note\_id,))

conn.commit()

update\_notes\_list()

# Функция для редактирования заметки

def edit\_note():

index = notes\_list.curselection() # Получаем выбранную заметку

if index:

selected\_note = notes\_list.get(index)

if selected\_note[0].isdigit(): # Проверяем, что это не дата

return

cur.execute("SELECT \* FROM notes WHERE title=?", (selected\_note.strip(),))

note = cur.fetchone()

if note:

edit\_window = customtkinter.CTkToplevel(root)

edit\_window.title("Просмотреть заметку")

edit\_window.geometry("400x600")

edit\_window.configure(bg='#757272')

edit\_window.grab\_set()

edit\_window.lift()

edit\_window.attributes('-topmost', True)

edit\_window.after(1, lambda: edit\_window.attributes('-topmost', False))

top\_frame = tk.Frame(edit\_window, bg="burlywood")

top\_frame.pack(side=tk.TOP, fill=tk.X, padx=10, pady=10)

edit\_title\_label = customtkinter.CTkLabel(top\_frame, text="Измените заголовок заметки:",

font=("Roboto", 12), text\_color="#555657")

edit\_title\_label.pack(pady=5)

edited\_title = tk.StringVar(value=note[1])

edit\_title\_entry = customtkinter.CTkEntry(top\_frame, textvariable=edited\_title)

edit\_title\_entry.pack(pady=5)

bottom\_frame = tk.Frame(edit\_window, bg="burlywood")

bottom\_frame.pack(side=tk.BOTTOM, fill=tk.BOTH, expand=True, padx=10, pady=10)

edit\_content\_label = customtkinter.CTkLabel(bottom\_frame, text="Измените содержимое заметки:",

font=("Roboto", 12), text\_color="#555657")

edit\_content\_label.pack(pady=5)

edit\_content\_entry = tk.Text(bottom\_frame, height=10, bg="burlywood", font=("Roboto", 14))

edit\_content\_entry.insert(tk.END, note[2])

edit\_content\_entry.pack(fill=tk.BOTH, expand=True)

save\_button = customtkinter.CTkButton(bottom\_frame, text="Сохранить",

command=lambda: save\_edited\_note(edit\_window, note[0],

edited\_title.get(),

edit\_content\_entry.get("1.0",

tk.END).strip()),

fg\_color="burlywood", text\_color="#555657")

save\_button.pack(pady=5)

delete\_button = customtkinter.CTkButton(bottom\_frame, text="Удалить",

command=lambda: delete\_note\_from\_edit(note[0], edit\_window),

fg\_color="burlywood", text\_color="#555657")

delete\_button.pack(pady=5)

edit\_window.protocol("WM\_DELETE\_WINDOW", lambda: edit\_window.grab\_release() or edit\_window.destroy())

# Функция для сохранения изменений заметки

def save\_edited\_note(edit\_window, note\_id, new\_title, new\_content):

cur.execute("UPDATE notes SET title=?, content=? WHERE id=?", (new\_title, new\_content, note\_id))

conn.commit()

update\_notes\_list()

edit\_window.grab\_release()

edit\_window.destroy()

# Функция для обновления списка заметок

def update\_notes\_list():

notes\_list.delete(0, tk.END)

cur.execute("SELECT title, created\_at FROM notes ORDER BY created\_at DESC")

notes = cur.fetchall()

current\_date = None

for note in notes:

title, created\_at = note

if created\_at is None:

created\_at = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

note\_date = created\_at.split(" ")[0]

if note\_date != current\_date:

notes\_list.insert(tk.END, note\_date)

notes\_list.itemconfig(tk.END, {'bg': '#cccccc'})

current\_date = note\_date

notes\_list.insert(tk.END, " " + title)

# Функция для очистки всех заметок

def clear\_notes():

cur.execute("DELETE FROM notes")

conn.commit()

update\_notes\_list()

# Функция для обработки двойного щелчка на заметке

def on\_note\_double\_click(event):

edit\_note()

# Функция для добавления заметки при нажатии на Tab

def add\_note\_with\_tab(event):

if event.keysym == "Tab":

save\_note()

# Функция для обработки нажатия клавиши Delete для удаления заметки

def on\_delete\_key(event):

delete\_selected\_note()

# Инициализация главного окна

root = customtkinter.CTk()

root.title("Приложение для заметок")

root.geometry("800x800")

root.resizable(0, 0)

# Метка для заметок

note\_label = customtkinter.CTkLabel(root, text="Заметка:")

note\_label.pack(pady=5)

# Поле для ввода заметок

note\_entry = tk.Text(root, height=5, width=50, bg="burlywood", font=("Robotto", 16))

note\_entry.pack(pady=5)

note\_entry.bind("<Tab>", add\_note\_with\_tab)

# Список для отображения заметок

notes\_list = tk.Listbox(root, width=50, height=10, bg="burlywood", font=("Robotto", 16))

notes\_list.pack(pady=5)

notes\_list.bind("<Double-Button-1>", on\_note\_double\_click)

notes\_list.bind("<Delete>", on\_delete\_key) # Привязываем удаление заметки к клавише Delete

# Полоса прокрутки для списка заметок

scrollbar = tk.Scrollbar(root, orient=tk.VERTICAL)

scrollbar.config(command=notes\_list.yview)

scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

notes\_list.config(yscrollcommand=scrollbar.set)

# Кнопка для очистки всех заметок

clear\_button = customtkinter.CTkButton(root, text="Очистить список", command=clear\_notes)

clear\_button.pack(pady=5)

# Инициализация базы данных и данных

db\_start()

initialize\_created\_at()

update\_notes\_list()

# Привязка событий к текстовому полю

note\_entry.bind("<KeyRelease>", capitalize\_first\_letter)

# Запуск главного цикла приложения

root.mainloop()

conn.close() # Закрываем соединение с базой данных после закрытия приложения