МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение

высшего образования

**«Северный (Арктический) федеральный университет имени М.В. Ломоносова»**

Высшая школа информационных технологий и автоматизированных систем

(наименование высшей школы / филиала / института / колледжа)

**КУРСОВОЙ ПРОЕКТ**

|  |  |  |
| --- | --- | --- |
| По дисциплине/междисциплинарному курсу/модулю | | Программирование |
|  | | |
|  | | |
| На тему | Органайзер | |
|  | | |

|  |  |
| --- | --- |
|  | Выполнил обучающийся:  Щукин Леонид Юрьевич |
|  | (Ф.И.О.) |
|  | Направление подготовки / специальность:  09.03.00 Прикладная информатика |
|  | (код и наименование) |
|  | Курс: 2 |
|  | Группа: 153319 |
|  | Руководитель:  Петров Дмитрий Николаевич |
|  | (Ф.И.О. руководителя, должность / уч. степень / звание) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Признать, что проект выполнен и защищен с отметкой |  |  |  |  |
|  |  | (отметка прописью) |  | (дата) |
| Руководитель |  |  |  |  |
|  |  | (подпись руководителя) |  | (инициалы, фамилия) |

Архангельск 2025

# Лист задания

***\*13 Органайзер***

Реализовать приложение-органайзер с возможностью отображения расписания на день и создания напоминаний, todo-листов и заметок. Добавить календарь, систему уведомлений по заданной дате и времени.

# Лист для замечаний

Оглавление

[Лист задания 2](#_Toc187810081)

[Лист для замечаний 3](#_Toc187810082)

[Введение 5](#_Toc187810083)

[1. Определение требований к разрабатываемому приложению 6](#_Toc187810084)

[2. Проектирование приложения 8](#_Toc187810085)

[3. Реализация приложения 13](#_Toc187810086)

[4. Тестирование 20](#_Toc187810087)

[5. Заключение 24](#_Toc187810088)

[6. Список использованных источников 25](#_Toc187810089)

[7. Приложение. Листинг приложения 26](#_Toc187810090)

# Введение

Цель данной курсовой работы — разработка органайзера, который позволяет пользователю эффективно управлять расписанием, напоминаниями, списками задач (todo-листами) и заметками. Приложение включает систему уведомлений по заданной дате и времени, а также календарь для удобного отображения событий.

Современные органайзеры играют важную роль в планировании и организации времени. Они помогают пользователям решать повседневные задачи, напоминать о важных событиях и следить за сроками выполнения заданий. Тем не менее, многие существующие решения сложны в использовании или имеют ограниченные функции, что делает их менее удобными для некоторых пользователей. Разрабатываемое приложение стремится устранить эти недостатки за счет интуитивного интерфейса и гибкой настройки уведомлений.

В ходе работы была поставлена задача: создать удобное и функциональное приложение-органайзер, которое будет удовлетворять потребности как студентов, так и работающих специалистов. Приложение разрабатывалось с использованием современных инструментов программирования, таких как Python и библиотека CustomTkinter.

Данная работа включает в себя полный цикл разработки: анализ существующих решений, определение требований к приложению, проектирование архитектуры и интерфейса, реализацию функционала, тестирование и документирование. Итогом работы является приложение, которое не только соответствует заявленным требованиям, но и может быть расширено в будущем для включения новых функций.

# Определение требований к разрабатываемому приложению

Приложение предназначено для упрощения планирования личного времени, повышения продуктивности и организации повседневных задач. Разработка приложения преследует следующие цели:

Получить удобный инструмент для планирования дня, включая просмотр расписания и создание событий.

Внедрить систему напоминаний, позволяющую уведомлять пользователей о важных событиях или задачах в заданное время.

Предоставить пользователям возможность создания и управления списками задач (to-do-листами) для структурирования текущих дел.

Проанализированы популярные органайзеры, такие как Microsoft To Do, Google Keep, и Notion. Основные недостатки — сложность интерфейса и необходимость подключения к сети интернет.

Исходя из вышеописанного к программе предоставляются следующие требования:

Возможность создавать расписание, напоминания, todo-листы и заметки.

Интеграция календаря.

Система уведомлений по заданной дате и времени.

Удобный пользовательский интерфейс.

Возможность сохранять и загружать данные.

Отсутствие необходимости подключения к сети интернет.

Состав и содержание работ по созданию программы состоят из:

Анализ требований.

Проектирование архитектуры приложения.

Разработка пользовательского интерфейса.

Реализация логики приложения.

Тестирование и документирование.

Приложение считается завершенным, если оно удовлетворяет требованиям ТЗ, описанного в требованиях к программе, и успешно проходит тестирование.

Документирование работы содержит пояснительную записку, содержащую описание кода, схемы и инструкции для пользователя.

# Проектирование приложения

В приложении существует единственная роль – пользователь. Он взаимодействует с интерфейсом, создает задачи, заметки и напоминания. Работа пользователя с приложением описывается на рисунке 2.1.

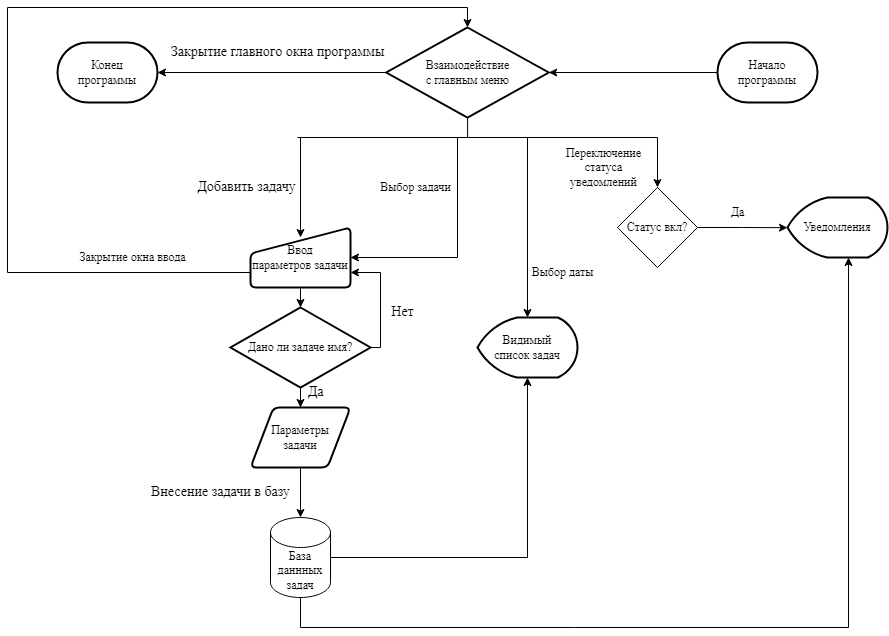


Рисунок 2.1 – блок схема программы

База данных задач состоит из таблицы tasks со следующими полями:

id INTEGER PRIMARY KEY AUTOINCREMENT,

name TEXT NOT NULL,

description TEXT,

start time TEXT,

end time TEXT,

date TEXT NOT NULL,

tags TEXT,

done INTEGER NOT NULL,

notified INTEGER NOT NULL,

date notif TEXT NOT NULL

где name – имя задачи, description – описание задачи, start\_time – время начала задачи, end\_time – время окончания задачи, date – день задачи, tags – теги задачи, done – статус выполнения задачи (0 или 1), notified – статус получения пользователем уведомления, date\_notif – дата получения уведомления по этой задаче.

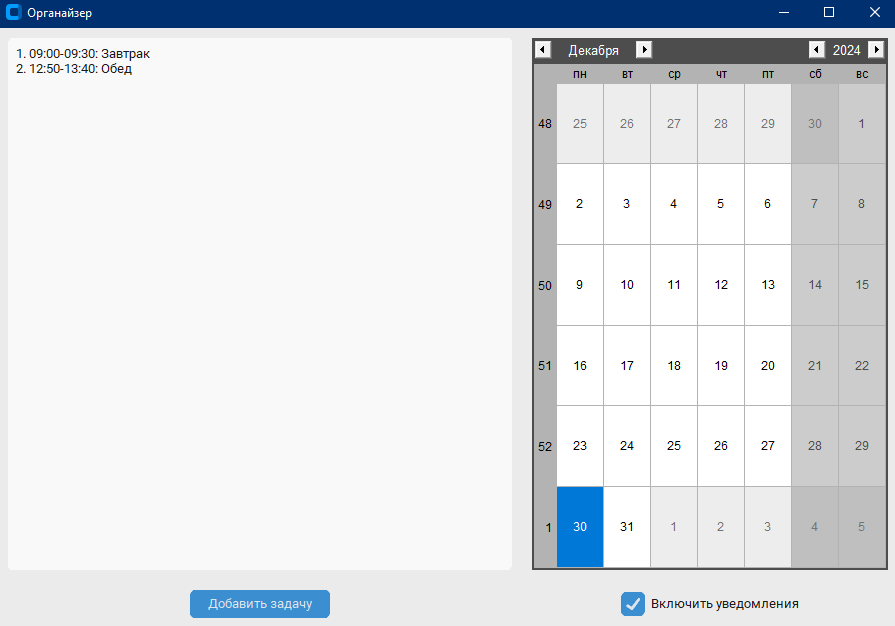
Взаимодействие с программой начинается в главном меню. 

Рисунок 2.2 – окно «Главное меню»

Слева отображается список задач выбранного в календаре дня (при запуске программы по умолчанию выбирается сегодняшний день). Справа расположен календарь. Выбрав дату, пользователь сможет просматривать расписание слева, а также создавать и редактировать задачи этого дня. Для создания задачи слева внизу находится кнопка «Добавить задачу», нажав на которую появится окно «Добавить задачу».

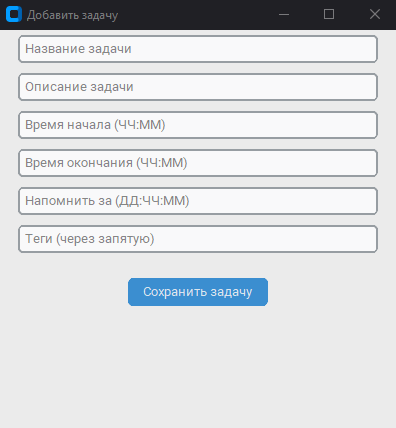
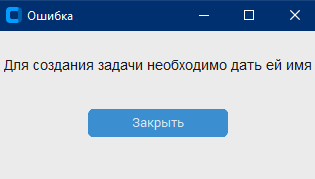


Рисунок 2.3 – окно «Добавить задачу»

В нем пользователю предлагается описать задачу, которую он хочет внести в свое расписание. Обязательным атрибутом задачи является «Название задачи». Без нее запрос о создании задачи будет отклонен, а на экране появится окно ошибки. Также ошибка будет возникать, если время начала, конца или напоминания введены неправильно.

 Рисунок 2.4 – окно «Ошибка»

В правом нижнем углу главного меню находится переключатель статуса получения уведомлений. Если справа от текста «Включить уведомления» стоит галочка, то пользователь будет получать уведомления в виде всплывающих окон со всеми атрибутами задачи.

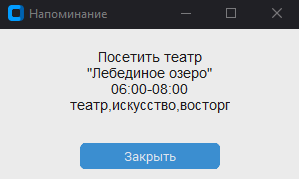
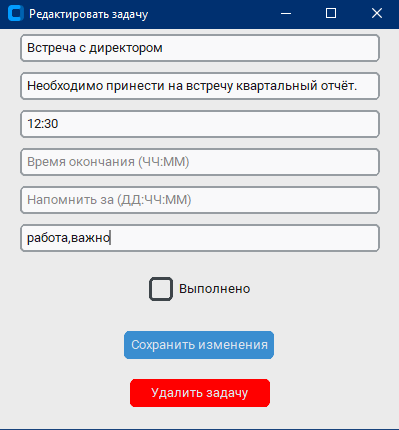


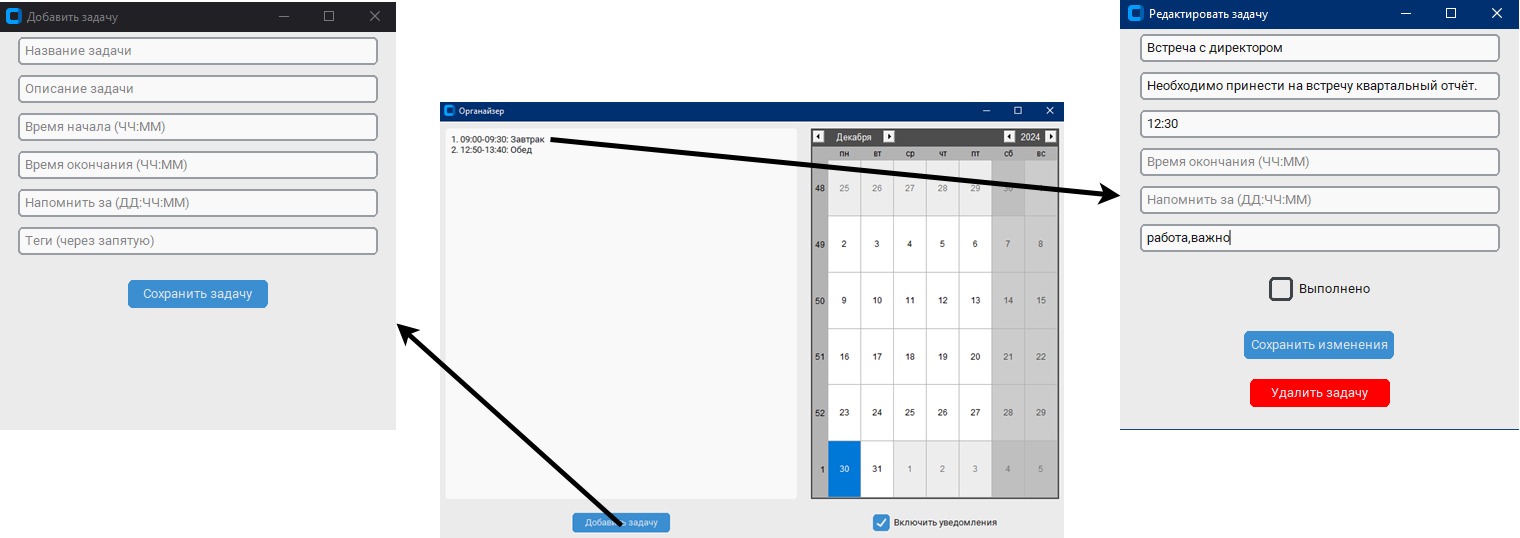
Рисунок 2.5 – окно «Напоминание»

Отредактировать раннее поставленную задачу можно с помощью списка в левой части главного меню. Достаточно нажать дважды левой кнопкой мыши по задаче, и появится окно редактирование, где также можно удалить задачу.

Рисунок 2.6 – окно «Редактировать задачу»

Взаимодействие модулей также дополнительно изображено на схеме ниже:

Рисунок 2.7 – схема взаимодействия модулей



# Реализация приложения

Для реализации приложения был избран язык Python вследствие его удобства, широкого спектра библиотек и открытой экосистемы. Для создания графического интерфейса пользователя (GUI) была выбрана библиотека CustomTkinter за ее совместимость с Python 3.12 и улучшенные визуальные элементы.

В качестве среды разработки использовалась PyCharm, которая обеспечивает удобство в отладке, навигации по коду и интеграции с системами контроля версий. Дополнительно применялись средства автоматизации, такие как встроенный инспектор кода и форматтеры.

Также для создания приложения использовались сторонние бибилотеки. Так, CustomTkinter использовался в разработке для создания современного интерфейса в Python, SQLite3 — для хранения задач в виде базы данных, re — для проверки формата ввода с помощью регулярных выражений, и Threading — для реализации фоновых потоков, таких как оповещения.

Разработка пользовательского интерфейса производилась с помощью CustomTkinter, посредством использования таких классов как ctk.CTk, CTkTextbox, CTkButton и многих других. Расположение объектов этих классов задавалось с помощью методов grid и pack.

В программе реализовано 4 класса. Класс Task используется для создания объектов-задач. Свойства класса описаны в таблице 3.1, его методы – в таблице 3.2.

Таблица 3.1 – свойства класса Task

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Описание |
| 1 | 2 | 3 |
| id | int | Уникальный номер задачи |
| name | str | Имя задачи |
| description | str | Описание задачи |
| start\_time | str | Время начала задачи в формате ЧЧ:ММ |
| end\_time | str | Время окончания задачи в формате ЧЧ:ММ |
| date | str | День задачи в формате ГГ-ММ-ДД |

Окончание таблицы 3.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| tags | str | Строка с перечислением тегов задачи |
| done | int | Статус выполнения задачи, принимающий значения 0 или 1 |
| notified | int | Статус уведомления задачи, принимающий значения 0 или 1 |
| date notif | str | Дата уведомления задачи в формате ГГ-ММ-ДД ЧЧ:ММ |

Таблица 3.2 – методы класса Task

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |
| init . | name: str, description: str, start\_time: str, end\_time: str, date: str, tags: str, done: int, notified: int, date\_notif: str, id: int | Task | Метод инициализации объекта класса Task |
| to\_dict | self | dict | Возвращает задачу в виде словаря с ключами-атрибутами |
| from\_dict | data: dict | Task | Возвращает задачу с параметрами из словаря |
| \_\_eq\_\_ | other: Any | bool | Метод сравнения класса |

Класс OrganizerApp представляет собой главное меню программы. Свойства класса описаны в таблице 3.3, его методы – в таблице 3.4.

Таблица 3.3 – свойства класса OrganizerApp

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Описание |
| 1 | 2 | 3 |
| tasks\_db | str | Имя файла БД |
| config\_file | str | Имя файла настройки приложения |
| date\_format | str | Формат даты, используемый для методов |
| cur\_tasks | list | Список Task, отображаемых в данный момент слева |
| title | str | Название окна |
| geometry | str | Строка с заданием размеров окна |
| today\_task | list | Список Task, соответствующих сегодняшнему дню |
| calendar | tkcalendar.Calendar | Календарь в правой части экрана |
| task\_listbox | ctk.CTkTextbox | Список задач слева |
| add\_task\_button | ctk.CTkButton | Кнопка добавления задачи |
| notifications\_enabled | ctk.BooleanVar() | Переменная, соответствующая статусу получения обновления |
| notifications\_enabled\_flag | bool | Потокобезопасный флаг проверки статуса получения уведомлений |
| checkbox | ctk.CTkCheckBox | Чекбокс, соответствующий статусу получения обновления |

Окончание таблицы 3.3

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| check\_thread | threading.Thread | Поток проверки о необходимости уведомления задачи |
| stop\_check\_time | threading.Event | Событие потока проверки необходимости уведомления |

Таблица 3.4 – методы класса OrganizerApp

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |
| init . | self | Отсутствует | Метод инициализации объекта класса OrganizerApp |
| create\_interface | self | None | Создает главное меню приложения |
| update\_task\_list | self, event: Event | None | Обновляет список задач в левой части главного меню |
| load\_config | self | None | Загружает файл с состоянием флага о получении уведомлений |
| save\_config | self | None | Сохраняет состояние флага в файл настройки, а также запускает thread проверки уведомлений, если флаг включен |
| open\_task\_editor | self, event: Event | None | Открывает окно редактора задач |

Окончание таблицы 3.3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| open\_add\_task\_window | self | None | Открывает окно «Добавить задачу» |
| check\_time | self | None | Поток для проверки необходимости отправить уведомление |
| show\_notification | self, task: Task | None | Создает окно уведомления и показывает его пользователю |

Класс BaseTaskWindow представляет собой предка окон добавления и редактирования задачи. Свойства класса описаны в таблице 3.5, его методы – в таблице 3.6.

Таблица 3.5 – свойства класса BaseTaskWindow

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Описание |
| 1 | 2 | 3 |
| parent | OrganizerApp | Главное меню программы |
| date | str | Дата добавляемой или редактируемой задачи |
| name\_entry | ctk.CTkEntry | Строка ввода имени задачи |
| description\_entry | ctk.CTkEntry | Строка ввода описания задачи |
| start\_time\_entry | ctk.CTkEntry | Строка ввода времени начала задачи |
| end\_time\_entry | ctk.CTkEntry | Строка ввода времени конца задачи |
| date\_notif\_entry | ctk.CTkEntry | Строка ввода времени уведомления задачи |
| tags\_entry | ctk.CTkEntry | Строка ввода тегов задачи |

Таблица 3.6 – методы класса BaseTaskWindow

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |
| init . | self, parent | Отсутствует | Метод инициализации объекта класса TaskWindow |
| get\_data | self | Task | Возвращает задачу по данным из полей ввода |
| get\_date\_notif | self, start\_time: str, date\_notif: str | str | Вычисляет date\_notif для task перед добавлением в БД |

Класс AddTaskWindow представляет собой окно добавления задачи. Свойства класса описаны в таблице 3.7, его методы – в таблице 3.8.

Таблица 3.7 – свойства класса AddTaskWindow

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Описание |
| 1 | 2 | 3 |
| title | str | Название окна |
| geometry | str | Строка с заданием размеров окна |
| save\_button | ctk.CTkButton | Кнопка для сохранения изменений задачи |

Таблица 3.8 – методы класса AddTaskWindow

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |
| init . | self, parent, task | Отсутствует | Метод инициализации объекта класса AddTaskWindow |
| add\_task | self | None | Обновляет задачу в БД |

Класс EditTaskWindow представляет собой окно редактирования задачи. Свойства класса описаны в таблице 3.9, его методы – в таблице 3.10.

Таблица 3.9 – свойства класса EditTaskWindow

|  |  |  |
| --- | --- | --- |
| Наименование | Тип данных | Описание |
| 1 | 2 | 3 |
| title | str | Название окна |
| geometry | str | Строка с заданием размеров окна |
| save\_button | ctk.CTkButton | Кнопка для сохранения изменений задачи |
| done\_status | ctk.IntVar | Переменная, связанная со статусом выполнения задачи |
| checkbox | ctk.CTkCheckBox | Чекбокс, связанный с статусом выполнения задачи |
| delete\_button | ctk.CTkButton | Кнопка для удаления задачи |

Таблица 3.10 – методы класса EditTaskWindow

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |
| init . | self, parent, task | Отсутствует | Метод инициализации объекта класса EditTaskWindow |
| update\_task | self | None | Обновляет задачу в БД |
| delete\_task | self | None | Удаляет задачу из БД |
| get\_what\_insert\_in\_date\_notif | self, task: Task | str | Вычисляет вставку в поле времени для уведомления |

Также в программе есть методы, не принадлежащие ни одному классу. Их описание можно найти в таблице 3.11.

Таблица 3.11 – методы, не принадлежащие классам

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Принимаемые параметры | Возвращаемое значение | Описание |
| 1 | 2 | 3 | 4 |
| create\_table | Отсутствуют | None | Создает базу данных с таблицей tasks |
| show\_error\_popup | message: str | None | Создает окно ошибки с текстовым сообщением внутри |
| get\_tasks\_by\_date | date: str, task\_db: str | list | Возвращает список tasks соответствующих переданной дате |

# Тестирование

Зависимость работы приложения от количества данных в общем случае равняется O(n), где n – число задач в БД. Однако, с помощью индексирования БД по тем или иным принципам (к примеру, индексировать задачи по совокупности тегов) можно добиться уменьшения времени запроса.

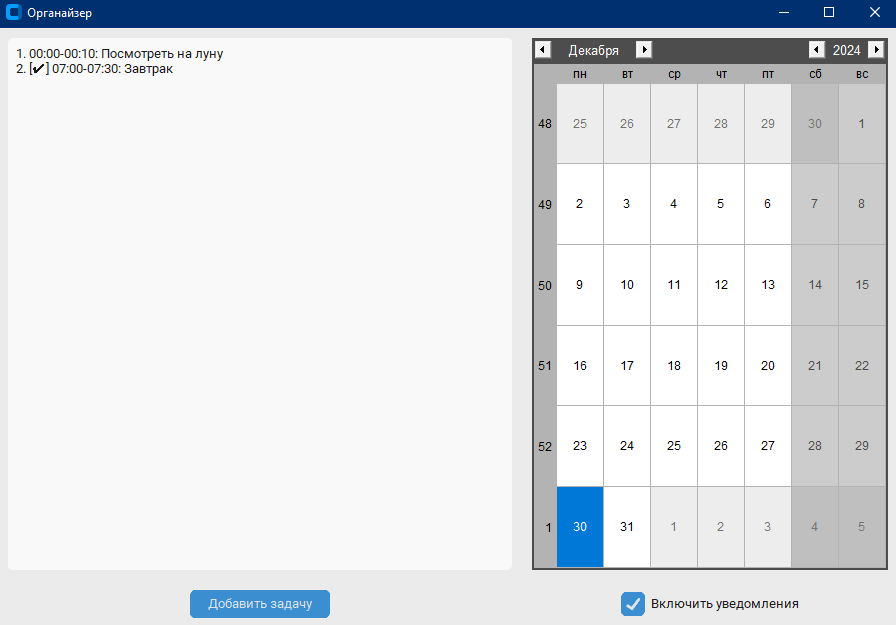
Unit тесты находятся в папке tests и могут быть использованы разработчиками в будущем. Описание тестов можно найти в таблице 4.1.

Таблица 4.1 – описание unit тестов

|  |  |  |
| --- | --- | --- |
| Наименование | Проверяемая единица | Описание |
| 1 | 2 | 3 |
| test\_task\_initialization | \_\_init\_\_ | Тестирует правильность инициализации объекта Task |
| test\_to\_dict | to\_dict | Тестирует правильность преобразования задачи в словарь |
| test\_from\_dict\_success | from\_dict | Тестирует создание задачи из корректного словаря |
| test\_from\_dict\_missing\_keys | from\_dict | Тестирует обработку отсутствующих ключей в словаре |
| test\_from\_dict\_key\_error\_message | from\_dict | Тестирует вывод ошибки при отсутствии ключа |
| test\_equal\_tasks | \_\_eq\_\_ | Тестирует, что два идентичных объекта считаются равными |
| test\_not\_equal\_tasks | \_\_eq\_\_ | Тестирует, что два разных объекта считаются неравными |
| test\_comparison\_with\_other\_type | \_\_eq\_\_ | Тестирует, что объект Task не равен объекту другого типа |

Окончание таблицы 4.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| test\_self\_equality | \_\_eq\_\_ | Тестирует, что объект равен самому себе |
| test\_create\_interface | create\_interface | Тестирует создание интерфейса |
| test\_load\_config | load\_config | Тестирует загрузку конфигурации |
| test\_save\_config | save\_config | Тестирует сохранение конфигурации |
| test\_update\_task\_list | update\_task\_list | Тестирует обновление списка задач |
| test\_open\_task\_editor | open\_task\_editor | Тестирует открытие окна редактора задачи |
| test\_open\_add\_task\_window | open\_add\_task\_window | Тестирует открытие окна добавления задачи |
| test\_check\_time | check\_time | Тестирует метод check\_time |
| test\_show\_notification | show\_notification | Тестирует создание окна уведомления |
| test\_create\_table | create\_table | Тестирует создание таблицы |
| test\_add\_task\_valid\_data | add\_task | Тестирует добавление задачи с валидными данными |
| test\_update\_task\_valid\_data | update\_task | Тестирует обновление задачи с валидными данными |
| test\_delete\_task | delete\_task | Тестирует добавление задачи с валидными данными |
| test\_get\_what\_insert\_in\_date\_notif | get\_what\_insert\_in\_date\_notif | Тестирует метод get\_what\_insert\_in\_date\_notif |
| test\_show\_error\_popup | show\_error\_popup | Тестирует, что окно ошибки создается корректно |
| test\_get\_tasks\_by\_date | get\_tasks\_by\_date | Тестирует функцию get\_tasks\_by\_date |

Рисунок 4.1 – скриншот работающего приложения

Взаимодействие с программой начинается в главном меню. Слева отображается список задач выбранного в календаре дня (при запуске программы по умолчанию выбирается сегодняшний день). Справа расположен календарь. Выбрав дату, пользователь сможет просматривать расписание слева, а также создавать и редактировать задачи этого дня. Для создания задачи слева внизу находится кнопка «Добавить задачу», нажав на которую появится окно «Добавить задачу».

В нем пользователю предлагается описать задачу, которую он хочет внести в свое расписание. Обязательным атрибутом задачи является «Название задачи». Без нее запрос о создании задачи будет отклонен, а на экране появится окно ошибки. Также ошибка будет возникать, если время начала, конца или напоминания введены неправильно.

В правом нижнем углу главного меню находится переключатель статуса получения уведомлений. Если справа от текста «Включить уведомления» стоит галочка, то пользователь будет получать уведомления в виде всплывающих окон со всеми атрибутами задачи.

Отредактировать раннее поставленную задачу можно с помощью списка в левой части главного меню. Достаточно нажать дважды левой кнопкой мыши по задаче, и появится окно редактирование, где также можно удалить задачу.

Приложение защищено от некорректного ввода времени, а потому некорректная эксплуатация не предоставляется возможной.

# Заключение

Разработанное приложение-органайзер представляет собой удобный и функциональный инструмент для управления временем и задачами. В ходе работы были реализованы основные возможности, соответствующие современным требованиям пользователей: отображение расписания, создание и управление списками дел, заметками, напоминаниями, а также интеграция календаря с поддержкой уведомлений.

Приложение помогает повысить личную и профессиональную продуктивность за счет централизованного управления информацией и задачами в одном интерфейсе. Его использование позволяет пользователям эффективно планировать свой день, избегать пропуска важных событий и структурировать текущие дела.

Поставленные цели были достигнуты, а созданное приложение обладает потенциалом для дальнейшего развития. Возможными направлениями улучшений могут стать поиск задач по тегам, ускорение работы приложения за счёт введения индексации в БД, вывод статистики выполнения задач, возможность настройки пользователем внешнего вида приложения.

# Список использованных источников

1. Date C. J. A Guide to the SQL Standard. – Addison-Wesley Longman Publishing Co., Inc., 1989.
2. Seetha H. et al. A gui based application for pdf processing tools using python & customtkinter //Int. J. Res. Appl. Sci. Eng. Technol. – 2023.
3. Бизли Д. Python //Подробный справочник. СПб. – 2010.
4. Прохоренок Н., Дронов В. Python 3. Самое необходимое, 2-е изд. – БХВ-Петербург, 2019.
5. Столбов М. Е., Салихов Р. К., Кошкина Л. Ю. РАЗРАБОТКА ПРОГРАММНОГО ПРИЛОЖЕНИЯ ДЛЯ РАСЧЕТА ВЕСОРОСТОВОГО ПОКАЗАТЕЛЯ //ББК 34.7 Текстовое электронное издание. – 2023. – С. 577.

# Приложение. Листинг приложения

Листинг 7.1 - Файл main.py

|  |
| --- |
| import customtkinter as ctk  from tkcalendar import Calendar  import os  import json  from TasksWindow import AddTaskWindow, EditTaskWindow  import sqlite3  import threading  from others import get\_tasks\_by\_date  import time  from datetime import datetime  from Task import Task  from tkinter import Event  # Настройка глобальных параметров CustomTkinter  ctk.set\_appearance\_mode("System") # Темный/светлый режим  ctk.set\_default\_color\_theme("blue") # Цветовая тема  TASKS\_DB = "tasks.db"  CONFIG\_FILE = "config.json"  DATE\_FORMAT = "%y-%m-%d %H:%M"  class OrganizerApp(ctk.CTk):  """Главное окно приложения."""  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.tasks\_db = TASKS\_DB  self.config\_file = CONFIG\_FILE  self.date\_format = DATE\_FORMAT  self.cur\_tasks = []  # Настройки главного окна  self.title("Органайзер")  self.geometry("900x600")  create\_table()  # Элементы интерфейса  self.create\_interface()  self.today\_task = []  def create\_interface(self) -> None:  """Создает главное меню приложения."""  self.grid\_rowconfigure(0, weight=1)  self.grid\_rowconfigure(1, weight=0)  self.grid\_columnconfigure(0, weight=1)  self.grid\_columnconfigure(1, weight=1)  # Календарь  self.calendar = Calendar(self, selectmode="day", date\_pattern="yy-mm-dd", locale='ru')  self.calendar.grid(row=0, column=1, sticky="nsew", padx=10, pady=10)  self.calendar.bind("<<CalendarSelected>>", self.update\_task\_list)  # Список задач  self.task\_listbox = ctk.CTkTextbox(self, width=400)  self.task\_listbox.configure(state="normal")  self.task\_listbox.bind("<Double-1>", command=self.open\_task\_editor)  self.task\_listbox.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)  # Кнопка для добавления задачи  self.add\_task\_button = ctk.CTkButton(self, text="Добавить задачу", command=self.open\_add\_task\_window)  self.add\_task\_button.grid(row=1, column=0, pady=10)  # Инициализация задач для текущей даты  self.update\_task\_list()  self.today\_task = get\_tasks\_by\_date(self.calendar.get\_date(), self.tasks\_db)  # Чекбокс для включения/выключения уведомлений  self.notifications\_enabled\_flag = False  self.notifications\_enabled = ctk.BooleanVar()  self.stop\_check\_time = threading.Event()  self.load\_config()  self.checkbox = ctk.CTkCheckBox(  self,  text="Включить уведомления",  variable=self.notifications\_enabled,  command=self.save\_config,  onvalue=True,  offvalue=False  )  self.checkbox.grid(row=1, column=1, pady=10)  self.check\_thread = threading.Thread(target=self.check\_time, daemon=True)  self.check\_thread.start()  def update\_task\_list(self, event=None) -> None:  """Обновляет список задач в левой части главного меню."""  self.cur\_tasks = []  selected\_date = self.calendar.get\_date()  self.task\_listbox.configure(state="normal")  self.task\_listbox.delete("1.0", "end")  for i, task in enumerate(get\_tasks\_by\_date(selected\_date, self.tasks\_db)):  status\_marker = "[✔] " if task.done == 1 else ""  task\_text = f"{i + 1}. {status\_marker}{task.start\_time}-{task.end\_time}: {task.name}\n"  self.task\_listbox.insert("end", task\_text)  self.cur\_tasks.append(task)  self.task\_listbox.configure(state="disabled")  def load\_config(self) -> None:  """Загружает файл с состоянием флага о получении уведомлений."""  if os.path.exists(self.config\_file):  with open(self.config\_file, "r") as f:  config = json.load(f)  self.notifications\_enabled.set(config.get("notifications\_enabled", True))  if self.notifications\_enabled.get():  self.stop\_check\_time.clear()  else:  self.stop\_check\_time.set()  def save\_config(self) -> None:  """Сохраняет состояние флага и запускает поток проверки уведомлений."""  self.notifications\_enabled\_flag = self.notifications\_enabled.get()  config = {"notifications\_enabled": self.notifications\_enabled\_flag}  with open(self.config\_file, "w") as f:  json.dump(config, f)  if self.notifications\_enabled\_flag:  self.stop\_check\_time.clear()  self.check\_thread = threading.Thread(target=self.check\_time, daemon=True)  self.check\_thread.start()  else:  self.stop\_check\_time.set()  def open\_task\_editor(self, event: Event) -> None:  """Открывает окно редактора задач."""  try:  index = int(self.task\_listbox.index("@%d,%d" % (event.x, event.y)).split(".")[0]) - 1  task = self.cur\_tasks[index]  EditTaskWindow(self, task)  except Exception as e:  print("Ошибка при открытии задачи:", e)  def open\_add\_task\_window(self) -> None:  """Открывает окно "Добавить задачу"."""  AddTaskWindow(self)  def check\_time(self) -> None:  """Поток для проверки необходимости отправить уведомление."""  while not self.stop\_check\_time.is\_set():  if self.notifications\_enabled\_flag:  self.today\_task = get\_tasks\_by\_date(datetime.today().strftime("%y-%m-%d"), self.tasks\_db)  now = datetime.now()  for task in self.today\_task:  if (now >= datetime.strptime(task.date\_notif, self.date\_format)) and not task.notified:  self.show\_notification(task)  connection = sqlite3.connect(self.tasks\_db)  cursor = connection.cursor()  cursor.execute(  'UPDATE tasks SET notified = ? WHERE id = ?',  (1, task.id)  )  connection.commit()  connection.close()  time.sleep(5)  def show\_notification(self, task: Task) -> None:  """Создает окно уведомления и показывает его пользователю"""  notification\_window = ctk.CTkToplevel(self)  notification\_window.title("Напоминание")  notification\_window.geometry("300x150")  text = f"{task.name}\n{task.description}\n{task.start\_time}-{task.end\_time}\n{task.tags}"  label = ctk.CTkLabel(notification\_window, text=text, font=("Arial", 14))  label.pack(pady=20)  close\_button = ctk.CTkButton(notification\_window, text="Закрыть", command=notification\_window.destroy)  close\_button.pack(pady=10)  def create\_table() -> None:  """Создает базу данных с таблицей tasks."""  conn = sqlite3.connect(TASKS\_DB)  cursor = conn.cursor()  cursor.execute("""  CREATE TABLE IF NOT EXISTS tasks (  id INTEGER PRIMARY KEY AUTOINCREMENT,  name TEXT NOT NULL,  description TEXT,  start\_time TEXT,  end\_time TEXT,  date TEXT NOT NULL,  tags TEXT,  done INTEGER NOT NULL,  notified INTEGER NOT NULL,  date\_notif TEXT NOT NULL  )  """)  conn.commit()  conn.close()  if \_\_name\_\_ == "\_\_main\_\_":  app = OrganizerApp()  app.mainloop() |

Листинг 7.2 - Файл Task.py

|  |
| --- |
| import logging  from typing import Any  logging.basicConfig(level=logging.ERROR)  class Task:  def \_\_init\_\_(self, name: str, description: str, start\_time: str, end\_time: str, date: str,  tags: str, done=0, notified=0, date\_notif='', id=0):  self.id = id  self.name = name  self.description = description  self.start\_time = start\_time  self.end\_time = end\_time  self.date = date  self.tags = tags  self.done = done  self.notified = notified  self.date\_notif = date\_notif  def \_\_eq\_\_(self, other: Any) -> bool:  if not isinstance(other, Task):  return False  return (  self.name == other.name and  self.description == other.description and  self.start\_time == other.start\_time and  self.end\_time == other.end\_time and  self.date == other.date and  self.tags == other.tags and  self.done == other.done and  self.notified == other.notified and  self.date\_notif == other.date\_notif and  self.id == other.id  )  def to\_dict(self) -> dict:  """Возвращает задачу в виде словаря с ключами-атрибутами."""  return {  "name": self.name,  "description": self.description,  "start\_time": self.start\_time,  "end\_time": self.end\_time,  "date": self.date,  "tags": self.tags,  "done": self.done,  "notified": self.notified,  "id": self.id,  "date\_notif": self.date\_notif  }  @staticmethod  def from\_dict(data: dict):  """Возвращает задачу с параметрами из словаря."""  try:  return Task(  data["name"],  data["description"],  data["start\_time"],  data["end\_time"],  data["date"],  data['tags'],  data['done'],  data['notified'],  data['date\_notif'],  data['id']  )  except KeyError as e:  logging.error(f"Ошибка: отсутствует ключ {e} в задаче. Задача будет пропущена.")  return None |

Листинг 7.3 - Файл TasksWindow.py

|  |
| --- |
| import sqlite3  from others import show\_error\_popup  import customtkinter as ctk  from Task import Task  from datetime import datetime, timedelta  import re  class BaseTaskWindow(ctk.CTkToplevel):  """Общий предок для окон добавления/редактирования задач."""  def \_\_init\_\_(self, parent):  super().\_\_init\_\_(parent)  self.parent = parent  self.date = None  # Поля для ввода данных  self.name\_entry = ctk.CTkEntry(self, placeholder\_text="Название задачи")  self.name\_entry.pack(pady=5, fill="x", padx=20)  self.description\_entry = ctk.CTkEntry(self, placeholder\_text="Описание задачи")  self.description\_entry.pack(pady=5, fill="x", padx=20)  self.start\_time\_entry = ctk.CTkEntry(self, placeholder\_text="Время начала (ЧЧ:MM)")  self.start\_time\_entry.pack(pady=5, fill="x", padx=20)  self.end\_time\_entry = ctk.CTkEntry(self, placeholder\_text="Время окончания (ЧЧ:MM)")  self.end\_time\_entry.pack(pady=5, fill="x", padx=20)  self.date\_notif\_entry = ctk.CTkEntry(self, placeholder\_text="Напомнить за (ДД:ЧЧ:MM)")  self.date\_notif\_entry.pack(pady=5, fill="x", padx=20)  self.tags\_entry = ctk.CTkEntry(self, placeholder\_text="Теги (через запятую)")  self.tags\_entry.pack(pady=5, fill="x", padx=20)  def get\_data(self) -> Task:  """Возвращает задачу по данным из полей ввода."""  name = self.name\_entry.get().strip()  if not name:  raise ValueError("Для создания задачи необходимо дать ей имя")  start\_time = self.start\_time\_entry.get().strip()  if not re.fullmatch(r"^\d{2}:\d{2}$", start\_time) and start\_time != '':  raise ValueError("Задайте время начала в формате ЧЧ:ММ")  end\_time = self.end\_time\_entry.get().strip()  if not re.fullmatch(r"^\d{2}:\d{2}$", end\_time) and end\_time != '':  raise ValueError("Задайте время окончания в формате ЧЧ:ММ")  date\_notif = self.date\_notif\_entry.get().strip()  if not re.fullmatch(r"^\d{2}:\d{2}:\d{2}$", date\_notif) and date\_notif != '':  raise ValueError("Задайте время напоминания перед началом события в формате ДД:ЧЧ:ММ")  date\_notif = self.get\_date\_notif(start\_time, date\_notif)  description = self.description\_entry.get().strip()  tags = self.tags\_entry.get().strip()  task = Task(name, description, start\_time, end\_time, self.date, tags, done=0, notified=1,  date\_notif=date\_notif)  return task  def get\_date\_notif(self, start\_time: str, date\_notif: str) -> str:  """Вычисляет date\_notif для task перед добавлением в БД."""  if len(start\_time):  initial\_date = self.date + ' ' + start\_time  else:  initial\_date = self.date + ' 00:00'  base\_date = datetime.strptime(initial\_date, self.parent.date\_format)  if len(self.date\_notif\_entry.get().strip()):  days, hours, minutes = map(int, date\_notif.split(":"))  time\_delta = timedelta(days=days, hours=hours, minutes=minutes)  date\_notif = (base\_date - time\_delta).strftime(self.parent.date\_format)  return date\_notif  else:  return initial\_date  class AddTaskWindow(BaseTaskWindow):  """Окно создание задачи."""  def \_\_init\_\_(self, parent):  super().\_\_init\_\_(parent)  self.title("Добавить задачу")  self.geometry("400x400")  self.date = self.parent.calendar.get\_date()  # Кнопка для сохранения задачи  self.save\_button = ctk.CTkButton(self, text="Сохранить задачу", command=self.add\_task)  self.save\_button.pack(pady=20)  def add\_task(self) -> None:  """Добавляет задачу в базу данных."""  try:  task = super().get\_data()  except ValueError as error:  show\_error\_popup(f"{error}")  return  if not (len(self.date\_notif\_entry.get().strip())):  notified = 1  else:  notified = 0  task.notified = notified  conn = sqlite3.connect(self.parent.tasks\_db)  cursor = conn.cursor()  cursor.execute("""  INSERT INTO tasks (name, description, start\_time, end\_time, date, tags, done, notified, date\_notif)  VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)  """,  (task.name, task.description, task.start\_time, task.end\_time, task.date, task.tags, task.done,  task.notified, task.date\_notif)  )  conn.commit()  conn.close()  self.parent.update\_task\_list()  self.destroy()  class EditTaskWindow(BaseTaskWindow):  """Окно редактирования задачи."""  def \_\_init\_\_(self, parent, task):  super().\_\_init\_\_(parent)  self.task = task  self.title("Редактировать задачу")  self.geometry("400x400")  self.date = task.date  # Поля для редактирования данных  self.name\_entry.insert(0, task.name)  if task.description:  self.description\_entry.insert(0, task.description)  if task.start\_time:  self.start\_time\_entry.insert(0, task.start\_time)  if task.end\_time:  self.end\_time\_entry.insert(0, task.end\_time)  if not bool(task.notified):  self.date\_notif\_entry.insert(0, self.get\_what\_insert\_in\_date\_notif(task))  if task.tags:  self.tags\_entry.insert(0, task.tags)  self.done\_status = ctk.IntVar(value=task.done)  self.checkbox = ctk.CTkCheckBox(  self,  text="Выполнено",  variable=self.done\_status,  onvalue=1,  offvalue=0  )  self.checkbox.pack(pady=20)  # Кнопки для сохранения или удаления задачи  self.save\_button = ctk.CTkButton(self, text="Сохранить изменения", command=self.update\_task)  self.save\_button.pack(pady=10)  self.delete\_button = ctk.CTkButton(self, text="Удалить задачу", fg\_color="red", command=self.delete\_task)  self.delete\_button.pack(pady=10)  def update\_task(self) -> None:  """Обновляет задачу в БД."""  try:  task = super().get\_data()  except ValueError as error:  show\_error\_popup(f"{error}")  return  task.done = self.done\_status.get()  if self.task.notified == 0 or len(self.date\_notif\_entry.get().strip()) != 0:  task.notified = 0  connection = sqlite3.connect(self.parent.tasks\_db)  cursor = connection.cursor()  cursor.execute('UPDATE tasks SET name = ?, description = ?, start\_time = ?, end\_time =?, tags=?, done=?, '  'notified=?, date\_notif=? WHERE id = ?',  (task.name, task.description, task.start\_time, task.end\_time, task.tags,  task.done, task.notified, task.date\_notif, self.task.id)  )  connection.commit()  connection.close()  self.parent.update\_task\_list()  self.destroy()  def delete\_task(self) -> None:  """Удаляет задачу из БД."""  connection = sqlite3.connect(self.parent.tasks\_db)  cursor = connection.cursor()  cursor.execute('DELETE FROM tasks WHERE id = ?', (self.task.id,))  connection.commit()  connection.close()  self.parent.update\_task\_list()  self.destroy()  def get\_what\_insert\_in\_date\_notif(self, task: Task) -> str:  """Вычисляет вставку в поле времени для уведомления."""  if task.start\_time:  dt1 = datetime.strptime(task.date + ' ' + task.start\_time, self.parent.date\_format)  else:  dt1 = datetime.strptime(task.date + ' 00:00', self.parent.date\_format)  dt2 = datetime.strptime(task.date\_notif, self.parent.date\_format)  delta = dt1 - dt2  total\_seconds = delta.total\_seconds()  days = int(total\_seconds // 86400)  hours = int((total\_seconds % 86400) // 3600)  minutes = int((total\_seconds % 3600) // 60)  result = f"{days:02}:{hours:02}:{minutes:02}"  return result |

Листинг 7.4 - others.py

|  |
| --- |
| import sqlite3  import customtkinter as ctk  from Task import Task  def show\_error\_popup(message: str) -> None:  """Создает окно ошибки с текстовым сообщением внутри."""  error\_window = ctk.CTkToplevel()  error\_window.title("Ошибка")  error\_window.geometry("320x150")  error\_window.grab\_set()  label = ctk.CTkLabel(error\_window, text=message, font=("Arial", 14))  label.pack(pady=20)  close\_button = ctk.CTkButton(error\_window, text="Закрыть", command=error\_window.destroy)  close\_button.pack(pady=10)  def get\_tasks\_by\_date(date: str, task\_db: str) -> list:  """Возвращает список tasks соответствующих переданной дате."""  conn = sqlite3.connect(task\_db)  cursor = conn.cursor()  query = """  SELECT name, description, start\_time, end\_time, date, tags , done, notified, date\_notif, id  FROM tasks  WHERE date = ?  """  cursor.execute(query, (date,))  rows = cursor.fetchall()  conn.close()  tasks = []  for row in rows:  task = Task(\*row)  tasks.append(task)  return tasks |