МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»**

Кафедра Математического и компьютерного моделирования

Разработка таск-трекера с использованием языка Python, пакета PySide и формата хранения данных JSON

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 5 курса 561 группы

направление 09.03.03 — Прикладная информатика

механико-математического факультета

Ардазишвили Лауры Саматовны

Научный руководитель
доцент, к.ф.-м.н. С.В. Иванов

Зав. кафедрой
зав. каф., д.ф.-м.н., доцент Ю.А. Блинков

Саратов 2023

Введение. Многие трекеры задач являются частью более крупного программного обеспечения для управления задачами, которое также включает в себя отслеживание времени, ведение отчетности и автоматизацию, помогает оставаться в курсе повторяющихся задач.

Трекер задач – это инструмент, представляющий собой список задач и целей, позволяющий назначать условия, контролирующие процесс деятельности.

Целью данной бакалаврской работы, является разработка таск-трекера с использованием языка Python, пакета PySide и формата хранения данных JSON.

Согласно цели, были поставлены следующие задачи:

1. Обосновать необходимость использования информационной системы.
2. Описать используемые технологии – язык программирования Python, технологию разработки интерфейса PySide, технологию хранения данных JSON.
3. Спроектировать приложение.
4. Описать процесс разработки информационной системы.

Основное содержание работы. Таск-трекер представляет собой экземпляр информационных систем управления проектами (ИСУП). ИСУП – это одно или несколько программных средств, включающих процессы сбора и использования данных проекта. Такие программы поддерживают планирование, исполнение и закрытие задач и организуют информационный поток.

Хороший трекер задач упростит рабочий процесс, сделает его более прозрачным, поможет планировать и следить за выполнением проектов из самых разных профессиональных областей. Таск-менеджер — это программа для управления проектами, которая позволяет централизованно руководить задачами и их своевременным выполнением.

Таск-трекеры — это распространенное решение, которое используют компании по всему миру уже не первое десятилетие. И если первые программы такого рода выглядели как простейшая канбан-доска для планирования, то со временем функционал усложнился.

В соответствии с рисунком 1, представлена целесообразность таск-трекера.

Рисунок 1 — Назначение современных таск-трекеров

Архитектурно приложение представляет собой монолит. Монолитная архитектура — это традиционная модель программного обеспечения, которая представляет собой единый модуль, работающий автономно и независимо от других приложений. Структура монолитного программного обеспечения представлена в соответствии с рисунком 2.

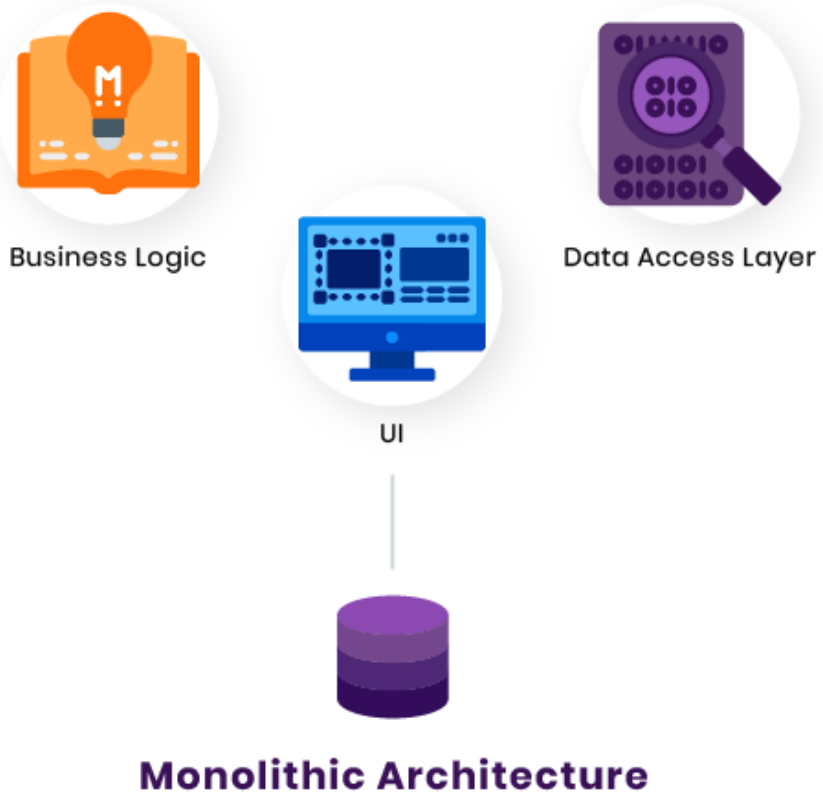

Рисунок 2 — Структура монолитной архитектуры

Работа с дизайном приложения выполнялась в Qt Creator. Qt Creator – это программа для работы с графическим фреймворком Qt. Данная IDE имеет удобный графический интерфейс для разработки приложений, основанный на применении Qt Widgets и QML, а также поддерживает огромное количество компиляторов.

Создание интерфейса приложения в соответствии с рисунком 3.

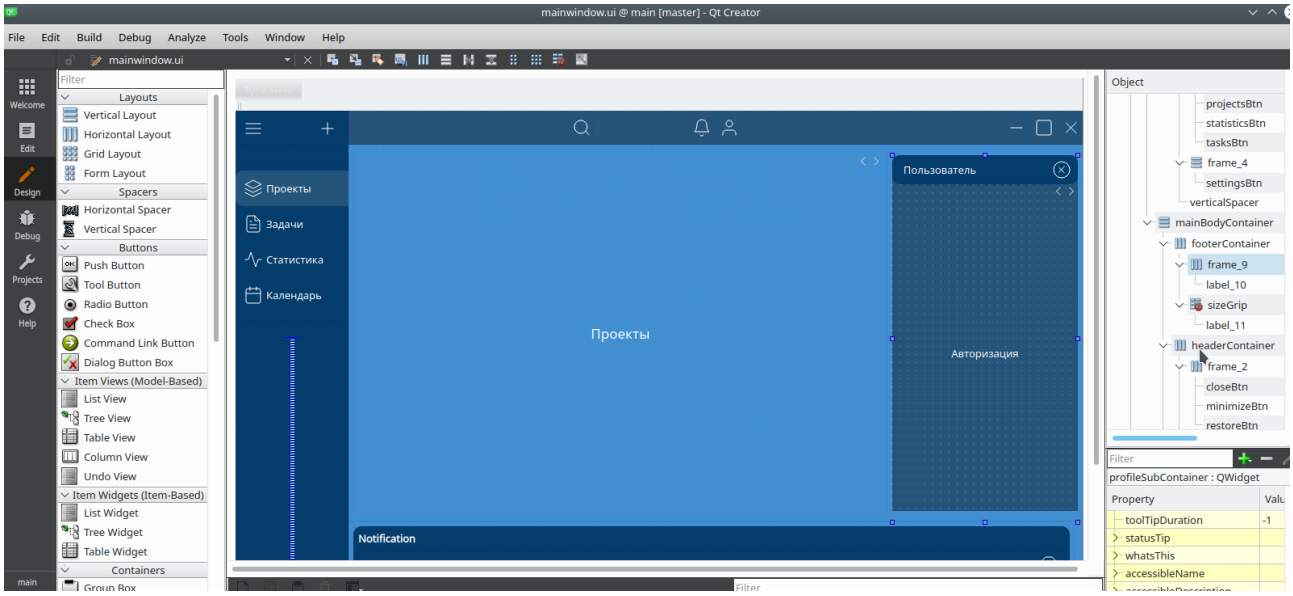

Рисунок 3 — Интерфейс приложения в Qt Creator

При проектировании приложения необходимо учитывать информацию, полученную при анализе предметной области, и отразить логику информационной системы. В этом случае, используется диаграмма вариантов использования – use case diagram. Она приведена в соответствии с рисунком 4.

Действующее лицо – пользователь. Пользователь инициирует следующие варианты использования:

- Авторизация – форма предоставления доступа к учетной записи системы;
- Регистрация – форма создания учетной записи в системе;
- Ошибка – сообщение об ошибке при неверно введенных данных;
- Создание – пользователь создает задачу или проект;
- Редактирование – пользователь редактирует задачу или проект;
- Информация – получение развернутой информации о задаче или проекте;

- Статистика – отображение статистики в соответствии со статусом;
- Статистика проектов – количество актуальных проектов за день, неделю, месяц, год;
- Статистика задач – количество актуальных задач за день, неделю, месяц, год.

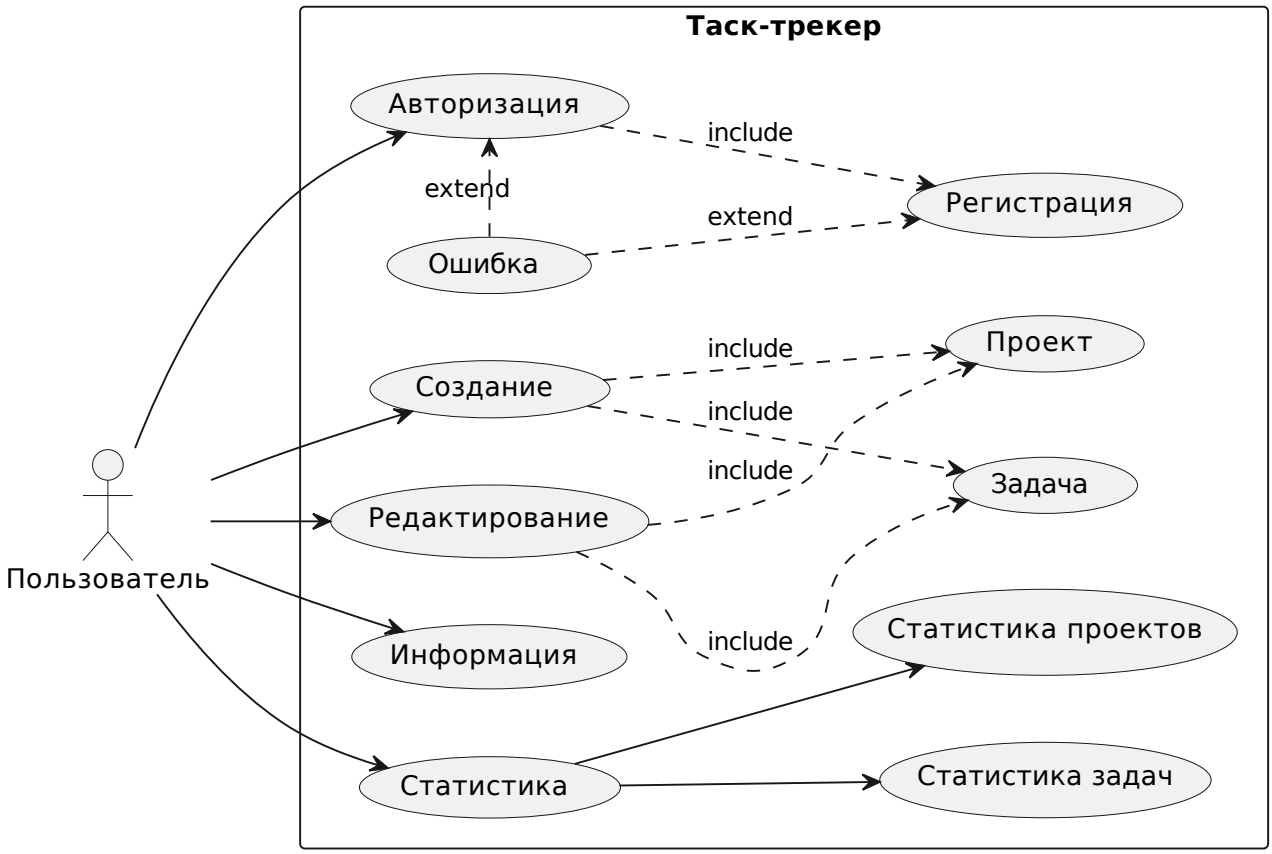

Рисунок 4 — Use case диаграмма

Процесс создания информационной системы обусловлен построением интеграции сложной и простой системы. Развитие от элементарной системы к более комплексной производит огромное количество иерархий. Структурно иерархии представлены классами и объектами. Функционирование любой информационной системы нуждается в выстраивании правильного взаимодействия ее элементов. Необходимые правила и критерии взаимодействия представлены в объектно-ориентированном программировании. Процессу и проблемам проектирования приложения при помощи объектно-ориентированной парадигмы посвящено большое количество работ. В соответствии с рисунком 5 приведена диаграмма классов.

Описание используемых классов:

1. Класс `MainWindow` – отвечает за отображение графического интерфейса. Для получения информации используются следующие классы: `TaskManager`, `ProjectManager`, `ProjectStatistics`, `TaskStatistics`, `UserManager`.
2. Классы `User`, `Task`, `Project` содержат информацию о пользователе, задачах и проектах.
3. `TaskManager` – содержит список задач. Его контракт – сообщать информацию о задачах по требованию.
4. `ProjectManager` – содержит список проектов. Его контракт – сообщать информацию о проектах.
5. `UserManager` – содержит список пользователей. Его контракт – сообщать информацию о пользователях.
6. `TaskStatistics` и `ProjectStatistics` подсчитывают задачи и проекты с определённым статусом за разные временные промежутки. Для этого они используют `TaskManager`/`ProjectManager`, так как вся информация о задачах и проектах должна быть получена только через них.
7. Классы `PlanPeriod`, `Status` и `Priority` – это перечисления, чтобы хранить типы временных промежутков, допустимые статусы и приоритеты.
8. `ProjectEncoder`, `UserEncoder` – эти классы помещают объекты задач и проектов в файлы projects.json и users.json. Сохраняют все объекты актуальными после выхода из приложения.

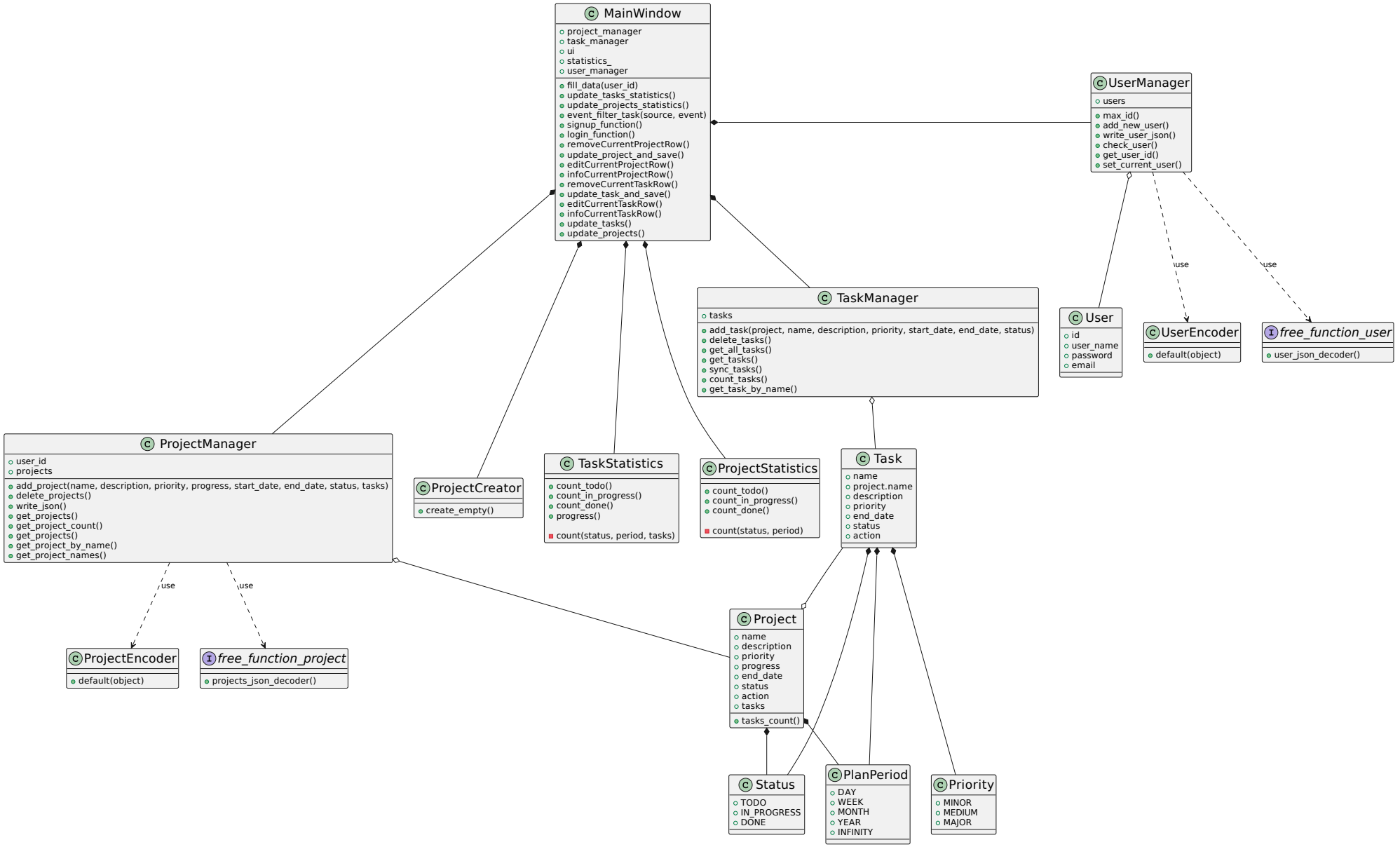

Рисунок 5 — Диаграмма классов

Для работы с JSON использован модуль `json`. Данный модуль предоставляет интерфейс для конвертации в формат JSON таких объектов, как, например, `dict`. Тип данной операции – `encoding`. Также, модуль `json` позволяет конвертировать имеющиеся данные в формате JSON в объекты программы написанной на Python. Тип такой операции – `decoding`. Простейшие методы для работы с `encoding/decoding` операциями – соответственно `dump/dumps` и `load/loads`. В соответствии с рисунком 6 представлены `ProjectEncoder` и `UserEncoder`.

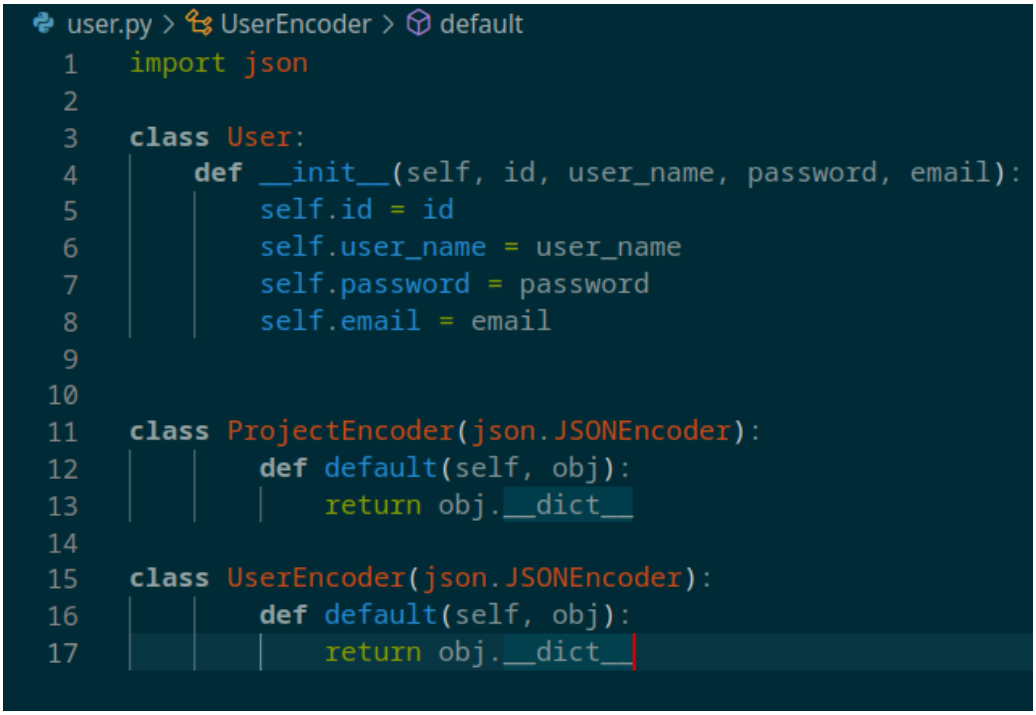

Рисунок 6 — Методы конвертации

В связи с тем, что для текущей задачи требуется работа с более сложными объектами, чем имеющаяся структура данных `dict`, то модуль `json` может быть параметризован соответствующими энкодером и декодером. Например, класс `Project` помимо полей имеет и метод, поэтому `ProjectEncoder` используется в качестве энкодера для данного класса. Так как данные, хранящиеся в формате JSON, включают как проекты, так и задачи, то при декодировании надо описать, каким образом считывать тот или иной объект в JSON модели. Этой цели служит функция `json_decoder`.

В общем случае компонент разрабатываемого монолитного приложения отвечающий за хранение данных может использовать различные хранилища

и иструменты, включая привычные реляционные базы данных. Использование JSON для данной цели служит преимущественно интересам прототипирования с целью ускорения разработки на начальном этапе проекта.

Для работы с таск-трекером необходимо зарегистрироваться или авторизоваться. Для этого реализовано окно пользователя. До входа в учетную запись таблицы не заполнены данными. Функции создания задачи или проекта недоступны. В соответствии с рисунком 7 приведена форма авторизации пользователя. При нажатии на кнопку «Нет аккаунта? Зарегистрируйтесь» открывается форма регистрации. Неверные данные при авторизации и регистрации вызывают сообщение об ошибке.

Таск-трекер представляет собой динамически откликающееся окно. Основными разделами являются «Проекты», «Задачи», «Статистика». Задачи и проекты записываются в табличную форму. Создание происходит в бокс-формах, в нижней половине страницы.

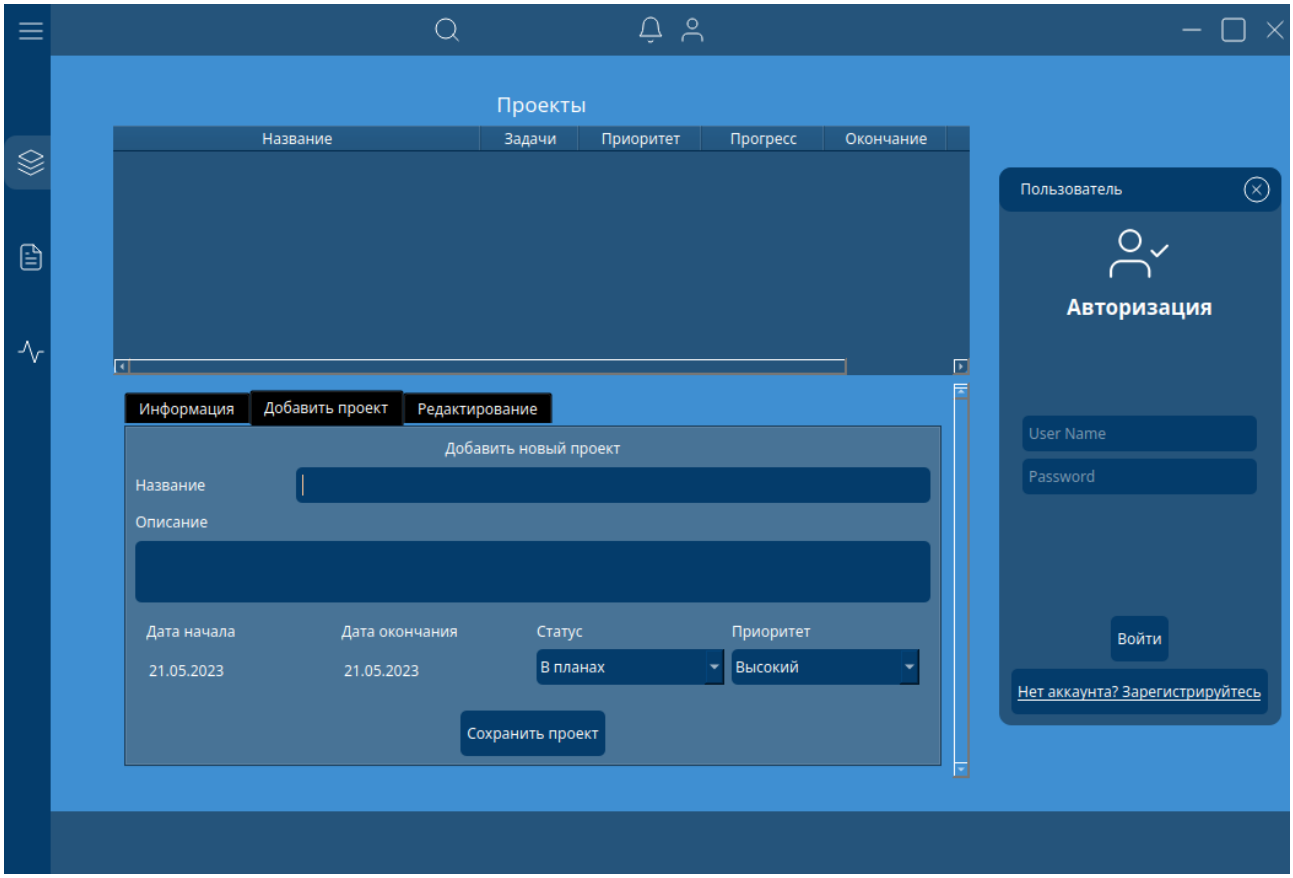

Рисунок 7 — Начало работы в приложении

Бокс-форма создания проекта имеет следующие поля заполнения: название, описание, дата начала, дата окончания, статус и приоритет. Установка дат представляет собой виджет редактирования даты, в формате «dd//mm//YY», срабатывает на прокрутку колеса мыши или клик(нажатие). Комбобокс «Статус» содержит список: «В планах», «В процессе», «Завершено». Комбобокс «Приоритет» содержит: «Высокий», «Средний», «Низкий».

В приложении можно управлять как проектами, так и задачами: создавать, редактировать и удалять. Проекты состоят из задач. Проект также может быть пустым до тех пор, пока в нем нет ни одной задачи. Выбор подходящего проекта осуществляется из комбобокса с названиями существующих проектов. Как проекты, так и задачи могут иметь различные приоритеты и статусы. Набор допустимых значений статуса и приоритета одинаков для проектов и задач. Создание новой задачи в соответствии с рисунком 8.

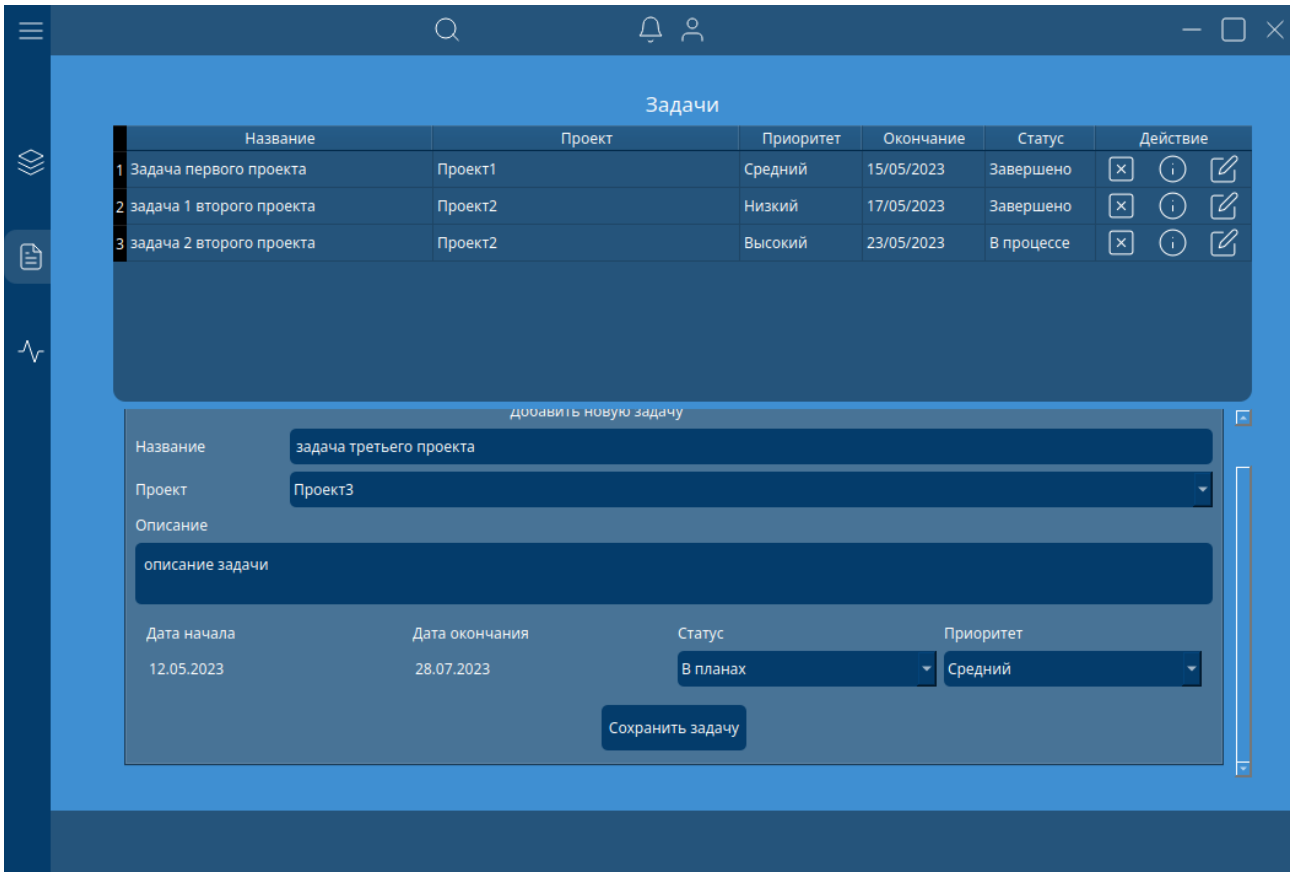

Рисунок 8 — Создание задачи

Заключение. Большинство людей подходят к отслеживанию времени с макроперспективы, стремясь получить общее представление о том, куда уходит их время. Но когда дело доходит до повышения производительности, важна каждая деталь. Отслеживание работы вплоть до выполнения задачи может предоставить массу практически полезной информации, которая поможет улучшить процессы, распределение ресурсов и прибыльность.

В процессе выполнения данной работы было проведено исследование информационной системы управления проектами. Освоены дополнительные инструменты проектирования и создания приложений, в которых ведется организационная и сопроводительная деятельность. Были применены ранее изученные материалы из курса обучения.

Большая часть работы проводилась над анализом и обработкой данных. Были изучены преимущества и недостатки технологии хранения данных JSON, технологии разработки пользовательского интерфейса PySide и языка Python.

В ходе бакалаврской работы были решены следующие задачи:

- Обоснована необходимость использования информационной системы.
- Описаны используемые технологии – язык программирования Python, технология хранения данных JSON.
- Спроектировано приложение.
- Описан процесс разработки информационной системы.

Цель – разработка desktop приложения – была описана в виде задачи, задача была фрагментирована на подзадачи:

1. Графическая часть.
2. Проектирование и реализация классов.
3. Работа с хранилищем данных JSON.

Подзадачи были выполнены с помощью таких инструментов как язык программирования Python, фреймворк PySide, средство для проектирования дизайна Qt Creator. В результате бакалаврской работы была достигнута цель – разработка таск-трекера с использованием языка Python, пакета PySide и формата хранения данных JSON. Помимо этого, были улучшены навыки программирования, получены навыки построения и поддержки архитектуры монолитного приложения, выяснены другие типы архитектур, а также

их плюсы и минусы. Получено понимание, какие последствия влечет то или иное неудачное решение в процессе построения графического интерфейса, либо недостаточно продуманных классов. Выяснена возрастающая роль тестирования в процессе разработки приложения.