**ФГБОУ ВО   
Уфимский университет науки и технологий**

**Кафедра ВМиК**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 100 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 90 |  |  |  |  |  |  |  |  |  |  |  |  |
| 80 |  |  |  |  |  |  |  |  |  |  |  |  |
| 70 |  |  |  |  |  |  |  |  |  |  |  |  |
| 60 |  |  |  |  |  |  |  |  |  |  |  |  |
| 50 |  |  |  |  |  |  |  |  |  |  |  |  |
| 40 |  |  |  |  |  |  |  |  |  |  |  |  |
| 30 |  |  |  |  |  |  |  |  |  |  |  |  |
| 20 |  |  |  |  |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

Исследование возможностей языковых моделей

**ОТЧЕТ**

**по научно-исследовательской работе**

**по** Производственной практике

(*наименование дисциплины*)

|  |
| --- |
| Производственная практика 2 |
| (обозначение документа) |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Группа |  |  | Фамилия, И., О. | Подпись | Дата | Оценка |
| МО-325Б |  |
|  |  |
| Студенты | | | Агафонов Р.В.,  Лепоринский Г.А.,  Шарыгин М.С.,  Ярмухаметов Б.И. |  |  |  |
| Преподаватель | | | Ризванов Д.А. |  |  |  |
| Принял | | |  |  |  |  |

**Уфа 2025 г****.**

Содержание

[1 Цель работы 3](#_Toc202143786)

[1.1 Общее задание 3](#_Toc202143787)

[1.2 Индивидуальное задание 3](#_Toc202143788)

[2 Основная часть (общее задание) 4](#_Toc202143789)

[2.1 Вариант и языковые модели 4](#_Toc202143790)

[2.2 Идея приложения 4](#_Toc202143791)

[2.3 Функции приложения 5](#_Toc202143792)

[2.4 Реализация приложения 6](#_Toc202143793)

[2.5 Исследование эффективности языковой модели «DeepSeek» 6](#_Toc202143794)

[2.6 Исследование эффективности языковой модели «Qwen» 12](#_Toc202143795)

[2.7 Исследование эффективности языковой модели «Perplexity» 17](#_Toc202143796)

[2.8 Исследование эффективности языковой модели «GigaChat» 22](#_Toc202143797)

[2.9 Сравнение эффективности языковых моделей 26](#_Toc202143798)

[3 Основная часть (индивидуальное задание) 27](#_Toc202143799)

[3.1 Языковые модели 27](#_Toc202143800)

[3.2 Задачи 27](#_Toc202143801)

[3.3 Исследование возможностей языковой модели «DeepSeek» 27](#_Toc202143802)

[3.4 Исследование возможностей языковой модели «GigaChat» 30](#_Toc202143803)

[3.5 Сравнение возможностей языковых моделей 32](#_Toc202143804)

[4 Вывод 33](#_Toc202143805)

[5 Приложение 34](#_Toc202143806)

[5.1 Созданные приложения 34](#_Toc202143807)

[5.2 Языковые модели 34](#_Toc202143808)

# Цель работы

## Общее задание

Выбрать 3-4 языковые модели (по 1 на каждого члена группы) – «ChatGPT», «Claude», «Perplexity», «Qwen», «DeepSeek», «GigaChat», «YandexGPT», и т.д. В соответствии с выбранным вариантом провести исследование эффективности решения задачи путем последовательного задания промптов и проверки результата. Цель – добиться желаемого результата, поставленного в задаче.

## Индивидуальное задание

Выбрать любые 2 языковые модели. Провести исследование их возможностей на примере решения трех задач, не связанных с программированием (например, бытовых): описать в промпте ситуацию и исследовать качество, точность и полноту ответов моделей.

# Основная часть (общее задание)

## Вариант и языковые модели

Для начала необходимо выбрать вариант и языковые модели, пусть это будут:

* Вариант:
  + Исследование эффективности при описании новых (для вас) технологий, на основе которых можно с нуля создать рабочее приложение на «Python»;
* Языковые модели:
  + «DeepSeek» [1];
  + «Qwen» [2];
  + «Perplexity» [3];
  + «GigaChat» [4].

## Идея приложения

Чтобы выполнить задание, нужно определить на примере создания какого приложения мы будем оценивать эффективность работы языковых моделей. Выберем идею для такого приложения также с помощью нейронных сетей, а именно путем задания двух запросов:

1. «Предложи 5 идей приложений, которые можно создать на Python»;
2. «Выдели тезисы из предыдущего ответа (т.е. оставь только идеи, а их описания исключи)».

В результате получаем идеи, представленные в таблице 2.1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **DeepSeek** | **Qwen** | **Perplexity** | **GigaChat** |
| **1** | Бот для автоматизации рутинных задач | Приложение для отслеживания привычек | Генератор паролей | Приложение для анализа расходов |
| **2** | Приложение для анализа настроений в соцсетях | Конвертер валют с интерфейсом | Веб-скрейпер | Простое мобильное приложение-календарь мероприятий |
| **3** | Персональный финансовый трекер | Личный дневник / заметки | Игра «Угадай число» | Чат-бот для автоматизации поддержки клиентов |
| **4** | Генератор мемов или контента | Скриншотер с функциями автоматизации | Приложение для заметок с графическим интерфейсом | Система управления задачами «ToDo List» |
| **5** | Мини-игра | Мини-игра: "Угадай число" или "Виселица" | Чат-бот | Программа распознавания текста с изображений |

Таблица 2.1 – Идеи для приложения

Проведя анализ результата, можно отметить, что идеи, предложенные разными языковыми моделями, достаточно схожи. На их основе мы выберем следующую концепцию – приложение «ежедневник-список задач» («To-Do Journal»), которое реализуем только на основе того кода, который сгенерируют выбранные нами нейронные сети.

## Функции приложения

Для начала с помощью промпта «основные функции To-Do Journal» получим от каждого из используемых ИИ[[1]](#footnote-1) список базовых задач такого приложения. Осуществив сравнение этих списков, выберем самые важные, которые и реализуем путем применения языковых моделей. Выбранные функции:

* создание, редактирование и удаление заметок и задач;
* категоризация заметок и задач;
* просмотр списка существующих заметок и задач;
* установка дедлайнов (сроков выполнения);
* поиск по категориям и ключевым словам;
* статистика и анализ по выполненным / невыполненным задачам.

## Реализация приложения

После того, как мы выбрали идею приложения, а также функции, которые оно должно осуществлять, перейдем к его реализации. Воплощать функции будем последовательно, чтобы иметь возможность проверить корректность работы каждой из них, а в случае обнаружения проблем и ошибок не было препятствий для их исправления, так как будет известно в какой части кода находится ошибка.

Будем создавать с помощью каждой языковой моделью отдельное приложение, анализируя в процессе реализации её эффективность.

## Исследование эффективности языковой модели «DeepSeek»

Создадим приложение с помощью ответов «DeepSeek». Для начала используем два промпта, которые позволят создать основу интерфейса программы:

* 1. «Создай интерфейс на Python, в котором в верхней части страницы будут три кнопки: "Задачи", "Заметки" и "Статистика", необходимые для перехода между страницами (кроме этого ничего не делай)»;
  2. «Подробно объясни созданный код из предыдущего ответа, поясняя каждую строку».

Благодаря таким запросам языковая модель создаёт небольшой код и подробно и доступно объясняет его, вследствие чего у не знакомого с библиотекой «tkinter» пользователя не возникнет проблем с пониманием написанного нейросетью. В результате получим скелет интерфейса приложения, который выглядит так, как показано на рисунке 2.1.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.1 – Основа интерфейса приложения

Следующими запросами доработаем страницу «Задач»:

* 1. «Создай на странице "Задачи" основное поле, в котором будет находиться таблица, отображающая задачи; в этой таблице должно быть 4 колонки: "Задача", "Категория", "Срок выполнения" и "Выполнено"» – позволяет создать область, в которой будут отображаться задачи, а также проверить корректность их вывода на экран;
  2. «Добавь в верхней части страницы "Задачи" текст "Поиск:", после которого будет следовать поле для ввода; поиск должен начинаться при любом изменении поля для ввода и проходить только по содержимому колонки "Задача"» и «Подробно объясни код в "# Панель поиска", "# Привязываем обработчик к изменению текста" и "def filter\_tasks"» – необходимы, чтобы организовать поиск по задачам, а также для понятного объяснения пользователю способа работы функций поиска;
  3. «Добавь под "Поиском" над "Основным полем" текст "Категория", после которого будет выпадающий список из категорий "Работа", "Учёба", "Дом", "Здоровье" и "Другое"; при выборе категории должны отображаться только те задачи, которые соответствуют данной категории» – нужен для добавления поиска по категория (работает аналогично строке поиска);
  4. «Добавь внизу страницы "Задачи" кнопки "Добавить", "Редактировать", "Удалить", "Отметить выполненной", под кнопками должно быть: "Задача:" и поле для ввода; - "Категория:" и выпадающий список для выбора; "Срок выполнения:" и поле для ввода даты (на данном этапе не нужно делать логику функций)» – этот промпт позволяет сделать интерфейс для управления задачами, а логика работы [интерфейса] специально будет добавлена после задания других соответствующих запросов, что позволит последовательно разрабатывать приложение небольшими вставками кода, способствуя пониманию пользователя;
  5. «Добавь логику для кнопки "Отметить выполненной"» (и т.д. со всеми созданными кнопками в предыдущем пункте) – запрос позволяет создать логику работы для каждой ранее созданной кнопки.

С помощью дополнительных промптов (как, например, «объясни self.tasks\_tree.pack») пользователь может разобраться в отдельных участках кода, которые ему непонятны. ИИ даст подробный ответ, структурированный по пунктам, в котором объяснит смысл технологии (например, в данном случае «.pack()» является методом геометрии (менеджер размещения), который корректно размещает элемент – таблицу задач – на странице).

На рисунке 2.2 показано, как выглядит финальная версия страницы «Задачи».

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.2 – Страница «Задачи»

Далее применяя знания о технологиях, полученные из промптов [2.1, 2.2, 2.3, 2.4, 2.5], заполним необходимыми объектами страницу «Заметки»:

* «Поиск» и «поле ввода», «Категория» и «выпадающий список» в верхней части страницы;
* «Основное поле», которое содержит информацию о заметках, в середине странице;
* Кнопки «Добавить», «Редактировать», «Удалить», «Заметка» и «поле ввода», «Категория» и «выпадающий список» в нижней части страницы.

В процессе реализации работы объектов страницы «Заметки» языковой модели также задавались запросы, направленные на уточнение знаний о технологиях. ИИ подробно отвечал на запросы пользователя, что говорит об его эффективности при описании.

В итоге страница «Заметок» выглядит следующим образом (рисунок 2.3).

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.3 – Страница «Заметки»

Так как данные о задачах и заметках необходимо сохранять при закрытии приложения, то с помощью запросов «Как сохранять записи и заметки при закрытии приложения?» и «Последовательно подробно объясни все функции для сохранения и загрузки данных» мы получаем от языковой модели код, который позволит записать данные в «JSON-файл», с подробными объяснениями.

Наконец добавим статистику по задачам и заметкам на вкладку «Статистика». Предварительно попросим нейросеть предложить варианты, по каким параметрам можно группировать данные о заметках и задачах. После этого для разработки используем следующие промпты:

* 1. «Создай в "Статистике" по функции для создания блоков в верхней и нижней половинах страницы (больше ничего не делай)» и «Раздели верхний блок на две части (больше ничего не делай)» и аналогичный запрос для правой части – позволяет разделить страницу на четыре части, чтобы в первых трёх создать блоки статистики для задач, а в последней – для заметок;
  2. «Добавь в левую верхнюю часть "Статистики" диаграмму категоризации задач» (и другие дополнительные запросы) – нужны, чтобы создать диаграмму, в которой будет представлены задачи по категориям;
  3. «Создай такой же график для заметок в правой верхней части» – аналогичный предыдущему запрос, позволяющий создать диаграмму категорий задач;
  4. «Создай в нижней левой части диаграмму, где вместо категорий будут выполненные и невыполненные задачи» – необходим для создания диаграммы, отображающей статистику по выполнению и невыполнению задач;
  5. «Реализуй просроченные задачи в нижней правой части» – позволяет реализовать диаграмму, в которой невыполненные задачи разделятся на просроченные, если время на выполнение истекло, и непросроченные в противном случае.

Итак, ознакомившись с объяснениями технологий от языковой модели, мы получили реализацию страницы «Статистика», представленную на рисунке 2.4.

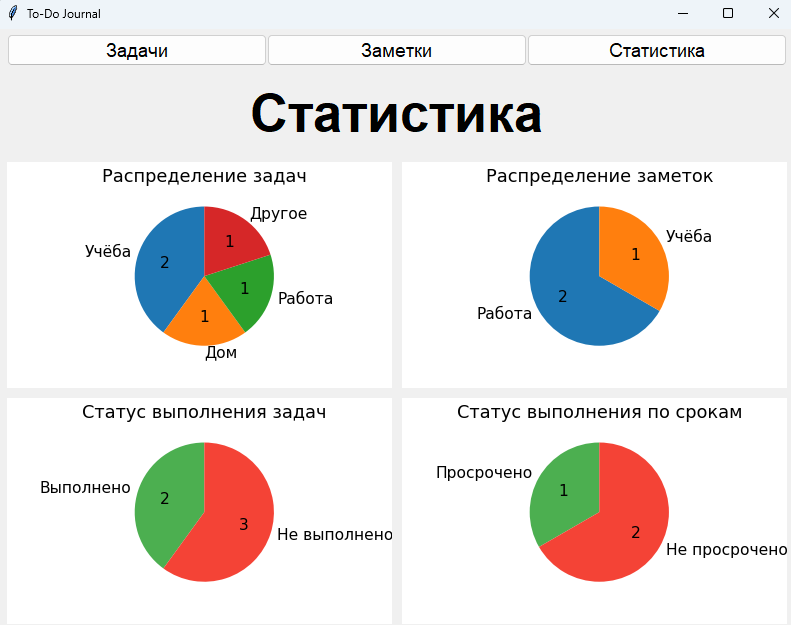


Рисунок 2.4 – Страница «Статистика»

В ходе исследования языковая модель «DeepSeek» проявила себя крайне эффективно. Ответы на вопросы были полными, подробными и понятными, а уточняющие и корректирующие [вопросы] в основном не требовались. Код, приводимый нейросетью, был рабочим и не требовал внесения правок со стороны пользователя. ИИ хорошо адаптируется к задаваемым промптам и отлично «понимает» их контекст, а, следовательно, его можно использовать для создания новых приложений на «Python» с нуля.

Созданное приложение можно посмотреть здесь: [1].

## Исследование эффективности языковой модели «Qwen»

Теперь создадим аналогичное приложение с помощью языковой модели «Qwen», оценивая в процессе эффективность описания создаваемого кода с точки зрения незнакомого пользователя.

Для начала с помощью промптов, аналогичных следующим [1.1, 1.2], создадим страницы «Задачи», «Заметки» и «Статистика». Затем с помощью запроса «Реализуй переход между страницами» получим от нейросети объяснения того, как добавить функционал переключения страниц, который дорабатываем дополнительными промптами, так как изначальный ответ не был полным. В конце добавим стиль для кнопок, а также вытянем их на всю ширину окна приложения, как показано на рисунке 2.5.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.5 – Основа интерфейса приложения

Перейдем к наполнению содержимым страницы «Задачи», используя запросы [2.1, 2.2, 2.3, 2.4, 2.5], при этом:

* дополнительными запросами корректируем объекты поиска и категории, чтобы настроить внешний вид (как, например, «Как вытянуть поле ввода по всей доступной ширине»);
* при создании блока управления задачами (кнопки «Добавить», «Редактировать» и т.д.) благодаря промпту «сделай без .grid()» нейросеть продолжила реализацию объектов с помощью упаковщика «.pack()», что позволило не использовать разметку страницы;
* используем такой запрос: «Добавь логику для кнопки "Редактировать", нужно, чтобы эта функция заполняла поля "Задача:", "Категория:" и "Срок выполнения:" данными выбранной задачи, после чего выбранная задача удаляется», в котором подробно описываем, как должна работать функция редактирования задач.

После задания языковой модели всех запросов мы получаем рабочую страницу «Задачи», которая изображена на рисунке 2.6.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.6 – Страница «Задачи»

Теперь реализуем работу страницы «Заметки», для чего попросим нейросеть выполнить следующие задачи:

* «На примере реализации страницы "Задачи", создай на странице "Заметки" те же элементы, при этом: таблица заметок должна содержать только два столбца: "Заметка" и "Категория"; отсутствует кнопка "Отметить выполненной" и соответствующий функционал; внизу страницы только: "Заметка:" с полем ввода и "Категория:" с полем выбора»;
* «Подробно и доступно объясни внесенные изменения».

В результате мы получим разработанную страницу «Заметки» (рисунок 2.7), работу которой подробно объяснила языковая модель.

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.7 – Страница «Заметки»

После этого с помощью запроса «Реализуй сохранение записей и заметок в один файл при закрытии приложения» мы получаем решение, которое позволяет сохранять записи после закрытия приложения и восстанавливать их при открытии.

Наконец реализуем диаграммы на странице «Статистика», для этого применим промпты [3.1, 3.2, 3.2, 3.3, 3.4], которые были описаны раньше. Как и в случаях, описанных выше, корректируем ответы языковой модели, чтобы получить похожий внешний вид (например, «Сделай так, чтобы диаграммы не выходили за пределы своего блока, а только полностью его заполняли» – нужен, чтобы графики правильно отображались в не полноэкранном режиме). Результат разработки представлен на рисунке 2.8.

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.8 – Страница «Статистика»

Данная языковая модель хорошо справилась с поставленными задачами, предлагая неплохие готовые решения и объясняя достаточно сложные концепции, но все же требовала задания дополнительных запросов, так как нередко даваемые ею ответы были не совсем полными. Однако важно отметить, что ответы уточняющие промпты были исчерпывающими и понятными, что означает, что «Qwen» подходит для эффективного описания технологий, связанных с программированием.

Созданное приложение можно посмотреть здесь: [2].

## Исследование эффективности языковой модели «Perplexity»

Исследуем эффективность нейронной сети «Perplexity» путем создания приложения «To-Do Journal», которое в итоге постараемся выполнить аналогичным сделанным ранее.

Как и в предыдущих пунктах, сначала создадим основу для интерфейса с помощью запросов [1.1, 1.2]. Результат приведен на рисунке 2.9. Затем реализуем переход между страницами с помощью промптов:

* «Реализуй переход между страницами»;
* «Приведи только изменения»;
* «Сделай без grid».

Также применим «Добавь стиль для верхних кнопок», чтобы сделать их [кнопки] более красивыми.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.9 – Основа интерфейса приложения

Применим запросы [2.1, 2.2, 2.3, 2.4, 2.5] для создания объектов и функций для работы страницы «Задачи». Также используем приведенные ниже дополнительные сообщения для корректировки работы:

* «Добавь "Все" в категории для отображения всех задач» – нужно, так как изначально общей категории создано не было;
* «Сделай, чтобы таблица обновлялась и при изменении категории (приведи только изменения)» – нужно для корректной работы поиска;
* «Сделай, чтобы "Задача", "Категория", "Срок выполнения" были в разных строках (приведи только изменения)» и «Сделай без grid» – используем для приведения к схожему с другими приложениями внешнему виду;
* «Добавь проверку на заполненность всех полей и всплывающее окно с предупреждением, если хотя бы одно поле не заполнено» и «» – применяем для того, чтобы сделать работу кнопки «Добавить» правильной.

После задания всех вопросов нейронной сети мы получаем следующий вид страницы «Задачи» (рисунок 2.10).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.10 – Страница «Задачи»

Далее перейдем к заполнению объектами и функциями страницы «Заметки», для чего используем запрос, который был применен в предыдущем пункте [2.6]:

* «На примере реализации страницы "Задачи", создай на странице "Заметки" те же элементы, при этом: таблица заметок должна содержать только два столбца: "Заметка" и "Категория"; отсутствует кнопка "Отметить выполненной" и соответствующий функционал; внизу страницы только: "Заметка:" с полем ввода и "Категория:" с полем выбора».

После задания промпта и внесения небольших корректировок в отступы, получим заполненную страницу «Заметки», приведенную на рисунке 2.11.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.11 – Страница «Заметки»

По аналогии с предыдущими пунктами [2.6] и [2.7] добавим возможность сохранения данных таблиц при закрытии приложения и их [данных] восстановлении при открытии, для чего используем сообщение «Реализуй сохранение записей и заметок в один файл при закрытии приложения». Также в процессе реализации возможности сохранения данных задаём дополнительные запросы для объяснения работы приводимых ИИ технологий.

Наконец, используя запросы [3.1, 3.2, 3.3, 3.4, 3.5], сделаем рабочей страницу «Статистика». Объединим эти запросы в один следующим образом:

* «Создай на странице статистики 4 круговых диаграммы:
  + категоризация задач,
  + категоризация заметок,
  + выполненные / невыполненные задачи,
  + просроченные / непросроченные задачи (выбирать только из невыполненных)».

После попросим языковую модель привести подробные и доступные объяснения используемых технологий, чтобы разобраться в коде. На рисунке 2.8 представлена страница «Статистика».

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.12 – Страница «Статистика»

Подводя итог исследования языковой модели «Perplexity», можно сказать, что она очень хорошо справилась с задаваемыми промптами, так как ответы на вопросы были корректными, а создаваемый код не содержал ошибок. Нейросеть быстро и точно отвечала не запросы, поэтому отметим, что её использование отлично поможет пользователю в ознакомлении с новыми информационными технологиями.

Созданное приложение можно посмотреть здесь: [3].

## Исследование эффективности языковой модели «GigaChat»

Наконец реализуем наше приложение с помощью запросов языковой модели «GigaChat», чтобы исследовать её эффективность.

Для начала с помощью промптов [1.1, 1.2] создадим основу интерфейса программы; после задания получим следующий результат, приведенный на рисунке 2.13.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.13 – Основа интерфейса приложения

Затем запросами [2.1, 2.2, 2.3, 2.4, 2.5] разработаем страницу «Задачи», при этом изменим некоторые запросы и добавим новые:

* «Добавь в верхней части страницы "Задачи" текст "Поиск:", после которого будет следовать поле для ввода; поиск должен начинаться при любом изменении поля для ввода и проходить только по содержимому колонки "Задача". Справа от поля для ввода добавь выпадающий список с категориями: "Работа", "Учёба", "Дом", "Здоровье" и "Другое"» – объединяет запросы [2.2] и [2.3], что позволяет лучше организовать интерфейс поиска задач;
* «При смене значения выпадающего списка сделай так, чтобы в «treeview» показывались задачи соответствующей категории» – необходим для улучшения работы поиска задач.

В итоге у нас будет готовая страница «Задачи», которая показана на рисунке 2.14.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.14 – Страница «Задачи»

Далее перейдем к реализации страницы «Заметки», для чего используем следующий запрос:

* «На примере реализации страницы "Задачи", создай на странице "Заметки" те же элементы, при этом: таблица заметок должна содержать только два столбца: "Заметка" и "Категория"; отсутствует кнопка "Отметить выполненной" и соответствующий функционал; внизу страницы только: "Заметка:" с полем ввода и "Категория:" с полем выбора»;

Языковая модель не смогла в полной мере выполнить предыдущий запрос, поэтому используем дополнительные:

* «Реализуй функционал кнопок для добавления, редактирования и удаления»;
* «При добавлении проверь, что все вводимые поля заполнены, и исправь IndexError при удалении последней заметки на странице».

На рисунке 2.15 представлена реализованная страница «Заметки».

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.15 – Страница «Заметки»

После этого с помощью запросов, как, например: «Реализуй сохранение записей и заметок в один файл при закрытии приложения» мы получаем решение, которое позволяет сохранять записи после закрытия приложения и восстанавливать их при открытии.

Наконец реализуем графики в разделе «Статистика», применив запросы:

* «Создай на странице статистики 4 диаграммы, из которых 3 будут относиться к задачам, а последняя к заметкам»;
* «Сделай все диаграммы круговыми».

В итоге страница «Статистика» будет выглядеть так, как показано на рисунке 2.16.

Изображение выглядит как текст, снимок экрана, диаграмма, Красочность

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2.16 – Страница «Статистика»

Языковая модель «GigaChat» оказалась неэффективной для описания технологий, на основе которых можно с нуля создать рабочее приложение на «Python». Ответы нейросети были в основном некорректными, а код, генерируемый ею, – требовал внесения исправления для работы. Корректирующие и уточняющие запросы редко давали положительного результата, так как ответы ИИ так и оставались неполными и неправильными. Неподготовленному пользователю создать приложение с нуля будет достаточно трудно, поэтому на данный момент «GigaChat» не подходит для данных целей.

Созданное приложение можно посмотреть здесь: [4].

## Сравнение эффективности языковых моделей

Подводя итог оценки эффективности, можно сказать, что «DeepSeek» [2.5] и «Perplexity» [2.7] справились с описанием технологий лучше всего, «Qwen» [2.6] также достаточно неплохо отвечал на вопросы, но при этом были нужны уточняющие запросы, а «GigaChat» [2.8] решил данный вопрос хуже всего.

# Основная часть (индивидуальное задание)

## Языковые модели

Для начала выберем языковые модели, которые мы будем исследовать. Так как с общим заданием нейронная сеть «DeepSeek» справилась отлично, а «GigaChat», наоборот, – достаточно плохо, то и возьмем эти модели для сравнения их возможностей решения задач, несвязанных с программированием.

## Задачи

Теперь решим, на примере каких задач, мы будем исследовать возможности ИИ. Пусть это будут:

1. Планирование выходных в Санкт-Петербурге;
2. Организация вечернего досуга для компании друзей;
3. Планирование похода в продуктовый магазин.

## Исследование возможностей языковой модели «DeepSeek»

Начнем исследование возможностей «DeepSeek» с «планирования выходных в Санкт-Петербурге», для чего зададим следующий промпт: «Спланируй два выходных дня в Санкт-Петербурге, в рамках которых каждый день нужно посещать 3-4 достопримечательности, причем в первый день нужно ознакомиться с историческими памятниками, а во второй - с современными». В результате нейросеть даёт исчерпывающий ответ о местах, которые можно посетить в городе (рисунок 3.1). После запросами:

* «Где можно остановиться на ночь?»,
* «Как передвигаться между достопримечательностями?»,
* «Где лучше всего питаться?»

получаем множество вариантов, где лучше всего остановиться (разные отели от премиум-класса до бюджетных вариантов, а также необычные места) и питаться (список самых разных возможных вариантов), а также как передвигаться между выбранными местами для посещения (советы по организации передвижения в историческом центре – пешком, и современных местах).

Изображение выглядит как текст, документ, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 3.1 – Ответ языковой модели

Теперь попросим языковою модель помочь в «организации вечернего досуга для компании друзей», для чего зададим ей соответствующий запрос. В ответ она [языковая модель] предлагает несколько вариантов:

1. Тематические вечеринки;
2. Активный отдых;
3. Творческие и необычные форматы;
4. Уютные посиделки.

Попросим нейросеть поподробнее раскрыть второй ответ, на что получим следующее:

* квест в реальности,
* лазертаг / пейнтбол,
* боулинг / бильярд,
* ночной велопоход / прогулка по городу,
* батутный парк / скалодром.

Выберем «батутный парк» и зададим промпт «Как организовать поход в батутный парк?», после чего получим рекомендации по поиску и подбору парка (использование онлайн карт для ознакомления с вариантами, их ценами и отзывами) и организации похода в него (бронирование времени и подготовка компании).

Наконец займемся «планированием похода в продуктовый магазин». Используем запрос «Как правильно спланировать поход в продуктовый магазин?» и получим ответ, состоящий из пунктов:

1. Составьте список покупок;
2. Установите бюджет;
3. Выберите подходящее время;
4. Возьмите с собой всё необходимое;
5. В магазине: придерживайтесь списка;
6. После магазина.

Каждый пункт снабжен комментариями, поясняющими его, что делает этот ответ качественным и полным. Также нейросеть предлагает и альтернативные варианты, например, такой как использование приложений магазина для формирования пакета продуктов, который можно забрать самому либо получить дома с помощью доставки курьером. Затем задаем запрос «Где лучше всего покупать продукты в Уфе?», чтобы узнать о магазинах, в которых можно приобрести продукты. Нейросеть приводит примеры магазинов («Магнит», «Пятёрочка», «Перекрёсток» и так далее), супермаркетов («Лента», «Ашан», «Metro»), рынков и сервисов доставок.

Итак, подводя итог, можно сказать, что «DeepSeek» отлично справился с задачами, несвязанными с программирование, дал полные, подробные и качественные ответы по всем трём темам.

## Исследование возможностей языковой модели «GigaChat»

Теперь исследуем возможности языковой модели «GigaChat». Начнем с первой задачи – «Планирование выходных в Санкт-Петербурге». Сначала зададим такой же запрос, как и в предыдущем пункте: «Спланируй два выходных дня в Санкт-Петербурге, в рамках которых каждый день нужно посещать 3-4 достопримечательности, причем в первый день нужно ознакомиться с историческими памятниками, а во второй - с современными»; после этого получаем ответ:

* День 1:
  + Эрмитаж;
  + Петропавловская крепость;
  + Михайловский замок;
* День 2:
  + Новая Голландия;
  + Дворец культуры имени Ленсовета;
  + Этажи.

Каждое предложенное место для посещения сопровождается объяснением, почему стоит его посетить. Затем задаем дополнительные запросы, которые описаны в предыдущем пункте (про отель, питание и передвижение), на что получаем полные и качественные ответы:

* приведены отели разных ценовых категорий, начиная с бюджетных и заканчивая роскошными вариантами;
* описаны все возможности, доступные для передвижения в городе; приведены их достоинства и недостатки;
* представлены разные варианты, разделенные на различные кухни (русская, европейская, стрит-фуд и так далее).

Далее спросим у нейронной сети, как «организовать вечерний досуг для компании друзей», на что получим следующие предложения:

* настольные игры;
* кинопросмотр;
* квест-комната;
* фотосессия;
* вечеринка с пиццей и напиткам;
* мастер-классы;
* спортивные соревнования;
* тематическая вечеринка.

Допустим, что выбрали «кинопросмотр», поэтому зададим промпт «Как устроить кинопросмотр», чтобы получить рекомендации по его организации: подготовка помещения, выбор фильма и покупка закусок, также каждый из этих пунктов имеет пояснения по его выполнению. С помощью запроса «Предложи какой-нибудь триллер, а также оптимальные закуски и напитки» мы получаем точный ответ: фильм – «Остров проклятых», а также рекомендации по закускам и напиткам: попкорн, чипсы, пицца и газировка, фруктовые соки и так далее.

Наконец обратимся к языковой модели с вопросом «планирования похода в продуктовый магазин», после чего получим следующие советы:

1. Составьте список покупок заранее;
2. Проверьте холодильник и кухонные шкафы;
3. Определите бюджет;
4. Выберете удобное время посещения магазина;
5. Используйте скидки и акции;
6. Обратите внимание на сроки годности;
7. Рассчитайте количество необходимых продуктов;
8. Убедитесь в наличии необходимого инвентаря для хранения продуктов.

Каждый совет сопровождается комментариями, что делает ответ ИИ полным. Затем спросим у нейросети «В каких магазинах лучше всего покупать продукты в Уфе?»; в ответ она [нейросеть] приводит различные магазины (популярные в России торговые сети), а также местные Уфимские рынки.

В итоге можно отметить, что языковая модель «GigaChat» успешно справилась с решением несвязанных с программированием задач, так как её ответы содержали много примеров и пояснений, что делало их точными, качественными и полными.

## Сравнение возможностей языковых моделей

Исходя из всех примеров решения языковыми моделями несвязанных с программированием задач, можно сделать вывод и сказать, что обе модели (и «DeepSeek» [3.3], и «GigaChat» [3.4]) отлично подходят для решения таких задач.

# Вывод

В ходе производственной практики мы провели исследование возможностей языковых моделей для решения различных задач (связанных и несвязанных с программированием), а также сравнили их эффективности.

# Приложение

## Созданные приложения

1. <https://github.com/Cyber-Limon/Mathematical_support___Industrial_practice_2/blob/main/App_on_DeepSeek.py>
2. <https://github.com/Cyber-Limon/Mathematical_support___Industrial_practice_2/blob/main/App_on_Qwen.py>
3. <https://github.com/Cyber-Limon/Mathematical_support___Industrial_practice_2/blob/main/App_on_Perplexity.py>
4. <https://github.com/Cyber-Limon/Mathematical_support___Industrial_practice_2/blob/main/App_on_GigaChat.py>

## Языковые модели

1. <https://deepseek.com/>
2. <https://chat.qwen.ai/>
3. <https://www.perplexity.ai/>
4. <https://giga.chat/>

1. ИИ – искусственный интеллект [↑](#footnote-ref-1)