Geekbrains

Разработка веб-приложения для управления заметками с интеграцией нейросетевых алгоритмов анализа текста

Backend-разработчик. Специалист
Андреев Василий Андреевич

Санкт-Петербург
2024

Содержание

Введение

Современные реалии и высокий темп жизни предполагают, что человеку, для комфортного существования, необходимо иметь возможность мгновенного доступа к информации, которая будет понятна и легка в усвоении. Для этого, в различных сферах жизни, происходит развитие и модернизация множества систем. В данном контексте особую актуальность приобретает разработка веб-приложений для управления заметками, способных обеспечивать удобный, интуитивно понятный и функционально насыщенный интерфейс для работы с текстовой информацией.

Веб-приложения для управления заметками — это программные приложения, доступные через веб-браузер, предназначенные для создания, редактирования, хранения и организации текстовой информации. Эти приложения предоставляют пользователям возможность систематизировать личные и профессиональные записи, обеспечивая доступ к ним с любого устройства, подключенного к интернету.

Данный дипломный проект призвана сформулировать и реализовать комплексные научно — технические решения для создания такого приложения.

Актуальность работы заключается в том, что с каждым годом объем информации, с которым сталкивается современный человек, неуклонно возрастает. Заметки и личные записи играют важную роль в организации рабочего процесса, учебной деятельности и повседневной жизни. Традиционные методы управления заметками, такие как бумажные записи или примитивные цифровые приложения, часто оказываются недостаточно эффективными для систематизации, поиска и обработки большого объема информации. В связи с этим возникает потребность в создании более интегрированных и интеллектуальных решений, способных облегчить

процесс управления заметками и повысить производительность пользователей.

Задачи дипломного проекта:

1. Провести анализ аналогов и существующих решений;
2. Определить требования к разрабатываемому приложению;
3. Проектирование архитектуры системы;
4. Разработка пользовательского интерфейса;
5. Реализация и интеграция функциональных модулей;
6. Тестирование и оптимизация;
7. Оценка эффективности и дальнейшее развитие.

Цель дипломного проекта: разработка веб-приложения для управления заметками с интеграцией нейросетевых алгоритмов анализа текста.

Объект выпускной квалификационной работы: веб-приложение для управления заметками с интеграцией нейросетевых алгоритмов анализа текста.

Предмет выпускной квалификационной работы: проблема организации удобного, функционального и надежного инструмента для управления заметками, соответствующего современным требованиям и ожиданиям.

Теоретическая значимость исследования: результаты исследовательской части могут быть использованы для создания логичного и удобного веб-приложения для управления заметками с интеграцией нейросетевых алгоритмов анализа текста.

Практическая значимость работы: результатом внедрения проекта будет создание более совершенной и удобной среды, которая позволит пользователям управлять заметками и проводить создавать их при помощи неросетевого анализа текста.

При работе над дипломным проектом, использовалось следующее программное обеспечение:

1. Visual Studio Code;
2. Git;
3. DBeaver;
4. Ollama;
5. ChatGPT;
6. Microsoft Office Word;

Состав команды: Андреев Василий Андреевич (разработчик, дизайнер, тестировщик).

ГЛАВА 1. Теоретическая часть

1.1. Обзор существующих решений

В настоящее время существует множество приложений и систем для управления заметками, которые предоставляют пользователям разнообразные функциональные возможности. Эти системы варьируются по сложности, ориентированы на различные платформы и обладают специфическими особенностями, направленными на удовлетворение широкого спектра пользовательских потребностей. В данном разделе проводится сравнительный анализ ряда существующих решений на рынке с целью выявления их преимуществ и недостатков, а также определения возможностей для дальнейшего улучшения разрабатываемого веб-приложения для управления заметками.

Таблица 1

Основные приложения и системы для управления заметками, их преимущества и недостатки

Название	Преимущества	Недостатки
Notion	1. Универсальность: можно использовать как для создания заметок, так и для управления проектами и базами данных. 2. Возможность совместной работы: доступ к заметкам можно предоставить другим пользователям. 3. Поддержка структурирования информации в виде страниц и блоков.	1. Кривая обучения: для новых пользователей может быть сложно разобраться в интерфейсе. 2. Может быть избыточным для простых заметок.

Evernote	1. Удобный интерфейс с возможностью создания заметок разных форматов (текст, изображения, аудио). 2. Возможность организации заметок в блокноты и использование тегов. 3. Мощные функции поиска.	1. Бесплатный план имеет ограниченные возможности по синхронизации устройств. 2. Некоторые пользователи отмечают, что приложение стало менее интуитивным.
Microsoft OneNote	1. Интеграция с другими продуктами Microsoft (Office, Outlook). 2. Возможность создания заметок в свободном формате, включая рукописные заметки. 3. Бесплатный доступ на разных устройствах.	1. Интерфейс может показаться загроможденным. 2. Ограниченные функции для поиска и организации в больших заметках.
Google Keep	1. Простота использования и интуитивный интерфейс. 2. Хорошая интеграция с другими сервисами Google. 3. Поддержка совместной работы и возможность создания напоминаний.	1. Ограниченные функции по форматированию текста. 2. Менее мощные возможности для организации

		заметок по сравнению с другими приложениями.
Obsidian	1. Возможность создания заметок в формате Markdown. 2. Поддержка внутренней связности заметок через ссылки, что удобно для ведения баз знаний. 3. Полная локальная синхронизация и отсутствие необходимости в облаке.	1. Потребность в некотором круглом процессе настройки. 2. Не подходит для пользователей, ищущих простоту и легкость.
Workflowy	1. Уникальная структура древовидного оформления заметок. 2. Идеально подходит для структурирования мыслей и идей. 3. Простота и минимализм интерфейса.	1. Не всем подходит подход с древовидной структурой. 2. Ограниченные возможности для мультимедийных заметок.

На основании анализа существующих решений можно выделить несколько направлений для потенциального улучшения разрабатываемого веб-приложения. На основе анализа аналогов можно опираться на следующие решения при разработке веб-приложения:

1. Удобный и интуитивный интерфейс: простота использования и интуитивный интерфейс - один из их главных плюсов.

Необходимо разработать интерфейс, который будет интуитивным и простым в использовании, даже для новичков, как в Google Keep.

2. Универсальность и гибкость: важно предоставить пользователям возможность создавать разные типы контента (тексты, таблицы, списки задач, медиа и т.д.), как в Notion.

3. Поиск и организация: мощный поисковой функционал, поддержка тегов и папок. Необходимо реализовать систему тегов для удобной организации заметок, а также функцию поиска, которая поддерживает поиск не только по тексту, но и по другим параметрам (например, тегам, датам), как в Evernote.

4. Интеграция с другими сервисами: интеграция с другими популярными сервисами (Google Drive, Dropbox, Slack, календарь и т.д.) добавит функциональность и удобство, как в OneNote.

5. Форматирование текста и мультимедийные элементы: разнообразие инструментов для форматирования текста, а также возможности добавления и редактирования медиа-файлов (изображения, видео, файлы). Это позволит пользователям делать заметки не только информативными, но и визуально привлекательными, как в Notion.

6. Совместная работа: инструменты для командной работы, включая возможность совместного редактирования заметок в реальном времени. Возможность оставлять комментарии и упоминать коллег также будет полезна, как в Notion и Evernote.

7. Поддержка работы оффлайн: возможность доступа к заметкам и редактирования их без подключения к интернету с последующей синхронизацией при доступе к сети, как в Evernote.

8. Безопасность данных и конфиденциальность: реализация шифрования данных, функции резервного копирования и восстановления информации. Обеспечение конфиденциальности и

безопасности данных должно быть неотъемлемой частью любого приложения.

9. Поддержка разных языков: мультиязычная поддержка поможет расширить круг пользователей по всему миру.

10. Аналитика и статистика: предоставление пользователям информации о том, как они используют приложение – например, сколько часов в неделю они тратят на управление заметками, какие проекты занимают больше всего времени и т.д.

Вывод: для успешного веб-приложения для управления заметками важно сочетать сильные стороны аналогов и избегать их слабостей. Универсальный, интуитивный, хорошо интегрированный с другими сервисами и функционально насыщенный интерфейс поможет создать конкурентоспособный продукт. Инвестирование времени в разработку функций, таких как мощный поиск, совместная работа и поддержка оффлайн-режима также принесут немало пользы и привлекательности приложению.

1.2. Основные концепции нейросетевого анализа текста

Нейросетевой анализ текста представляет собой междисциплинарную область, сочетающую достижение искусственного интеллекта, лингвистики и обработки естественного языка (ОЕЯ) для решения широкого спектра задач, включая классификацию текстов, извлечение информации, машинный перевод и генерацию текста. Основополагающие концепции данного направления базируются на применении глубоких нейронных сетей, которые позволяют моделировать сложные зависимости в текстовых данных и обеспечивают высокую точность выполнения разнообразных задач на естественном языке.

В последние десятилетия интерес к нейросетевым методам анализа текстов на естественном языке значительно возрос благодаря разработкам в области искусственных нейронных сетей. Эти технологии предоставляют мощные инструменты для обработки, анализа и интерпретации текстовых данных, что находит широкое применение в различных областях, включая лингвистику, информацию retrieval, автоматический перевод и многие другие.

Таблица 2

Основные концепции нейросетевого анализа текста

Основная концепция	Описание
Искусственные нейронные сети (ИНС)	Математические модели, вдохновленные биологическими структурами мозга. Состоят из множества слоев искусственных нейронов, которые выполняют простейшие вычисления
Глубокое обучение	Подвид машинного обучения, использующий глубокие нейронные

	сети (ДНС) с несколькими скрытыми слоями. Позволяет автоматически извлекать высокоуровневые признаки из необработанных данных
Свёрточные нейронные сети (CNN)	Модели, более применимые для обработки данных с фиксированной сеткой, таких как изображения и текст, где они часто используются для задач классификации и извлечения признаков.
Рекуррентные нейронные сети (RNN)	Модели, лучше подходящие для последовательных данных, таких как текст или временные ряды, благодаря своей способности учитывать предыдущую информацию в последовательности.
Семантический анализ текста	Семантический анализ направлен на извлечение смысла из текста, что требует точной интерпретации и категоризации понятий и отношенческих структур. Нейросетевые модели, включая сети прямого распространения и глубокие ИНС, показали свою эффективность в выполнении этих задач. Они успешно используются для классификации текстов, извлечения информации и других операций, связанных с пониманием и интерпретацией текстов.

Кластеризация и тематический анализ	Одним из интересных аспектов нейросетевого анализа текста является кластеризация, которая позволяет автоматически разделять тексты на тематические группы. Разработанные нейросетевые алгоритмы кластеризации способны автоматически управлять уровнем детализации этих групп, что облегчает создание семантических баз знаний и повышает эффективность работы экспертов по текстовым данным.

Таким образом, нейросетевой анализ текста представляет собой сложный и многогранный процесс, который включает применение различных типов искусственных нейронных сетей и методов глубинного обучения для достижения точных и масштабируемых результатов. Его использование продолжает расширяться, предлагая новые возможности для автоматизации анализа текстов и улучшения качества обработки естественного языка.

Вывод: нейросетевой анализ текста представляет собой быстро развивающуюся область, способную преобразовать подходы к обработке и интерпретации текстовых данных. Внедрение и развитие глубоких нейронных сетей, механизмов внимания и эффективных методов преподготовки и дообучения моделей продолжают открывать новые горизонты в способности машин понимать и генерировать естественный язык.

1.3. Технологии и инструменты для разработки веб-приложений

Веб-приложения стали неотъемлемой частью повседневной жизни и бизнеса, обеспечивая доступ к широкому спектру услуг и ресурсов через Интернет. Их разработка требует применения различных технологий и инструментов, способных обеспечить функциональность, производительность, безопасность и масштабируемость. В данном пункте рассматриваются современные технологии и инструменты, используемые при создании веб-приложений.

Клиентская часть. Клиентская часть веб-приложений (фронтенд) отвечает за взаимодействие с пользователем, предоставляя интерфейс для ввода и вывода данных.

Существует множество языков программирования, используемых для разработки клиентской части веб-приложений. Некоторые из них:

1. HTML (HyperText Markup Language): HTML является стандартным языком разметки для создания веб-страниц. Он определяет структуру текста и мультимедийных документов в сети Интернет, обеспечивая базовую основу для разработки веб-интерфейсов. Современные версии, такие как HTML5, включают новые семантические элементы, графические возможности (Canvas и SVG) и поддержку мультимедиа (встроенное аудио и видео)
2. CSS (Cascading Style Sheets): CSS используется для описания внешнего вида и форматирования HTML-документов. Он позволяет отделить содержание документа от его представления, обеспечивая гибкость и контроль над визуальными аспектами веб-страниц. Современные технологии, такие как CSS3, включают расширенные возможности для анимации, переходов и оформления, а также обеспечивают адаптивный дизайн, подстраивающийся под различные устройства и экраны.

3. JavaScript: JavaScript является языком программирования высокого уровня, который позволяет создавать динамический и интерактивный контент на веб-страницах. В сочетании с HTML и CSS, он формирует основу современной веб-разработки. За последние годы JavaScript существенно эволюционировал, став ключевым элементом различных JavaScript-фреймворков и библиотек, таких как React, Angular и Vue.js, которые значительно упрощают процесс разработки сложных и масштабируемых веб-приложений.

Серверная часть. Серверная часть (бэкенд) веб-приложений занимается обработкой данных, взаимодействием с базами данных и другими серверными ресурсами, а также обеспечивает бизнес-логику приложения.

Существует множество языков программирования, используемых для разработки серверной части веб-приложений. Некоторые из них:

1. Node.js: Среда выполнения JavaScript вне браузера, позволяющая использовать JavaScript для написания серверной логики. Node.js отличается высокой производительностью и поддержкой неблокирующего ввода-вывода.
2. Python: Язык программирования, известный своей простотой и читаемостью, часто используемый в веб-разработке благодаря фреймворкам, таким как Django и Flask.
3. Ruby: Язык программирования, известный своим удобством для веб-разработчиков, особенно в сочетании с фреймворком Ruby on Rails, который предлагает конвенции над конфигурацией.
4. PHP: Один из наиболее распространённых языков для веб-разработки, особенно популярен благодаря своей простоте и широкому распространению.

Веб-серверы. Веб-серверы играют ключевую роль в обслуживании запросов клиентов и предоставлении им доступа к веб-контенту и приложениям. Среди наиболее популярных веб-серверов выделяются:

1. Apache HTTP Server: Один из старейших и наиболее используемых веб-серверов, известный своей гибкостью и расширяемостью.
2. Nginx: Веб-сервер и обратный прокси-сервер, отличающийся высокой производительностью и низким потреблением ресурсов.

Базы данных. Базы данных хранят и управляют данными веб-приложений. Существуют различные типы баз данных, каждый из которых имеет свои преимущества и случаи использования:

1. Реляционные базы данных (SQL): MySQL, PostgreSQL. Эти базы данных используют таблицы для хранения данных и поддерживают язык SQL для выполнения запросов.
2. Нереляционные базы данных (NoSQL): MongoDB, Cassandra. Эти базы данных предназначены для хранения и обработки больших объемов данных, обеспечивая гибкость и масштабируемость.

Фреймворки и библиотеки. Фреймворки и библиотеки играют важную роль в разработке веб-приложений, предлагая готовые решения для типичных задач и сокращая время разработки.

Фреймворки для клиентской части:

1. React: Библиотека для создания пользовательских интерфейсов, разработанная Facebook. Предоставляет виртуальный DOM, улучшая производительность и упрощая разработку сложных интерфейсов.
2. Angular: Фреймворк от Google, предлагающий комплексное решение для создания одностраничных приложений.

3. Vue.js: Прогрессивный фреймворк, который можно интегрировать по частям или использовать для полноценной разработки веб-приложений.

Фреймворки для серверной части:

1. Express.js: Минималистичный фреймворк для Node.js, который упрощает создание веб-приложений и API.
2. Django: Высокоуровневый фреймворк для Python, который следует принципу «батарейки включены» и предлагает множество встроенных возможностей для быстрой разработки.
3. Ruby on Rails: Фреймворк для языка Ruby, известный своим эмпирическим подходом к разработке и высокой продуктивностью.

Системы управления версиями. Системы управления версиями (VCS, Version Control Systems) необходимы для разработки веб-приложений в команде, обеспечивая контроль за изменениями кода и координацию работы между разработчиками.

1. Git: наиболее популярная система управления версиями, известная своей распределенной природой и гибкостью. Git используется совместно с платформами, такими как GitHub, GitLab, и Bitbucket, которые предлагают дополнительные инструменты для совместной разработки и управления проектами.

Инструменты для развертывания и оркестрации. Современные веб-приложения требуют инструментов для автоматизации процесса развертывания, обеспечения непрерывной интеграции и непрерывного развертывания (CI/CD).

1. Docker: Платформа для автоматизации развертывания приложений в контейнерах, обеспечивающая изоляцию и портативность.

2. Kubernetes: Система оркестрации контейнеров, предоставляющая инструменты для управления кластерами контейнеров и автоматизации их развертывания, масштабирования и поддержания работоспособности.

Вывод: Разработка веб-приложений требует использования широкого спектра технологий и инструментов, каждый из которых играет свою роль в создании эффективных, производительных и масштабируемых решений. Современные подходы к веб-разработке основываются на использовании передовых фреймворков и библиотек, мощных серверных платформ, а также систем автоматизации и управления версиями, что позволяет создавать сложные веб-приложения с минимальными затратами времени и ресурсов.

ГЛАВА 2. Практическая часть

В качестве объекта для разработки было выбрано веб-приложение для управления заметками с интеграцией нейросетевых алгоритмов анализа текста. За основу для разработки приложения была взята технология хранения и структурирования данных «Zettelkasten», созданная Никласом Луманом. Технология «Zettelkasten» (нем. "коробка с карточками") была разработана немецким социологом Никласом Луманом (Niklas Luhmann) и представляет собой систему организации знаний и заметок. Луман использовал эту систему для систематизации своих идей и исследований, что позволило ему стать одним из самых продуктивных ученых своего времени.

Основные принципы технологии «Zettelkasten» включают:

1. Карточки-заметки: каждая идея, мысль или информация записывается на отдельную карточку, что помогает сохранить и развивать каждую мысль автономно;
2. Идентификаторы: каждая карточка получает уникальный идентификатор, что позволяет легко ссылаться на нее и связывать с другими карточками;
3. Связи между карточками: карточки связаны друг с другом с помощью ссылок, создавая сеть идей. Луман называл это "внутренней связностью", которая позволяет легко находить связанные идеи и темы;
4. Гибкость и масштабируемость: система позволяет добавлять новые карточки и связи бесконечно, что делает ее очень гибкой и масштабируемой;
5. Неформальная структура: нет необходимости следовать жестким правилам или иерархиям, что дает простор для креативности и свободного потока идей.

Используя эту систему, Луман смог написать более 70 книг и около 400 статей. Технология «Zettelkasten» сегодня используется многими учеными,

писателями и исследователями для управления знаниями, ведения заметок и разработки идей. Она также вдохновила разработку различных цифровых инструментов и приложений, которые автоматизируют процесс создания и управления заметками в стиле «Zettelkasten».

В дипломном проекте было решено объединить принципы работы технологии «Zettelkasten» и неросетевой анализ текста. Пример работы веб-приложения «нейродневник»: пользователь читает некую статью в интернете, основную информацию из которой он бы не хотел потерять. Для этого он создает в приложении «нейродневник» юнит, в которой вставляет ссылку на прочитанную ранее статью. Нейросеть анализирует текст статьи из прикрепленной ссылке, вносит его в заметку и делает его суммаризацию, а также присваивает основные теги, при помощи которых данная статья может быть связана с другими статьями в личной базе знаний пользователя. На выходе пользователь получает: текст статьи, его суммаризацию, теги и ссылку на источник. Так же в приложении задумана функция создания проектов. Пользователь имеет возможность записывать собственные проекты, или идеи в приложении. При добавлении текста в раздел «проект», «нейродневник» с помощью тегов будет предлагать заметки, которые пользователь когда-то создал и которые были бы полезны при разработке нового проекта.

Веб-приложение содержит следующие функции:

1. Создание и редактирование заметок;
2. Создание юнитов с помощью нейросетевой обработки текста;
3. Присваивание тегов;
4. Возможность создать личную базу знаний;
5. Возможность получать идеи из базы знаний при разработке проекта.

1.1. Постановка требований к веб-приложению

Создание веб-приложения требует тщательного анализа и определения требований, которые обеспечат его функциональность, надежность и удовлетворение нужд конечных пользователей. Понимание и формализация этих требований являются критически важными этапами, которые направляют последующие этапы разработки, тестирования и внедрения системы. В данном разделе будет рассмотрен процесс постановки требований к веб-приложению, описаны методы сбора информации, анализ и классификация требований, а также представлены основные функциональные и нефункциональные требования.

Методы сбора требований. Сбор и анализ требований к веб-приложению предусматривают применение различных методических подходов и инструментов. Основные методы:

1. Интервью и анкетирование: проведение прямых бесед с потенциальными пользователями и заинтересованными сторонами, а также распространение формализованных опросных листов;
2. Анализ существующих систем: изучение текущих программных решений с целью выявления их сильных и слабых сторон, что позволяет определить желаемые улучшения и новые функции;
3. Мозговой штурм и воркшопы: организация групповых обсуждений среди команды разработчиков, проектировщиков и пользователей для генерации новых идей и выработки общих решений;
4. Создание сценариев использования (Use Case): определение типичных ситуаций, в которых будет использоваться веб-приложение, что помогает лучше понять взаимодействие пользователей с системой.

В данном дипломном проекте использовались такие методы сбора требований, как анализ существующих систем, мозговой штурм и создание сценариев использования.

Классификация поставленных требований. Для удобства анализа и управления требованиями, их классификация представляется необходимым этапом. Требования к веб-приложению можно классифицировать следующим образом:

1. Функциональные требования: определяют конкретные функции и особенности системы, которые необходимы для выполнения задач пользователя. Эти требования описывают, что именно должна делать система;
2. Нефункциональные требования: указывают на атрибуты качества системы, такие как производительность, безопасность, надежность, удобство использования и совместимость;
3. Системные требования: включают аппаратные и программные ресурсы, необходимые для развертывания и функционирования системы.

Таблица 3

Требования поставленные в процессе разработки веб-приложения

Тип требований	Описание
Функциональные требования	1. Регистрация и аутентификация пользователей. Дает возможность будущим пользователям создавать свою учетную запись, входить в аккаунт, а также восстанавливать к нему доступ при необходимости; 2. Управление контентом. Возможность добавлять, редактировать и удалять тест заметок;

	3. Интерфейс поиска и фильтрации. Дает возможность быстро искать и сортировать информацию по заданным критериям; 4. Интеграция с внешними сервисами. Обеспечивает возможность взаимодействия со сторонними системами и сервисами, такими как нейросети.
Нефункциональные требования	1. Удобство использования. Веб-приложение должно быть интуитивно понятно; 2. Производительность и масштабируемость: Система должна обеспечивать приемлемое время отклика и быть способной обрабатывать увеличивающийся объем данных и пользователей; 3. Безопасность: Защита данных пользователей, предотвращение несанкционированного доступа и обеспечение конфиденциальности информации.

После определения основных требований к разрабатываемому веб-приложению была начата разработка архитектуры системы.

Вывод: Процесс постановки требований к веб-приложению является основополагающим этапом, определяющим успех проекта в целом. Важность тщательного сбора, анализа и документирования требований не может быть переоценена, так как эти требования формируют базу для всех последующих этапов разработки. Понимание и учет потребностей пользователей, а также соблюдение нефункциональных требований обеспечивают создание качественного, надежного и удовлетворяющего ожидания веб-приложения.

1.2. Архитектура системы

Архитектура системы представляет собой структурное и логическое представление компонентов системы и их взаимодействий. Она служит основой для дальнейшей реализации, обеспечивая понимание структуры и функционирования системы всеми заинтересованными сторонами, включая разработчиков, тестировщиков, архитекторов и пользователей. В этом разделе будет описана архитектура создаваемого веб-приложения, включающая рассмотрение общей логической структуры, технологического стека, компонентов и их взаимодействий, а также схемы развертывания.

Общая логическая структура

Основной целью архитектуры системы является обеспечение надежного, масштабируемого и поддерживаемого решения. Логическая структура веб-приложения строится на принципах модульности и четкого разделения ответственности. Основные компоненты логической архитектуры разрабатываемого веб-приложения «Нейродневник» включают:

1. Презентационный слой (Frontend): отвечает за взаимодействие с пользователем. Включает интерфейсные компоненты, реализующие отображение и обработку данных, вводимых пользователем. Используемые технологии: HTML, CSS, JavaScript;
2. Слой бизнес-логики (Backend): реализует прикладные задачи и логику обработки данных. Отвечает за выполнение бизнес-процессов и взаимодействие с базой данных. Используемые технологии: Pyton, Django, Ollama;
3. Слой данных (Database): обеспечивает хранение, управление и доступ к данным. Используемая база данных должна поддерживать ACID-транзакции и быть способной масштабироваться в зависимости от нагрузки. Используемые технологии: SQLLite.

Технологический стек

Взаимодействие компонентов. Компоненты системы взаимодействуют между собой через четко определенные интерфейсы и протоколы. Основные типы взаимодействий включают:

1. Клиент-серверное взаимодействие. Клиентские запросы отправляются на сервер, сервер обрабатывает их и возвращает соответствующие ответы;
2. Взаимодействие между сервером и базой данных;
3. Межмодульное взаимодействие в серверной части: Компоненты backend взаимодействуют посредством вызова функций и модулей, обеспечивая выполнение бизнес-логики и обработку данных.

Все взаимодействия компонентов в разрабатываемом веб-приложении производились с помощью фреймворка Django, который позволяет воспроизводить в себе основные типы взаимодействий.

Безопасность и мониторинг

Обеспечение безопасности и мониторинга веб-приложения является неотъемлемой частью его архитектуры. Основные меры безопасности включают:

1. Аутентификация и авторизация;
2. Шифрование данных: Применение HTTPS для защиты данных, передаваемых между клиентом и сервером;
3. Мониторинг и логирование используется для отслеживания состояния системы, анализа производительности и быстрого реагирования на инциденты.

Вывод: Разработанная архитектура системы предоставляет чёткую и структурированную основу для дальнейшего проектирования и реализации

веб-приложения. Применение современных технологий и подходов к разработке обеспечивает высокую производительность, масштабируемость и безопасность системы, что соответствует требованиям современных пользователей и бизнес-задач.

1.3. Разработка пользовательского интерфейса (frontend)

Пользовательский интерфейс (UI) является важнейшим компонентом любого программного продукта, поскольку он обеспечивает взаимодействие пользователя с системой. Удобный, интуитивно понятный и эстетически привлекательный интерфейс играет ключевую роль в восприятии качества программного обеспечения. В этом разделе будут рассмотрены методы и инструменты разработки интерфейса, основные этапы проектирования, требования к интерфейсу, а также применяемые технологии и принципы.

Процесс проектирования пользовательского интерфейса включает в себя несколько ключевых принципов:

1. Интуитивность: Пользователи должны легко понимать и использовать интерфейс без необходимости длительного обучения;
2. Последовательность: Элементы интерфейса должны быть единообразными, чтобы уменьшить когнитивную нагрузку на пользователя;
3. Обратная связь: Система должна своевременно предоставлять пользователю информацию о выполненных действиях и текущем состоянии;
4. Предотвращение ошибок: Интерфейс должен минимизировать возможность совершения ошибок и предоставлять механизмы их исправления.

Прототипирование

Прототипирование позволяет визуализировать и оценить концепции интерфейса на ранних этапах разработки. Основные виды прототипов:

1. Бумажные прототипы: простые наброски, которые используются для первоначального обсуждения идей.

2. Интерактивные прототипы: создаются с использованием специализированного программного обеспечения (например, Figma, Sketch) и позволяют моделировать взаимодействие пользователя с системой.

В данном проекте использовался бумажный прототип (см.: приложения 1, рис.1).

Реализация интерфейса

1. Технологический стек

Для реализации пользовательского интерфейса выбраны следующие технологии и инструменты:

HTML: Стандартный язык гипертекстовой разметки для создания структуры веб-страниц.

CSS: Каскадные таблицы стилей для оформления и визуального представления элементов интерфейса.

JavaScript: Язык программирования для обеспечения динамического поведения и интерактивности.

Данные языки программирования были выбраны для проекта, поскольку имеют преимущества в виде простоты разработки и скорости верстки страниц.

2. Инструменты и среды разработки, используемые в проекте:

Visual Studio Code: Интегрированная среда разработки (IDE) с поддержкой множества плагинов и расширений.

Структура и компоненты интерфейса

1. Главная страница

Главная страница-первое, что видит пользователь в веб-приложении. Она является некой «отправной точкой», с которой начинается путь пользователя по приложению, поэтому главная страница должна быть минималистичной, интуитивно понятной и содержать основные элементы системы навигации и информирования, чтобы пользователи могли сразу понять, какие возможности предоставляет система. В данном дипломном проекте главная страница содержит карточки заметок пользователя, меню, вход/выход, поиск, фильтрация по тегам, аватарку пользователя, атак же элементы фирменного стиля. См рис

2. Формы

Формы являются важными элементами ввода/вывода данных. Они должны быть простыми и понятными, с четкими инструкциями и механизмами валидации ввода, чтобы минимизировать ошибки пользователей. В разрабатываемом приложении они представлены в виде полей ввода текста и кнопок. см рис..

3. Навигация

Навигационная структура должна быть логичной и последовательной. Основные элементы навигации в разрабатываемом веб-приложении включают:

Меню: Глобальная навигация, доступная на всех страницах.

Хлебные крошки: Индикация текущей позиции пользователя в структуре сайта.

Поиск: Возможность быстро находить информацию по ключевым словам.

4. Тестирование и оценка

Тестирование интерфейса направлено на выявление проблем и получение обратной связи от пользователей. Основной метод тестирования, используемый в проекте — это юзабилити-тестирование, где пользователи выполняют задачи с использованием прототипа, а разработчики наблюдают за их действиями.

Вывод: Разработка пользовательского интерфейса является многогранным и многоплановым процессом, включающим исследование, дизайн и тестирование. Успешная реализация интерфейса требует не только технических навыков, но и понимания потребностей пользователей и принципов эргономики. Внедрение современных технологий и инструментов позволяет создавать интерфейсы, которые обеспечивают высокую производительность, удобство использования и эстетическое восприятие.

1.4. Разработка бизнес-логики веб-приложения (backend)

Веб-приложения представляют собой сложные системы, в которых бизнес-логика играет ключевую роль в обеспечении функциональности и взаимодействия с пользователями. Бизнес-логика определяет, как данные обрабатываются, как пользователи взаимодействуют с приложением и как результат взаимодействия отображается. В данной части дипломного проекта рассматривается процесс разработки бизнес-логики для веб-приложения, включая выбор архитектурного подхода, проектирование моделей данных, реализацию бизнес-процессов и интеграцию с frontend-частью приложения.

Технологический стек

Для разработки backend-части веб-приложения необходим выбор подходящего технологического стека, включающего язык программирования, фреймворк и базы данных. Для разработки веб-приложения «нейродневник» использовался стек Python с Django, так как он упрощает разработку за счет встроенных инструментов и библиотек. Возможностей этого стека хватает для проекта, учитывая имеющиеся ресурсы и технические требования.

Проектирование моделей данных

1. Определение сущностей

На этом этапе необходимо определить основные сущности, которые будут использоваться в приложении. Например, если приложение предназначено для электронной коммерции, то сущностями могут быть: пользователи, товары, заказы, отзывы. Каждая сущность должна быть описана с учетом уникальных атрибутов и взаимосвязей между ними. В разрабатываемом приложении основными сущностями являются:

1.1. Пользователь (User)

ID: Уникальный идентификатор пользователя.

Имя: Имя пользователя.

Email: Электронная почта, используемая для входа в систему.

Пароль: Хешированный пароль для аутентификации.

Дата регистрации: Время регистрации пользователя.

Роль: Роль пользователя (например, администратор или обычный пользователь).

1.2. Заметка (Note)

ID: Уникальный идентификатор заметки.

Заголовок: Название заметки.

Контент: Основное содержимое заметки.

Дата создания: Время создания заметки.

Дата последнего обновления: Время последнего редактирования заметки.

ID автора: Идентификатор пользователя, создавшего заметку.

Статус: (Опционально) Статус заметки (например, черновик, опубликована).

1.3. Метка (Tag)

ID: Уникальный идентификатор метки.

Название: Название метки.

Цвет: (Опционально) Цвет для визуального выделения метки.

1.4. Привязка заметки к метке (NoteTag)

ID: Уникальный идентификатор записи, связывающей заметку и метку.

ID заметки: Идентификатор заметки.

ID метки: Идентификатор метки.

1.5. Юнит (Unit)

ID: Уникальный идентификатор заметки.

Заголовок: Название заметки.

URL: Ссылка на источник

Контент: Основное содержимое заметки.

Суммаризация: Основная мысль

Дата создания: Время создания заметки.

Дата последнего обновления: Время последнего редактирования заметки.

ID автора: Идентификатор пользователя, создавшего заметку.

Статус: (Опционально) Статус заметки (например, черновик, опубликована)

1.6. Привязка заметки к категории (UnitTag)

ID: Уникальный идентификатор записи, связывающей заметку и категорию.

ID заметки: Идентификатор заметки.

ID категории: Идентификатор категории.

Создание схемы базы данных

После определения сущностей следует разработка схемы базы данных, которая включает в себя таблицы, поля и связи между ними. Использование реляционных баз данных (в данном проекте SQLLite) подразумевает нормализацию данных для устранения избыточности и обеспечения целостности.

Реализация бизнес-процессов

1. Написание бизнес-логики

Бизнес-логика должна быть сосредоточена на обработке данных и выполнении операций, таких как создание, чтение, обновление и удаление (CRUD). Реализация бизнес-процессов включает:

1.1. Валидацию входных данных
1.2. Обработку ошибок
1.3. Взаимодействие с базой данных через ORM (Object-Relational Mapping)

2. Интеграция с frontend

2.1. Совместимость и взаимодействие

Интеграция вывода данных с помощью API требует четкой обмениваемой схемы данных между клиентом и сервером. Фронтэнд разработка должна учитывать ограничения и возможности бэкенда для оптимизации пользовательского опыта.

2.2. Тестирование

Тестирование бизнес-логики включает юнит-тестирование для проверки отдельных компонентов и интеграционное тестирование для проверки взаимодействия между модулями. Это помогает выявлять и исправлять ошибки на ранних стадиях разработки.

Вывод: Разработка бизнес-логики веб-приложения является сложным, но увлекательным процессом, требующим системного подхода и глубокого понимания как технологических, так и бизнес-процессов. Грамотная реализация описанных этапов способствует созданию успешного и функционального продукта, способного удовлетворить потребности пользователей и предложить конкурентные преимущества на рынке.

1.5. Интеграция нейросетевых алгоритмов

Интеграция нейросетевых алгоритмов в современные информационные системы представляет собой одну из наиболее актуальных задач в области искусственного интеллекта и машинного обучения. Нейросети применяются для решения широкого спектра задач: от простой классификации изображений до сложных предсказательных моделей и обработки естественного языка. В этом разделе будет рассмотрен процесс интеграции нейросетевых алгоритмов в разработанную систему, включая теоретические аспекты, используемые технологии и конкретные примеры их применения.

Описание процесса интеграции

Процесс интеграции нейросетевых алгоритмов включает в себя несколько ключевых этапов:

1. Определение задачи: Формулирование конкретной задачи, которую должна решить нейросеть, будь то классификация, регрессия или кластеризация.
2. Сбор и подготовка данных: Сбор необходимых данных, их очистка и предварительная обработка для повышения качества обучения модели.
3. Построение и обучение модели: Создание архитектуры нейронной сети, настройка гиперпараметров и обучение модели на подготовленных данных.
4. Валидация и тестирование: Оценка качества работы модели посредством методов валидации и кросс-валидации, тестирование на новых данных.

5. Развертывание модели: Интеграция обученной модели в программную систему и настройка её для работы в реальных условиях.

В данном проекте была использована готовая нейросеть Ollama 3.1 – 7b.

Вывод: Интеграция нейросетевых алгоритмов открывает новые возможности для создания интеллектуальных систем, способных решать широкий спектр задач с высокой точностью и эффективностью. Применение современных инструментов и технологий позволяет существенно упростить процесс разработки и развертывания нейронных сетей, обеспечивая их адаптацию к разнообразным прикладным сценариям.

1.6. Тестирование и отладка

Тестирование и отладка являются неотъемлемыми этапами разработки программного обеспечения и воплощают в себе весь комплекс мер, направленных на обеспечение корректного функционирования системы, соответствия её спецификациям и требованиям, а также нахождения и устранения ошибок. В данном разделе описаны процессы тестирования и отладки разработанного программного обеспечения с использованием методик и инструментов, принятых в отрасли.

Цели и задачи тестирования.

Основными целями тестирования являются:

1. Подтверждение соответствия программного обеспечения заявленным функциональным и нефункциональным требованиям.
2. Обнаружение дефектов и сбоев в программной системе.
3. Оценка качества программного обеспечения и проверка его готовности к выпуску.

Задачи включают:

1. Создание тестовых сценариев и тестовых данных.

2. Проведение тестирования на различных уровнях (юнит-тестирование, интеграционное тестирование, системное тестирование и приемочное тестирование).
3. Документирование результатов тестирования и анализ полученных данных.

План тестирования

Практическая реализация тестирования предполагает четкую структуру и последовательность действий, определяемых планом тестирования. Основные этапы плана:

Подготовительный этап:

1. Определение целей и задач тестирования.
2. Сбор необходимых данных, таких как спецификации требований, техническая документация и архитектурные схемы.

Проектирование тестов:

1. Разработка стратегии тестирования.
2. Создание подробных тест-кейсов, описывающих шаги выполнения тестов и ожидаемые результаты.
3. Определение критериев входа и выхода для каждой стадии тестирования.

Выполнение тестов:

1. Проведение тестирования в соответствии с разработанными тест-кейсами.
2. Регистрация всех выявленных дефектов и ошибок в систему управления дефектами.

Анализ результатов:

1. Сравнение фактических результатов выполнения тестов с ожидаемыми.
2. Выведение заключений относительно качества программного обеспечения и определения областей для доработки.

Отчетность и документация:

1. Подготовка отчетов о проведенном тестировании, включающих в себя статус каждого тестового сценария, описание выявленных дефектов и предложенные меры по их устранению.

Методы тестирования

Для проведения тестирования разработанного программного обеспечения использованы следующие методы:

1. Статическое тестирование: анализ кода, проектной документации и спецификаций без выполнения программы. Включает в себя обзор кода и архитектурных решений.

2. Динамическое тестирование: активное выполнение программного кода с целью выявления ошибок в функциях и логике. Подразумевает черный ящик (тестирование функциональных требований без знания внутренней структуры), белый ящик (тестирование внутренней структуры и логики программного кода), совместимость, нагрузочное и стресс-тестирование.

Отладка

Процесс отладки включает в себя выявление и устранение ошибок в программном обеспечении. Основными этапами отладки являются:

1. Идентификация ошибок: анализ логов, трассировка выполнения кода, использование отладочных средств для выявления причин возникших ошибок.

2. Диагностика причин: определение условий, вызывающих ошибки, их локализация в исходном коде.
3. Устранение ошибок: внесение корректировок в программный код с целью исправления выявленных ошибок, последующее повторное тестирование для подтверждения исправления.
4. Проверка изменений: выполнение регрессионного тестирования с целью убедиться, что внесенные исправления не привели к появлению новых ошибок.

Вывод: Проведенное тестирование и отладка разработанного программного обеспечения показали высокое качество кода и соответствие заявленным требованиям. Все выявленные дефекты были своевременно устранены, что позволило обеспечить надежную и стабильную работу системы. Полученные результаты свидетельствуют о готовности программного продукта к эксплуатации и его внедрению в рабочую среду. Этот раздел полного описывает процессы тестирования и отладки, которые были выполнены в ходе разработки программного обеспечения, подчеркивая важность данных этапов для достижения общего успеха проекта.

Заключение

Дипломный проект на тему "Разработка веб-приложения для управления заметками с интеграцией нейросетевых алгоритмов анализа текста" направлен на создание инновационного инструмента, который объединяет функции организации личных и рабочих заметок с мощными возможностями анализа текста, предоставляемыми современными нейросетевыми технологиями.

Основные итоги проекта:

1. Создание функционального веб-приложения:

Были разработаны основные компоненты веб-приложения, которое позволяет пользователям удобно создавать, редактировать и управлять своими заметками. Веб-приложение обладает интуитивно понятным интерфейсом, обеспечивающим легкость и эффективность взаимодействия.

2. Интеграция нейросетевых алгоритмов:

В проекте успешно интегрированы нейросетевые алгоритмы для анализа текста. Эти алгоритмы позволяют проводить лексический и семантический анализ заметок, выделять ключевые слова, автоматически классифицировать и группировать заметки на основе их содержания, а также предлагать пользователю релевантные теги.

Достижения и вклад в область:

Проект представляет собой значительный вклад в область управления информацией и использования искусственного интеллекта для улучшения пользовательского опыта. Интеграция нейросетевых алгоритмов анализа текста позволяет значительно упростить процесс упорядочивания и поиска информации, увеличивая эффективность работы пользователей. Данное веб-

приложение может быть полезным как для индивидуальных пользователей, так и для корпоративного сектора.

Перспективы дальнейшего развития:

1. Расширение функционала: Интеграция дополнительных функций, таких как голосовое управление, поддержка мультимедийных заметок (видео, аудио), и системы напоминаний.
2. Улучшение алгоритмов анализа текста: Использование более мощных и точных моделей машинного обучения, включая модели, обученные на специализированных корпусах текстов.
3. Интернационализация и локализация: Обеспечение поддержки различных языков и культурных особенностей для выхода на международный рынок.
4. Взаимодействие с другими сервисами: Интеграция с популярными офисными и коммуникационными платформами для создания единого рабочего пространства.

Список используемой литературы

1. Бехрузи, Х. "Веб-программирование с использованием HTML, CSS и JavaScript". — Москва: БХВ-Петербург, 2019.
2. Фланаган, Д. "JavaScript: Подробное руководство". — 7-е изд. — Москва: Вильямс, 2018.
3. Фримен, Э. "HTML и CSS. Разработка и дизайн веб-сайтов". — Москва: Эксмо, 2020.
4. Глинский, А. "Современный веб-дизайн: User Experience и User Interface". — Эксмо, 2019.
5. Рашид, Ф., Мамуров, Б. "Машинное обучение для анализа текстов: Практическое руководство". — Альпина Паблишер, 2020.
6. Миллер, Дж. "Python для начинающих: от простого к сложному". — Питер, 2020.
7. Фаррелл, Б. "Python глазами хакера". — Питер, 2021.
8. Розумовский, Д. "Django 3: Практическое руководство по созданию эффективных веб-приложений". — Москва: ДМК Пресс, 2021.
9. Вац, Р. "Анализ социальных сетей и текстовых данных с использованием машинного обучения на примере Python". — Wiley, 2018.
10. Шеменков, П. «Разработка и исследование модели нейросетевого метода анализа текстовых документов», 2009.
11. Шишаев,М. «Нейросетевые модели в задачах семантического анализа текстов на естественном языке», 2020.
12. Kennedy, A. "Building Websites with Django". — Apress, 2018.
13. Devlin, J., Chang, M.-W., Lee, K. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", 2018.
14. Mills, R. "FastText: Efficient text classification and representation learning", 2016.
15. Интернет-статья «ТОП-15 лучших приложений для заметок»: https://timeweb.com/ru/community/articles/top-15-luchshih-prilozheniy-

dlya-zametok

16. Интернет-статья «Обзор 10 программ для ведения заметок: выбираем лучший вариант»: https://3dnews.ru/1014032/obzor-10-programm-dlya-zametok

Приложения

Приложение 1. Разработка пользовательского интерфейса (frontend)

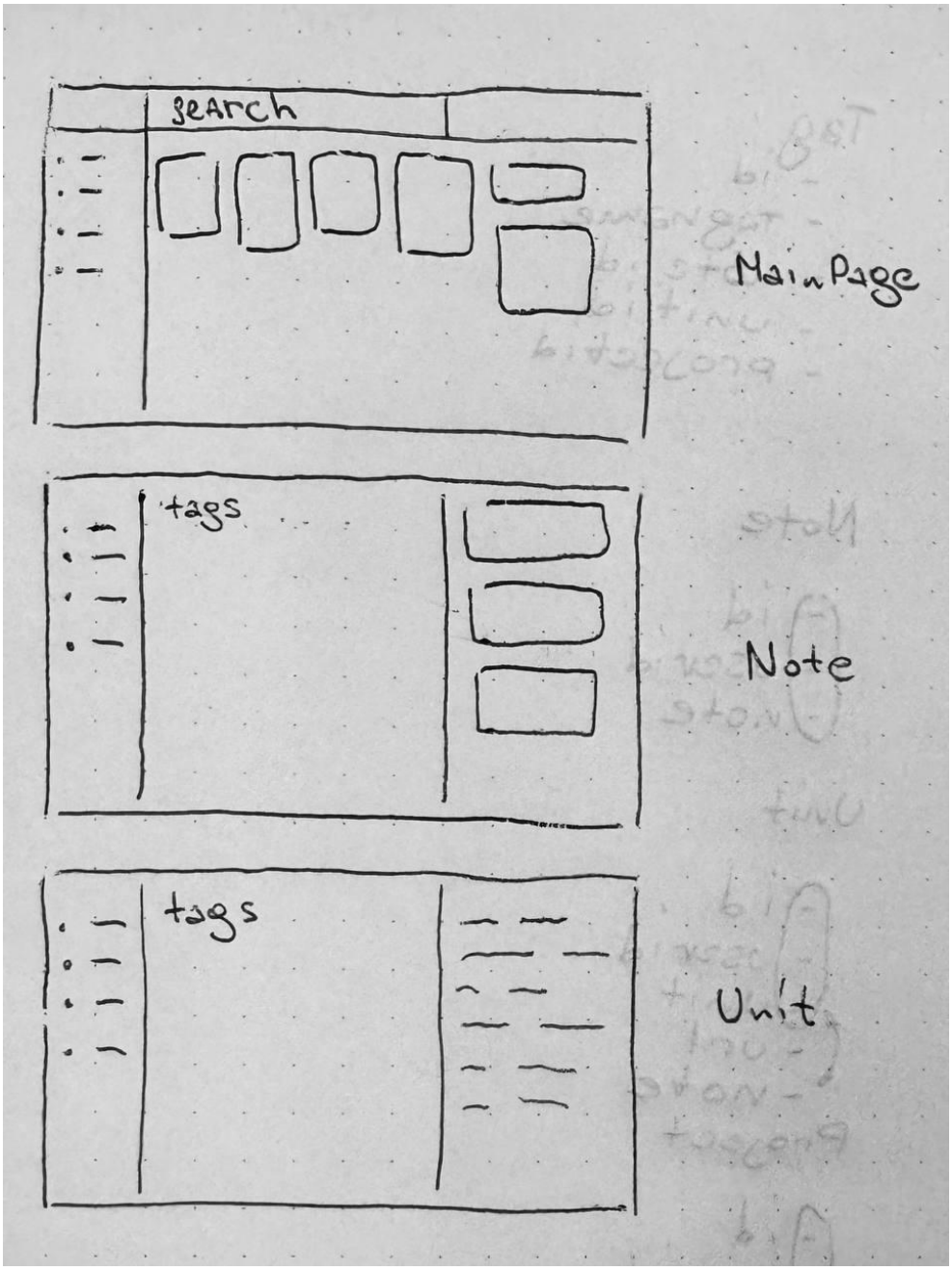

Рис. 1-Бумажный прототип разрабатываемого веб-приложения

Продолжение приложения 1. Разработка пользовательского интерфейса (frontend)

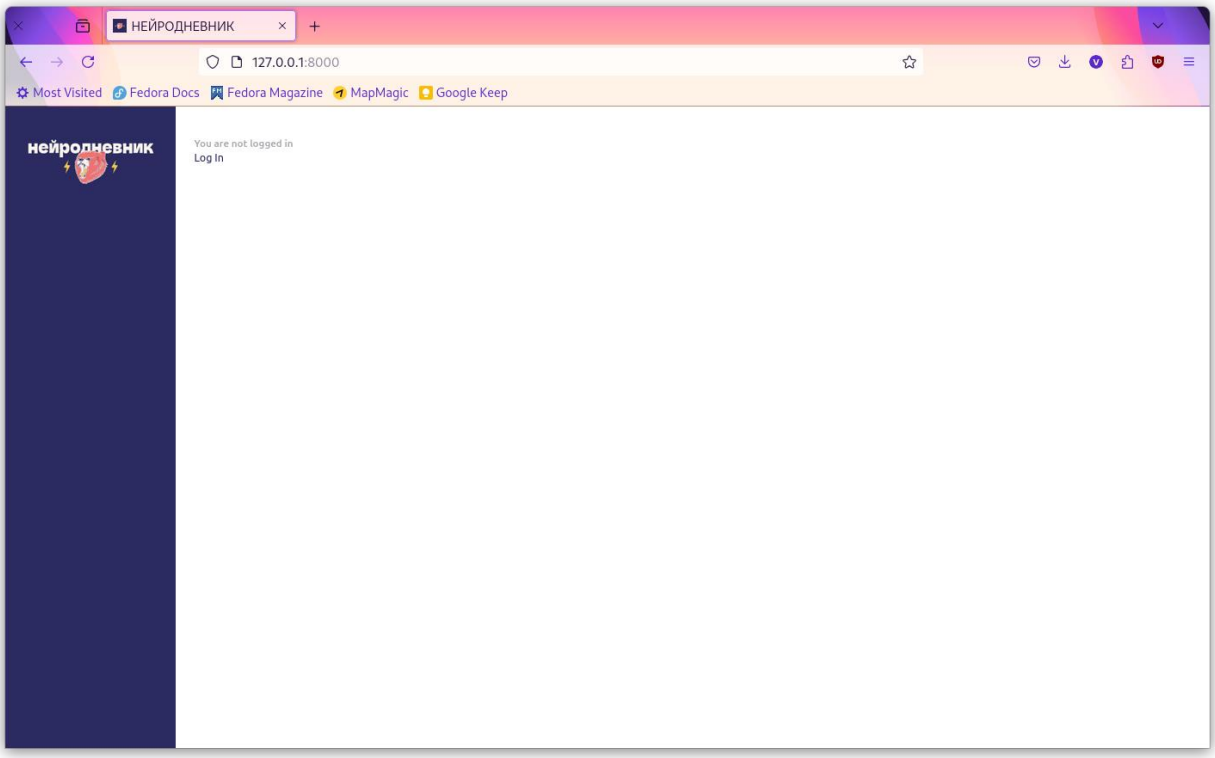

Рис. 2-Страница предложения входа

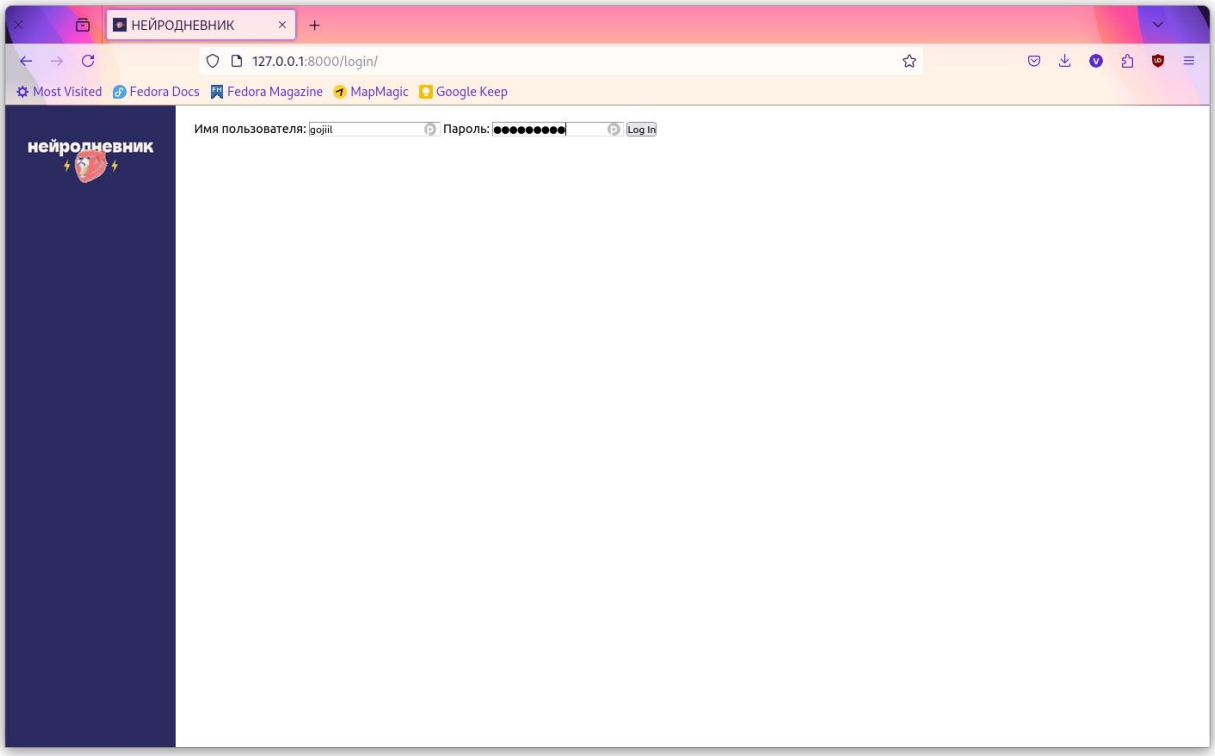

Рис. 3-Страница ввода логина и пароля

Продолжение приложения 1. Разработка пользовательского интерфейса (frontend)

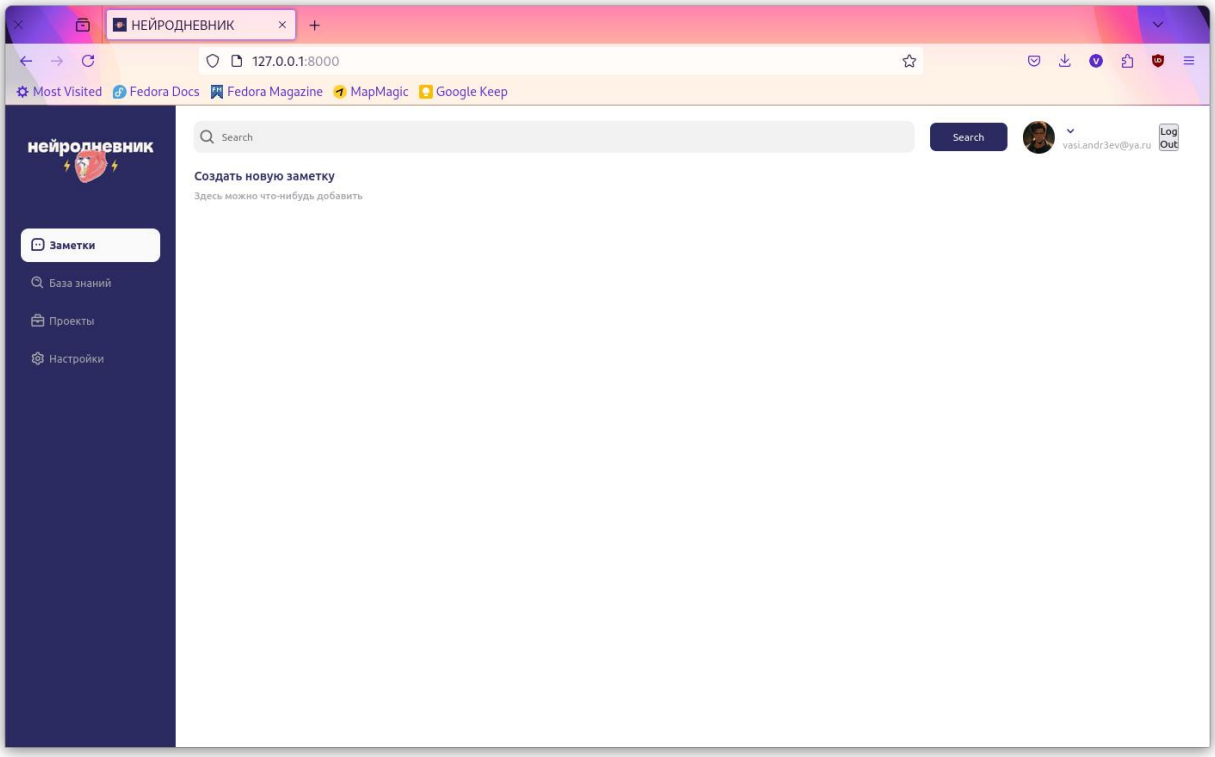

Рис. 4-Главная страница

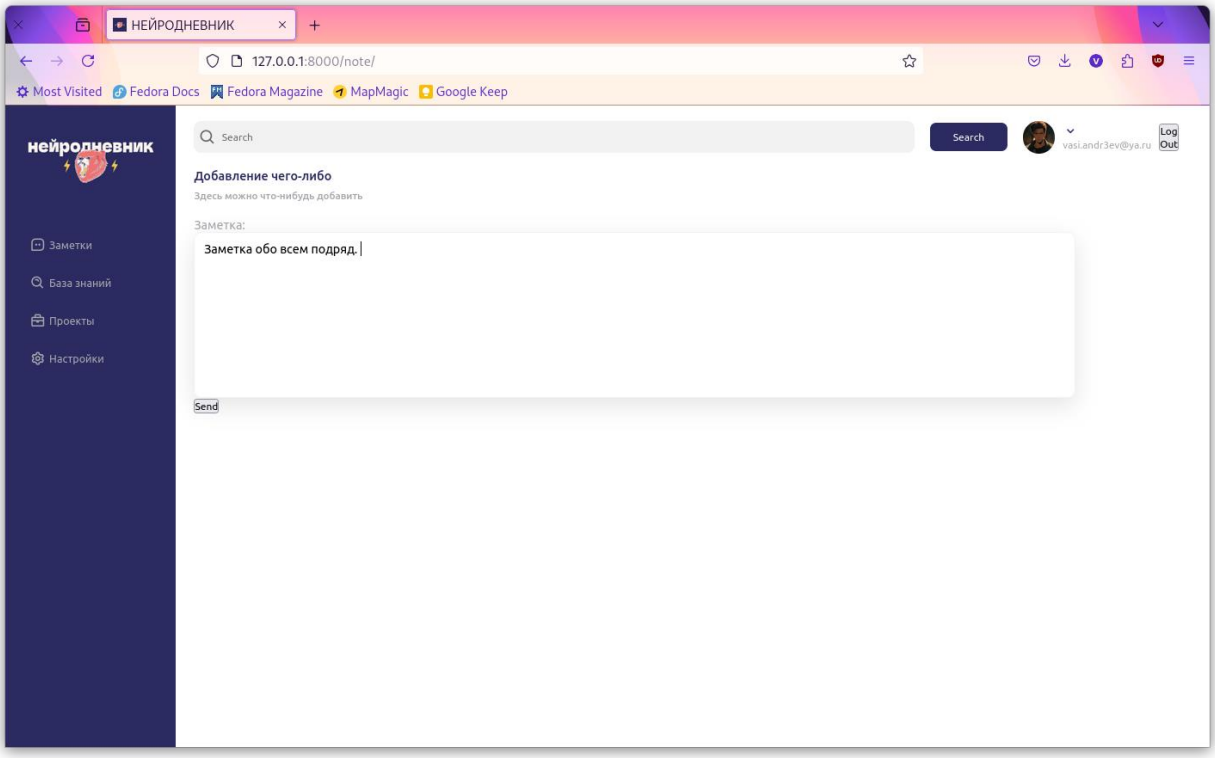

Рис. 5-Создание заметки

Продолжение приложения 1. Разработка пользовательского интерфейса (frontend)

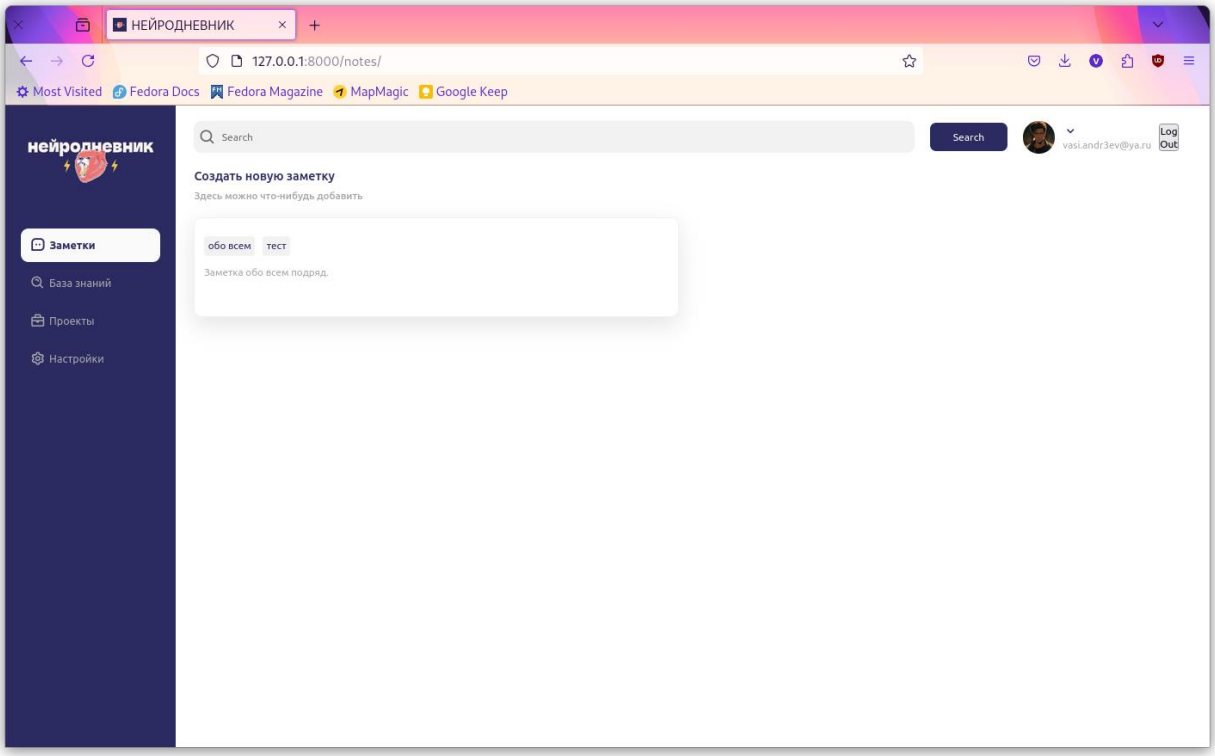

Рис. 6-Главная страница с созданной заметкой

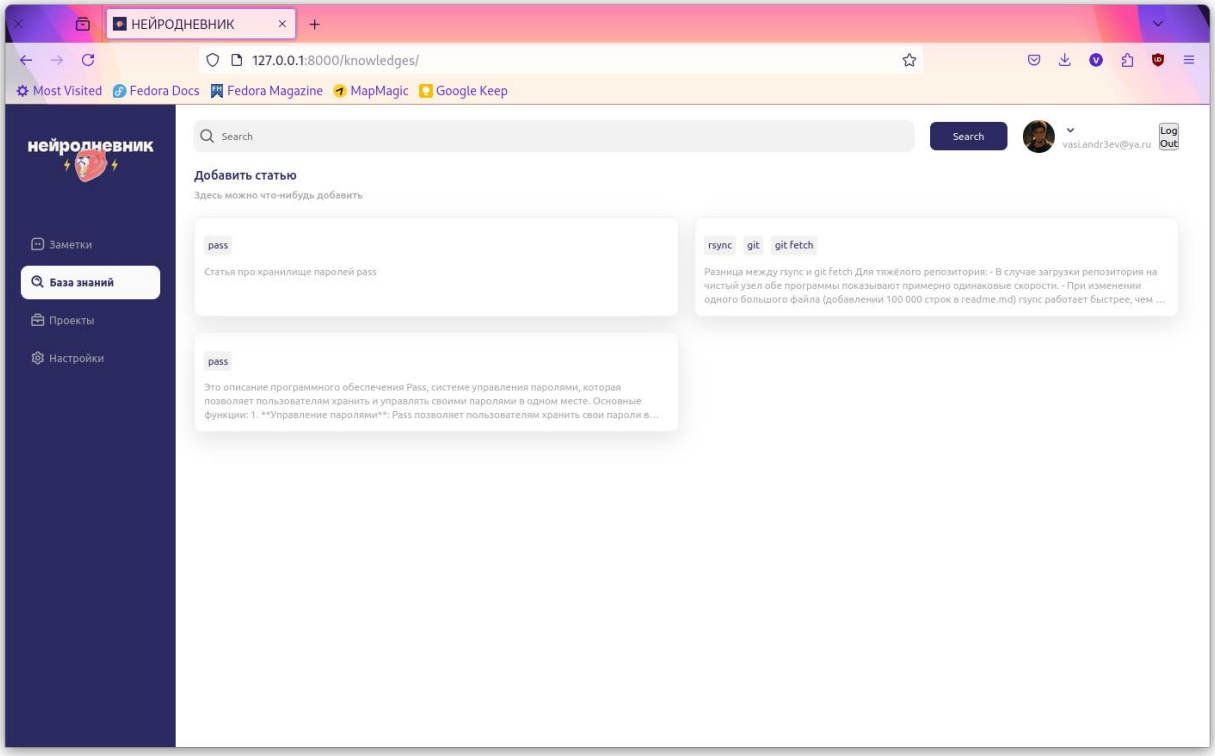

Рис.7-Главная страница базы знаний

Продолжение приложения 1. Разработка пользовательского интерфейса (frontend)

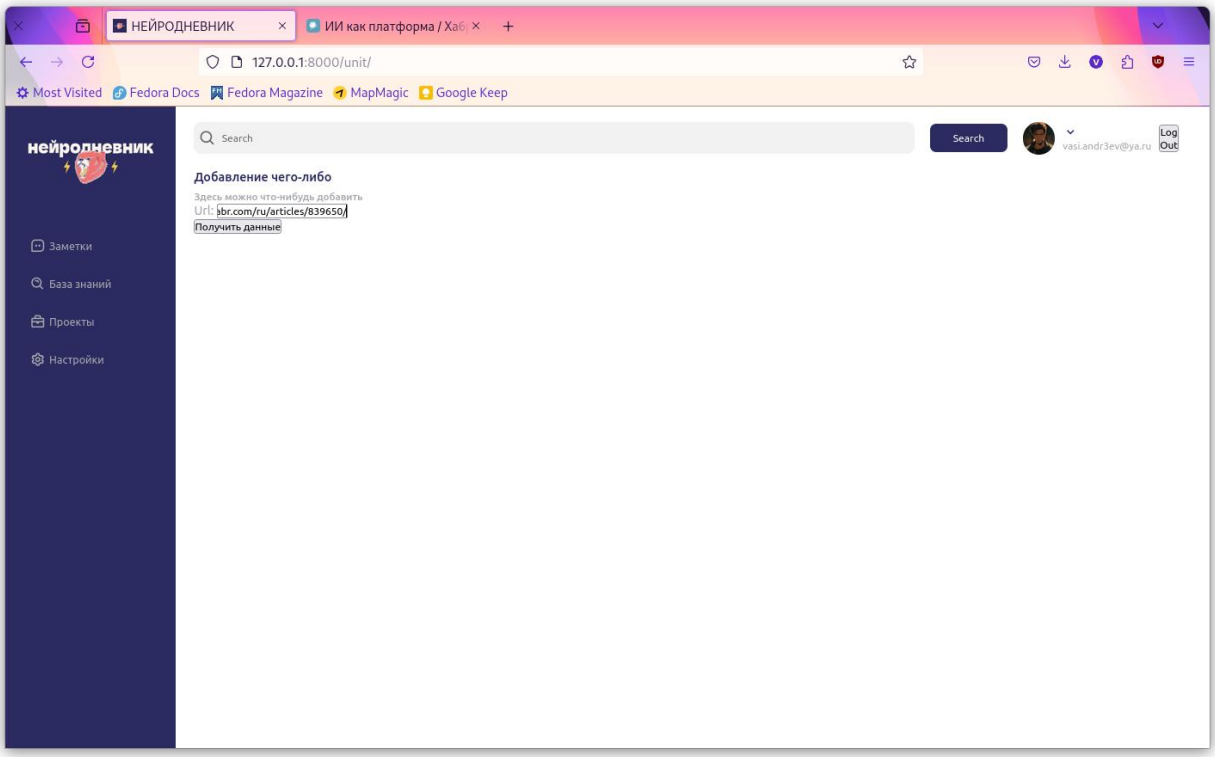

Рис. 8-Создание юнита

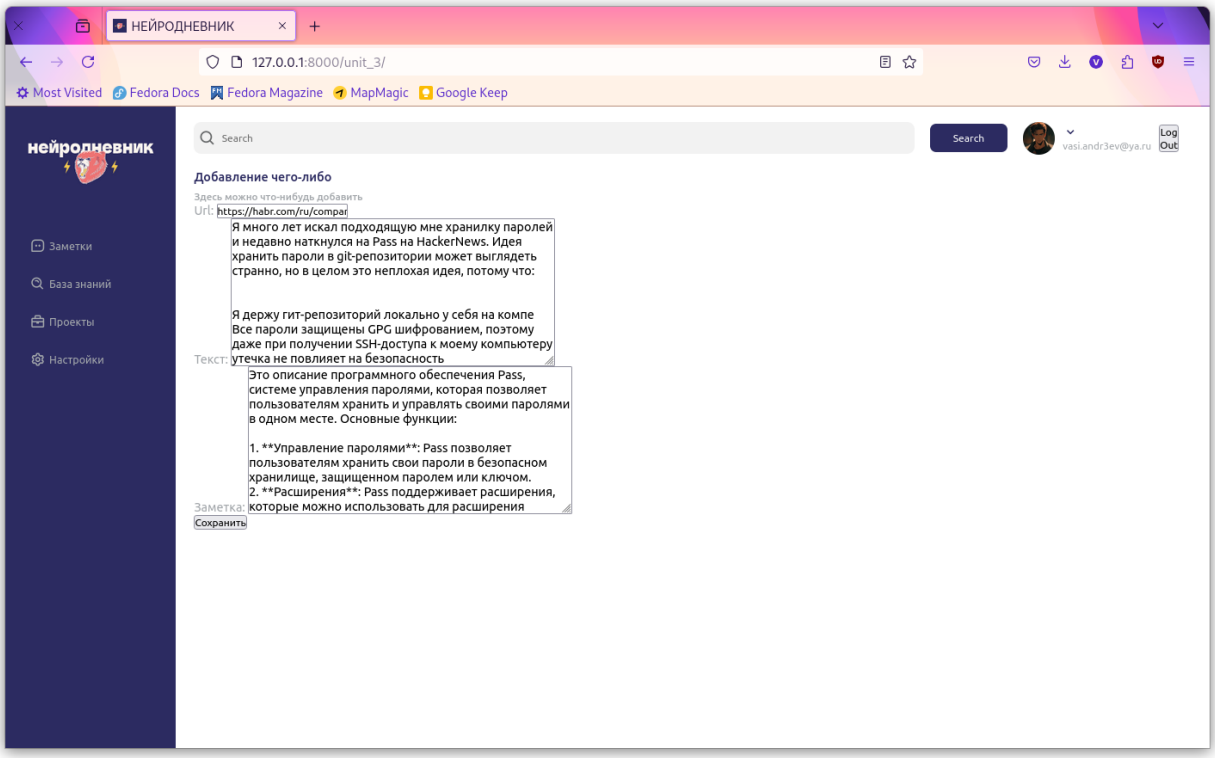

Рис. 8-Редактирование юнита