МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ПО БРОНИРОВАНИЮ СЕРВИСНЫХ УСЛУГ С ИСПОЛЬЗОВАНИЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА ДЛЯ ОЦЕНКИ ОТЗЫВОВ ПОЛЬЗОВАТЕЛЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Китаева Дмитрия Александровича

Научный руководитель
к. ф.-м. н., доцент _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

В эпоху цифровых технологий и стремительного развития интернета веб-приложения становятся неотъемлемой частью различных сфер деятельности, включая индустрию услуг. Пользователи привыкли к быстрому и удобному доступу к услугам через интернет, выбирая из широкого спектра предложений, сравнивая цены, читая отзывы и делясь впечатлениями о сервисе. Подобные приложения обрабатывают и хранят огромные потоки информации, такие как логины, пароли, имена, даты и заказы. Эффективное управление этими данными играет важную роль в обеспечении непрерывного и качественного обслуживания пользователей. Безопасность и надежность хранения данных также необходимы, так как пользователи доверяют приложениям свою личную информацию.

Учитывая вышеупомянутые аспекты, цель данной работы заключается в разработке веб-приложения, которое позволит пользователям выбирать сервисы, мастеров и бронировать их услуги через интернет, оставлять отзывы о проделанной работе. Данное приложение будет включать все необходимые функции для удобного взаимодействия с пользователями, обеспечивать безопасное хранение и обработку данных, а также интегрировать интеллектуальные системы для улучшения качества предоставляемых услуг. Одним из ключевых элементов будет внедрение искусственного интеллекта для формирования оценки работы мастеров на основе анализа тональности текстов оставленных отзывов.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Рассмотреть методы обработки запросов при помощи языка Java.
2. Рассмотреть технологии хранения данных.
3. Проанализировать существующие технологии реализации внешней оболочки web-приложения.
4. Исследовать способы анализа человеческого языка.
5. Исследовать методы анализа тональности.
6. Исследовать методы интеграции обученной модели искусственного интелекта в web-приложения.
7. Разработать веб-приложение, которое позволит пользователям выбирать сервисы, мастеров и бронировать их услуги через интернет, оставлять отзывы о проделанной работе, оценка которых будет производиться при помощи искусственного интеллекта.

Структура и объём работы. Бакалаврская работа состоит из введения, 3 разделов, заключения, списка использованных источников и 6 приложений. Общий объем работы — 68 страниц, из них 48 страниц — основное содержание, CD-носитель с кодом в качестве приложения, список использованных источников информации включает 21 наименование.

КРАТКОЕ СОДЕЖАНИЕ РАБОТЫ

Первый раздел «Технологии разработки web-приложений с использованием искусственного интеллекта» посвящен иссдедованию существующих технологий, которые можно использовать для решения поставленных задач.

В рамках данной работы были рассмотрены:

Методы отправки, получения и обработки запросов на языке Java: библиотека HttpURLConnection, библиотека HttpClient, Spring WebClient. Spring WebClient, Spring MVC, Spring Boot.

Основные SQL-базы данных: MS SQL Server, Oracle, MySQL, PostgreSQL.

Основные нереляционные модели баз данных: ключ-значение, документно-ориентированная, колоночно-ориентированная, графовая.

Технологии реализации внешней оболочки web-приложения:

1. SPA (Single Page Application) — загружается единственная страница в браузере, содержимое которой меняется динамически, без перезагрузки страницы.
2. AJAX (Asynchronous JavaScript and XML) — совокупность технологий, обеспечивающих асинхронное взаимодействие клиентской части веб-приложения с сервером без необходимости полной перезагрузки страницы.
3. MPA (Multi Page Application) — приложение содержит несколько страниц, каждая из которых полностью перезагружается при навигации между ними.

Способы анализа человеческого языка:

NLP (Natural Language Processing, обработка естественного языка) — это направление в машинном обучении, посвященное распознаванию, генерации и обработке устной и письменной человеческой речи.

Основные способы решения задач NLP при помощи машинного обучения: модели обучения с учителем, обучение без учителя, модели трансформеров и глубокое обучение.

Методы анализа тональности:

Анализ тональности текста — это подраздел обработки естественного языка (NLP) целью которого является классификация текста по тональности.

На данный момент существуют следующие методы анализа тональности текста: правила на основе словаря, при помощи машинного обучения, при помощи глубокого обучения.

Методы интеграции ML-модели в web-приложение на Java: нативная интеграция (In-Process), использование JNI/JNA для нативных библиотек, использование PMML (Predictive Model Markup Language), при помощи REST API.

Второй раздел «Используемые технологии» содержит теоретическокую информацию об используемых технологиях, выборанных программных библиотеках и фреймворках.

`Spring Boot` — это часть экосистемы Spring, которая предоставляет готовую для использования платформу для создания автономных, готовых к запуску приложений на языке Java. Spring Boot значительно упрощает конфигурацию и настройку Spring-приложений, предоставляя инструменты для быстрой и легкой разработки.

`Spring Boot Web` — это часть экосистемы Spring, которая предоставляет все необходимое для быстрого создания веб-приложений.

`Spring Boot Data Jpa` — модуль, который добавляет поддержку Java Persistence API с использованием Spring Data JPA. Он обеспечивает простое взаимодействие с реляционными базами данных. Функции: CRUD операции, производные запросы, поддержка транзакций, интеграция с JPA.

`Spring Boot Security` — модуль, который добавляет поддержку безопасности в Spring приложение, обеспечивая механизмы аутентификации и авторизации. Функции: аутентификация, авторизация, управление сессиями, безопасность HTTP, BCryptPasswordEncoder.

`Hibernate Core` — это основной модуль Hibernate, который предоставляет возможности объектно-реляционного отображения (ORM) для взаимодействия с реляционными базами данных. Функции: объектно-реляционное отображение (ORM), управление жизненным циклом объектов, кеширование, язык запросов HQL.

`PostgreSQL` представляет собой продвинутую объектно-реляционную систему управления базами данных (ОРСУБД) с открытым исходным кодом, обеспечивающую высокую надежность, масштабируемость и соответствие стандарту SQL. Особое внимание уделяется расширяемости, система позволяет добавлять новые типы данных, функции, операторы, агрегатные функции, методы индексирования.

Для реализации архитектурного подхода Single Page Application была вы-

брана библиотека React для JavaScript — библиотека для создания пользовательских интерфейсов, основной принцип работы которой заключается во взаимодействии инкапсулированных компонент, способных самостоятельно управлять своим состоянием, что позволяет повторно отрисовывать компоненты в случае, если внутри них что-то изменилось, по-отдельности.

Основные концепции:

- хуки — это функции, позволяющие использовать состояние и другие возможности React без написания классовых компонентов;
- props — механизм передачи данных от родительского компонента к дочернему в React. Props являются неизменяемыми (read-only);
- promise — объект, представляющий промежуточный результат асинхронной операции. Имеет три состояния: pending (ожидание), fulfilled (выполнено успешно), rejected (выполнено с ошибкой);
- async/awai — синтаксический сахар для работы с promise, позволяющий писать асинхронный код в синхронном стиле;
- state (состояние) — объект, хранящий данные компонента, при изменении которого происходит перерендеринг компонента;
- event handlers (обработчики событий) — функции, которые вызываются в ответ на определенные действия пользователя или системные события;
- higher-order components (компоненты высшего порядка) — функция, которая принимает компонент и возвращает новый компонент с расширенной функциональностью.
- react route — библиотека маршрутизации для React-приложений, обеспечивающая навигацию между компонентами без перезагрузки страницы.
- local storage — Web API для хранения данных в браузере в формате ключ-значение. Данные сохраняются между сессиями браузера.
- HTTP-клиент (axios) — библиотека для выполнения HTTP-запросов, основанная на Promise.
- material-ui — библиотека React-компонентов, реализующая спецификацию Material Design от Google.
- conditional rendering (условный рендеринг) — механизм отображения различного контента в зависимости от условий.
- component lifecycle (жизненный цикл компонента) — последовательность этапов, через которые проходит компонент от создания до удаления.

— context API — механизм React для передачи данных через дерево компонентов без необходимости передавать props на каждом уровне.

Для машинного обучения используется язык Python и библиотека, которая называется NumPy (от англ. numerical Python — «численные методы для Python»). Она включает очень эффективную реализацию операций с векторами (таких, как вычисление скалярного произведения) и методы для работы с данными: статистические функции, создание массивов умножение матриц, векторное произведение.

Иные использованные библиотеки и методы:

torch (PyTorch) — фреймворк для построения и обучения нейронных сетей с автоматическим дифференцированием и GPU-ускорением;

pandas (pd) — обработка структурированных данных (DataFrame) и их предварительная подготовка;

transformers — для работы с NLP-моделями;

train_test_split из sklearn — разделение данных на обучающую и тестовую выборки;

compute_class_weight — вычисление весов классов для несбалансированных данных;

f1_score, accuracy_score — метрики оценки качества классификации;

Dataset из datasets — работа с данными в формате, оптимизированном для NLP-задач;

matplotlib.pyplot (plt) — визуализация результатов (графики, кривые обучения);

nn из torch — базовые компоненты нейронных сетей (слои, функции активации, loss-функции);

rubert-tiny2 — предобученная модель семейства BERT (Bidirectional Encoder Representations from Transformers) для анализа тональности отзывов.

Блокноты Jupyter Notebook — отличный способ проведения экспериментов по глубокому обучению. Они широко применяются в сообществах машинного обучения и науки о данных. Блокнот (notebook) — это файл, сгенерированный приложением Jupyter Notebook, который можно редактировать в браузере. В блокнот можно вставлять код на Python и сопровождать результаты его выполнения примечаниями с богатым оформлением. Блокноты позволяют разбить объемный эксперимент на несколько коротких шагов, выполняемых независи-

мо, что добавляет интерактивности в разработку и избавляет от необходимости повторно запускать предыдущий код, если что-то пошло не так на следующем шаге в эксперименте.

FastAPI — это высокопроизводительный веб-фреймворк для создания RESTful API на языке Python, сочетающий в себе преимущества статической типизации и асинхронной модели выполнения. В отличие от традиционных фреймворков для Python, которые построены на WSGI (The Web Server Gateway Interface) — это спецификация синхронного стандарта Python для подключения кода приложения к веб-серверам), FastAPI построен на ASGI (Asynchronous Server Gateway Interface, Интерфейс шлюза асинхронного сервера), что позволяет ему работать быстрее традиционных фреймворков, поскольку синхронное взаимодействие может приводить к активному ожиданию (busy-waiting) операций, скорость выполнения которых значительно ниже производительности CPU, таких как обращение к диску или сетевые запросы. Подключенные библиотеки: FastAPI для создания API-интерфейса, Uvicorn для запуска ASGI-сервера с развертыванием FastAPI-приложения.

Третий раздел «Описание разработки приложения» содержит описание алгоритма функционирования приложения, структуры базы данных, архитектуры взаимодействия компонент серверной и клиентской частей приложения, процесса обучения модели и интеграции модели с остальными частями приложения.

Веб-приложение позволяет пользователям регистрировать свои профили, входить в систему, просматривать данные о категориях сервисов, выводить списки сервисов кокрентной категории (пользователь) или всех сервисов (администратор), получать информацию о конкретном сервисе (или всех сервисах у администратора) и мастере, работающем в этом сервисе, а также бронировать сервисные услуги, оставлять отзывы вручную или при помощи обученной модели ИИ. Вся соответствующая информация хранится в базе данных, откуда она предоставляется пользователю и куда пользователь может передавать свои данные путем взаимодействия с формами.

Веб-сервер для функционирования backend развернут на порту 8080, для функционирования frontend — на порту 3000, для взаимодействия с моделью — 8000.

При запуске программы и переходе по адресу localhost:3000/ пользовате-

ля пересылает на главную страницу сайта, где он может переключаться между вкладками «Главная», «О нас», войти в свой профиль, если таковой существует или же зарегистрироваться, забронировать услугу, выбрав мастера в соответсвующем сервисе и дату, оставить отзыв на странице профиля на вкладке «История заказов». Вся информация, доступная пользователю, получается из базы данных, также, как и вводимая им — передаётся в БД. Пароль, введенный пользователем при регистрации, хэшируется при помощи BCryptPasswordEncoder и также передается в базу данных.

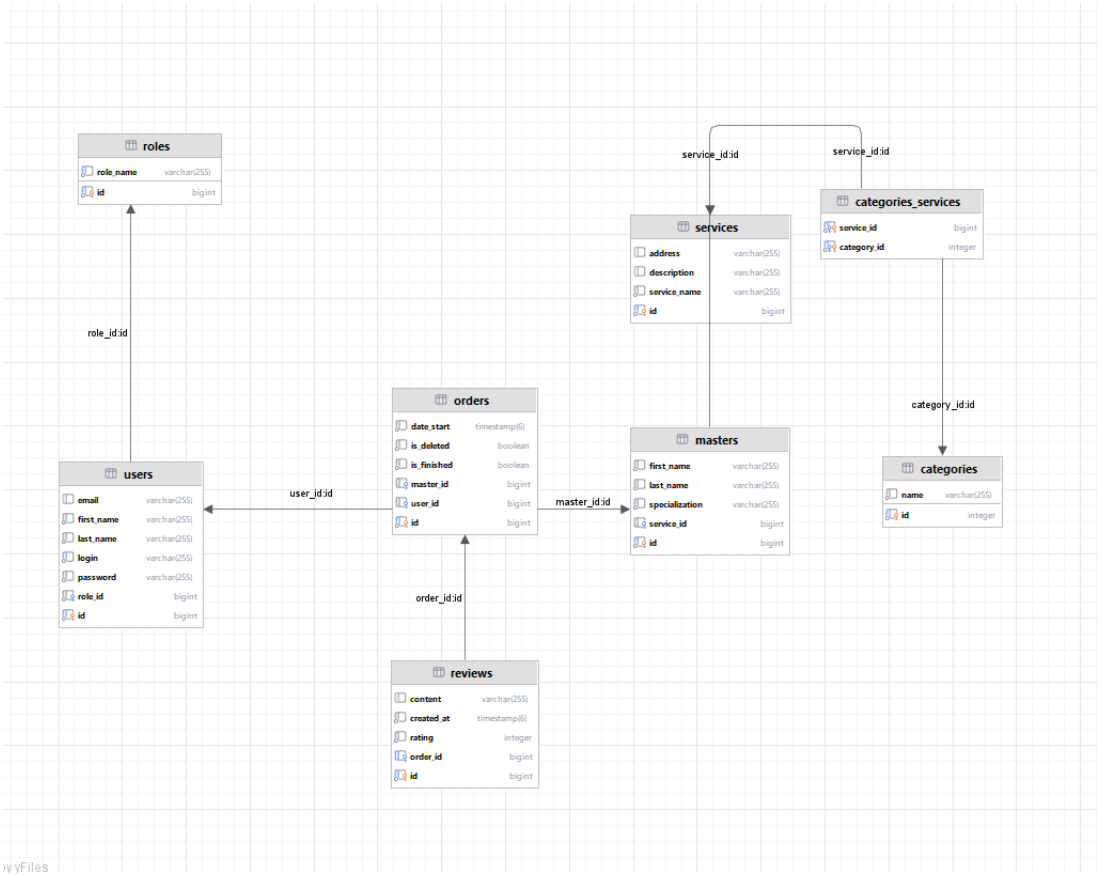

Рисунок 1 – Схема базы данных

Взаимодействие компонент серверной части приложения происходит в следующем порядке: от клиента приходит запрос, если в запросе есть сущность — она преобразуется в DTO, обрабатывается сервисом, через репозиторий изменяется (или создается) и сохраняется в БД.

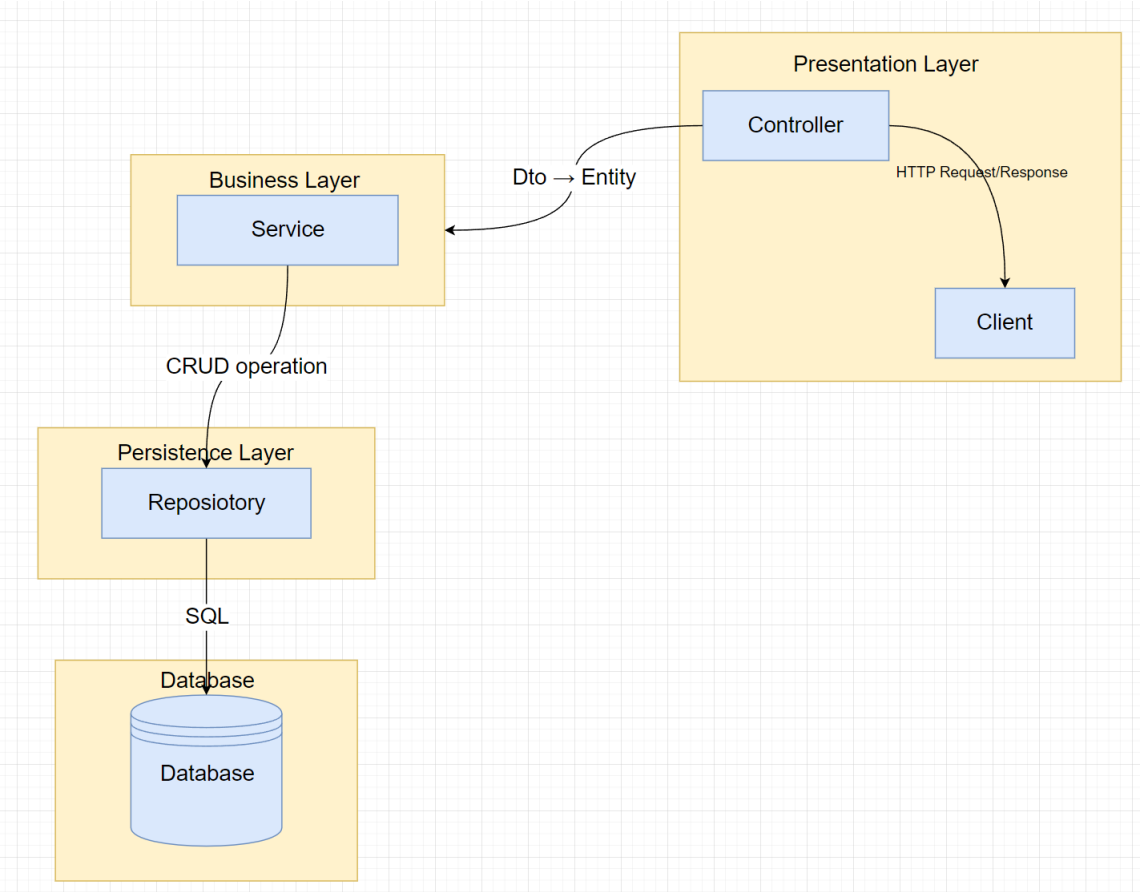

Рисунок 2 – Слоистая архитектура компонент серверной части приложения

Данные для обучения модели взяты с из датасета отзывов об организациях, опубликованных на Яндекс Картах. Описание датасета: состоит из 500 000 уникальных отзывов, содержит только отзывы на организации в России, доступные на Яндекс Картах, которые были опубликованы с января по июль 2023 года. Не содержит коротких односложных отзывов, тексты очищены от персональных данных (номеров телефонов, адресов почты).

Перед обучением была проведена подготовка датасета: удалены дублирующие значения, отзывы с рейтиногом 0, нормализацией путем смещения на единицу для соответствия индексации с нуля, выбораны релевантные колонки (текст и рейтинг), проведена балансировка датасета всвязи со значительным перевесом количества положительных отзывов над остальными.

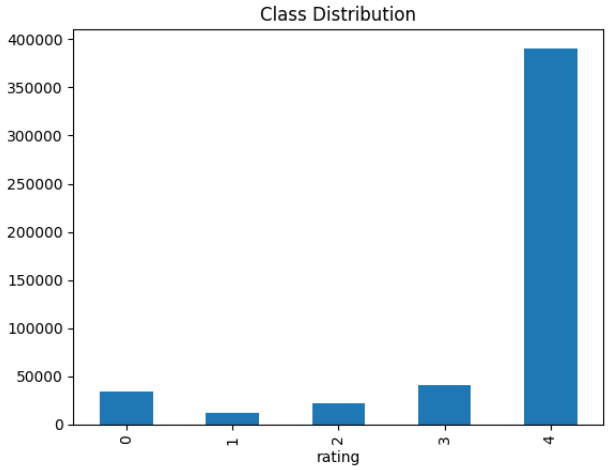

Рисунок 3 – Распределение отзывов по оценкам

Взаимодействие обученной модели с остальными компонентами приложения осуществляются через REST API: пользователь выбирает, как оценить отзыв (с помощью ИИ или вручную), текст отзыва передается на модель, ставится оценка, возвращается JSON с текстом и оценкой, создается DTO, передается на серверную часть, где создается сущность отзыва и помещается в БД, ответ вместе с созданным отзывом отправляется на клиентскую часть.

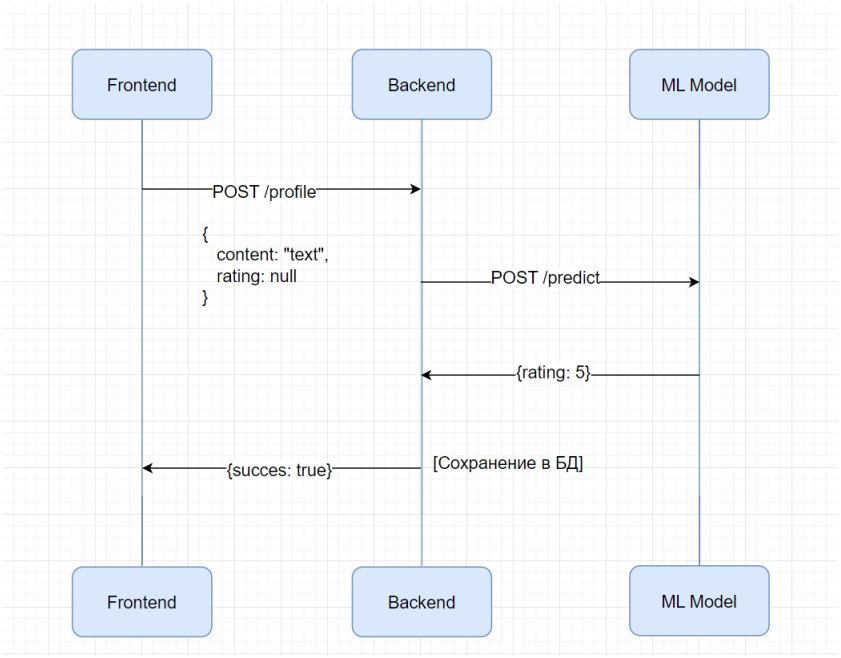

Рисунок 4 – Взаимодействие компонент приложения

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы было разработано веб-приложение, которое позволяет пользователям выбирать сервисы, мастеров и бронировать их услуги через интернет, оставлять отзывы о проделанной работе, оценка которых может быть произведена как непосредственно пользователем, так и с использованием обученной модели искусственного интеллекта.

Для достижения поставленной цели были рассмотрены и исследованы методы обработки запросов при помощи языка Java, технологии хранения данных, существующие технологии реализации внешней оболочки web-приложения, способы анализа человеческого языка, методы анализа тональности, методы интеграции обученной модели искусственного интелекта в web-приложения, что позволило подобрать оптимальное, современное решение и построить качественную, масштабируемую микросервисную архитектуру приложения с использованием Java Spring Boot, Spring Security, Spring Data JPA, React JS, FastAPI, все компоненты взаимодействуют друг с другом посредством REST API