УДК 004.9

АРХИТЕКТУРА КРАУДСОРСИНГОВОЙ ПЛАТФОРМЫ ДЛЯ ЗАКАЗА И ОЦЕНКИ УСЛУГ ТАЙНЫХ ПОКУПАТЕЛЕЙ

Вайнилович Ю.В., Овсянников И.В.
Межгосударственное ОУ ВО «Белорусско-Российский университет»,
г. Могилев, Беларусь

Статья посвящена разработке краудсорсинговой платформы для поиска «тайных покупателей». Описывается архитектурный подход к проектированию информационной системы, обосновывают выбор монолитной архитектуры и технологий для реализации серверной и клиентской частей приложения. Особое внимание уделяется вопросам безопасности и аутентификации пользователей. Представлена структурная схема развертывания программы, демонстрирующая физическую архитектуру приложения и взаимодействие между ее компонентами.

***Ключевые слова**: краудсорсинг, оценка персонала, тайный покупатель, информационная система, монолитная архитектура, ASP.NET, React, PostgreSQL, безопасность данных, аутентификация*

В условиях динамично меняющейся бизнес-среды коммерческие организации стремятся к максимизации прибыли, что невозможно без обеспечения высокого уровня квалификации персонала. Оценка сотрудников играет ключевую роль в поддержании и повышении качества их работы.

Многие компании, внедряя популярные механизмы оценки работы торговых сотрудников, сталкиваются с проблемами адаптивности и объективности процедур. Метод «тайный покупатель», основанный на принципе контрольной закупки и наблюдении с элементами имитации реальных профессиональных ситуаций, является наиболее распространенным инструментом оценки персонала в области продаж товаров и услуг [1].

Однако компании часто испытывают трудности при поиске квалифицированных тайных покупателей [2]. Для решения этой проблемы была разработана краудсорсинговая платформа «Mystery Shopper» – цифровое решение, учитывающее специфические потребности рынка оценки качества обслуживания [3].

Для проектирования ИС «Mystery Shopper» была выбрана монолитная архитектура, разделенная на три слоя: слой доступа к данным, слой бизнес-логики и API-слой. На слое доступа к данным располагаются сущности, отражающие модель базы данных и репозитории, осуществляющие операции с ними. К бизнес-логике относятся сервисы и различные вспомогательные компоненты, например, валидаторы. На данном слое выполняются действия, отражающие бизнес-процессы в системе, производится валидация данных и т.д. В API-слое находятся контроллеры, задача которых – принятие запросов, вызов необходимых сервисов и отправка ответов [4].

Выбор монолитной архитектуры обусловлен наличием большого количества тесно связанных сервисов, что сильно усложняет взаимодействие между ними в случае разделения на микросервисы. Монолитная архитектура также упрощает процесс разработки, запуска и поддержки системы

Для проектирования этой ИС был выбран архитектурный подход, т.к. для данной системы он является наиболее предпочтительным по ряду причин:

–архитектурный подход способствует созданию гибких систем, которые могут легко адаптироваться к изменениям требований, технологий и бизнес-процессов;

–архитектурное проектирование позволяет выявить и минимизировать риски на ранних стадиях разработки, что снижает вероятность проблем в будущем;

–эффективная архитектура помогает оптимально использовать ресурсы, такие как вычислительные мощности, память и сетевые ресурсы, что снижает затраты и повышает производительность;

–архитектурный подход облегчает интеграцию различных компонентов и систем, обеспечивая их совместимость и взаимодействие.

Серверная часть приложения разработана на платформе ASP.NET с использованием Entity Framework для упрощения работы с базой данных. Для хранения данных в системе используется СУБД PostgreSQL.

В разработке клиентской части приложения принято решение использовать React – фреймворк для разработки пользовательских интерфейсов, и его функциональные компоненты.

Выбор функциональных компонентов вместо классов обосновывается следующими преимуществами: улучшенная читаемость кода, более простая логика, легкая тестируемость и возможность использования хуков для более эффективного управления состоянием.

Для обеспечения эффективного взаимодействия между клиентской и серверной частями была выбрана библиотека axios, позволяющая легко выполнять HTTP-запросы, отправлять данные на сервер и обрабатывать ответы.

Важным аспектом разработки приложения является обеспечение безопасности пользовательских данных, в частности, паролей. Для хеширования паролей была выбрана библиотека bcrypt. Эта библиотека предоставляет надежные алгоритмы хеширования, что делает хранение паролей в базе данных более безопасным.

Для обеспечения аутентификации и авторизации пользователей в приложении применяется аутентификация с помощью JSON Web Tokens (JWT), которые используются для подтверждения личности отправителя запроса.

На рисунке 1 представлена структурная схема развертывания программы. Диаграмма развертывания краудсорсинговой платформы «Mystery Shopper» представляет физическую архитектуру приложения, демонстрируя основные узлы и взаимодействие между ними. Взаимодействие начинается с клиентского устройства, которое может быть представлено ПК, планшетом или смартфоном. Пользователь взаимодействует с приложением через браузер, отправляя запросы и получая данные посредством интерфейса. Пользовательский интерфейс (UI), разработанный с использованием React, отвечает за визуализацию данных и удобство взаимодействия.

Передача данных между клиентским устройством и сервером осуществляется через защищённое соединение по протоколу HTTPS. Этот протокол обеспечивает шифрование данных, защищая их от несанкционированного доступа и перехвата. Сервер бизнес-логики, работающий на ASP.NET, принимает запросы, обрабатывает их в соответствии с бизнес-логикой системы и возвращает ответ. Здесь также происходит выполнение CRUD-операций (создание, чтение, обновление и удаление данных), взаимодействуя с базой данных через Entity Framework.

Сервер базы данных, реализованный на PostgreSQL, служит хранилищем всей информации системы Mystery Shopper. Здесь сохраняются данные пользователей, заказы, отчеты и другие данные.

Такое разбиение системы на логически обособленные уровни обеспечивает стабильность, безопасность и производительность приложения. Использование современных технологий, таких как React, ASP.NET, PostgreSQL и Entity Framework, позволяет достичь высокой надёжности, а защищённое соединение через HTTPS гарантирует безопасность данных. Предложенная архитектура платформы «Mystery Shopper» ориентирована на удобство пользователя, её легко масштабировать и развивать, что делает её эффективным решением для управления заказами и отчетами.

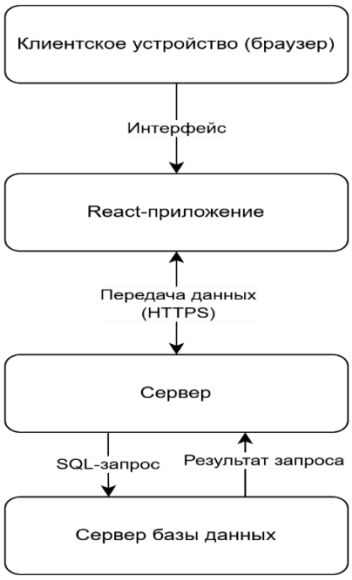

Рисунок 1 – Структурная схема развертывания

Список использованных источников

1. Еремина А.С. «Тайный покупатель» как метод оценки персонала // Российская наука и образование сегодня: проблемы и перспективы. 2023. № 5(53). С. 56-59.

2. Вайнилович Ю.В., Овсянников И.В. Оптимизация процесса оценки качества обслуживания: краудсорсинговое решение для тайных покупателей // Энергетика, информатика, инновации-2024 (математическое моделирование и информационные технологии в производстве и строительстве, микроэлектроника и оптотехника): XIV Международная научно-техническая конференция: сборник трудов. Смоленск: Б.и., 2024. С. 32-35.

3. Овсянников И.В., Вайнилович Ю.В. Технология использования краудсорсинговой платформы для заказа и оценки услуг тайных покупателей // Молодая наука. 2024: Сборник материалов XV Национальной научно-практической конференции с международным участием молодых ученых, аспирантов и студентов, 2024. С. 102-105.

4. 15 советов по архитектуре сайта: как составить SEO-структуру, навигацию, разметку и перелинковку URL.: https://pr-cy.ru/news/p/7111-15-sovetov-po-seo-arkhitekture-sayta (дата обращения: 09.02.2025).

ARCHITECTURE OF THE CROWDSOURCING PLATFORM TO ORDER AND EVALUATE THE SERVICES OF SECRET BUYERS

Vainilovich Yu.V., Ovsyannikov I.V.

Interstate Educational Institution of Higher Education «Belarusian-Russian University», Mogilev, Belarus

The article is devoted to the development of a crowdsourcing platform for the search for "secret buyers". An architectural approach to the design of an information system is described, and the choice of a monolithic architecture and technologies for implementing the server and client parts of the application is justified. Special attention is paid to the issues of user security and authentication. A block diagram of the program deployment is presented, demonstrating the physical architecture of the application and the interaction between its components.

***Keywords**: crowdsourcing, personnel assessment, secret buyer, information system, monolithic architecture, ASP, NET, React, PostgreSQL, data security, authentication*