Лабораторная работа №3

Часть 1.

Проектирование архитектуры

1. Тип приложения

Веб-приложение для бронирования столиков в ресторане.

2. Стратегия развертывания

Разработанное приложение может быть развернуто на собственных серверах, в облаке или на платформе виртуализации. В зависимости от предпочтений и требований бизнеса можно выбрать оптимальный вариант развертывания.

3. Обоснование выбора технологий

Node.js является эффективным и масштабируемым серверным фреймворком, который позволяет разрабатывать высокопроизводительные и масштабируемые веб-приложения. Он также обеспечивает возможность использования JavaScript как языка программирования как на стороне сервера, так и на стороне клиента, что позволяет легко обмениваться данными между сервером и клиентом.

React.js является мощной библиотекой JavaScript для разработки пользовательского интерфейса. Он обеспечивает модульность, повторное использование компонентов и эффективную отрисовку интерфейса, что делает его отличным выбором для создания динамичных и отзывчивых веб-приложений.

Express.js является популярным фреймворком для разработки веб-приложений на Node.js. Он обладает простым и интуитивно понятным интерфейсом, что упрощает создание маршрутов, обработку запросов и реализацию бизнес-логики приложения.

MongoDB является гибкой и масштабируемой базой данных NoSQL, которая хранит данные в формате JSON-подобных документов. Она хорошо подходит для приложений, требующих гибкой схемы данных и быстрого масштабирования.

4. Показатели качества

Доступность - приложение должно быть доступно для пользователей в любое время без существенных задержек или сбоев.

Масштабируемость - приложение должно быть способно обрабатывать растущую нагрузку и масштабироваться горизонтально или вертикально при необходимости.

Безопасность - приложение должно обеспечивать защиту данных пользователей и предотвращать несанкционированный доступ к системе.

Отзывчивость - приложение должно быстро реагировать на действия пользователей и обеспечивать плавное взаимодействие с интерфейсом.

Надежное функционирование системы должно быть обеспечено выполнением организационно-технических мероприятий, таких как:

- использование лицензионного программного обеспечения;
- организация бесперебойного питания путем использования блоков бесперебойного питания;
- обеспечение минимального времени восстановления после отказа.

Загрузка и отображение основного окна программы не должны превышать 5 секунд.

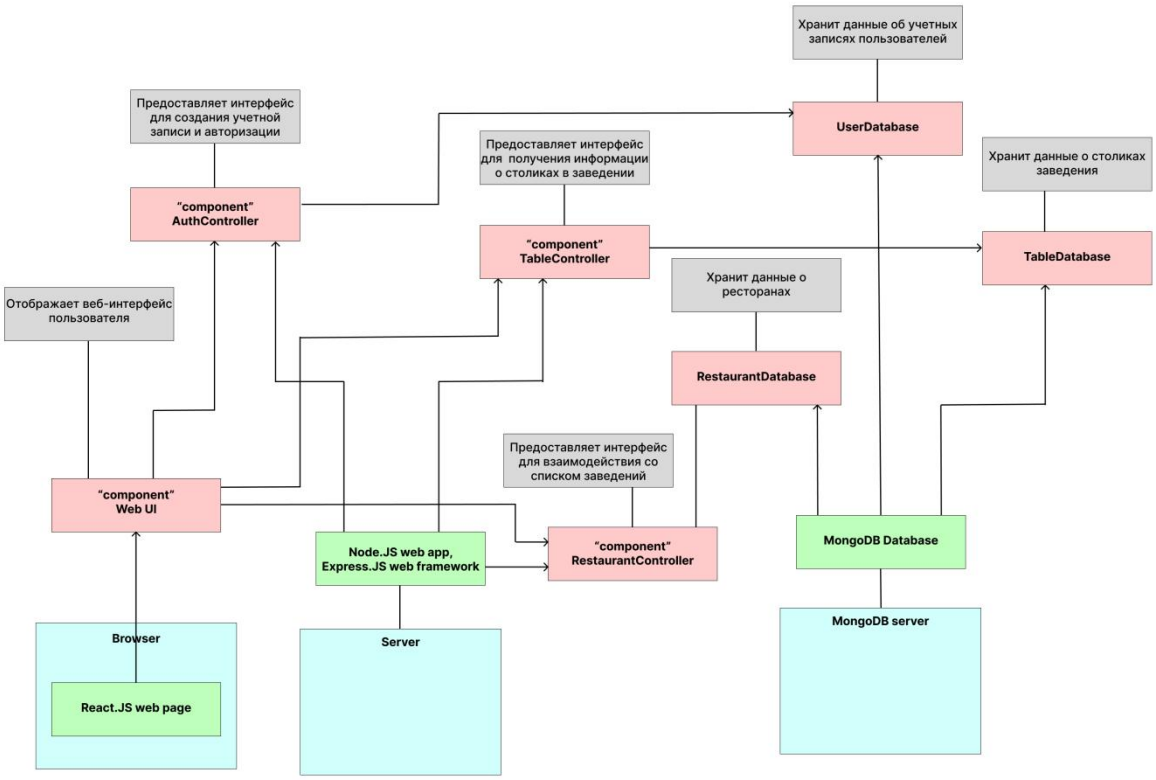
5. Реализация сквозной функциональности

Аутентификация и авторизация - реализация системы аутентификации и авторизации пользователей для защиты данных и обеспечения безопасности приложения.

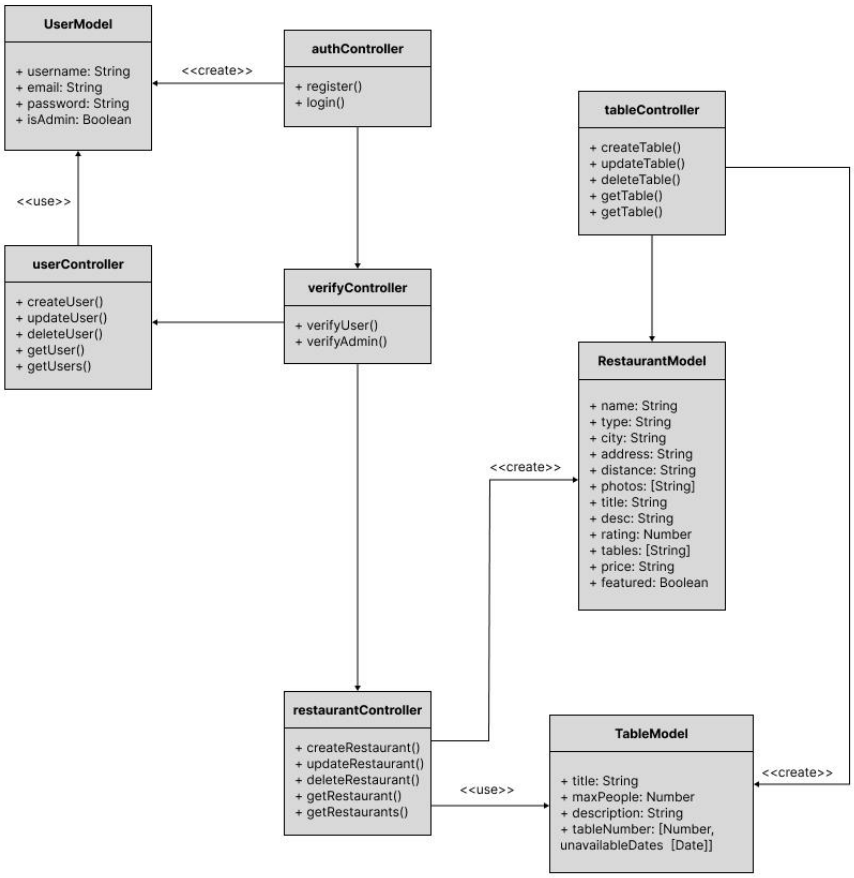
Управление столиками - реализация функциональности для бронирования, отмены и управления столиками в ресторане.

Уведомления - реализация системы уведомлений для отправки подтверждений о бронировании, напоминаний и других уведомлений связанных с бронированием столиков.

6. Структурная схема приложения



Часть 2.
Анализ архитектуры
Диаграмма классов:



Часть 3.
Сравнение и рефакторинг

Можно сделать вывод, что архитектуры "As is" (существующая архитектура) и "To be" (желаемая архитектура) имеют схожие черты и принципы. Оба варианта архитектуры предусматривают использование клиент-серверной модели, где клиентское приложение взаимодействует с сервером для получения данных о ресторане и бронирования столиков. Это обеспечивает отделение фронтенда от бэкенда и повышает гибкость и масштабируемость приложения. Кроме того, оба варианта архитектуры рассматривают использование базы данных для хранения информации о ресторане, столиках и бронированиях. Это позволяет эффективно управлять данными и обеспечивает надежность и целостность информации.