Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И. В. Рудаков
(И.О.Фамилия)
« _____ » _____ 20 ____ г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине Распределённые системы обработки информации

Студент группы ИУ7И-21М

Диас Сергей Рамирович
(Фамилия, имя, отчество)

Тема курсового проекта Система бронирования отелей

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) Кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание
Разработать прототип системы бронирования отелей на базе веб-интерфейса. Система должна состоять из микросервисов, каждый из которых отвечает за свою задачу: сервис пользовательского интерфейса; сервис авторизации;сервис информации о пользователе;сервис отелей; сервис бронировании; сервис статистики; Каждый сервис при необходимости может иметь доступ к связанной с ним базе данных, но не должен иметь доступа к базам данных других сервисов. Все запросы между сервисами требуют авторизацию. Все ошибки должны обрабатываться; в случае недоступности некритичного функционала должна осуществляться деградация функциональности. Все сервисы должны собираться в Docker образы и разворачиваться при помощи Kubernetes..

Оформление курсового проекта:

Расчетно-пояснительная записка на 20-30 листах формата А4.

Дата выдачи задания « __ » _____ 2024 г.

Руководитель курсового проекта _____ А.А.Ступников
(Подпись, дата) (И.О.Фамилия)

Студент _____ С.Р.Диас
(Подпись, дата) (И.О.Фамилия)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение эвм и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Система бронирования отелей»

Студент группы **ИУ7И-21М** _____ С.Р.Диас
(Подпись, дата) (И.О.Фамилия)

Руководитель _____ А.А.Ступников
(Подпись, дата) (И.О.Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель данной курсовой работы заключается в разработке прототипа веб-системы для бронирования гостиниц. Система должна обеспечивать пользователям возможность выбора и бронирования номеров в различных отелях, а также управления своими бронированиями. Для достижения этой цели необходимо решить следующие задачи:

- Провести анализ требований к системе и определить ключевые функциональные возможности.
- Изучить и выбрать подходящие технологии и инструменты для разработки веб-приложения.
- Разработать прототип веб-системы, включающий интерфейс пользователя и основные функции бронирования.

Аналитическая часть

Постановка задачи

Разрабатываемая система должна предоставлять пользователям возможность бронирования отелей через веб-интерфейс. Система будет состоять из нескольких микросервисов, каждый из которых отвечает за определенную задачу: сервис пользовательского интерфейса для взаимодействия с пользователями; сервис авторизации для управления доступом; сервис информации о пользователе для хранения и обработки данных пользователей; сервис отелей для предоставления информации о доступных номерах; сервис бронирования для обработки запросов на бронирование; и сервис статистики для сбора и анализа данных о пользователях и бронированиях.

Описание системы

Результатом проекта должна стать система, которая связывает пользователей с отелями для бронирования номеров. В системе предусмотрены два типа пользователей: администраторы и клиенты. Клиенты смогут просматривать список отелей, получая общую информацию о каждом из них. Администраторы будут иметь возможность видеть статистику о бронированиях.

Общие требования к системе

1. Разрабатываемое программное обеспечение должно обеспечивать работу системы круглосуточно, 7 дней в неделю, 365 дней в году (24/7/365) с гарантией доступности не менее 99.9% в год. Максимально допустимое время простоя системы в течение года составляет $24 \cdot 365 \cdot 0.001 = 8.76$ часов.
2. Время на восстановление системы после сбоя не должно превышать 15 минут.
3. Каждый микросервис должен иметь возможность автоматического восстановления после сбоев.
4. Система должна поддерживать «горячее» переконфигурирование, позволяя добавлять новые микросервисы в процессе работы без необходимости перезапуска.
5. Обеспечить устойчивость системы к сбоям через отказоустойчивость микросервисов

Требования к функциональным характеристикам

1. Медианное время отклика системы на запросы пользователей о получении информации не должно превышать 2.5 секунды.
2. Медианное время отклика системы на запросы, связанные с добавлением или изменением информации на портале, не должно превышать 5 секунд.
3. Медианное время отклика системы на действия пользователей должно составлять менее 1 секунды при условии работы на рекомендованной аппаратной конфигурации, задержках между взаимодействующими сервисами менее 0.3 секунды и при количестве одновременно работающих пользователей менее 500 на каждом сервере.
4. Система должна обеспечивать корректную работу в современных браузерах, чтобы как минимум 90% пользователей Интернета могли пользоваться ею без потери функциональности.

Функциональные требования к системе с точки зрения пользователя

Система должна обеспечивать реализацию следующих функций:

1. Система должна предоставлять аутентификацию пользователей.
2. Система должна различать пользователей по двум ролям:
    - Клиент
    - Администратор
3. Система должна предоставлять клиенту следующие функции:
    - Просмотр списка доступных отелей
    - Управление бронированиями
    - Получение информации о ценах
    - Просмотр статуса программы лояльности
    - Просмотр количества своих бронирований
    - Просмотр процента скидки по программе лояльности
4. Система должна предоставлять администратору следующие функции:
    - Просмотр статистики

Входные данные клиента:

- Имя пользователя
- Email
- Дата начала бронирования
- Дата окончания бронирования

Входные данные администратора:

- Email

Выходные данные для клиента:

- Список отелей
- Информация о гостинице
- Статус программы лояльности
- Количество бронирований
- Процент скидки по программе лояльности
- Список бронирований

Выходные данные для администратора:

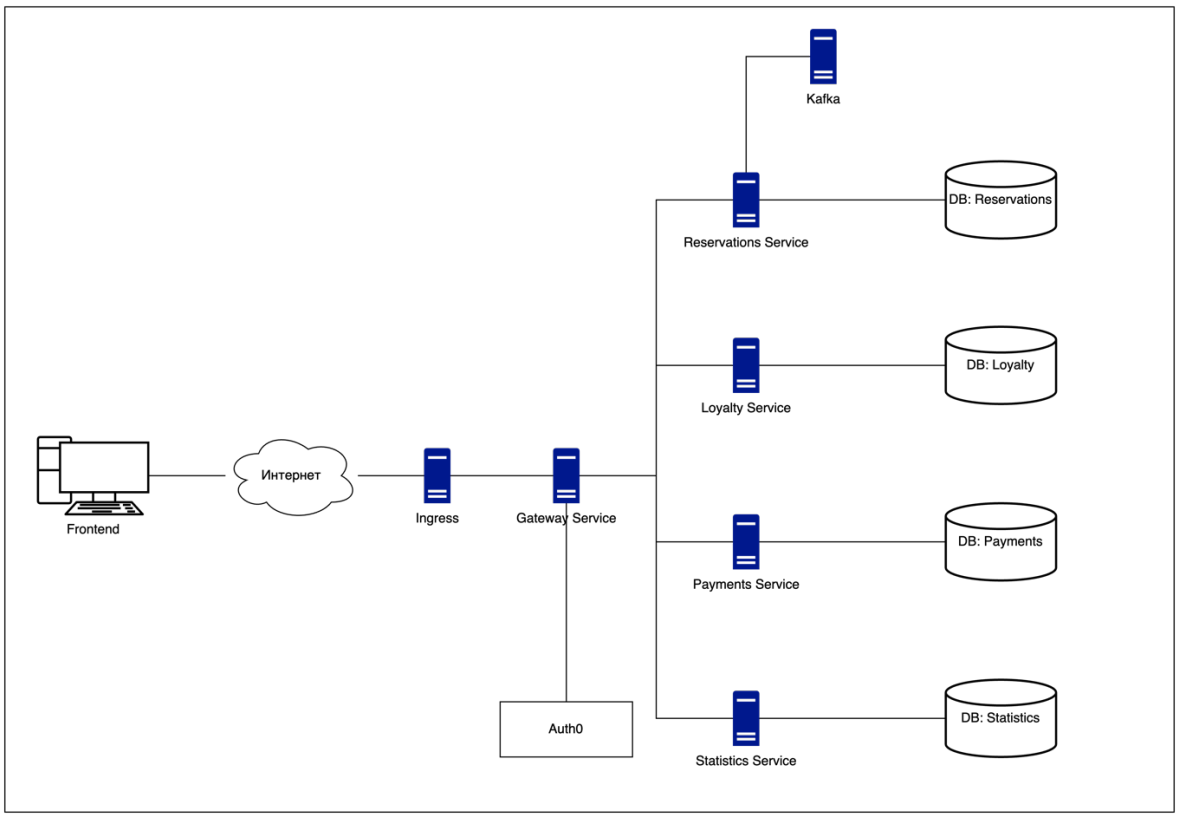
- Количество бронирований
- Количество отмен

Требования к программной реализации

1. Создать сервис, выполняющий функцию Identity Provider, реализовать протокол OpenID Connect. Реализовать scope:
2. openid – всегда передается, в ответ возвращается JWT;
3. profile – базовая информация о пользователе;
4. email – email пользователя.

5. Для получения токена на Identity Provider реализовать Authorization Flow. UI рассматривается как 3rd party application и авторизация пользователя так же выполняется через Authorization Flow.
6. Все методы /api/** (кроме /api/v1/authorize и /api/v1/callback) на всех сервисах закрыть token-based авторизацией. 4Так же реализовать ролевую модель:
7. руками создать пользователя с ролью Admin;
8. для зарегистрированных пользователей по-умолчанию роль User.
9. На Identity Provider сделать возможность создания новых пользователей (запрос от пользователя с ролью Admin).
10. Каждый сервис при получении запроса выполняет валидацию JWT токена с помощью JWKs, которые он получает из Identity Provider (запрос к Identity Provider делать не нужно).
11. JWT токен пробрасывать между сервисами, при получении запроса валидацию токена так же реализовать через JWKs.
12. Убрать заголовок X-User-Name и получать пользователя из JWT-токена.
13. Если авторизация некорректная (отсутствие токена, ошибка валидации JWT токена, закончилось время жизни токена (поле exp в payload)), то отдавать 401 ошибку.
14. Для функционала, реализованного в рамках лабораторных работ, реализовать пользовательский интерфейс как Single Page Application (React, Angular, Vue) или мобильный клиент. Использование CSS обязательно.
15. Выделить сервис статистики: в него отправлять через Kafka информацию о выполненных действиях. В зависимости от задания по пришедшим данным строить отчет, доступ к которому должен быть только у пользователя с ролью Admin.
16. Как и в ЛР #4 нужно развернуть Managed Kubernetes Cluster и настроить Ingress Controller (для публикации сервисов наружу можно использовать только Ingress).
17. Для деплоя использовать helm charts, они должны быть универсальным для всех сервисов и отличаться лишь набором параметров запуска.

Топология системы



Разрабатываемая система состоит из фронтэнда и 4 подсистем:

- Ingress
- Gateway Service
- Reservations Service
- Payments Service
- Loyalty Service
- Statistics Service

Фронтенд: Пользовательский интерфейс, который отправляет запросы к бекенду.

Ingress: это компонент, который управляет входящими запросами к микросервисам. Он служит точкой доступа для внешних пользователей, направляя их запросы к соответствующим сервисам в зависимости от URL-адреса или других параметров.

Gateway Service: представляет собой центральную точку доступа к системе, объединяющую запросы от клиентов и перенаправляющую их к соответствующим микросервисам.

Сервис бронирований (Reservations Service): отвечает за управление процессом бронирования отелей. Он обрабатывает запросы на создание и отмену бронирований.

Сервис платежей (Payments Service): отвечает за обработку всех финансовых транзакций, связанных с бронированием.

Сервис лояльности (Loyalty Service): управляет программами лояльности для клиентов. Он отслеживает активность пользователей, рассчитывает бонусы и скидки, а также предоставляет информацию о статусе программы лояльности.

Сервис статистики (Statistics Service): собирает и анализирует данные о действиях пользователей. Он предоставляет отчеты и аналитические данные.

Топология системы позволяет визуализировать связи между ее компонентами.

Конструкторский раздел

Концептуальный дизайн

Концептуальный дизайн системы содержит наиболее общие схемы описания функционала приложения с точки зрения пользователей. Одной из таких схем является IDEF0-модель и графические модели, входящие в нее. На рисунке отображена контекстная диаграмма верхнего уровня, которая обеспечивает наиболее общее или абстрактное описание работы системы. Данный вид диаграммы позволяет формализовать описание запросов пользователя и ответов системы на данные запросы, отобразив систему в виде “черного” ящика.

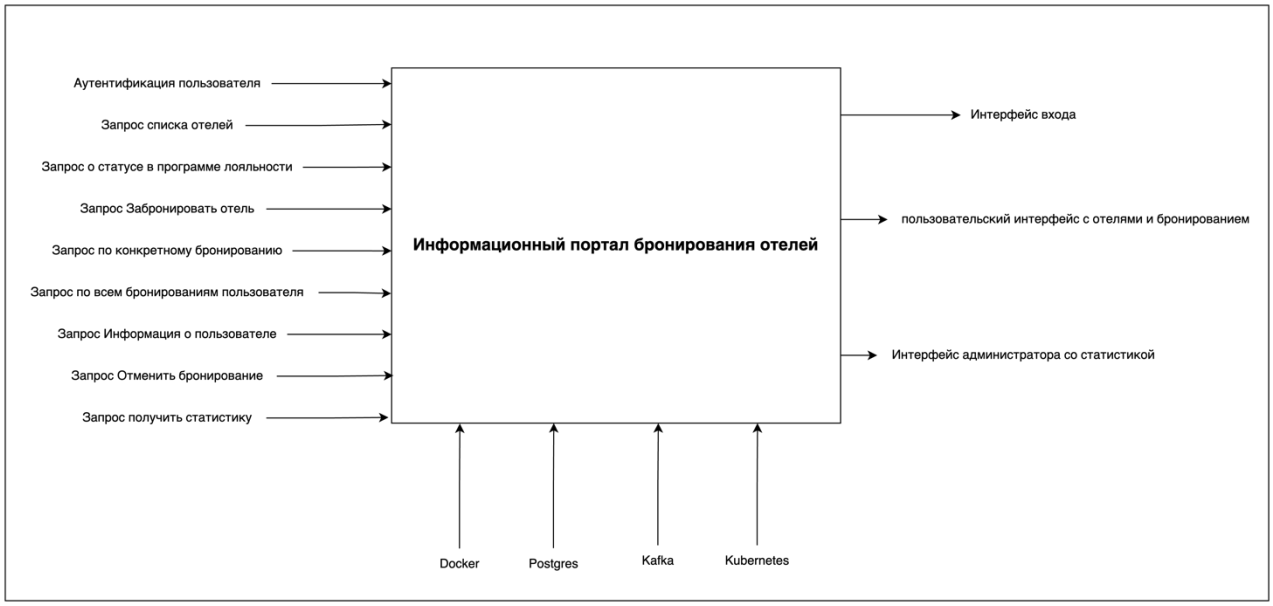


Рис. 1 IDEF0

Для уточнения деталей работы системы применяется декомпозиция функций, отображенных на диаграмме верхнего уровня, при помощи создания дочерних диаграмм.

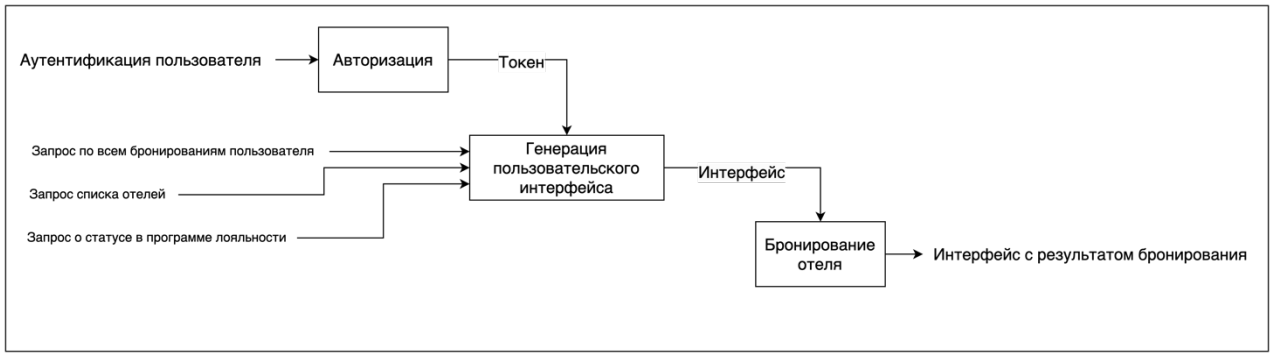


Рис. 2 IDEF0

Спецификация классов

Таблица 1 – Атрибуты класса StatisticsEntity

Атрибуты	Имя	Тип	Описание
1	name	private: string	уникальное название статистики
2	number	private: int	значение статистики

Таблица 2 – Атрибуты класса HotelEntity

Атрибуты	Имя	Тип	Описание
1	id	private: int	уникальный идентификатор отеля
2	hotel_uid	private: uuid	уникальный идентификатор в формате UUID
3	name	private: string	название отеля
4	country	private: string	страна
5	city	private: string	город
6	address	private: string	адрес
7	stars	private: int	количество звёзд
8	price	private: int	цена за ночь

Таблица 3 – Атрибуты класса ReservationEntity

Атрибуты	Имя	Тип	Описание
1	id	private: int	уникальный идентификатор бронирования
2	reservation_uid	private: uuid	уникальный идентификатор в формате UUID
3	username	private: string	имя пользователя, сделавшего бронирование
4	payment_uid	private: uuid	уникальный идентификатор платежа
5	hotel_id	private: int	идентификатор отеля (ссылка на таблицу hotels)
6	status	private: string	статус бронирования (PAID, CANCELED)
7	start_date	private: timestamp with time zone	дата начала бронирования
8	end_date	private: timestamp with time zone	дата окончания бронирования

Таблица 4 – Атрибуты класса PaymentEntity

Атрибуты	Имя	Тип	Описание
1	id	private: int	уникальный идентификатор платежа
2	payment_uid	private: uuid	уникальный идентификатор в формате UUID
3	status	private: string	статус платежа (PAID, CANCELED)
4	price	private: int	сумма платежа

Таблица 5 – Атрибуты класса LoyaltyEntity

Атрибуты	Имя	Тип	Описание
1	id	private: int	уникальный идентификатор программы лояльности
2	username	private: string	имя пользователя (уникальное)
3	reservation_count	private: int	количество сделанных бронирований
4	status	private: string	статус программы лояльности (BRONZE, SILVER, GOLD)
5	discount	private: int	процент скидки

Диаграммы последовательности действий

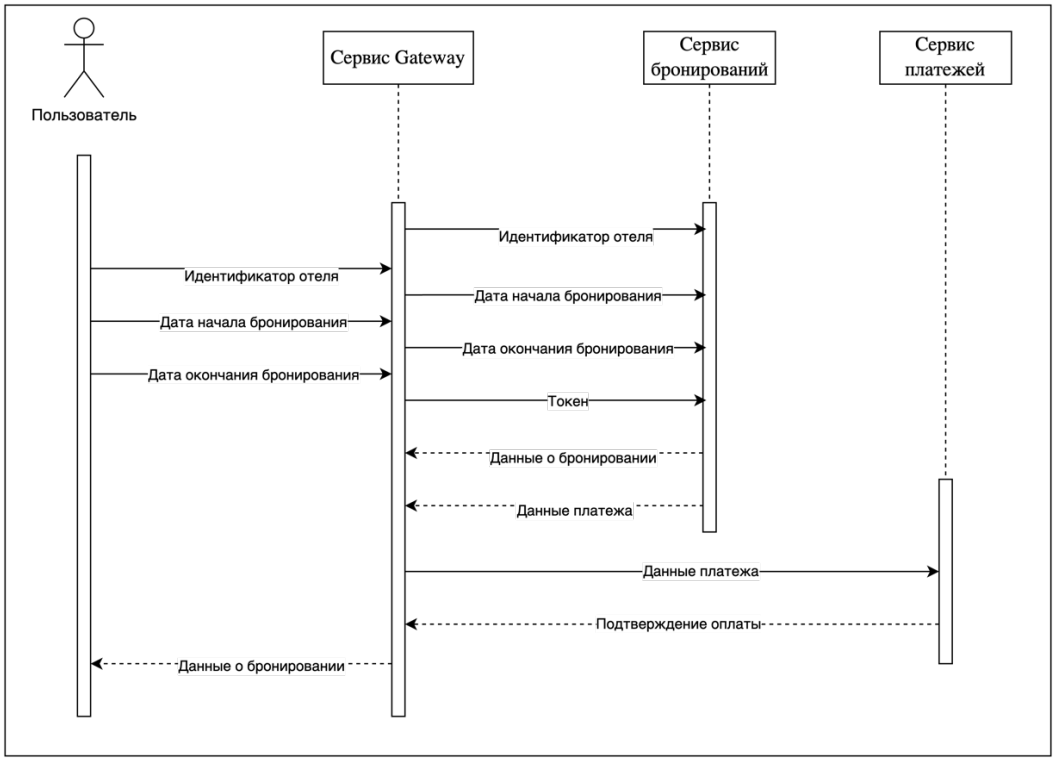


Рис. 3 Диаграмма последовательности действий при бронировании

Технологическая часть

Выбор средств разработки

Выбор СУБД

В рамках разработки системы бронирования отелей был выбран реляционная система управления базами данных (СУБД) PostgreSQL [1]. Данный выбор обусловлен следующими факторами:

1. **Надежность и стабильность**: PostgreSQL является одной из самых стабильных СУБД на рынке, обеспечивая высокую степень целостности данных и надежности, что критично для систем, работающих с финансовыми транзакциями и личной информацией пользователей.
2. **Поддержка сложных запросов**: PostgreSQL предлагает мощные возможности для выполнения сложных SQL-запросов, что позволяет эффективно обрабатывать данные о пользователях, отелях, бронированиях и платежах.
3. **Расширяемость**: PostgreSQL поддерживает множество расширений, которые могут быть использованы для добавления дополнительных функциональных возможностей, таких как поддержка JSONB для хранения и обработки неструктурированных данных.
4. **Поддержка ACID-принципов**: СУБД гарантирует выполнение транзакций в соответствии с принципами атомарности, согласованности, изолированности иDurability (ACID), что обеспечивает надежное и безопасное управление данными.
5. **Сообщество и поддержка**: Большое и активное сообщество разработчиков PostgreSQL обеспечивает доступ к обширной документации и ресурсам, что облегчает процесс разработки и устранения возможных проблем.
6. **Геопространственные возможности**: Для будущих улучшений системы, таких как интеграция с картографическими сервисами, PostgreSQL предлагает поддержку геопространственных данных через расширение PostGIS.

Таким образом, выбор PostgreSQL в качестве СУБД для системы бронирования отелей обоснован её функциональными возможностями, надежностью и поддержкой со стороны сообщества. [1]

Выбор языка разработки компонентов

Для разработки компонентов системы бронирования отелей был выбран язык программирования **JavaScript** с использованием **Node.js** [2] для серверной части и **React [3]** для клиентской части. Выбор этих технологий обусловлен следующими причинами:

1. **Широкая популярность и поддержка**: JavaScript является одним из самых популярных языков программирования в веб-разработке. Он имеет большое сообщество и множество библиотек и фреймворков, что облегчает разработку и поддержку системы.
2. **Асинхронная обработка**: Node.js основан на неблокирующем вводе/выводе, что делает его идеальным выбором для приложений, требующих высокой производительности и способности обрабатывать множество одновременных соединений, таких как система бронирования с высоким трафиком.
3. **Единая кодовая база**: Использование JavaScript как на клиентской, так и на серверной стороне позволяет разработчикам работать с одной кодовой базой, что упрощает разработку, тестирование и отладку приложения.

4. **Гибкость и масштабируемость**: Node.js предоставляет гибкость в проектировании архитектуры приложения и позволяет легко масштабировать систему, добавляя новые компоненты и микросервисы по мере необходимости.
5. **Интуитивно понятный и быстрый процесс разработки**: React позволяет создавать интерфейсы, основанные на компонентах, что ускоряет разработку и улучшает производительность клиентской части приложения. Возможность повторного использования компонентов упрощает обновление и поддержку интерфейса.
6. **Совместимость с современными стандартами**: JavaScript и его экосистема поддерживают современные стандарты веб-разработки, включая ES6+, что позволяет использовать современные функции языка и улучшает читаемость кода.

Таким образом, выбор JavaScript с использованием Node.js [2] и React [3] как языков разработки компонентов системы бронирования отелей оправдан их популярностью, производительностью и возможностями для создания современных веб-приложений.

Выбор фреймворка для разработки портала

Для разработки портала был выбран фреймворк React, который обеспечивает гибкость, высокую производительность и возможность создания интерактивного пользовательского интерфейса. React позволяет эффективно управлять состоянием приложения и обновлять интерфейс в ответ на изменения данных, что является важным для динамичных веб-систем. [3]

Для реализации функций аутентификации и авторизации пользователей была интегрирована платформа Auth0. Использование Auth0 упрощает процесс управления пользователями и обеспечивает надежную защиту данных. Платформа предлагает множество готовых решений для аутентификации через социальные сети и другие методы, что значительно ускоряет процесс разработки и повышает безопасность приложения. [4]

Таким образом, выбор React в сочетании с Auth0 позволяет создать современный и безопасный портал для системы бронирования отелей.

Сборка и деплой системы

Система бронирования отелей будет собираться и разворачиваться с использованием **Docker** [5] и **Kubernetes [6]**, что обеспечит модульность, масштабируемость и легкость в управлении микросервисами. Процесс сборки и деплоя системы включает несколько ключевых этапов:

1. **Создание Docker-образов:**
    - Каждый микросервис (например, сервисы для бронирования, платежей, лояльности и статистики) будет упакован в отдельный Docker-образ. Это позволяет создать изолированную среду выполнения для каждого сервиса, что снижает риск конфликтов между зависимостями и упрощает развертывание.
    - Dockerfile для каждого сервиса будет содержать инструкции по установке необходимых зависимостей, компиляции кода и запуску приложения.
2. **Конфигурация Docker Compose:**
    - Для локальной разработки будет использоваться файл docker-compose.yml, который описывает все сервисы и их взаимодействия. Это позволит разработчикам легко запускать всю систему на своей машине для тестирования и отладки.

- Docker Compose также поможет настроить сети между контейнерами и указать зависимости между сервисами.

3. **Оркестрация с помощью Kubernetes:**
   - Для управления развертыванием и масштабированием сервисов в продакшене будет использоваться Kubernetes. Он обеспечит автоматическое развертывание, управление конфигурацией и мониторинг состояния контейнеров.
   - Каждый сервис будет развернут в отдельном Pod, что позволит масштабировать их независимо друг от друга в зависимости от нагрузки.

Таким образом, процесс сборки и деплоя системы обеспечит высокую степень автоматизации, надежности и гибкости, что важно для поддержки современного веб-приложения и его масштабируемости.

Заключение

В ходе работы была разработана система бронирования отелей, основанная на сервис-ориентированной архитектуре с использованием микросервисов. Система включает в себя компоненты для управления отелями, бронированиями и программами лояльности, обеспечивая пользователям удобный интерфейс для взаимодействия.

Для обеспечения гибкости и масштабируемости все сервисы были контейнеризированы с использованием Docker и развернуты через Kubernetes. Это позволяет эффективно управлять ресурсами и повышает надежность системы.

СПИСОК ЛИТЕРАТУРЫ

1. Group P.G.D. // PostgreSQL. 2024. URL: https://www.postgresql.org/ (accessed: 28.09.2024).
2. Node.js - run JavaScript everywhere [Electronic resource] // Node.js -. URL: https://nodejs.org/en (accessed: 28.09.2024).
3. React [Electronic resource] // React Blog RSS. URL: https://react.dev/ (accessed: 28.09.2024).
4. Secure access for everyone. but not just anyone. [Electronic resource] // Auth0. Auth0. URL: https://auth0.com/ (accessed: 28.09.2024).
5. Accelerated Container Application Development [Electronic resource] // Docker. 2024. URL: https://www.docker.com/ (accessed: 28.09.2024).
6. Production-grade container orchestration [Electronic resource] // Kubernetes. 2024. URL: https://kubernetes.io/ (accessed: 28.09.2024).