Проектная работа по курсу Otus System Design студента Бурдина А.Ю.

Оглавление

1. Анализ требований

- Определите функциональные и нефункциональные требования.
- Укажите целевые метрики (например, доступность 99.9%, задержка обработки запросов ≤ 200 мс).

Функциональные требования:

- Пользователь выбирает желаемый город, период проживания, функциональные характеристики объекта размещения, бюджет, а система обрабатывает его заявку в ответ предоставляет последовательный список объектов размещения, подходящий под заданные условия, с указанием стоимости и возможности бронирования.
- Для создания заявки на поиск размещений пользователь может использовать браузер, мобильное приложение
- В качестве пользователя может также выступать партнерский сервис для перепродажи услуг -порталы путешествий, порты авиакомпаний
- Пользователь может просмотреть карточку объекта с указанием функциональных характеристик объекта
- Пользователь может забронировать понравишься объект размещения.
- Оплата объекта размещения реализуется через сторонние сервисы и не рассматривает подробно в рамках данного проекта.
- Администратор со стороны объектов размещения может создавать карточку объекта, редактировать карточку объекта, указывать информацию о занятости номеров объекта размещения. Обновление информации о занятости номеров объектов размещения также может реализоваться через API доступное для интеграции со сторонними системами.

Нефункциональные требования:

- Система должна выдерживать пиковую нагрузку в 50 000 запросов в секунду и обеспечивать согласованность данных (из задания).
- 50 000 запросов в секунду, а не в пиковые часы среднее количество запросов предположим, что составляет в 5 раз меньше ~ 15 000 запросов в секунду
- Доступность системы 99.9%
- Задержка обработки запросов (95% перцентиль) ≤ 200 мс, Задержка обработки запросов (99% перцентиль) ≤ 200 мс

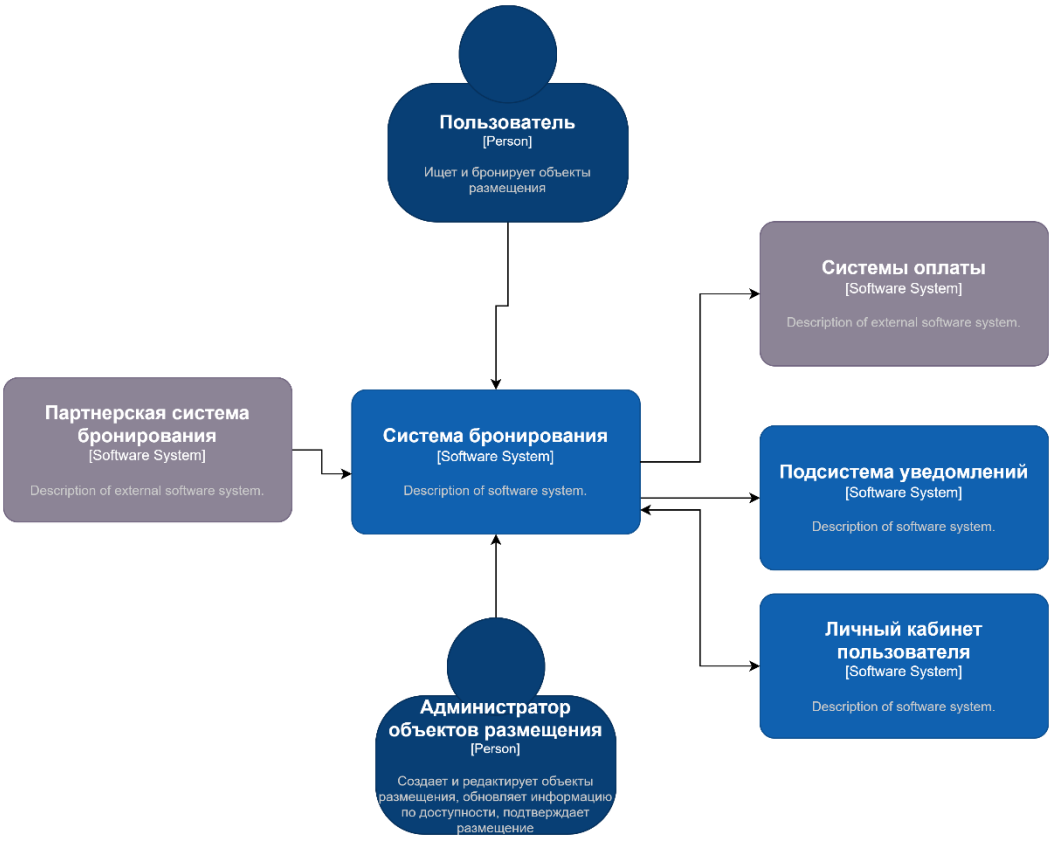
2. Архитектурное проектирование

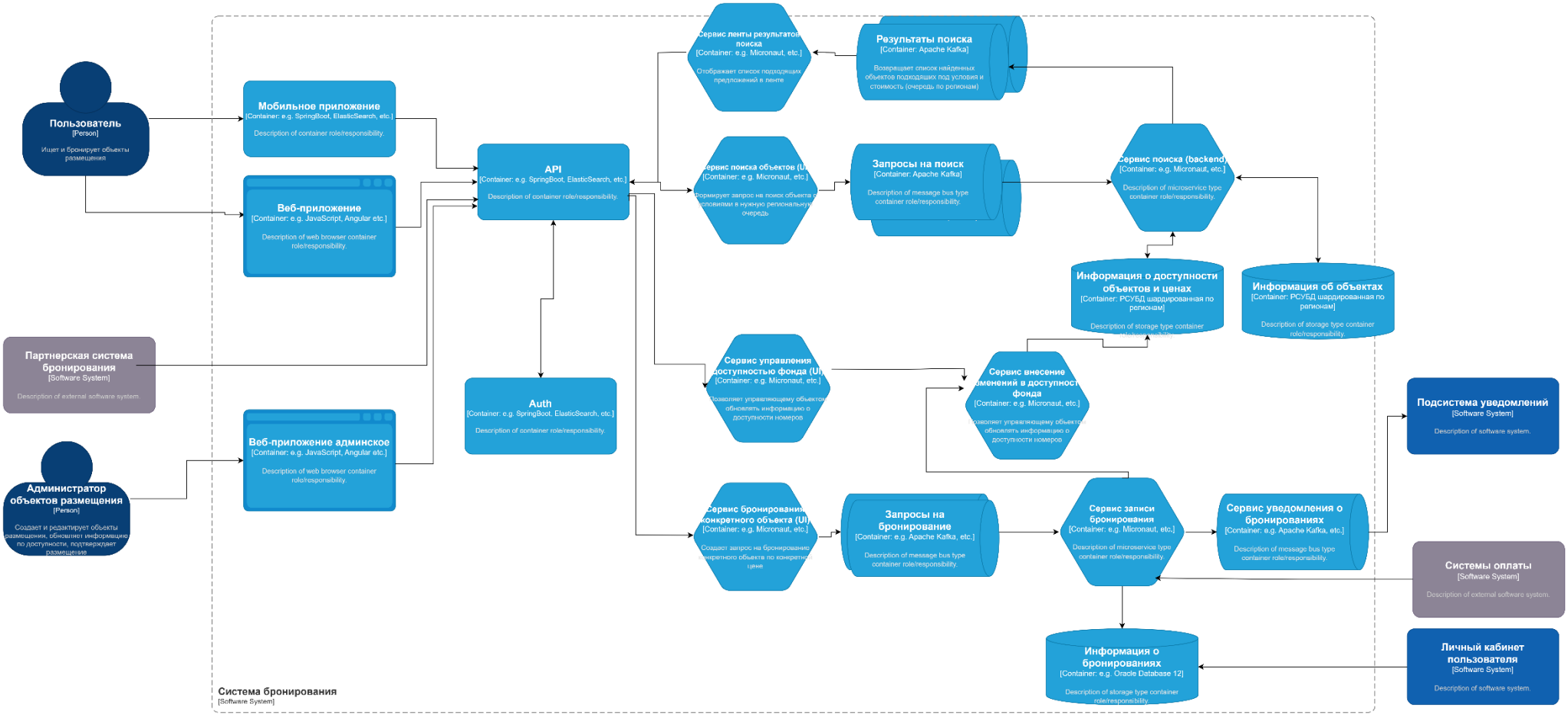
- Спроектируйте общую архитектуру системы. Укажите основные компоненты (например, базы данных, балансировщики нагрузки, API-шлюзы, очереди сообщений).
- Определите, как будут взаимодействовать компоненты системы (описание протоколов, паттернов, схемы взаимодействия).
- Предложите решения для обеспечения масштабируемости и высокой доступности (например, вертикальное/горизонтальное масштабирование, георазпределение).

Критерии оценки:

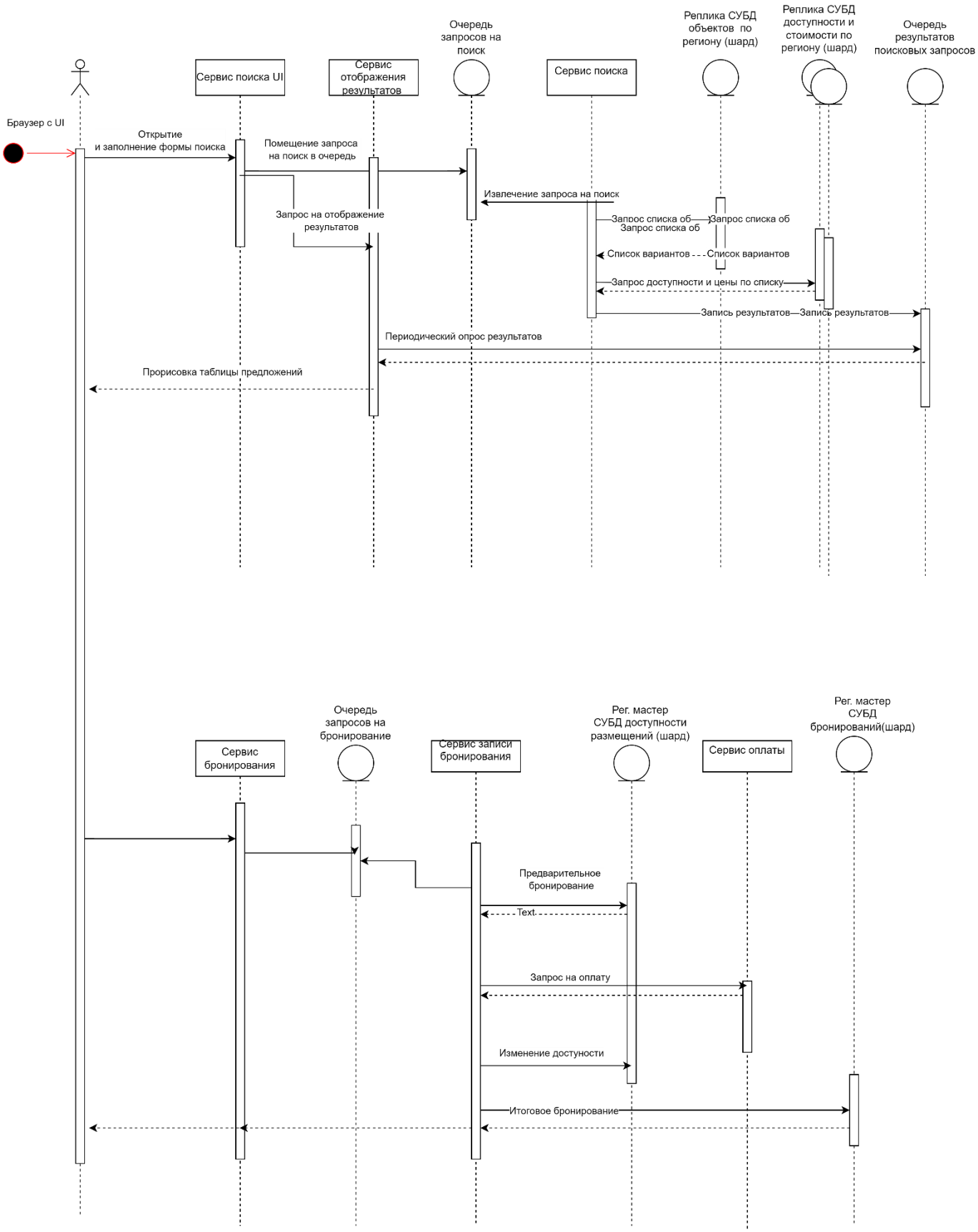
- Архитектура соответствует заданным требованиям.
  - Указаны функциональные и нефункциональные требования.
  - Предложены подходы для обеспечения масштабируемости и отказоустойчивости.

2.1. Общая архитектура системы




Пользователь
[Person]
Ищет и бронирует объекты размещения
Мобильное приложение
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Веб-приложение
[Container: e.g. JavaScript, Angular etc.]
Description of web browser container role/responsibility.
API
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Auth
[Container: e.g. SpringBoot, ElasticSearch, etc.]
Description of container role/responsibility.
Партнерская система бронирования
[Software System]
Description of external software system.
Администратор объектов размещения
[Person]
Создает и редактирует объекты размещения, обновляет информацию по доступности, подтверждает размещение
Веб-приложение администрое
[Container: e.g. JavaScript, Angular etc.]
Description of web browser container role/responsibility.
Сервис ленты результатов поиска
[Container: e.g. Micronaut, etc.]
Отображает список подходящих предложений в ленте
Результаты поиска
[Container: Apache Kafka]
Возвращает список найденных объектов подходящих под условия и стоимость (очередь по регионам)
Сервис поиска объектов (UI)
[Container: e.g. Micronaut, etc.]
Формирует запрос на поиск объекта с условиями в нужную региональную очередь
Запросы на поиск
[Container: Apache Kafka]
Description of message bus type container role/responsibility.
Сервис поиска (backend)
[Container: e.g. Micronaut, etc.]
Description of microservice type container role/responsibility.
Информация о доступности объектов и ценах
[Container: РСУБД шардированная по регионам]
Description of storage type container role/responsibility.
Информация об объектах
[Container: РСУБД шардированная по регионам]
Description of storage type container role/responsibility.
Сервис управления доступностью фонда (UI)
[Container: e.g. Micronaut, etc.]
Позволяет управляющему объектом обновлять информацию о доступности номеров
Сервис внесения изменений в доступность фонда
[Container: e.g. Micronaut, etc.]
Позволяет управляющему объектом обновлять информацию о доступности номеров
Сервис бронирования конкретного объекта (UI)
[Container: e.g. Micronaut, etc.]
Создает запрос на бронирование конкретного объекта по конкретной цене
Запросы на бронирование
[Container: e.g. Apache Kafka, etc.]
Description of message bus type container role/responsibility.
Сервис записи бронирования
[Container: e.g. Micronaut, etc.]
Description of microservice type container role/responsibility.
Сервис уведомления о бронированиях
[Container: e.g. Apache Kafka, etc.]
Description of message bus type container role/responsibility.
Информация о бронированиях
[Container: e.g. Oracle Database 12]
Description of storage type container role/responsibility.
Подсистема уведомлений
[Software System]
Description of software system.
Системы оплаты
[Software System]
Description of external software system.
Личный кабинет пользователя
[Software System]
Description of software system.
Система бронирования
[Software System]

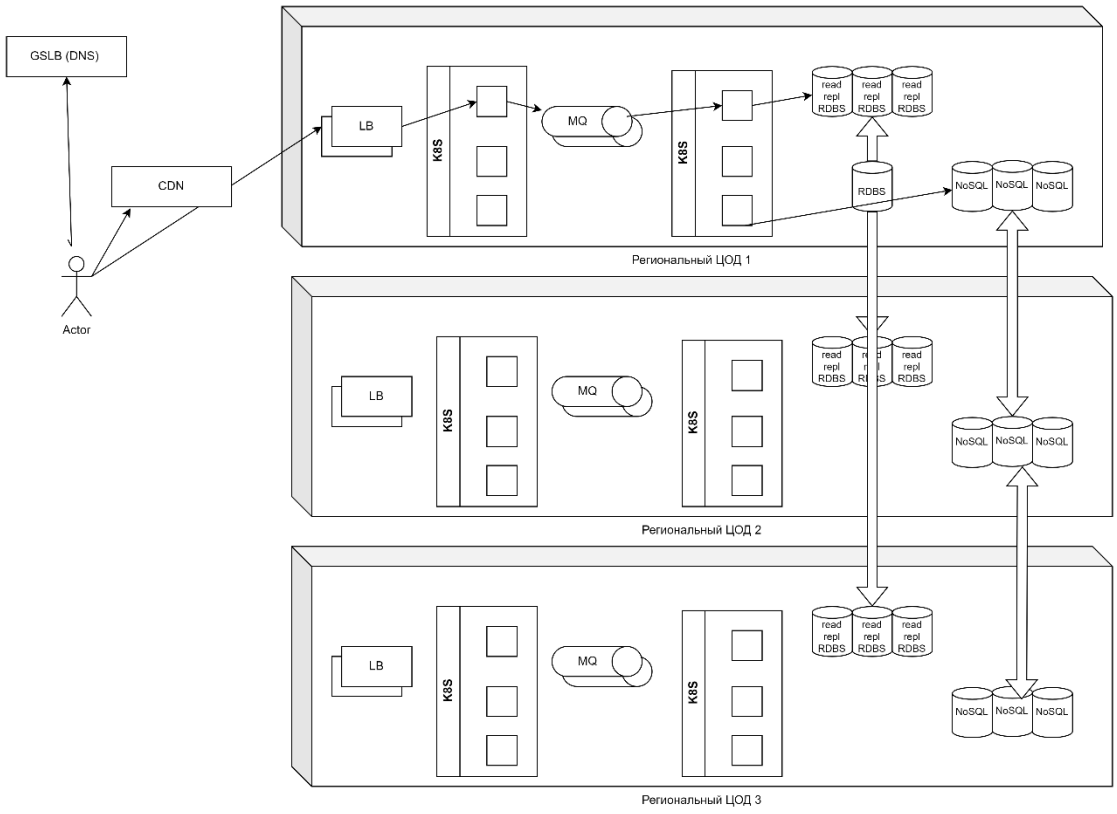
2.2. Взаимодействие компонентов системы



2.3. Масштабирование системы

Для масштабирования и обеспечения устойчивости системы планируется использовать следующие подходы:

- Использование GSLB балансировщиков для георасределения с помощью DNS для направления запроса пользователя в ближайший территориально к нему региональный ЦОД. Желательно использовать для каждого региона минимум 3 ЦОД для обеспечения отказоустойчивости
- Для кэширования статического контента будет использован сторонний CDN сервер, который имеет возможность масштабироваться
- Балансировщики (типа Nginx Ingress Controller) в рамках одного регионального ЦОД для направления запроса пользователя на экземпляр front-end сервиса.
- Использование очередей для асинхронной обработки поисковых запросов позволяет масштабировать сервиса поиска объектов размещения
- Масштабирование экземпляров сервисов с помощью scaling K8S на основании текущей утилизации и исторических пиков
- РСУБД объектов размещения масштабируется путем 1) шардирования по регионам объектов 2) масштабирование по репликам на чтение для поисковых запросов в рамках региональных групп
- В рамках РСУБД используется кэширование для сохранения результатов наиболее типовых запросов (например отель на завтра в крупных городах)
- Для хранения информации об доступности и стоимости объектов размещения на произвольные даты используется NoSQL хранилище, которое шардируется по локации объектов размещения
- Отображение поисковых результатов тоже использует ассинхронный подход



3. Выбор технологий

- Опишите, какие технологии вы бы использовали для реализации компонентов и почему (например, базы данных, брокеры сообщений, веб-серверы).
- Обоснуйте выбор (учтите производительность, стоимость, возможности масштабирования).
- Выбор технологий и подходов обоснован (например, объяснено, почему выбрана NoSQL или SQL база данных).

Планируемые к использованию технологии:

- Реляционная БД для хранения информации об объектах размещения. Реляционная СУБД обеспечивает быстрый поиск подходящих по множеству условий из справочников (тип объектов размещения, их функциональные возможности) с возможности нормализации хранимых данных по различным справочникам
    - Шардирование на уровне регионов для масштабирования по объемам данных и записи
    - Реплики БД для масштабирования сервиса поиска размещений, размещаемые в нескольких регионах для обеспечения отказоустойчивости
- NoSQL для хранения информации об доступности и стоимости объектов размещения на комбинации дат с размещением данных в оперативной памяти и шардированием по местоположению объектов обеспечивает быструю отдачу предрасчитанных знаний о доступности и стоимости объектов размещения

- Использование брокеров сообщений (типа apache kafka) для обеспечения асинхронной связи между сервисами:
    - асинхронной обработки поискового запроса и отображения результатов
    - асинхронного бронирования конкретного объекта размещения
- SOA/микросервисы в K8S для обеспечения масштабирования в рамках своего функционала

4. Реализация нефункциональных требований

- Продумайте и опишите подходы к обеспечению безопасности системы (например, использование TLS, авторизация через OAuth).
- Укажите методы мониторинга и управления системой (например, логирование, алертинг, инструменты мониторинга).

4.1. Подход к обеспечению безопасности системы

Для обеспечения безопасности системы планируется использовать комплексный подход:

- Перед системой будет стоять Anti-DDOS фильтр Curator
- Взаимодействия пользователя с интерфейсом веб-сайта будет через зашифрованный канал https/tls 1.3 подписанный публичным сертификатом
- Для первичного поиск размещений система будет принимать неавторизованные запросы от пользователей и в ответ выдавать session-id
- Слишком частые запросы к системе будут блокироваться с помощью rate limiter
- Все запросы от API к backend сервисам будут использовать JWT токены или API
- Для бронирования будет поддерживаться аутентификация пользователя через OAth

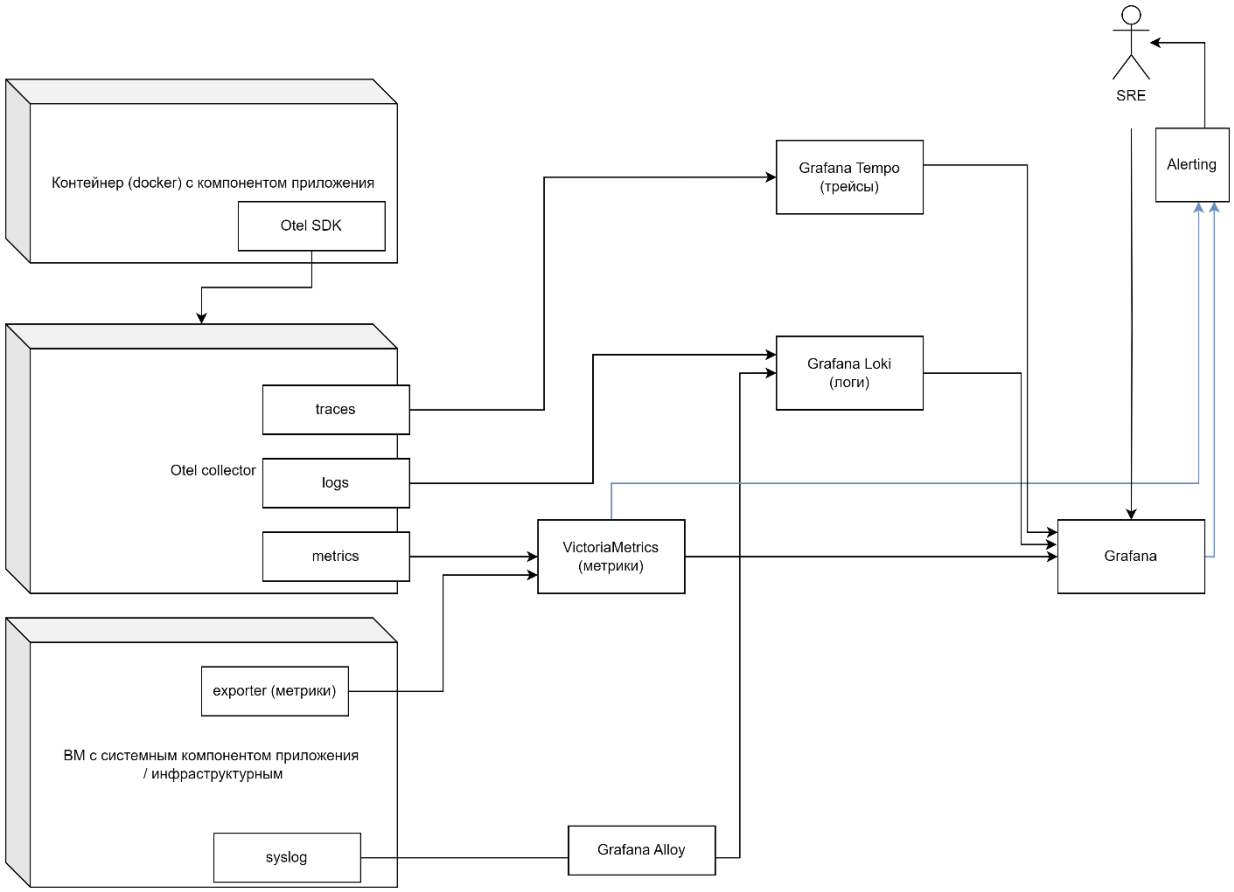
4.2. Инструменты для управления системой

Для управления запуском и масштабированием отдельных сервисов (микро-сервисов) приложения планируется использовать функционал кластеры.

4.3. Инструменты для реализации Observability

Для реализации Observability в рамках проектируемого приложения планируется использовать следующий инструментарий:

- SDK OpenTelemetry встраиваемое в прикладные экземпляры отдельных сервисов (микро-сервисов) приложения для предоставления трассировок, логов и метрик
- Otel Collector запускаемый рядом с экземплярами отдельных сервисов (микро-сервисов) приложения для сбора предоставленных экземпляром прикладного сервиса трассировок, логов и метрик и передачи их в централизованную систему мониторинга
- Отдельные exporter для предоставления метрик с инфраструктурных сервисов инфраструктурных сервисов (кластеры k8s, СУБД, proxy) и syslog
- VictoriaMetrics для сбора, обработки и хранения метрик отдельных прикладных и инфраструктурных сервисов (кластеры k8s, СУБД, proxy)
- Grafana Loki для сбора, обработки и хранения логов
- Grafana Tempo для сбора, обработки и хранения трассировок
- Grafana для визуализации метрик, логов и трассировок и для рассылки уведомлений по сбоям командам эксплуатации / SRE



5. Риски и компромиссы

- Опишите возможные риски в системе и компромиссы, которые вы приняли при проектировании (например, баланс между согласованностью и доступностью).

5.1. Риски eventual consistency результатов поиска и бронирования

В рамках бронирования пользователь выбирает вариант и нажимает забронировать, но данный вариант размещения в момент отображения может быть уже забронирован другим пользователем.

Запросов на бронирование существенно меньше (на пару порядков), чем запросов на предварительный поиск, поэтому по моему мнению компромисс оправдан.

При непосредственном бронировании также

5.2. Увеличение объемов данных

Для обеспечения масштабируемости сервисов результаты запросов передаются в ненормализованном формате, что приводит к дублированию данных в различных поисковых запросах, увеличивает объемы хранимых и передаваемых данных.

5.3. Увеличение количества вариантов объектов размещения

На скорость выполнения запросов будут влиять количество вариантов размещения, по которым надо проверять доступность. Подход, который может работать с 1000 вариантов размещения в одном городе, может не сработать при 100 000 и более вариантов размещения в одном городе.

Для ускорения запросов для хранения информации об доступности вариантов выбран подход с отдельной распределённой БД NoSQL в качестве кэша стоимости и доступности всех вариантов различных дат размещения.