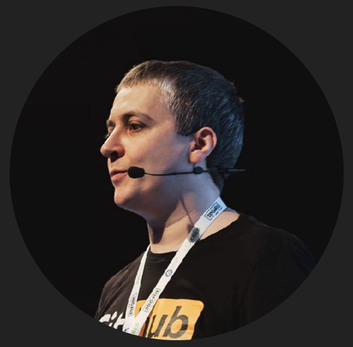


Рустам Курамшин
Дизайн микросервисной архитектуры

Александр Янчий
Дизайн API интерфейсов

Рустам Гулямов
Дизайн модели данных

Рустам Зулкарниев
Проектирование бизнес-процессов

Владислав Калинин
Проектирование инфраструктуры

КОМАНДА «JAVA BOYS»

ЗАДАЧА «СИСТЕМА ПРИЕМА ЗАКАЗОВ»

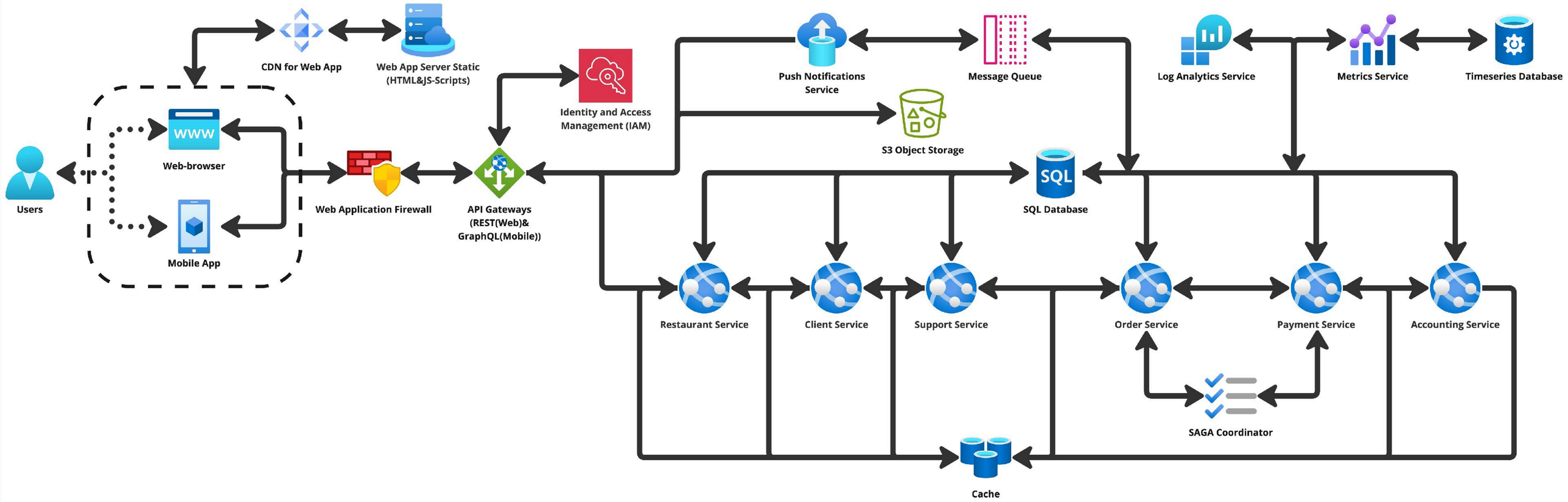
ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ (ФТ)

- Реализовать единую систему приёма заказов для сети ресторанов.
- Веб-версия и мобильное приложение на клиенте.
- Роли в системе: *гость, официант, администратор ресторана, бухгалтер, администратор системы*.
- Гость: формирует заказы (в зале или на сайте/в приложении), оплачивает заказ, просматривает статус заказа, обращается в поддержку.
- Официант: обрабатывает только свои заказы, меняет состав и статус заказов.
- Администратор ресторана: назначает столики в своём ресторане, формирует QR-коды для столиков, формирует смены официантов, назначает столики на официантов. Видит все заказы своего ресторана. Просматривает обращения в службу поддержки ресторана или всей сети (в разделе «Общие вопросы»). Дополнительно может настраивать меню своего ресторана.
- Бухгалтер: работает на всю сеть, просматривает все заказы во всех ресторанах за любой период времени, выгружает оплаченные заказы.
- Администратор системы: имеет полный доступ к системе, кроме бизнес и финансовых данных.

НЕФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ (НФТ)

- Среднее число заказов в сутки в ресторанах достигает 20 000 и в дальнейшем будет расти. В каждом ресторане может быть до 10 официантов. Планируется подключить к системе до 100 ресторанов.
- *Производим планирование в прицеле на 5 лет.*
- *В перспективе на 5 лет вперёд, мы должны учесть возможное увеличение нагрузки. Пусть, для безопасности, мы ожидаем увеличение нагрузки в 2 раза.*
- *Основные метрики: Общее количество заказов в день через 5 лет: 40 000. Пиковая нагрузка в часы пик: предположим, что 10% заказов приходится на один час, то есть 4 000 заказов в час.*

СИСТЕМА ПРИЁМА ЗАКАЗОВ



ОПИСАНИЕ КОМПОНЕНТОВ

- Web/mobile-клиент - Web/Mobile клиент предоставляет ресторанному персоналу и пользователям возможность эффективно управлять всеми аспектами работы заведения и процессом заказа. Клиент включает в себя функции управления меню, заказами, персоналом, отслеживания финансовых показателей, аналитики и отчетности. Пользователи могут просматривать меню и делать заказы.
- Web Application Firewall (WAF) - межсетевой экран для блокировки сетевых атак и DDoS.

ОПИСАНИЕ КОМПОНЕНТОВ

- API Gateway - API Gateway включает два компонента: REST (web) и GraphQL (mobile). REST обеспечивает стандартизированные RESTful API для веб-клиентов, поддерживая операции управления меню, заказами, персоналом, финансовыми показателями и аналитикой. GraphQL оптимизирован для мобильных клиентов, позволяя запрашивать только необходимые данные, что улучшает производительность и минимизирует трафик. Вместе эти компоненты создают единый вход для всех клиентов.
- Identity and Access Management (IAM) - Сервис управления идентификацией и доступом (IAM) обеспечивает аутентификацию, авторизацию и управление учетными записями пользователей. IAM гарантирует доступ к ресурсам только авторизованным пользователям, поддерживая высокий уровень безопасности. Реализован с использованием Keycloak, открытой платформы для управления доступом и идентификацией. Ролевая модель системы (роли) реализована с помощью прикрепления роли и её скоупа к JWT-токену. API Gateway после валидации JWT-токена у issuer’а определяет роль и скоуп её действия и принимает решение пропускать http-запрос или нет.

ОПИСАНИЕ КОМПОНЕНТОВ

- CDN for Web App - Сетевая инфраструктура, которая ускоряет загрузку веб-страниц, изображений, видео и других ресурсов за счет распределения их по серверам, расположенным ближе к конечным пользователям.
- Web App Server Static (HTML&JS-Scripts) - сервер, который предназначен для хранения и предоставления статических файлов, таких как HTML, CSS, JavaScript и другие ресурсы, не требующие генерации контента на сервере при каждом запросе. Используется для размещения фронтенд-части веб-приложений.

ОПИСАНИЕ КОМПОНЕНТОВ

- S3 Object Storage - Служба, позволяющая пользователям хранить и управлять любыми объемами данных в форме объектов (таких как файлы, изображения, видео и другие файлы любых форматов) в облачном хранилище.
- SQL Database - Сервер управления реляционными базами данных. Хранит данные бизнес-доменов системы в табличном представлении, реализован с использованием failover кластера PostgreSQL.

ОПИСАНИЕ КОМПОНЕНТОВ

- Cache - Служба хранения данных в быстродействующей памяти для быстрого доступа, снижения нагрузки на серверы и улучшения производительности системы. В качестве реализации используется Redis - высокопроизводительная key-value база данных.
- Restaurant Service - Сервис, отвечающий за управление информацией о ресторанах, включая конфигурацию столиков, расписание смен официантов и привязку столов к официантам.

ОПИСАНИЕ КОМПОНЕНТОВ

- Client Service - Микросервис, управляющий всей информацией, связанной с клиентами, включая регистрацию и управление профилями.
- Support Service - Обрабатывает все запросы и обращения от клиентов и персонала ресторанов, включая общие вопросы и специфические проблемы.
- Order Service - Сервис для обработки всевозможных аспектов заказов: создание, изменение, отслеживание статуса заказа. Сервис также обеспечивает функциональность для официантов по управлению заказами.

ОПИСАНИЕ КОМПОНЕНТОВ

- Payment Service - Сервис управляет процессами оплаты заказов, включая интеграцию с платежными системами и обработку транзакций. Платежный шлюз.
- SAGA Coordinator - Компонент, отвечающий за координацию транзакций между сервисом управления заказами и сервисом обработки платежей. Saga координатор управляет последовательностью операций, гарантируя атомарность распределенных транзакций между микросервисами.
- Accounting Service - Предоставляет функционал для просмотра и анализа финансовой информации по всем ресторанам для бухгалтерии и , выгрузки данных для налоговой отчетности и других бухгалтерских операций.

ОПИСАНИЕ КОМПОНЕНТОВ

- Log Analytics Service - Представляет собой инструмент для сбора, анализа и визуализации журналов и логов из различных источников в информационной системе.
- Metrics Service - веб-приложение позволяющее строить дашборды с графиками на основе данных из Analytics API. Позволяет просматривать статистику посещения веб-страниц интернет-магазина. Для визуализации и анализа данных Metrics Service использует Grafana – мощную платформу для визуализации временных рядов и метрик.

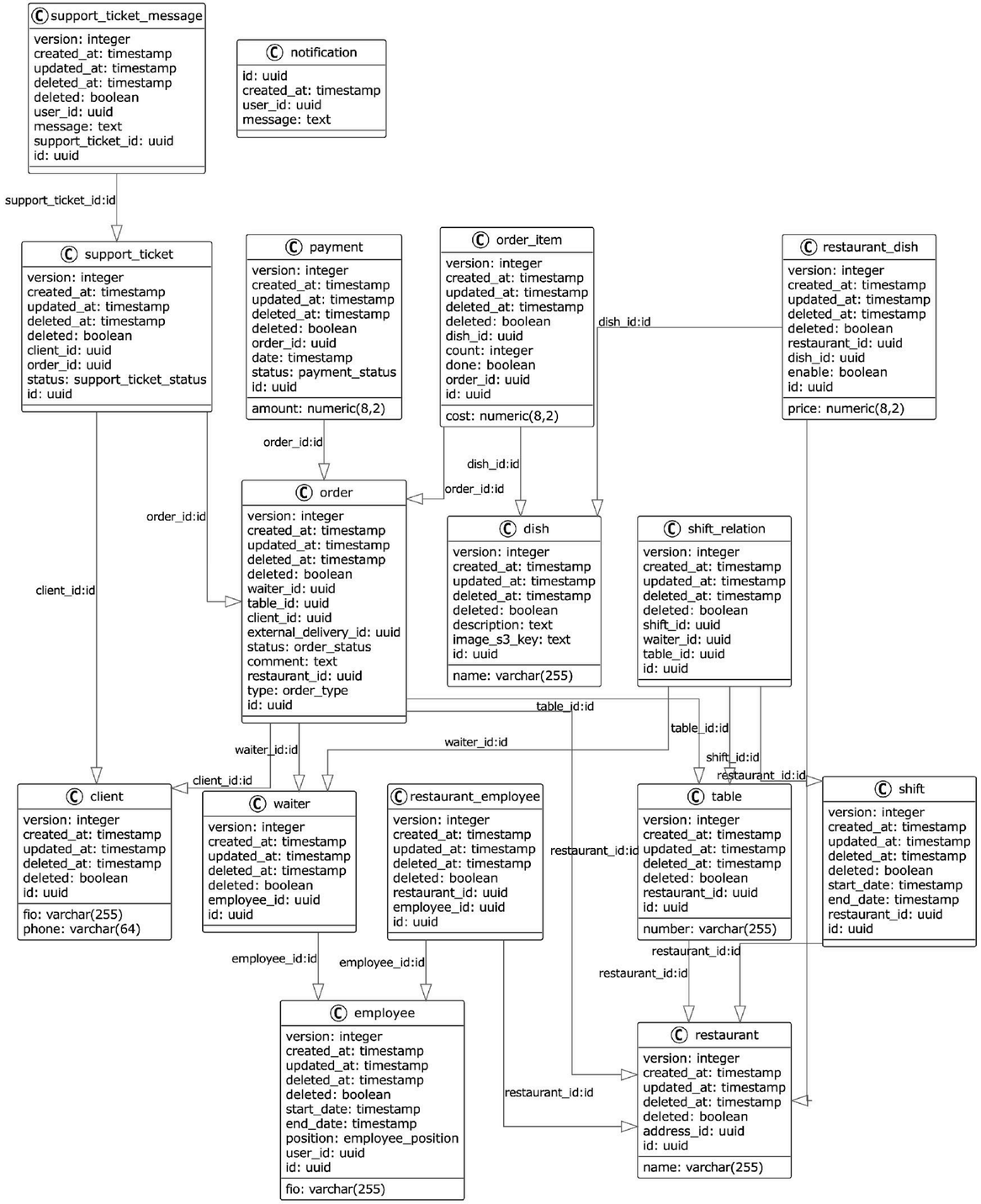
ОПИСАНИЕ КОМПОНЕНТОВ

- Timeseries Database - база данных для хранения метрик предоставляет гибкое хранилище данных, предназначенное для эффективного хранения временных рядов и числовых метрик. Примером такой базы данных является Prometheus, которая специализируется на хранении и обработке временных данных, таких как метрики производительности, системного мониторинга и IoT данных.
- Push Notification Service - это сервис, который позволяет отправлять персонализированные уведомления от сервера приложения к клиентским устройствам.

ОПИСАНИЕ КОМПОНЕНТОВ

- Message Queue - представляет собой механизм, который обеспечивает асинхронную передачу данных между различными компонентами системы. В данном сценарии сервисы-отправители помещают уведомления в темы (topics) Kafka, где они могут быть последовательно обработаны и доставлены Push Notification Service.

АРХИТЕКТУРА ДАННЫХ



ОПИСАНИЕ ОБЩИХ АТРИБУТОВ СУЩНОСТЕЙ

- id - поле первичного ключа, используем UUIDv4.
- version - поле версии сущности. Используется для реализации оптимистической блокировки из расчета, что в системе не так часто происходят гонки данных в БД.
- created_at, updated_at, deleted_at - поля с временными штампами создания, обновления и мягкого удаления сущности соответственно.
- deleted - поле-признак отметки сущности на удаление (мягкое удаление).

ОПИСАНИЕ СУЩНОСТЕЙ

- client - Хранение клиентов системы (ФИО, телефон).
- employee - Сотрудник сети ресторанов (ФИО, дата начала работы, дата окончания работы, должность(Официант, Администратор, Бухгалтер, Администратор системы), идентификатор пользователя keycloak).
- waiter - Официант, хранит информацию об официанте (идентификатор сотрудника)

ОПИСАНИЕ СУЩНОСТЕЙ

- table - Информация о столе в ресторане (идентификатор ресторана, инвентарный номер).
- restaurant - Ресторан, хранит информацию о ресторане (наименование, идентификатор адреса из справочника ГАР).
- restaurant_employee - Привязка сотрудника к ресторану (идентификатор ресторана, идентификатор сотрудника).

ОПИСАНИЕ СУЩНОСТЕЙ

- shift - Информация о смене (дата и время начала смены, дата и время окончания смены).
- shift_relation - Привязка столов к официантам в рамках смены (идентификатор смены, идентификатор официанта, идентификатор стола).
- dish - Справочник блюд на уровне всей сети ресторанов (наименование, описание, s3 ключ фотографии).
- restaurant_dish - Справочник блюд для конкретного ресторана, присутствует привязка к глобальному справочнику блюд. Таким образом каждый ресторан может выбирать из глобального справочника блюдо, которое будет у него в меню, а также выставлять цену (идентификатор из глобального справочника блюд, идентификатор ресторана, цена, доступность для клиентов).

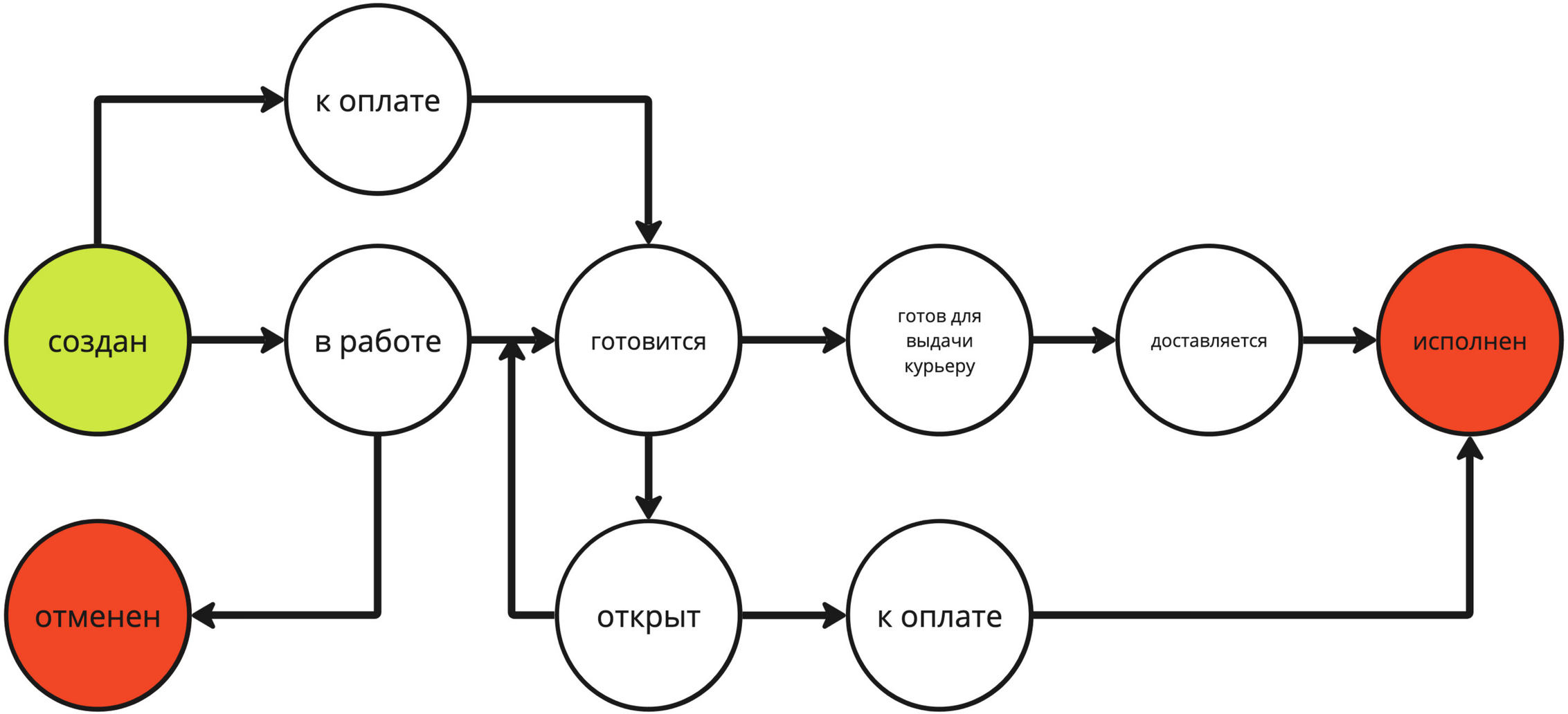
ОПИСАНИЕ СУЩНОСТЕЙ

- order - Заказ, хранит общую информацию о заказе (идентификатор ресторана, связанный официант, связанный стол, связанный клиент, идентификатор доставки, статус (создан, отменен, завершен, в работе, открыт, готов для доставки, доставляется), комментарий пожеланий клиента, тип (на месте, самовывоз, доставка)).
- order_item - Позиция заказа, хранит информацию о блюде, количеств и пр. (идентификатор заказа, идентификатор блюда, количество порций, общая стоимость, флаг готовности).
- payment - Оплата, хранит информацию об оплатах за заказы (идентификатор заказа, сумма, статус (успешно, с ошибкой)).

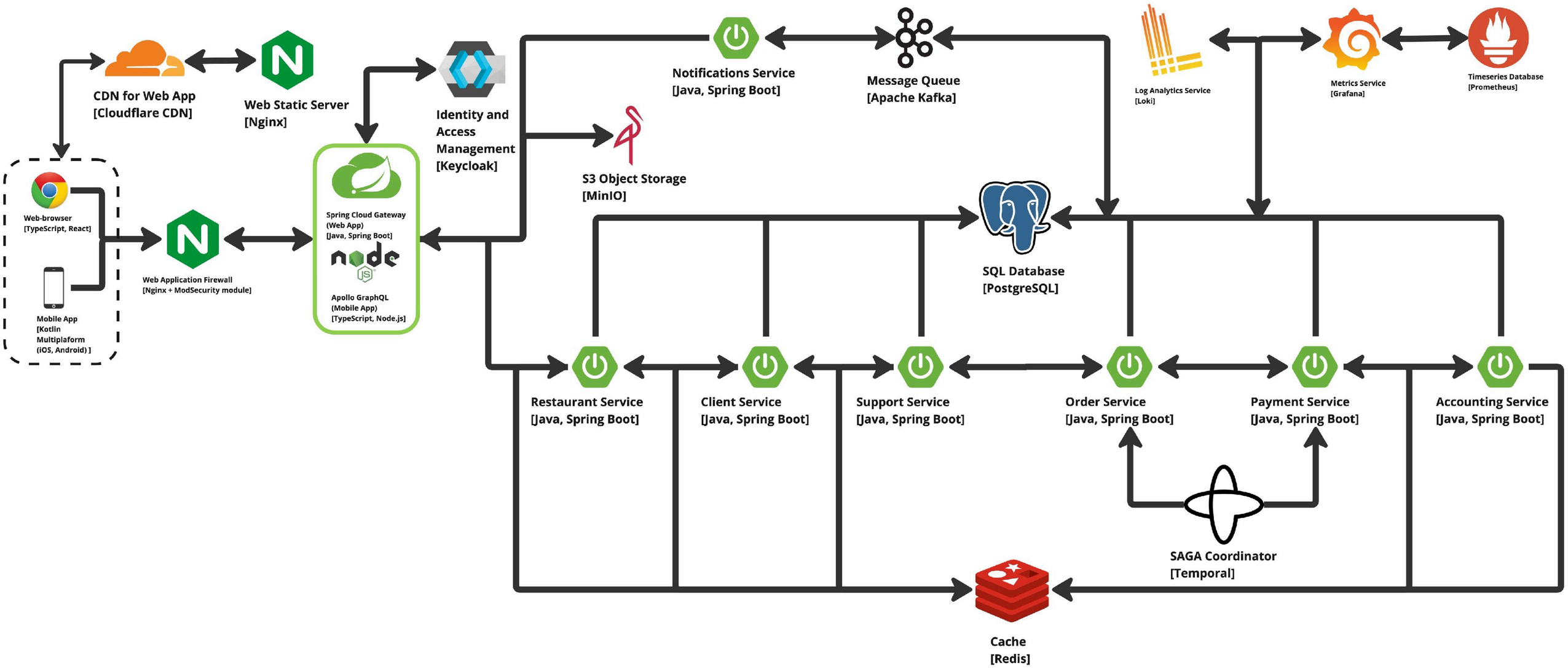
ОПИСАНИЕ СУЩНОСТЕЙ

- support_ticket - Обращение в службу поддержки, хранит информацию обо всех обращениях в системе (идентификатор клиента открывшего обращение, идентификатор заказа, статус(создан, активный, закрыт)).
- support_ticket_message - Сообщение в рамках обращения в службу поддержки (идентификатор обращения в службу поддержки, идентификатор пользователя, сообщение).
- notification - Хранение нотификаций для пользователя. (идентификатор пользователя, сообщение).

ГРАФ СОСТОЯНИЙ ЗАКАЗА



АРХИТЕКТУРА СИСТЕМЫ



KEYCLOAK

- Ролевая модель в нашем приложении основана на Keycloak и обеспечивает эффективное управление доступом через гибкую настройку ролей. Keycloak позволяет точно определять разрешения и доступ к функциональности для различных категорий пользователей, обеспечивая высокий уровень безопасности и удобное администрирование доступа
- Ролевая модель системы (роли) реализована с помощью прикрепления роли и её скоупа к JWT-токену. API Gateway после валидации JWT-токена у issuer’а определяет роль и скоуп её действия и принимает решение пропускать http-запрос или нет.

РАЗДЕЛЕНИЕ API GATEWAY

- Разделение API Gateway на два шлюза – для веб-клиента (REST) и мобильного приложения (GraphQL) – обеспечивает эффективность обработки запросов и управления данными в зависимости от требований каждого типа клиента. Этот подход позволяет оптимизировать производительность приложения, предоставляя адаптированные интерфейсы для различных устройств и способов взаимодействия с API. Важно также отметить, что такое разделение упрощает масштабирование и обеспечивает более гибкое управление системой в целом.

ПАТТЕРН SAGA И TEMPORAL

- Saga, отвечающий за координацию транзакций между сервисом управления заказами и сервисом обработки платежей, обеспечивает атомарность операций при выполнении сложных бизнес-процессов. Когда инициируется распределенная транзакция, Saga запускает последовательность шагов, каждый из которых представляет собой операцию в одном из сервисов. Этот подход гарантирует, что все этапы транзакции будут либо успешно выполнены, либо корректно отменены в случае ошибки, обеспечивая надёжность и целостность выполнения бизнес-логики в распределенной среде.
- Saga реализована с помощью Temporal.

MESSAGE QUEUE

- Apache Kafka – это высокопроизводительная распределённая система сообщений, которая обеспечивает надёжный обмен данными в реальном времени. В нашем сценарии сервисы-отправители используют топики Kafka для размещения уведомлений, которые последовательно обрабатываются и доставляются сервису Push Notification Service. Kafka гарантирует эффективную передачу сообщений с минимальной задержкой благодаря своей масштабируемости и механизмам репликации данных.

ИСПОЛЬЗОВАНИЕ CDN

- CDN (Content Delivery Network) – это сетевая инфраструктура, оптимизирующая доставку веб-страниц, изображений, видео и других ресурсов путем их распределения по серверам, размещенным географически ближе к конечным пользователям. Это позволяет значительно сократить время загрузки контента благодаря уменьшению задержек и лучшей отзывчивости веб-приложений. CDN улучшает производительность веб-сайтов и приложений, предоставляя быстрый доступ к контенту независимо от местоположения пользователя.

RESTAURANT SERVICE API

- Добавление ресторана
  POST /restaurants
  { "name": "Claude Monet", "addressId": "0d12a95a-3c0e-4ef0-bb5d-202e8562cd2a" }
- Получение информации ресторане
  GET /restaurants/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Изменение ресторана
  PUT /restaurants/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  { "name": "New Claude Monet", "addressId": "0d12a95a-3c0e-4ef0-bb5d-202e8562cd2a" }
- Удаление ресторана
  DELETE /restaurants/187b3000-b8c1-46c7-b0b9-76d484b40ca5

RESTAURANT SERVICE API

- Добавление столика
  POST /tables
  { "restaurantId": "187b3000-b8c1-46c7-b0b9-76d484b40ca5", "number": "127" }
- Получение информации о столике, в том числе сгенерированный QR-код
  GET /tables/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Изменение столика
  PUT /tables/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  { "restaurantId": "187b3000-b8c1-46c7-b0b9-76d484b40ca5", "number": "148" }
- Удаление столика
  DELETE /tables/187b3000-b8c1-46c7-b0b9-76d484b40ca5

RESTAURANT SERVICE API

- Добавление смены
  POST /shifts
  { "startDate": "2024-06-29T08:00:00", "endDate": "2024-06-29T22:00:00" }
- Получение информации о смене
  GET /shifts/e871838f-7b8f-4ae6-9519-733ec1cebe0c
- Добавление официанта а смену
  POST /shifts/e871838f-7b8f-4ae6-9519-733ec1cebe0c/relation
  { "waiterId": "187b3000-b8c1-46c7-b0b9-76d484b40ca5", "tableId": "cab127cd-c68a-4e83-b0e6-34e52367114a" }
- Удаление официанта со смены
  DELETE /shifts/e871838f-7b8f-4ae6-9519-733ec1cebe0c/relation/e05be066-8bd9-40c7-a36f-deec5fe93fad

RESTAURANT SERVICE API

- Добавление блюда
  POST /dishes
  { "name": "Шаурма королевская", "description": "Пальчики оближешь", "image_s3_key": "/images/good-food.jpg" }
- Получение информации о блюде
  GET /dishes/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Изменение блюда
  PUT /dishes/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  { "name": "Шаурма королевская XXL", "description": "Пальчики оближешь", "image_s3_key": "/images/good-food.jpg" }
- Удаление блюда
  DELETE /dishes/187b3000-b8c1-46c7-b0b9-76d484b40ca5

RESTAURANT SERVICE API

- Добавление блюда
  POST /restaurant-dishes
  { "dishId": "b1b6c8f0-cdf3-4d9e-877c-161238f988ff", "price": 250 }
- Получение информации о блюде
  GET /restaurant-dishes/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Исключение блюда из меню ресторана
  PATCH /restaurant-dishes/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  { "enable": false }
- Удаление блюда
  DELETE /restaurant-dishes/187b3000-b8c1-46c7-b0b9-76d484b40ca5

CLIENT SERVICE API

- Регистрация клиента
  POST /clients
  { "fio": "Василий Пиццерной", "phone": "81234567890", "email":
  "pizzaman@mail.ru" }

- Получение списка клиентов
  GET /clients

- Изменение клиента
  PUT /clients/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  { "fio": "Петр Пиццерной", "phone": "81234567811", "email": "pizzaman@mail.ru" }

SUPPORT SERVICE API

- Создание обращения
  POST /support-tickets
  { "restaurant_id": "187b3111-b8c2-46c7-b069-76d484b40ca5", "message": "Всем привет, шаурма просто огонь."}
- Получение информации об обращении
  GET /support-tickets/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Добавление сообщения в обращение
  POST /support-tickets/187b3000-b8c1-46c7-b0b9-76d484b40ca5/messages
  { "message": "Всем привет, шаурма просто огонь." }

ORDER SERVICE API

- Создание заказа
  POST /orders/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  { "type": "EAT_IN", "comment": "Добавить больше перца", "items":
  [{ "dishId": "77db1f5b-5534-42ca-9608-7a5716cdc6d9", "count": 2, "cost":
  360 }] }
- Получение информации о заказе
  GET /orders/187b3000-b8c1-46c7-b0b9-76d484b40ca5

ORDER SERVICE API

- Добавление позиции в заказ
  POST /orders/187b3000-b8c1-46c7-b0b9-76d484b40ca5/items
  { "dishId": "21b17c9a-88d8-41ab-89d9-8a6bf8ec16c1", "count": 1, "cost": 120 }

- Удаление позиции из заказа
  DELETE /orders/187b3000-b8c1-46c7-b0b9-76d484b40ca5/items/21b17c9a-88d8-41ab-89d9-8a6bf8ec16c1

PAYMENT SERVICE API

- Обработка платежей за заказы
  POST /payments
  { "order_id": "12345","amount": 100.50,"payment_method": "credit_card", "card_info": {"card_number": "4111111111111111","card_expiry": "12/24","card_cvv": "123"}}
- Получение информации о платеже
  GET /payments/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Обновление информации о платеже
  PUT /payments/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  {"status": "confirmed"}
- Отмена платежа
  DELETE /payments/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Список всех платежей для заказа
  GET /payments?order_id=187b3000-b8c1-46c7-b0b9-76d484b40ca5

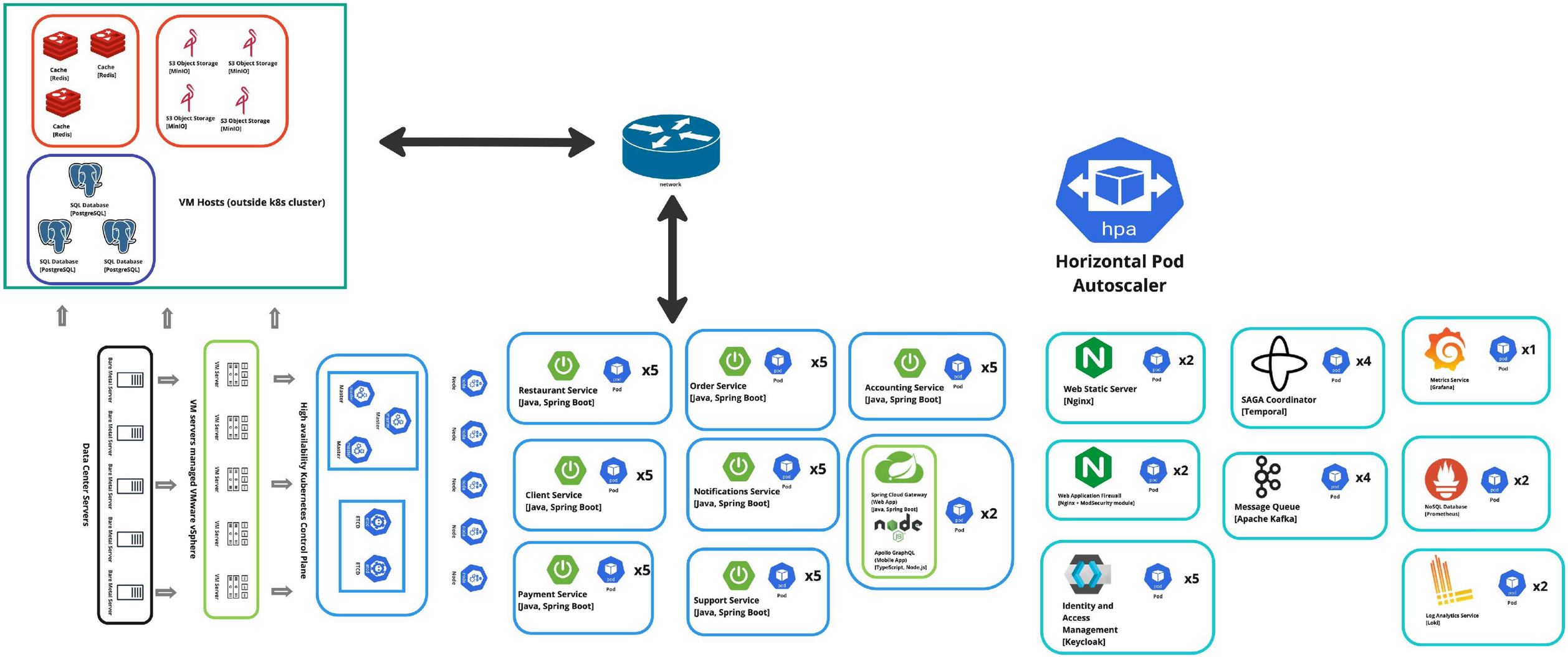
ACCOUNTING SERVICE API

- Получение списка транзакций за определенный период
  GET /transactions
- Получение детальной информации о транзакции
  GET /transactions/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Добавление новой транзакции
  POST /transactions
  {"date": "2024-06-28", "amount": 200.00, "type": "income", "restaurant_id": "21b17c9a-88d8-41ab-89d9-8a6bf8ec16c1"}
- Обновление информации о транзакции
  PUT /transactions/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  {"amount": 250.00, "type": "income"}
- Удаление транзакции
  DELETE /transactions/187b3000-b8c1-46c7-b0b9-76d484b40ca5

NOTIFICATIONS SERVICE API

- Отправка уведомления пользователю
  POST /notifications
  {"user_id": "123e4567-e89b-12d3-a456-426614174000", "message": "Your order has been processed!", "channel": "email"}
- Получение списка уведомлений для пользователя
  GET /notifications/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Обновление уведомления
  PUT /notifications/187b3000-b8c1-46c7-b0b9-76d484b40ca5
  {"status": "read"}
- Удаление уведомления
  DELETE /notifications/187b3000-b8c1-46c7-b0b9-76d484b40ca5
- Получение деталей уведомления
  GET /notifications/187b3000-b8c1-46c7-b0b9-76d484b40ca5/details

АРХИТЕКТУРА РАЗВЕРТЫВАНИЯ



РАЗМЕР И ХАРАКТЕРИСТИКИ КЛАСТЕРА KUBERNETES И ВИРТУАЛЬНЫХ ХОСТОВ

- Расчёт ресурсов:
- Микросервисы: Java и Spring Boot могут быть довольно требовательны к памяти. Предположим, что средний микросервис потребляет около 512 МБ RAM и 1 vCPU.
- Базы данных и хранилища: PostgreSQL: Для высокой доступности и производительности рекомендуется развернуть кластер из минимум 3 нод. Каждая нода может потреблять до 4 vCPU и 8 ГБ RAM.
- Redis: Высокопроизводительное кэширование, предположим 2 vCPU и 4 ГБ RAM на ноду.
- MinIO: Для обеспечения отказоустойчивого хранения, начнём с 4 vCPU и 8 ГБ RAM.
- Kafka: Минимум 3 ноды для управления потоками данных, каждая с 4 vCPU и 8 ГБ RAM.
- IAM (кластерный режим Keycloak), Temporal, Nginx + WAF, API Gateways: Различные нагрузки, начиная с 2 vCPU и 4 ГБ RAM на сервис.
- Расчёт нод Kubernetes: Предположим, что каждая нода может эффективно управлять 16 vCPU и 32 ГБ RAM.
- Количество микросервисов и других компонентов требует, чтобы общая сумма ресурсов в кластере была достаточной для обработки пиковой нагрузки и обеспечения отказоустойчивости.

РАЗМЕР И ХАРАКТЕРИСТИКИ КЛАСТЕРА KUBERNETES И ВИРТУАЛЬНЫХ ХОСТОВ

- Расчет количества нод. Пусть у нас будет 20 микросервисов (с учетом управления, мониторинга и логирования).
- Микросервисы: 20 * (1 vCPU и 0.5 ГБ RAM) = 20 vCPU и 10 ГБ RAM.
- PostgreSQL: 3 * (4 vCPU и 8 ГБ RAM) = 12 vCPU и 24 ГБ RAM.
- Redis: 3 * (2 vCPU и 4 ГБ RAM) = 6 vCPU и 12 ГБ RAM.
- MinIO: 4 vCPU и 8 ГБ RAM.
- Kafka: 3 * (4 vCPU и 8 ГБ RAM) = 12 vCPU и 24 ГБ RAM.
- Прочие сервисы (IAM (кластер Keycloak), Temporal, Gateways, Nginx): ~10 vCPU и 20 ГБ RAM.
- Итого: ~64 vCPU и ~98 ГБ RAM.
- Выбор и настройка нод. Для удобства управления и масштабирования, а также для обеспечения отказоустойчивости: Предполагаемое количество нод: 64 vCPU / 16 vCPU per node = 4 ноды, округлим до 5 нод для резерва. Также с заделом на будущее можно удвоить кол-во нод в кластере. Каждая нода: 16 vCPU и ~32 ГБ RAM.

KUBERNETES

- Высокодоступный кластер kubernetes с репликацией Master и ETCD в Control Plain’e.
- Мониторинг и логирование: будем использовать Prometheus для мониторинга метрик с кластера и Loki для агрегации логов, что позволит эффективно наблюдать за состоянием кластера и оперативно реагировать на проблемы.
- Масштабируемость: kubernetes предоставляет средства для автоматического горизонтального масштабирования сервисов в ответ на изменения нагрузки (HPA, Horizontal Pod Autoscaling).
- Управление конфигурациями и секретами: будем использовать ConfigMaps и Secrets для управления конфигурационными данными и чувствительной информацией без необходимости внесения изменений в образы контейнеров.

POSTGRESQL

- Высокодоступный кластер PostgreSQL. Использование кластеризации с репликацией для обеспечения непрерывности бизнес-процессов даже при сбоях отдельных узлов.
- Режимы нод в Read/Write и ReadOnly. Репликация между нодами с помощью Oracle GoldenGate.
- Резервное копирование и восстановление: Настройка регулярных резервных копий и стратегий восстановления данных критически важна для защиты данных.
- Масштабирование: Планирование вертикального или горизонтального масштабирования в зависимости от роста объемов данных и нагрузки.

REDIS

- Мастер-слейв репликация: Настройка репликации для повышения производительности чтения и отказоустойчивости.
- Персистентность данных: Конфигурация персистентности для предотвращения потери данных при перезапуске узлов.
- Кэширование сессий: Использование Redis для кэширования сессий и часто запрашиваемых данных снижает нагрузку на основную базу данных.

MINIO

- Хранение объектов: MinIO обеспечивает распределенное хранение объектов с высокой доступностью и защитой данных.
- Масштабирование: Масштабируемость MinIO позволяет легко увеличивать хранилище по мере роста потребностей в данных.
- Интеграция: Легкая интеграция с другими сервисами через S3-совместимый API.

APACHE KAFKA

- Управление потоками данных: Kafka служит для надежной передачи уведомлений от сервисов к сервису нотификации.
- Высокая пропускная способность: Настройка кластера для обеспечения обработки больших объемов сообщений с минимальной задержкой.
- Долговременное хранение: Конфигурация удержания сообщений позволяет выполнять анализ данных и восстановление после сбоев.

API GATEWAY

- Балансировка нагрузки: Распределение входящего трафика между микросервисами для оптимизации загрузки и времени ответа.
- Управление API: Управление версиями API, авторизация доступа, ограничение частоты запросов для защиты сервисов.
- Агрегация сервисов: Объединение данных из нескольких микросервисов в единые ответы API.

KEYCLOAK

- Высокодоступный кластерный режим развертывания Keycloak.
- Желательно использование внешнего кэша Infinispan в кластерном режиме.
- Управление идентификацией и доступом: Централизованное управление пользователями, сессиями и политиками безопасности.
- Интеграция: Легкая интеграция с API Gateway через стандарты OAuth2 и OpenID Connect.
- Использование JWT-токенов для передачи информации о ролях и их скоупах на API Gateway. За счет этого реализовано ролевое управление доступом.

NGINX + MODSECURITY MODULE

- Безопасность: Защита веб-приложений от известных угроз и атак с помощью правил WAF.
- Оптимизация производительности: Кэширование статического контента и оптимизация загрузки веб-страниц.
- SSL/TLS: Управление зашифрованным трафиком для обеспечения безопасности данных.


JAVA BOYS
JAVA
1 0 1 1 1 0 1 1 0 1