Техническое задание на разработку партнёрской программы

Пояснительная записка

Партнёрская программа представляет собой высоконагруженный сервис (далее – «сервис»), связывающий веб-мастеров (партнёров, издателей или рефереров) с рекламодателем. Сервис должен обеспечивать стабильную работу при пиковой нагрузке до 100 000 запросов в секунду.

Структура сервиса должна включать:

- модуль обработки трафика (TDS);
- интеграция с основным сайтом рекламодателя;
- личный кабинет партнёра;
- личный кабинет рекламодателя с разграничением ролей (администратор и маркетолог).

UI/UX-дизайн должен быть выполнен в соответствии с брендбуком компании.

1. Требования к безопасности

1.1. Защита данных пользователей

- **Шифрование данных:**
  - Обеспечить защиту передаваемых данных с использованием современных протоколов шифрования (например, TLS/SSL).
  - Обеспечить хранение конфиденциальной информации (пароли, персональные данные) в зашифрованном виде с использованием проверенных алгоритмов (например, bcrypt или Argon2 для паролей).
- **Конфиденциальность и целостность:**
  - Гарантировать, что доступ к данным осуществляется только авторизованными пользователями и сервисами.
  - Реализовать механизмы проверки целостности данных при передаче и хранении.

1.2. Аутентификация и авторизация

- **Многофакторная аутентификация (MFA):**
  - Внедрить MFA для доступа к личным кабинетам и административным функциям, чтобы повысить уровень защиты учетных записей.
- **Разграничение прав доступа:**
  - Обеспечить разделение ролей для различных категорий пользователей (веб-мастера, рекламодатель: администратор и маркетолог).
  - Применять принцип наименьших привилегий, предоставляя пользователям доступ только к необходимым функциям.

- **Управление сессиями:**
    - Использовать современные методы управления сессиями, такие как токены (например, JWT), для проверки подлинности запросов.
    - Реализовать автоматический выход из системы после определённого периода бездействия.

1.3. Защита от атак

- **Предотвращение инъекционных атак:**
    - Обеспечить защиту от SQL-инъекций, XSS (межсайтовый скриптинг), CSRF (подделка межсайтовых запросов) и других распространённых атак с использованием проверенных библиотек и фреймворков.
- **Rate limiting и ограничение запросов:**
    - Реализовать механизмы ограничения количества запросов от одного источника для защиты от атак типа brute force и других видов злоупотреблений.
- **Мониторинг и обнаружение аномалий:**
    - Внедрить систему мониторинга для отслеживания подозрительной активности, позволяющую своевременно обнаруживать и реагировать на попытки несанкционированного доступа.

1.4. Защита от DDoS-атак

- **Системы обнаружения и смягчения DDoS-атак:**
    - Интегрировать специализированные сервисы и программные решения для защиты от распределённых атак.

- Рассмотреть возможность использования облачных сервисов, способных автоматически масштабировать ресурсы в случае пиковых нагрузок.

- **Резервирование и отказоустойчивость:**
  - Обеспечить резервирование критически важных компонентов системы, чтобы минимизировать влияние атак на работоспособность сервиса.

1.5. Логирование и аудит безопасности

- **Детальное логирование:**
  - Вести журналирование действий пользователей и системных событий, что позволит отслеживать все ключевые операции.

- **Анализ и аудит:**
  - Регулярно анализировать логи для выявления аномальных или подозрительных действий.
  - Организовать систему оповещений для немедленного реагирования на инциденты безопасности.

1.6. Обновление и патчинг

- **Регулярное обновление ПО:**
  - Обеспечить своевременное обновление используемых библиотек, фреймворков и операционных систем.
  - Внедрить процесс быстрого реагирования на выявленные уязвимости и выпуск патчей.

- **Аудит безопасности:**
  - Проводить периодический аудит системы и тестирование на проникновение (penetration testing) для оценки устойчивости к атакам.

1.7. Регламент работы с инцидентами безопасности

- **План реагирования на инциденты:**
  - Разработать и задокументировать процедуру реагирования на инциденты, включающую этапы обнаружения, анализа, устранения и восстановления после атак.
- **Коммуникация и уведомления:**
  - Определить ответственных лиц за обработку инцидентов.
  - Установить порядок уведомления пользователей и заинтересованных сторон в случае утечки данных или других серьезных нарушений безопасности.

2. Производительность и масштабируемость

2.1. Отказоустойчивость и резервирование

- **Цель отказоустойчивости:**

  Обеспечить непрерывную работу сервиса даже при возникновении сбоев отдельных компонентов или целых кластеров, минимизируя время простоя и предотвращая потерю данных.

- **Архитектурные решения:**
  - **Кластеризация и балансировка нагрузки:**

    Реализовать кластерное развертывание ключевых компонентов сервиса с использованием балансировщиков нагрузки, что позволит распределять запросы между несколькими серверами и избегать перегрузки отдельных узлов.

  - **Резервирование компонентов:**
    - **Активное/пасивное резервирование:**

      настроить систему таким образом, чтобы в случае отказа активного узла автоматически подключался резервный.

    - **Резервирование данных:**

      Обеспечить регулярное создание резервных копий (бэкапов) базы данных и других критически важных данных. Бэкапы должны храниться на отдельных серверах или в облачных хранилищах с высокой доступностью.

  - **Мониторинг и автоматическое восстановление:**
    - Внедрить систему мониторинга состояния всех компонентов сервиса для оперативного выявления сбоев.

        - Настроить автоматизированные процедуры перезапуска или переключения на резервные узлы при обнаружении проблем.

- **Планирование отказов:**
    - Определить критические точки отказа и разработать сценарии аварийного восстановления.
    - Провести регулярное тестирование сценариев отказа (failover testing), чтобы убедиться в работоспособности резервных механизмов.

2.2. Механизмы масштабирования

- **Горизонтальное масштабирование:**
    - **Добавление узлов:**

      При увеличении нагрузки предусмотреть возможность добавления дополнительных серверов или виртуальных машин для распределения входящего трафика.

    - **Распределённая архитектура:**

      Разработать систему так, чтобы её компоненты могли работать в распределённой среде, что позволяет масштабировать отдельные модули (например, модуль обработки трафика, личные кабинеты) независимо друг от друга.

    - **Автоматическое масштабирование:**

      Реализовать или интегрировать решения для автоматического масштабирования (auto-scaling) в облачной инфраструктуре, которые будут динамически увеличивать или уменьшать количество активных серверов в зависимости от текущей нагрузки.

- **Балансировщики нагрузки:**

  Использовать балансировщики нагрузки для равномерного распределения запросов между всеми доступными серверами. Это обеспечивает как отказоустойчивость, так и возможность горизонтального масштабирования.

- **Вертикальное масштабирование:**
  - **Увеличение ресурсов серверов:**

    При необходимости можно повысить вычислительную мощность отдельных серверов за счёт увеличения объёма оперативной памяти, количества ядер процессора или объёма дискового пространства.

  - **Ограничения вертикального масштабирования:** При вертикальном масштабировании необходимо учитывать, что аппаратные ограничения могут стать узким местом, а также возможны перебои при масштабном увеличении ресурсов.

  - **Совмещение с горизонтальным масштабированием:**

    В идеале архитектура должна предусматривать возможность использования как горизонтального, так и вертикального масштабирования, позволяя гибко реагировать на изменения в нагрузке.

2.3. Дополнительные требования по производительности

- **Пиковая нагрузка:**
  Сервис должен выдерживать пиковую нагрузку до 100 000 запросов в секунду без существенного ухудшения времени отклика.

- **Кэширование:**
  Для оптимизации производительности использовать

механизмы кэширования (например, Redis, Memcached) для снижения нагрузки на базу данных и ускорения обработки запросов.

- **Тестирование производительности:**

  Перед вводом сервиса в эксплуатацию провести нагрузочное тестирование (stress testing, load testing), чтобы убедиться, что выбранные архитектурные решения удовлетворяют требованиям по масштабируемости и отказоустойчивости.

Дополнительно

По упрощённой оценке, для обработки 100 000 запросов в секунду может потребоваться от 20 до 50 серверов только для веб-слоя и не учитывает дополнительные компоненты, такие как:

- Серверы для работы с базой данных (возможно, кластер баз данных или отдельные реплики).
- Серверы для кэширования.
- Балансировщики нагрузки.
- Дополнительные сервисы (например, системы мониторинга, логирования и т.д.).

3. Мониторинг и логирование

3.1. Мониторинг работоспособности сервиса

- **Цель мониторинга:**
  Обеспечить непрерывный контроль за состоянием и производительностью сервиса, своевременно обнаруживать отклонения от нормальной работы и оперативно реагировать на возможные проблемы.
- **Основные метрики для мониторинга:**
  - **Системные ресурсы:** загрузка CPU, использование оперативной памяти, диск и сеть.
  - **Показатели производительности:** время отклика, количество обрабатываемых запросов, ошибки выполнения (например, 4xx/5xx коды HTTP).
  - **Доступность сервиса:** процент времени безотказной работы (uptime), время восстановления после сбоев.
  - **Логирование бизнес-событий:** количество регистраций, транзакций, успешных/неудачных операций в личных кабинетах и прочих модулях сервиса.
- **Инструменты мониторинга:**
  - **Системы сбора и визуализации метрик:**
    Prometheus, Grafana, Zabbix или аналогичные решения, позволяющие строить графики, задавать пороговые значения и настраивать автоматические оповещения.
  - **Алертинг:**
    - Настроить систему уведомлений (например, email, SMS, мессенджеры) для информирования ответственных лиц о критических сбоях или

достижении пороговых значений по ключевым метрикам.

        - Определить перечень событий, при которых срабатывают алерты (например, падение доступности, резкое увеличение времени отклика, рост ошибок запросов).

- **Процесс мониторинга:**
    - **Сбор метрик:** агрегация данных с различных серверов и компонентов сервиса в единую систему мониторинга.
    - **Анализ и визуализация:** регулярное представление метрик в виде графиков и дашбордов, что позволяет оперативно выявлять отклонения от нормы.
    - **Отчетность:** формирование периодических отчетов о состоянии сервиса для анализа трендов и выявления потенциальных проблем.

3.2. Логирование событий

- **Объекты логирования:**
    - **Системные события:**
        - Запуск и остановка сервисов, изменения в конфигурации, обновления ПО.
        - Системные ошибки, исключения, аварийные ситуации.
    - **Пользовательские действия:**
        - Аутентификация (успешные и неудачные попытки входа).
        - Действия пользователей в личных кабинетах (регистрация, изменение настроек, транзакции).

    - **События безопасности:**
        - Попытки несанкционированного доступа, подозрительная активность, события, связанные с нарушением политик безопасности.
    - **Запросы к API и веб-сервисам:**
        - Запросы, их параметры, время обработки, коды ответа сервера, IP-адреса клиентов.
- **Стандарты логирования:**
    - **Формат логов:** использовать структурированные форматы (например, JSON) для удобства последующего анализа.
    - **Уровни логирования:** определение уровней (DEBUG, INFO, WARNING, ERROR, CRITICAL) для классификации сообщений по важности.
    - **Ротация и хранение логов:**
        - Настроить периодическую ротацию логов, чтобы избежать переполнения дискового пространства.
        - Определить период хранения логов и процедуры архивирования/удаления старых записей.
- **Централизованное логирование:**
    - Реализовать сбор логов со всех компонентов сервиса в централизованное хранилище (например, с использованием ELK-стека: Elasticsearch, Logstash, Kibana, или аналогичных решений), что позволит проводить комплексный анализ событий и быстро находить корреляции между инцидентами.

3.3. Анализ логов

- **Автоматизированный анализ:**

    - Внедрить инструменты для автоматического анализа логов с целью выявления аномальных или подозрительных событий.
    - Настроить автоматические алерты на основе определённых шаблонов или пороговых значений (например, превышение количества ошибок за короткий промежуток времени).
- **Регулярный аудит логов:**
    - Организовать регулярные проверки логов для выявления скрытых проблем в работе сервиса и раннего обнаружения инцидентов безопасности.
    - Проводить анализ логов в рамках еженедельных или ежемесячных отчётов о состоянии системы.
- **Интеграция с SIEM-системами:**
    - Рассмотреть возможность интеграции с системами управления информационной безопасностью (SIEM) для более глубокого анализа и корреляции событий из различных источников.
    - Это позволит оперативно реагировать на инциденты и автоматизировать процесс расследования.

4. Интеграция с внешними системами

4.1. Интеграция с основным сайтом рекламодателя

- **Цель интеграции:**
  Обеспечить бесперебойный обмен данными между сервисом партнёрской программы и основным сайтом рекламодателя (kupibilet.ru). Это позволит:
  - Синхронизировать учетные записи и данные пользователей (например, регистрация, авторизация, изменение профилей).
  - Передавать статистические данные (клики, конверсии, транзакции) для оперативного анализа и управления рекламными кампаниями.
  - Обеспечить единый механизм управления кампаниями и объявлениями, где обновления на одном сайте сразу отражаются на другом.
- **Основные сценарии взаимодействия:**
  - **Синхронизация пользователей:** автоматическая передача данных о новых регистрациях и изменениях профилей.
  - **Обмен статистикой:** регулярное обновление информации о посещениях, кликах, и конверсиях.
  - **Управление кампаниями:** получение актуальных данных о кампаниях и обновление их статусов в реальном времени.

4.2. API и протоколы взаимодействия

- **Протоколы обмена:**
  - Основным протоколом для интеграции является **HTTP/HTTPS**. Для обеспечения безопасности и целостности данных все запросы должны

осуществляться по защищённому соединению (HTTPS).

- При необходимости обмена данными в режиме реального времени можно рассмотреть использование **WebSocket**.

- **Структура API:**
  - **REST API:** рекомендуется реализовать API в соответствии с принципами REST. Передача данных должна осуществляться в формате JSON, что обеспечит простоту интеграции и широкую совместимость.
  - **Аутентификация и авторизация:**
    - Для защиты API использовать современные стандарты аутентификации, такие как **OAuth2.0** или **JWT**. Это гарантирует, что доступ к данным получат только авторизованные клиенты.
  - **Документация:**
    - Все методы API, входные параметры и форматы ответов должны быть подробно задокументированы, предпочтительно с использованием спецификации OpenAPI (Swagger). Это облегчит интеграцию с основными системами и сторонними сервисами.
- **Методы API (примерный перечень):**
  - **Пользовательские операции:**
    - Регистрация/аутентификация пользователей, обновление данных профиля.
  - **Статистика и отчётность:**

        - Получение статистических данных по рекламным кампаниям, кликам, конверсиям.
    - **Управление кампаниями:**
        - Создание, обновление и удаление рекламных кампаний, а также получение их текущего состояния.
    - **Уведомления:**
        - Передача событий (например, успешных транзакций или ошибок) для оперативного реагирования.

4.3. Дополнительные аспекты интеграции

- **Тестовая среда (Sandbox):**
    - Предусмотреть отдельную тестовую (sandbox) среду, где можно будет отладить интеграцию и протестировать работу API без воздействия на продуктивную систему.
- **Обработка ошибок и логирование:**
    - В случае сбоя или некорректного ответа API необходимо возвращать информативные сообщения об ошибках с указанием кода ошибки и описанием проблемы.
    - Логировать все запросы и ответы для дальнейшего анализа и устранения неполадок.
- **Версионирование API:**
    - При внесении изменений в API предусмотреть механизм версионирования, чтобы обеспечить совместимость с уже интегрированными системами и минимизировать влияние на текущих пользователей.

5. Управление ошибками и аварийное восстановление

5.1. Обработка ошибок и сбои

- **Логирование ошибок:**
    - Все ошибки, исключения и аварийные события должны автоматически фиксироваться в централизованном хранилище логов с указанием времени, уровня (DEBUG, INFO, WARNING, ERROR, CRITICAL) и подробного описания проблемы.
    - Использование структурированного формата логов (например, JSON) обеспечит удобство последующего анализа и корреляции событий.
- **Классификация и уведомление:**
    - Определить категории ошибок (информационные, предупреждения, критические) и разработать для них алгоритмы уведомления.
        - **Критические ошибки:** мгновенное уведомление ответственных специалистов (через email, SMS или мессенджеры) для оперативного реагирования.
        - **Ошибки уровня WARNING:** накопление и периодическая проверка с последующим уведомлением, если их количество превышает установленный порог.
- **Обработка сбоев в реальном времени:**
    - Внедрить механизмы автоматического обнаружения сбоев на основе мониторинга ключевых метрик (например, недоступность сервисов, резкое увеличение времени отклика).
    - Реализовать алгоритмы повторных попыток (retry) для автоматического восстановления операций, где это

применимо, а также автоматический перезапуск или переключение на резервные узлы в случае отказа.

- **Документирование инцидентов:**
    - Каждая аварийная ситуация должна сопровождаться подробным отчетом, включающим анализ причин, предпринятые меры и рекомендации по предотвращению повторных инцидентов.
    - Регулярно проводить анализ накопленных логов для выявления закономерностей и оптимизации процессов обработки ошибок.

5.2. Процедуры резервного копирования и восстановления

- **Резервное копирование данных:**
    - **Периодичность:** настроить автоматизированное резервное копирование критически важных данных (базы данных, файловые системы, конфигурационные файлы) с определённой частотой (например, ежедневно или еженедельно) в зависимости от объёма и важности данных.
    - **Географическая избыточность:** хранить резервные копии в нескольких географически распределённых местах или использовать облачные сервисы, что обеспечивает защиту от локальных сбоев и катастрофических ситуаций.
    - **Безопасность резервных копий:** Резервные копии должны шифроваться и храниться с ограниченным доступом, чтобы предотвратить несанкционированный доступ.
- **Процедуры восстановления:**
    - **Документированный план аварийного восстановления (DR):**

        - Разработать пошаговую инструкцию, описывающую порядок действий при аварийной ситуации, включая идентификацию отказавших компонентов, последовательность восстановления сервисов и переключение на резервные ресурсы.
        - Определить роли и обязанности членов команды, ответственных за реализацию плана восстановления.
    - **Тестирование восстановления:**
        - Регулярно проводить тестовые сценарии восстановления (failover testing) для проверки корректности резервных копий и эффективности плана аварийного восстановления.
        - Документировать результаты тестов и корректировать план восстановления при выявлении недостатков.
- **Мониторинг восстановления:**
    - Внедрить систему контроля за процессом восстановления, позволяющую отслеживать статус выполнения плана и своевременно информировать руководство и ответственных специалистов о прогрессе и возможных проблемах.
- **Актуализация документации:**
    - Все процедуры резервного копирования и аварийного восстановления должны регулярно пересматриваться и обновляться с учетом изменений в инфраструктуре, новых угроз и результатов тестирования.

6. Документация и поддержка

6.1. Документация

- **Комплексная техническая документация:**
    - **Руководство пользователя:**
        - Подробное описание функциональности сервиса для конечных пользователей (веб-мастеров, рекламодателей, администраторов, маркетологов).
        - Пошаговые инструкции по работе с личными кабинетами, описания интерфейсов, сценарии типовых операций и ответы на часто задаваемые вопросы (FAQ).
    - **Руководство для разработчиков:**
        - Подробное описание архитектуры системы, внутренних модулей и взаимодействия компонентов.
        - Документация по API и протоколам интеграции, включая спецификации (например, OpenAPI/Swagger), схемы данных и примеры запросов/ответов.
    - **Руководство администратора:**
        - Инструкции по установке, настройке, обновлению и мониторингу сервисов, а также порядок выполнения резервного копирования и восстановления системы.
    - **Документация по безопасности:**
        - Описание механизмов защиты, регламентов обработки ошибок, аварийного восстановления и процедуры реагирования на инциденты.

- **Форматы и доступность документации:**
    - Документация должна быть доступна в электронном виде (например, PDF, онлайн-вики, интерактивная база знаний) с возможностью быстрого поиска и навигации.
    - Обеспечить версионирование документов, чтобы все участники проекта имели доступ к актуальной информации.
    - Документация должна обновляться синхронно с выпуском новых версий сервиса, а также корректироваться на основе обратной связи от пользователей и команды поддержки.

6.2. Поддержка

- **Организация службы поддержки:**
    - **Контактные каналы:**
        - Организовать несколько каналов для получения запросов от пользователей (email, телефон, онлайн-чат, система тикетов).
        - Предусмотреть централизованную систему обработки обращений, позволяющую отслеживать и документировать все инциденты.
    - **Классификация и уровни поддержки:**
        - Определить уровни поддержки (первичная, вторичная, эскалация) с соответствующими сроками реагирования (SLA) и путями эскалации для критических инцидентов.
        - Разработать процедуры обработки обращений, включая регистрацию, диагностику, решение проблемы и информирование пользователя о статусе обращения.

- **Обслуживание и сопровождение:**
  - Обеспечить регулярное техническое обслуживание сервиса, обновление ПО, выпуск патчей и модернизацию системы.
  - Определить процедуру проведения плановых профилактических работ, включая уведомление пользователей о возможных перерывах в работе.
- **Обучение и инструктаж:**
  - Провести обучение для сотрудников технической поддержки, администраторов и конечных пользователей.
  - Разработать обучающие материалы и проводить регулярные тренинги или вебинары для ознакомления с новыми функциональными возможностями и изменениями в системе.

- **Мониторинг и обратная связь:**
  - Внедрить систему сбора обратной связи от пользователей, которая позволит оперативно выявлять проблемные области и корректировать работу сервиса.
  - Регулярно анализировать статистику обращений в службу поддержки для определения повторяющихся проблем и выработки рекомендаций по улучшению функциональности и стабильности сервиса.