МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РЕАЛИЗАЦИЯ ПЛАТФОРМЫ ДЛЯ ВЗАИМОДЕЙСТВИЯ ОРГАНИЗАТОРОВ И ДИСТРИБЬЮТОРОВ МЕРОПРИЯТИЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студентки 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Посохова Вадима Алексеевича

Научный руководитель
зав. каф., к. ф.-м. н., доцент _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 21 июня 2024 г.

ВВЕДЕНИЕ

Массовые мероприятия представляют собой важную часть культурной жизни общества, обеспечивая возможность для артистов и организаторов взаимодействовать с аудиторией. Однако, процесс организации и распространения таких мероприятий, например концертов, зачастую сопровождается множеством сложностей, связанных в первую очередь с процессом покупки билетов.

В современных условиях цифровизации всё более актуальным становится применение информационных технологий для оптимизации этих процессов.

Таким образом, целью дипломной работы является разработка платформы, которая обеспечит понятное и доступное взаимодействие организаторов и дистрибьюторов мероприятий.

Более того, при создании данной платформы нужно учитывать современные подходы к построению высоконагруженных распределенных систем, с целью обеспечения отказоустойчивости к большому потоку запросов.

Для достижения этой цели были поставлены следующие задачи:

1. Изучение существующих решений на рынке и их анализ
2. Исследование принципов построения микросервисной архитектуры и её преимуществ для разработки распределённых систем.
3. Разработка архитектурного решения платформы
4. Реализация работающего прототипа продукта

Структура и объём работы.

Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 3 основные главы, заключение, список использованных источников из 22 наименований и 4 приложений. Работа изложена на 109 страницах, содержит 18 рисунков.

В первой главе исследуется предметная область, приводится анализ существующих аналогов, выявляются бизнес требования к продукту, производится выбор инструментальных средств, необходимых для выполнения дипломной работы, рассматриваются примеры различных подходов к построению высоконагруженных распределенных систем.

Вторая глава посвящена реализации платформы, удовлетворяющей всем выявленным бизнес требованиям. В данной главе приведено описание архитектуры серверной части, схемы базы данных, а также присутствуют листинги реализованных сервисов.

В третьей главе демонстрируется пример использования платформы.

Выпускная квалификационная работа заканчивается заключением, списком использованных источников, а также приложениями с кодом А-Г.

1 Основное содержание работы

1.1 Технические аспекты продукта

Актуальность продукта

В условиях стремительного роста индустрии мероприятий и увеличения числа событий, требующих эффективной организации и распространения, разработка специализированных платформ становится особенно актуальной. Современные технологии позволяют создавать интегрированные решения, которые значительно упрощают процессы планирования, продажи билетов и управления мероприятиями.

Бизнес требования

Разрабатываемый продукт должен обеспечивать:

- Эффективное управление мероприятиями: Возможность организаторам создавать, редактировать и управлять мероприятиями, включая информацию о дате, месте проведения, доступных билетах и других деталях.
- Поддержку сотрудничества: Платформа должна позволять дистрибьюторам и организаторам предлагать сделки, в рамках которых будет распределяться бюджет покупок и доходов от продажи билетов.
- Механизм виджетов: Платформа должна включать механизм виджетов для одобренных сделок, позволяющий сторонним пользователям совершать покупки билетов непосредственно через интегрированные виджеты.
- Интеграцию с внешними сервисами: Платформа должна иметь возможность интеграции с различными внешними сервисами, такими как платежные системы, системы управления мероприятиями и социальные сети, для расширения функциональности и улучшения пользовательского опыта.
- Отказоустойчивость и гибкость: Платформа должна быть отказоустойчивой, обеспечивая непрерывность работы даже в случае отказа отдельных компонентов системы. Она также должна быть гибкой, чтобы легко адаптироваться к изменяющимся бизнес-требованиям.

Описание продукта

Разрабатываемая платформа представляет собой решение для взаимодействия организаторов и дистрибьюторов мероприятий. Основная идея продукта заключается в создании единого цифрового пространства, где обе стороны могут эффективно взаимодействовать, договариваться о сотрудничестве и осу-

ществлять продажу билетов на мероприятия.

Каждый новый пользователь при регистрации выбирает одну из двух доступных ролей:

1. Организатор - занимается планированием и проведением мероприятий. Он имеет возможность создавать новые мероприятия, управлять их параметрами и предлагать сотрудничество распространителям.
2. Распространитель - занимается продвижением и продажей билетов на мероприятия. Распространитель может просматривать все доступные мероприятия и предлагать свои услуги организаторам.

Процесс сотрудничества представлет собой следующую концепцию: платформа обеспечивает двустороннее взаимодействие между организаторами и распространителями, в котором при согласии обеих сторон на сотрудничество по конкретному мероприятию, создается виджет покупки билета, с учетом обговоренных в договоредолей распределения доходов от продажи билетов.

Одной из ключевых особенной проекта является генерация данного виджета. Виджет представляет собой интерактивный элемент, который распространяется через дистрибьютора. То как будет происходить распространение данного виджета зависит лишь от пользователя. Таким образом, он может внедрить ссылку на виджет, например, в html код собственного сайта, или внутрь приложения, группы в социальной сети.

Любой человек, который приобретает билет через этот виджет, получает электронный билет на свою электронную почту. После покупки билета деньги автоматически распределяются между организатором и дистрибьютором в соответствии с условиями их договора. Это обеспечивает прозрачность и справедливость финансовых операций, упрощая процесс учета и управления доходами.

Таким образом можно выделить следующие преимущества платформы:

1. Упрощение взаимодействия.
2. Автоматизация процессов распределения денежных средств.
3. Гибкость использования виджета.

Используемые технологии

Для реализации API был выбран подход с использованием gRPC вместо стандартного REST API. gRPC использует протокол HTTP/2, который обеспечивает более высокую скорость передачи данных по сравнению с HTTP/1.1, используемым в REST API. Это особенно важно для приложений, работающих

с большими объемами данных и требующих низкой задержки.

Для реализации платформы использовались следующие технологии:

— Golang;
— PostgreSQL;
— Docker;
— gRPC, grpc-web;
— Envoy Proxy;
— React Native.

Принципы микросервисной ахритектуры

Микросервисная архитектура представляет собой подход к разработке программного обеспечения, при котором приложения разбиваются на независимые модули (микросервисы), каждый из которых выполняет определённую бизнес-функцию. Этот подход позволяет улучшить масштабируемость, управляемость и гибкость приложений.

Такая архитектура противопоставляется монолитной архитектуре, где все процессы происходят в одном приложении, что увеличивает громоздкость кода, делает его менее читабельным.

Правильная декомпозиция приложений, эффективное взаимодействие микросервисов с базой данных и поддержка консистентности данных являются ключевыми аспектами успешной реализации микросервисной архитектуры.

1.2 Реализация продукта

К основным этапам разработки данной платформы отсносятся:

1. Проектирование микросервисной архитектуры
2. Реализация функционала сервисов, удовлетворяющих требованиям продукта
3. Контейнеризация серверной части
4. Реализация клиентской части платформы в виде web-страницы
5. Связывание клиента и сервера при помощи Envoy Proxy

Архитектура приложения

Архитектура платформы включает в себя несколько микросервисов, каждый из которых отвечает за определённую функциональность системы, а также клиентский интерфейс. Компоненты составляющие данную архитектуру:

— Пользовательский интерфейс (веб браузер).
— Proxy выполняющий роль посредника между клиентом и gRPC API.

— Сервис аутентификации (Auth Service): Отвечает за управление правами пользователя, вход в систему и регистрацию. Проводит его аутентификацию и генерирует JWT-токены, которые в дальнейшем проверяются в каждом сервисе по общему ключу.
— Сервис мероприятий (Event Service): Отвечает за создание, обновление и управление мероприятиями. Пользователь может создавать новые мероприятия, задавать их параметры и публиковать на платформе.
— Сервис билетов (Ticket Service): Отвечает за генерацию билетов на мероприятие.
— Сервис платежей (Payment Service): Отвечает за выполнение транзакции при выполнении покупки билета.
— Сервис сделок (Deals Service): Сервис сделок отвечает за управление предложениями сотрудничества между организаторами и распространителями.
— Общая база данных (Shared DB): В данной архитектуре используется общая база данных для всех сервисов. Несмотря на это, каждый сервис имеет собственный интерфейс для работы с нужными ему методами БД. Также для получения информации между сервисами настроено gRPC взаимодействие. Таким образом вместо непосредственного запроса в БД делает клиентский запрос к сервису. Данный подход позволяет в любой момент с легкостью перейти на концепцию «DB per service», что может быть полезно для распределения нагрузки при реплицировании экзмепляров БД.

Реализация каждого сервиса разделена на следующие логические слои:

— Слой приложения отвечает за инициализацию и координацию работы других слоёв, запуск gRPC сервера.
— Слой gRPC хендлеров отвечает за обработку gRPC запросов, описанных в интерфейсе.
— Сервисный слой содержит основную бизнес-логику сервиса.
— Слой базы данных содержит методы для выполнения CRUD операций.

Реализация сервиса аутентификации

Сервис аутентификации отвечает за логирование и регистрацию пользователя, а также определение его роли на платформе.

Методы сервиса:

— Register — Регистрация нового пользователя.

— Login — Вход пользователя в систему.
— IsOrganiser — Проверка, является ли пользователь организатором.
— IsDistributor — Проверка, является ли пользователь дистрибьютором.
— IsBuyer — Проверка, является ли пользователь покупателем.
— IsAdmin — Проверка, является ли пользователь администратором.

Реализация сервисов управления мероприятиями и билетами

Сервисы EventService и TicketService предоставляют возможности для создания, удаления и обновления мероприятий и билетов. Эти операции с изменением данных требуют наличия соответствующих прав, которые могут быть у организатора или администратора. Для обеспечения безопасности и проверки прав пользователя был настроен middleware, который осуществляет парсинг JWT токена и определяет роль пользователя.

Методы сервиса мероприятий:

— AddEvent — Добавление нового мероприятия.
— UpdateEvent — Обновление данных существующего мероприятия.
— DeleteEvent — Удаление существующего мероприятия.
— GetEvent — Получение информации об мероприятии.
— GetEventCategory — Получение информации о типах билетов для мероприятия.
— GetAllEvents — Получение всех мероприятий.
— GetPrevEvents - Получение пройденных мероприятий для определенного организатора.

Методы сервиса билетов:

— AddTicket — Создание нового билета.
— GetTicketImage — Получение билета в виде фотографии.
— GetTicketInfo — Получение общей информации о билете.
— ActivateTicket — Активация билета.

Реализация обработки транзакций

Сервис обработки транзакций выполняет проверку возможности покупки билета, создание билета и распределение условных денежных средств между организатором и дистрибьютором.

Он содержит два метода: PurchaseTicket и SendTicket.

Процесс покупки билета включает следующие шаги:

— Формирование структуры PurchaseInfo на основе данных из запроса, вклю-

чая информацию о покупателе и выбранных билетах.

— Создание нового токена покупки с помощью метода s.tm.NewPurchaseToken(). Если создание токена не удалось, возвращается ошибка с соответствующим статусом (codes.Internal).
— Проверка возможности покупки билетов. Если проверка не удалась или билетов недостаточно, возвращается ошибка с соответствующим статусом (codes.Internal или codes.FailedPrecondition).
— Создание билетов с использованием метода s.payment.CreateTickets(ctx, purchaseInfo, purchaseToken). Если создание билетов не удалось, возвращается ошибка с соответствующим статусом (codes.Internal).
— Принятие сделки с помощью метода s.payment.SubmitPurchase(ctx, tikcetsID, purchaseInfo.WidgetID). Если процесс принятия сделки не удался, возвращается ошибка с соответствующим статусом (codes.Internal).
— Возвращение ответа с идентификатором созданных билетов (tikcetsID) при успешном завершении всех операций.

Данный сервис почти не работает напрямую с базой данных. Для получения этих данных он реализует клиентские gRPC запросы в сервисы `Event`, `Tickets` и `Deals`. Для того чтобы это реализовать необходимо сначала выполнить tcp соединение между сервисами, используя сгенерированные proto файлы.

Реализация процесса сделок

Сделка оформляется между двумя сторонами ровно на одно мероприятие, так как виджет покупки выдается на одно мероприятие. Сделка может существовать в одном из трех состояний:

— Принята (ACCEPTED)
— Отклонена (REJECTED)
— В ожидании ответа (PENDING)

Сервис `deals` предоставляет возможность предлагать сделку, отклонять, или принимать предложение. Также присутствуют механизмы просмотра сделок по ее текущему состоянию.

Методы сервиса сделок:

— OfferDeal — Предложение сделки.
— AcceptDeal — Принятие сделки.
— RejectDeal — Отклонение сделки.
— GetSentDeals — Получение всех отправленных сделок.

— GetProposedDeals — Получение всех предлагаемых сделок.
— GetDealsByStatus — Получение всех сделок в зависимости от статуса (принята, отклонена, в ожидании ответа).
— GetDeal — Получение информации о сделке.
— GetDealWidget — Получение виджета покупки.

Реализация взаимодействия клиента и сервера

Для реализации обработки gRPC запросов со стороны бразуера для клиента используется gRPC-Web. Однако, gRPC-Web запросы имеют отличия от обычных gRPC запросов, и многие серверные реализации gRPC не поддерживают gRPC-Web напрямую.

Envoy Proxy служит переводчиком, преобразовывая gRPC-Web запросы в стандартные gRPC запросы, что позволяет браузерным клиентам взаимодействовать с gRPC серверами без необходимости внесения изменений в серверную часть. Envoy обеспечивает поддержку двух протоколов (HTTP/1.1 и HTTP/2), что делает его мощным инструментом для управления и маршрутизации трафика.

Для клиента реализованы следующие страницы:
— Страницы логина/регистрации.
— Страница дистрибьютора с возможностью просматривать, предлагать/отклонять сделки, просматривать баланс.
— Страница организатора, с подобным фуннционалом, с учетом добавления мероприятия.
— Страница виджета покупки.

1.3 Использование платформы

Развертывание проекта

Для развертывания проекта на своей локальной машине пользователю потребуются:
— Docker и Docker-Compose
— MakeFile
— Node.js и npm

Демонстранция использования

При входе на сайт пользователю предоставляется возможность регистрации с учетом выбора роли. Для выполнения процесса регистрации нужно будет ввести следующие данные: логин, почта, пароль, выбранная роль. Также, поль-

зователь уже зарегистрированный имеет возможность войти в систему, введя почту и пароль.

Организатору предоставляется функционал для управления мероприятиями: создание, удаление, редактирование, просмотр всех существующих, а также только собственных мероприятий.

Список мероприятий для дистрибьютора изображается несколько иначе чем для организатора. Предоставляется возможность предложить сделку организатору на конкретное мероприятие, указав процент комиссии.

Организаторы и дистрибьюторы имеют возможность просматривать список сделок, принимать или же отклонять их.

При принятии сделки у дистрибьютора появляется возможность распространять виджет покупки. На данном виджете указана основная информация о мероприятии, предоставляется выбор билетов для покупки, и поля для ввода данных о покупателе.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломной работы была реализована платформа, облегчающая процесс взаимодействия организаторов и дистрибьюторов. В процессе разработки были изучены и применены следующие технологии и методики, необходимые для построения современной и эффективной системы:

1. Архитектура микросервисов: реализация системы с разделением на независимые микросервисы, что обеспечивает гибкость и масштабируемость проекта.
2. Протокол gRPC: внедрение gRPC для высокопроизводительного взаимодействия между микросервисами.
3. Работа с базой данных PostgreSQL: использование ORM для эффективного взаимодействия с базой данных и обеспечения целостности данных.
4. Контейнеризация с Docker: создание Docker-контейнеров для каждого микросервиса, что облегчает развертывание и управление инфраструктурой.
5. Оркестрация контейнеров: применение инструментов оркестрации для автоматического масштабирования и обеспечения высокой доступности системы.
6. Разработка клиентского интерфейса на React: создание интуитивно понятного и функционального интерфейса для пользователей системы.
7. Логирование и мониторинг: внедрение средств для отслеживания состояния системы, мониторинга производительности.

Разработанный продукт уже обладает значительной функциональностью и реализует главную бизнес идею. Тем не менее, на текущем этапе существуют важные функции, которые необходимо реализовать для достижения финальной версии продукта.

К данному функционалу для дальнейшего развития относится: внедрение платежной системы, разработка мобильного приложения с функцией сканирования билетов, генерация электронных билетов, добавление возможности интеграции платформы с внешними сервисами, реализация дополнительных способов распространения виджетов, таких как внедрение их в сгенерированный HTML код для использования на сайтах дистрибьюторов, помимо распространения по прямой ссылке.