Астахов К.А. Разработка архитектурных решений для создания ВЭБ-сервиса по развертыванию сайтов научных мероприятий. // Международный журнал информационных технологий и энергоэффективности. – 2024. – Т. 9 № 10(48) с. 42–50

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.15

РАЗРАБОТКА АРХИТЕКТУРНЫХ РЕШЕНИЙ ДЛЯ СОЗДАНИЯ ВЕБ-СЕРВИСА ПО РАЗВЕРТЫВАНИЮ САЙТОВ НАУЧНЫХ МЕРОПРИЯТИЙ

Астахов К.А.
ФГБОУ ВО "МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)", Москва, Россия, (125993, Москва, Волоколамское ш., д. 4), e-mail: kir190477@mail.ru

Цель: Определить основные подходы и концепции построения архитектуры веб сервиса, который позволил бы создавать и разворачивать типовые сайты научных мероприятий.
Метод: исследование всевозможных подходов разработки приложений, а также стеков технологий, которые могли бы быть использованы для решения задачи создания подобных сервисов.
Результат: получены основные концепции для решения данной задачи, определен стек технологий и реализована концепция построения архитектуры веб сервиса, позволяющая оптимально и эффективно использовать ресурсы, которые могли бы быть предоставлены в рамках реализации поставленного в задаче интернет приложения.

Ключевые слова: Разработка; веб-разработка; архитектура приложений.

DEVELOPMENT OF ARCHITECTURAL SOLUTIONS FOR THE CREATION OF A WEB SERVICE FOR THE DEPLOYMENT OF SCIENTIFIC EVENT SITES

Astakhov K.A.
MOSCOW AVIATION INSTITUTE (NATIONAL RESEARCH UNIVERSITY), Moscow, Russia, (125993, Moscow, Volokolamskoye shosse, 4), e-mail: kir190477@mail.ru

Purpose: To identify the main approaches and concepts for building a web service architecture that would allow you to create and deploy typical sites for scientific events.
Method: the study of various approaches to application development, as well as technology stacks that could be used to solve the problem of creating such services.
Result: the basic concepts for solving this problem have been obtained, the technology stack has been defined and the concept of building a web service architecture has been implemented, allowing optimal and efficient use of resources that could be provided as part of the implementation of the Internet application set in the task.

Keywords: Development; web development; application architecture.

Современная научная среда стремительно развивается, и проведение научных мероприятий становится неотъемлемой частью этого процесса. Создание веб-сайтов для научных мероприятий является важным элементом успешной организации и популяризации мероприятий. В данной работе рассматривается идея разработки интернет-сервиса, позволяющего легко и быстро создавать типовые сайты для научных мероприятий.

Актуальность, разрабатываемого интернет-сервиса для развертывания типовых сайтов научных мероприятий, объясняется несколькими важными факторами:

1) Современная диджитализация.

В современном мире диджитализация стала неотъемлемой частью организации событий. Создание веб-сайтов является одним из ключевых элементов современной коммуникации и

информационной публикации. Он предоставляет участникам мероприятий быстрый и удобный доступ к информации.[1]

2) Сокращение времени и ресурсов.

Традиционное создание веб-сайтов может быть сложным и затратным процессом, требующим навыков разработки и дизайна. Интернет-сервис для создания сайтов определенного формата позволяет экономить время и ресурсы организаторов мероприятий.

3) Доступность.

Сервисы для создания сайтов делают эту технологию доступной для широкого круга пользователей. Даже люди без специальных знаний в веб-разработке могут создать профессионально выглядящий сайт для своего мероприятия.[2]

4) Снижение ошибок и обновление информации.

Интернет-сервисы предоставляют удобные инструменты для редактирования и обновления информации на сайте. [3] Это помогает избежать ошибок в расписании и предоставлении актуальной информации.

5) Усиление профессионального восприятия.

Данный интернет-сервис обладает всеми необходимыми инструментами для создания сайта, имеющего привлекательный дизайн и необходимый функционал для сайта с такой спецификой.

В рамках поставленной задачи следует понимать, чтобы получить наиболее эффективное и оптимизированное решение для данной задачи, необходимо учитывать следующие факторы:

- Масштабируемость: возможность легко добавлять новые функции и обрабатывать большой объём запросов.
- Надёжность: система должна быть отказоустойчивой.
- Производительность: быстрая обработка запросов и развертывание сайтов.

Основываясь на данных критериях, мы будем определять подходы и стеки технологий в следующих областях:

- архитектурные подходы
- стек технологий для сервисной разработки(backend)
- стек технологий для frontend разработки
- стек технологий для развертывания подобного сервиса

Рассмотрим архитектурные подходы, которые могли бы быть использованы в рамках данной работы:

Варианты архитектурного подхода:

- Микросервисная архитектура: каждый компонент системы может быть выделен в отдельный сервис. [4] Это позволяет масштабировать отдельные компоненты и упрощает сопровождение.
- Монолитная архитектура: все компоненты интегрированы в одно приложение. Этот вариант может быть проще в разработке и развертывании на начальном этапе, но менее гибок при масштабировании.
- Serverless/Function-as-a-Service (FaaS): использование функций, которые выполняются в облаке только по мере необходимости (например, развертывание сайта по триггеру). Подходит для событийно-ориентированных задач и снижения затрат на инфраструктуру.

Наиболее подходящий вариант — микросервисная архитектура, так как она обеспечивает высокую масштабируемость и гибкость в добавлении новых функций и изменении существующих. [5] Рассмотрев критерии и возможности каждого подхода, микросервисная архитектура наиболее целесообразна для решения данной задачи. Она обеспечит баланс между гибкостью, производительностью и возможностью масштабирования.

Далее проведем анализ и определим стеки технологий, которые могли бы быть использованы в рамках данной задачи

Backend часть.

Язык программирования: **golang (Go)**

Для задачи развертывания типовых сайтов научных мероприятий важно учитывать производительность, параллелизм, простоту развертывания и обслуживания. Go известен своей производительностью, простотой и встроенной поддержкой параллелизма, что делает его удобным для подобных задач. Go компилируется в машинный код, что обеспечивает высокую скорость выполнения. [6] Это важно для сервисов, которые должны быстро обрабатывать запросы и выполнять операции по развертыванию сайтов. Также язык golang предоставляет встроенную поддержку горутин и каналов, которые позволяют легко создавать конкурентные программы. Это удобно для задач развертывания, которые могут выполняться параллельно (например, развертывание нескольких сайтов одновременно).

Данный язык обладает простым и лаконичным синтаксисом, что упрощает разработку и поддержку кода. Разработка на Go позволяет быстро создавать надежные сервисы без сложных конструкций.

Go создает статически связанный исполняемый файл, который можно запускать на любой системе без дополнительных зависимостей. Это упрощает процесс развертывания и обеспечивает кроссплатформенность.[7]

Также Go оснащен богатой стандартной библиотекой, включающей инструменты для работы с сетью, HTTP-серверами, синхронизацией, JSON и многое другое. [8] Это позволяет быстро создавать веб-сервисы без необходимости в дополнительных лишних зависимостей. Также golang предлагает встроенную систему сборки и управления зависимостями (Go modules), которая упрощает управление проектом и его развертывание.

Исходя из преимуществ языка можно отметить, что go - отличный выбор для создания высокопроизводительных, надежных и легко масштабируемых интернет-сервисов.

Web фреймворк: **Fiber**

Для создания интернет-сервиса на Go есть несколько веб-фреймворков, и выбор подходящего зависит от требований задачи. В данном случае Fiber может быть хорошим выбором, учитывая его особенности и преимущества для создания высокопроизводительных веб-сервисов. Fiber построен на основе fasthttp, одного из самых быстрых HTTP-стеков для Go. Это делает Fiber крайне производительным и эффективным при обработке большого количества запросов, что важно для сервиса, который может обслуживать множество пользователей и быстро развертывать сайты. а благодаря оптимизированному процессу обработки запросов, Fiber обеспечивает низкие задержки, что способствует быстрому реагированию сервиса на пользовательские запросы.[9]

Fiber позволяет легко обрабатывать запросы асинхронно, что важно для задач, которые требуют высокой пропускной способности и быстрого развертывания сайтов. Более того, Fiber

предлагает гибкую и мощную систему маршрутизации, которая позволяет легко управлять различными путями и их обработчиками, что удобно при создании RESTful API для управления сайтами.[10]

Встроенные Middleware. Fiber поставляется с набором встроенных middleware (например, для работы с CORS, сессиями, сжатиями), которые могут быть полезны для разработки интернет-сервиса, избавляя от необходимости разрабатывать эти компоненты с нуля.

Малый размер и легковесность: Fiber разработан с акцентом на минимализм, что делает его легковесным и быстрым в развертывании. Это снижает нагрузку на систему и позволяет эффективнее использовать ресурсы.

Совместимость с существующей экосистемой: Fiber хорошо сочетается с другими библиотеками и инструментами Go, что позволяет использовать дополнительные модули для базы данных, авторизации и других задач без сложной интеграции.

Использование Fiber в контексте данной задачи позволит быстро и эффективно разработать высокопроизводительный и отзывчивый интернет-сервис для развёртывания типовых сайтов научных мероприятий.

База данных: **CockroachDB**

CockroachDB — это распределенная реляционная база данных, которая известна своей высокой доступностью, горизонтальной масштабируемостью и совместимостью с SQL. В контексте задачи по развёртыванию типовых сайтов научных мероприятий эти особенности могут быть особенно полезными. CockroachDB автоматически масштабируется горизонтально, добавляя новые узлы к кластеру. Это означает, что база данных может легко обрабатывать увеличение нагрузки, что полезно, когда количество сайтов и пользователей растет. Выбранная СУБД, спроектирована таким образом, чтобы быть устойчивой к сбоям, обеспечивая высокую доступность данных. Она использует автоматическое распределение и репликацию данных между узлами, что делает её способной продолжать работу даже при выходе из строя отдельных узлов. Это критически важно для интернет-сервиса, который должен быть доступен и надежен.[11]

Совместимость с SQL. CockroachDB полностью совместима с SQL, что упрощает работу с базой данных для разработчиков, знакомых с традиционными реляционными системами. Это облегчает создание и выполнение сложных запросов для управления данными о сайтах и пользователях.[12]

Geo-репликация и низкая задержка. CockroachDB поддерживает географическое распределение данных, что позволяет хранить данные ближе к пользователям, уменьшая задержки при доступе к ним. Это особенно полезно для сервиса, который может иметь глобальных пользователей, посещающих сайты научных мероприятий из разных регионов.

Простое управление и автоматическое распределение. CockroachDB автоматизирует задачи распределения данных и балансировки нагрузки между узлами, что снижает административные усилия. Это позволяет сосредоточиться на разработке функционала сервиса, а не на управлении базой данных. [13] CockroachDB обеспечивает строгую согласованность транзакций, что гарантирует целостность данных при выполнении операций, таких как создание или обновление информации о сайтах. Это важно для обеспечения корректности и надежности данных.

Также CockroachDB легко развернуть в различных средах, включая локальные, облачные и гибридные инфраструктуры. Это упрощает настройку базы данных в зависимости от потребностей проекта.

Кеширование данных (не реляционная база данных): **redis**

Redis — это высокопроизводительное хранилище данных в памяти, которое часто используется для кэширования, управления сессиями и очередей задач. В контексте интернет-сервиса по развёртыванию типовых сайтов научных мероприятий Redis может существенно улучшить производительность и эффективность системы. Redis хранит данные в памяти, обеспечивая чрезвычайно быструю скорость доступа. Это делает его идеальным для кэширования часто запрашиваемых данных, таких как конфигурации сайтов, шаблоны страниц или результаты запросов к базе данных. Кэширование позволяет снизить нагрузку на основную базу данных и ускорить время отклика сервиса.

Redis отлично подходит для управления сессиями пользователей благодаря своей скорости и поддержке различных структур данных. Это позволяет хранить состояние сессий и быстро его извлекать, что особенно полезно для аутентификации и авторизации пользователей. Также данная БД поддерживает работу с очередями, что может быть использовано для распределения задач по развертыванию сайтов. Например, запросы на создание или обновление сайтов можно помещать в очередь, а затем обрабатывать асинхронно, чтобы не блокировать основной поток обработки запросов.

Публикация и подписка (Pub/Sub). Redis предоставляет механизм Pub/Sub, который позволяет создавать эффективную систему обмена сообщениями между различными компонентами сервиса. Это может быть полезно для оповещения различных частей системы о событиях, таких как завершение развертывания сайта.

Гибкость и поддержка различных структур данных. Redis поддерживает множество структур данных, включая строки, хеши, списки, множества и т.д. Это позволяет гибко использовать Redis для различных целей, будь то кэширование сложных объектов или хранение простых ключ-значений.

Простота развертывания и управления. Redis легко развернуть и управлять им. Он может быть быстро интегрирован в существующую инфраструктуру, а также предлагает механизмы резервного копирования и восстановления данных.

Высокая доступность и репликация. Redis поддерживает репликацию данных, что обеспечивает высокую доступность и отказоустойчивость. При сбоях в одном из экземпляров Redis может переключиться на резервный, обеспечивая непрерывность работы сервиса.

Контейнеризация: **Docker**

Docker — это инструмент контейнеризации, который позволяет изолировать и упаковать приложения с их зависимостями в контейнеры. В контексте интернет-сервиса для развёртывания типовых сайтов научных мероприятий использование Docker может упростить развертывание, управление и масштабирование приложения. Docker позволяет упаковать приложение и все его зависимости в единый контейнер. Это обеспечивает консистентное рабочее окружение при переносе приложения между различными этапами разработки, тестирования и производства. Разработчики и DevOps-инженеры будут уверены, что приложение работает одинаково везде, независимо от особенностей среды.

С Docker развертывание приложений становится проще и быстрее. Вы можете создавать образы контейнеров с настроенным приложением, которые легко развертываются на любой

машине, поддерживающей Docker. Это сокращает время настройки и развертывания новых инстансов сервиса. Docker позволяет легко масштабировать сервис, запуская дополнительные экземпляры контейнеров при росте нагрузки. Оркестрационные инструменты, такие как Kubernetes, могут использоваться вместе с Docker для управления кластером контейнеров, обеспечивая автоматическое масштабирование и балансировку нагрузки.

Облегчение CI/CD. Docker интегрируется с системами CI/CD, такими как GitLab CI/CD или GitHub Actions, позволяя автоматически собирать, тестировать и развертывать контейнеры. Это ускоряет процесс разработки и поставки новых версий приложения, обеспечивая непрерывную интеграцию и доставку.

Контейнеры Docker используют меньше ресурсов по сравнению с виртуальными машинами, так как они разделяют ядро операционной системы, но при этом изолируют процессы. Это позволяет более эффективно использовать ресурсы сервера и запускать больше инстансов сервиса на одном оборудовании. Также Если интернет-сервис разрабатывается по микросервисной архитектуре, Docker облегчает управление каждым микросервисом как отдельным контейнером. Это позволяет независимо разрабатывать, тестировать и развертывать компоненты системы.

Простота в управлении зависимостями.С Docker, все зависимости приложения (библиотеки, инструменты, конфигурации) включены в контейнер. Это устраняет проблемы с несовместимостями зависимостей и конфигураций на разных серверах или окружениях.В дополнение можно добавить, что Docker позволяет быстро создавать изолированные окружения для тестирования. Это удобно для автоматизации тестов, создания тестовых инстансов сервисов и проверки работы приложения в различных сценариях.

Использование Docker в рамках данной задачи обеспечивает эффективное управление окружением, упрощает развертывание и масштабирование, а также улучшает процесс разработки и тестирования интернет-сервиса для развёртывания типовых сайтов научных мероприятий.

Frontend часть сервиса:

Фреймворк: React

React — это JavaScript-библиотека для создания пользовательских интерфейсов, известная своей эффективностью и гибкостью. В контексте задачи по разработке интернет-сервиса для развёртывания типовых сайтов научных мероприятий, React может быть полезен для создания интуитивного и динамичного интерфейса административной панели и пользовательского взаимодействия. React позволяет создавать приложения из отдельных, переиспользуемых компонентов. Для административной панели, где есть множество повторяющихся элементов, таких как формы, списки и модальные окна, это облегчает разработку и поддержку кода. Компоненты можно легко собирать, изменять и использовать в разных частях приложения. React использует виртуальный DOM, который минимизирует операции с реальным DOM и повышает производительность приложения. Это важно для создания отзывчивого интерфейса административной панели, особенно при частом обновлении данных, например, при изменении настроек сайтов или просмотре статистики.С React также легко управлять состоянием приложения, что упрощает создание сложных и интерактивных пользовательских интерфейсов. Для административной панели это полезно для отслеживания состояния форм, фильтрации и сортировки данных, а также для управления сложными пользовательскими взаимодействиями.

React имеет обширную экосистему сторонних библиотек и инструментов, таких как React Router для управления маршрутизацией, Redux или Zustand для управления состоянием, а также множество компонентов и виджетов. Это ускоряет процесс разработки и позволяет легко расширять функциональность приложения.

React хорошо интегрируется с инструментами для серверного рендеринга (SSR) и статической генерации страниц, такими как Next.js. Это может быть полезно для создания страниц сайтов научных мероприятий с лучшей производительностью и SEO-оптимизацией.

React позволяет легко добавлять интерактивные элементы и динамические обновления интерфейса. Это важно для создания удобного и отзывчивого интерфейса, который позволяет администраторам быстро настраивать и развертывать сайты. Стоит добавить, что React также легко интегрируется с RESTful API, что упрощает взаимодействие с серверной частью интернет-сервиса, например, для управления сайтами, получения статистики и других операций.

Использование React для создания административной панели и пользовательского интерфейса в рамках данной задачи позволяет разработать производительное, динамичное и легко масштабируемое приложение, которое обеспечивает удобство взаимодействия для администраторов и пользователей.

Развертывание веб-сервиса (DevOps часть): **CI/CD: GitLab CI/CD**

GitLab CI/CD — это интегрированная в GitLab система непрерывной интеграции и доставки, которая позволяет автоматизировать сборку, тестирование и развертывание приложений. В контексте задачи по разработке интернет-сервиса для развёртывания типовых сайтов научных мероприятий, GitLab CI/CD может существенно облегчить процесс разработки и обеспечить быструю поставку новых функций. GitLab CI/CD тесно интегрирован с репозиториями в GitLab, что позволяет автоматически запускать процессы CI/CD при каждом изменении в коде (например, при коммитах или создании merge request). Это обеспечивает быстрый и автоматизированный процесс сборки и развертывания приложения.

GitLab CI/CD позволяет автоматизировать все этапы разработки, включая сборку, тестирование, линтинг, деплой и даже проверку безопасности. Это снижает количество ручной работы и минимизирует риск ошибок при развертывании новых версий приложения.

С GitLab CI/CD можно настраивать сложные пайплайны, состоящие из нескольких этапов и задач. Например, можно создать пайплайн, который сначала запускает тесты, затем создает Docker-образы для микросервисов, а после успешного прохождения всех проверок автоматически развертывает их на сервере или в облаке.

Поддержка контейнеров и Docker. GitLab CI/CD отлично поддерживает работу с Docker. Это позволяет создавать Docker-образы в рамках пайплайна, тестировать их и развертывать в контейнерных средах. Для задачи по развёртыванию сайтов это удобно, так как сам сервис и его компоненты могут быть упакованы в контейнеры. GitLab предоставляет мощные инструменты контроля доступа и безопасности, такие как доступ на основе ролей, интеграция с системами аутентификации, и проверка кода на уязвимости. Это важно для обеспечения безопасности процесса разработки и развертывания.

Мониторинг и трассировка: GitLab CI/CD предоставляет инструменты для мониторинга состояния пайплайнов, отслеживания истории развертываний и выявления проблем. Это облегчает отслеживание и диагностику проблем при развертывании новых версий приложения.

Простота в использовании. Одним из важных преимуществ GitLab CI/CD является факт того, что он имеет интуитивно понятный YAML-синтаксис для определения пайплайнов, что позволяет быстро настраивать и изменять процессы CI/CD. Это делает его доступным для разработчиков с различным уровнем опыта.

Использование GitLab CI/CD в рамках данной задачи обеспечивает автоматизацию, надежность и безопасность процесса разработки и развертывания интернет-сервиса для развёртывания типовых сайтов научных мероприятий, что ускоряет выпуск новых функций и повышает качество приложения.

Выводы

В ходе работы были определены и описаны технологии и методы для реализации веб-сервиса для развертывания типовых научных сайтов мероприятий. Были разобраны достоинства и причины, по которым были отобраны стеки технологий, описанных в статье. Был определен вид веб-сервисной архитектуры, наиболее подходящий для этой задачи. Опираясь на разработку решений задачи по созданию интернет-сервиса проведенной в статье, проясняется список требований и действий, позволяющий разработчику приступить к написанию такого интернет-сервиса.

Список литературы

1. Фаулер М. - "Шаблоны архитектуры корпоративных приложений" - Addison-Wesley Professional - 2015 - 560с.
2. Донован А. А., & Керниган, Б. В. - "Язык программирования Go" - Addison-Wesley Professional - 2015 - 400с.
3. Балбаерт И. - "Go веб-программирование" - Manning Publications - 2016 - 256с.
4. Ньюман С. - "Создание микросервисов: проектирование мелкозернистых систем" - O'Reilly Media - 2021 - 616с.
5. Бэнкс А., Порчелло Э. - "Изучаем React: современные шаблоны для разработки приложений на React" - O'Reilly Media - 2020 - 350с.
6. Виерух Р. - "Дорога к React: ваш путь к освоению чистого и практичного React.js" - Self-published - 2018 - 200с.
7. Редмонд Э., & Уилсон, Дж. Р. - "Семь баз данных за семь недель: руководство по современным базам данных и движению NoSQL" - Pragmatic Bookshelf - 2012 - 352с.
8. Меркель Д. - "Docker: легковесные контейнеры Linux для согласованной разработки и развертывания" - Linux Journal - 2014 - 100с.
9. Паль К., & Броги А. - "Технологии облачных контейнеров: обзор современных технологий" - Journal of Cloud Computing: Advances, Systems and Applications - 2019 - 20с.
10. Джоши С. - "Redis на практике" - Packt Publishing - 2013 - 124с.
11. Карлос С., & Чинчилла, Дж. - "Redis для чайников" - Wiley - 2019 - 256с.
12. Бёрнс Б., Беда Дж., & Хайтауэр К. - "Kubernetes: в действии" - O'Reilly Media - 2017 - 250с.
13. Копленд М. - "Книга о Docker: контейнеризация — это новая виртуализация" - Self-published - 2015 - 260с.

References

1. Fowler, M. - "Patterns of enterprise application architecture" - Addison-Wesley Professional - 2015 - p.560
2. Donovan, A. A., & Kernighan, B. V. - "The Go Programming Language" - Addison-Wesley Professional - 2015 - p.400
3. Balbaert, I. - "Go web programming" - Manning Publications - 2016 - p.256
4. Newman, S. - "Creating microservices: designing fine-grained systems" - O'Reilly Media - 2021 – p. 616
5. Banks, A., Porcello, E. - "Studying React: modern templates for developing applications on React" - O'Reilly Media - 2020 - p.350
6. Vieruch, R. - "The road to React: your path to mastering clean and practical React.js" - Self-published - 2018 - pp.200
7. Redmond, E., & Wilson, J. R. - "Seven Databases in seven Weeks: a guide to Modern databases and the NoSQL Movement" - Pragmatic Bookshelf - 2012 - p. 352
8. Merkel, D. - "Docker: Lightweight Linux containers for coordinated development and deployment" - Linux Journal - 2014 – p.100
9. Pal, K., & Brogi, A. - "Cloud Container technologies: an overview of modern technologies" - Journal of Cloud Computing: Advances, Systems and Applications - 2019 - p .20
10. Joshi, S. - "Redis in practice" - Packt Publishing - 2013 - p.124
11. Carlos, S., & Chinchilla, J. - "Redis for dummies" - Wiley - 2019 - p.256
12. Burns, B., Bede, J., & Hightower, K. - "Kubernetes: in Action" - O'Reilly Media - 2017 - p.250
13. Copland, M. - "The book about Docker: containerization is the new virtualization" — Self-published - 2015 - 260 p.