УДК 004.4

# АРХИТЕКТУРА SERVERLESS

***Кузнецов А.А., д.т.н., проф., Казаков В.Е., к.т.н., доц., Мурычева В.В., к.т.н., доц.***

*Витебский государственный технологический университет, г. Витебск, Республика Беларусь*

Реферат. *В статье представлен обзор современного шаблона построения архитектуры информационной системы, инструментов и платформ, применяющихся для её реализации.*

Ключевые слова: веб-приложение, SERVERLESS, облачные технологии.

Вначале перечислим предпосылки и возможности, которые привели к появлению serverless архитектуры в реализации приложений. Современная разработка программного обеспечения тесно связана с облачными технологиями.

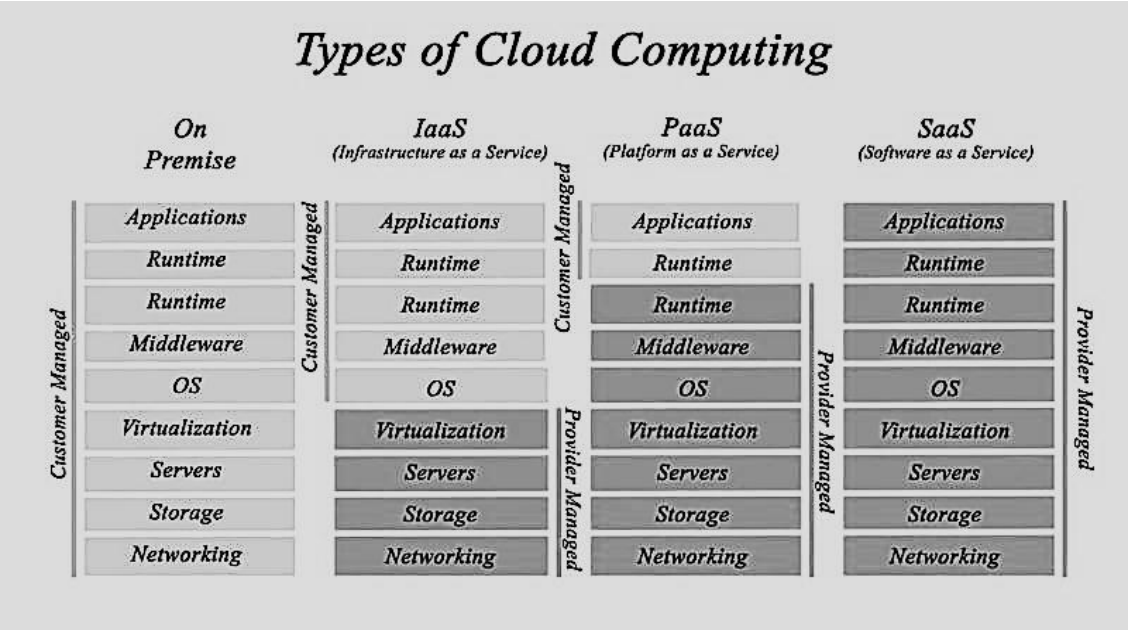


Рисунок 1 – Модели предоставления облачных услуг

На сегодняшний момент существует несколько моделей предоставления услуг в области облачных технологий (рис. 1).

IaaS «Инфраструктура как услуга» – это модель, которая концентрируется не на программных продуктах, а на необходимых для их работы аппаратных мощностях. Потребителям предоставляются фундаментальные информационно-технологические ресурсы – виртуальные серверы с заданной вычислительной мощностью, операционной системой (чаще всего – предустановленной провайдером из шаблона) и доступом к сети.

PaaS «Платформа как сервис» – это более частный случай IaaS. Пользователь арендует и аппаратные мощности, и уже адаптированные под выполнение конкретной задачи инструменты, такие как система управления базами данных.

SaaS «Программное обеспечение как сервис»: полностью готовое, заранее настроенное окружение из необходимых программных продуктов. Поставщик услуги берёт на себя: базовые настройки, регулярные обновления, поиск подходящей аппаратной части, обеспечение доступности сервиса, резервное копирование и т. д.

Кроме того, выделились и другие, более специфические модели предоставления облачных услуг:

1) DaaS «рабочее место как услуга»: размещённые в облаке виртуальные станции, на которых установлен весь нужный для работы софт;

2) MSaaS «управляемое ПО как услуга»: SaaS с большим количеством настроек и возможностей адаптировать приложение к индивидуальным бизнес-процессам компании;

3) MBaaS, «мобильный бэкэнд как услуга»: унифицированные средства подключения мобильных и веб-приложений к облачным серверным службам;

4) DCaaS «датацентр как услуга»: частный случай IaaS, адаптированный конкретно под хранение данных;

5) ITMaaS (Information Technology Management as a Service), «IT-менеджмент как услуга»: пакет приложений, обеспечивающих контроль над информационной инфраструктурой компании.

Другой предпосылкой для появления serverless архитектуры стало применение открытой REST архитектуры в различных сервисах различных провайдеров. Вследствие этого появилась возможность интеграции различных приложений облачных провайдеров между собой, а также разработки на их основе приложений независимыми разработчиками.

Термин serverless был впервые использован для описания приложений, которые значительно или полностью включают сторонние облачные приложения и службы для управления логикой и состоянием серверной стороны. Как правило, это приложения для «rich clients», одностраничные веб-приложения или мобильные приложения, которые используют обширную экосистему облачных баз данных (например, Parse, Firebase), служб аутентификации (например, Auth0, AWS Cognito) и так далее.

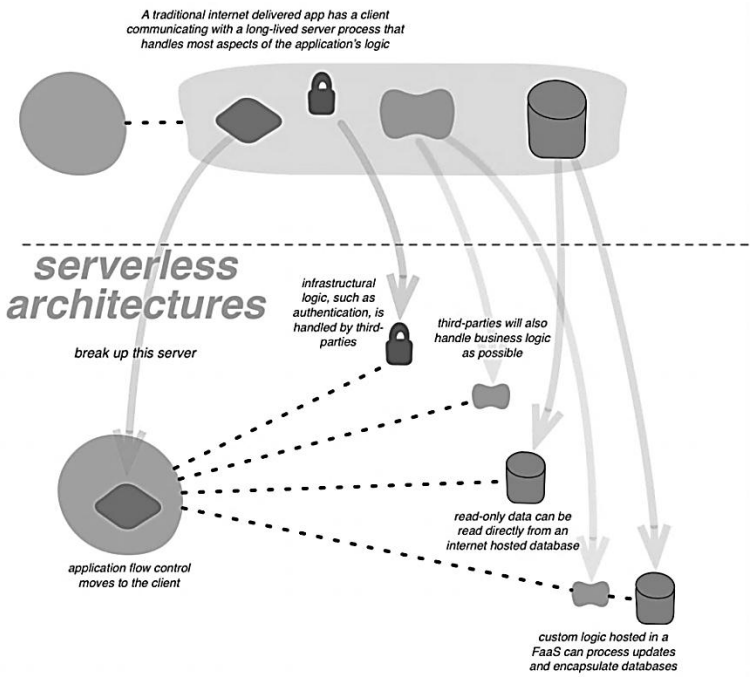


Рисунок 2 – Serverless архитектура

В Serverless-приложении нет централизованной реализации различных аспектов системы, таких как: авторизация, хранение данных и т.д. (рис. 2). Все необходимые функции такое приложение получает от сторонних сервисов. На данный момент количество таких сервисов и функционал, который они предоставляет, очень богат и постоянно увеличивается. Стоит только упомянуть, что любой информационный сайт имеет открытый REST API для предоставления доступа к своим услугам с помощью программных средств.

Прикладное приложение, использующее serverless архитектуру, реализует функции диспетчеризации запросов к различным сервисам, интерфейс с пользователем, а также прикладные функции, которые реализуют обработку информации, согласно бизнес-правилам.

Serverless может также означать приложение, где логика серверной стороны по-прежнему написана разработчиком приложения, но, в отличие от традиционных архитектур, она работает в контейнерах без состояния, которые запускаются событиями, эфемерны (может длиться только для одного запроса) и полностью управляются третьей стороной.

Развертывание таких приложений сильно отличается от традиционных систем, так как нет серверных приложений для запуска. Приложение использует так называемую FaaS (Function as a Service) модель использования облачного провайдера.

По сути, FaaS – это запуск бэкэнд-кода без управления собственными серверами или собственными серверными приложениями. Предложения FaaS не требуют кодирования в конкретную структуру или библиотеку. Функции FaaS являются регулярными приложениями, когда дело доходит до языка и окружающей среды. Например, функции AWS Lambda могут быть реализованы «первым классом» на Javascript, Python, Go, любом языке JVM (Java, Clojure, Scala и т.д.) или на любом языке .NET.

В среде FaaS мы загружаем код для нашей функции поставщику FaaS, а поставщик делает все остальное, необходимое для предоставления ресурсов, диспетчеризации запросов, управления процессами и т.д.

Конечно, такая архитектура имеет и свои недостатки.

Контроль над определённым функционалом приложения отдаётся стороннему поставщику, может приводить к простоям системы, неожиданные ограничения, изменения затрат, потеря функциональности, принудительное обновление API и многое другое.

Проблемы с разграничением доступа между клиентскими процессами, работающими на сервере провайдера (например: один клиент может получит доступ к данным другого, или ошибка в программном обеспечении одного клиента, вызывающая сбой в программном обеспечении другого).

Если вы хотите поменять поставщика определённой услуги, то вам почти наверняка нужно будет изучить новые инструменты развертывания, мониторинга. С большой вероятностью придется изменить код или даже архитектуру подсистемы, для использования другого API. Среди поставщиков существует стратегия «привязки» клиента к своему сервису, которая может заключаться в разработке специфического API, мешающего применять универсальные подходы к доступу даже сходных по задачам сервисов разных поставщиков.

Список использованных источников


1. Roberts, M. Serverless Architectures [Электронный ресурс]. – Режим доступа: https://martinfowler.com/articles/serverless.html. – Дата доступа: 04.05.2021.
2. Knorr, E. What is cloud computing? Everything you need to know. – Режим доступа: https://www.infoworld.com/article/2683784/what-is-cloud-computing.html. – Дата доступа: 24.04.2021.
3. Материалы из раздела serverless сайта amazon.com. – Режим доступа: https://aws.amazon.com/ru/blogs/rus/category/serverless/. – Дата доступа: 15.04.2021.
4. Обзор облачных сервисов для разработки бэкенда мобильных приложений. – Режим доступа: https://habr.com/ru/company/surfstudio/blog/463435/. – Дата доступа: 04.05.2021.



УДК 004.8


## МНОГОСЛОЙНЫЙ ПЕРСЕПТРОН И АЛГОРИТМ ОБРАТНОГО РАСПРОСТРАНЕНИЯ ОШИБКИ


***Лапко М.Л., студ., Дунина Е.Б., к.ф.-м.н., доц., Корниенко А.А., д.ф.-м.н., проф.***

*Витебский государственный технологический университет, г. Витебск, Республика Беларусь*



Реферат. *В статье рассмотрены теоретические основы моделирования алгоритма обратного распространения ошибки.*



Ключевые слова: многослойный персептрон, алгоритм, нейронная сеть, активационная функция, весовые коэффициенты, метод обучения с учителем.


Алгоритм обратного распространения ошибки – популярный алгоритм обучения нейронных сетей (многослойных персептронов). В основу метода положен алгоритм