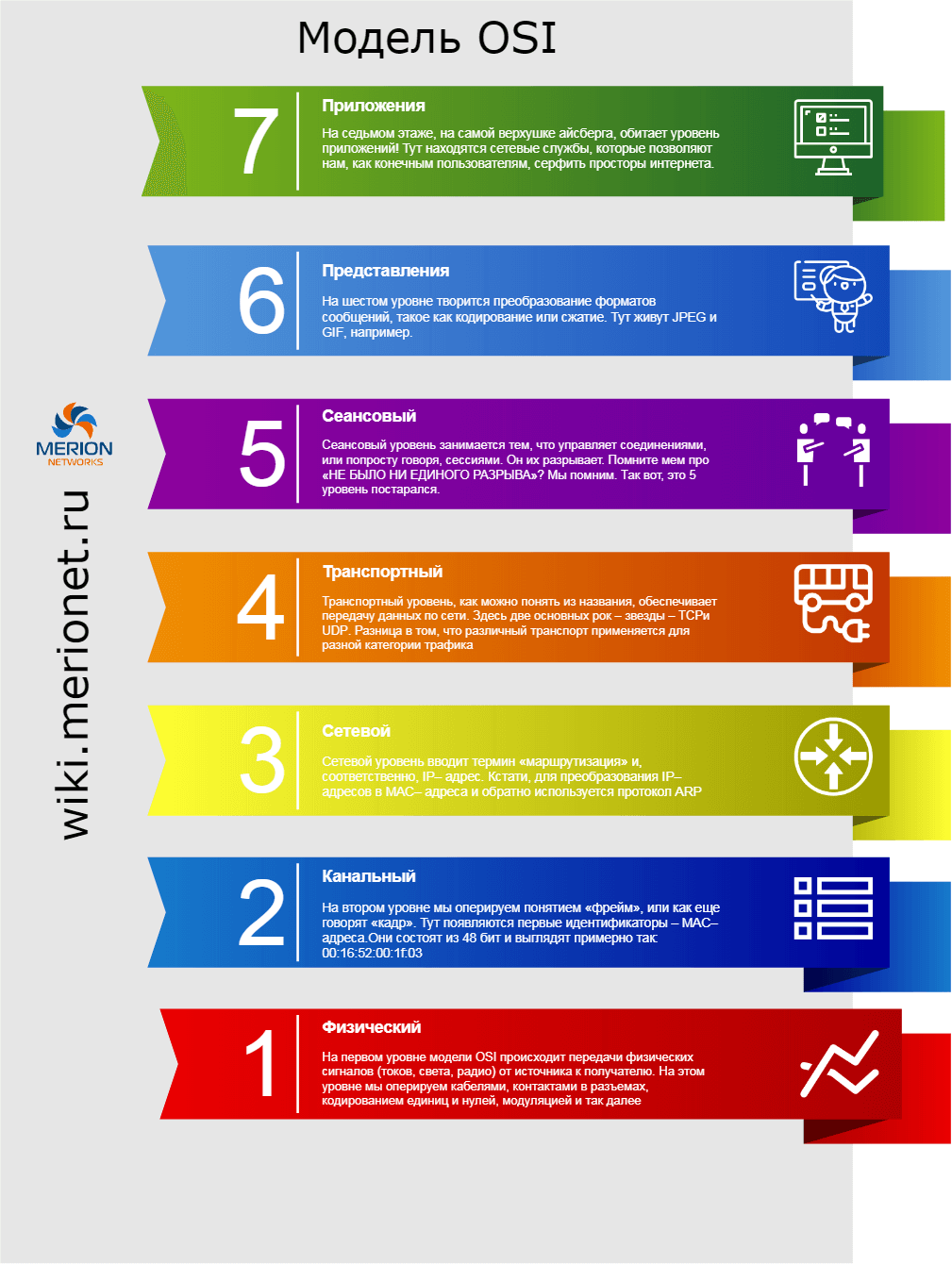
<https://www.britannica.com/technology/client-server-architecture>

Client-server architecture, architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer). Client computers provide an interface to allow a computer user to request services of the server and to display the results the server returns

<https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/microservices>

**What are microservices?**

* Microservices are small, independent, and loosely coupled. A single small team of developers can write and maintain a service.
* Each service is a separate codebase, which can be managed by a small development team.
* Services can be deployed independently. A team can update an existing service without rebuilding and redeploying the entire application.
* Services are responsible for persisting their own data or external state. This differs from the traditional model, where a separate data layer handles data persistence.
* Services communicate with each other by using well-defined APIs. Internal implementation details of each service are hidden from other services.
* Supports polyglot programming. For example, services don't need to share the same technology stack, libraries, or frameworks.



|  |  |  |
| --- | --- | --- |
| **Параметр** | **HTTP** | **HTTPS** |
| Название | Hypertext Transfer Protocol | Hypertext Transfer Protocol Secure |
| Безопасность | Менее безопасен. Данные могут быть доступны для злоумышленников | Он предназначен для предотвращения доступа хакеров к критически важной информации. Защищен атак типа Man-in The-Middle. |
| Порт | По умолчанию – 80 | По умолчанию 443 |
| Начинается на | http:// | https:// |
| Область применения | Это хорошо подходит для веб-сайтов общего назначения, таких как блоги.\* | Если на сайте нужно вводить конфиденциальную информацию, то данный протокол подходить больше |
| Защита | Нет защиты передаваемой информации. Любой, кто прослушивает трафик может получить доступ к данным | HTTPS шифрует данные перед передачей их по сети. На стороне получателя, данные расшифровываются. |
| Протокол | Работает с TCP/IP | Нет специального протокола. Работает поверх HTTP, но использует TLS/SSL шифрование. |
| Проверка названия домена | Сайтам с HTTP не нужен SSL | Для работы с HTTPS нужен SSL сертификат |
| Шифрование данных | Не использует шифрование | Данные шифруются |
| Рейтинг поиска | Не влияет на рейтинг поиска | Помогает увеличивать поисковый рейтинг |
| Скорость | Быстро\*\* | Относительно медленно |
| Уязвимость | Уязвима для злоумышленников | Лучше защищен, использует шифрование данных. |

| [**GET**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/GET)  Метод GET запрашивает представление ресурса. Запросы с использованием этого метода могут только извлекать данные.  [**HEAD**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD)  HEAD запрашивает ресурс так же, как и метод GET, но без тела ответа.  [**POST**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/POST)  POST используется для отправки сущностей к определённому ресурсу. Часто вызывает изменение состояния или какие-то побочные эффекты на сервере.  [**PUT**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/PUT)  PUT заменяет все текущие представления ресурса данными запроса.  [**DELETE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/DELETE)  DELETE удаляет указанный ресурс.  [**CONNECT**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/CONNECT)  CONNECT устанавливает "туннель" к серверу, определённому по ресурсу.  [**OPTIONS**](https://developer.mozilla.org/ru/docs/Web/HTTP/Methods/OPTIONS)  OPTIONS используется для описания параметров соединения с ресурсом.  [**TRACE**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/TRACE)  TRACE выполняет вызов возвращаемого тестового сообщения с ресурса.  [**PATCH**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PATCH)  PATCH используется для частичного изменения ресурса.  **В процессе обмена, как правило, используется два сокета — сокет отправителя и сокет получателя.**  **socket = гнездо** |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |