**PPT 8:**

- **Protocolo HTTP:**
  - **Funciona como protocolo de pedido-resposta:**
    1. Um browser envia um pedido HTTP para um servidor;
    2. O servidor retorna uma resposta HTTP para o cliente;
    3. A resposta contém dados de informação e o conteúdo requerido pelo cliente.
  - É um protocolo muito simples para tocar mensagens de texto entre duas máquinas.
- **HTTP 0.9:**
  - **Principais Objetivos:**
    - Permitir transferências de ficheiros de texto;
    - Permitir pesquisas em ficheiros hypertexto;
    - Permitir negociação de formatos de ficheiros;
    - Permitir referenciar outros servidores ao cliente.
  - **O protótipo inicial tinha algumas funcionalidades:**
    - O pedido do cliente é um conjunto de caracteres terminados com um CRLF;
    - A resposta é em HTML;
    - A ligação é terminada após a resposta.
- **HHTP 1.0:**
  - Surgiu no boom da internet para colmatar limitações anteriores:
    - Servir mais formatos de ficheiros, e não só HTML;
    - Providenciar metadados sobre os pedidos e respostas;
    - Permitir negociação de conteúdos;
    - ETC.
  - Nunca chegou a ser um standard.
  - **Métodos:**
    - **GET** -> Pede um recurso por URL;
    - **HEAD** -> Semelhante ao GET mas sem o conteúdo, apenas com os headers da resposta;
    - **POST** -> Requer ao servidor que aceite um certo conteúdo anexado.
- **HTTP 1.1:**
  - Santard que resolve ambiguidades das versões anteriores e otimiza a performance:
    - Reutilização de ligações;
    - Pedidos de bytes específicos;
    - Entre outros…
  - **Novos Métodos:**
    - **PUT** -> Requer ao servidor que aceite um certo conteúdo anexado, por URL, e se já existir é modificado;
    - **DELETE** -> Remove o recurso especificado no URL;
    - **TRACE** -> Faz eco o pedido recebido;
    - **OPTIONS** -> Retorna os métodos HTTP disponíveis;
    - **PATCH** -> Aplica alterações parciais ao recurso no URL;
    - **CONNECT** -> Encaminha o pedido.
  - **Limitações**:
    - Pedidos e respostas são sequenciais;
    - Para haver paralelismo tem de haver múltiplas ligações ao mesmo servidor;
    - Entre outras…
- **HTTP 2.0:**
  - Surgiu a partir do protocolo experimental SPDY da Google com os seguintes objetivos:
    - Melhorar a latência percebida sobre HTTP/1.1;
    - Não requerer múltiplas ligações paralelas;
    - Reter semântica do HTTP/1.1;
    - Entre outros…
  - É um protocolo binário.
  - **Ligações**:
    - Todas as comunicações são feitas numa ligação TCP;
    - Uma stream é um canal virtual dentro de uma ligação e tem um identificador;
    - Uma mensagem é uma mensagem HTTP, e pode estar dividida em vários frames.
  - **Multiplexagem**:
    - Permite enviar mensagens em várias streams paralelamente.
  - O protocolo HTTP/2.0 ainda está a ser adotado mundialmente.
- **HTTP/3:**
  - Apesar do protocolo HTTP/2 ter uma utilização mundial de cerca de 34%, já está a ser criada a versão 3;
  - Será baseado no protocolo experimental da Google (QUIC);
  - Utilizará o UDP como protocolo de transporte, invés do TCP.

**PPT 9:**

- **Nomes:**
  - **São Utilizados Para:**
    - Identificar entidades;

        - Referir localizações;
        - Partilhar recursos.
    - Sequências de caracteres para se referir a recursos.
    - Um recurso pode ter um ou muitos nomes, mas um nome apenas está associado a um recurso.
- **Endereços:**
    - Um recurso pode ter um ou mais pontos de acessos designados por endereços.
    - **Podem ser:**
        - Endereços IP; MAC ou de Memória.
- **Resolução de Nomes em Sistemas:**
    - **Nomeação Plana:**
        - Sistemas onde identificadores são nomes não-estruturados.
        - Soluções Simples:
            - **Brroadcast:**
                - Pedidos de resolução são enviados para todos os nós da rede;
                - Cada nó verifica se é o destinatário;
                - O destinatário responde com o seu endereço.
                - (Consome muitos recursos na rede)
            - **Multicast:**
                - Pedidos são enviados apenas para nós pertencentes ao grupo;
                - O destinatário responde com o seu endereço.
        - **Tabelas de Dispersão Distribuídas:**
            - Estrutura de dados que associa chaves de pesquisa a valores, onde cada nó tem apenas uma parte pequena da tabela total.
            - **Algoritmo baseado em tabelas de dispersão distribuídas:**
                - Cada nó tem um nó sucessor e um predecessor;
                - Cada nó reconhece n sucessores e n predecessores.
            - **Para resolver um nome:**
                - O nó inicial verifica se sabe o endereço do destinatário;
                - Senão, pergunta aos seus sucessores.
    - **Nomeação Estruturada:**
        - Nomes não estruturados tendem a ser pouco convenientes para redes mais complexas.
        - Nomes estruturados tendem a ser compostos por nomes simples e legíveis, organizados num namespace.
        - **Namespace:**
            - Pode agrupar os vários componentes do nome num formato tipo árvore:
                - **Raiz** -> Nome de mais alto nível;
                - **Diretórios** -> Subdividem um nome em subnomes;
                - **Folhas** -> Representam os recursos que se procura.
        - **Distribuição de Nomes:**
            - O spacename para redes de larga escala está normalmente associado hierarquicamente em **camadas lógicas**:
                - **Camada Global** -> Nós de mais alto nível e subnomes importantes;
                - **Camada Administrativa** -> Nós intermédios;
                - **Camada Final** -> Nós que representam os recursos.
        - **Resolução de Nomes Estruturados:**
            - Dado o nome de um recurso num namespace estruturado, o endereço é obtido por uma de duas formas:
                - **Iterativamente:**
                    - O servidor resolve a parte do nome que conseguir e devolve o restante ao cliente, que reencaminha o pedido para o próximo servidor.
                - **Recursivamente:**
                    - O servidor resolve a parte do nome que conseguir e reenvia o pedido a outro servidor, até completar a resolução do nome.
    - **Nomeação Baseada em Atributos:**
        - **Em certos casos, pode-se querer indexar os nomes dos recursos por atributos:**
            - Serviços que oferecem;
            - Capacidades que possuam;
            - Características que possuam.
        - **Diretórios de Serviços:**
            - Diretórios em que as entidades descrevem os seus atributos.

**PPT 10:**

- **Webservices**: Serviço criado de modo a suportar a interação entre máquinas sobre uma rede.
- **Funcionamento**: A aplicação solicita uma das operações disponíveis no webservice, e o webservice efetua o processamento e envia os dados para a aplicação.
- **Tecnologias Mais Usadas:**
    - **AJAX:**
        - Permite aos browsers obter informação de um servidor de forma assíncrona sem reler a página toda novamente.
    - **SOAP:**
        - Protocolo de mensagens que usa XML para trocar informação com um servidor.

    - **REST - Representational State Transfer:**
        - Arquitetura para interoperabilidade entre PCs na internet.
        - Servidores expõem serviços através de recursos na web.
        - É suposto ser mais leve que o SOAP e outros.
- **Em Suma:**
    - Webservices são serviços disponíveis na WEB;
    - São para ser consumidos por aplicações e não por utilizadores finais;
    - AJAX e REST são os tipos mais usados;
    - REST usa métodos HTTP para a sua semântica de interoperabilidade.