Sistemas Multimédia

TPW-II
Tecnologia e Prática da Web II

12
HTTP
(HiperText Transport Protocol)

José Joaquim Moreira

HTTP

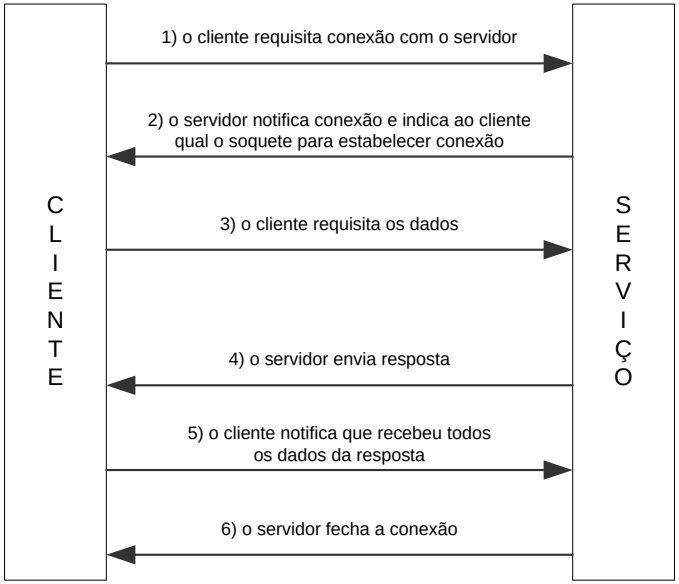
- Modelo de comunicação
- Sistemas intermediários
- Mensagens HTTP
  - Request
  - Response
- Exemplo Request-Response

HTTP

Modelo comunicação HTTP

- O servidor recebe a requisição e realiza o processo requerido, respondendo depois ao cliente com os dados resultantes, geralmente com texto HTML ou XML. Quando a transferência é concluída, a conexão entre o cliente e servidor é terminada.

- As comunicações HTTP são estabelecidas utilizando um mecanismo conhecido como *handshaking* simples. A figura seguinte ilustra esse mecanismo.



Handshake de comunicação HTTP

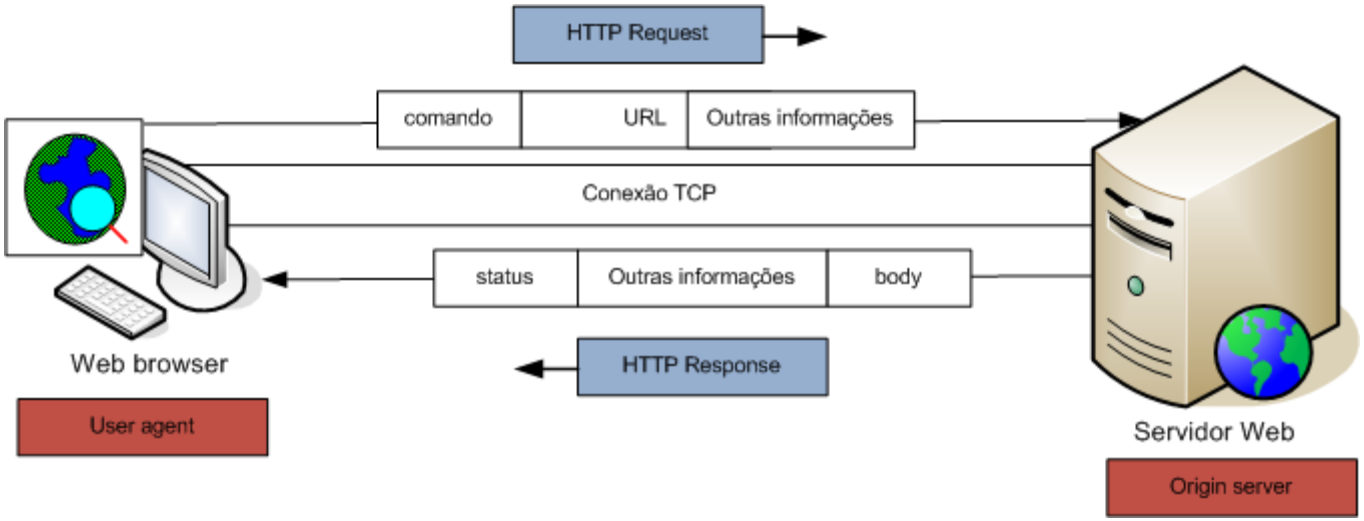
Modelo comunicação HTTP (cont)

- As comunicações são sempre da iniciativa do cliente, que faz uma conexão com o servidor que está à espera de pedidos de requisição HTTP.
- Quando o servidor recebe o pedido de conexão envia uma mensagem de notificação ao cliente, informando que está disponível e por qual *soquete* o cliente deve enviar os dados.
- O cliente cria uma conexão com o servidor utilizando o *soquete* indicado e envia o pedido, que inclui algumas informações e um cabeçalho, bem como eventuais parâmetros necessários.
- O servidor recebe o pedido e processa-o. Depois de concluído o processamento, envia os dados resultantes do processamento como resposta ao cliente.
- Durante o tempo de processamento, a conexão entre cliente e o servidor permanece activa, estando o cliente bloqueado enquanto espera a resposta.
- Quando o cliente recebe a resposta, notifica o servidor da recepção, após o qual o servidor termina a conexão com o cliente.

Modelo comunicação HTTP (cont)

- O cliente, designado na terminologia do protocolo por **user agent** (em regra um Web browser) abre uma conexão TCP com o servidor que, na terminologia do protocolo é designado por **origin server** (em regra um servidor Web).

- Em seguida, o cliente emite um **HTTP Request** consistindo num determinado comando, também designado por método (que indica qual a acção pretendida), um URL, outros parâmetros e informação adicional.



Comunicação através do protocolo HTTP. A figura sugere as interacções entre o cliente (user agent) e o servidor (origin server) no modelo de comunicação definido através do protocolo HTTP.

Modelo comunicação HTTP (cont

- O servidor tenta satisfazer o pedido do cliente e envia uma **HTTP Response.** Essa mensagem contém um código que especifica se o pedido pôde ser satisfeito com sucesso ou se se verificou algum erro, bem como informação sobre o servidor e sobre o próprio conteúdo do recurso enviado.

- O sucesso do protocolo em parte deve-se à sua simplicidade, consistindo num conjunto de comandos codificados em ASCII, ou seja, texto claro;

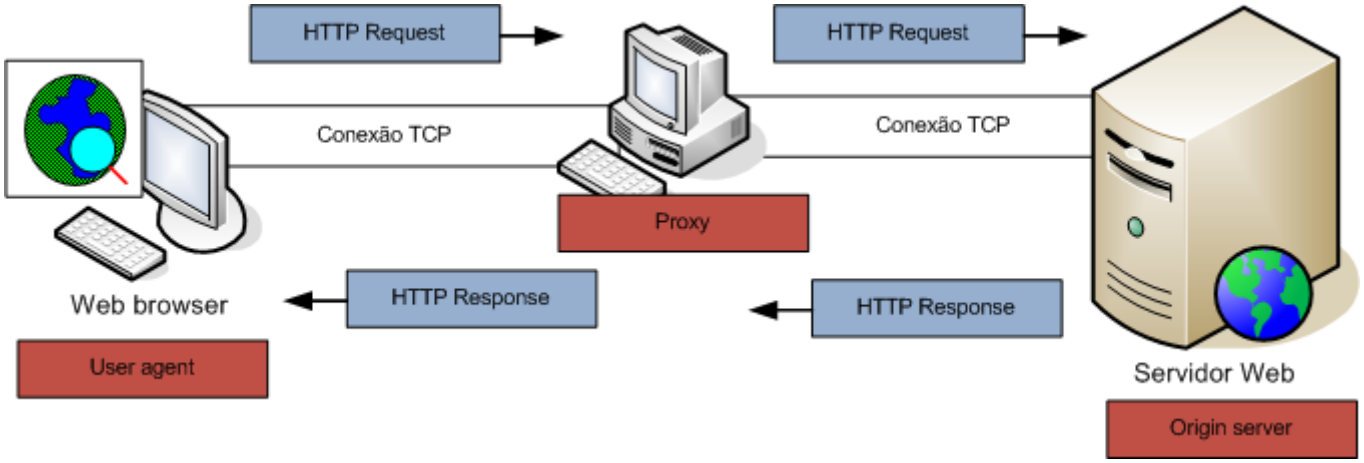
- No entanto, pode tratar dados binários, neste caso os dados serão codificados em ASCII ou tratados como anexos.

- O protocolo encontra-se estável à vários anos e todos os fornecedores de tecnologia de informação oferecem suporte nos seus produtos.

Sistemas intermediários

Proxies

- Numa ligação HTTP, um proxy é um sistema que actua como sistema intermédio funcionando como servidor para **o user agent** e como um cliente para o **origin server.** Como podem existir vários proxies numa ligação HTTP, um proxy pode actuar como servidor perante cliente e como cliente perante um servidor.



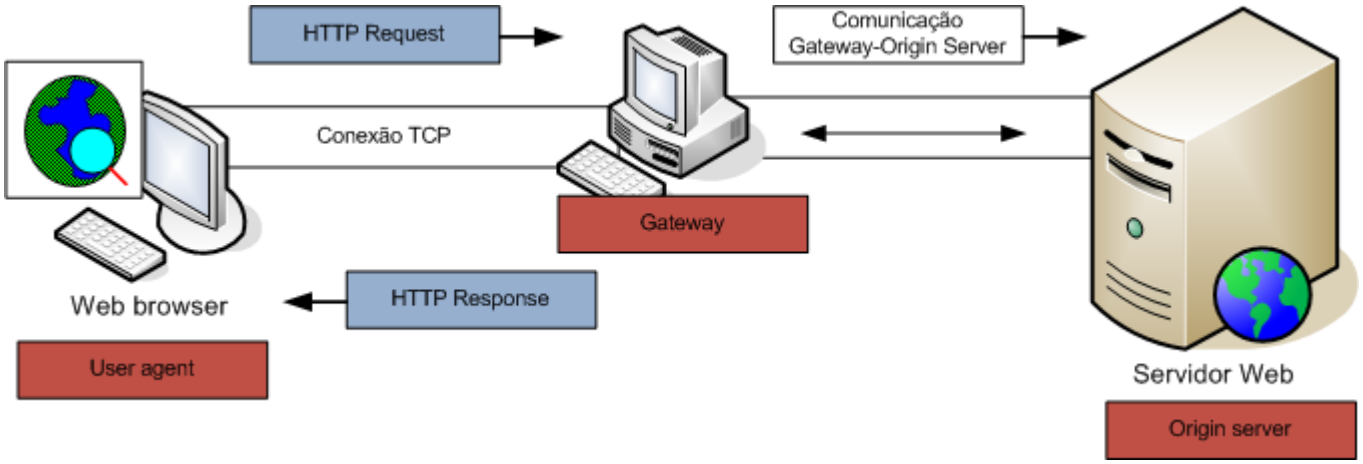
Os proxies são sistemas intermédios que funcionam como servidores para os user agents e como clientes para os origin servers.

Gateways

- No contexto de uma transacção HTTP, um **gateway** é um sistema que funciona perante o cliente como se fosse de facto o **origin server**. Esta situação pode ocorrer em vários cenários como por exemplo os seguintes.

1. O gateway funciona em conjunto com um firewall do lado do servidor, impedindo o contacto directo de pedidos exteriores com o origin server.

2. O gateway serve como um intermediário que permite a ligação a servidores que não são servidores Web. Essa função de intermediário permite que um recurso solicitado através do protocolo HTTP possa ser satisfeito por um servidor não Web.

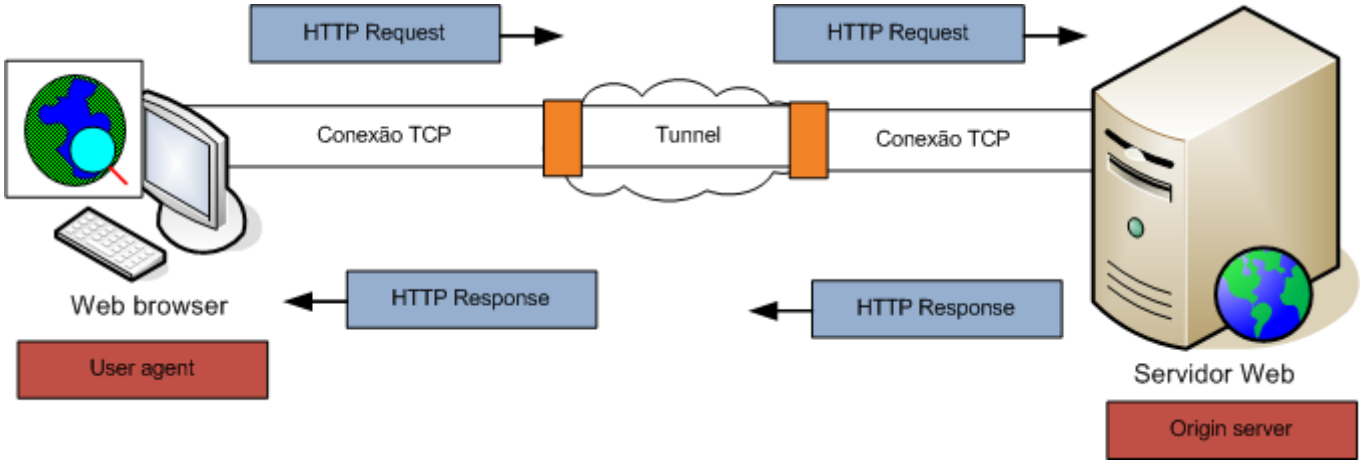


Os gateways são sistemas que se apresentam perante os user agents como se fossem os origin servers. Os gateways podem ter diferentes tipos de comunicação com os origin servers.

Tunnels

- Um tunnel é um sistema intermédio que pode ser utilizado numa ligação HTTP. No entanto, o contrário dos proxies e dos gateways, um tunnel não executa qualquer operação ao nível os **HTTP Requests** e dos **HTTP Responses**.

- Um tunnel pode ser utilizado quando se torna necessária a existência de um sistema intermédio entre o user agent e o origin server para que uma comunicação HTTP possa atravessar uma rede. O tunnel não tem de compreender o significado das mensagens, tem apenas de assegurar a comunicação dessas mensagens.

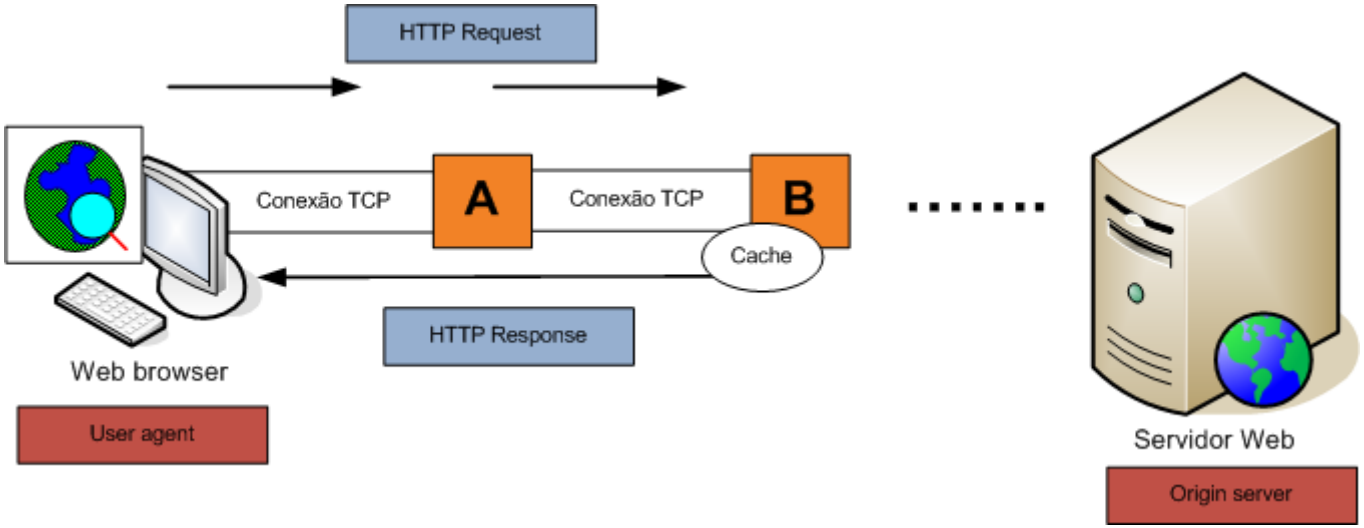


Um tunnel é um sistema intermédio que em certas circunstâncias construído para que a comunicação entre o cliente e o servidor possa atravessar uma rede.

Caches

- Uma cache é um dispositivo que pode funcionar em qualquer sistema intermédio com a excepção dos tunnel.
- No essencial, uma cache é um dispositivo de armazenamento que permite reter recursos que tenham sido previamente acedidos. Em certas circunstâncias, quando um dispositivo intermédio recebe um pedido para um recurso que se encontra na cache, não necessita de aceder ao origin server para satisfazer o pedido do cliente.
- O sistema de cache pode revelar-se eficiente em muitas circunstâncias, possibilitando que os pedidos dos clientes sejam mais rapidamente satisfeitos.
- Noutros casos, porém, o sistema de cache pode revelar-se indesejável, uma vez que pode entretanto ter havido alterações ao conteúdo do recurso solicitado e a cache conter um versão anterior desse recurso.
- Por esse motivo, existem mecanismos no protocolo HTTP que permitem ao cliente e ao servidor especificar em que condições o mecanismo de cache pode funcionar. A próxima imagem mostra o funcionamento do mecanismo de cache. O pedido do cliente não necessitou chegar ao servidor uma vez que um dos sistemas intermédios dispunha na sua cache do recurso solicitado.

Caches



Os dispositivos intermédios como os proxies e os gateways podem implementar sistemas de cache, armazenando recursos que tenham sido previamente solicitados por clientes. Este procedimento pode acelerar o acesso aos recursos. No entanto, é importante compreender que o armazenamento em cache pode levar a que sejam fornecidos recursos (por exemplo páginas Web) que entretanto tenham ficado desactualizadas. Existem mecanismos que permitem aos user agents e aos servers determinar as condições em que determinado recurso pode ser armazenado em cache. Na imagem, o sistema intermédio B possui em cache o recurso solicitado, pelo que não se torna necessário o acesso ao origin server.

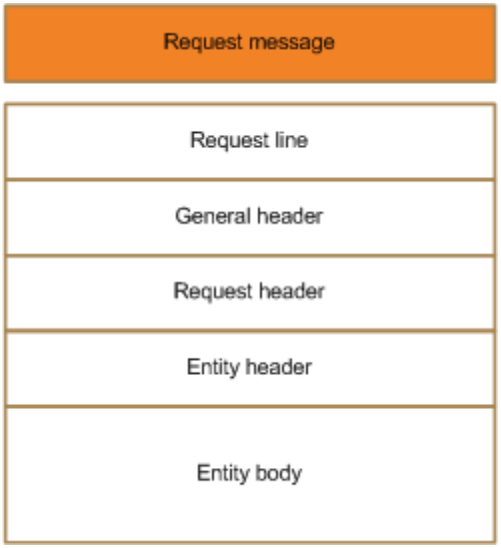
Mensagens HTTP

Estrutura mensagens HTTP

- Em HTTP existem dois tipos de mensagens:
  - **Request**
  - **Response**
- As mensagens **Request** têm origem no cliente para o servidor.
- As mensagens **Response** constituem a resposta do servidor ao pedido do cliente.

Estrutura mensagens **Request**

- Uma mensagem HTTP Request tem a seguinte estrutura:

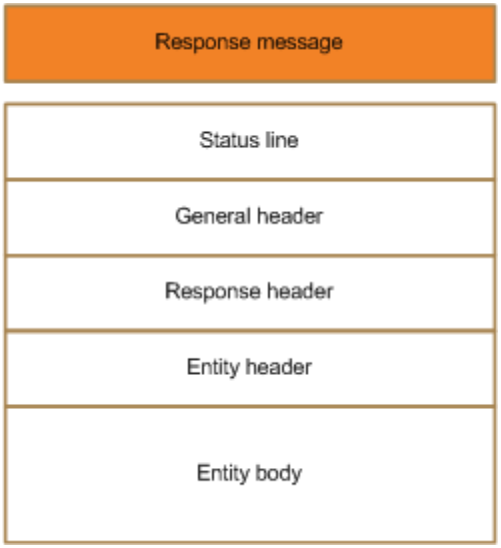


Estrutura mensagens **Request**

- **Request Line**
  - Esta linha identifica o tipo de mensagem, o comando ou método utilizado, bem como o recurso a que pretende aceder.
- **General Header**
  - Este componente é comum tanto às **Request messages** quando às **Response messages.** Contém campos que especificam certas características da comunicação.
- **Request Header**
  - Trata-se de um componente que contém informação sobre o cliente e sobre o tipo de request.
- **Entity Header**
  - Este componente é comum às mensagens **Request e Response.** Conté informação sobre o recurso e sobre o Entity body.
- **Entity Body**
  - Este componente contém o corpo da mensagem.

Estrutura mensagens **Response**

- Uma mensagem HTTP Response tem a seguinte estrutura:



Estrutura mensagens **Response**

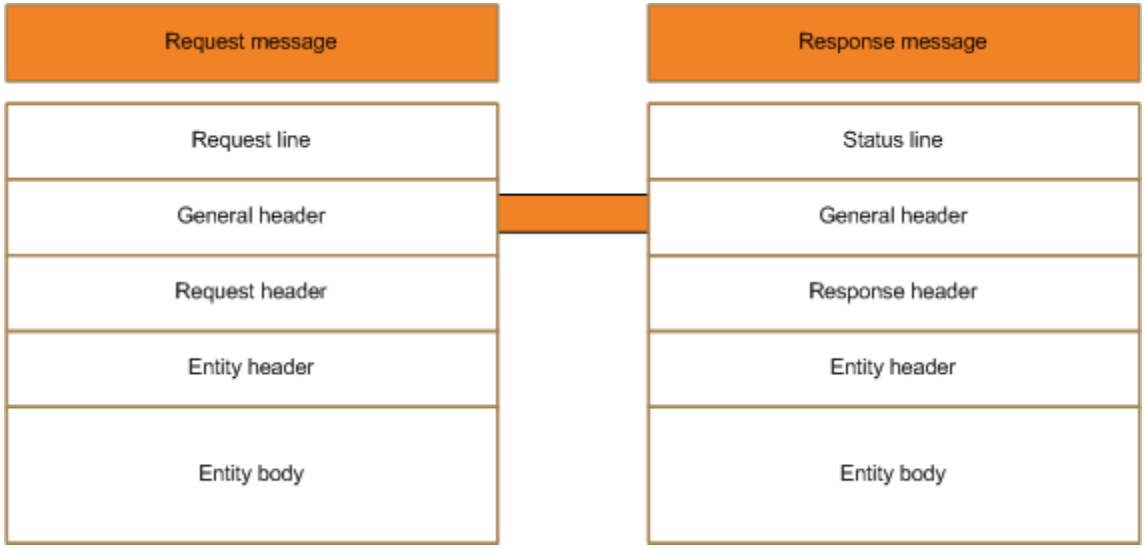
- **Status Line**
  - Esta linha contém informação de status sobre a resposta. Pode indicar que tudo correu bem e que o pedido pode ser satisfeito, ou que ocorreu algum tipo de erro ou situação que impediu que o pedido possa ser satisfeito.
- **General Header**
  - Trata-se de um componente que contém informação sobre as características da transacção – também está presente nas Request messages.
- **Response Header**
  - Trata-se de um componente que contém informação sobre a resposta dada pelo servidor.

  Entity Header
  - Trata-se de um componente que figura igualmente nas mensagens **Request** e que contém informação sobre o recurso e sobre o Entity body.
- **Entity Body**
  - Este componente contém o corpo da mensagem.

- Cada componente de uma mensagem HTTP é constituído por um conjunto de campos. Cada campo é constituído por um linha que contém o nome do campo seguido do valor do campo.

- **Os campos do General Header**

  - O General header é um componente comum às mensagens Request e Response, como se pode observar através da figura que mostra, lado a lado, a estrutura de cada um dos tipos de mensagens.



Os campos do General Header

Cache-control

- Este campo contém informações ou directivas que controlam o modo como sistemas intermédios podem ou não implementar mecanismos de caching.
- As directivas do campo Cache control são as seguintes:
  - Cachable
    - A directiva **Cachable** pode ser incluída numa mensagem Response. Através desta directiva, o servidor informa os sistemas intermédios de que um determinado recurso pode ser colocado em cache para uso futuro.
  - Max-age
    - Esta directiva informa os sistemas intermédios do limite de tempo durante o qual um determinado recurso pode ser mantido na cache e utilizado.
  - Private
    - Esta directiva pode ser incluída numa Response para indicar que a mensagem, ou partes da mensagem se destinam apenas a um determinado use r age,. e só devem ser colocadas em cache se essa cache for exclusivamente controlada por esse user agent.
  - No-cache
    - Quando incluída numa mensagem Request, esta directiva informa os sistemas intermédios de que o pedido deve ser feito ao origin serve r e não deve ser respondido por qualquer sistema de cache intermédio. Quando incluída numa mensagem Response, indica que a mensagem ou partes da mensagem não devem ser armazenadas em cache para uso futuro.

- **Os campos do General Header** (cont)
  - Connection
    - O campo Connection é utilizado para conexões TCP entre dois sistemas (não tunnel) e pode, portanto, no caso de uma mensagem HTTP não se aplicar à completa ligação entre o *user agente* e o *origin server.*
    - Este campo pode figurar quer numa mensagem Request quer numa mensagem Response. O corpo deste campo pode conter palavras-chave que apenas sei aplicam a essa conexão TCP.
    - Uma das palavras-chave contidas na especificação do protocolo é Keep-alive, significando que o emissor pretende manter aberta uma conexão TCP para além da transacção corrente.
  - Date
    - Contém a data e a hora da mensagem.
  - Forwarded
    - Este campo pode ser utilizado pelos proxies e gateways numa ligação HTTP. Cada proxy ou gateway pode acrescentar um campo Forwarded contendo o, seu URL.
  - Keep-Alive
    - Se a palavra-chave Keep-alive for incluída no campo **Connection,** o campo **Keep-Alive** pode ser utilizado para conter informação sobre o tempo em que a conexão se manterá aberta enquanto aguarda pelo próximo Request, ou o número máximo de Requests que serão permitidos nesta conexão TCP.

- **Os campos do General Header** (cont)
  - MIME-version
    - Indica que a mensagem está de acordo com o tipo MIME indicado.
  - Pragma
    - Este campo pode ser utilizado para conter directivas específicas que podem ser aplicáveis a qualquer sistema na cadeia request/response.
  - Upgrade
    - Este campo permite que os extremos da comunicação possam negociar diferentes versões e/ou protocolos para a comunicação.

Request messages

Request message

- Qualquer Request message é iniciada com uma linha (Request line) que contém o nome o comando ou método utilizado, o URL pretendido e a versão do protocolo HTTP utilizada.

  - GET www.exemplo.com/pag.htm HTTP/1.1

  - Esta Request line utiliza o método Get, tem como destino o recurso identificado pelo URL www.exemplo.com/pag.htm e utiliza a versão 1.1 do protocolo HTTP.

Métodos usados em mensagens Request

- O protocolo HTIP define vários métodos que podem ser usados nas Request lines. No entanto, penas três desses métodos são obrigatoriamente implementados: o método **Get**, o método **Head** e o método **Post**

- **GET**
  - O método GET é utilizado para aceder a recursos em servidores Web. Além do nome do método é indicado o endereço URL do recurso e a versão do protocolo HTIP usada.

- **HEAD**
  - O método HEAD é semelhante ao método GET com a diferença essencial de que o seu objectivo é apenas o de testar a existência, disponibilidade ou alterações relativamente ao recurso referido no endereço URL. Dessa forma, a mensagem de resposta ao método HEAD não contém o ficheiro em si, mas apenas informação sobre o ficheiro. Por esse motivo, a mensagem Response não conterá o Entity body.

- **POST**
  - O método POST é utilizado para enviar dados do cliente para o servidor. Este comando é utilizado, por exemplo, quando o cliente preenche um formulário numa página HTML e envia os dados para o servidor. Neste caso, o protocolo usa o método POST para enviar esses dados.

Métodos usados em mensagens Request (Cont)

- **PUT**
  - O método PUT tem igualmente por finalidade enviar dados do cliente para o servidor mas este método não é muito comum nem se encontra amplamente implementado.
- **DELETE**
  - O método DELETE poderia ser utilizado pelo cliente para determinar a eliminação de um determinado recurso no servidor. É um método cuja implementação é limitada por óbvias razões de segurança.
- **LINK**
  - O método LINK também não é amplamente suportado. A sua finalidade seria a de estabelecer uma ligação (Iink) entre um URL e outros recursos.
- **UNLINK**
  - O método UNLINK funcionaria de forma contrária à do método LINK e serviria para eliminar a ligação (Iink) entre um URL e outros recursos.

Campos do Request header

- O componente REQUEST HEADER de uma mensagem REQUEST pode conter vários campos que proporcionam informação complementar sobre a mensagem.

- **Accept**
  - Este campo pode ser utilizado para conter o tipo de *media* que a resposta ao pedido pode conter.

- **Accept-charset**
  - Indica a lista dos character sets que a resposta pode incluir.

- **Accept-encoding**
  - Os ficheiros transmitidos através do protocolo ,HTTP podem ser comprimidos ou encriptados. Este campo lista os tipos de codificação que podem ser utilizados.

- **Accept-language**
  - Pode ser utilizado para restringir as línguas aceitáveis na resposta.

- **Authorization**
  - Este campo pode ser usado pelo cliente para conter um valor que lhe permite uma autenticação perante o servidor. Esse valor é designado por credentials.

Campos do Request header (cont)

- **From**
  - Campo que pode ser utilizado para conter o endereço de e-mail da pessoa que utiliza o user agent que emite o pedido

- **Host**
  - Contém a indicação do host da Internet para o recurso solicitado.

- **If-modified-since**
  - Este campo torna o comando GET condicional. O recurso só será transferido se tiverem ocorrido modificações desde a hora e data especifica das neste campo. A utilização deste campo pode ser útil em dispositivos que utilizem cache. Esses dispositivos podem enviar periodicamente comandos GET, recebendo apenas uma pequena mensagem no caso de não ter havido alterações ao documento.

- **Proxy-authorization**
  - Este campo é utilizado para permitir que um cliente faça a sua autenticação perante um proxy, quando essa autorização é requerida.

Campos do Request header (cont)

- **Range**
  - É um campo para implementação futura. O seu objectivo é permitir que, através do método GET, um cliente possa requisitar apenas uma parte do recurso.

- **Referer**
  - O URL do recurso a partir do qual o URL da Request line foi obtido. O objectivo é o de permitir que o servidor possa criar uma lista de back-links.

- **Unless**
  - A funcionalidade deste campo é semelhante à do campo **If-Modified-Since.** As diferenças são as seguintes: a comparação baseia-se em valores incluídos no campo **Entity header** e não se restringe ao método GET.

- **User-agent**
  - Contém informação sobre o user agent que origina a mensagem **Request.**

Response messages

Response message

- Cada mensagem Response contém uma linha **Status line,** contendo um valor numérico seguido de um texto que informa sobre a natureza do código.

- Os códigos são agrupados em várias categorias e a cada categoria corresponde um intervalo de números.

Categoria	Códigos	Descrição
informational	100-199	Mensagens especificas da aplicação
Successful	200-299	A mensagem Request foi processada com sucesso
Redirection	300-399	O cliente deve executar alguma acção complementar
Cliente error	400-499	Ocorrência de problemas ao nível do cliente
Server error	500-599	Ocorrência de problemas ao nível do servidor

Alguns códigos

- **200 OK**
  - Não ocorreu qualquer erro. O pedido foi satisfeito com sucesso
- **201 Created**
  - A mensagem com o método POST foi satisfeita.
- **202 Accepted**
  - A mensagem foi aceite mas não confirma que o pedido tenha sido processado.
- **204 No Content**
  - A mensagem de REQUEST foi recebida com sucesso, mas nada existe para ser apresentado no browser cliente.
- **300 Multiple Choices**
  - O recurso requisitado na mensagem REQUEST existe em várias localizações. O servidor indica a localização preferida no campo LOCATION da mensagem RESPONSE.
- **301 Moved Permanently**
  - O recurso especificado no URL foi transferido permanentemente para outro URL, pelo que pedidos subsequentes devem ser dirigidos a esse novo URL. O novo URL é indicado no campo LOCATION da mensagem.

Alguns códigos (cont)

- **302 Moved Temporarily**
  - O recurso solicitado foi temporariamente movido para outro URL. Esse novo URL é especificado no campo LOCATION. Subsequentes pedidos devem continuar a ser dirigidos ao mesmo URL.
- **304 Not Modified**
  - Trata-se da resposta a um comando GET condicional. Indica que o recurso não foi modificado desde a data especificada no campo If-Modified-Since na Request.
- **400 Bad Request**
  - A mensagem REQUEST não pode ser correctamente interpretada pelo servidor.
- **401 Unauthorized**
  - Não autorizado o acesso ao recurso especificado. Pode resultar da necessidade de autenticação ou de uma autenticação não válida.
- **403 Forbidden**
  - O servidor não autoriza o acesso ao recurso pretendido.
- **404 Not Found**
  - O URL especificado não foi encontrado no servidor.

- Alguns códigos (cont)
  - **500 Internal Server Error**
    - Ocorrência de um erro interno no servidor.
  - **501 Not Implemented**
    - O tipo de REQUEST não é suportado pelo servidor.
  - **502 Bad Gateway**
    - Significa que o gateway ou proxy recebeu uma resposta inválida do servidor contactado
  - **503 Service Unavailable**
    - O servidor está temporariamente impedido de satisfazer o pedido solicitado.

Campos do Request header

- O componente RESPONSE HEADER de uma mensagem RESPONSE pode conter vários campos.
- **Location**
  - Pode ser utilizado para indicar um URL que constitua uma localização alternativa para o recurso solicitado.
- **Proxy-authenticate**
  - Um dos códigos de erro é proxy authenticated required (o que significa que o cliente tem previamente de ser autenticado perante um proxy). Este campo conterá o esquema de autenticação e os parâmetros requeridos.
- **Public**
  - Apresenta uma lista dos métodos não standard que são suportados pelo servidor.
- **Retry-after**
  - Quando o cliente recebe um código de erro do tipo service unavailable, este campo pode especificar durante quanto tempo se espera que o serviço esteja indisponível
- **Server**
  - Identifica o software utilizado no origin server.
- **www-authenticate**
  - Este campo é incluído quando o servidor emite uma mensagem do tipo Unauthorized O campo conterá o esquema de autenticação requerido bem como os parâmetros necessários.

Entities

- Quer numa mensagem Request quer numa mensagem Response existe um Entity header e um Entity body.
- **Entity header**
  - O Entity header pode conter vários campos opcionais que caracterizam a parte Entity body da mensagem.
  - Allow
    - Uma mensagem Response pode conter um status code de erro Method Not allowed significando que o método invocado na Request não é suportado para o recurso pretendido. Nesse caso, o campo Allow no Entity header pode conter uma lista dos métodos suportados para esse recurso.
  - Content-encoding
    - Indica que tipo de codificação foi utilizada para o conteúdo do recurso.
  - Content-language
    - Indica a linguagem ou linguagens naturais para o recurso.
  - Content-length
    - Contém o tamanho do corpo da mensagem em bytes.

Entities (cont)

- **Entity header (cont)**

  - **Content-type**
    - Identifica o tipo de *media* do recurso.
  - **Title**
    - O texto que constitui o title (título) da entidade.
  - **Transfer-encoding**
    - Especifica o tipo de transformação aplicado ao corpo da mensagem para efeitos de transmissão. A opção comum é um procedimento que consiste em separar o corpo do Entity body em blocos com respectivas labels que são transmitidos separadamente.

Entities (cont)

- **Entity body**
  - O Entity body é a informação do recurso identificado pelo URL. Trata-se de uma sequência de bits que podem representar vários tipos de conteúdo: texto, imagens, sons ou vídeo, por exemplo.
  - A interpretação a dar aos bytes do corpo da mensagem é determinada pelo conteúdo dos campos do componente Entity header.

Exemplo Request-Response

- Request – Response
  - Exemplo:
    - No servidor http://www.exemplo.pt exista o seguinte página HTML: introWebServer.html

```
<html>
  <body>
      <p>olá mundo!</p>
  </body>
</html>
```

Request – Response

Pedido HTTP (request)

```
GET /introWebServer.html HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-
        shockwave-flash, application/vnd.ms-excel, application/vnd.ms-
        powerpoint, application/msword, application/x-pdf */*
Accept-Language: en-gb,pt;q=0.5
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
    1.1.4322; .NET CLR 2.0.50727)
Host: www.exemplo.pt:79
Connection: Keep-Alive
Cookie: infoview_userCultureKey=useBrowserLocale
```

Request – Response

- Resposta HTTP (response)

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
X-Powered-By: ASP.NET
Date: Thu, 25 May 2006 14:02:51 GMT
Content-Type: text/html
Accept-Ranges: bytes
Last-Modified: Thu, 25 May 2006 14:02:12 GMT
ETag: "cd3bdd2380c61:ba9"
Content-Length: 54
<html>
<body>
<p>olá mundo</p>
</body>
</html>
```

Request – Response

- Embora estes valores variem de acordo com o browser que utilizarmos e com o servidor Web que responde a este pedido HTTP, muito do conteúdo será sempre igual:

- Pedido – request

  - **GET /introWebServer.html HTTP/1.1**
    - estamos a requisitar através do protocolo http versão 1.1 o ficheiro introWebServer.html que está na raiz do servidor

- Resposta - response

  - **HTTP/1.1 200 OK**
    - o pedido é válido (200 OK) e o conteúdo segue em baixo. Como podemos ver depois de mais um conjunto de dados aparece finalmente o conteúdo do ficheiro html.

- Independentemente de estarmos a falar de páginas dinâmicas ou páginas estáticas este será sempre o fluxo que o pedido/resposta entre o servidor e o browser irá provocar.

Exemplo Request-Response páginas dinâmicas

Request – Response

- A origem do conteúdo enviado pelo servidor Web numa resposta a um pedido HTTP, como já vimos anteriormente, pode ser:
  - estática - se vier directamente de um ficheiro já existente no servidor
  - dinâmica - se for criada dinamicamente por outro programa, script ou API chamado pelo servidor.
- No caso de uma página dinâmica, o pedido, depois de recebido, é processado pelo servidor Web que vai criar dinamicamente o conteúdo que depois será enviado para o cliente.
- As páginas dinâmicas têm a vantagem de poderem ser programadas, ou seja usando alguma linguagem de programação (que dependendo do servidor Web pode ser PHP, Java, Perl, VB.NET, C#, etc…) podemos criar programas que correm no servidor Web, que eventualmente acedem a base de dados e cujo resultado é enviado para o browser.

Request – Response

- Exemplo de página dinâmica - introWebServer.asp

```
<html>
<body>
        <%  for i=1 to 10
              Response.Write("<p>olá mundo</p>")
            next  %>
</body>
</html>
```

Se no browser tentarmos aceder a este ficheiro (http://www.exemplo.pt/introwebserver.asp), a sequência pedido / resposta iria produzir os seguintes comandos:

Request – Response

- Exemplo de página dinâmica - introWebServer.asp (cont)

- Pedido

GET /introwebserver.asp HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*

Accept-Language: en-gb,pt;q=0.5

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.50727)

Host: localhost:79

Connection: Keep-Alive

Cookie: infoview_userCultureKey=useBrowserLocale; ASPSESSIONIDQSRCCSAS=KJLFNNNCNHKODOIOCIICJFBA

Request – Response

- Exemplo de página dinâmica - introWebServer.asp (cont)
- Resposta

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.1
Date: Thu, 25 May 2006 14:20:34 GMT
X-Powered-By: ASP.NET
Content-Length: 198
Content-Type: text/html
Cache-control: private
<html>
<body> olá mundo olá mundo olá mundo olá mundo olá mundo olá mundo olá mundo olá mundo olá mundo olá mundo
</body>
</html>

- Como podemos ver o pedido/resposta de um ficheiro estático ou de um ficheiro dinâmico gera fluxos de informação praticamente iguais, isto é, na Web a informação que circula é essencialmente a mesma. a diferença é que um ficheiro dinâmico tem que ser primeiro processado pelo servidor Web.