

JOÃO PAULO GRACIANO GONÇALVES BARROCAS

Acesso Offline a Conteúdos Dinâmicos

Orientador: Prof. Dr. José Quintino Rogado

**Universidade Lusófona de Humanidades e Tecnologias
Departamento de Engenharia Informática e Sistemas de Informação**

Lisboa

2013

JOÃO PAULO GRACIANO GONÇALVES BARROCAS

Acesso Offline a Conteúdos Dinâmicos

Dissertação apresentada para a obtenção do Grau de Mestre em Eng.^a Informática e Sistemas de Informação no Curso de Mestrado em Engenharia Informática e Sistemas de Informação, conferido pela Universidade Lusófona de Humanidades e Tecnologias.

Orientador: Prof. Doutor José Quintino Rogado

Universidade Lusófona de Humanidades e Tecnologias
Departamento de Engenharia Informática e Sistemas de Informação

Lisboa

2013

O motivo que convencerá a maioria das pessoas a comprar um computador para a casa será vinculando essa pessoa à uma rede nacional de comunicações. Somente estamos na etapa inicial do que será um avanço realmente notável para a maioria das pessoas, tão notável quanto o telefone.

Steve Jobs (24/2/1955 – 5/10/2011)

AGRADECIMENTOS

É minha intenção agradecer a todos aqueles que estiveram ao meu lado durante este tempo, oferecendo-me o que podiam, para me dar todo o ânimo necessário à sua conclusão.

Agradeço também a todos os outros que me dificultaram a vida neste período, pois foi aí que fui arrancar toda a força necessária para terminar este trabalho, nesta fase difícil da minha vida.

A Qiong (Joan) Luo do Departamento de Ciências e Engenharia da Computação da Universidade de Ciências e Tecnologias de Hong Kong pela amabilidade de por à disposição os seus trabalhos na área, que levaram a uma ideia mais concreta do desenvolvimento desta solução.

Um agradecimento especial ao Professor José Rogado pela paciência que teve para me ouvir e aconselhar durante todo o mestrado.

RESUMO

A grande maioria dos sites hoje em dia usa a criação de páginas dinâmicas para fornecer informação. A criação dessas páginas utiliza geralmente informação armazenada em bases de dados. Hoje em dia o recurso à rede tornou-se comum e necessário em todas as áreas, desde o lazer ao trabalho. Em condições adversas de acesso à rede, tais como localizações remotas, veículos em movimento, catástrofes naturais, guerras, etc., e em situações em que o acesso a recursos existentes na rede seja indispensável, a sua indisponibilidade pode causar graves problemas.

Esta tese apresenta um modo alternativo de funcionamento que, independentemente de falhas do acesso à rede, permite continuar a usar alguns recursos essenciais por esta fornecidos, criando para isso um sistema autónomo local constituído por um servidor de páginas e um *proxy*, que se mantêm a trabalhar independente do acesso à rede.

The vast majority of sites uses dynamic pages creation to provide information. The creation of those pages usually uses information stored in databases. Nowadays, the access to the Internet has become common and necessary in all areas, from pleasure to work. Under adverse access conditions, such as in remote locations, in moving vehicles, during natural disasters, war, etc., in situations where access to resources on the Internet is mandatory, its absence can cause innumerable problems.

This thesis presents an alternative solution that creates an autonomous working system, through which essential resources can still be accessed, regardless of the availability of Internet access.

ÍNDICE

AGRADECIMENTOS	1
RESUMO	2
ÍNDICE	3
1.Introdução	7
<i>1.1 Motivação</i>	9
<i>1.2 Contribuição</i>	9
<i>1.3 Estrutura da Tese</i>	10
2. Arquitectura geral de um servidor de páginas	11
<i>2.1 Páginas estáticas</i>	11
<i>2.2 Páginas dinâmicas</i>	12
3. Proxies	13
<i>3.1 Proxy Ftp e Proxy Web</i>	14
<i>3.2 Forward e Reverse Proxy</i>	15
<i>3.3 Caching</i>	16
3.3.1 <i>Caching em Proxies Web</i>	16
3.3.2 <i>Páginas criadas previamente</i>	16
3.3.3 <i>Caching por contexto</i>	17
3.3.4 <i>XML</i>	18
3.3.5 <i>Proxies granulares</i>	19
4. Replicação e Sincronização	20
<i>4.1 Replicação e sincronização em Bases de Dados</i>	20
<i>4.2 Replicação e sincronização de Ficheiros</i>	21
5 Solução desenvolvida	22
<i>5.1 Esquema Funcional</i>	23

5.2	<i>Servidor Original</i>	24
5.3	<i>Servidor Local</i>	25
6	Biblioteca ICu	26
6.1	<i>Ficheiro de configuração</i>	27
6.2	<i>Entrada (Entry Point)</i>	28
6.3	<i>Gestor de Módulos</i>	30
6.4	<i>Gestor de Filtros Iniciais</i>	31
6.5	<i>Gestor de Web Services</i>	32
6.6	<i>Gestor de Serviços Internos</i>	33
6.7	<i>Pré-processamento da página</i>	34
7	Sistema de Licenças	36
7.1	<i>Número de Série</i>	36
7.2	<i>Chave de Identificação</i>	36
7.3	<i>Validação de uma Instalação da Biblioteca ICu</i>	37
7.3.1	<i>Criação da Assinatura</i>	38
7.3.2	<i>Cadeia de Validação de Numero de Série</i>	38
7.4	<i>Validação de uma instância de uma Biblioteca ICu</i>	38
7.5	<i>Web Services</i>	40
7.5.1	<i>WSBeginRegister</i>	40
7.5.2	<i>WSEndRegister</i>	40
7.5.3	<i>WSCheckSignature</i>	41
8	Módulo AProxy	42
8.1	<i>Configuração do Módulo</i>	42
8.2	<i>Adição de um novo Protocolo de Comunicação</i>	43
8.2.1	<i>Esquema Funcional</i>	45
8.2.2	<i>InitComm()</i>	46
8.2.3	<i>BeginComm()</i>	47

8.2.4 WriteComm()	47
8.2.5 ReadComm()	48
8.2.6 EndComm()	48
<i>8.3 Sincronização de Sistemas</i>	48
8.3.1 Ficheiro de Envio	49
8.3.2 Sincronização de Ficheiros	52
8.3.3 Sincronização da Base de Dados	53
8.3.3.1 Equivalência de Identificadores das Bases de Dados	54
8.3.3.2 Modificação da Estrutura das tabelas	55
8.3.3.3 Equivalência de Chaves Primárias entre Sistemas	55
8.3.3.4 Criação da Lista de Registos a Sincronizar	56
<i>8.4 Administração do Sistema</i>	56
8.4.1 – Configuração de Sincronismo do Sistema de Ficheiros	57
8.4.2 – Configuração de Sincronismo do da Base de Dados	58
9 Servidor Local	61
Conclusão	65
Bibliografia	68
APÊNDICES	I
APÊNDICE I	II

Índice de Figuras

Figura 1 – Arquitectura Geral de um Servidor de Páginas.....	11
Figura 2 - Esquema Geral da Solução	23
Figura 3 - Diagrama Funcional do Cliente.....	25
Figura 4 – Diagrama Funcional da Biblioteca.....	26
Figura 5- Validação de uma instalação da Biblioteca ICu	37
Figura 6 - Validação de um Servidor Original	39
Figura 7 - Esquema Funcional do uso de Protocolos de Comunicação	45
Figura 8 - Escolha do site	57
Figura 9 - Escolha do tipo de configuração.....	57
Figura 10 - Definição de exportação de ficheiros	57
Figura 11 - Definição de exportação de tabelas	59
Figura 12 - Instalação do Servidor Local / Site.....	61
Figura 13- Interface de configuração	63

1.Introdução

A globalização decorrente da generalização do acesso à Internet introduziu significativas mudanças nas sociedades. Essas mudanças são visíveis na alteração que muitos comportamentos tiveram, pelo simples facto da distância e do tempo de comunicação se terem reduzido drasticamente.

Hoje em dia assiste-se a uma dependência crescente relativa ao suporte informático para a realização de tarefas que, na sua ausência, deixam de ser possíveis de realizar. Se no que respeita aos aspectos recreativos ou de lazer esta falta de acessibilidade poderá não incutir pressão nos agentes, já a nível empresarial e profissional as suas consequências poderão acrescentar níveis de pressão adicionais, podendo ter consequências negativas na produtividade.

Apesar da evolução enorme que se tem assistido na cobertura territorial, no aumento da qualidade e da largura de banda da infra-estrutura necessária ao suporte da comunicação, existem ainda grandes superfícies do planeta em que essa comunicação é insuficiente ou mesmo impossível. Se adicionarmos a esta situação casos em que o acesso à Internet ainda não é generalizado, tais como aviões, comboios e autocarros, ou em casos de acidentes naturais, apagões, etc, perduram ainda muitas situações em que o utilizador fica impossibilitado de aceder à Internet.

Por outro lado, o sentido actual da evolução do software acentua cada vez mais a diluição entre os aplicativos e a rede (Internet), tornando o utilizador cada vez mais dependente do acesso à infra-estrutura. A necessidade de acesso a uma grande diversidade de informação e a necessidade de aceder a bases de dados de grande volume, como por exemplo, aplicações baseadas em sistemas SIG, criou uma nova direcção no desenvolvimento aplicativo.

No sentido de fazer face a situações em que o acesso à rede não está disponível de forma constante são desenvolvidos muitos aplicativos que permitem continuidade na utilização, mesmo sem acesso à rede. A implementação deste tipo de aplicativos implica um esforço extra de desenvolvimento e manutenção de versões diferenciadas, devido à sua dependência relativamente aos diversos sistemas operativos e plataformas onde vão correr. Por este motivo não é vulgar encontrar no mercado aplicativos que permitam uma utilização transversal nas variadas plataformas utilizadas actualmente, especialmente no mercado de aparelhos móveis.

Como consequência, nos casos em que a actividade dos utilizadores está dependente da continuidade de serviço, o respectivo rendimento pode ficar seriamente comprometido em caso de falha de acesso à rede.

Hoje em dia a utilização de browsers para aceder à informação é generalizada e comum. Esta tecnologia é no momento actual a que se encontra mais desenvolvida no suporte às variadas plataformas existentes no mercado, quer de sistemas operativos, quer da diversidade de aparelhos existentes.

O desenvolvimento de aplicativos para esta plataforma é menos oneroso, os seus tempos de desenvolvimento são reduzidos e a necessidade de criar versões diferenciadas por aparelhos e sistemas operativos é reduzida. A manutenção de correcções e a criação de novas versões é bastante simplificada pois é feita nos servidores de forma centralizada.

Uma questão importante relacionada com este projecto prende-se com a filosofia que se pretende no alcance do mesmo. Não é intenção deste trabalho desenvolver um sistema distribuído com um mecanismo de tolerância a falhas, ou seja, desenvolver um sistema que consiga repor o mais rapidamente possível a informação em todos os seus nós, mas sim, o desenvolvimento de um sistema que consiga trabalhar offline aproveitando o restabelecimento irregular das comunicações para efectuar o sincronismo entre o cliente e o respectivo servidor da aplicação.

Com esta premissa de base, não se enquadra no âmbito deste estudo qualquer exigência no sentido do sistema reagir com rapidez às alterações efectuadas por utilizadores remotos. O seu objectivo último é permitir que, em caso de condições adversas de comunicações, se consiga continuar a trabalhar com aplicações Web e que o seu sincronismo se efectue logo que possível, dentro das limitações de um quadro de comunicações bastante irregulares ou com latências elevadas.

O objectivo desta dissertação é de propor um sistema que permita a utilização de um recurso Web, mesmo que a ligação à rede seja inexistente ou irregular, e fornecer uma solução autónoma que utilize preferencialmente sistemas e conceitos já existentes e comprovados, criando para o efeito uma base de trabalho para o desenvolvimento de aplicativos *Web Based*, de modo a que estes possam continuar a ser utilizados quando não é possível garantir uma ligação permanente à rede.

1.1 Motivação

O crescente aumento da cobertura móvel de alto débito e o seu constante aumento de largura de banda estão a direccionar-nos para o uso massivo da rede.

O aumento da disponibilidade geográfica e da diminuição dos tempos de acesso à rede permitiu aumentar o uso do computador para efeitos de trabalho remoto. Na realidade o número de pessoas a trabalhar remotamente tem aumentado nos últimos anos [1]. Muito deste trabalho tem características residenciais, em que o acesso à rede pode, em certas situações, ser de fraca qualidade ou inexistente.

Por outro lado, o acréscimo de mobilidade permitiu ampliar o automatismo de tarefas profissionais a novas situações, sendo as vendas um caso típico. Com efeito, grande parte das empresas hoje em dia recorre ao uso de comunicações móveis para o suporte ao trabalho, no campo, dos seus vendedores. Estes, muitas vezes, e por falta de acesso à rede em certos locais, têm que deixar para mais tarde a actualização da sua actividade, aumentando assim o tempo necessário à execução das suas tarefas.

Embora já existam trabalhos [2] no sentido da criação de um standard para redes de comunicação tolerantes a falhas ou latência na comunicação, no entanto, ainda não se encontram no mercado soluções finais, não proprietárias, prontas a ser utilizadas pelo analista e pelo programador para o desenvolvimento de aplicações tolerantes a falhas de comunicação.

1.2 Contribuição

Quando existem situações em que é necessário o trabalho online remoto, e as condições de acesso à rede não são fiáveis, a solução passa normalmente por desenvolver software específico para os dispositivos usados. A solução aqui proposta permite que, para situações em que o acesso à rede é instável ou inexistente, se possam desenvolver aplicações *Web Based* utilizáveis em várias plataformas, evitando desenvolver aplicações específicas para cada dispositivo ou sistema operativo.

Este facto permite assim a redução de custos na implementação de sistemas de trabalho remoto e ao mesmo tempo aumentar o tempo de ciclo de vida desse sistemas, pois o desenvolvimento deixa de ficar dependente das evoluções que os sistemas operativos e os dispositivos possam sofrer.

Espera-se também, que com a elaboração deste projecto, seja possível proporcionar uma ferramenta que automatize ao máximo o processo de replicação e sincronização, quer de dados, quer de ficheiros, fornecendo assim ao desenvolvimento uma ferramenta que permita

construir aplicações com suporte de continuidade de acesso, em condições extremas de acesso à rede.

Por outro lado, mesmo que o acesso à rede não seja um problema, o facto de transferir o processamento de páginas para um servidor local, residente na mesma máquina onde é gerado o pedido, irá permitir diminuir a carga dos respectivos servidores. Neste caso, como o conteúdo recebido pelo requisitante é produzido na própria máquina, o tempo de resposta irá diminuir significativamente, tornando a operação do aplicativo mais rápida e sem atrasos significativos para o utilizador.

1.3 Estrutura da Tese

O primeiro capítulo introduz o tema da Dissertação, os seus objectivos e sintetiza as suas principais contribuições. O segundo capítulo descreve sucintamente o funcionamento de um servidor de páginas. O terceiro capítulo fornece uma explicação geral sobre *proxies* e os seus vários tipos. O quarto capítulo fornece uma explicação geral sobre sincronização e replicação de dados e ficheiros. O quinto capítulo descreve a solução desenvolvida, que é constituída basicamente pelo desenvolvimento de uma biblioteca em PHP. Os capítulos subsequentes são dedicados à descrição de cada uma das partes mais importantes para o desenvolvimento do sistema. Assim, no sexto capítulo descreve-se a estrutura base de funcionamento e acesso à biblioteca, no sétimo capítulo descreve-se o sistema de licenciamento que permite a identificação de todas as instalações, no oitavo capítulo descreve-se o ***AProxy***, módulo encarregado de gerir todo o processamento de replicação e sincronização da biblioteca. Finalmente, no nono capítulo é descrita a instalação e configuração da solução, num sistema local. Finalmente a conclusão, que reporta o que foi feito e aponta direcções de trabalho futuro.

2. Arquitectura geral de um servidor de páginas

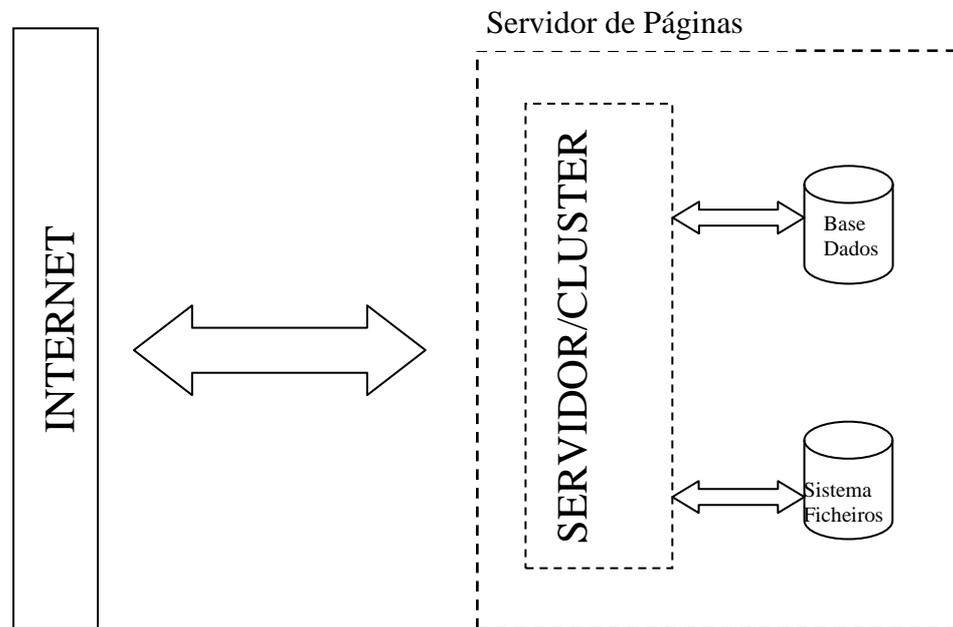


Figura 1 – Arquitectura Geral de um Servidor de Páginas

Um servidor de páginas é um conjunto de aplicativos cuja função é, através de uma rede, servir conteúdos a utilizadores finais. A estrutura geral de um servidor de páginas encontra-se representada na Figura 1.

Ao receber um pedido da rede o servidor recorre à informação contida em ficheiros armazenados ou a programas que podem fazer uso de informação contida numa base de dados, e assim devolver a informação pedida pelo utilizador remoto.

Fundamentalmente, um servidor de páginas utiliza dois métodos para a construção de uma página:

2.1 Páginas estáticas

O requisitante indica um endereço (URI) ao servidor, que corresponde directamente a um ficheiro que reside no sistema de ficheiros do servidor. Este vai analisar o tipo de ficheiro, e de acordo com a sua extensão, entende que o requisitante está a pedir um ficheiro que não necessita de modificações, isto é, no ficheiro encontra-se incluída toda a informação necessária para a sua visualização. O ficheiro é então enviado ao requisitante, sem que o servidor de páginas modifique o seu conteúdo.

2.2 Páginas dinâmicas

É o caso em que o ficheiro requerido pelo utilizador corresponde a um ficheiro de código que necessita de passar por um interpretador de uma linguagem, o caso mais comum, ou ser executado como um ficheiro binário pelo sistema operativo de suporte.

Na sua generalidade, esse código faz uso de acesso a dados que se encontram armazenados em bases de dados. Neste caso, o processamento de mais um aplicativo além de sobrecarregar os processadores com mais processamento, atrasa também a entrega da página ao utilizador que a requisitou.

No entanto, e devido às exigências cada vez maiores de qualidade do serviço prestado, o recurso à utilização de páginas dinâmicas para criação de conteúdos tem aumentado muito, originando a necessidade de serem criados sistemas cada mais complexos para que os tempos de acesso à informação se mantenham dentro de valores aceitáveis para o utilizador.

Como o *layout*, e o conteúdo destes ficheiros, são construídos na execução do código existente na página, e como geralmente os conteúdos tem proveniência em informação armazenada em bases de dados, informação essa que pode ser alterada a cada instante, dois pedidos do mesmo URI não têm que produzir necessariamente o mesmo resultado.

O problema mais crítico de resolver quando se quer desenvolver um sistema que permita o acesso a um site que esteja sempre acessível, mesmo em casos de falha de rede, reside exactamente na operação de geração das páginas dinâmicas, pois, como normalmente estas vão pesquisar ou alterar informação residente em bases de dados, informação essa que pode estar em constante alteração, a sincronização da informação entre os vários sistemas implica necessariamente um processamento acrescido.

3. *Proxies*

Um *proxy* é um componente que colocado entre um ou mais servidores e os seus respectivos utilizadores finais, permite efectuar o *caching* da informação de modo a que enquanto não for alterada, essa informação só seja enviada uma vez pelo servidor onde tem origem. Um *proxy* deve responder a dois requisitos básicos:

- Servir como acoplador entre dois sistemas diferentes, isolando os dois sistemas um do outro e permitindo apenas a passagem do tipo de informação esperado.
- Efectuar as operações necessárias sobre a informação de modo a reduzir o tráfego entre a origem da informação e os utilizadores finais.

Ao receber um pedido de um utilizador final, dirigido a máquinas que se encontram do outro lado do *proxy*, este vai analisar o pedido e decidir se pode satisfazer esse pedido com a informação que tem armazenada.

Caso não possa satisfazer esse pedido com a informação armazenada, o pedido do utilizador é direccionado para a máquina a que era dirigido inicialmente. Esse pedido, pode levar informação suplementar, acrescentada pelo *proxy*, ou simplesmente ser uma cópia do pedido original.

Em qualquer dos casos, o endereço do requisitante que efectuou o pedido original, pode ser escondido à máquina à qual é destinado o pedido, sendo só dado a conhecer o endereço do *proxy* que está a efectuar essa mesma requisição. Esta característica faz “esconder” todas as máquinas que estão a usar o *proxy*, fazendo parecer aos servidores requisitados, que apenas uma máquina está a efectuar todos os pedidos.

Esta característica faz com que um *proxy* seja muitas vezes confundido com uma *gateway*. No entanto, e embora o funcionamento de ambos seja muito semelhante, existem diferenças que os tornam necessariamente distintos:

- Uma *gateway* é um componente que, quando usado, tem que ter a sua existência obrigatoriamente conhecida pelo sistema utilizador, enquanto tal requisito já não é obrigatório para um *proxy*.
- Uma *gateway* é um componente passivo, isto é, serve apenas de tradutor de endereços entre as duas redes a que está ligada.
- Um *proxy* tem uma funcionalidade de armazenamento local da imagem (*caching*) dos objectos para o qual foi desenhado.
- Um *proxy* pode ter uma função de filtro que pode inibir, segundo condições predefinidas, a passagem de informação.

Caching é o processo que, usando uma ou mais caches, permite aos *proxies* fornecer a um utilizador final a informação solicitada, sem que para isso tenha que vir a ser requisitada outra vez ao servidor original.

A cache aplicada aos *proxies*, é uma área de armazenamento temporário interna, onde os dados mais solicitados são armazenados para um acesso mais rápido. O processo de *caching* mais usado pelos *proxies*, pode ser descrito pelas seguintes operações:

a) O pedido de informação é recebido. É verificado se a informação requisitada se encontra guardada na cache local. Se sim, é imediatamente devolvida ao requerente a informação pedida.

b) Caso a informação não se encontre guardada na cache local, esta é pedida ao servidor original. Quando a informação é recebida do servidor original, é primeiramente devolvida ao utilizador requerente, sendo de seguida guardada na cache local.

c) A informação que vai ficando armazenada na cache local, vai sendo eliminada de acordo com um algoritmo de expiração. De acordo com a sua idade, tamanho e histórico de acesso, são usados dois algoritmos simples: LRU (Least Recently Used) e LFU (Least Frequently Used). LRU remove os documentos existentes em cache há mais tempo, enquanto o LFU remove os documentos menos pedidos.

Num *proxy* é possível filtrar as transacções dos utilizadores. Um *proxy* pode inibir o acesso a serviços, quer de utilizadores individuais, quer de computadores ou de domínios. Alguns servidores *proxy* permitem filtros específicos, tais como:

- Decidir os pedidos que serão respondidos e os que serão recusados.
- Especificar as URL ou a URL Mask (Máscara de rede) dos servidores da rede que não serão atendidos pelo *proxy*.
- Especificar que protocolos podem ser usados, por exemplo, pode ser permitido a um cliente que possa fazer pedidos HTTP, mas impedi-lo de utilizar o protocolo FTP.

Um *proxy* pode também ser concebido para trabalhar num só conjunto de protocolos, por exemplo, o protocolo ftp (File Transfer Protocol) ou o http (Hypertext Transfer Protocol), para falar nos mais usados.

3.1 Proxy Ftp e Proxy Web

Um *Proxy* FTP é o mais eficiente em termos de *caching*, pois envolve basicamente operações de manutenção remota e transferência de ficheiros. O protocolo ftp tem, entre outras características, a capacidade de poder, na maior parte das implementações, saber a data da última actualização de qualquer ficheiro remoto, informação preciosa em qualquer sistema

de *caching*. Os algoritmos de *caching* deste tipo de *proxies* continuam a ser os mesmos que são usados na generalidade dos *proxies*, no entanto, o algoritmo de expiração de ficheiros da cache é geralmente substituído pelo seguinte algoritmo:

a) O *proxy* recebe o pedido de um ficheiro. Vai verificar a sua existência e data da última actualização no servidor requerido. Caso o ficheiro já não exista, remove-o da cache local, se necessário, e retorna o erro correspondente.

b) Se o ficheiro existir no servidor remoto e a data da última actualização não for fornecida pelo servidor de FTP da origem, são usados então os algoritmos de expiração, já referidos atrás, para o *proxy* decidir se o ficheiro remoto deverá ser actualizado na cache local.

c) Se a data da última actualização do ficheiro for fornecida pelo servidor FTP da origem, essa data é confrontada com a data em que a cópia local foi guardada em cache. Caso a data seja anterior à que está guardada na cache local, a imagem guardada na cache é devolvida ao requisitante do ficheiro, caso contrário o ficheiro é pedido ao servidor de FTP da origem e guardado com a nova data de actualização na cache local. Finalmente o ficheiro pedido é devolvido ao requisitante.

Os *Proxies Web*, ou *Proxy http*, são em tudo semelhantes aos *Proxies FTP*, tendo que entrar em conta com dois problemas acrescidos:

- A data da última alteração de um ficheiro raramente pode ser conhecida pelo cliente.
- A frequência com que esses ficheiros são alterados é muito elevada.

No contexto do protocolo *http*, o ficheiro é chamado página e contém toda a informação, em *HTML*, necessária para construção e desenho da página num browser.

3.2 Forward e Reverse Proxy

Um *proxy* pode ser colocado em qualquer lado da cadeia que vai desde o requisitante da informação, até ao fornecedor da mesma. Dependendo do sítio onde é colocado e da sua função, pode ser classificado como *Forward Proxy* ou *Reverse Proxy*.

Um *Forward Proxy* é normalmente colocado à entrada de um nó de rede mais rápido, como por exemplo, à entrada de uma rede local para minimizar o tráfego, de modo a reduzir quer os custos de transmissão, quer o tempo de acesso dos utilizadores à informação.

Estes *proxies* também podem funcionar de um modo transparente, isto é, os portos *TCP-IP* são redireccionados pelas *gateways* directamente para os *proxies*, sem que os utilizadores disso de apercebam, o que os torna ideais para serem usados por *ISP's* (*Internet*

Service Provider) que, por exemplo, oferecem opções comerciais de tráfego ilimitado, para assim poderem diminuir os custos com a transmissão de dados.

Também podem estar fisicamente ligados à mesma rede local a que estão ligados os servidores de páginas, caso em que estamos a falar de um *Reverse Proxy*. Todas as ligações provenientes da Internet são então redireccionadas e passam por esse servidor *proxy*. Isto permite, entre outras coisas, que o *proxy* possa:

- Equilibrar e encaminhar o tráfego para um conjunto de servidores, apresentando-se na Internet como se de um só servidor se tratasse;
- Fornecer uma ou mais camadas de segurança, protegendo melhor os servidores de páginas de ataques externos;
- Proporcionar *caching* imediato de páginas estáticas;
- Comprimir toda a informação antes de ser enviada para a Internet.

Destes exemplos, se infere que os *proxies* são um componente importantíssimo na Internet, e como tal, a rapidez de acesso à informação na WEB depende em muito das suas prestações.

3.3 Caching

3.3.1 Caching em Proxies Web

Uma página dinâmica, como já foi dito, é uma página que é criada no servidor no momento em que é requisitada pelo utilizador. Ao contrário, a página estática é criada apenas uma vez, gravada como um ficheiro e acedida posteriormente pelo servidor de páginas. Uma página estática muda muito pouco ou mesmo nada durante o seu tempo útil de vida.

Um dos maiores problemas no *caching* de páginas dinâmicas reside no pouco tempo em que uma página está disponível sem sofrer nenhuma alteração, quer no seu *layout*, quer no seu conteúdo. Acresce a isto, a possibilidade de uma mesma página poder ser diferente no seu *layout* e conteúdo, mesmo quando se utiliza o mesmo URL, por exemplo, no caso de o site ser orientado para o utilizador e o controlo do utilizador ser feito internamente no servidor de páginas [4].

3.3.2 Páginas criadas previamente

Uma das primeiras tentativas de resolver o *caching* de páginas dinâmicas foi conduzida por Jim Challenger, Arun Iyengar e Paul Dantzig [3]. Esta solução proponha que a criação de páginas dinâmicas não fosse efectuada pelo servidor de páginas na altura sua

requisição, mas sim por um *daemon*¹ independente do servidor de páginas. Seria assim da responsabilidade deste *daemon* a criação de novas páginas cada vez que existissem alterações, quer ao seu conteúdo quer ao seu *layout*. Deste modo as páginas dinâmicas seriam tratadas pelos *proxies* e pelos servidores de páginas como páginas estáticas. Este método, ainda hoje usado em sites de informação geral, é bastante eficiente, especialmente se o site tem grande afluxo de utilizadores e o tempo médio útil de vida da informação, isto é, o tempo médio em que uma página existe sem ser modificada, é superior em média a um minuto [3]. Esta solução, embora resultando muito bem em páginas que não mudam frequentemente, não trás nenhuma vantagem para aquelas que mudam o seu conteúdo frequentemente, mesmo que seja numa pequena parcela. Também não tem vantagens nas páginas personalizadas, isto é, páginas que diferem, por exemplo, no nome de utilizador visualizado na página, mantendo tudo o resto igual.

3.3.3 *Caching* por contexto

Uma outra solução, apontada por Qiong [5] e Pai Cao [13] vai no sentido da instalação de *applets* nos servidores, nos *proxies* e nos destinatários finais. Esta aproximação, muito complexa, requer a interceptação do fluxo da informação e a sua respectiva interpretação. Assim, de cada vez que é detectado numa página dinâmica instruções de acesso a informação proveniente de SGBD, essas instruções são substituídas por uma marca cuja finalidade é a de ser reconhecida mais tarde pelos *proxies* e/ou pelos utilizadores finais como uma zona de *caching*. Na altura de reconstruir a página no *proxy* ou no utilizador final, será então da responsabilidade das *applets* comunicarem com os servidores iniciais para verificar se existem ou não dados mais recentes e assim finalizar a página. Embora este algoritmo seja eficiente ao nível do *caching*, quer nos *proxies* quer nos *browsers*, tem alguns problemas relacionados com o desempenho, no diz respeito aos servidores de páginas. O processo inicial de interpretação da página corre no servidor de páginas, por este motivo os servidores de páginas, além das tarefas normais, têm de correr mais um processo por cada utilizador que efectua um pedido. Esta operação, como é óbvio, tem um grande impacto no tempo de resposta do servidor.

Ikram Chabbouh e Mesaac Makpangou [6] propuseram uma solução em que uma página dinâmica é analisada sob dois contextos distintos: o seu *layout* e o seu conteúdo, devendo ser identificadas as partes da página que podem ser consideradas estáticas (*layout*) e

¹ Um *daemon* é um programa informático que corre em segundo plano, sem ser controlado directamente pelo utilizador e que é activado por um evento ou condição específica.

aquelas que podem ser consideradas como dinâmicas (conteúdo). Nesta perspectiva, e após o primeiro envio da página, bastaria aos *proxies* passarem a requisitar ao servidor de páginas apenas as partes da página consideradas dinâmicas. O único problema desta solução reside na presunção de que o *layout* das páginas muda muito poucas vezes durante o seu tempo útil de vida, o que não é verdade em muitos casos, e o maior exemplo disso continua a ser o caso das páginas orientadas para o utilizador, em que a mesma página pode aparecer completamente diferente a dois utilizadores distintos.

Mas esta não foi a única direcção de investigação seguida no sentido de aproveitar a independência existente entre o *layout* de uma página e o seu conteúdo.

3.3.4 XML

*XML (Extensible Markup Language)*² é um subconjunto do *SGML (Standard Generalized Markup Language* - um standard internacional criado em 1986, pela W3C, para a definição da descrição de estruturas e conteúdos de diferentes tipos de documentos electrónicos). O seu objectivo é o de permitir que citações genéricas de *SGML* sejam servidas, recebidas e processadas na *Web* de modo idêntico ao que é possível com o *HTML*.

No essencial, quando surge um pedido para servir uma determinada página, o servidor retorna a página como se de uma página estática se tratasse. Essa página contém apenas o *layout* da mesma. Incluso deverá vir a informação necessária para a referência da página, ou páginas, que contêm os dados. Esta aproximação leva a que a página inicial (*layout*) seja tratada como uma página estática. Essas páginas (*layouts*) contêm, no seu texto, o endereço ou endereços das páginas que contêm os dados a inserir no respectivo *layout*, para que o *proxy* ou o *browser* procedem ao respectivo pedido desses ficheiros para completar a página com os dados recebidos.

O XML acabou por gradualmente deixar de se usar na criação de páginas devido basicamente à não inclusão no modelo inicial de um processo de *cache* do ficheiro de dados em XML, e à grande necessidade de tempo de processamento para refrescar o ecrã de cada vez que o conteúdo muda. O XML provou ser um método bastante mais eficaz em outras aplicações, como por exemplo na importação e exportação de dados, do que no processo de geração de páginas dinâmicas;

² O *XML* é um protocolo público e não um protocolo proprietário de alguma entidade. As especificações da versão 1.0 foram aceites pelo W3C, como uma recomendação em 10 de Fevereiro de 1998

3.3.5 *Proxies granulares*

T.T. Tay and Y. Zhang [7], continuando numa direcção semelhante à de Ikram Chabbouh, e Mesaac Makpangou [6], foram mais longe no conceito proposto pelos primeiros e sugeriram um processo que consiste na criação de uma estrutura em árvore constituída pelos objectos incluídos na página sujeita ao processo de *caching*. Após uma análise à página identificam-se os objectos sujeitos a sofrer alterações e, dependendo do resultado, cria-se uma política de *caching* que vai pedir apenas ao servidor de origem as alterações que esses objectos tenham tido entretanto. Os resultados obtidos nos testes durante um período de 87 dias, a que corresponderam 162053 objectos, mostram que o tamanho médio das actualizações correspondeu a mais ou menos 20% do total da informação presente em todas as páginas pedidas [7].

Este método, embora diminuindo em cinco vezes o volume total de informação, continua a não ser muito eficiente com as páginas dinâmicas que mudam também o seu *layout* para o mesmo endereço. Neste caso, para se poder ter um controlo eficiente sobre as modificações ao *layout*, a árvore criada por cada página, passaria a ter que considerar que qualquer das suas folhas poderia ser actualizada a qualquer altura, o que obriga a uma verificação, a cada pedido, de toda a página.

4. Replicação e Sincronização

Os termos replicação e sincronização, muitas vezes usados como sinónimos, são, no entanto, duas operações de naturezas diferentes.

Quando se fala em replicação estamos a referir-nos a uma operação de duplicação de uma entidade, por exemplo, se nos estivermos a referir a bases de dados, estamos a falar da criação de uma cópia exacta de uma base de dados, de uma tabela, etc., procedimento vulgar em casos de sistemas sujeitos a uma elevada taxa de acesso, permitindo assim dividir esses acessos pelas várias bases de dados replicadas ou copiadas, diminuindo assim o tempo de resposta dos sistemas.

Quando se fala em sincronização, estamos a referir-nos ao processo de reposição de um estado comum entre duas ou mais entidades, por exemplo, duas bases de dados, em que uma contem toda a informação existente (base de dados original), e outra que apenas contem a informação necessária para satisfazer pedidos específicos das aplicações que a ela acedem. Neste caso a informação armazenada nas duas bases de dados não têm que ser necessariamente a mesma.

4.1 Replicação e sincronização em Bases de Dados

“Replication is the process of copying and maintaining database objects in multiple databases that make up a distributed database system. Changes applied at one site are captured and stored locally before being forwarded and applied at each of the remote locations. Replication provides user with fast, local access to shared data, and protects availability of applications because alternate data access options exist. Even if one site becomes unavailable, users can continue to query or even update the remaining locations.”
[8]

A replicação e/ou sincronização de bases de dados, são técnicas utilizadas em sistemas com elevado numero de acessos, pois ao distribuir por várias bases de dados os pedidos, o tempo de acesso de cada pedido é assim reduzido [9].

Existem muitos estudos sobre replicação e sincronização de bases de dados, mas na sua generalidade cada implementação exige uma aproximação particular [10][10]. No entanto, outros estudos vão no sentido de uniformizar os métodos de replicação e sincronização [10][11][12] para que motores de bases de dados heterogéneos possam comunicar entre si directamente e procederem às operações de replicação e/ou sincronização.

Hoje em dia as bases de dados existentes já incluem na sua distribuição original mecanismos de replicação e sincronização. Embora esta situação pareça à partida uma mais-

valia, na realidade prende o desenho de um sistema distribuído de gestão de base de dados a um só fabricante, pois os métodos postos à disposição são proprietários e exclusivos de fabricante para fabricante.

Esta situação é talvez a mais limitativa à implementação de um sistema que necessite de manter cópias sincronizadas entre vários sistemas, especialmente quando esses sistemas são desenhados e mantidos por entidades distintas.

4.2 Replicação e sincronização de Ficheiros

A replicação e sincronização de ficheiros, é o método de distribuir cópias de ficheiros por vários sistemas, mantendo a consistência dos mesmos através dos vários sistemas [14].

Existem muitas aplicações que usam a replicação e sincronização de ficheiros, desde os sistemas operativos que proporcionam acesso a ficheiros offline, sistemas distribuídos, computação paralela, etc. O tipo de aplicações mais conhecidas e utilizadas são as aplicações que fazem a partilha de ficheiros por um universo de utilizadores em várias máquinas (*peer-to-peer*) [15][16].

No âmbito deste desenvolvimento as necessidades de implementação de um sistema complexo de replicação e sincronização de ficheiros não é crítica, pois a necessidade básica é apenas a da consistência dos ficheiros entre o servidor original e os vários clientes.

Os ficheiros de aplicativos de um site normalmente não são sujeitos a alterações frequentes após o site ser posto em produção, logo, a necessidade de implementar um sistema complexo para manter a consistência dos mesmos seria contra procedente.

5 Solução desenvolvida

O objectivo do trabalho desenvolvido foi focado na facilidade e transparência da implementação em aplicações já a correr em servidores de páginas.

Usou-se no seu desenvolvimento o maior número de aplicativos existentes e comprovados no mercado, permitindo assim uma maior fiabilidade da solução proposta.

Nessa perspectiva, e para não tornar muito complexa a solução final, optou-se por reduzir o grupo alvo, quer de sistemas operativos, quer de linguagens usadas, no desenvolvimento de aplicações WEB.

Sendo assim, a solução foi desenvolvida especificamente para sistemas cliente a correr o sistema operativo Windows, embora a sua migração para sistemas Linux não seja complexa, e para aplicações WEB que usem o PHP como linguagem base.

Na parte do servidor de páginas foi desenvolvido uma biblioteca em PHP, para o suporte às aplicações que pretendam usar o método desenvolvido para proporcionar um acesso off-line à respectiva aplicação.

A biblioteca suporta, como motor de base de dados, o mysql.

O desenvolvimento desta biblioteca pautou-se pelo cuidado em manter o mais transparente possível a sua presença para que a sua utilização em aplicações já existentes, seja a menos intrusiva possível.

Na parte do cliente, sendo a parte mais sensível de toda solução, teve-se o cuidado de tentar encontrar a solução que menos interacção com o utilizador tivesse. Também aqui, procurou-se usar como base aplicativos bastantes divulgados e com um tempo de utilização no mercado que permitisse assegurar a maior fiabilidade possível.

A escolha recaiu sobre o Apache como servidor local e *proxy*, o mysql como base de dados local e o PHP a linguagem local das aplicações.

O servidor de páginas WEB da apache.org, é o aplicativo mais usado na rede há um considerável número de anos, tendo provado ser uma plataforma fiável, bem assim como o mysql. Como ambos existem em numerosos sistemas operativos, a sua escolha garante um bom grau de portabilidade da solução aqui proposta, que poderá ser um caso a equacionar caso se pretendam suportar outras tecnologias.

5.1 Esquema Funcional

A solução desenvolvida encontra-se esquematizada na figura seguinte:

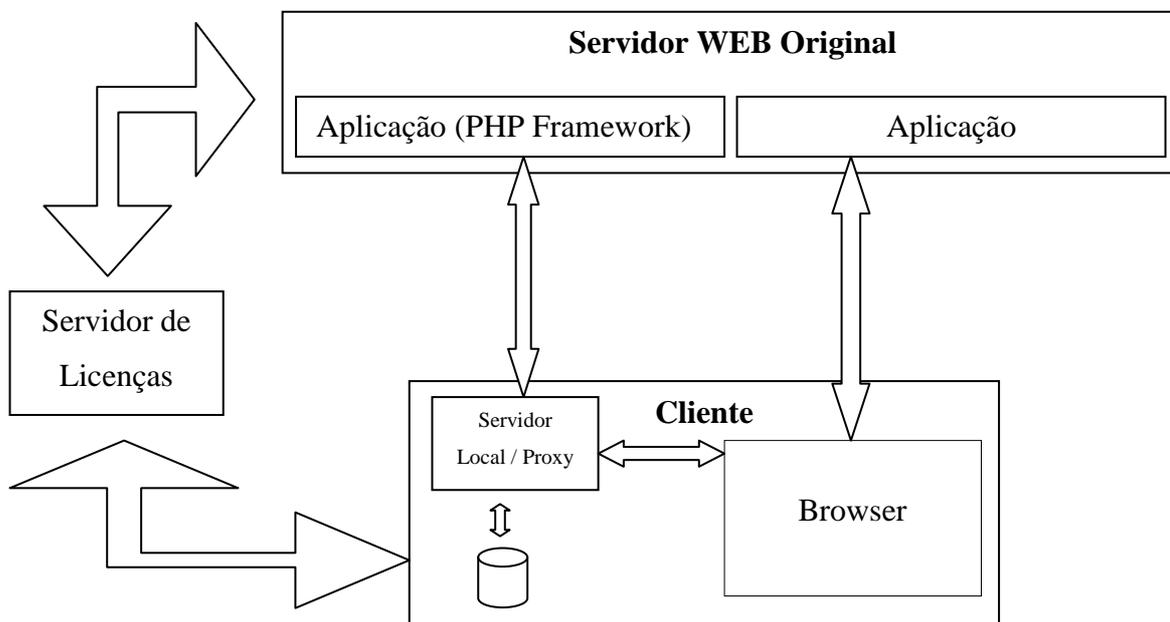


Figura 2 - Esquema Geral da Solução

Como representado na Figura 2, um servidor de páginas não necessita de alterações para poder suportar o sistema proposto. Uma aplicação que deseje utilizar o sistema apenas necessita de incluir, no código fonte de PHP, a referência à biblioteca que gere todo o processo.

Quando um cliente contacta um servidor de páginas que suporta esta solução, a biblioteca trata de reconverter o código da página requisitada de modo a que, quando esta chegar ao browser do cliente, sejam executadas um conjunto de operações que vão detectar se o cliente já tem instalado um sistema local.

A existência de um servidor local de páginas vai permitir continuar a utilizar o site mesmo que a ligação à rede seja inexistente. Caso este não esteja ainda instalado, o sistema procede à sua instalação após confirmação do utilizador.

Concluído o passo anterior o código inserido pelo servidor de páginas original vai detectar se o site requisitado se encontra instalado no servidor local de páginas. Caso não esteja, prepara-o para a instalação do site remoto, de modo a que este possa passar a ser acedido localmente.

Uma vez redireccionado o endereço do site remoto, através do *proxy* local, para o servidor local de páginas, o Sistema Local passa a efectuar todas as operações necessárias para que a cópia local do site e respectiva base de dados estejam sempre sincronizadas com o site original.

Concluído este procedimento o site passará sempre a ser acedido localmente, tornando assim possível que, mesmo desligado da rede, o utilizador possa continuar a utilizá-lo, cabendo ao Sistema Local todo o processamento necessário para manter a coerência entre o servidor local de páginas e o servidor original do site.

5.2 Servidor Original

Para que o sistema funcione no servidor original não é necessário nenhuma alteração ao servidor de páginas original. A alteração a efectuar, quando se pretender aplicar a um sistema já desenvolvido, consiste na inclusão de uma linha de código em PHP no início de cada página que serve de entrada directa ao site.

Exemplificando, supondo um sistema já desenvolvido que contenha as seguintes páginas como páginas directas de entrada:

- index.php
- abc.php
- def.php

As alterações a efectuar em cada página é a inclusão do seguinte código, logo ao início de cada zona do PHP

```
// Call ICu Library  
require_once("<pathToICuLibrary>" . "/ICu/ICu.inc");
```

onde *pathToICuLibrary* é uma variável que indica a localização da biblioteca.

É da responsabilidade desta biblioteca fornecer todos os serviços necessários para que um site possa ser replicado e sincronizado de modo a ser acedido offline. É também responsabilidade desta biblioteca todo o processamento, na parte do cliente, para criar a cópia do site e mantê-lo sincronizado.

5.3 Servidor Local

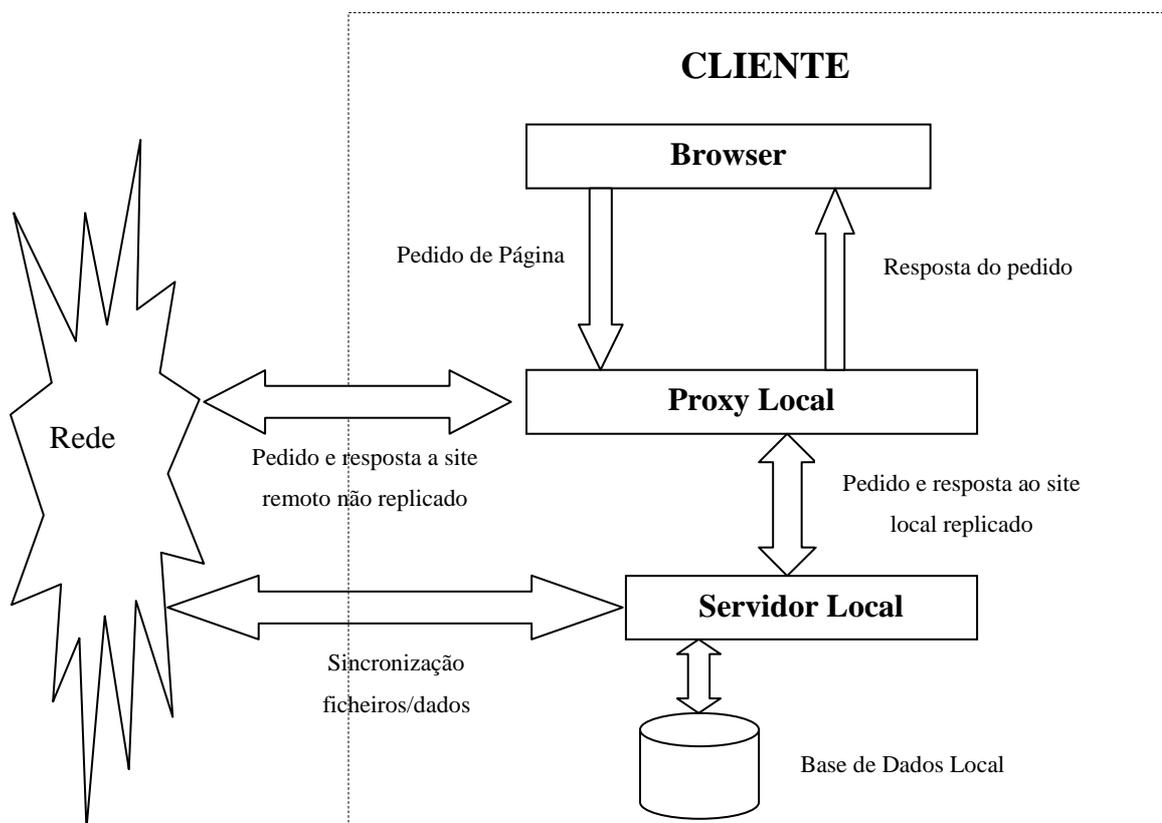


Figura 3 - Diagrama Funcional do Cliente

O Servidor Local (Servidor Páginas + *Proxy* + SGBD) é um conjunto de aplicativos que podem ser instalados na máquina do cliente ou num servidor a que o cliente tenha acesso na rede local. Mais uma vez usou-se o *apache* como servidor de páginas e *proxy*, o PHP como interpretador e o *mysql* como o sistema de gestão de base de dados.

Após estar ligado ao *proxy* local todos os pedidos gerados no browser passarão a ser processados por este e direccionados, de acordo com o URL pedido, para o processamento normal do *proxy* ou para o servidor de páginas local, se se tratar de um site replicado.

No fundo é como se o cliente estivesse directamente ligado ao servidor do site que está a aceder. A replicação inicial do site e as consequentes sincronizações são da responsabilidade do Servidor Local. Sendo assim os sites passam a ser acedidos localmente, e mesmo que a rede tenha falhas o acesso por parte do utilizador mantêm-se sempre disponível.

6 Biblioteca ICu

A biblioteca ICu, desenvolvida em PHP, é responsável por todo o processamento de suporte ao site e da sua respectiva sincronização com os clientes. Esta biblioteca foi desenvolvida de forma modular para permitir, de um modo simplificado, a adição de módulos que permitam acrescentar serviços e funcionalidades à biblioteca.

Embora necessite apenas de se acrescentar uma linha de código para que o sistema fique a funcionar, esta biblioteca foi criada com o objectivo de fornecer, com algumas restrições referidas mais à frente, suporte para novas aplicações, pondo à disposição do programador um conjunto de funcionalidades adicionais.

A figura seguinte representa o diagrama funcional da biblioteca.

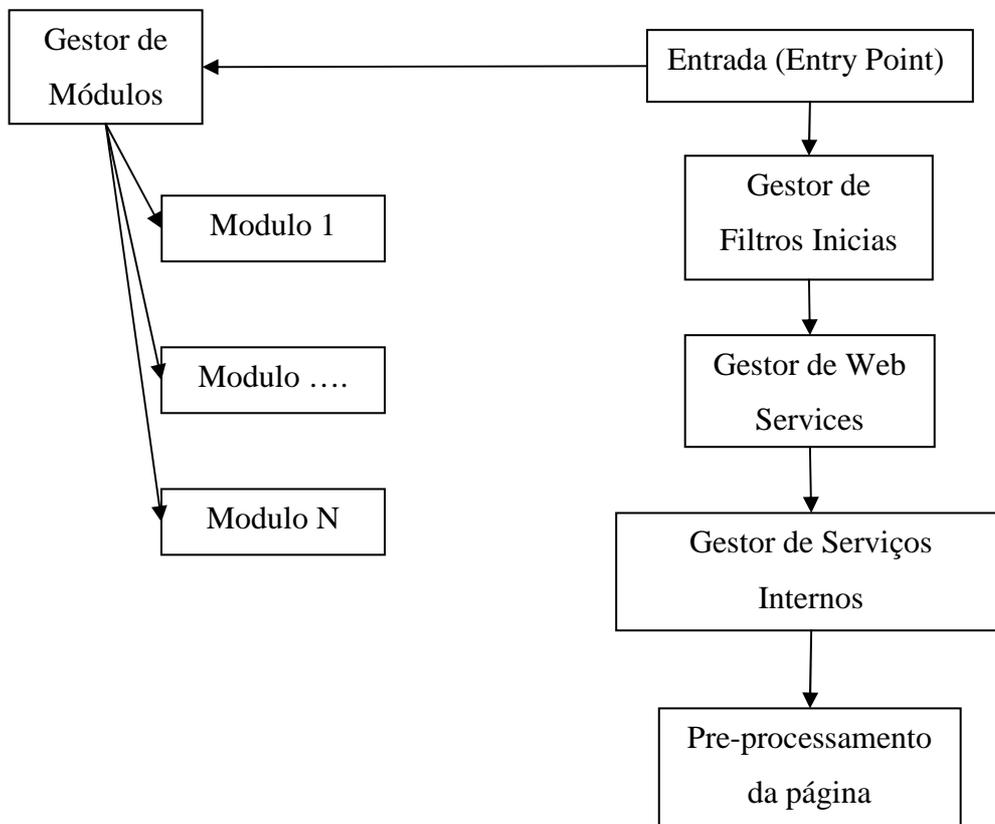


Figura 4 – Diagrama Funcional da Biblioteca

Os requisitos mínimos para a instalação desta biblioteca são a existência de um servidor de páginas que integre um interpretador PHP e a possibilidade de acesso a uma base de dados mysql, sendo a presença desta necessária para o funcionamento da biblioteca.

6.1 Ficheiro de configuração

Ao ser iniciada a biblioteca a configuração de arranque é lida de um ficheiro de configurações. Esse ficheiro, *.Config*, contém toda a informação para a sua configuração. Ainda na inicialização, depois da leitura deste ficheiro, a biblioteca vai verificar a existência de um ficheiro, com o mesmo nome, na directoria de entrada do site. Caso este exista essa configuração irá ser adicionada e/ou sobrepor-se às directrizes do ficheiro de configuração inicial da biblioteca.

A sintaxe deste ficheiro é a vulgarmente utilizada em ficheiros de configuração do Linux - o ficheiro é dividido em secções, sendo definida em cada secção uma ou mais variáveis. O ficheiro base de configuração da biblioteca encontra-se descrito no Apêndice I.

Na secção *Groups* (cf. Apêndice I) são definidas algumas entradas que caracterizam secções que se encontram definidas mais à frente no ficheiro. Por exemplo a entrada:

```
Global          = "GB,Constant";
```

Indica que mais à frente irá ser definida uma secção *Global* (cf. Apêndice I), na qual todos os identificadores das suas variáveis devem começar por “GB” e que a biblioteca deverá criar automaticamente as respectivas constantes em PHP.

Exemplificando, e dentro do exemplo da linha anterior, a seguinte entrada, existente na secção *Global* (cf. Apêndice I):

```
ICUGBDirectoryBaseLibrary = "BaseLibrary/";
```

vai definir no código, em PHP, a constante *ICUGBDirectoryBaseLibrary*, neste caso específico com a directoria onde se encontra o código base da biblioteca.

Todos os módulos existentes, se necessitarem de declarar objectos gerais, deverão estar configurados neste ficheiro.

Qualquer um destes valores poderá ser alterado, como referido atrás, no ficheiro de configuração, se existente, localizado na directoria de entrada do site.

As secções (cf. Apêndice I) obrigatórias para que a biblioteca possa ser utilizada correctamente, são os seguintes:

Global	Contem informação relativa à versão, localização de directorias e definições de constantes usadas na biblioteca
--------	---

Application	Valores por omissão usados pelo site. Deverão ser sobrepostos no ficheiro de configuração local do site
Databases	Dados da base de dados onde a biblioteca guarda a informação
Modules	Indicação dos módulos que devem ser carregados previamente a qualquer operação.
StandAloneCalls	Lista dos Web Services internos e seus identificadores
PreFilters	Lista das rotinas a serem chamadas antes de qualquer processamento e após o carregamento dos módulos
WebServices	Definição das constantes a usar para a implementação de Web Services pela aplicação
WebServicesList	Lista de todos os Web Services disponíveis com a identificação do ponto de entrada no código.

6.2 Entrada (Entry Point)

A biblioteca fornece à aplicação dois modos distintos de funcionamento.

- Modo Transparente

Neste modo o site é desenvolvido independente da biblioteca.

Quando uma página é requisitada e tiver incluído no seu código a chamada à biblioteca:

```
// Call ICu Library
require_once($pathToICuLibrary . "/ICu/ICu.inc");
```

a rotina de inicialização vai interpretar o pedido (URI) da página. Se esse pedido vier especificamente de um Servidor Original, esse pedido é então enviado para rotinas internas da biblioteca para que seja processado. Uma vez o pedido satisfeito, o processamento poderá, ou não, seguir para a página onde originalmente foi efectuada a chamada à inicialização da biblioteca.

- Modo ICu

Neste modo o desenvolvimento do site é baseado na biblioteca ICu. Embora não seja substancialmente diferente do anterior, existem as seguintes diferenças:

- A directoria de entrada do site, pode conter só a página de entrada (normalmente `index.php`), estando todo o resto do site numa directoria

à parte. Neste caso deverá incluir-se, antes da chamada de inicialização da biblioteca, o seguinte código:

```
// If my site is on other directory
$ICUAPDirectoryEntry = ".././newDirectory";
```

A entrada deverá ser efectuada sempre usando a página de entrada (normalmente index.php). A biblioteca põe à disposição métodos para se aceder a todas as outras páginas,

- Para se fazer upload de ficheiros deverá sempre usar-se as rotinas da biblioteca.

Em ambos os casos, a biblioteca vai acrescentar ao header da página de resposta a seguinte entrada:

AProxy: <Versão> <Assinatura> <Protocolos>

Onde

- <Versão> indica a versão do módulo do **AProxy**, descrito mais à frente, que está instalada, na forma numérica 99.99.999,
- <Assinatura> é substituído por uma cadeia alfanumérica, única por cada instalação da biblioteca, identificando assim univocamente o sistema onde está instalada, e
- <Protocolos> é substituído por;

<phpClassLibrary> <OrdemPreferência> <Código> (<Descrição>)

Onde

- <phpClassLibrary> designa o identificador da classe, em PHP, que a existir na biblioteca, deverá ser usado,
- <OrdemPreferência> um numérico, precedido pelo carácter '#', que indica a ordem de preferência do protocolo que deve ser usado na sincronização, no caso da versão da biblioteca instalada no servidor que está a ser acedido ser diferente da versão da biblioteca instalada no sistema do cliente que o está a aceder,

- <Código> uma sequência alfanumérica única por protocolo que o identifica e
- <Descrição> um texto de leitura humana que descreva o protocolo.

Se existir mais que um protocolo disponível na biblioteca deverá acrescentar-se o carácter ‘,’ (virgula) e repetir-se <Protocolos>.

6.3 Gestor de Módulos

A biblioteca é constituída por um núcleo que apenas lê os ficheiros de configuração e gere a instalação de módulos. Um módulo é aqui entendido como uma peça de software que pode ser invocada a qualquer momento apenas fazendo uso de uma chamada a qualquer um dos seus métodos.

Quanto menos ficheiros o interpretador de PHP tiver que ler do sistemas de ficheiros, maior a rapidez com que os scripts são processados. Muitas das vezes, a incerteza da operação requisitada leva a não se poder conhecer à priori quais os ficheiros que devem carregados, levando, na maior parte dos casos, a incluir no código todos ficheiros que poderão ser usados.

Um dos propósitos do sistema de gestor de módulos é gerir automaticamente todas as chamadas a procedimentos de modo a que apenas sejam carregados os ficheiros PHP necessários à operação em curso. Este procedimento apenas está disponível se a aplicação for desenvolvida em modo ICu (cf. página 28).

Um problema no desenvolvimento é a inclusão de ficheiros adicionais, ficheiros CSS ou de javascript, em muitos casos diferentes, que deverão ser passados ao cliente, dependendo do objectivo de cada operação.

É sabido que quantos menos ficheiros o browser tiver que abrir em cada visualização de uma página, mais rápida essa operação se torna. A opção geralmente usada, quando não se sabe à partida que código vai ser necessário, é juntar todos os ficheiros num só, criando assim um ficheiro único que contem todos os ficheiros usados pela aplicação, mesmo que alguns ou grande parte deles não se vá usar. Esta opção torna o ficheiro maior que o necessário.

Depois de inicializada, a biblioteca devolve o controlo à aplicação. Quando esta termina, a biblioteca retoma o controlo, e de acordo com os módulos usados pela aplicação, vai juntar os ficheiros CSS e javascript necessários a cada módulo em dois ficheiros, um contendo todos os ficheiros CSS e outro contendo todo o código em Javascript. Depois de criados são ambos compactados para reduzir o seu tamanho e tornar mais rápida a comunicação.

A instalação e configuração de novos módulos deve respeitar as seguintes condições:

- Cada módulo deverá ser criado numa directoria própria, e deverá incluir-se nela todos os ficheiros necessários ao seu funcionamento.
- Dentro dessa directoria deve existir um ficheiro identificado por *.Info* que vai conter toda a informação sobre os ficheiros que compõem o módulo.
- Cada módulo deve criar, pelo menos, uma classe em PHP. Todas as entradas de procedimentos devem ser objectos pertencentes a essa classe.
- Embora possam ser definidas mais que uma classe em cada módulo, apenas se pode usar uma classe como entrada principal do módulo.
- Os identificadores das classes de entrada do módulo devem começar pela letra ‘m’ minúscula. O nome da directoria onde se encontram todos os ficheiros do módulo deve ser igual ao nome atribuído à classe de entrada, sem o ‘m’ minúsculo, por exemplo, se o identificador da classe for *mMinhaClasse*, o nome a dar à directoria que contem o módulo, deverá ser *MinhaClasse*.

Embora os módulos estejam instalados, por omissão, na directoria *Modules* que se encontra imediatamente abaixo da directoria de instalação da biblioteca, a sua localização pode ser alterada modificando a entrada *ICUGBDirectoryModuleLibrary* existente na secção *Global* do ficheiro inicial de configuração da biblioteca. Pode também adicionar-se directorias para a pesquisa de módulos de modo a que, quando é invocado um método, a biblioteca vá verificar a sua existência a essas directorias. A única limitação é que a operação de acrescentar directorias à lista de pesquisa só é possível programaticamente, fazendo uso do método:

mICuModules::ModuleAddDirectorySearch(sDirectoriasAcrescentar)

6.4 Gestor de Filtros Iniciais

A responsabilidade deste módulo, é a de efectuar chamadas a métodos antes da biblioteca retornar o controlo à página que fez a chamada sua à inicialização.

A lista de métodos a serem invocados é definida no ficheiro de configuração *.Config*. Os métodos são invocados pela ordem da lista definida no ficheiro. A cada chamada a um método é analisado o valor retornado de acordo com o seguinte:

- *ICUPRStopAndExit* – O processamento da página é interrompido imediatamente sendo devolvido todo o output gerado, até ao momento, ao cliente.
- *ICUPRStopChain* – O processamento da lista de métodos é interrompido, e o processamento é passado para a etapa seguinte na biblioteca.
- *ICUPRContinue* – A chamada ao método resultou em sucesso, sendo então efectuada a chamada ao próximo método da lista.

Como exemplo, o módulo *AProxy* usa este sistema para se identificar ao cliente como sendo um sistema base de replicação e sincronização.

A lista de métodos a serem invocados encontra-se definida no ficheiro inicial de configuração da biblioteca, dentro da secção *PreFiltersList*. A sua sintaxe é a seguinte:

<ID do Filtro>="<Método>"

Onde:

- *<ID do Filtro>* - É um identificador único na lista.
- *<Método>* - Identificador do método a ser invocado.

Como exemplo temos:

ActiveProxyPreFilter = "mAProxy::PreFilter";

Neste caso será invocado o método *mAProxy::PreFilter* antes de a biblioteca retornar, após a chamada à sua inicialização.

6.5 Gestor de Web Services

A biblioteca inclui um módulo que automatiza e uniformiza a construção e implementação de Web Services. A biblioteca vai analisar o URI requisitado pelo cliente, e se verificar que se trata de uma chamada a um Web Service definido vai então proceder à chamada do respectivo método. A sintaxe de uma chamada a um Web Service é a seguinte:

?WSRQST=<ID do Serviço>

De acordo com o identificador do serviço requisitado, a biblioteca vai proceder à chamada do método respectivo. Imediatamente antes do método ser evocado, a biblioteca começa capturar o output da aplicação até se dar o retorno do método. Esse output pode ser

descartado, dependendo do valor retornado pelo método. O valor de retorno é analisado e processado de acordo com o seguinte:

- *ICUWSContinueAndClean* – Continua o processamento, mas o output do método invocado é eliminado do output final.
- *ICUWSContinue* – Continua o processamento, mantendo o output efectuado pelo método.
- *ICUWSError* – O método retornou uma condição de erro. Ao output gerado pelo método, é inserido no início do mesmo, o texto ‘WS-ERROR: ‘, o sistema interrompe imediatamente o processamento devolvendo o output gerado.
- *ICUWSOK* - Ao output gerado pelo método é inserido no início do mesmo o carácter ‘O’ e o sistema interrompe imediatamente o processamento, devolvendo o output gerado.
- *ICUWSOKOnly* – O output gerado pelo método é retornado e o processamento é imediatamente interrompido.

A lista de *Web Services* disponíveis encontra-se definida no ficheiro inicial de configuração da biblioteca, na secção *WebServicesList*. A sintaxe de cada linha, a que corresponde um *Web Service*, é a seguinte:

<IDdo Web Service>="<Método>"

Onde:

- *<ID do Web Service>* - É um identificador único na lista.
- *<Método>* - Identificador do método a ser invocado.

Como exemplo temos:

APGUPT = "mAProxy::WSGetUpdates";

Neste caso *APGUPT* corresponde ao ID do serviço e o lado direito da atribuição indica qual o método a ser evocado.

6.6 Gestor de Serviços Internos

Este módulo é interno à biblioteca e existe para que operações internas sejam invocadas. Por exemplo, como referido atrás, quando a biblioteca cria um ficheiro único que

contém todas as declarações CSS definidas nos módulos que foram usados, a chamada ao serviço que devolve o ficheiro único compactado é a seguinte:

?ICUSAC=LSC&ID=<ID do ficheiro>

Em que *ICUSAC* indica que é uma chamada a um serviço interno, *LSC* o serviço requisitado, neste caso o ficheiro compactado das declarações CSS, e *ID* refere-se ao nome do ficheiro que foi criado.

6.7 Pré-processamento da página

Em modo transparente (cf. página 28), após a inicialização da biblioteca ICu, o processamento passa imediatamente para a página que invocou a inicialização da biblioteca.

O modo ICu (cf. página 28), permite que todo o site possa ser instalado numa directoria não acessível através do servidor de páginas. Isto acrescenta ao sistema um pouco mais de segurança, especialmente se o site contiver um ficheiro de configuração local ao mesmo, tornando-o assim inacessível.

A biblioteca, após inicializada, vai chamar a página de entrada especificada no ficheiro de configuração. Antes de retornar a biblioteca ICu começa a capturar todo o output. Após o retorno da página de entrada, a biblioteca vai modificar o output gravado, de modo a inserir todas as novas instruções que resultam das chamadas aos vários módulos efectuadas pela aplicação.

Este modo permite incluir ficheiros CSS, javascript e código em javascript programaticamente, através dos seguintes métodos:

- *mICuModules::LoadJavaScript(<Ficheiro>)* – Inclui o ficheiro *<Ficheiro>* de javascript na página.
- *mICuModules::LoadCSS(<Ficheiro>)* – Inclui o ficheiro *<Ficheiro>* de CSS na página.
- *mICuModules:: AddToInitJS(<Código>)* – Vai incluir o código em javascript, definido na variável *<Código>*, imediatamente antes do término da secção *<head>* da página em HTML.
- *mICuModules:: AddToInitLoadJS(<Código>)* – Vai incluir o código em javascript, definido na variável *<Código>*, sendo este executado logo após a página tenha sido toda carregada pelo browser.

Para o caso em que o analista tenha escolhido colocar o site numa directoria inacessível através do Servidor de Páginas, o link para páginas específicas do site pode ser criado usando o seguinte método:

mICuLink::LinkToPage(<Ficheiro | Directoria>)

O método devolve um link da forma

http://xxx.xxxxx.xx/?ICUSAC=LSJ&ID=<Ficheiro | Directoria>

em que:

- *xxx.xxxxx.xx* – Refere o site.
- *<Ficheiro | Directoria>* - Refere a directoria ou o ficheiro desejado. Este valor é sempre considerado a partir da directoria raiz do site.

Isto permite que, se o site estiver numa directoria acessível através do Servidor de Páginas, ou não, o valor a usar em *<Ficheiro | Directoria>* seja sempre o mesmo, independente da directoria onde esteja ao site.

7. Sistema de Licenças

Cada instalação da biblioteca ICu necessita ser activada por um servidor de licenças. A função deste servidor é a de identificar e autenticar todas as instalações existentes de modo a que, quando um site é replicado, exista a certeza onde está instalado.

Cada instalação da biblioteca recebe um número de série e uma chave no momento em que é instalada. Essa chave é usada como identificação pelas várias instalações da biblioteca.

O servidor de licenças é usado também para verificação da autenticidade das chaves e, por consequência, da validação da autenticidade da biblioteca.

Na implementação do Servidor de Licenças foi usada a biblioteca ICu, sendo assim o servidor segue as normas definidas na biblioteca para a comunicação, mais concretamente os Web Services.

O armazenamento dos dados é efectuado na base de dados de controlo, que é criada quando da instalação da Biblioteca ICu.

7.1 Número de Série

Cada instalação da biblioteca tem que obrigatoriamente ter activo um número de série único. Esse número de série é composto por uma cadeia alfanumérica de 32 caracteres. Os caracteres usados são do carácter 'A' ao 'Z'. O ultimo carácter da cadeia serve de carácter de controlo e é construído efectuando seguintes operações:

1. É guardada a soma do código ASCII dos outros 31 caracteres,
2. É dividido por 26 e toma-se o resto da divisão,
3. Ao resultado é somado o valor de 65 (valor ASCII do carácter 'A')

Os números de série são criados pelo servidor de licenças e armazenados com a indicação de inválidos, até serem validados por uma instalação da biblioteca ICu.

7.2 Chave de Identificação

Quando um número de série é criado pelo servidor de licenças é criada também uma chave através da aplicação do algoritmo de hashing md5 sobre esse mesmo número de série. Esta chave é usada, por questões de segurança, nas operações de validação de uma instalação e de verificação da autenticidade da mesma, substituindo a utilização do número de série nas comunicações

A chave é constituída por 32 caracteres, resultante da aplicação do algoritmo de hashing e é criada e armazenada no mesmo instante em que é criado o número de série.

7.3 Validação de uma Instalação da Biblioteca ICu

Como referido anteriormente, só após a validação da instalação a biblioteca ICu pode ser utilizada. Quando é efectuada a chamada à inicialização da biblioteca esta vai verificar se já se encontra validada com um número de série válido. Se não for o caso, é solicitado ao utilizador a introdução do número de série respeitante a essa instalação.

O número de série deve ser solicitado à entidade que gere as licenças. Após essa solicitação, e uma vez aceite, o número de série é enviado por email para o utilizador que o solicitou.

Uma vez introduzido o número de série a operação de validação é efectuada pela biblioteca, como descrito na figura seguinte:

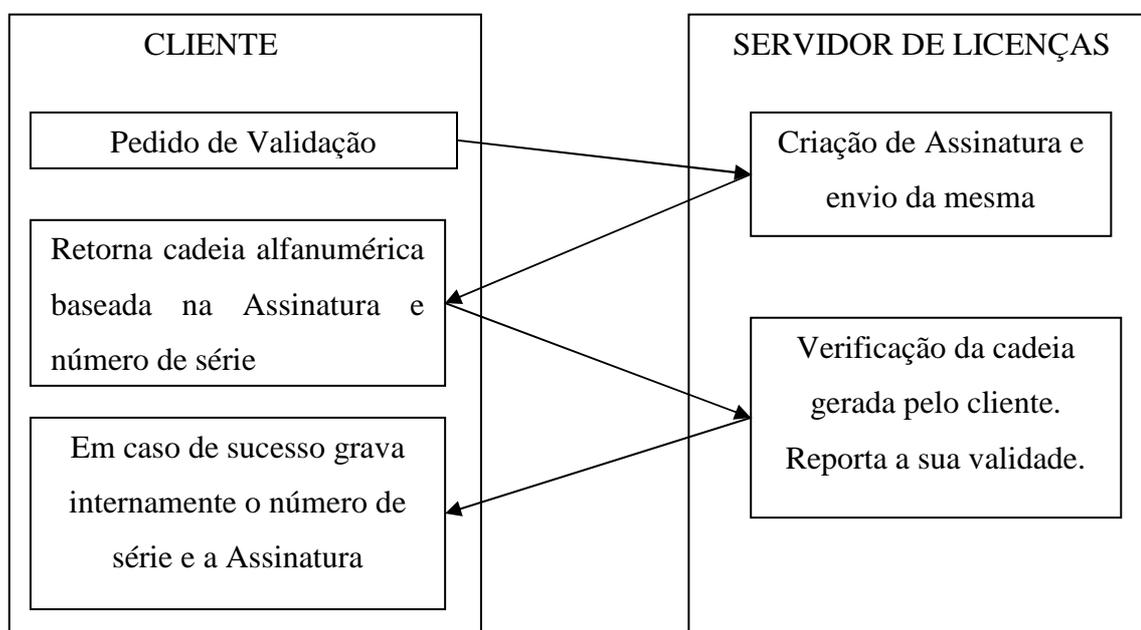


Figura 5- Validação de uma instalação da Biblioteca ICu

O conjunto de operações definidas permitem que o número de série seja transmitido usando uma chave auxiliar, para que o mesmo não se encontre explicitamente representado em qualquer cadeia de caracteres trocados entre o cliente e o Servidor de Licenças.

A autenticação é efectuada em dois passos distintos, o primeiro em que o cliente requisita uma cadeia alfanumérica auxiliar ao Servidor de Licenças. Após a recepção dessa cadeia o cliente vai construir uma nova cadeia alfanumérica baseada na cadeia (*Assinatura*) que vai ser usada no segundo passo da validação.

7.3.1 Criação da Assinatura

Quando o Servidor de Licenças recebe um pedido de validação de um número de série, para a activação de uma biblioteca, vai criar uma cadeia alfanumérica com o seguinte formato:

xxYxxmxxdxxHxxixxsxx

em que:

- Y – representa o ano da hora do pedido com 4 dígitos
- m – representa o mês da hora do pedido com dois dígitos
- d – representa o dia do mês da hora do pedido com dois dígitos
- H – representa a hora da hora do pedido com dois dígitos em formato de 24 horas
- i – representa os minutos da hora do pedido com dois dígitos
- s – representa os segundos da hora do pedido com dois dígitos

Esta cadeia é designada como *Assinatura* e é válida durante 30 segundos. Se não for efectuado um pedido de validação de um número de série nos subsequentes 30 segundos, esta assinatura torna-se inválida e o processo de validação terá que ser repetido de início outra vez.

7.3.2 Cadeia de Validação de Numero de Série

Após a recepção da *Assinatura*, o cliente vai gerar uma cadeia alfanumérica que deve enviar ao servidor de licenças para que este retorne a validade do número de série introduzido pelo utilizador. Esta cadeia é construída baseada na *Assinatura* recebida do servidor de licenças e do número de série introduzido pelo utilizador, usando a seguinte sequência de operações:

1. Aplica o algoritmo de hashing md5 ao numero de série,
2. À cadeia alfanumérica resultante vai concatenar a *Assinatura* recebida do Servidor de Licenças,
3. À cadeia alfanumérica resultante vai aplicar o algoritmo de hashing md5, resultando assim uma cadeia alfanumérica de 32 caracteres.

Após a criação desta cadeia alfanumérica, a mesma será enviada ao Servidor de Licenças, para validação.

7.4 Validação de uma instância de uma Biblioteca ICu

Quando um cliente já tem instalado o sistema local para o acesso offline a aplicações, nas suas operações de sincronismo e de instalação de novos sites tem que conseguir reconhecer que o servidor original donde provem o site é um servidor registado e confiável.

Como descrito mais a frente, o Servidor Local mantém uma lista de servidores que fornecem a possibilidade de sites offline. Quando um site é pedido pela primeira vez por um cliente e o browser detecta que o mesmo usa a Biblioteca ICu, a informação é passada ao Servidor Local.

Este vai verificar se o servidor desse site já se encontra na sua lista de Servidores Originais. Se não se encontrar ainda, o Servidor Local vai pedir ao Servidor Original que envie a sua *Assinatura* para que a possa validar, numa primeira fase, com o Servidor de Licenças, como ilustra a figura seguinte:

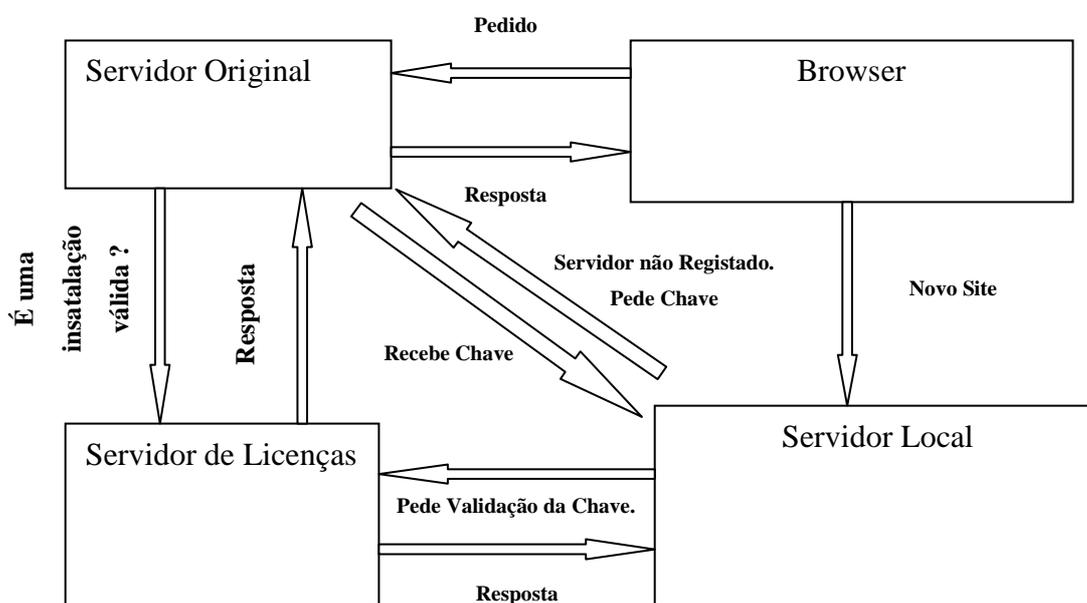


Figura 6 - Validação de um Servidor Original

A validação inicial de um Servidor Original é sempre iniciada pelo Servidor Local quando este recebe, pela primeira vez, um pedido de um site cujo Servidor Original ainda não se encontra na lista de servidores mantidas pelo Servidor Local.

Nesse caso a sequência de operações é a seguinte:

1. Envia um pedido ao Servidor Original da sua chave de identificação, enviando a sua própria chave de identificação junto ao pedido,
2. O Servidor Original usa a chave enviada pelo Servidor Local para a validar com o Servidor de Licenças,
3. Caso essa chave seja validada pelo Servidor de Licenças, o Servidor Original devolve a sua própria chave ao Servidor Local,

4. O Servidor Local, após receber a chave enviada pelo Servidor Original, vai enviar um pedido ao Servidor de Licenças para validar a chave recebida,
5. O Servidor de Licenças valida a chave,
6. O Servidor Local armazena a informação no seu repositório local e a partir daí todos os sites provenientes desse Servidor Original que se encontrem já instalados no Servidor Local, passarão a ser servidos pelo Servidor Local.

7.5 Web Services

O Servidor de Licenças, que se encontra em <http://www.icuframework.com>, põe à disposição três Web Services para a execução das operações atrás descritas:

7.5.1 WSBeginRegister

Este serviço põe à disposição o primeiro passo da validação de um número de série na instalação de uma biblioteca ICu:

Utilização:

<http://www.icuframework.com?WSRQST=USBGRG>

Retorno:

‘O’ <Assinatura> - em caso de sucesso,

‘WS-ERROR: E-USBGRG-001’ – em caso de erro

Em caso de sucesso <Assinatura>, é como especificado em 7.3.1 Criação da Assinatura

7.5.2 WSEndRegister

Este serviço põe à disposição o segundo passo da validação de um número de série na instalação de uma biblioteca ICu:

Utilização:

<http://www.icuframework.com?WSRQST=USENRG&K=<CadeiaAlfanumerica>>

< CadeiaAlfanumerica> tem o formato especificado em 7.3.2 Cadeia de Validação de Numero de Série.

Retorno:

‘O’ - em caso de sucesso,

‘WS-ERROR:’ <MensagemErro> – em caso de erro

MensagemErro tem os seguintes significados:

- E-USENRG-001 – Parâmetro K no URL não existente
- E-USENRG-002 – Assinatura com prazo de validade ultrapassado.

- E-USENRG-003 – Acesso ao sistema temporariamente encerrado
- E-USENRG-004 – *Assinatura* já em uso
- E-USENRG-005 – Numero de Série inválido
- E-USENRG-006 – *Assinatura* inválida.

7.5.3 WSCheckSignature

Este serviço disponibiliza a verificação de uma *Assinatura*.

Utilização:

<http://www.icuframework.com?WSRQST=USCHSG&K=<CadeiaAlfanumerica>>

<CadeiaAlfanumerica> é criada aplicando o algoritmo de *hashing* md5 à *Assinatura* que se deseja validar.

Retorno:

‘O’ - em caso de sucesso,

‘WS-ERROR:’ <MensagemErro> – em caso de erro

MensagemErro tem os seguintes significados:

- E-USENRG-001 – Parâmetro K no URL não existente
- E-USENRG-002 – *Assinatura* inválida.

8 Módulo *AProxy*

Todo o processamento que permite a utilização do site offline foi implementado como um módulo na biblioteca ICu. É este módulo que gere todas as tarefas necessárias, quer na parte do Servidor Original, quer na parte do cliente.

Uma aplicação, ao inserir a linha de código para a inclusão da biblioteca ICu, fica imediatamente preparada para poder ser replicada e sincronizada, desde que essa opção esteja activa no ficheiro *.Config*, secção *ActiveProxy*, variável *APRProxyServer*. Se for atribuído o valor 0 (zero), o módulo encontra-se desactivado, se for atribuído o valor 1, o módulo encontra-se activo.

Pode-se dividir este módulo em três componentes, em conformidade à sua operacionalidade:

- Modo Servidor Original
- Modo Servidor Local
- Modo Replicação e Sincronismo

De acordo com a instalação da biblioteca ICu como Servidor Original, ou como Servidor Local, o módulo *AProxy* vai conduzir as suas operações.

Este módulo necessita de armazenar grandes quantidades de informação relativa aos sites que são geridos. Essa informação é armazenada na SGBD local, na base de dados de controlo usada pela biblioteca ICu.

8.1 Configuração do Módulo

O módulo *AProxy* encontra-se instalado numa directoria abaixo da directoria onde estão instalados os módulos da biblioteca ICu. O ficheiro de configuração do módulo que se encontra na directoria raiz do módulo, é o seguinte:

```
[Module]
Name=Active Proxy
Code=CRDL1000-0002
MainClass=mICuAProxy
Version=1.0

[Files]
File1=EntryPoint.inc
File2=AProxy.css
File3=AProxy.js

[EntryPoints]
APDBCH      = "mAPDBStructure::LoadStructure";
```

```
[CommProtocols]
SSLProtocol      = "mAPSSLProtocol #1 APSSL-1.0 (AProxy SSL V1.0)";

[General]
TablePrefix     = "icup_";

[Admin]
Directory       = "Admin/";
```

A descrição das secções é a seguinte:

- *Module* - Contêm informação geral relativo ao módulo.
- *Files* - Contem os ficheiros que a biblioteca ICu deve carregar quando o **AProxy** é evocado.
- *EntryPoints* - Define entradas para rotinas internas.
- *CommProtocols* - Define todos os protocolos disponíveis pelo **AProxy** para efectuar comunicações entre o Servidor Original e os clientes.
- *General* - Contem o prefixo usado nos identificadores das tabelas usadas na base de dados de controlo da biblioteca ICU.
- *Admin* - Designa a directoria onde se encontram os ficheiros que compõem a aplicação de administração do **AProxy**.

8.2 Adição de um novo Protocolo de Comunicação

Embora o **AProxy** já venha configurado de origem na instalação da biblioteca ICu, existe a possibilidade de adicionar interfaces de comunicação aos já existentes (SSL e RAW).

O procedimento para acrescentar mais um protocolo de comunicação é o seguinte:

1. Acrescentar à secção *CommProtocols* mais uma entrada. A sintaxe da mesma é a seguinte

```
<Variavel>="<IDClasseEntrada> #<OrdemPreferencia> <Codigo> <Descritvo>"
```

Em que

- *<Variavel>* - É um identificador único sem significado especial, só usado como identificador da entrada.
- *<IDClasseEntrada>* - É o identificador da classe, em PHP, que deverá ser usada pela biblioteca ICu nas chamadas aos métodos. Este identificador deve respeitar a norma de criação de identificadores descrito anteriormente, isto é, deve começar pelo carácter ‘m’ seguido do nome da classe.

- *<OrdemPreferencia>* - É a ordem de preferência que deve ser usada na escolha do protocolo de comunicação, quando a biblioteca inicia um processo de comunicação para efeitos de replicação ou sincronização com outro sistema.
 - *<Codigo>* - É um código, único que vai identificar o protocolo entre os vários sistemas. O código deverá incluir também a sua versão.
 - *<Descritivo>* - É uma descrição geral do protocolo.
2. Criar uma directoria abaixo da directoria *Filters*, que se encontra dentro da directoria de instalação do **AProxy**. O identificador da nova directoria deve ser exactamente idêntico ao identificador dado à classe, retirando o carácter ‘m’ inicial. Dentro da directoria criada deverá existir um ficheiro com o mesmo identificador dado à directoria seguido da extensão ‘.inc’.
 3. Criação do código, em PHP, relativo ao protocolo. O protótipo do código a desenvolver é o seguinte:

```
class mRawProtocol {  
  
    // Function call by AProxy when it pretends do begin a communication with a remote  
    // Can return Information to send to the remote point  
    static public function InitComm($nCdsites) {  
        return "";  
    }  
  
    // Function call by AProxy when it begins to communicate with the remote point.  
    // The parameter receives any data send from the remote point, data resulting from  
the return of InitPoint() function of the remote point.  
    // Must be call once for each communication session  
    // Returns TRUE on success and FALSE on error  
    static public function BeginComm($sGeneralData) {  
        return TRUE;  
    }  
  
    // Function call by AProxy for each buffer of data, just before it is send to the other  
point  
    // Must return the new data buffer  
    static public function WriteComm($sMsg) {  
        return $sMsg;  
    }  
  
    // Function call by AProxy after read a chunk of data from the other point, but before  
it is use by AProxy  
    // Must return the new data buffer
```

```

static public function ReadComm($sMsg) {
    return $sMsg;
}

// Function call by AProxy after it finishes the communication with the remote point
// Usually use to clean internals and data.
static public function EndComm() {
    return ;
}
}
    
```

8.2.1 Esquema Funcional

O módulo **AProxy** é o responsável pela comunicação, para fins de replicação ou sincronização, entre o Servidor Local e o Servidor Original. O controlo só é devolvido à biblioteca ICu após a conclusão da comunicação. O processo de replicação ou de sincronização é da total responsabilidade do Servidor Local.

Esta operação pode demorar alguns segundos, como demorar algumas horas. Como em PHP não existem métodos portáveis para que se possam criar processos separados e controlados por uma mesma instância, esta operação de sincronização é efectuada em processos independentes que são lançados a correr no sistema operativo hospedeiro.

Como o PHP também não disponibiliza métodos portáveis para a comunicação entre processos, os processos de replicação ou sincronização utilizam uma tabela da base de dados de controlo da biblioteca ICu para os efectuar.

O diagrama funcional de uma operação de replicação ou sincronização é o seguinte:

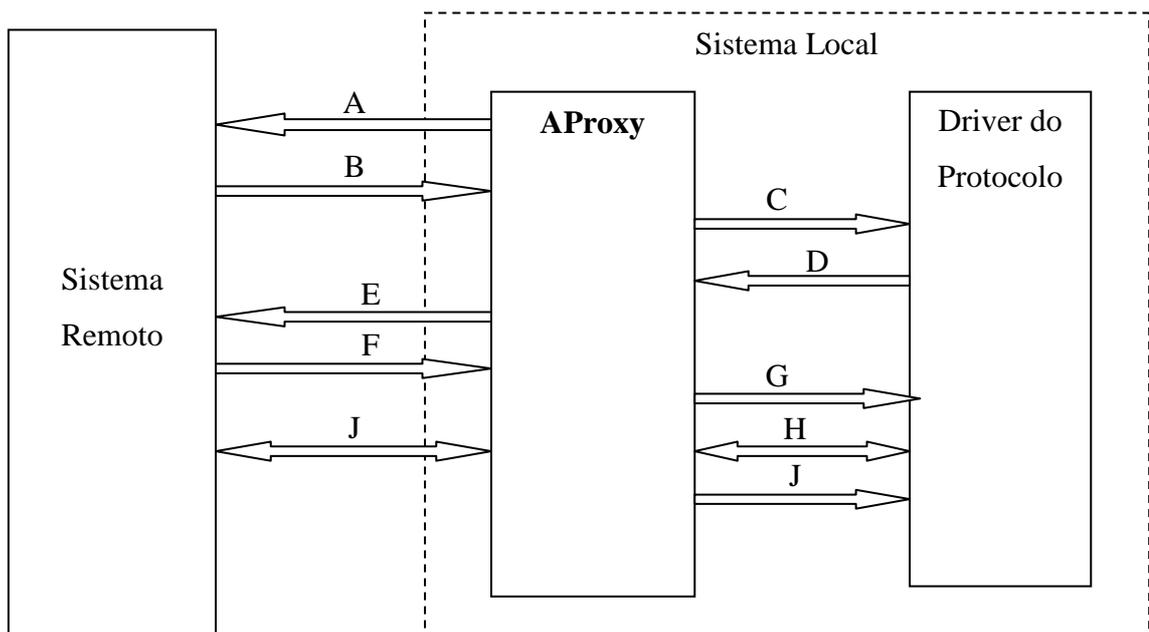


Figura 7 - Esquema Funcional do uso de Protocolos de Comunicação

Explicação da Figura 7:

- A. Requisita o início de replicação ou sincronismo, identificando-se e enviando os protocolos suportados.
- B. Resposta, indicando a disponibilidade para a operação requisitada e qual o protocolo a usar nas comunicações, incluindo informação adicional opcional para o driver do protocolo que vai ser usado.
- C. Chamada à função *InitComm()* do driver do protocolo retornado em B.
- D. Retorno da chamada ao método *InitComm()*, indicando a disponibilidade de uso do driver e informação adicional.
- E. Envio do pedido ao Servidor Original para iniciar a comunicação, incluindo a informação retornada em D.
- F. Resposta, indicando a autorização, ou não, para começar a troca de dados.
- G. Chamada ao método *BeginComm()* enviando, caso a tenha recebido, a informação adicional enviada pelo sistema remoto em B.
- H. Uso dos métodos *WriteComm()* e *ReadComm()*, para que internamente a informação seja alterada pelo protocolo.
- I. Envio e recepção dos dados resultado do ponto H, entre o sistema local e o sistema remoto.
- J. Indicação ao driver de término da comunicação.

8.2.2 *InitComm()*

Este método é invocado pelo módulo **AProxy**, antes de se iniciar a comunicação com o sistema remoto.

Utilização:

InitComm(<CdSites>)

<CdSites> indica o ID único do registo na base de dados de controlo da biblioteca ICu, que identifica o servidor remoto que se quer contactar.

Retorno:

FALSE – Em caso de não se poder prosseguir com a operação,

TRUE – Se a operação puder se prosseguir e não é necessário enviar nenhuma informação suplementar ao servidor remoto,

Cadeia Alfanumérica - Se a operação puder ser prosseguida. A cadeia alfanumérica deverá ser entregue ao servidor remoto.

8.2.3 BeginComm()

Este método é invocado pelo módulo **AProxy** após iniciar a comunicação com o sistema remoto e antes de começar a recepção e o envio de dados ao servidor remoto

Utilização:

BeginComm(<CadeiaAlfanumerica>)

< CadeiaAlfanumerica> No caso do servidor remoto ter enviado informação no passo B (Figura 7), essa informação será passada aqui.

Retorno:

FALSE – Em caso de não se poder prosseguir com a operação,

TRUE – Se a operação puder ser prosseguida.

8.2.4 WriteComm()

Este método é invocado pelo módulo **AProxy** sempre que tiver dados para enviar ao servidor remoto.

Utilização:

WriteComm(<CadeiaAlfanumerica>)

<CadeiaAlfanumerica> Cadeia alfanumérica que representa a informação a enviar ao servidor remoto. Se em *<CadeiaAlfanumerica>* vier o valor FALSE, isso indica ao método que, se necessário, por questões próprias de cada protocolo, a função continue a tratar a cadeia alfanumérica que veio em *<CadeiaAlfanumerica* na chamada anterior ao mesmo método.

Retorno:

FALSE – Em caso da operação estar completa,

Cadeia Alfanumérica – Informação a ser enviada para o servidor remoto.

Nota:

O **AProxy**, vai efectuar chamadas repetidas a este método até o método retornar o valor FALSE. Na primeira chamada ao método envia a cadeia alfanumérica a ser tratada. Nas chamadas subsequentes envia o valor FALSE. Os valores retornados pelo método vão sendo concatenados por ordem, até que o método retorne o valor FALSE. A cadeia alfanumérica criada é então enviada ao servidor remoto.

Se, por restrições do protocolo usado, a cadeia alfanumérica a retornar pelo método tiver que ter um número fixo de caracteres, e a cadeia apresentada ao método for de dimensão

diferente, deverá a função, se necessário, efectuar o retorno das várias partes em sucessivas chamadas e no fim do processamento retornar o valor FALSE.

8.2.5 ReadComm()

Este método é invocado pelo módulo **AProxy** sempre que tiver dados disponíveis, recebidos do servidor remoto.

Utilização:

ReadComm(<CadeiaAlfanumerica>)

<CadeiaAlfanumerica> Cadeia alfanumérica que representa a informação recebida do servidor remoto.

Retorno:

FALSE – Em caso da operação estar incompleta e necessitar de mais informação para ficar completa,

Cadeia Alfanumérica – Informação resultante da aplicação do protocolo à informação recebida.

Nota:

Por cada segmento de informação que recebe do servidor remoto, o **AProxy** efectua uma chamada ao método. Se este devolver o valor FALSE, significa que o método necessita de mais informação para poder construir a cadeia alfanumérica final.

Foi definido este algoritmo pois, por exigências de alguns protocolos, poderá a informação recebida não corresponder a um segmento completo, que possa ser tratado pela função *ReadComm()*.

8.2.6 EndComm()

Este método é invocado pelo módulo **AProxy** sempre que uma comunicação com o servidor remoto tiver sido encerrada.

Utilização:

EndComm()

Retorno:

TRUE.

8.3 Sincronização de Sistemas

Após a identificação e validação entre o sistema local e o remoto, o módulo **AProxy** vai proceder às operações necessárias para sincronizar, quer o sistema de ficheiros, quer a base de dados local, com a informação residente no Servidor Original.

Após o contacto inicial efectuado pelo Servidor Local (ver Figura 7) ao Servidor Original, ambos vão construir um ficheiro com a informação a passar para a outra parte. Esse ficheiro inclui tanto as alterações efectuadas a ficheiros como aos dados das respectivas bases de dados.

Como se assumiu que na comunicação entre os dois sistemas poderão existir perturbações, ambos os sistemas vão dividir os respectivos ficheiros criados em segmentos de igual tamanho. Depois de criados os segmentos, começam a enviar segmento a segmento. Após a recepção de todos os segmentos, os servidores vão reconstruir os respectivos ficheiros, juntando os segmentos recebidos e de seguida proceder às actualizações necessárias.

A informação a ser recebida é guardada temporariamente em tabelas da base de dados de controlo da biblioteca ICu. Por definição, a responsabilidade do início de uma comunicação deverá ser sempre do Servidor Local. Esse pedido de início de comunicação deve ser sempre seguido do envio de uma lista da informação a sincronizar, por parte do sistema de remoto. Em resposta a esse pedido, o Servidor Original irá então enviar a sua lista de informação a sincronizar localmente.

8.3.1 Ficheiro de Envio

A estrutura do ficheiro de envio é codificada pelo algoritmo JSON e representa uma estrutura em array. Foi escolhida esta estrutura pois a maioria das linguagens usadas hoje em dia no desenvolvimento para a Web têm este algoritmo implementado, tornando assim mais rápido o desenvolvimento de soluções, além de o desenvolvimento de código que trata informação ser facilitado com o uso de arrays.

A definição desse array é a seguinte, em que o identificador 'F' identifica o array enviado:

F[0] – Inclui toda a informação respectiva ao sistemas de ficheiros,

F[1] - Inclui toda a informação respectiva aos dados da base de dados,

F[2] - Inclui toda a informação relativa a equivalências de chaves primárias.

Registo do sistema de ficheiros:

F[0][0..n][0] – A data de alteração em tempo GMT,

F[0][0..n][1] – O nome do ficheiro, sem caminho,

F[0][0..n][2] – O tamanho do ficheiro em bytes,

F[0][0..n][3] - Caminho completo do nome do ficheiro, a partir da directoria de entrada do site,

F[0][0..n][4] – Imagem do ficheiro codificada usando a função *escape* do Java Script.

F[0][0..n][5] – 1 (um) caso seja um ficheiro de upload local, 0 (zero) caso contrário.

Em que:

- n, é o numero de ficheiros enviados.

Registo da informação da base de dados

F[1][0..m][0] - Identificação da base de dados,

F[1][0..m][1..n][0] - Identificação da tabela,

F[1][0..m][1..n][1][0..p][0] – Identificação do campo da tabela,

F[1][0..m][1..n][1][0..p][1] – Se pertencer à chave primária, qual a ordem na mesma,

F[1][0..m][1..n][1][0..p][2] – Tipo do campo:

- S – Character,
- F – Numerico,
- D – Data,
- M – Binário

F[1][0..m][1..n][1][0..p][3] - Comprimento, em bytes, do campo,

F[1][0..m][1..n][1][0..p][4] – Valor por defeito, em caso de inserção de registos,

F[1][0..m][1..n][1][0..p][5] – 0 (zero) não permite o valor NULL, 1 (um) caso contrário.

F[1][0..m][1..n][1][0..p][6] – Caso o campo seja uma chave estrangeira, o identificador da respectiva tabela.

F[1][0..m][1..n][2] - Tipo de operação: I – Inserção, U-Modificação de dados, D-Remoção do registo,

F[1][0..m][1..n][3][0..q][0..p] – Valor do campo, codificado usando a função *escape* do Java Script.

Registo de chaves primárias:

F[2][0..m][0] - Identificação da base de dados,

F[2][0..m][1..n][0] - Identificação da tabela,

F[2][0..m][1..n][1] – Valor da chave primária no Servidor Local,

F[2][0..m][1..n][1] – Valor da chave primária no Servidor Original

Em que:

- m, é o numero diferentes de bases de dados usadas,
- n, é o numero de tabelas usadas, por base de dados,
- p, é o numero de campos de cada tabela,
- q. é o numero de registos, por tabela.

Esta estrutura, permite que sejam mais tarde, se necessário, acrescentados mais itens ao ficheiro de envio.

A posição F[2] do array só é criada e enviada pelo Servidor Original em resposta a um pedido de sincronização efectuado por um Servidor Local.

A informação dos ficheiros a considerar pelo Servidor Original, quando cria a lista de ficheiros a enviar, é armazenada numa tabela da base de dados de controlo da biblioteca ICu.

As tabelas que são usadas pelo **AProxy** na base de dados de controlo da biblioteca ICu usam o prefixo 'icuap_'.

O módulo **AProxy** cria as seguintes tabelas na base de dados de controlo:

- icuap_domains – Esta tabela define, para cada site, um conjunto de domínios de exportação. Estes domínios podem ser entendidos como perfis de sincronização. De momento apenas se encontra definido um perfil, fixado como universal e de acesso público.
- icuap_files – Esta tabela mantêm a lista de ficheiros, por domínio, a considerar na criação da lista de ficheiros a sincronizar.
- icuap_db – Esta tabela mantêm a lista de bases de dados, por domínio, a considerar na criação da lista de dados a sincronizar.
- icuap_dbtables – Esta tabela mantêm a lista de tabelas a considerar na criação da lista de dados a sincronizar, por cada registo da tabela icuap_db.
- icuap_dbcols – Esta tabela mantêm a lista de colunas a considerar na criação da lista de dados a sincronizar, por cada tabela da tabela icuap_dbtables.
- icuap_proxylist – Inclui toda a informação sobre comunicações, por site, que Servidores Locais remotos efectuaram com o Servidor Original. Esta tabela guarda, entre outras informações, a hora da última comunicação concluída com sucesso pelo Servidor Local remoto.

- icuap_dbkeys – Esta tabela, usada apenas nos sistemas locais, é usada para guardar as equivalências das chaves primárias entre o sistema local e o sistema original do site.
- icuap_dblist – Esta tabela guarda todas as operações de inserir, modificar e eliminar registos que foram efectuadas nas tabelas das bases de dados.

Baseado na informação destas tabelas, os servidores, quer os locais, quer o original, vão criar o array descrito anteriormente.

8.3.2 Sincronização de Ficheiros

Quando um Servidor Original é solicitado a criar a sua lista de envio de ficheiros, o sistema vai verificar se existem alterações aos ficheiros que compõem toda a estrutura do site, baseado na informação anteriormente gravada na tabela icuap_files. São efectuadas duas verificações para determinar se um ficheiro sofreu alterações desde a última sincronização:

- O sistema vai recolher à tabela icuap_proxylist a data da última sincronização efectuada pelo Servidor Local remoto.
- Essa data é confrontada com a última data de alteração do ficheiro no sistema operativo anfitrião. Caso a ultima data de alteração do ficheiro seja superior à data da última sincronização com o Servidor Local remoto, esse ficheiro passa imediatamente a ser incluído na lista de envio.
- Se no passo anterior o ficheiro não tiver sido eleito para ser enviado, o sistema vai comparar o tamanho do ficheiro, em bytes, com a informação armazenada na tabela icuap_files na última verificação. Se os respectivos valores não coincidirem, o ficheiro passa imediatamente a ser incluído na lista de envio.

No caso especial de uma aplicação efectuar upload de ficheiros, o Servidor Local deverá ir efectuar esta operação nas directorias indicadas como de upload na tabela icuap_files e incluir apenas na sua lista de envio os ficheiros que se encontram dentro dessas directorias.

Quando recebe a lista de envio do Servidor Original, o Servidor Local deverá simplesmente efectuar a gravação da imagem recebida no local designado.

Um aspecto importante desta operação é a determinação da directoria exacta de colocação do ficheiro. Na tabela icuap_files um dos campos da mesma indica qual a directoria principal de entrada no site. Os ficheiros que vêm na lista de envio, trazem a informação da directoria onde devem ser gravados. Esse caminho é sempre considerado a partir da directoria

de entrada principal do site. Deste modo é garantido que os ficheiros serão gravados na sua localização correcta.

8.3.3 Sincronização da Base de Dados

A sincronização da informação contida nas bases de dados é um processo um pouco mais complexo, pois existem algumas restrições que requerem um tratamento mais cuidado do problema:

- a) A possibilidade de não se poder atribuir um identificador para a base de dados do Sistema Local idêntico ao identificador da base de dados instalada no Servidor Original,
- b) A possibilidade de existir uma modificação na estrutura da base de dados usada pela aplicação.
- c) A integridade das chaves primárias, nos vários sistemas, nas operações de sincronização.
- d) A manutenção da integridade dos dados segundo a sua temporalidade, resultado das alterações aos dados nas bases de dados usadas pelos vários sistemas.

Um dos problemas mais complexos que podem aparecer em sistemas deste tipo é a introdução de dois registos, com um contexto lógico idêntico, efectuado por dois utilizadores distintos em dois sistemas remotos distintos, como por exemplo, a introdução simultânea de informação idêntica. Na solução aqui apresentada, esse problema não é equacionado.

Um dos aspectos mais importantes relacionado com sistemas deste tipo é a questão de um registo ser actualizado por dois utilizadores distintos em sistemas remotos distintos. Nesta implementação, o Servidor Original representa, por assim dizer, um papel de ponte de comunicação entre os dois sistemas remotos. O atraso provocado pela sincronização entre os dois sistemas remotos, resultante de uma primeira comunicação entre o primeiro sistema remoto e o sistema original a somar ao tempo da comunicação do segundo, poderá levar a sobreposição de informação introduzida por dois utilizadores em sistemas remotos diferentes.

Para solucionar este problema, a data/hora que se usou no desenvolvimento desta solução foi a hora GMT em todos os sistemas. Esta data é guardada e usada à posteriori pelo Servidor Original de modo a manter a ordem temporal de alteração dos registos provenientes dos vários sistemas remotos na base de dados do Servidor Original. No caso descrito no parágrafo anterior, este método garante assim que a informação com que um registo fica no

Servidor Original e nos Servidores Locais é sempre a informação introduzida pelo último utilizador.

8.3.3.1 Equivalência de Identificadores das Bases de Dados

A tabela `icua_db` mantém, no caso do Servidor Original, a informação das bases de dados usadas pela aplicação local. No caso dos Servidores Locais, esta tabela é usada para efectuar a equivalência dos identificadores das bases de dados locais do Servidor Local às bases de dados remotas dos Servidores Originais, às quais os Servidores Locais estão a replicar.

Esta equivalência de identificadores origina um pequeno problema no código usado no desenvolvimento das aplicações. Se por um lado, e como geralmente, a informação necessária para se efectuar uma ligação a uma base de dados é codificada apenas num ficheiro, por questões de não repetição de código, é de lembrar que o sistema aqui desenvolvido replica não só dados, mas também os respectivos ficheiros de código, inclusive aqueles que contem a informação necessária a essa ligação às bases de dados.

Existem alguns métodos para esvaziar este problema, um deles seria, por exemplo, usar um ficheiro de configuração especial que à partida se garantiria que não seria sincronizado, mas isso levava sempre a uma intervenção manual do utilizador para indicar a localização desse mesmo ficheiro e quais os seus valores, exigindo da parte do utilizador um conhecimento que à partida não se pode exigir.

A biblioteca ICu disponibiliza uma solução para quando a aplicação utiliza apenas uma base de dados na sua aplicação, a classe `mICuDB`. Esta classe pode ser acedida em qualquer parte do código, desde que o uso da biblioteca ICu tenha sido incluído no código.

Os dados de configuração da base de dados podem ser incluídas no ficheiro local de configuração local `.Config` na secção `Databases`, como se mostra de seguida:

```
[Databases]
; Database Info
ICUDBLocalServer          = "localhost";
ICUDBLocalDatabase        = "base_dados_local";
ICUDBLocalUser            = "administrador_local";
ICUDBLocalPassword        = "password_local";
```

A biblioteca ICu garante que, quando da instalação inicial do site no Servidor Local, estes valores serão actualizados automaticamente e não serão modificados por qualquer sincronização posterior.

Quando a chamada à inclusão da biblioteca ICu for efectuada, uma das operações processadas pela biblioteca é a verificação da ligação à respectiva base de dados e se os dados estiverem correctos, é iniciada uma ligação.

A partir deste ponto a classe *mICuDB*, põe à disposição do programador a propriedade *mICuDB:: \$hSqlLocal* que representa uma ligação activa à base de dados local.

8.3.3.2 Modificação da Estrutura das tabelas

Em cada lista de envio vem (cf 8.3.1 Ficheiro de Envio) a estrutura completa da tabela. Esta informação vai permitir ao sistema que recebe a lista de envio verificar se a estrutura da tabela local se manteve sem alterações desde a última recepção da lista de envio. Caso existam diferenças, antes de ir sincronizar os dados, o sistema, vai proceder a essas alterações nas tabelas locais.

Para minimizar o impacto que esta operação possa ter no site durante uma operação deste tipo, a biblioteca vai trancar o site até ao fim de toda a actualização da informação que vem na lista de envio, só passando a permitir o seu acesso depois de toda a lista processada, ficheiros e informação de dados. Para que este mecanismo tenha o efeito desejado, a sincronização da informação dos dados é sempre efectuada primeiro que a sincronização de ficheiros.

8.3.3.3 Equivalência de Chaves Primárias entre Sistemas

Um dos maiores problemas na sincronização de tabelas entre dois sistemas cuja operação seja distinta, é a equivalência dos valores das chaves primárias entre os vários sistemas. A solução utilizada, neste caso, passou por criar uma tabela intermédia, *icup_dbkey*, onde são guardados, para cada registo de uma tabela, o valor da chave primária local e o correspondente valor da chave primária remota. É da responsabilidade do processo de sincronização quando recebe um registo novo, actualizar os valores dessa tabela, para que os dois sistemas se mantenham sincronizados.

Este processo, só usado nos servidores locais, serve para manter a coerência dos seus registos com os registos da base de dados que reside no sistema original.

Deste modo, quando é recebido um registo do sistema remoto, o **AProxy** vai consultar a sua chave primária e de seguida verifica se a mesma já se encontra na tabela *icup_dbkey*. Caso já exista, a operação a executar será uma operação de modificação de dados no registo identificado pelo valor da chave primária armazenada na tabela *icup_dbkeys*, caso ainda não exista, o sistema vai proceder à inserção de um novo registo,

registando o valor da nova chave e o valor da chave primária recebido, criando assim um novo registo de equivalência de chaves.

Quando a aplicação Web local cria um registo numa tabela, o novo valor deve dar origem a um novo registo na tabela *icup_dbkeys*, de modo a que quando o sistema for solicitar a criação de uma nova lista de envio, o valor local já se encontre registo na tabela, sendo assim garantido a sua sincronização com o Servidor Original.

Antes de criar o ficheiro de envio e após a recepção do ficheiro do Servidor Local, o módulo *AProxy*, vai também modificar os campos dos registos que são identificados como chaves estrangeiras, usando a tabela *icup_dbkey*.

8.3.3.4 Criação da Lista de Registos a Sincronizar

O sistema mantém uma tabela, *icup_dblist* onde são registadas todas as operações efectuadas nos registos das tabelas. Esta tabela apenas regista, para cada registo, a operação executada, o identificador da tabela envolvida e o respectivo valor da chave primária. A sua operação, para cada tabela da base de dados, pode resumir-se:

- Inserção de um novo registo na tabela – O sistema vai criar um novo registo na tabela *icup_dblist*, guardando o valor da nova chave primária, a data/hora GMT de inserção e o tipo de operação.
- Alteração de um registo – O sistema vai guardar no respectivo registo da tabela *icup_dblist* a data/hora GMT de alteração e o tipo de operação.
- Eliminação de um registo – O sistema vai guardar no respectivo registo da tabela *icup_dblist* a data/hora GMT de alteração e o tipo de operação.

Para que estas operações possam ser executadas, para as tabelas indicadas como passíveis de sincronismo, são criados triggers às operações de inserção, alteração e eliminação de registos. Estes triggers existem nas bases de dados de suporte, quer nos Servidores Locais, quer no Servidor Inicial.

8.4 Administração do Sistema

A administração do sistema de sincronização pode ser efectuada através de um interface disponibilizado pelo módulo *AProxy*, através da chamada ao interface de administração do sistema providenciado pelo módulo *ICuAdmin*. O seu acesso só se encontra disponível no Servidor Original do site.

No ecrã de entrada deverá escolher-se qual o site que se quer configurar, como se mostra na figura seguinte:



Figura 8 - Escolha do site

Após a escolha do site, o utilizador pode escolher entre a configuração do sistema de ficheiros ou das tabelas, com se mostra na figura seguinte:



Figura 9 - Escolha do tipo de configuração

8.4.1 – Configuração de Sincronismo do Sistema de Ficheiros

Ao escolher esta opção o utilizador poderá definir quais os ficheiros que devem ser sincronizados com os Servidores Locais, como mostra a figura seguinte:

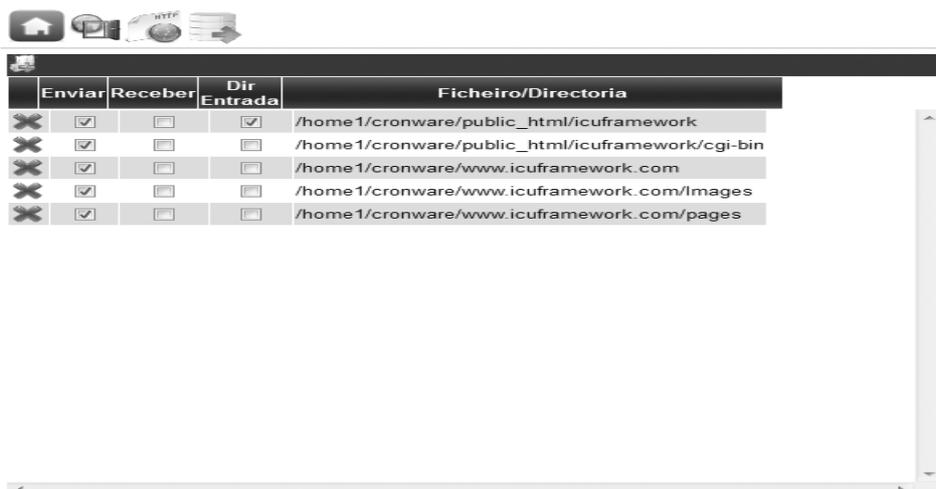


Figura 10 - Definição de exportação de ficheiros

Ao utilizador serão apresentadas automaticamente todas as directorias presentes no site. Esta lista é construída automaticamente pelo aplicativo de configuração, embora possam sempre ser acrescentadas novas directorias.

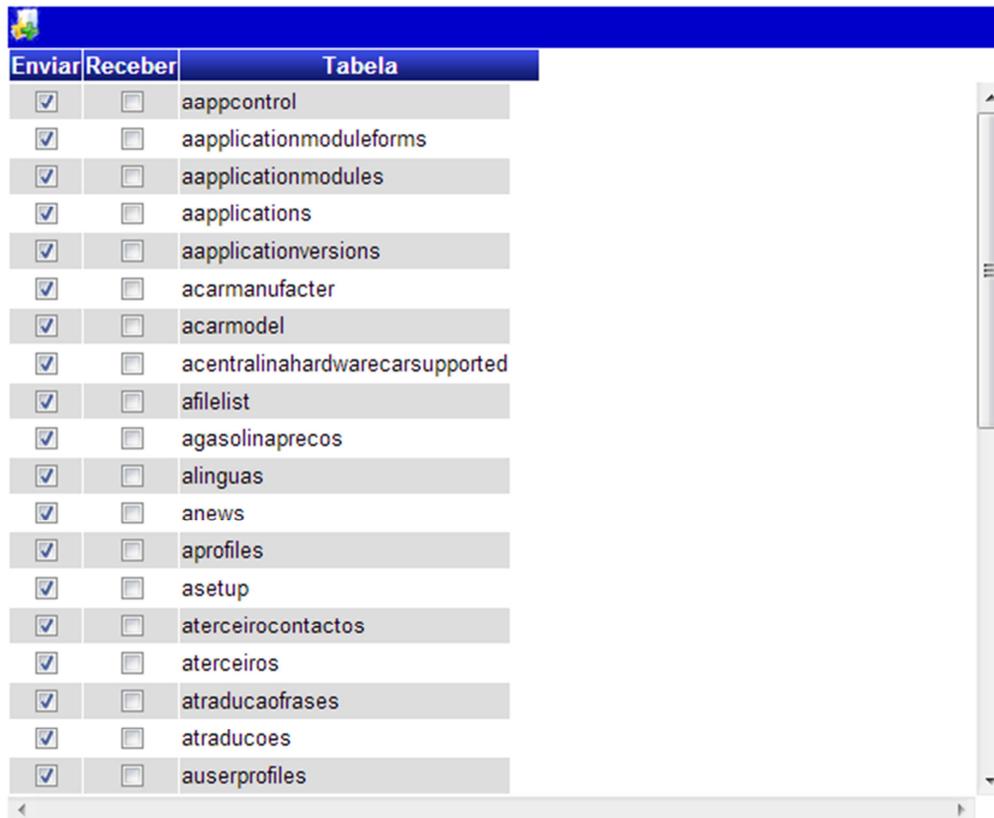
O aplicativo para criar esta lista vai ler todos os ficheiros presentes no sistema local de ficheiros, verificando a sua sintaxe em PHP, interpretando todas as chamadas às funções *require()*, *require_once()*, *include()* e *include_once()*, para assim percorrer toda a árvore de ficheiros do site.

A configuração do tipo de operação a executar na replicação e sincronização nos ficheiros existentes em cada directoria, é a seguinte:

- **Enviar** – Indica ao sistema que todos os ficheiros desta directoria devem ser exportados para os Servidores Locais de cada vez que são modificados.
- **Receber** – Indica ao sistema que todos os ficheiros desta directoria devem ser só importados dos Servidores Locais para o Servidor Original do site. Estas directorias, são directorias que são usadas quando os aplicativos necessitam de operações de upload de ficheiros. Para que, ao importar, não apareçam ficheiros com identificadores iguais, o aplicativo deverá fazer uso do método *AProxy::GetUploadFilename()* da biblioteca ICu. Este método providencia um identificador único, e garante que no momento da exportação do ficheiro, este não vai ter um identificador igual a nenhum outro de outros Servidores Locais.
- **Dir Entrada** – Indica ao sistema qual a directoria raiz de entrada do site. Normalmente esta directoria é identificada automaticamente pelo sistema, mas no caso de não estar correcto, permitir-se a sua correcção.

8.4.2 – Configuração de Sincronismo do da Base de Dados

Ao escolher esta opção o utilizador poderá definir quais as tabelas que devem ser sincronizados com os Servidores Locais, como mostra a figura seguinte:



Enviar	Receber	Tabela
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aappcontrol
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aapplicationmoduleforms
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aapplicationmodules
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aapplications
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aapplicationversions
<input checked="" type="checkbox"/>	<input type="checkbox"/>	acarmanufacturer
<input checked="" type="checkbox"/>	<input type="checkbox"/>	acarmodel
<input checked="" type="checkbox"/>	<input type="checkbox"/>	acentralinahardwarecarsupported
<input checked="" type="checkbox"/>	<input type="checkbox"/>	afilelist
<input checked="" type="checkbox"/>	<input type="checkbox"/>	agasolinaprecos
<input checked="" type="checkbox"/>	<input type="checkbox"/>	alinguas
<input checked="" type="checkbox"/>	<input type="checkbox"/>	anews
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aprofiles
<input checked="" type="checkbox"/>	<input type="checkbox"/>	asetup
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aterceirocontactos
<input checked="" type="checkbox"/>	<input type="checkbox"/>	aterceiros
<input checked="" type="checkbox"/>	<input type="checkbox"/>	atraducaofrases
<input checked="" type="checkbox"/>	<input type="checkbox"/>	atraducoes
<input checked="" type="checkbox"/>	<input type="checkbox"/>	auserprofiles

Figura 11 - Definição de exportação de tabelas

Ao utilizador será apresentado automaticamente, todas as tabelas presentes na base de dados usada pelo site. Esta lista é construída automaticamente pelo aplicativo de configuração.

O aplicativo, para criar esta lista, vai ler as tabelas de sistema onde se encontra a informação sobre a definição das tabelas de uma base de dados.

A configuração do tipo de operação a executar em cada tabela da base de dados, é a seguinte:

- **Enviar** – Indica ao sistema que a respectiva tabela e os dados nela contidos deverão ser exportados para os Servidores Locais.
- **Receber** – Indica ao sistema que a respectiva tabela e os dados nela contidos deverão ser importada dos Servidores Locais.
- **Tabela** – Indica a designação da tabela.

Quando a uma tabela é marcada a opção de Receber, está a ser indicado ao sistema que sobre essa mesma tabela, são efectuadas nos Servidores Locais alterações aos dados.

Na situação anterior, deverá sempre marcar-se ambas as operações, Enviar e Receber, para que a informação alterada num Servidor Local seja replicada em todos os outros Servidores Locais.

9 Servidor Local

Quando um utilizador acede a um site que esteja a usar a biblioteca ICu, e ainda não tiver instalado o Servidor Local, é apresentado ao utilizador um diálogo onde lhe é perguntado se deseja instalar o Servidor Local, para que possa passar a poder usufruir do acesso ao site, mesmo quando o acesso à rede é inexistente.

O módulo **AProxy**, se activo, e com origem no Servidor Original, vai incluir código, em Javascript, na página de entrada, para detectar a existência, ou não, do Servidor Local. O código usado para a detecção dessa existência, implica que o browser usado suporte HTML 5, pois usa o evento *onmessage*, só definido a partir desta versão.

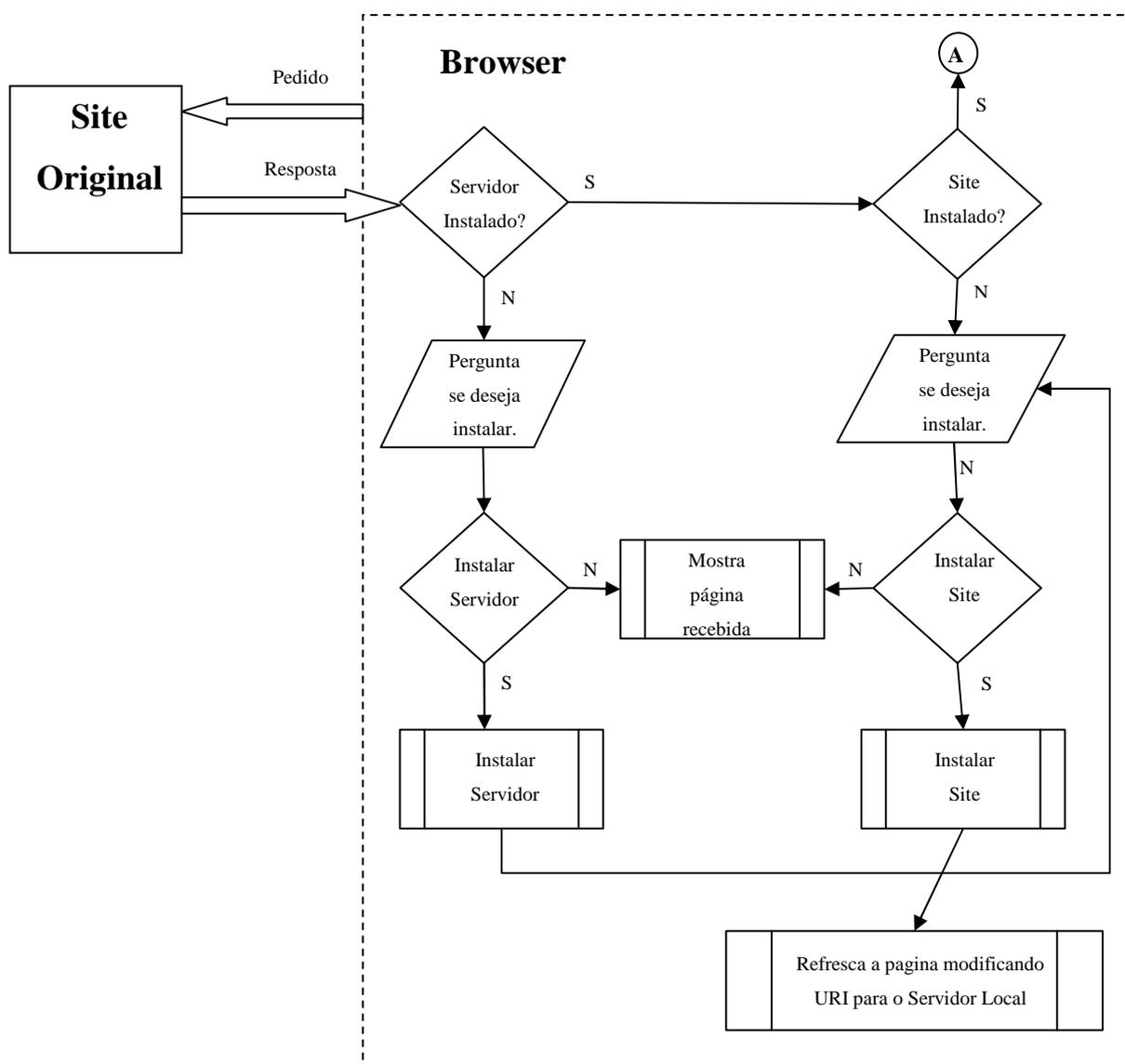


Figura 12 - Instalação do Servidor Local / Site

Como se pode ver na Figura 12, quando o utilizador requisita uma página que se encontra num servidor cujo site foi desenvolvido com a biblioteca ICu, esta inclui código em Javascript que efectua algumas verificações:

- Primeiro é efectuada a verificação da existência de um Servidor Local. Essa verificação é codificada pelo script, criando um objecto *iframe* na página local, cujo endereço é <http://localhost>. Após a interpretação do resultado, caso não exista um Servidor Local, é perguntado ao utilizador se deseja instala-lo, e se confirmado pelo utilizador, instala o Servidor Local
- Como o Servidor Local não insere nenhum script nas páginas que são requisitadas, o script parte do princípio que o mesmo ainda não se encontra instalado no Servidor Local, pedindo se o utilizador deseja instalar o site localmente. Se confirmado pelo utilizador, instala o site no Servidor Local.
- Finalmente o script efectua o reload da página.

O Servidor Local é instalado usando o apache como servidor de páginas e *proxy* e o mysql como base de dados. A instalação é efectuada fazendo o download de um programa de instalação que vai instalar o servidor de páginas no computador local.

Para aceder ao programa de configuração do servidor local, o utilizador deverá digitar no seu browser o seguinte endereço <http://localhost>. O interface da biblioteca ICu será visualizado como mostra a figura seguinte:



ICu Framework

Bem vindo |  Entrar

O que é o ICu?

ICu é um framework para PHP que põe ao dispor do programador um conjunto de ferramentas e propõe um novo modo de criar e usar sites. Com o ICu pode criar sites que, mesmo sem estarem ligados á internet, continuarão a poder ser usados em todas as suas opções, desde entrada de dados a uploads de ficheiros. A partir do momento em que o sistema detecta que existe uma ligação á internet, imediatamente os dois sistemas são sincronizados.

O ICu pode ser usado de dois modos distintos, ou no seu computador pessoal, onde terá que instalar um aplicativo (actualmente só existe ainda a versão em windows) ou num proxy instalado num servidor que dê acesso a um conjunto de utilizadores.

Figura 13- Interface de configuração

Para poder ter acesso à configuração, caso exista, deverá o utilizador premir o botão do rato em *Entrar* e certificar-se como utilizador autorizado. O nome do utilizador de administração que vêm por omissão é *xadmin* e a respectiva palavra-chave é *password*.

No Servidor Local não é possível configurar os dados de replicação/sincronização, mas é possível adicionar utilizadores e se o site tiver opções de configuração local, será aqui que essas alterações deverão ser efectuadas.

O Servidor Local também pode ser instalado numa máquina na rede local e servir como *proxy*, para todas as máquinas. Neste caso, o script que é enviado pelo Servidor Original vai também verificar se o browser está a utilizar um *proxy* e tenta contactar o mesmo para verificar se o *proxy* é um *proxy* que suporta a biblioteca ICu. Se o *proxy* não suportar a biblioteca ICu, o sistema informa que não é possível a instalação do Servidor Local.

O sistema usado para o acesso às imagens dos sites residentes na máquina local, é efectuado, nos sistemas Linux e Windows, modificando o ficheiro *hosts* em ambos os sistemas. O processo usado para que a máquina local aceda directamente a estes sites é

acrescentar no ficheiro *hosts* uma entrada que redirecciona o respectivo domínio do site para o endereço local da máquina (*localhost*).

Foi adoptada esta solução, pois a configuração do browser, para que este passe a redireccionar os seus pedidos por um *proxy*, não é possível directamente.

Conclusão

Na generalidade, considera-se que os objectivos deste trabalho foram alcançados, na medida em que foi possível chegar a uma solução operacional, validando assim a questão inicialmente levantada: é possível garantir a utilização de recursos Web mesmo em caso de falha do acesso à Internet.

A opção adoptada como suporte ao sistema proposto, baseada em software livre, cumpriu os requisitos propostos, permitindo aumentar a fiabilidade do sistema. Por outro lado, o facto de terem sido essencialmente utilizadas linguagens universalmente utilizadas nas principais plataformas de execução aplicacional (Javascript e PHP), foi igualmente assegurada a portabilidade do sistema, assim como a dos aplicativos utilizados.

O desenvolvimento da biblioteca ICu decorreu de acordo com o previsto. Foi instalada e testada num Servidor Original sem problemas e com sucesso. A estrutura arquitectada no sentido de proporcionar facilidade na instalação de novas funcionalidades e correcções a funcionalidades já existentes, foi testada e demonstrou responder às exigências.

Para testar o Sistema Local, foi desenvolvido um site com funcionalidades básicas tendo sido testadas todas as funcionalidades referidas neste documento. Também neste caso o sistema correspondeu às expectativas esperadas:

- Instalação do Sistema Local;
- Instalação de novos sites no Sistema Local;
- Sincronização de ficheiros e das respectivas bases de dados.

Foram detectados alguns problemas relativos à portabilidade do sistema na instalação do sistema local e do serviço responsável pelo sincronismo do Sistema Local com os vários Servidores Originais, devido a especificidades do sistema operativo, neste caso o Windows 7. Esta limitação obrigou à implementação de algum código e operações de forma dependente do sistema operativo, o que de certa forma impede a portabilidade integral do sistema.

Por outro lado, no âmbito deste trabalho não era possível, por questões temporais, desenvolver um sistema completo, ficando por definir e implementar alguns aspectos para que este sistema possa dar resposta a um número maior de casos. Estes desenvolvimentos futuros podem ser incorporados facilmente na biblioteca ICu, pois na sua construção houve o cuidado de criar um modelo em que a introdução de novos módulos é simples.

Assim sugerem-se os seguintes possíveis desenvolvimentos futuros:

- A construção e de listas de acesso à replicação e sincronização, no lado do Servidor Original, de modo a que se possa controlar o acesso ao servidor por Servidores Locais.
- Embora no modelo de replicação/sincronização já esteja contemplado o conceito de domínios, estes não foram implementados. Este conceito é importante para que se possa fazer a distinção entre vários perfis.
- Na construção do modelo de replicação / sincronização foi apenas considerada a actualização directa de um Servidor Local a partir do Servidor Original. A estrutura de dados criada para essa operação foi preparada para que as actualizações possam ser efectuadas tanto directamente a partir do Servidor Original, como a partir de um outro qualquer Servidor Local. No entanto essa funcionalidade não foi implementada, pois a mesma necessitaria obrigatoriamente da implementação do referido no primeiro ponto desta lista. A implementação deste ponto permitiria, entre outras coisas, que os sistemas pudessem ser actualizados, por exemplo, através de uma sequência de máquinas que podiam construir uma cadeia de réplicas até ao Servidor Original que esteve na origem do site.
- O desenvolvimento de uma interface de utilizador, suportada pela biblioteca ICu.
- A notificação aos vários Servidores Locais da existência de alterações dos ficheiros e dos dados, por parte do Servidor Original.
- Quando o browser local está a utilizar um *proxy* que não suporte a biblioteca ICu para o seu acesso à rede, neste momento não é possível a instalação do Servidor Local, pois tal seria ineficaz, uma vez que todos os pedidos são enviados directamente para esse *proxy*. Esta situação deveria ser resolvida.
- Este modelo só válido para sites não seguros, isto é, sites que suportam o protocolo *http*. Embora se possa configurar o Servidor Local para responder a pedidos que usem o protocolo *https*, a questão da unicidade de certificados, inviabiliza à partida a sua utilização. Uma solução para este problema poderia passar pela emissão de certificados pelo site geral da biblioteca ICu (<http://www.icuframework.com>), sendo então estes instalados na mesma operação de instalação do Servidor Local.
- O modelo proposto continua a depender da linguagem usada, tendo sido unicamente desenvolvido para o caso do PHP. Sendo desejável a sua abertura a outras plataformas, uma abordagem possível seria a do desenvolvimento de uma

meta-linguagem de programação, na qual seria implementado o template do programa original, a partir do qual seria gerado o código equivalente na linguagem usada por diferentes instâncias do Servidor Local. Esta solução implicaria um estudo da possibilidade da modelação de uma linguagem universal de desenvolvimento, ou de um conjunto de regras gramaticais que pudessem ser aplicadas a um código origem de modo a poderem funcionar como “tradutores” de linguagem.

Bibliografia

- [1] Global Workplace Analytics. (6/05/2012). Latest Telecommuting Statistics. Obtido em 06/05/2012, de <http://www.globalworkplaceanalytics.com/>: <http://www.globalworkplaceanalytics.com/telecommuting-statistics>
- [2] Jet Propulsion Laboratory (2009). Strategic Technology Directions 2009
- [3] Jim Challenger, Arun Iyengar and Paul Dantzig (1999). A scalable System for consistently Caching Dynamic Web Data. Proc. IEEE INFOCOM 99
- [4] Pei Cao, Jin Zhang and Kevin Beach (1998). Active Cache: Caching Dynamic Contents on the Web. Proc. IFIP International Conference on Distribute Systems Plataforms and open Processing.
- [5] Qiong Luo, Jeffrey F. Naughton, Rajasekar Krishnamurthy, Pei Cao, and Yunrui Li (2000). Active Query Caching for Database WebServers. WebDB (pp 29-34). A revised version appeared in Lecture Notes in Computer Sciences, Vol. 1997, Springer Verlag, 2001: 92-104.
- [6] Ikram Chabbouh, and Mesaac Makpangou (Outubro 2004). Web Content Caching and Distribution. 9th International Workshop, WCW 2004, Beijing, China, (pp. 18-20).
- [7] T.T. Tay and Y. Zhang (Abril 2005). Minimal web patch generation for incremental web Caching IEE Proc.-Commun., Vol. 152, No. 2
- [8] Oracle. (28/04/2013). Obtido em 28/04/2013, de <http://www.csee.umbc.edu:> <http://www.csee.umbc.edu/portal/help/oracle8/server.815/a67781/c31repli.htm#12636>
- [9] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso. Understanding Replication in Databases and Distributed Systems
- [10] Zangeneh, I., Moradi, M., & Mokhtarbaf, A. (2011). The Comparison of Data Replication in World Academy of Science, Engineering and Technology 59, (pp. 315-317).
- [11] Jiandan Zheng, B.S.; M.A. (2008). Universal Data Replication Architecture
- [12] Qiong Luo, Sailesh Krishnamurthy, C. Mohan, Hamid Pirahesh, Honguk Woo, Bruce Lindsay, and Jeffrey F. Naughton (2002). Middle-tier Database Caching for e-Business .SIGMOD (pp 600-611)

- [13] Pei Cao, Jin Zhang and Kevin Beach: Active Cache: Caching Dynamic Contents on the Web. Computer Sciences Department, University of Wisconsin-Madison
- [14] S. Balasubramaniam and B. C. Pierce (Outubro 98). What is a file synchronizer? In Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking(MobiCom '98). Technical report #507, Abril 1998.
- [15] R. G. Guy, P. L. Reiher, D. Ratner, M. Gunter, W. Ma, and G. J. Popek (1998). Rumor: Mobile data access through optimistic peer-to-peer replication. ER '98 Workshop on Mobile Data Access, (pp 254–265)
- [16] N. Ramsey and E. Csirmaz (2001). An algebraic approach to file synchronization. 8th European Software Engineering Conference, ACM Press, (pp 175–185).

APÊNDICES

APÊNDICE I

.

[Groups]

; ICu Generic Global Constants

Global = "GB,Constant";

; Related to Application Info

Application = "AP,Constant";

; Related to Database Info

Databases = "DB,Constant";

; Services List

Services = "SR";

; Modules List

Modules = "MD";

; International

International = "IT,Constant";

; Input

Input = "IP,Constant";

; Stand alone Calls

StandAloneCalls = "SA,Constant";

; Pre Filters

PreFilters = "PR,Constant"

; Active Proxy Calls

ActiveProxy = "APR,Constant";

; Web Services

WebServices = "WS,Constant";

; Menus

Menus = "MN,Constant";

; Users

Users = "US,Constant";

; Application Licenses

Licenses = "LC,Constant";

; Administrator Tools

Admin = "AD,Constant";

; Images

ICuImages = "IM,Constant";

[Global]

; ICu System Directorys

ICUGBDirectoryBaseLibrary = "BaseLibrary/";

ICUGBDirectoryModuleLibrary = "Modules/";

ICUGBDirectoryRemoteAccess = "RemoteSrv/";

; Compact Options

ICUGBCompactJSFiles = 1;

ICUGBCompactCSSFiles = 1;

ICUGBCompactHTMLFiles = 1;

```
; Application Phases
ICUGBAppDevelopPhase      = 1;
ICUGBAppTestPhase         = 2;
ICUGBAppProductionPhase   = 4;

; Serial Number
ICUGBLicenseKey           = "FUDNQANTPKVQCRSDRSIKUVJSDJIUDXXY";

; System States
ICUGBSystemStateHalted    = 1;
ICUGBSystemStateRunning   = 2;
ICUGBSystemStateUpdating  = 4;
; Actual System State
ICUGBSystemState          = 2;

; General Information
ICUGBVersion               = 1;

; Debug System
ICUGBDebugActive          = TRUE;

; Output Stack Type
ICUGBOutputMeta           = 0;
ICUGBOutputCSS            = 1;
ICUGBOutputJS             = 2;
ICUGBOutputJSFile        = 3;

[Application]
; Default Application Parameters
ICUAPAppPhase             = 4;
ICUAPIndexPage            = "index.php,index.html,index.htm";
ICUAPAppVersion           = "";
ICUAPAppName              = "";
ICUAPAppCode              = "";
;ICUAPAutoEntryPage       = "Desktop: Desktop/EntryPoint.php, Tablet:none; Mobile:none"

[Databases]
; Database Info
ICUDBMasterServer         = "localhost";
ICUDBMasterDatabase       = "cronware_master";
ICUDBMasterUser           = "cronware_sysadm";
ICUDBMasterPassword       = "tn040404";

[Modules]
; System Modules (entrance order is important)
mjQuery                   = "jQuery,TRUE"
```

```

mICuError                = "ICuError.inc,TRUE";
mICu                      = "ICu.inc";
mICuIntl                  = "ICuIntl.inc";
mICuLicense                = "ICuLicense,TRUE";
mICuInput                  = "ICuInput,TRUE";
mICuMenu                   = "ICuMenu";
mICuUser                   = "ICuUser";
mICuDB                     = "ICuDB.inc";

```

```
; **** ICu Standalone Call Prefixs
```

```
[StandAloneCalls]
```

```
; Normal Calls
```

```

ICUSAGlobalID              = "ICUSAC";
ICUSALoadStaticCSS         = "LSC";
ICUSALoadDynamicCSS       = "LDC";
ICUSALoadStaticJS         = "LSJ";
ICUSALoadDynamicJS        = "LDJ";
ICUSALoadInitJS           = "LIJ";
ICUSALoadInitLoadJS       = "LIL";
ICUSACleanTmpFiles        = "CTF";
ICUSAShowTmpFiles         = "STF";
ICUSAShowInstModules      = "SIM";

```

```
[PreFilters]
```

```

ICUPRStopAndExit          = 1;
ICUPRStopChain            = 2;
ICUPRContinue              = 3;

```

```
[PreFiltersList]
```

```

LicencePreFilter          = "mICuLicense::PreFilter";
ActiveProxyPreFilter      = "mAProxy::PreFilter";

```

```
[WebServices]
```

```

ICUWSRequestWebService    = "WSRQST";
ICUWSOK                    = 1;
ICUWSError                 = 2;
ICUWSContinue              = 3;
ICUWSOKOnly                = 4;
ICUWSContinueAndClean     = 5;

```

```
[WebServicesList]
```

```
; User Module
```

```
ICUUSRLO                    = "mICuUser::DoLogoff";
```

```
; AProxy Module
```

```
APGUPT                      = "mAProxy::WSGetUpdates";
```

```

APCFLS                = "mAProxy::WSExportFilesInfo";
APUFLS                = "mAProxy::WSUpdateFilesInfo";
APGTTB                = "mAProxy::WSGetTotalBlocks";
APGTBL                = "mAProxy::WSGetBlockContent";
APSLPG                = "mAProxy::WSShowLoadingPage";
APADMN                = "mAProxy::WSAdmin";
APGTST                = "mAProxy::WSGetState";
APGTIF                = "mAProxy::WSGetInstallForms";
APDBPR                = "mAProxy::WSDBProps";
APFLPR                = "mAProxy::WSFileProps";
APFLUD                = "mAProxy::WSFileUpdateDB";

; Licenses Module
USBGRG                = "mICuLicense::WSBeginRegister";
USENRG                = "mICuLicense::WSEndRegister";
USGTRG                = "mICuLicense::WSGetRegister";
USLSAP                = "mICuLicense::WSListApps";
USCHSG                = "mICuLicense::WSCheckSignature";
USCMSR                = "mICuLicense::WSCheckMailSerial";
USCMJS                = "mICuLicense::WSCheckMailSerialByJS";

; Admin Tools
ADSHCP                = "mICuAdmin::WSControlPanel";
ADMCLS                = "mICuAdmin::WSGetModuleList";

; ICu Images
IMGISI                = "mICuImages::WSGetImage";

; Users
USDLGN                = "mICuUser::WSDoLogin";
USDLGO                = "mICuUser::WSDoLogout";
USSRQS                = "mICuUser::WSSkinRequest";

; Input
IPSOOF                = "mICuInput::WSSaveOneField";
IPFRM                = "mICuInput::WSSaveForm";

; File download
FLFLDN                = "mFileDownload::WSGetFile";

; Table
TBGTRW                = "mICuTable::WSGetLines";
TBGTNL                = "mICuTable::WSGetNewLine";
TBGTGD                = "mICuTable::WSGetDelMsg";
TBGTDL                = "mICuTable::WSDelLine";

[International]
ICUITDefaultLanguage = "en";
ICUITDefaultDevLanguage = "pt";

[Licenses]
ICULCDemoLicense     = "DEMOLICENSEKEY";
ICULCSignatureTimeout = 30;

```

```
ICULCLicenseTimeLimit      = "ICULCLTL";
ICULCCheckAndSaveSerial    = "ICULCCASS";
ICULCIDTypeCentralServer   = 0;
ICULCIDTypeAProxyFIPServer = 1;
ICULCIDTypeAProxyDIPServer = 2;
ICULCIDTypeAProxyClient    = 3;
ICULCIDStateToVerify       = 0;
ICULCIDStateVerified       = 1;
```

```
; #####
```

```
; #####   MODULES   ###
```

```
; #####
```

```
[ActiveProxy]
```

```
APRRunCmd                  = "APRNCMD";
APRLoadDBChanges           = "APDBCH";
APRLineType                 = "APRLNTP";
APRLineTypeTPN             = "APRLTPM";
APRLocalServerDir          = "%APR%/wwwroot";
APRProxyServer              = 1;
```