Trabalho Final de Curso

71 - Plataforma de Transferência Segura de Ficheiros Client/Server

Relatório

Realizado por:

- **Duarte Gaspar (a20080589)**
- **Tiago Quadrado (a20082952)**

Lig / Lei - 2011/2012

Índice

Índices de Quadros, Figuras, Tabelas

Resumo

Resolver um problema no intercâmbio de informação entre os Sistemas Centrais e as Lojas.
Uma empresa que tem a sua actividade na área do retalho alimentar.
Os Sistemas de Informação necessitam enviar e receber milhares de linhas de informação.
Existe a necessidade de enviar informação sob a forma de documentos estruturados, dos sistemas centrais para as várias lojas; das várias lojas para os sistemas centrais.
Estes documentos destinam-se a: actualização diária os preços dos produtos nas lojas; receber nos sistemas centrais as existências de stock nas lojas; enviar para as lojas as transferências de produtos; receber nos sistemas centrais a informação sobre as vendas dos produtos nas lojas; as picagens de ponto dos colaboradores nas lojas.
Por dia, os documentos a transferir poderão chegar às várias dezenas de milhar, dependendo do número de lojas, do número de produtos por loja e do volume de vendas por loja. Estes documentos são extremamente críticos para o negócio, estando na base do processamento da informação do negócio.
Existe a necessidade de haver grande rapidez na transferência desta informação, de uma forma segura e confiável.
O que se pretende atingir é uma melhoria de 70% no tempo para o transporte dos ficheiros.

Abstract

Solve a problem in the exchange of information between the Central System and the systems in the Stores.

A company that has its activity in the area of food retail.

Central Systems need to send and receive thousands of lines of information. There needs to send information in the form of structured documents, systems central to many shops, several of the stores to the central systems.

These documents are intended to: daily update product prices in the shops; receiving systems in central stocks of stock in stores, shops for sending shipments of products; receiving information in central systems on sales of products in stores; chipping point of the employees in the stores.

By day, the documents to be transferred may reach the tens of thousands, depending on the number of stores, the number of products per store and sales volume per store. These documents are extremely critical to the business, as the basis of business information processing.

There is a need for large transfer of this information quickly, securely and reliably.

The aim is to achieve a 70% improvement in the time taken for transporting the files.

1. Introdução

A troca de informação sob a forma de ficheiros estruturados destina-se a tornar mais seguros os sistemas centrais que se estivessem expostos em demasia poderiam colocar em risco processos “Core” do negócio.

Se os sistemas centrais estiverem expostos a integrar informação directamente da fonte (por exemplo um software de frente loja) estes podem ficar fragilizados, pois: a integração de movimentos errados leva à integração também de movimentos de correcção; a rede na loja compreende uma infra-estrutura wireless que em último recurso pode ser violada.

A existência de sistemas considerados no todo como um ETL (a nossa solução pode ser considerada como uma parte deste sistema) trás, para além da sistematização da informação, mais um nível de protecção aos sistemas centrais criando validações intermédias.

Supõe-se que na integração desta informação irá ser efectuada validação da mesma (processo não constante desta solução).

Existem perguntas que devem ser respondidas em primeiro lugar: Quantas vezes vão integrar as vendas por dia? Como estimamos os stocks nas Lojas?

A informação com origem na loja destina-se a ser integrada nos sistemas ERP e nos sistemas de Data Warehouse. A informação com destino às lojas tem origem no ERP e nos sistemas de Logística (Warehouse e processos automáticos de Replenishment).

O objectivo é desenvolver software para transferência segura e rápida de milhares de documentos, que se destinam a ser integrados em sistemas de informação.

Este software destina-se a ser usado em ambiente empresarial para transporte de ficheiros informáticos entre os vários sistemas.

2. Enquadramento Teórico (com revisão bibliográfica)

Como referenciado no livro "Distributed Systems: Concepts and Design" de Coulouris, Dollimore & Kindberg, num Sistema Distribuído é necessário ter em conta algumas características:

- Heterogeneidade, capacidade para suportar a variedade e a diferença;
- Abertura, Característica de um sistema que determina a sua capacidade de extensão e de modificação;
- Segurança;
- Escalabilidade, propriedade de um sistema que indica a sua capacidade em responder de forma linear a aumentos significativos da carga, número de utilizadores, etc.;
- Gestão de Falhas;
- Concorrência, existência de vários fluxos de execução implica acessos simultâneos a recursos partilhados;
- Transparecia, capacidade de esconder do utilizador e das aplicações a natureza distribuída do sistema;

O nosso Sistema irá operar sobre uma camada Midleware instalada, neste caso Java, para que consiga suportar e utilizar vários recursos Hardware e Sistema Operativos diferentes (Heterogeneidade).

Será possível aceder realizar modificações ao código sem por em causa o restante funcionamento do Sistema pois será implementada uma camada de "Interface" (Abertura).

Utilização de mecanismos de segurança específicos para garantir a Confidencialidade (Tecnologias de encriptação como chaves Assimétricas);

Autentificação para garantir quem é o utilizador com base em login e Certificados Digitais, definição de Politicas de Acesso para garantir a devida Autorização (Segurança).

Para garantir que a informação enviada pelos intervenientes é realmente aquela gerindo "Digest" criados por algoritmos de hash (Integridade);

Não-Repudiação ou seja garantir que a informação recebida por B foi mesmo enviada por A, utilizando certificados Digitais ou PKI (Public Key Infrastructure);

Privacidade, utilizando encriptação de ficheiros.

Escalabilidade, serão utilizados mecanismos de controlo de Threads para definir um numero de quantas deverão ser utilizadas tendo em conta os recursos da Maquina Local e mecanismos de comunicação que poderão aceitar e gerir várias comunicações evitando "Overflow . A aplicação também irá conter um algoritmo de ordenação de ficheiros e dados para que consiga da melhor forma responder atempadamente os pedidos.

Gestão de falhas, registo de todas as falhas em "logs" para o Administrador analisar sempre que necessário, em caso de Overflow de threads ou de processos, o sistema encontra-se dividido em camadas para suportar um arranque imediato e continuo da sua função. Sempre que necessário poderão ser realizadas actualizações para que possam corrigidas as falhas. As actualizações serão notificadas ao cliente tendo este a obrigação de realizá-las antes da utilização da sua plataforma.

Utilizará mecanismos para detectar falhas, registar, "mascarar" utilizando a retransmissão de pacotes por exemplo, será tolerante a falhas o suficiente para continuar o serviço ou recuperar.

Concorrência, o SD será capaz de aceitar simultaneamente os pedidos, diferencia-los por tipo e cliente e responder atempadamente evitando o sistema parar até que seja resolvido (utilização de monitores ou Semáforos).

Transparência, ira permitir acessos locais e remotos através de operações idênticas, acesso a recursos sem conhecimento da sua localização física ou do endereço ao mesmo tempo poderão estar vários processos a funcionar simultaneamente sem interferências e isolamento das falhas fazendo com que os utilizadores possam completar as suas tarefas.

Web Service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Permite com que os recursos da aplicação do software estejam disponíveis sobre a rede de uma forma normalizada.

Utilizando a tecnologia WebService, uma aplicação pode invocar outra para efectuar tarefas simples ou complexas mesmo que as duas aplicações estejam em diferentes sistemas e escritas em linguagens diferentes. Por outras palavras, os WebServices fazem com que os

seus recursos estejam disponíveis para que qualquer aplicação cliente possa operar e extrair os recursos fornecidos pelo WebService.

O sistema irá operar com WebServices nomeadamente Mysql utilizando o protocolo SOAP, no entanto sempre que for necessário criar um novo será implementada essa função.

Estas referências foram recolhidas do livro “Web Services: Principles and Technology”, de Michael Papazoglou.

3. Método

a. Análise

Situação Actual

Necessidade de comunicar vários milhares de iDocs (ficheiro com informação estruturada), dos sistemas centrais para as lojas da empresa e das lojas para os sistemas centrais.

O problema actual prende-se com a demora na entrega destes ficheiros, especialmente em alturas de congestionamento, devido ao grande volume e à reduzida capacidade de entrega destes ficheiros.

O processo actual compreende uma solução que usa FTP para esta transferência. Os ficheiros são enviados um de cada vez e em "plain text". Os vários sistemas de loja enviam por ftp os ficheiros para os sistemas centrais, fazendo validação de segurança com um utilizador e password. Actualmente quem conhecer o utilizador e a password pode reproduzir este ftp a partir de qualquer equipamento na rede.

Existem óbvios problemas de performance e de segurança.

Antes de avançarmos é necessário especificar alguns dos processos de negócio mais afectados.

Depois das lojas fecharem, ficheiros contendo os detalhes das vendas (idocs de vendas) são enviados para os sistemas centrais. As várias centenas de lojas enviam, na mesma janela de tempo estes iDocs. Demoram cerca de 1 hora e meia a estarem todos disponíveis nos sistemas centrais.

Posteriormente estes iDocs são integrados no ERP e no Data Warehouse, para imediato processamento. Estes processamentos nocturnos dão origem a relatórios de vendas e actualizam as tabelas de reaprovisionamento automático, entre outros processos.

Os dados para reaprovisionamento são durante manhã confirmados pela loja; de seguida são construídos novos iDocs a enviar aos fornecedores com as novas necessidades de entrega de produtos e aos armazéns para identificação do que devem distribuir para as lojas.

Se bem que com os bens não perecíveis o tempo não é considerado crítico, pois existe algum stock em armazém; para os bens perecíveis o factor tempo é crítico, pois devem ser entregues pelo fornecedor ao armazém no mesmo dia, para na manhã seguinte serem entregues nas lojas.

Quando existe actualização de características dos produtos (p.ex. preço, imposto, quantidades a enviar para a loja) estas são comunicadas aos sistemas de loja por meio de iDocs que tem origem no ERP.

Se existe uma alteração de características numa família de produtos, por exemplo alteração de taxa de IVA, esta é comunicada a todas as lojas no final do dia para que no dia da aplicação da nova taxa tudo esteja legal. Esta comunicação normalmente envolve muitos milhares de iDocs que demoram algumas horas a serem enviados.

Outras situações ocorrem com a descontinuação de produtos ou a abertura de novos produtos nas lojas; abertura de novas lojas, com o envio de todos os artigos para a loja; etc.

Torna-se pois relevante desenvolver um sistema que tenha performance e que seja seguro. Isto vai permitir que outros processos core ao negócio possam ser iniciados mais cedo, no sistemas centrais, libertando capacidade de processamento para novos processos.

Optimização deste transporte pode levar à optimização da capacidade de processamento actual, nomeadamente no ERP e no Data Warehouse, podendo posteriormente ser traduzidos em redução de custos.

Outro dos benefícios que pode advir de ter um sistema de transporte rápido é a capacidade de poder entregar no Data Warehouse as vendas parcelares (ao longo do dia) para produção de relatórios para análise análise (p.ex. verificar o sucesso ou não de uma campanha de promoção ao longo do dia).

Descrição e Identificação de Objectivos

Fazer uma comunicação segura e com maior performance dos documentos a transferir.

Compactar a informação a transferir para optimizar a ocupação da rede.

Criar um Serviço Central para a comunicação entre os vários sistemas.

Aceder via web a status de serviço e de transferências.

Conseguir um nível elevado de disponibilidade de serviço.

Garantir que não se perdem documentos na comunicação.

Fazer a gestão dos utilizadores que são usados na comunicação (processos automáticos) e na gestão da aplicação (administração/operação pelas equipas responsáveis.

Requisitos

A solução deve funcionar em computadores com Sistema Operativo Windows (nas lojas) e com alternativa Sistema Operativo Linux (para os sistemas centrais).

A nomenclatura de utilizadores e passwords deve respeitar as politicas existentes na empresa.

A linguagem em que se vai desenvolver a aplicação é o JAVA.

Os iDocs (ficheiros com informação estruturada) devem ficar guardados depois de enviados.

Todos os status das comunicações ou erros devem ser guardados em ficheiros de log, podendo ser consultados mais tarde. Estes podem ainda servir para alimentar um software de monitorização/alerta para a equipa de operação.

Esta aplicação não verifica/valida o conteúdo dos iDocs.

Software multi-thread capaz de suportar grande volume de envio e recepção de documentos.

Disponibilização de serviço https para consulta de logs e status de serviços.

Replicação da informação quando um cliente deixa de estar disponível.

Garantir que se cumprem princípios de Autenticação, Confidencialidade, Autorização e Integridade.

Por Autenticação entendemos que há que garantir que todos os intervenientes são quem dizem ser e que estão registados.

Confidencialidade, como garante da de que só os intervenientes têm acesso à informação.

Pelas correctas Autorizações garantimos que só tem acesso à informação quem deve ter.

Este processo deve garantir que no transporte da informação nada se perde, é transformada ou adicionada. Garantir que se entrega o que foi enviado. Garantir integridade.

Este Sistema deve comportar uma gestão de chaves e certificados digitais de modo a tratar toda a informação da maneira mais confidencial e segura possível. Garantir a possibilidade de adição de serviços e clientes quando necessário.

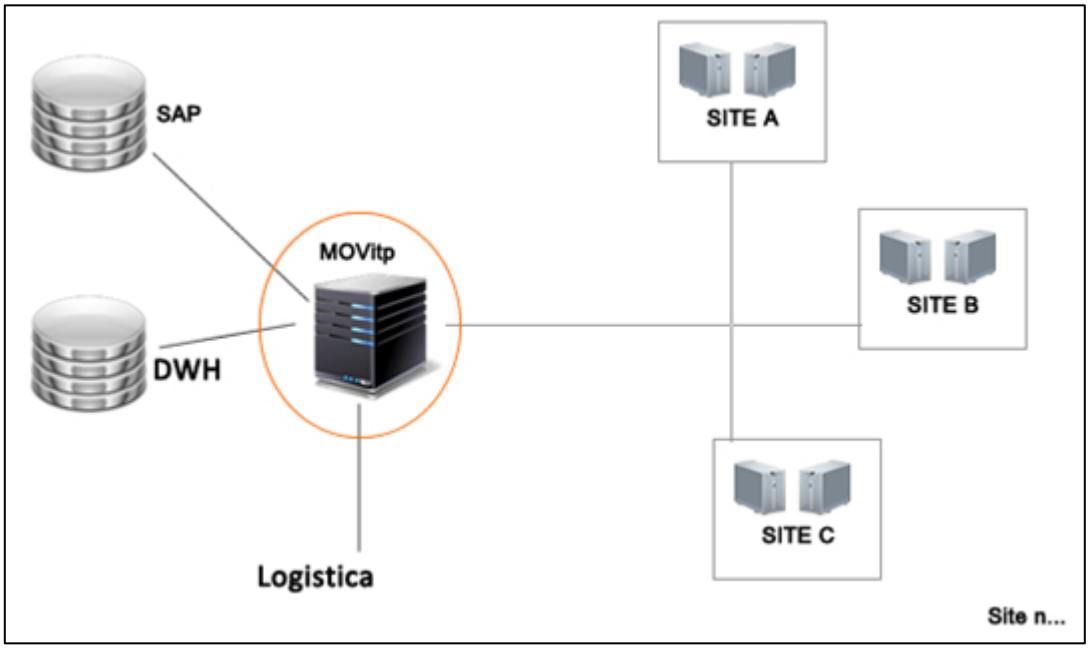
b. Concepção

O software a desenvolver foi batizado de MOVitp. Nome não registado mas validada o seu não uso através de diversas pesquisas na internet.

Aquitectura

Esta é uma arquitectura distribuída que implica a existência de uma solução Servidor Central, gestora de todos os intercâmbios de iDocs e de uma ou várias soluções Cliente, que recebem e enviam iDocs para o Servidor.

Do lado direito da figura em baixo temos as várias soluções centrais que se ligam ao servidor da aplicação (MOVitp); do lado esquerdo temos os vários sites, soluções clientes.



Arquitectura de Componentes

Arquitectura de Camadas

Uma aplicação em N camadas, é uma aplicação desenvolvida de forma a ter várias camadas logicas. Cada camada é contida o suficiente de modo que a aplicação possa ser divida em vários processos em máquinas tendo assim uma rede distribuída.

A nossa arquitectura estará divida em várias camadas.

(1) Camada Interface (Camada de Apresentação)

Cliente, nesta camada operam as funções base para efectuar o login, irá estar dividida em Cliente e ClienteWork.

"Cliente" apenas permite procurar o servidor Movitp, realizar o login, procurar actualizações e efectua-las.

"ClienteWork", podendo ser substituída a pedido do Servidor Movitp como actualização, realiza operações de comunicação com o Servidor Movitp e outras funções.

Manutenção

Esta camada irá operar como interface para utilizadores especificamente definidos para acesso ao Servidor Movitp para alterar configurações específicas do Servidor.

(2) Camada Controlo

Operam todas as funções básicas de segurança e comunicação com todos os intervenientes.

A mesma sempre que necessária pede informação às Camadas de Gestão e de Dados para responder aos seus pedidos.

(3) Camada Gestão

Nesta camada opera todas as funções base para configuração do Servidor, criação de contas clientes ou outras, acesso a dados dos serviços Web como por exemplo Mysql, configurações Ips, criação de Certificados etc., definição de portas e ips para acesso específico, e em caso de Overflow da Camada Controlo, recupera novamente.

(4) Camada Dados

Contem todos os dados Soap em xml para serem exportados. Nesta camada contem todos os ficheiros guardados. São criadas as listas de prioridades e ordenadas através de um algoritmo de ordenação como por exemplo QuickSort.

Arquitectura de Camadas

Plataforma Servidor

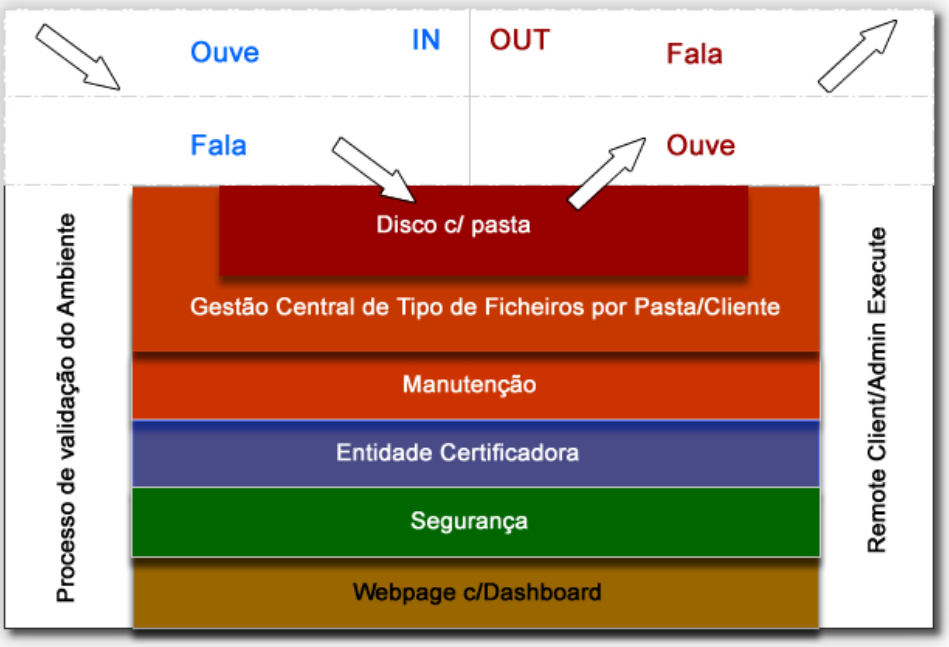
A plataforma Servidor inclui todas as funcionalidades da plataforma cliente e é acrescida de: Função de Gestão; Função de Controlo e Função de Dados.

A Função de Gestão faz a gestão de toda a plataforma servidora. É aqui que a configuração é feita; é aqui que os administradores poderão desligar e lançar novos processos; é aqui que são criadas as várias contas de exploração; é aqui que são criados certificados de segurança; é aqui que é feita a gestão de IPs banidos. As actualizações de código são também geridas por esta função.

A Função Gestão é realçada pela relação entre o utilizador e o sistema. Gestão e Manutenção.

A Função Controlo é uma função que corre em background e que com alguma inteligência faz a validação das comunicações entre Cliente e Servidor e entre Servidor e Cliente. Esta função é responsável pela comunicação segura; pela encriptação e desencriptação dos ficheiros; pela autenticação dos utilizadores e sistemas clientes.

A Função de Dados é onde vamos ter o método de acesso à base de dados e os SOAP Webservices. A base de dados que implementamos é o mySQL. É aqui que se controla o registo e leitura da informação em Base de Dados.



Camada Controlo

Comunicação

Aguardar por comunicações vindas do cliente ou do Servidor.

A) Comunicação recepcionada do Cliente.

Como a comunicação foi aceite e é possível ler, então iremos ler o pedido e registar a sua Key. Lemos o pedido, se for LoginManutenção ou LoginCliente executamos a função LoginManutenção(); que irá devolver o respecitvo resultado, se PedidoFicheiro ou seja o Cliente pede um ficheiro ao servidor então o mesmo ira chamar a função PedidoFicheiro();, se recebe um comando vindo da area Manutenção ou Cliente então devolve a resposta, se Repositório então o servidor irá executar a função repositório que tem como função armazenar os ficheiros que o cliente irá enviar e por ultimo se for uma actualização();

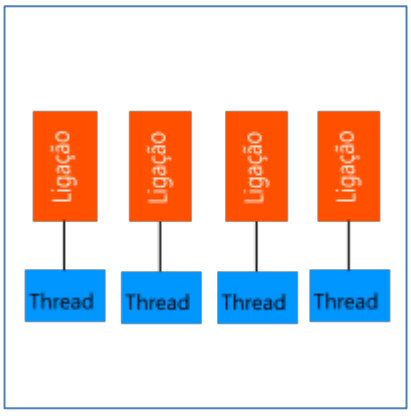
B) Comunicação Servidor

O servidor poderá devolver vários pedidos de resultados de funções como loginCliente ou comandoManutenção, sendo este caso enviar o resultado ao interveniente, caso não foi enviado regista o pedido e volta novamente a enviar.

No entanto o servidor poderá também colocar pedidos seus ao cliente, sendo que este envia também da mesma forma um pedido ao cliente sendo depois enviada para o servidor como RespostaParaServidor();

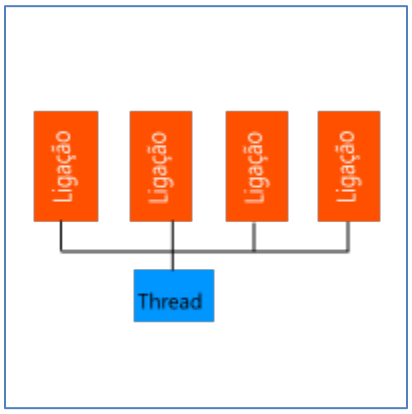
Como irá o servidor gerir as ligações?

Existem vários Design de Arquitectura de Servidores, entre os quais:



Conexão Thread por Ligação

A criação de uma Thread por cada ligação aceite pelo servidor permite um melhor controlo, mais fácil de programar, é necessário mecanismos como Semáforos para tratar da troca de Threads, no entanto poderá consumir imensos recursos da Maquina local, provocando o overflow de Threads não traduzindo numa melhor escalabilidade para o nosso Sistema Distribuído, pois deverá aceitar inúmeras ligações e saber interagir com elas da melhor forma.



Coneção 1 Thread N-ligações

Este desenho de Arquitectura de servidor permite manipular variadíssimas ligações numa só thread. Como é possível? O servidor utiliza Key(); e Seletors(); atribuindo a cada ligação uma key ao selector. Separa por estados de Connectable (Ligar o Cliente ao Servidor), Accept (O servidor aceita ligações de Cliente), isReadable (Possibilidade de ler), isWritable (Possibilidade de escrever). Para cada ligação para determinar qual o seu estado serão registados como OP_CONNECT; OP_ACCEPT; OP_READ; OP_WRITE;.

Este tipo de desenho permite de facto manipular imensas ligações numa thread, mas como desvantagem revela-se de facto mais difícil a sua programação pois é necessário registar todos os estados de cada cliente e manipular os seus pedidos utilizando a sua key, registar em maplist enquanto outra key esta a ser lida/escrita, requer um pouco mais de tempo para a sua programação.

(Ver Anexo – Fluxograma Comunicação Servidor)

Segurança

(Ver Anexo 5,6,7,8)

Camada Gestão

Na camada gestão a aplicação do Servidor irá imprimir para o ecra um menu com um conjunto de opções, sendo elas importantes para o funcionamento de todo o sistema.

O utilizador poderá escolher se pretende iniciar a aplicação ServidorMovitp, mas terá que ter anteriormente realizado todas as configurações basicas como por exemplo certificados para clientes e keystores, contas de clientes e atribuição de perfis, criação de Sites, configuração dos serviços webservices (MySql).

Quando for seleccionada a opção "Iniciar", irá proceder ao arranque dos processos de Controlo e de Dados.

O utilizador poderá sempre que necessário criar contas e certificados, no entando, a alteração de configurações da máquina local (threads, hora e data) necessitam de reiniciar o sistema.

Camada Dados

Numa primeira fase após o arranque a camada dados irá utilizar um algoritmo baseado em quicksort(), para ordenar e indexar todos os ficheiros em prioridades tendo como base a data.

São executados todos os serviços Webservices configurados.

Plataforma Cliente

Definições:

- Destino é o local físico onde posso ter vários equipamentos com o software cliente instalado.
- Cliente é o equipamento que no destino tem o software cliente.
- Central é o local físico onde está o Servidor.
- Servidor é o equipamento onde está instalado o software servidor.

Funcionalidade do Cliente

Os Clientes podem ter vários sistemas operativos. Vamos identificar e aprofundar os ambientes Windows e Linux, nomeadamente com as versões Windows 7 Ultimate e OpenSuse.

O Servidor identifica os clientes em destinos por um mapa de IPs, cruzando cada um com o certificado.

O Cliente aceita o servidor pela identificação de nome e certificado.

Não é pelo simples facto de um cliente ter o software instalado que vais conseguir comunicar com o servidor.

Só após actualização da tabela de IPs clientes no servidor, os clientes podem ser reconhecidos pelo ambiente e ter início e processo de comunicação.

O facto de um destino ter um ou vários clientes só beneficia o destino na possibilidade de ter maior disponibilidade. Quero dizer, quando existe mais do que um cliente no destino, só um cliente comunica com o servidor. Os outros estão em standby. Só entram no processo de comunicação quando o cliente activo deixa de ter a possibilidade de comunicar com o servidor. Neste momento o servidor vai procurar no mapa de IPs do destino, clientes em standby. O primeiro encontrado nestas condições inicia processo de sincronização e entra em modo activo. Todos os outros clientes do destino, entram em modo de sincronização com o cliente activo no destino e em modo standby com o servidor.

Desta forma, replicando os dados de comunicação entre o cliente activo e client standby de um destino, mantemos a possibilidade de a todo o instante substituir um cliente por outro, sem perda de dados e sincronia com o servidor. Tentamos assim aproximar o mais possível à disponibilidade 100% e à falha 0%.

O Cliente e o Servidor podem ter Timezones diferentes. Assim deve ser sempre calculado o ajuste de Timezone para identificar qual o ficheiro que foi transferido em que altura.

O Cliente assim que inicia deve fazer análise de largura de banda de rede com o servidor. Esta análise deve ser repetida a cada unidade de tempo (parametrizável), se não houver comunicação. Se a comunicação existir, deve ser feita análise aos dados enviados/recebidos por unidade de tempo para ajustar os processos de comunicação com o servidor.

Neste processo de iniciação do cliente deve ser validado o tipo de hardware: que CPU? Quantos cores (CPU)? Que memória? Qual o espaço disponível para o directório aplicacional? A existência de link com o servidor. Qual a latência entre cliente e servidor?

A cada unidade de tempo (parametrizável) deve ser feita análise de CPU disponível, Memória disponível e Disco disponível. Estes dados são registados para determinar modo

tipo de execução para o cliente. Este processo permite ajustamento de execução para diferentes hardwares.

O ajuste automático do número de threads a estabelecer com o servidor deve ter em conta o seguinte:

- Largura de banda de rede;
- Número de Cores;
- CPU disponível;
- Memória disponível;
- Quantidade de ficheiros a enviar/receber;

Características técnicas do Cliente

Equipamentos a funcionar com:

- SO 64 bits – Windows 7 Ultimate ou OpenSuse Linux;
- Processador Intel i5 (Core 2 Dual Hyperthreading – Base ; recommended Core 2 Quad)
- 4 GB RAM;
- 1 Disco SATA 7200 RPM 500 GB;
- Dependendo do SO, 2 ou mais partições. A Partição para a aplicação cliente deve ser independente do SO, pois aqui vão ficar não só os binários mas também toda a estrutura de directórios e ficheiros.
- Placa de Rede 100MB

Estrutura de Directórios criadas pela aplicação na primeira vez que arranca.

Windows:

D:\MovitP

(Directoria geral da aplicação)

\archive

(ficheiros dados comprimidos com mais de xx dias que passam de \data\ para aqui)

\in\

\type\

\<day date>\

\out\

\type\

\<day date>\

\bin\

(ficheiros de executáveis da aplicação)

\data\

(ficheiros dados em comunicação com menos de xx dias)

\active\

\in\

\type\

\<day date>\

\out\

\type\

\<day date>\

\passive\

\in\

\type\

\<day date>\

\out\

\type\

\<day date>\

\etc\

(ficheiros de configuração da aplicação)

\log\

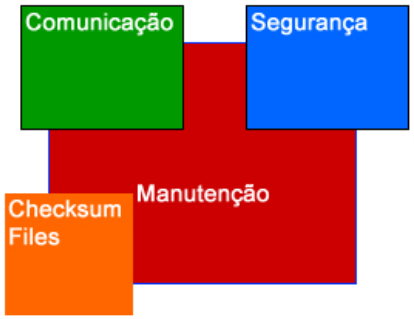
(ficheiros de registo de audit dos vários processos da aplicação)

Linux:

As mesmas directorias que em Windows.

Processos

O programa irá funcionar em 4 Processos: Processo de Manutenção, Segurança, CheckSumFiles e Comunicação.



Manutenção

Consiste num processo que irá suportar toda Aplicação, realizará tarefas de administração, gestão, controlo, será uma entidade de Inteligência Artificial, a qual será afectada por vários aspectos envolvidos ao ambiente da aplicação, Aspectos como largura de banda, numero de ligações Cliente ao Servidor, carga CPU etc. Tendo em conta estes factores que iram influenciar, irá tomar todas as decisões necessárias.

Manutenção será composto pela interface de:

Administração:

Gestão de Ips banidos e Domínios

Limpas Logs

Enviar Actualização a todas as Maquinas

Monitorizar Threads de toda aplicação

Aviso de Manutenção do Sistema (OFFLINE)

Definições Maquina Local:

Horas/Data

Consultar Logs.

Ficheiros: (Ficheiro -> Nome -> Categoria exemplo: Nome: Pree2012 Categoria: Preços

Alterar/Criar Novo Ficheiro (Nome, Categoria, Descrição, Data Criação)

Alterar/Criar Nova Categoria de Ficheiro (Nome, Descrição)

Lista de Ficheiros: Ilustrar lista de ficheiros por tipo e qual a localização a que deverá recolher.

Actualizados, desactualizados.

Data de Atraso: definir números de dias antes do dia actual.

Clientes:

Alterar/Criar Novo (Nome, Localização (numero), Localidade (zona), Morada, IpFixo, Domínio, Responsável Técnico, Seleccionar Ficheiro a pedir ou Categoria)

Consultar Lista

Iniciar, Reiniciar, Desligar

Caso 1) O servidor Inicia-se, irá aparecer a interface e sempre que terminar qualquer alteração deverá no fim Reiniciar. É importante este passo para que possa sempre reiniciar também os processos de Comunicação, Segurança, e CheckscumFiles, com as novas alterações.

Caso 2) Sempre que o Sistema tiver necessidade de parar as comunicações, antes deve proceder ao envio de uma mensagem avisando da necessidade de manutenção.

Se não for efectuada nenhuma alteração poderá apenas Iniciar.

CheckSumFiles

Este processo irá analisar se cada ficheiro se encontra devidamente integra.

1º) Verificar que todos os ficheiros que estão na pasta são apenas os pedidos na lista de Tipo. pre, etc. todos os outros, são colocados em "Others". Verificar por extensão e nome ficheiro.

2º) Irá consultar a lista de ficheiros estabelecida pelo processo de manutenção e que por sua vez criou um log (Manutenção -> logListaFicheiro) que contem todas as listas de ficheiros com a localização que deverá pedir.

Ficheiro LogListaFicheiro

Ficheiro	Localização	Estado	Data
Pre02020120425...	020	False (Não Enviado)	----------
Pre02220120425	022	True(enviado)	2012-04-25

e com os ficheiros = False então o LogListaPedido, contem todos os "ficheiros" que ainda não recebeu.

(este ficheiro foi criado para diminuir o numero de queries de pesquisa ao ficheiro principal LogListaFicheiro, isto é sempre, que houver uma condição False, o processo checksumfiles, automaticamente cria e/ou altera o ficheiro LogListaPedido apenas com os ficheiros false, listando apenas os false irá ajudar na pesquisa rápida);

Ficheiro LogListaPedido

Pre02220120425	022	False	-----

Depois do Processo de Comunicação receber em buffer e criar no disco do Movitp o ficheiro "Pre02220120425" o checksum actualiza o ficheiro LogListaFicheiro o estado para True. Não será necessário o mesmo processo actualizar o LogListaPedido Pois o LogListaPedido é um looping de todos os "ficheiro" em estado "False", a cada 1 segundo.

O processo de Comunicação irá sempre consultar o ficheiro LogListaPedido a cada 40 Minutos.
(Este tempo reserva-se para quaisquer tentativas entre o cliente e o Movitp,de comunicação ou problemas na largura de banda (lentidão)).

3) Calcular por data quais os mais actualizados e Desactualizados e informar o processo de Manutenção quais os actualizados e desactualizados. Serão considerados desactualizados todos os ficheiros que estão a mais da data pedida. Exemplo 1 dia de Atraso, 7 dias de Atraso.
LogListaActualizados
LogListaDesactualizados =>LoglistaPedidos =False

4) No início do dia seguinte comprime, todos os ficheiros tratados do dia anterior, em zip file.

Comunicação
O processo de Comunicação será o intermediário das comunicações entre o Cliente, o processo de Segurança e o checkSumFiles

1) Algoritmo de Controlo:

1. Criar -> Ouve -> Thread 1(Start) e Thread 2(stop), em semaforo, quando exception ou overflow então thread 1 Stop e Thread 2 Inicia Depois X tempo. e vice-versa;

2. Fala -> Thread 1 (Start) e Thread 2 (Stop),em semaforo, quando exception ou overflow então thread 1 Stop e Thread 2 Inicia Depois X tempo e vice-versa.

Ouve

Thread1 >

Cria Socket com a porta TCP 8000 e TCP 8001 para receber pedidos de ligação, regista em hastable=”StatusClientes” Ip e em buffer os dados de User e Password, encaminha para o processo Segurança para que este valide a informação

Cria Socket com a porta UDP 8002 recebe informação de log Recebido pelo buffer do cliente. Hastable LogFicheiroRecebido, ip buffer String nomeficheiro.

Thread 2 >
overflow ou exception, Thread 1(Stop) carrega logRecebido e verifica quais estão em false (não recebidos) então verifica no ficheiro LogFicheiroPedidos (ChecksumFiles) se False, e carrega novamente o pedido. Thread 1 Reset mas não (Start), apenas quando Thread 2 Overflow ira carregar a Thread 1(Start).

Cria Socket TCP 8004 - 8014 Recebe Buffer com Ficheiro (de Cliente). Em blocos de 1024 bytes

Fala

Thread 1 >
Cria Socket UTP 8015 e UTP 8016, encaminha informação de ficheiro recebido em movitp
Cria socket 8017 UTD para informar ao cliente da autentificação.
Guarda Informação em logFala.

Thread 2 >
Overflow ou exception
carrega logFala e continua a enviar o ultimo passo de informação
Limpa LogFala. Abre os mesmos sockets e continua o processo. Quando bloquear inicia o Thread1 Novamente.

3. Verifica estado das threads 1 e 2 de ouve e fala e inicia e reinicia sempre que necessário

4. Encaminha socket tcp 8000 e 8001 para o Processo segurança e aguarda receber do mesmo Thread 3 a autentificação estado True e False de Aprovado ou reprovado. Envia para a Thread Fala a Resposta Socket 8017 UDP.

5. Lé o ficheiro LogFicheiroPedidos para saber quais os que precisa de pedir aos clientes.

Segurança

1) Carrega o ficheiro Clientes de checksumFiles, gerido pela Manutenção;
2) Verifica depois da autentificação se a hora e data da máquina local do cliente está certa para que este possa estar o mais sicnronizado possível com o Movitp;
3) Verifica se tem a mesma actualização que o MovitpClient.

Levantamento de Perfis de Utilização da Aplicação

Na fase de concepção, tendo em conta analise do problema, foi necessário determinar os perfis de utilização tendo em conta as plataformas.

Plataforma Cliente

Características	Implicação para o Design
Idade/género/saúde 18-50/M-F/Condição visual razoável	A nível de tamanho fonte/letra na linha de comandos poderá influenciar.
Linguagem	Língua Portuguesa e Inglesa
A experiencia no manuseamento de equipamentos Hardware/ Software	É necessário um utilizador com conhecimentos base de TCP/IP e com conhecimentos de manuseamento no SO que estiver a executar aplicação. Apesar de a comunicação com o servidor, o envio de ficheiros, actualização de Software e as restantes tarefas serem realizadas de forma automática pela Aplicação, poderá ser necessário a intervenção de um utilizador. Nomeadamente para configurações base da aplicação cliente.

Plataforma Servidor

Características	Implicação para o Design
Idade/género/saúde 18-50/M-F/Condição visual razoável	A nível de tamanho fonte/letra na linha de comandos poderá influenciar.
Linguagem	Língua Portuguesa e Inglesa
A experiencia no manuseamento de equipamentos Hardware/ Software	É necessário um utilizador com conhecimentos TCP/IP, SOAP, conhecimento do SO onde estiver a ser executada a aplicação, experiencia em manuseamento com serviços WebServices e a sua configuração (mysql, SAP), Segurança como Certificados Digitais, Keystores.

Perfis de Tarefa

Plataforma Cliente

O cliente deverá ser capaz de realização actualizações de Software, autentificar-se como Cliente, enviar/receber Ficheiros e realizar configurações base para a sua aplicação.

Tarefa : Actualização

Objectivo, pretende-se que sempre que houver uma nova actualização de software, deverá realizar primeiro antes de realizar qualquer gestão de ficheiros.

Tarefa: Gestão Ficheiros

Objectivo, o utilizador deverá receber na sua maquina ficheiros e poder enviar ao servidor.

Subtarefas:

FalaSeguro: Deverá ser capaz de analisar quais os ficheiros que contem onde estão, defenir prioridades de envio tendo em conta a data, mais antigo para mais recente criado.

OuveSeguro: Deverá receber uma lista(log) a indicar quais os ficheiros que irá receber do servidor, recebe os ficheiros do servidor e depois actualiza a lista com indicação quais recebeu.

Tarefa: Configuração

O utilizador poderá realizar configurações base, para o bom funcionamento da aplicação Cliente.

Subtarefas:

Configurações de acesso ao Servidor (dominio).

Definição de envio e recepção simultaneo de ficheiros

Definição de porta de recepção/envio de dados

Configurações de Perfil da conta de Cliente

Configuração de Certificado Digital

Tarefa: Autentificar

Introdução dos dados de login para acesso à plataforma.

Plataforma Servidor

Algumas das tarefas serão realizadas de forma automática, contudo será necessária uma configuração por parte de um Operdor/Administrador.

O servidor tem a responsabilidade de autentificar apenas quem está registado e autorizado, aceitar comandos, pedidos de clientes, responder aos mesmos, realizar a gestão de ficheiros, webservices, etc.

Tarefa: Comunicação

Subtarefas:

Fala/Ouve: Deverá receber comandos, pedidos do cliente/utilizador e responder aos mesmos.

Fala/Ouve Seguro: Deverá enviar/receber ficheiros.

Configurações:

Configurações de acesso ao Servidor (dominio).

Definição de envio e recepção simultaneo de ficheiros

Definição de porta de recepção/envio de dados

Configurações de Perfil da conta de Cliente

Configuração de Certificado Digital

Tarefa: Segurança

Subtarefas:

Encriptação/Desencriptação de Ficheiros

Subtarefa: Autentificação

Subtarefa: Certificação

Criar/Alterar/Remover Certificados

Subtarefa: Criar/Alterar/Remover Certificado x509 (“Request”, Assinados,AutoAssinados)

Criar/Alterar/Remover Keystores/Chaves

Subtarefa: Criar/Alterar/Remover Chaves Publicas/Privadas, Keystores (RSA,SHA-1)

Assinar Certificados

Atribuir pacote Keystore com Certificado

Subtarefa: Contas

Criar/Alterar/Remover contas para Utilizador;

Criar/Alterar/Remover Perfil;

Criar/Alterar/Remover Tipo de conta;

Atribuir Permissões/Perfil/Tipo de Conta;

Manutenção de restrição IP’s.

Tarefa:Gestão de Ficheiros

Algoritmia de Ordenação de Ficheiros, deverá ser capaz de ordenar os ficheiros, atribuir prioridades para envio e recepção tendo em conta as datas.

Tarefa: Webservices

Criar/Alterar/Remover Webservice

Criar XML para leitura de dados como objecto

Configurações de acesso ao serviço webservice (dominio)

c. Desenvolvimento

Com base na análise anterior procedeu-se a respectiva programação da solução utilizando a linguagem Java com a API Eclipse, a Shell (linha de comandos), ligando através de um router SMC por ligação cabo dois ou mais computadores para realização de Testes Unitários.

Tentamos enquadrar a metodologia de Software "Agile" para o Projecto. Utilizando os diagramas/fluxogramas desenvolvidos (Guião de Programação) consultar anexo foi criado um pequeno Guião de Programação para orientar a programar por fases todo o Sistema, ao mesmo tempo todo o projecto foi documentado faseadamente em Microsoft Project.

Fase	Nº Tarefa	Tarefa	Requisitos	Concluido %
1	1	Ligação Cliente Servidor Simples com Socket Simples	Ligação Cliente Servidor Simultanea	100%
1	2	Criação de várias ligações por Thread e sincronização das mesmas	Sincronizar Tarefas	100%
1	3	Envio de Ficheiros Cliente Servidor Simples	Enviar Ficheiro	100%
1	4	Envio de Ficheiros Cliente Servidor em Threads		100%
1	5	Criar Ligação Segura Simples	SSLSOCKET	100%
1	6	Criar Ligação Segura ASSINCRONA	SSLSOCKET	87%
1	7	Enviar ficheiros por Ligação Segura		100%
1	8	Sockets Non Bloking Implementaçao ficheiros e funções	Converter ligações para NIOSocket77%	77%
2	9	Criar contas clientes e respectivas tarefas de Gestão		58%
2	10	Criação de Certificados	AutoAssinados, request e assinar	68%
2	11	Criação Keystores e atribuiçao certificados		100%
3	12	Desenvolver algoritmo QuickSort em java		100%
3	13	Implementação de um Webservice Mysql		98%
3	14	Return dados do Serviço Mysql		100%
4	15	Utilizar RMI como metodo de invocação para todas as Funções	Dividir Sistema em Camadas	86%
4	16	Criar Interface para atribuir funçoes		100%
5	17	Criar meio para receber actualizações Cliente e instalar		47%

d. Implementação

Requisitos para Implementação

Possível instalação de patchs sem causar quebras de sistema.

Gestão de contas de Cliente/Gestor Aplicacional, chaves de segurança e certificados Digitais, novos serviços (webservices, sap etc.).

Planeamento DisasterRecovery, em caso de "ataque" ou em todas as situações que podem ameaçar a integridade da informação é necessário criar um sistema Backup para protecção da mesma.

O sistema deve ser capaz de monitorizar os seus recursos de modo a poder precaver-se em todas as situações necessárias (carga de memoria, cpu latency, pings da rede).
Suportar novas ligações e falhas das mesma sem prejudicar todo o Sistema.
Deverá mostrar um aviso a todos os clientes e serviços a informar de qualquer indisponibilidade do **MOVitp**.
Deverá conter uma DHT (Distributed Hash Table), capaz de identificar o estado de toda a rede, contendo a sua carga, id localização, data da ultima actualização da maquina e ou nó.

Fase de Preparação de Dados

Numa primeira fase é necessário definir quem são os clientes, qual será o domínio e o ip onde irá operar o servidor, e definir os certificados. É necessário reunir toda a informação necessária pois a plataforma Servidor não poderá funcionar correctamente se não tiver contas de administrador, certificados e chave criadas para o efeito.

Fase de Preparação de Arquivos

Nesta fase é necessário tendo em conta o Sistema Operativo, Linux, HP-UX ou Windows, copiar o pacote final da aplicação Servidor Movitp para uma pasta onde deverá ser executado.
Deverá verificar se a mesma vem com as respectivas bibliotecas necessárias como “Naga”, “bouncycastle”, sendo as mesmas responsáveis para a criação de NIO sockets e certificados X509.

Fase de Instalação e Execução

Instalação Servidor
1º MovitpDados -> 2º MovitpGestão-> 3º MovitpControlo
O servidor deverá ser o primeiro a ser implementado.
A.1 Java SE versão 7, Java jre7 instalados na maquina local utilizando os comandopara executar: java <nomeaplicaçao> e para compilar javac <nomeaplicaçao>.
A.2 Verificar se os WebServices encontram-se correctamente instalados preparados e definidas as suas portas e ips no ficheiro config.in
A.3 Verificar na infra-estrutura de rede se as portas definidas em config.in estão abertas

A.4 compilar e executar a aplicação MovitpDados para que a mesma consiga detectar os services Soap (webservices)

A.5 compilar MovitpGestao e MovitpControlo utilizando as bibliotecas naga.

Compilar em Linux:

javac -cp '.:naga_versao.jar' MovitpGestao.java

javac -cp '.:naga_versao.jar' MovitpControlo.java

Compilar em Windows:

javac -cp .;naga_versao.jar MovitpGestao.java

javac -cp .;naga_versao.jar MovitpControlo.java

Executar em Linux:

java -cp '.: naga_versao.jar' MovitpGestao.java

java -cp '.: naga_versao.jar' MovitpControlo.java

Executar em windows

java -cp .;naga_versao.jar MovitpGestao.java

java -cp .;naga_versao.jar MovitpControlo.java

A.6 Utilizando a interface deverá criar as respectivas contas de cliente, certificados definidas na Fase de Preparação de Dados

A.7 Escolha a opção Iniciar (Start) para iniciar a aplicação.

3. B) Fase de Instalação Cliente

B.1 Java SE versão 7 instalados na máquina local utilizando os comando para executar:

java <nomeaplicaçao> e para compilar javac <nomeaplicaçao>.

B.2 Verificar na infra-estrutura de rede se as portas definidas em config.in estão abertas

B.3 Compilar e executar utilizando os comandos:

Compilar em Linux:

javac -cp '.:naga_versao.jar' MovitpCliente.java

Compilar em Windows:

javac -cp .;naga_versao.jar MovitpCliente.java

Executar em Linux:

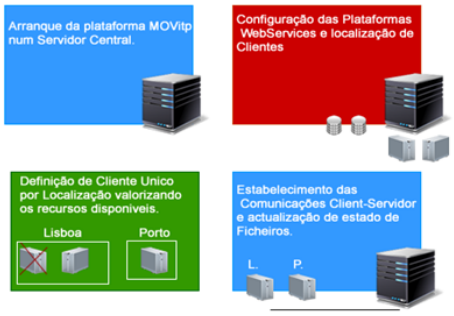
java -cp '.: naga_versao.jar‘ MovitpCliente.java

Executar em windows

java -cp .;naga_versao.jar MovitpCliente.java

B.4 Efectuar o Login

Esquema para “startup” da infra-estrutura:



4. Resultados

Os resultados que se pretendem atingir por um lado é a maior rapidez na transferência dos ficheiros, por outro maior segurança no transporte destes dados.

Estes resultados irão ser medidos em oposição o método de FTP.

De momento ainda estamos a efectuar os testes e melhorias de código.

Bibliografia

Livros:

"Secure, Managed File Transfer", Don Jones, Realtime Publishers (http://nexus.realtimepublishers.com/)

"Distributed Systems: Concepts and Design" (4th Edition), Coulouris, Dollimore & Kindberg, Ed. Addison-Wiley, Maio 2005; ISBN 0321263545 http://books.google.pt/books?id=d63sQPvBezgC&printsec=frontcover&dq=Distributed+Systems:+Concepts+and+Design&source=bl&ots=qvrv-myq2D&sig=t7yKzyxfe8clX4g3ZUIyWbtfLyI&hl=pt-PT&sa=X&ei=yusPUNXCIceHmQXV6YGQAw&ved=0CDEQ6AEwAA#v=onepage&q=Distributed%20Systems%3A%20Concepts%20and%20Design&f=false

"Distributed Systems: Principles and Paradigms" (2nd Edition), Tanenbaum & van Steen, Prentice Hall, 2006

"Web Services: Principles and Technology", Michael Papazoglou, Prentice Hall, 2007

"Tecnologia dos Sistema Distribuídos", J. Marques e P. Guedes, FCA Editora; Maio 1998

Sites:

Java RMI

http://docs.oracle.com/javase/tutorial/rmi/index.html

http://onjava.com/onjava/2002/09/04/nio.html

Java NIOSocket

http://code.google.com/p/naga/

http://www.cs.brown.edu/courses/cs161/papers/j-nio-ltr.pdf

http://java.sun.com/developer/technicalArticles/ALT/sockets/

http://docs.oracle.com/javase/tutorial/essential/io/fileio.html

Certificados

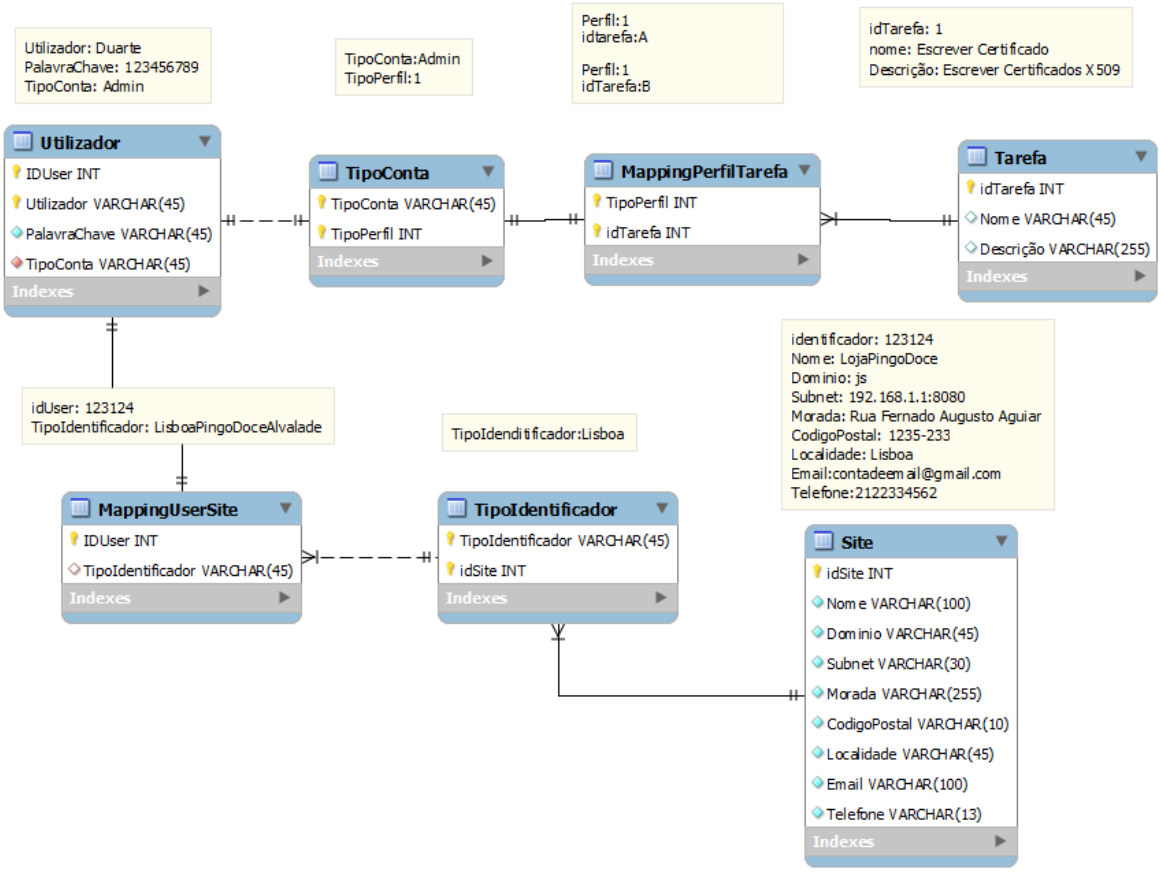
http://www.mayrhofer.eu.org/create-x509-certs-in-java/

http://www.bouncycastle.org/

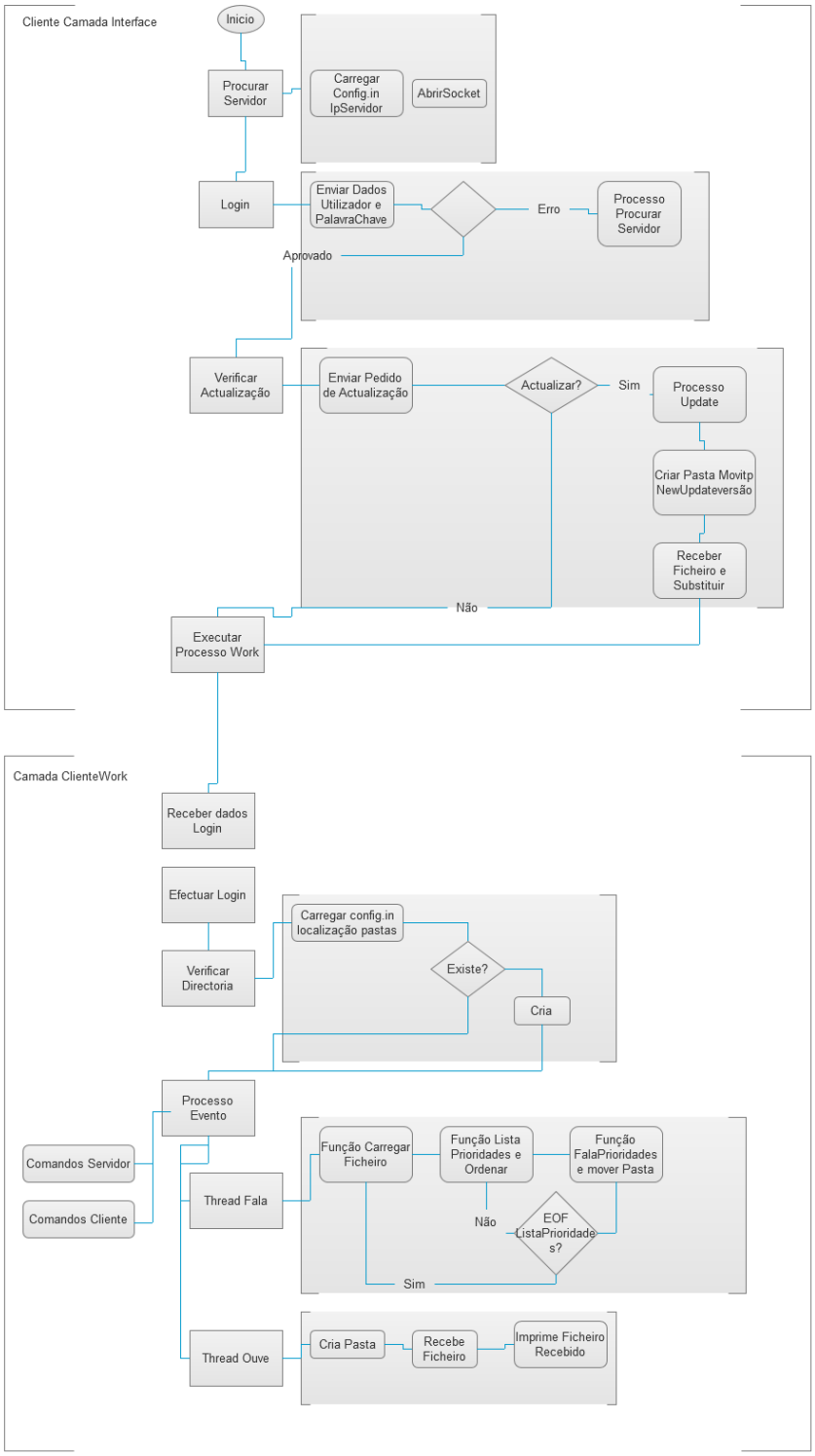
http://www.java2s.com/Tutorial/Java/0490__Security/Catalog0490__Security.htm

Anexo 1 – Estrutura de Base de Dados

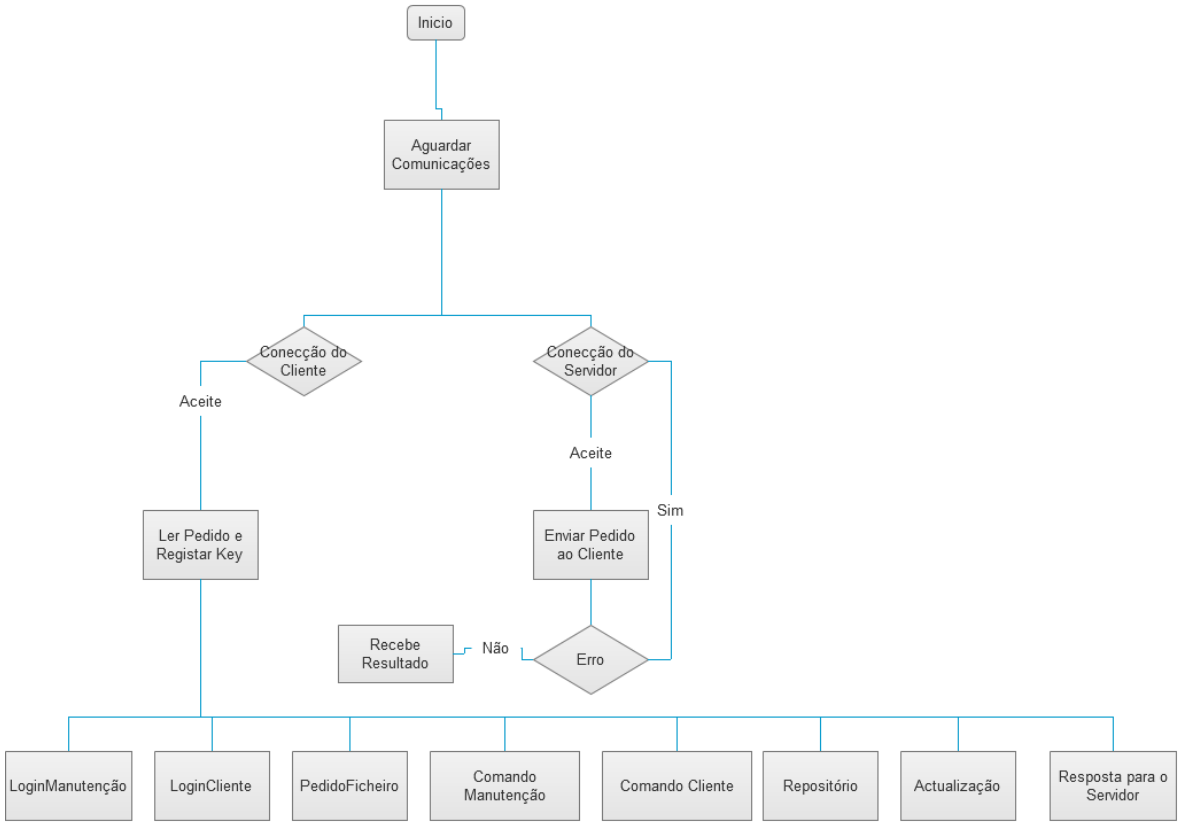
Estrutura de Base de Dados para validação de utilizadores e perfis de utilização:



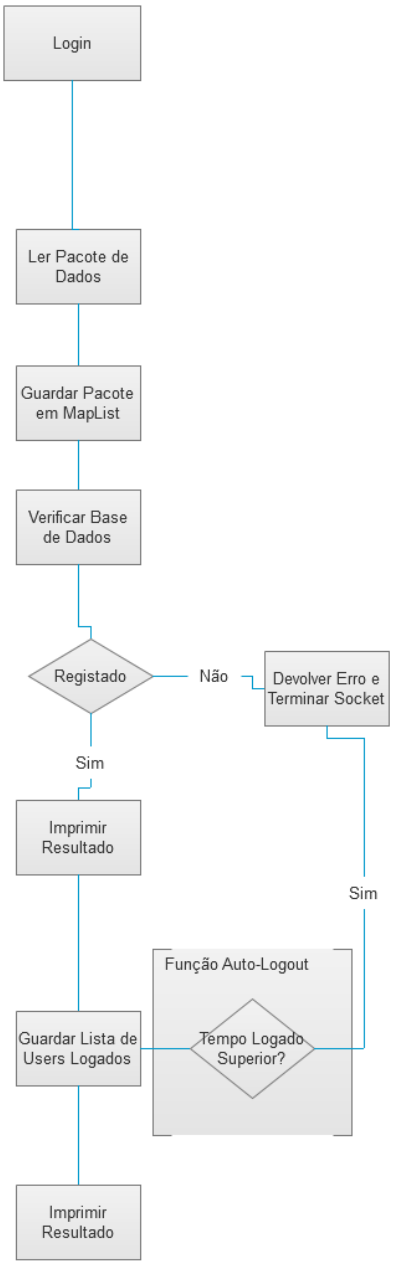
Anexo 2 – Fluxograma da Camada Cliente



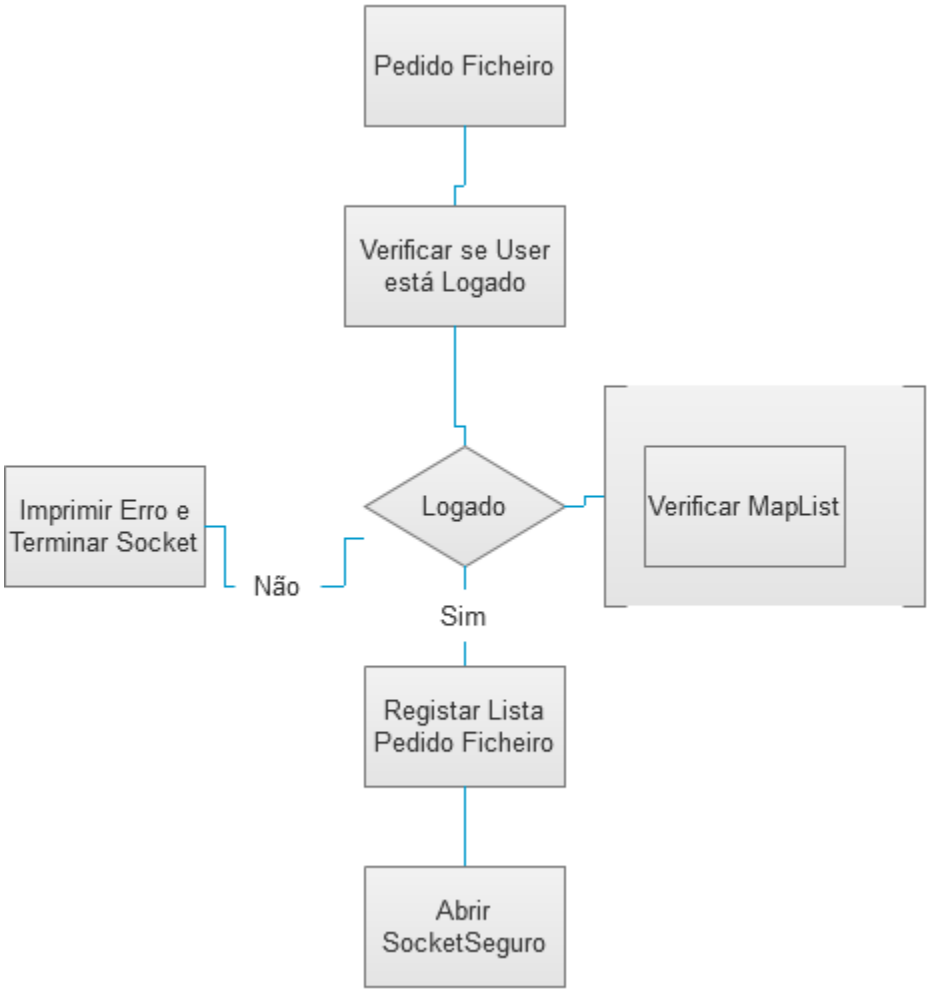
Anexo 3 – Fluxograma da Comunicação Servidor



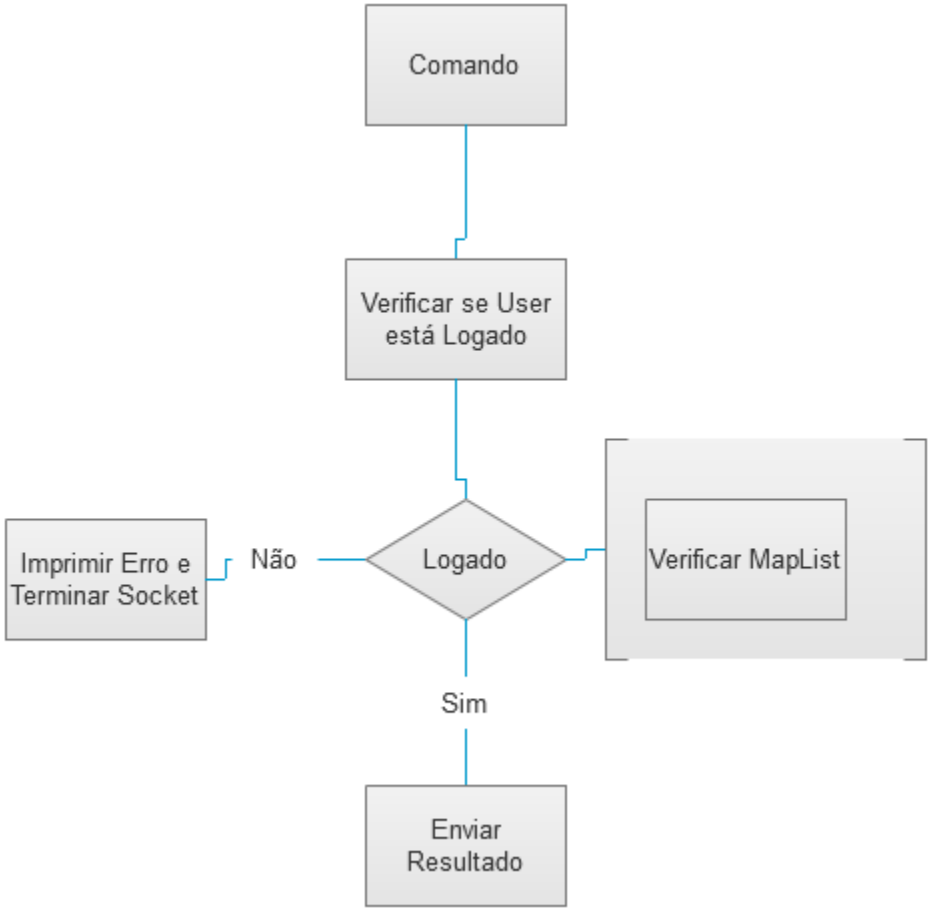
Anexo 4 – Fluxograma de Segurança para Login



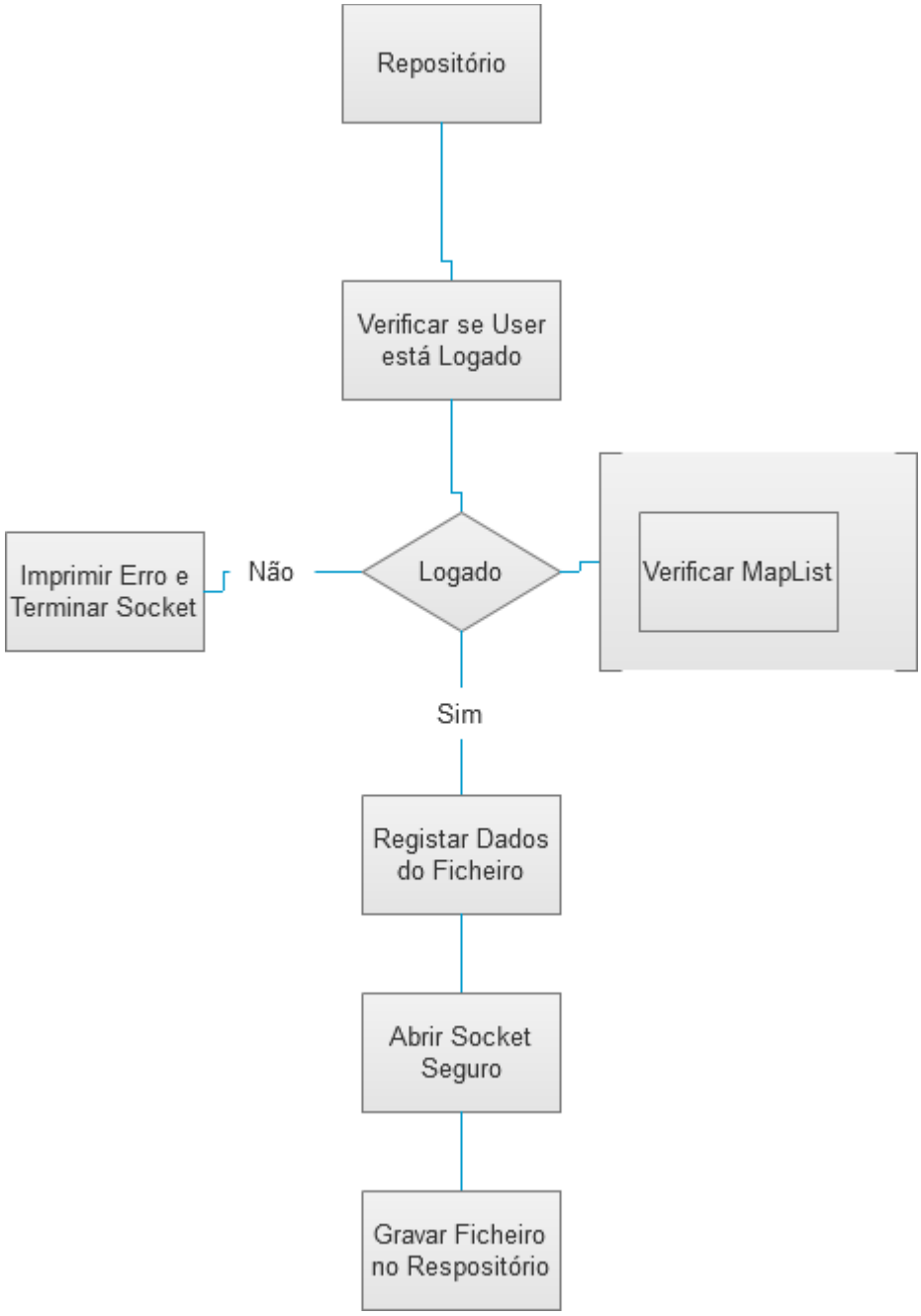
Anexo 5 – Fluxograma de Segurança para PedidoFicheiro



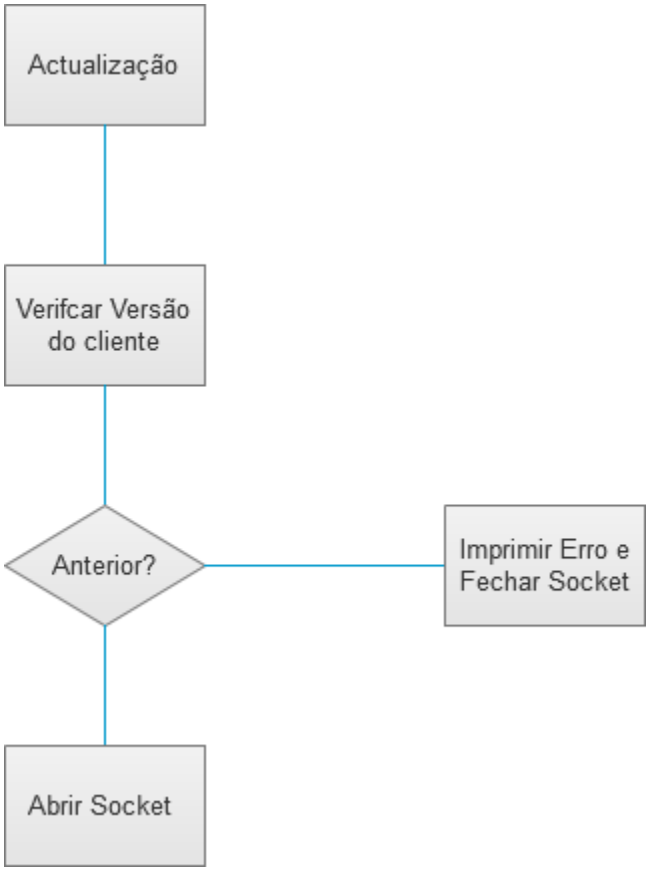
Anexo 6 – Fluxograma de Segurança para Comando



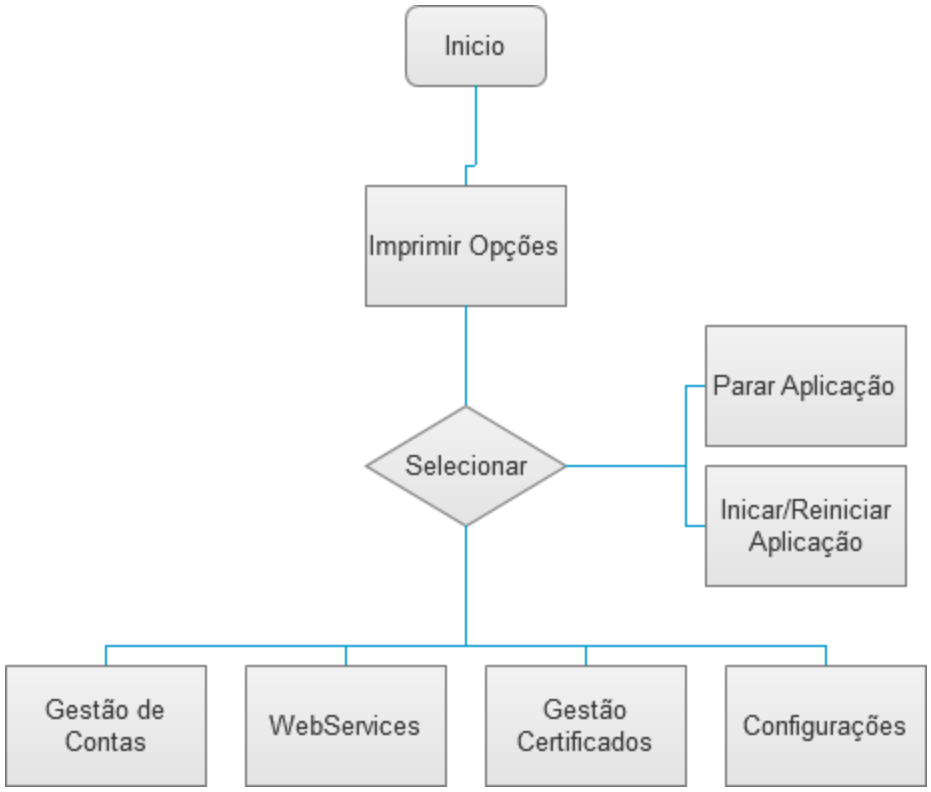
Anexo 7 – Fluxograma de Segurança para Repositório



Anexo 8 – Fluxograma para Actualização



Anexo 9 – Fluxograma para Camada de Gestão



Anexo 10 – Fluxograma para Camada de Dados