Alberto Manuel da Silva Gomes

Gestão e Manutenção de Múltiplos Terminais em Redes de Próxima Geração

October 2010

Universidade do Minho
Escola de Engenharia

Gestão e Manutenção de Múltiplos Terminais em Redes de Próxima Geração
Alberto Manuel da Silva Gomes

UMinho | 2010

Universidade do Minho
Escola de Engenharia

Alberto Manuel da Silva Gomes

Gestão e Manutenção de Múltiplos Terminais em Redes de Próxima Geração

Mestrado em Engenharia Informática

Trabalho realizado sob a orientação do
Professor Doutor António Manuel Nestor Ribeiro
Professor Doutor PauloManuel Martins de Carvalho
Engenheiro Ricardo Azevedo Pereira

October 2010

Agradecimentos

Esta dissertação assinala a conclusão de mais um ciclo da minha vida académica. No entanto, tal não teria sido possível sem a ajuda e apoio de algumas pessoas.
Queria deixar o meu mais sincero obrigado aos meus pais, Salvador e Maria, e à minha irmã, Márcia, pelo apoio e compreensão incondicional que sempre me proporcionaram e por permitirem que eu chegasse até aqui.
Gostaria de agradecer ao meu orientador, Professor Doutor António Manuel Nestor Ribeiro e ao meu co-orientador, Professor Doutor Paulo Manuel Martins de Carvalho, pelo apoio e disponibilidade que sempre apresentaram para com o meu trabalho.
Uma palavra de agradecimento ao Engenheiro Ricardo Azevedo Pereira, que me acompanhou no decorrer do trabalho na PT Inovações, pela prontidão, disponibilidade e paciência que demonstrou para comigo.
Não podia deixar de agradecer também aos meus colegas e amigos do Instituto de Telecomunicações - Pólo de Aveiro e da PT Inovação, que me proporcionaram uma fácil integração nestas duas instituições e que directa ou indirectamente contribuíram para o sucesso deste trabalho.

Resumo

O número de dispositivos que cada utilizador possui para aceder aos diferentes serviços de comunicações, tem vindo progressivamente a aumentar. Ao mesmo tempo, os fornecedores de serviços têm tentado oferecer os mesmos serviços em múltiplas plataformas para tentar alcançar o maior número de clientes possível. No entanto, este cenário levanta questões relacionadas com a mobilidade de sessão, mobilidade de identidade, dados privados do utilizador e partilha de recursos que não têm hoje uma resposta adequada. Em geral, cada dispositivo acede a um serviço sem o conhecimento da existência de outros dispositivos. Informação privada do utilizador é partilhada por um conjunto de dispositivos, podendo existir situações onde a informação necessária não está disponível no dispositivo que o utilizador está a usar no momento.

Num cenário futuro de "Internet of Things" o número de dispositivos/objectos, pertencentes aos utilizadores será ainda maior. Devido a este facto, torna-se essencial a disponibilização de uma gestão simplificada e uma visão global de todos os dispositivos pertencentes ao utilizador. Assim, pretende-se garantir que o utilizador possa aceder a um serviço usando o melhor dispositivo que tem ao seu dispor, com toda a informação necessária e sem quebra de privacidade ou segurança que o possam prejudicar.

Esta dissertação propõe o estudo e criação de uma arquitectura que permita ao utilizador efectuar uma gestão dos seus dispositivos por forma a aceder aos serviços contratados e proporcionar a mobilidade de sessão e identidade. Neste sentido foi elaborada uma arquitectura que oferece ao utilizador uma visão unificada de todos os seus dispositivos, por forma a disponibilizar uma forma simples e inovadora do utilizador gerir tais dispositivos. Esta arquitectura proporciona também a mobilidade de terminal, bem como a mobilidade de identidade dentro do mesmo dispositivo e até mesmo entre dispositivos diferentes, oferecendo assim uma utilização mais eficiente dos dispositivos do utilizador. Com o propósito de testar a arquitectura elaborada, foram efectuados testes funcionais que demonstram que as especificações inicialmente propostas foram cumpridas. Com base nos testes efectuados, foi realizada uma avaliação do impacto no utilizador que a arquitectura induz, concluindo-se que tal impacto é mínimo.

Abstract

The number of devices that each user has available to access different communications services has been increasing gradually. At the same time, the service providers intent to offer the same services over multiple platforms to reach as many customers as possible. However, this scenario raises several issues related to session mobility, identity mobility, user's private data and shared resources, which, so far, have not received an adequate response. In general, each device accesses a service without knowing the existence of other devices. User's private data is shared among a set of devices and, in some situations, the required information may not be available in the device currently in use.

In a future scenario of "Internet of Things", the number of devices/objects belonging to users will be even higher. Therefore, it is essential to achieve a simplified management scheme providing an overview of all devices belonging to the user. The aim is to ensure that a user can access a service using the best available device, with all the required information and without privacy or security concerns.

This thesis proposes the study and creation of an architecture that allows a user to manage its devices in order to access the negotiated services, providing also session mobility and identity mobility. In this context, the proposed architecture offers a unified view of all devices to the user, providing a simple and innovative approach to manage such devices. This architecture also allows terminal mobility and identity mobility within the same device or between user devices, leading to a more efficient use of those devices. To test and evaluate the architecture, functional tests were performed, demonstrating that the initially proposed specifications were met. Based on the tests, the impact the architecture induces on the user was assessed, and the conclusion was that such impact is minimal.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

3G	Third Generation
ADSL	Assymetric Digital Subscriber Line
API	Application Programming Interface
CPU	Central Processing Unit
DNS	Domain Name System
HIP	Host Identity Protocol
HSPA	High Speed Packet Access
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IP	Internet Protocol
IKEv2	Internet Key Exchange Protocol Version 2
JSP	JavaServer Pages
LMA	Local Mobility Anchor
LTE	Long Term Evolution
MAG	Mobile Access Gateway
MDA	Model Driven Architecture
MIPv6	Mobile IPv6
OASIS	Organization for the Advancement of Structured Information Standards
PAN	Personal Area Network

PDA	Personal Digital Assistant
PIN	Personal Identification Number
PMIPv6	Proxy Mobile IPv6
RTT	Round-Trip Time
SAML	Security Assertion Markup Language
SDP	Session Description Protocol
SIP	Session Initiation Protocol
SSO	Single Sign-On
SSOff	Single Sign-Off
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UML	Unified Modeling Language
VoIP	Voice over IP
Wi-Fi	Wireless-Fidelity
Wi-MAX	Worldwide Interoperability for Microwave Access
XML	Extensible Markup Language

Capítulo 1

Introdução

Hoje em dia é normal um utilizador ter diversos dispositivos (como telemóvel, PDA, computador, entre outros) com acesso a uma infinidade de serviços, que podem ser acedidos independentemente da sua localização e do dispositivo em causa. Exemplo destes serviços são: Internet, chamadas de voz e televisão. Ao mesmo tempo, os fornecedores de serviço têm tentado oferecer os mesmos serviços em múltiplas plataformas. Posto isto, um utilizador poderia a qualquer instante transferir uma sessão de vídeo do seu televisor para o telemóvel ou, uma vez que o televisor tem uma imagem com dimensões muito maiores, transmitir uma videochamada de um dispositivo móvel para o televisor, de uma forma simples e eficaz, sem que haja qualquer perda de informação. No entanto, este cenário levanta questões relacionadas com a mobilidade de sessão, o uso dos dados privados do utilizador e também a partilha de recursos, que têm de ser abordadas de uma forma adequada. Quando um utilizador solicita um serviço específico, através de um dispositivo particular, os seus dispositivos restantes não têm conhecimento de tal acção. Por outras palavras, os dispositivos são incapazes de comunicar entre si de forma a terem conhecimento de, por exemplo, qual é o dispositivo que tem melhor conectividade e capacidade para prestar o serviço pretendido.

Olhando para um cenário futuro no contexto de "Internet of Things", onde há uma infinidade de dispositivos ligados entre si, um utilizador pode ter um conjunto enorme de dispositivos disponíveis para aceder ao mesmo serviço. Nesta situação torna-se difícil gerir e optimizar o uso do melhor dispositivo. Para tal, é necessário um sistema de gestão capaz de garantir que em qualquer momento o utilizador está a usar um determinado serviço no melhor dispositivo sem quebras de privacidade ou de segurança.

1.1 Enquadramento e Motivação

Ao longo do tempo a Internet tem vindo a evoluir e, de certa forma, a tornar os seus utilizadores mais dependentes dela. Através da Internet é possível aceder a diversos serviços que permitem partilhar informação (ex: fotos, vídeos, documentos), efectuar compras, aceder à conta bancária, pedir declarações, efectuar comunicações de voz, ver televisão, entre outros. Para suportar estes serviços surgiram tecnologias de banda larga (ex: ADSL, WiFi, Wi-MAX, 3G, HSPA, LTE) que oferecem ao utilizador um melhor acesso a este tipo de serviços. Com o aparecimento destas tecnologias cresceu também o número de dispositivos capazes de prestar tais serviços aos utilizadores. Consequentemente, os utilizadores têm cada vez mais dispositivos, o que torna a sua gestão mais complexa.
O crescimento da Internet faz também com que os utilizadores criem mais contas, tenham diferentes perfis, tenham mais *passwords* e outro tipo de informação que têm de ser geridas. Este acréscimo de informação leva muitas vezes os utilizadores a cometer erros (ex: usar a mesma *password* em diversos *sites*). Para tal, é essencial ter uma forma de gerir a informação pessoal dos utilizadores para garantir que tal não acontece.
Este cenário leva à necessidade da elaboração de um componente capaz de gerir os dispositivos e a informação pessoal dos utilizadores, bem como a coordenação entre os diversos dispositivos. A gestão da informação pessoal dos utilizadores, passa pela criação de identidades virtuais.

1.2 Objectivos

Tendo como base as questões apresentadas e os desafios relacionados com a mobilidade de sessão, mobilidade de identidade e mobilidade de terminal, pretende-se estudar e desenvolver uma arquitectura de *software* que, de uma forma simples, dê-a ao utilizador uma visão global de todos os seus dispositivos e ofereça uma gestão simplificada das suas identidades. Tendo a arquitectura de *software* definida será desenvolvido uma aplicação que suporte toda a arquitectura. Esta aplicação será desenvolvida seguindo uma arquitectura Model Driven Architecture (MDA) [3], e assim poder fazer a modelação da aplicação e respectiva geração de código de uma forma automatizada.
O trabalho a desenvolver apresenta os seguintes objectivos específicos:

- realização de um estudo das arquitecturas existentes que permitem a criação de conjuntos de terminais sobre um mesmo elemento;

- especificação de cenários concretos de utilização num cenário de serviços de um operador;
- especificação, desenvolvimento e prototipagem da solução orientada a um cenário previamente descrito;
- teste e validação do protótipo num cenário real.

1.3 Síntese das principais contribuições

A principal contribuição resultante do trabalho desenvolvido traduz-se na proposta e implementação de uma solução que ajuda os utilizadores a gerir os seus dispositivos e informação pessoal. Esta gestão traduz-se no uso e criação de identidades virtuais, de forma a que o utilizador possa usar qualquer serviço de uma forma simples e segura.

1.4 Estrutura da dissertação

A estrutura da dissertação está organizada de seguinte forma: neste capítulo, Capítulo 1, é apresentada uma introdução do problema, o enquadramento e as principais motivações que levaram ao desenvolvimento da dissertação, os objectivos pretendidos e uma síntese das principais contribuições resultantes do trabalho desenvolvido.
No Capítulo 2 é feita uma abordagem do estado da arte, onde é apresentado o conceito de mobilidade, como sendo a mobilidade de objecto, que inclui mobilidade de terminal, identidade e de rede, e a mobilidade de serviço. Neste capítulo são também descritas algumas tecnologias, como SIP, HIP, PMIPv6, SAML, OpenId e OAuth, e identificadas algumas questões relacionadas com privacidade e identidade dos utilizadores. Por fim, é apresentada uma proposta para o desenvolvimento do Terminal Virtual proposto por Marc Barisch.
No Capítulo 3 é apresentada a arquitectura do Terminal Virtual e cada uma das suas componentes, assim como a integração do Terminal Virtual com o gestor de identidades da PT Inovação [4]. Neste capítulo é também apresentado o modelo de dados para suporte da arquitectura do Terminal Virtual, assim como, a implementação da arquitectura proposta.
No Capítulo 4 são apresentado cenários de utilização, onde se mostram a forma como a arquitectura do Terminal Virtual funciona.
No Capítulo 5 são apresentados os resultados obtidos sob a forma de testes funcionais, nos quais são verificadas as especificações do Terminal Virtual e, por fim, é efectuada uma avaliação do

impacto no utilizador que esta nova solução acarreta.
Finalmente, no Capítulo 6 são apresentadas as conclusões, evidenciando assim o trabalho realizado e as principais contribuições prestadas. Por fim, é trabalho futuro que poderá ser efectuada tendo em vista o enriquecimento da arquitectura do Terminal Virtual.

Capítulo 2

Estado da arte

2.1 Introdução

Neste capítulo pretende-se apresentar o levantamento do estado da arte que foi efectuado, bem como uma breve análise ao seu conteúdo. Mais concretamente, é apresentado o conceito de mobilidade distinguindo a mobilidade de objecto da mobilidade de serviço. São também apresentadas neste capítulo algumas tecnologias que permitem efectuar a mobilidade de terminal, de identidade e de sessão, como SIP, HIP, PMIPv6, SAML, OpenId e OAuth. Por fim, são apresentadas questões relacionadas com a privacidade e identidade dos utilizadores e a abordagem do conceito de Terminal Virtual proposto por Marc Barisch.

2.2 Conceito de Mobilidade

O termo mobilidade pode ter diferentes significados, dependendo do cenário e/ou do domínio que é aplicado. Quando se fala sobre mobilidade, pode-se falar a dois níveis diferentes, vertical e horizontal [5]. A mobilidade horizontal refere-se à mobilidade dentro da mesma camada, geralmente dentro de uma mesma tecnologia. Um exemplo é a mobilidade de sessão entre diferentes dispositivos. Por outro lado, a mobilidade vertical refere-se à mobilidade entre diferentes camadas, geralmente entre diferentes tecnologias. Um exemplo deste tipo de mobilidade é o processo de transferência entre 3G e Wi-Fi. Além disso, é possível distinguir a mobilidade em termos daquilo que é movido. Nesse sentido, é ainda possível distinguir dois tipos de mobilidade: a mobilidade de objecto e mobilidade de serviço.

2.2.1 Mobilidade de objecto

Mobilidade de objecto é um tipo de mobilidade abstracto. Pode ser separado em quatro tipos diferentes: mobilidade de terminal, mobilidade pessoal, mobilidade de rede e, finalmente, mobilidade de sessão [6] [7].

- **Mobilidade de terminal:** A mobilidade de terminal refere-se à possibilidade de um terminal em movimento, manter uma sessão activa. Por outras palavras, a capacidade de aceder a um serviço enquanto está em movimento ou quando se desloca de um local para outro.
- **Mobilidade de identidade:** A mobilidade de identidade permite ao utilizador usar as suas identidades em qualquer dispositivo. Este tipo de mobilidade é também denominado muitas vezes por mobilidade pessoal. A mobilidade de identidade não depende apenas da disponibilidade de um dispositivo mas também depende das capacidades de segurança do dispositivo e de alguma informação de contexto baseada na identidade do utilizador.
- **Mobilidade de rede:** A mobilidade de rede permite que uma rede local com terminais fixos ou móveis mude de ponto de acesso de forma transparente para os terminais. Por exemplo, um utilizador pode ter múltiplos dispositivos como telemóvel, computador portátil e PDA ligados à Internet através de uma rede de computadores pessoais, isto é, uma PAN (Personal Area Network), onde o telemóvel é o ponto de acesso. Quando a rede à qual o telemóvel está ligado muda, todos os restantes dispositivos que estão ligados ao telemóvel através da PAN, vão também mudar, efectuando assim a mobilidade da rede PAN [8].
- **Mobilidade de sessão:** A mobilidade de sessão permite a transferência de uma sessão de um dispositivo para outro. Para efectuar este tipo de mobilidade é necessário os dispositivos estarem disponíveis e terem capacidade para lidar com a sessão específica.

2.2.2 Mobilidade de serviço

A mobilidade de serviço permite a um utilizador aceder a um determinado serviço independentemente da sua localização. Estes serviços podem ser contínuos ou descontínuos.

- **Serviço contínuo:** Um serviço contínuo corresponde à capacidade de um objecto em movimento poder aceder a um determinado serviço, mantendo o seu estado actual, ou seja,

manter um serviço contínuo. Como serviços contínuos pode-se falar do simples *handover* e do *seamless handover*. Os dois são muito idênticos, a diferença é que ao contrário do simples *handover* o *seamless handover* não causa nenhum impacto durante e depois da transferência [5].

- **Serviço descontínuo:** Um serviço descontínuo corresponde à capacidade de um objecto em movimento poder aceder a um serviço independentemente da sua localização. Mas neste caso o seu estado actual não é mantido. Por exemplo, um utilizador pode escolher um ponto de acesso diferente parando completamente a sessão corrente e voltar a iniciá-la mais tarde.

Em termos de mobilidade, este estudo irá focar-se na mobilidade de terminal, mobilidade de identidade e mobilidade de sessão, bem como na mobilidade de serviço. Estes tipos de mobilidade são cruciais para o desenvolvimento do conceito de Terminal Virtual. Desta forma, um utilizador pode mudar de terminal, de localização e/ou de rede e manter a mesma sessão activa.

2.3 Tecnologias

Nesta secção são apresentadas as principais tecnologias estudadas que permitem a implementação dos diferentes tipos de mobilidade. Com vista a implementação da mobilidade de sessão, foi estudado o protocolo SIP. Para a mobilidade de terminal foram estudados os protocolos HIP e PMIPv6. De forma a possibilitar a mobilidade de identidade foram também estudadas as tecnologias SAML, OpenID e OAuth.

2.3.1 Session Initiation Protocol (SIP)

O SIP [9] é um protocolo de controlo da camada da aplicação que pode estabelecer, modificar e terminar sessões multimédia, tais como chamadas VoIP. Este protocolo permite também convidar participantes para uma sessão a decorrer, como conferências multicast. Dados multimédia podem ser adicionados e removidos de uma sessão já existente. O protocolo SIP suporta mapeamento de nomes e serviços de redireccionamento de forma transparente. Desta forma um utilizador pode manter o seu identificador sempre visível independente da sua localização. O protocolo SIP tem cinco funcionalidades para estabelecer e terminar comunicações multimédia:

- ***User Location:*** Determina o sistema a ser usado na comunicação.

- ***User Availability:*** Determina a disponibilidade do receptor para realizar a comunicação.
- ***User Capabilities:*** Determina a capacidade de gerar e processar um determinado tipo de conteúdos e os respectivos parâmetros para serem usados na comunicação.
- ***Session Setup:*** Estabelece os parâmetros da sessão para o emissor e o receptor.
- ***Session Management:*** Transfere e termina sessões, modifica os parâmetros da sessão e invoca serviços.

O protocolo SIP não é um sistema de comunicações integradas verticalmente. É apenas um protocolo de sinalização de sessão abstracto, que precisa de ser usado com outros protocolos a fim de fornecer, por exemplo, uma arquitectura multimédia completa. Sendo um protocolo abstracto, não fornece serviços, o SIP fornece primitivas que podem ser usadas para implementar diferentes serviços. Um exemplo de um serviço é a localização. O SIP pode ser usado para recuperar a informação de localização de um utilizador e enviar essa informação como um objecto opaco.
A arquitectura SIP é constituída por quatro componentes: Agentes do Utilizador SIP, Servidores de Registo SIP, Servidores Proxy SIP e Servidores de Redireccionamento SIP. Com estes componentes e com o Protocolo de Descrição de Sessão (SDP) [10] é possível efectuar uma sessão SIP. Abaixo é apresentada uma descrição de cada componente SIP e o papel que desempenha neste processo.

SIP User Agents são terminais SIP ou um *software* instalado nos dispositivos do utilizador, como telemóveis, computadores ou PDAs que são utilizados para criar e gerir as sessões SIP estabelecidas. O SIP User Agent cliente inicia as mensagens e o SIP User Agent servidor responde a essas mensagens.

SIP Registrar Servers são os servidores que contêm o registo da localização de todos os SIP User Agents dentro de um domínio. Estes servidores são responsáveis por procurar e enviar os endereços IP (Internet Protocol) [11] e outra informação pertinente para o SIP Proxy Server.

SIP Proxy Servers são servidores que aceitam pedidos de sessão dos SIP User Agents e perguntam ao SIP Registrar Server a informação de endereçamento do destinatário. Tendo esta informação encaminha a sessão directamente para o SIP User Agent destinatário, se este pertencer ao mesmo domínio ou encaminha-a para o SIP Redirect Servers se o SIP User Agent destinatário pertencer a outro domínio.

SIP Redirect Servers são os servidores que permitem aos SIP Proxy Servers redireccionar a sessão SIP para um domínio externo. Os SIP Redirect Servers podem estar na mesma máquina dos SIP Registrar Servers e SIP Proxy Servers.

O protocolo SIP permite o estabelecimento de três tipos diferentes de sessão. Desta forma podem ser estabelecidas sessões SIP ponto a ponto, ou seja, entre dois SIP User Agents que conhecem os endereços IP um do outro. Podem ser estabelecidas sessões SIP dentro do mesmo domínio. Neste caso, os SIP User Agents não têm conhecimento dos endereços IP uns dos outros, para isso é necessário o SIP Proxy Server para ajudar no estabelecimento da sessão e também é necessário que os SIP User Agents estejam já registados no SIP Registrar Server. Por fim, podem ser estabelecidas sessões SIP entre diferentes domínios. Tal como no estabelecimento de sessões SIP dentro do mesmo domínio, aqui os SIP User Agents não têm conhecimento dos endereços IP uns dos outros. Para além disso, é necessário um SIP Redirect Server para o encaminhamento das mensagens para o SIP Proxy Server do domínio externo. O conteúdo transmitido em qualquer uma destas sessões pode ser áudio, vídeo, texto ou uma combinação destes.

A Figura 2.1 mostra um exemplo do estabelecimento de uma sessão SIP entre o User A (emis-

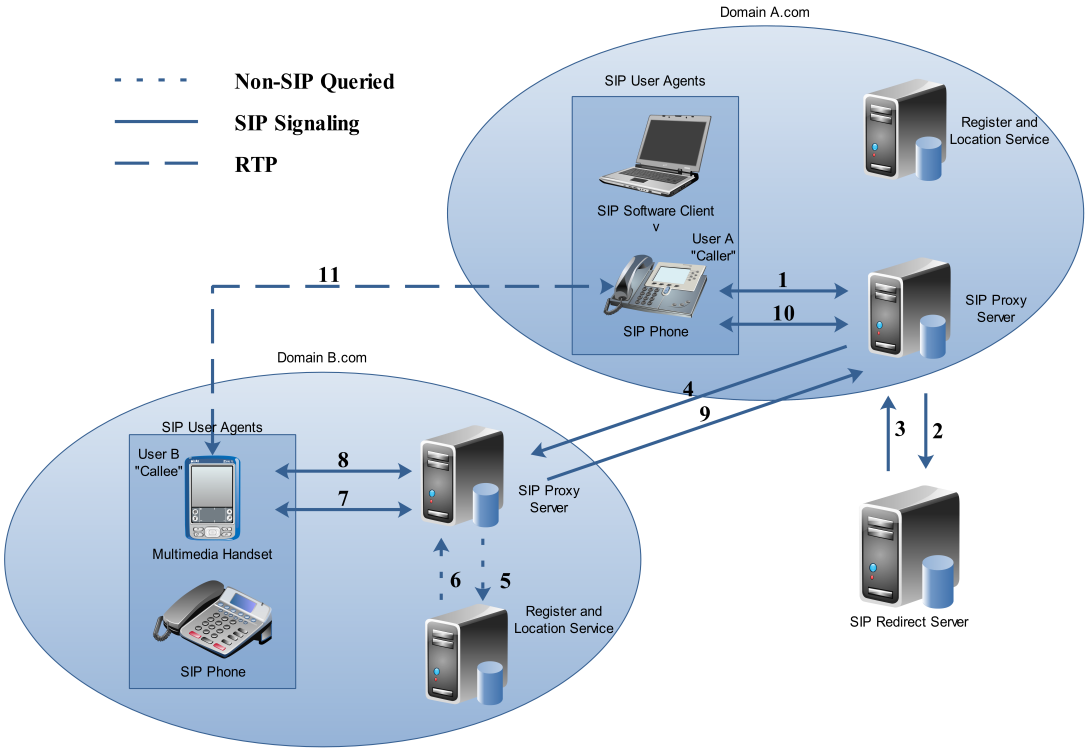

Figura 2.1: Estabelecimento de uma sessão SIP em domínios diferentes

sor) e o User B (receptor) que estão em domínios diferentes. O User A envia um pedido de estabelecimento de sessão com o User B ao SIP Proxy Server do domínio A. Como o User B não está no mesmo domínio, o SIP Proxy Server do domínio A pergunta ao SIP Redirect Server qual é o endereço do User B. O SIP Redirect Server pode utilizar várias alternativas para determinar o endereço do User B, tais como, DNS (Domain Name System) [12] ou bases de dados. Posto isto, o SIP Redirect Server responde ao SIP Proxy Server do domínio A com as informações relativas ao User B. O SIP Proxy Server do domínio A envia o pedido para o SIP Proxy Server do domínio B, que por sua vez questiona o SIP Registrar Server do domínio B. Este responde com o endereço IP do User B e outras informações relevantes. O SIP Proxy Server do domínio B transmite o pedido ao User B. Quando o User B recebe este pedido, responde com a informação de que aceita o pedido e o SIP Proxy Server do domínio B transmite a aceitação do pedido ao SIP Proxy Server do domínio A. Ao receber esta informação, o SIP Proxy Server do domínio A informa o User A da aceitação do pedido por parte do User B. Concluídos estes passos os dois dispositivos já se conhecem e estabelecem uma sessão directamente entre os dois. A partir daqui, os servidores SIP não são mais necessários para a sessão em curso, a menos que a sessão seja alterada ao que algum dos dispositivos deseje finalizar a sessão.

O protocolo SIP não oferece serviços de controlo de conferência e não determina como a conferência é gerida. O SIP pode ser usado para iniciar uma sessão que usa o mesmo protocolo de controlo de conferência. Como as mensagens SIP e as sessões que estabelecem podem passar através de redes totalmente diferente, o protocolo SIP não pode fornecer qualquer tipo de reserva de recursos de rede. O protocolo SIP funciona quer em IPv4, quer em IPv6.

2.3.2 Host Identity Protocol (HIP)

A Internet tem dois importantes espaços de nomes globais: o endereçamento IP e a resolução de nomes DNS. Estes dois espaços de nomes têm um conjunto de características e abstracções que têm alimentado a Internet para aquilo que é hoje. Contudo eles também têm uma série de deficiências, porque basicamente, tenta-se fazer tudo com eles. Uma grande sobrecarga semântica e a extensão de funcionalidade, têm complicado muito esse espaço de nomes [13].

A proposta do espaço de nomes Host Identity preenche uma lacuna importante entre os espaços de nomes IP e DNS, introduzindo uma nova camada na pilha protocolar entre a camada de transporte e a camada de rede (Figura 2.2). O espaço de nomes Host Identity assenta num conceito de Host Identifiers (HIs), representado através de uma chave pública de um par de chaves assimétricas. As chaves públicas são normalmente, mas não necessariamente, auto-geradas. Cada Host

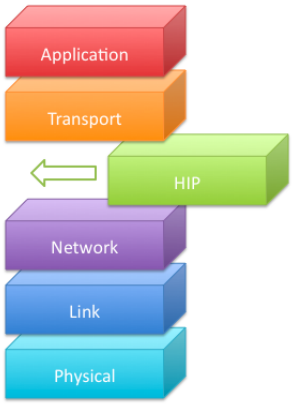

Figura 2.2: Camada HIP na pilha protocolar

tem pelo menos um Host Identity mas, normalmente, tem mais do que um. Cada Host Identity identifica um único Host, ou seja, não existem dois Hosts com o mesmo Host Identity. O Host Identity, e o correspondente Host Identifier, podem ser tanto público ou privado. Os sistemas cliente tendem a ter identidades públicas como privadas.

Há uma diferença pequena mas importante entre Host Identities e Host Identifiers. Uma Identity refere-se à entidade abstracta que é identificada. Um Identifier, refere-se concretamente ao padrão de *bits* que é utilizado no processo de identificação.

Embora os Host Identifiers possam ser usados em muitos sistemas de autenticação, como o protocolo IKEv2 [14], a arquitectura apresentada introduz um novo protocolo, chamado de Host Identity Protocol (HIP) [15]. O HIP prevê formas limitadas de confiança entre sistemas, mobilidade, multi-homing, e IP dinâmico.

Quando o HIP é usado, a carga real do tráfego entre dois Hosts é tipicamente, embora não necessariamente, protegido com IPsec. As Host Identities são usadas para criar a associação de segurança IPsec necessária e para autenticar os Hosts.

Como demonstra a Figura 2.3, o HIP fornece um método de separar a identificação da localização, papeis estes que actualmente são desempenhados pelos endereços IP. Os endereços IP continuam a funcionar como localizadores mas são os Host Identifiers que assumem o papel de identificadores do Host. É importante entender que os nomes do Host com base em Host Identifiers são ligeiramente diferentes dos nomes da interface. Um Host Identity pode ser simultaneamente acessível através de várias interfaces.

O protocolo SIP e HIP são muito úteis para sustentar a mobilidade mas em diferentes camadas. Assim, a junção dos dois protocolos não se torna conflituosa, muito pelo contrário, é como que um complemento um do outro. Mais concretamente, o protocolo SIP proporciona a mobilidade

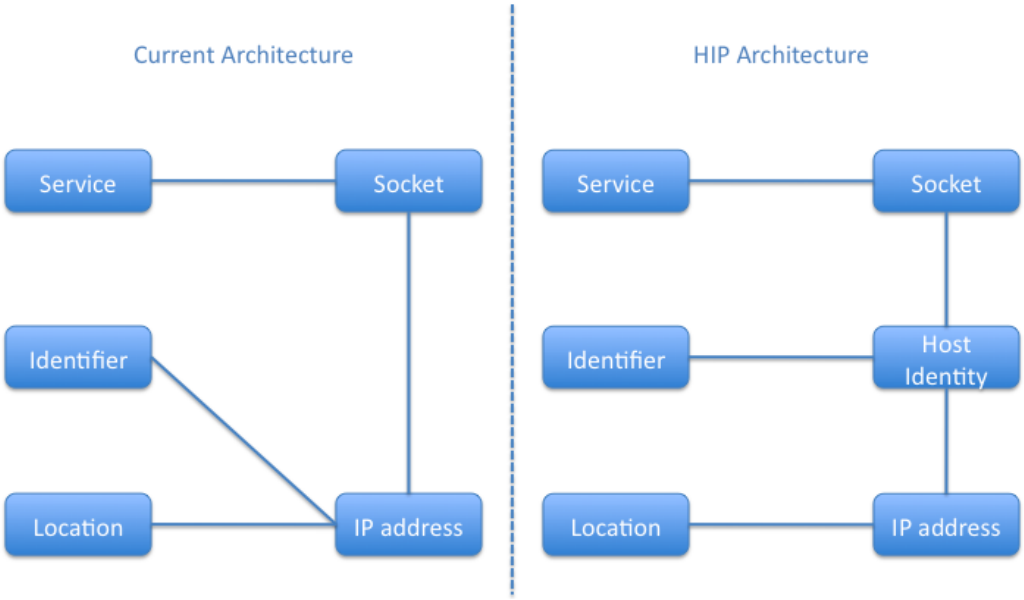

Figura 2.3: Arquitectura corrente vs Arquitectura HIP

de utilizadores e de sessões ao passo que o protocolo HIP proporciona mobilidade de terminal [16].

2.3.3 Proxy Mobile IPv6 (PMIPv6)

O PMIPv6 [17] é um protocolo de gestão de mobilidade ao nível da rede. A gestão de mobilidade baseado na rede permite a mobilidade de IP para um terminal sem que este necessite de enviar qualquer sinalização de mobilidade. A rede é responsável por gerir a mobilidade de IP em vez do terminal. As entidades de mobilidade na rede são responsáveis por monitorizar os movimentos do terminal e quando necessário iniciar a sinalização de mobilidade em nome do terminal.
Ao contrário do MIPv6 [18], que é um protocolo de mobilidade de terminal que exige funcionalidade de cliente ao terminal na pilha IPv6, a mobilidade baseada em rede suporta a mobilidade dos nós IPv6 sem o envolvimento do terminal na troca de mensagens de sinalização entre o terminal e o *home agent*.Por causa do uso e extensão da sinalização do Mobile IPv6 e das funcionalidade do *home agent*, este protocolo é chamado de PMIPv6.

As entidades funcionais principais da infra-estrutura de rede são o Local Mobility Anchor (LMA) e o Mobile Access Gateway (MAG). O LMA é responsável por manter o estado dos nós móveis acessíveis e é o ponto de ligação topológico para os nós móveis de prefixo de rede doméstica. O MAG é a entidade que realiza a gestão da mobilidade em nome de um nó móvel, e

que reside no ponto de acesso, onde o nó móvel está ligado. O MAG é responsável por detectar os movimentos do nó móvel e iniciar a sinalização de mobilidade para o LMA. Pode haver múltiplos LMAs num domínio PMIPv6, cada um servindo diferentes grupos de nós móveis.

2.3.4 Security Assertion Markup Language (SAML)

Desenvolvido pelo Comité Técnico para Serviços de Segurança da OASIS [19] , o SAML [20] é uma linguagem estruturada baseada em XML para comunicar a autenticação, autorização e atributos de um utilizador. O SAML permite que as entidades de negócios façam afirmações sobre a identidade, atributos e autorizações de um utilizador para outra entidade.
O SAML é um protocolo desenhado para ser flexível e extensível, e ser usado por outra normalização. A *Liberty Alliance*, o projecto *Internet2 Shibboleth* e o *OASIS Web Services Security (WS-Security) Technical Committee* adoptaram o SAML como base tecnológica para diversas finalidades.
A primeira versão do SAML (SAML v1.0) foi aprovada como uma norma OASIS em Novembro de 2002. Em Setembro de 2003, seguiu-se o SAML v1.1, que teve bastante sucesso e ganhou impulso na área dos serviços financeiros, ensino superior, governo e outros segmentos da indústria. O SAML tem sido bastante implementado pelos principais fabricantes de aplicações Web para gestão de acesso. É também usado em aplicações para servidores, assim como em *Web Services* e em segurança.
O SAML v2.0, aprovado como norma OASIS em Março de 2005, baseia-se no sucesso de unificar a estrutura para a federação de identidades definida no SAML v1.1 com a estrutura tanto da iniciativa Shibboleth como da Liberty Alliance's Identity Federation Framework. Como tal, a versão SAML v2.0 é um passo crítico para a convergência completa das normas de identidade federada. Esta versão também aproveitou a oportunidade para abordar as questões que surgiram a partir da experiência com versões anteriores e adicionado suporte para os recursos que tinha sido adiada por razões de calendário.

2.3.5 OpenID

O OpenID [21] permite a um utilizador usar uma conta já existente para entrar em múltiplos *websites*, sem que seja necessário criar novas *passwords*.
O utilizador pode escolher um OpenID *Provider* (que irá gerir as suas informações de autenticação) que melhor se adapte às suas necessidades e que ele confie. O utilizador pode mudar de

Provider sem que seja preciso mudar o seu OpenID.
O OpenID foi criado no verão de 2005 por uma comunidade *open source* para tentar resolver um problema que não foi facilmente resolvido por outras tecnologias de identificação existentes. Como tal, o OpenID é descentralizado e é uma tecnologia não propriedade. Hoje, qualquer pessoa pode optar por usar um OpenID ou tornar-se num OpenID *Provider* livremente, sem ter de se registar ou ser aprovado por qualquer organização.
A OpenID *Foundation* (OIDF) foi formada para promover, proteger e desenvolver a comunidade e tecnologia OpenID em Junho de 2007. A OIDF serve como uma organização de confiança do público que representam a comunidade aberta de programadores, fornecedores e utilizadores. A OIDF também auxilia a comunidade, fornecendo infraestruturas necessárias e ajuda na promoção e apoio à adopção expandida do OpenID. Isto implica a gestão da propriedade intelectual e de marcas, bem como fomentar o crescimento e participação global no crescimento do OpenID. São membros da OIDF, entre outros, os seguintes membros: AOL, Facebook, Google, Microsoft, MySpace, Sun, Yahoo. O OpenID tem alguns problemas ao nível da segurança, ao ser vulnerável a ataques de *Phishing* e *Man-in-Middle Attacks*. Este tipo de problemas são do conhecimento da OIDF, que alerta e produz recomendações a fim de minimizar estes problemas de segurança, como por exemplo usar o OpenID apenas em comunidades seguras, usar SSL com um certificado assinada por uma autoridade certificadora confiável e impor aos OpenID *Providers* que alertem e eduquem os seus utilizadores para este tipo de problemas.

2.3.6 OAuth

O OAuth [22] é um protocolo aberto que permite o acesso autorizado a aplicações *desktop* e *web* através da utilização de uma API segura. Blaine Cook deu início à sua implementação nos finais de 2006, enquanto trabalhava na implementação do OpenID para o Twitter. Em conjunto com Chris Messina procuravam uma forma de conjugar o uso do OpenID com a Twitter API para delegação de autenticação. Reuniram-se com David Recordon, Larry Halff entre outros no encontro CitizenSpace OpenID a fim de discutir as soluções existentes. Depois de analisarem as funcionalidades existentes no OpenID, chegaram à conclusão que não existia nenhum padrão aberto para a delegação de uma API de acesso.
Em Abril de 2007, foi criado o Google Group com um pequeno grupo de implementadores para escrever uma proposta de um protocolo aberto. Em Julho de 2007, a equipa elaborou uma especificação inicial e o grupo foi aberto a qualquer pessoa interessada em contribuir. Em Dezembro de 2007 surgiu a especificação final OAuth Core 1.0.

O OAuth permite a partilha de recursos privados (vídeos, fotografias, dados bancários, etc) armazenados num fornecedor de serviços, sem ser necessário fornecer o *username* e *password* de acesso. Desta forma cede-se o acesso a recursos privados a terceiros sem que exista uma partilha de identidade ou de alguma parte desta.
Actualmente o OAuth está implementado em todos os Google Data API. É também suportado por diversas organizações e entidades como a Google, Yahoo, Twitter, Flickr, Youtube, Orkut, entre outras.
O OAuth não é uma extensão do OpenID mas os dois podem ser usados em conjunto. O OAuth possibilita aos utilizadores gerirem permissões de acesso e utilização dos seus recursos, o OpenID centra-se na Identidade desses utilizadores, na gestão dessa identidade e na forma de um utilizador poder comprovar ser quem diz ser. Assim, estes dois mecanismos podem e devem trabalhar em conjunto, para bem dos utilizadores em geral.

2.4 Privacidade e Identidade

O mundo das comunidades na Internet abriu novas possibilidades de um utilizador publicar informações pessoais. No entanto, os utilizadores podem querer revelar informações pessoais para activar um determinado serviço mas também podem não querer revelar essas informações a qualquer pessoa. Para isso, torna-se imprescindível ter controle sobre quem vê essas informações [23]. Por vezes é preciso estabelecer uma sessão ocultando informações pessoais, tais como a identidade. Por exemplo, o João quer manter o anonimato quando liga para programa de televisão para dar a sua opinião sobre um assunto político. Para fazer isso, um utilizador pode enviar um convite com o campo de cabeçalho **De** definido como "anónimo"em vez do seu nome. No entanto, os fornecedores de serviços, normalmente, não permitem utilizadores anónimos para utilização dos seus serviços de encaminhamento. Cada utilizador precisa identificar-se no seu fornecedor de serviços mas manter-se anónimo para o resto da rede.
Os SIP User Agents podem solicitar diferentes tipos de privacidade de um fornecedor de serviços usando um campo do cabeçalho **Privacy** [24]. O fornecedor de serviços geralmente autentica o utilizador e, em seguida encripta, remove ou simplesmente muda o conteúdo dos campos do cabeçalho com informações pessoais sobre o utilizador [25].
Uma pessoa pode ser representada por um conjuntos de dados (atributos), que podem ser geridos através de meios técnicos, as chamadas identidades digitais. Dependendo da situação e do contexto, apenas um subconjunto desses atributos são necessários para representar uma pessoa, tanto no mundo físico como no mundo digital. Estas identidades são também conhecidas como

identidades parciais (digitais) [26]. Aqui podem separar-se vários grupos de identidades parciais, tais como, identidades de trabalho, lazer, etc. Um sistema de gestão de identidade fornece ferramentas para a gestão dessas identidades parciais no mundo digital.

Uma pessoa normalmente utiliza diferentes identidades parciais para o trabalho, para actividades de lazer (como praticar desporto) ou para lidar com o mundo empresarial (como bancos, livrarias, hipermercados). Algumas identidades parciais são estáticas (como a data de aniversário), enquanto outros podem mudar dinamicamente (como gostos, interesses). Dependendo do contexto da situação e do parceiro de comunicação, cada pessoa pode querer decidir qual a identidade parcial a ser usada nessa comunicação. Quando uma identidade parcial é escolhida podem ser vinculados nomes diferentes (como nomes de utilizador ou pseudónimos) [1]. A Figura 2.4 mostra um conjunto de atributos e as respectivas identidades parciais. Existe uma infinidade de

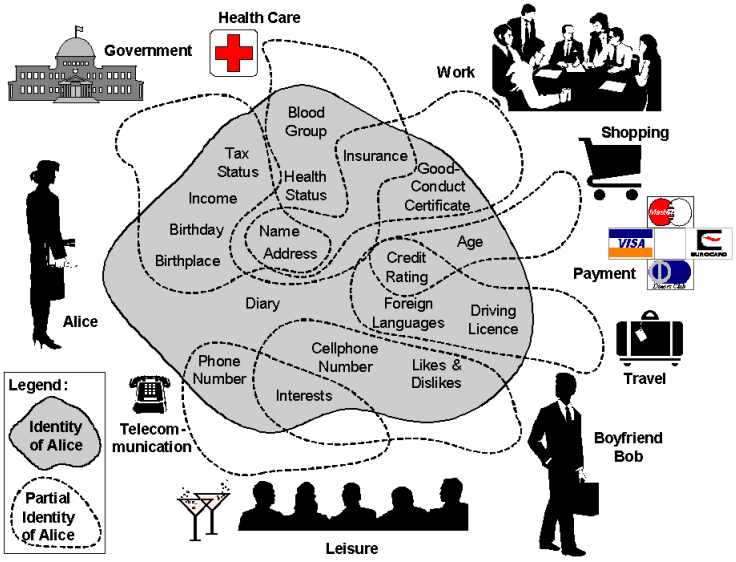

Figura 2.4: Identidades Parciais [1]

serviços *web* que introduzem novas ameaças à segurança. Um exemplo dessas ameaçar é o roubo de identidade, devido à reutilização de nomes de utilizador e senhas em contas diferentes. Além disso, existe a necessidade de melhorar a comodidade dos utilizadores, fornecendo os atributos necessários aos fornecedores de serviço de uma forma simples mas segura. Estas tendências facilitaram o desenvolvimento de soluções padrão e interoperáveis como a Gestão de Identidades [27].

As soluções existentes de Gestão de Identidades geralmente diferenciam três funções: Utilizador, Fornecedor de Identidades e Fornecedor de Serviços. O utilizador autentica-se no Fornecedor de Identidades com o qual tem um contrato pré-estabelecido. Com base numa relação de confiança

pré-estabelecida, o Fornecedor de Serviços confia nas declarações do Fornecedor de Identidades que expressa a autenticação do utilizador bem sucedida e não exige uma autenticação explícita. Este princípio é chamado de Single Sign-On (SSO). Além disso, o Fornecedor de Identidades pode fornecer serviços adicionais, como Single Sign-Off ou o fornecimento de atributos do utilizador para o Fornecedor de Serviços. A Gestão de Identidades fornece capacidades SSO e permite a troca de atributos básicos do utilizador entre os Gestores de Identidades e os Fornecedores de Serviços. Além disso, é comum a utilização de identificadores do utilizador hierarquicamente organizados, dividindo em identificadores dos Fornecedores de Identidades e uma parte específica do utilizador.
O protocolo HIP e Gestão de Identidades do utilizador são muito diferentes quando se considera a camada em que operam. Isso impõe vários desafios, mas deixa espaço para novas melhorias. No entanto, a integração do protocolo HIP com a Gestão de Identidades exige a resolução desses desafios. O primeiro desafio prende-se com os mecanismos de segurança adequados, que são necessários para permitir confiar na Identidade do Nó apresentada com base num sistema de Gestão de Identidades. O segundo desafio está relacionado com as diferentes estruturas do protocolo HIP e o espaço de nomes da Gestão de Identidades e seus identificadores.

2.5 Conceito do Terminal Virtual

O Conceito do Terminal Virtual proposto por Marc Barisch fornece uma vista integrada de um conjunto de dispositivos [2]. Utilizando algumas das arquitecturas e protocolos mencionados anteriormente, a arquitectura apresentada nesta secção, torna a tarefa de gestão diversos dispositivos simples. O Conceito do Terminal Virtual é uma junção de vários tipos de mobilidade como: Mobilidade de Identidade, Mobilidade de Sessão e Mobilidade de Terminal.

2.5.1 Descrição do Terminal Virtual

A introdução de identidades virtuais permite seleccionar e criar diferentes identidades digitais para diferentes contextos. O uso deste paradigma aumenta a complexidade para o utilizador. Além disso, o facto de que nem todas as identidades podem ser usadas em todos os dispositivos, diminui a usabilidade para os utilizadores. Se uma identidade virtual está presente em diferentes dispositivos, então é necessário fornecer informação adicional para seleccionar o dispositivo mais apropriado.

A identidade do Terminal Virtual, que consiste num dispositivo virtual e num identificador correspondente, é a agregação das identidades do dispositivo. Uma identidade do dispositivo representa qualquer tipo de dispositivo que é usado por um utilizador para usufruir de determinados serviços. Uma vez que um Terminal Virtual representa a agregação de um conjunto de dispositivos, então um Terminal Virtual é uma instanciação de um dispositivo especial. A Figura 2.5 mostra três categorias diferentes de recursos que é necessário descrever no contexto de um dispositivo.

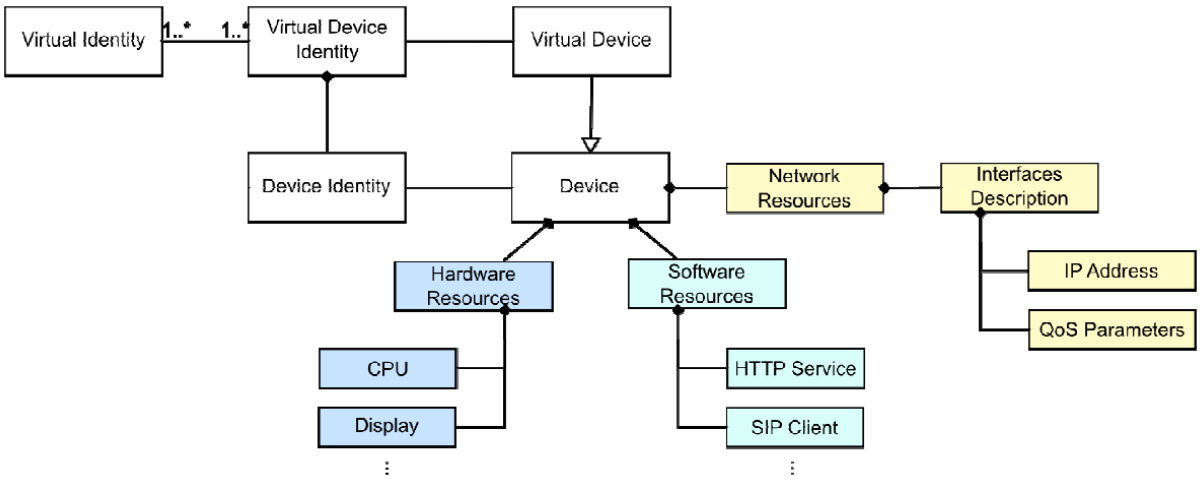

Figura 2.5: Descrição do Terminal Virtual [2]

Recursos de Hardware: Descreve todas as propriedades físicas de um dispositivo. Estas propriedades podem ser entre outras o poder de processamento do CPU ou o tamanho do ecrã.

Recursos de Software: Descreve os serviços particulares que estão disponíveis num dispositivo. Por exemplo se o dispositivo tem um cliente SIP instalado ou não. Este tipo de informações são necessárias para suportar mobilidade de sessão, por exemplo.

Recursos de Rede: Descreve os recursos de rede essenciais para a utilização dos serviços, bem como para organizar o Terminal Virtual. É necessário para descrever as interfaces de rede com as suas capacidades e características actuais. Exemplo de tais propriedades são os endereços IP ou os parâmetros de qualidade de serviço para uma determinada aplicação.

2.5.2 Papéis do Terminal Virtual

Um Terminal Virtual consiste em pelo menos dois dispositivos, em que apenas um deles é o dispositivo principal (*master*) como mostra a Figura 2.6. Este dispositivo tem responsabilidades que os restantes dispositivo não tem. Tais responsabilidades são:

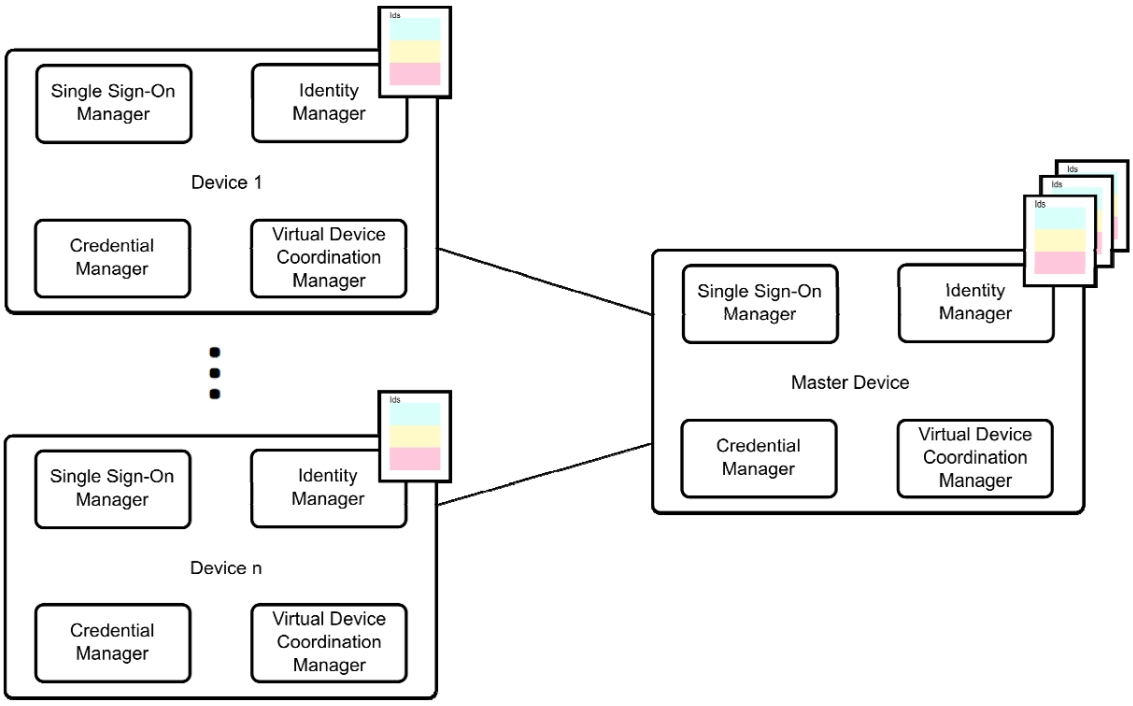

Figura 2.6: Conceito de Terminal Virtual [2]

Envolvimento no Estabelecimento de Sessões: O dispositivo primário é responsável pelo estabelecimento de novas sessões. Existem duas possibilidades diferentes para o estabelecimento de uma sessão.

No primeiro caso, um dos intervenientes na comunicação solicita o estabelecimento da sessão. Isto significa que o dispositivo primário tem a responsabilidade de seleccionar o dispositivo mais adequado para lidar com a solicitação do pedido de estabelecimento de sessão.

No segundo caso, o utilizador do Terminal Virtual solicita o estabelecimento da sessão. Para isso é necessário algum tipo de pré-selecção por parte do utilizador. No entanto, o dispositivo primário tem de ser informado para prestar assistência adicional de forma a optimizar a sessão de comunicação.

Descoberta de dispositivos dentro do Terminal Virtual: O dispositivo primário é responsável por obter informação sobre a disponibilidade do dispositivo secundário que faz parte do Terminal Virtual.

Transferência de identidade: O dispositivo primário desempenha um papel importante na transferência/mobilidade de identidade, tal como apresentado no Swift Deliverable 302 [28].

A selecção do dispositivo primário dentro de um Terminal Virtual precisa ser bem elaborada. Essa selecção pode ser estática ou dinâmica. Quando a selecção é feita de forma estática, um dos dispositivos do Terminal Virtual é sempre o dispositivo primário. No caso da selecção ser feita de forma dinâmica, essa selecção é efectuada seguindo alguns critérios apresentados abaixo:

Disponibilidade e Acessibilidade: O dispositivo primário é o membro mais importante do Terminal Virtual, por essa razão este dispositivo tem de estar sempre disponível e acessível.

Poder Computacional: O dispositivo primário deve ter um bom poder computacional, de forma a responder rápido e eficazmente a todos os pedidos.

Conectividade de Rede: Como todos os dispositivos estão interligados através do dispositivo primário, então o dispositivo primário deve ter uma conectividade boa e estável.

2.5.3 Contexto dos dispositivos dentro de um Terminal Virtual

Até agora as informações de contexto não foram consideradas. No entanto, algumas informações de contexto devem ser levadas em conta como:

Nível de segurança do dispositivo: Cada dispositivo do Terminal Virtual pode ter propriedades diferentes em termos de níveis de segurança. Este tipo de propriedades devem ser consideradas para quando for efectuadas, por exemplo, mobilidade de sessões, ser efectuada de forma segura.

Contexto de utilização: Um dispositivo pode ser usado em diversos contextos. Posto isto, é necessário garantir que o dispositivo está a ser usado no contexto correcto, isto é, se está a ser usado num contexto empresarial, pessoal ou qualquer outro bem definido.

2.5.4 Conceito de Sessão

Uma das responsabilidades da arquitectura do Terminal Virtual é optimizar e gerir correctamente as sessões activas. Isto pode ser, por exemplo, o controlo da mobilidade de sessão entre dois dispositivo, de forma a optimizar os recursos e dar ao utilizador uma melhor experiência de utilização.

A Figura 2.7 ilustra as dependências de uma sessão ao longo do tempo. A fim de estabelecer uma sessão de serviço, é necessário uma sessão de um Fornecedor de Identidades (IdP). Por

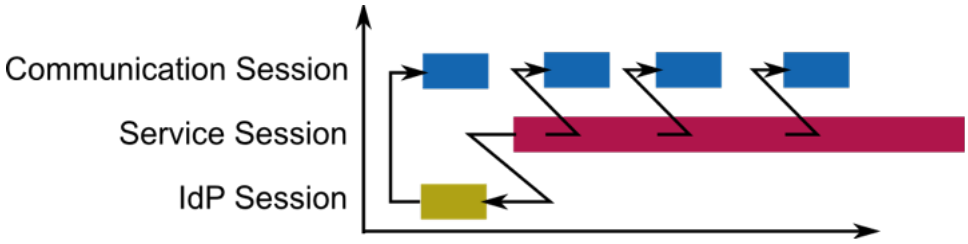

Figura 2.7: Dependências de uma sessão atómica ao longo do tempo [2]

outro lado, uma sessão IdP exige uma sessão de comunicação, que permite autenticação e restabelecimento das credenciais de autenticação necessárias (por exemplo, Single Sign-On). Com base nisso, uma sessão de serviço pode provar que também depende de sessão de comunicação. Uma sessão de serviço não precisa necessariamente de uma sessão de comunicação ao longo de toda a sua existência.

A Figura 2.8 mostra que uma sessão de serviço consiste pelo menos numa sessão de aplica-

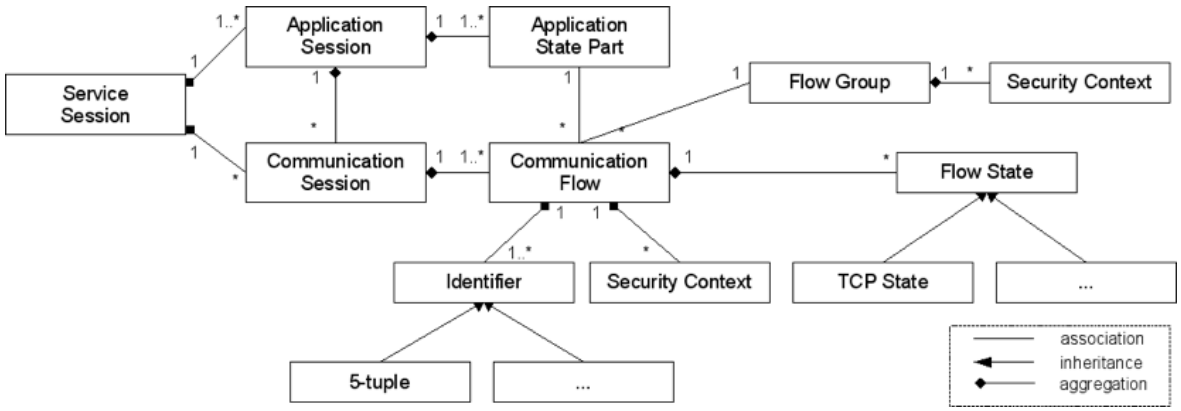

Figura 2.8: Modelo de Sessão [2]

ção. No caso de serviços de não-comunicação, por exemplo, aplicações num dispositivo sem interacção do utilizador através da rede ou serviços que só comunicam temporariamente, não há

necessidade de uma sessão de comunicação. Uma sessão de aplicação é composta pelo menos por uma parte do estado da aplicação. Várias partes do estado da aplicação podem existir para diferenciar, por exemplo, o proprietário controlo e do conteúdo no caso de uma comunicação VoIP. Uma parte do estado da aplicação, por sua vez, pode ter associações de comunicação que constroem a sessão de comunicação. A comunicação precisa ser identificada, por exemplo, pelo chamado 5-tuplo que consiste no endereço IP origem e destino, na porta origem e destino e o protocolo de transporte. Por exemplo, no caso do TCP também existe um estado de fluxo. Além disso, pode ter um contexto de segurança associado, por exemplo, para reflectir um associação TLS [29]. Os fluxos de comunicação de sessões de comunicação diferentes podem depender uns dos outros. Por exemplo, no caso de uma associação de segurança na camada de rede entre dois dispositivos, é permitida que dependam uns dos outros. O modelo introduzido mostra que são necessárias soluções para transferir não só o estado da aplicação mas também o estado contido nas camadas de transporte e de rede.

2.6 Sumário

Este capítulo apresentou o estado da arte estudado, no qual foi apresentada a mobilidade de objecto e a mobilidade de serviço, que constituem o conceito de mobilidade. A mobilidade de objecto foi descrita de forma a separar os quatro tipos de mobilidade que a compõem, são eles: a mobilidade de terminal, a mobilidade identidade, a mobilidade de rede e, por fim, a mobilidade de sessão. Da mesma forma, a mobilidade de serviço foi descrita sob a forma de mobilidade contínua, como é o exemplo dos *handovers*, e de mobilidade descontínua, que se refere a um serviço que não preserva o seu estado actual.

Neste capítulo foram apresentadas tecnologias que podem ser utilizadas para possibilitar a mobilidade de terminal, mobilidade de identidade e mobilidade de sessão. Foram então apresentadas as tecnologias SIP, HIP, PMIPv6, SAML, OpenID e OAuth. O SIP é um protocolo de controlo da camada da aplicação que pode estabelecer, modificar e terminar sessões multimédia. O HIP é um protocolo que visa a introdução de uma nova camada na pilha protocolar entre a camada de transporte e a camada de rede, por forma a que seja fornecido um método de separar os papeis de identificação e de localização, que actualmente são desempenhados apenas pelos endereços IP. O PMIPv6 é um protocolo de gestão de mobilidade baseado na rede. Esta gestão de mobilidade permite a mobilidade de um IP para um dispositivo sem que este necessite de enviar qualquer sinalização de mobilidade, deixando do lado da rede a responsabilidade da gestão de mobilidade. A tecnologia SAML é uma linguagem estruturada baseada em XML para comunicar a autenti-

cação, autorização e atributos de um utilizador, que permite que as entidades de negócios façam afirmações sobre a identidade, atributos e autorizações de um utilizador para outra entidade. O OpenID permite ao utilizador usar uma conta já existente para entrar em múltiplos portais, sem que seja necessário criar novas *passwords*. O OAuth é um protocolo aberto que permite o acesso autorizado a aplicações *desktop* e web através da utilização de uma API segura.
Neste capítulo foram também apresentadas questões relacionada com privacidade e identidade do utilizador, que vão ao encontro da necessidade de criar identidades parciais. Por último, foi apresentado o conceito do Terminal Virtual proposto por Marc Barisch, o qual foi seguido com especial atenção, pelo facto da arquitectura do Terminal Virtual proposta nesta dissertação, ter sido elaborado tendo por base esta abordagem.

Capítulo 3

A arquitectura do Terminal Virtual

3.1 Introdução

O aumento significativo do número de dispositivos que cada utilizador tem hoje em dia, leva a que na sua maioria não seja feita a melhor gestão de cada um desses dispositivos. Por outro lado, são oferecidos cada vez mais serviços em diferentes plataformas, o que leva os utilizadores a criar cada vez mais contas, para as quais é necessárias fornecer dados privados dos utilizadores (como nome, idade, morado, contacto, entre outros). Todas as contas de um determinado utilizador estão protegidas por passwords. Devido ao aumento do número de contas de cada utilizador, estes acabam por cometer erros (por vezes graves) quando escolhem a password. O uso da mesma password em diversas contas é o erro mais comum, no entanto existem outros bastante típicos, como passwords sequenciais (ex: 123456), nome e data de nascimento. Mais concretamente, e a título de exemplo, este tipo de erros leva muitas vezes a que um utilizador use a mesma password num site sem qualquer tipo de segurança e para aceder à conta bancária *online*.
Partindo destes cenários e tendo em conta o estado da arte estudado (em particular a abordagem proposta por Marc Barisch [2]), neste capítulo apresenta-se uma arquitectura que ofereça ao utilizador uma visão única de todos os dispositivos, denominada por "Arquitectura do Terminal Virtual". Neste capítulo é apresentado também a integração do Terminal Virtual com o gestor de identidades da PT Inovação (GIPTIN). Por fim, é apresentado o modelo de dados que dá suporte à arquitectura proposta, assim como a implementação da arquitectura. São também demonstrados cada componente da arquitectura e o papel de cada um deles dentro do Terminal Virtual. O Terminal Virtual é um conjunto de dispositivos pertencentes ao mesmo utilizador e estão ligados entre si através do servidor de terminais virtuais denominado por "Access Server".

3.2 Arquitectura do Terminal Virtual

O Terminal Virtual é responsável por fazer a gestão e coordenação dos dispositivos de um utilizador, assim como a gestão das suas identidades. A elaboração da arquitectura do Terminal Virtual passou por várias etapas até chegar à arquitectura final, a qual foi implementada.
A arquitectura do Terminal Virtual teve a sua primeira versão composta por duas componentes fundamentais: o conjunto de dispositivos do utilizador e o Access Server, como pode ser visto na Figura 3.1. O conjunto dos dispositivos do utilizador são os dispositivos físicos pertencentes

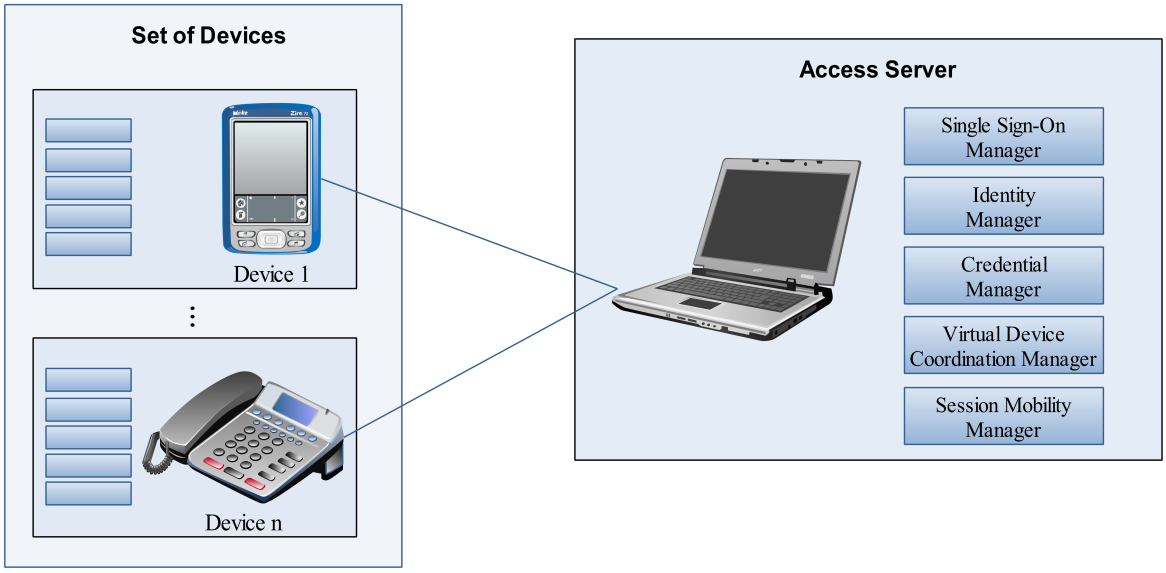

Figura 3.1: Arquitectura do Terminal Virtual (Versão Inicial)

ao utilizador. O Access Server é o responsável pela gestão e manutenção do Terminal Virtual e é também ele um dispositivo físico do utilizador. Assim, qualquer dispositivo poderia ser o Access Server, definindo-se desta forma uma arquitectura distribuída. Posto isto, o Access Server seria constituído por cinco componentes: o gestor de SSO, gestor de identidades, gestor de credenciais, gestor de coordenação dos dispositivos virtuais e gestor de mobilidade de sessão.
O gestor de SSO é a componente responsável por gerir todas as sessões autenticadas e por efectuar o SSO dentro do mesmo dispositivo e entre dispositivos diferentes.
O gestor de identidades é a componente responsável por adicionar, remover e editar as identidades de um determinado utilizador. Estas identidades podem ser apenas identidades parciais. É também da responsabilidade deste componente associar as identidades do utilizador aos respectivos serviços.

O gestor de credenciais é a componente responsável por gerir todas as credenciais de um determinado utilizador, sejam elas palavra-passe, *CardSpace*, certificados, entre outros.
O gestor de coordenação dos dispositivos virtuais é a componente responsável por gerir de forma eficiente todas os dispositivos do Terminal Virtual, com base nas características dos dispositivos, previamente definidas.
O gestor de mobilidade de sessão é a componente responsável por assegurar a mobilidade de sessão entre os dispositivos do Terminal Virtual. Antes de ser efectuada a mobilidade de sessão, existem parâmetros que têm de ser validados (como disponibilidade do outro dispositivo, condições de segurança, entre outras). O gestor de mobilidade de sessão é que é responsável por essas verificações.
De forma a simplificar a componente que fica instalada nos dispositivos do utilizador e pelo facto de que nem todos os dispositivos físicos do utilizador têm capacidade para efectuar a gestão do Terminal Virtual, optou-se por redefinir a arquitectura, tornando-a numa arquitectura centralizada no Access Server, que passa agora a ser um servidor que não pertence ao utilizador, dando origem à arquitectura representada na Figura 3.2. Assim, foi apenas colocado nos dispositivos do

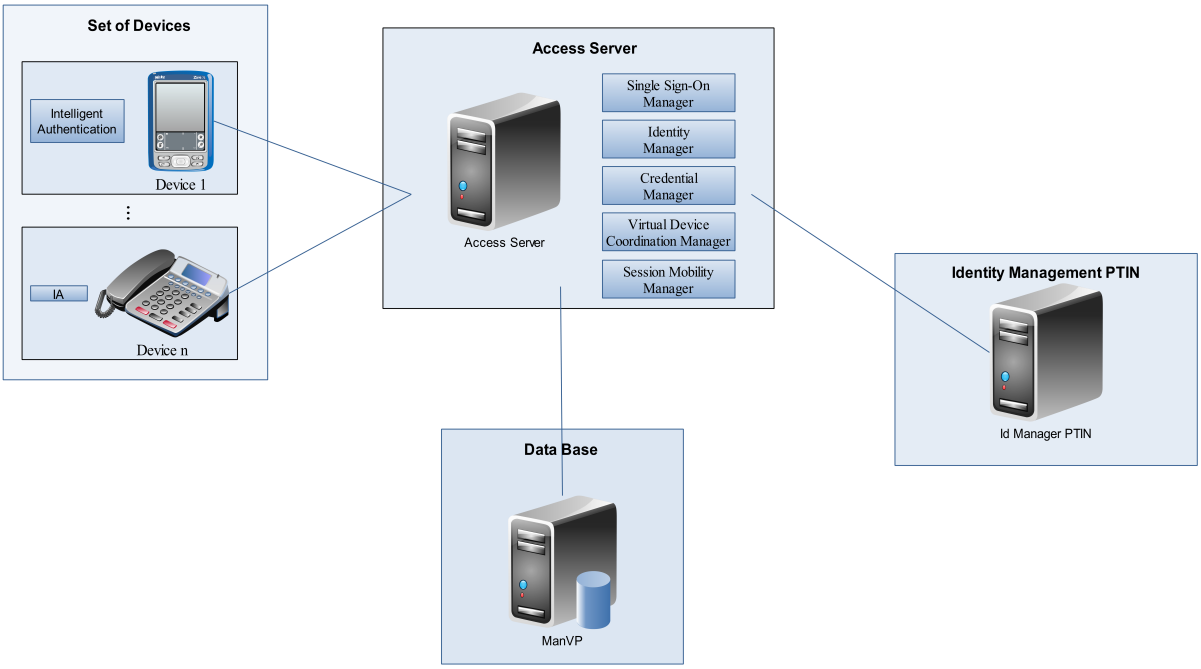

Figura 3.2: Arquitectura do Terminal Virtual (Versão Intermédia)

utilizador um gestor de autenticação, denominado por "Intelligent Authentication". Este gestor não é mais que um serviço que o utilizador activa/desactiva no seu dispositivo. Uma vez activo, o

gestor fica em espera, e sempre que for necessário efectuar uma autenticação, este é responsável por comunicar ao Access Server quem é o utilizador, qual é o dispositivo que está usar e qual a sessão activa.

Posteriormente, fez-se a integração do Terminal Virtual com o GIPTIN apresentado na Secção 3.3, para ter acesso às identidades virtuais de cada utilizador e as respectivas contas associadas. Como o GIPTIN não disponibiliza toda a informação necessária para a implementação do Terminal Virtual, como por exemplo, não permite agrupar as identidades virtuais de um utilizador. Por outro lado, também não existe qualquer registo dos dispositivos do utilizador, pelo que é extremamente complicado saber qual é o dispositivo que o utilizador está a usar a cada momento. Para tal, foi adicionado uma base de dados para suportar as informações relacionadas com o agrupamento das identidades virtuais e dos dispositivos do utilizador e também a associação das identidades virtuais a cada dispositivo. Para efectuar a gestão das identidades e dos dispositivos do utilizador, foi introduzida mais uma componente denominada por "MultiManager".

Para que esta arquitectura pudesse ser implementada, foi necessário a inclusão de mais uma componente que ficará nos servidores dos portais *web*, denominada por "WebGate". A inclusão desta componente deu origem à arquitectura final do Terminal Virtual, representada na Figura 3.3.

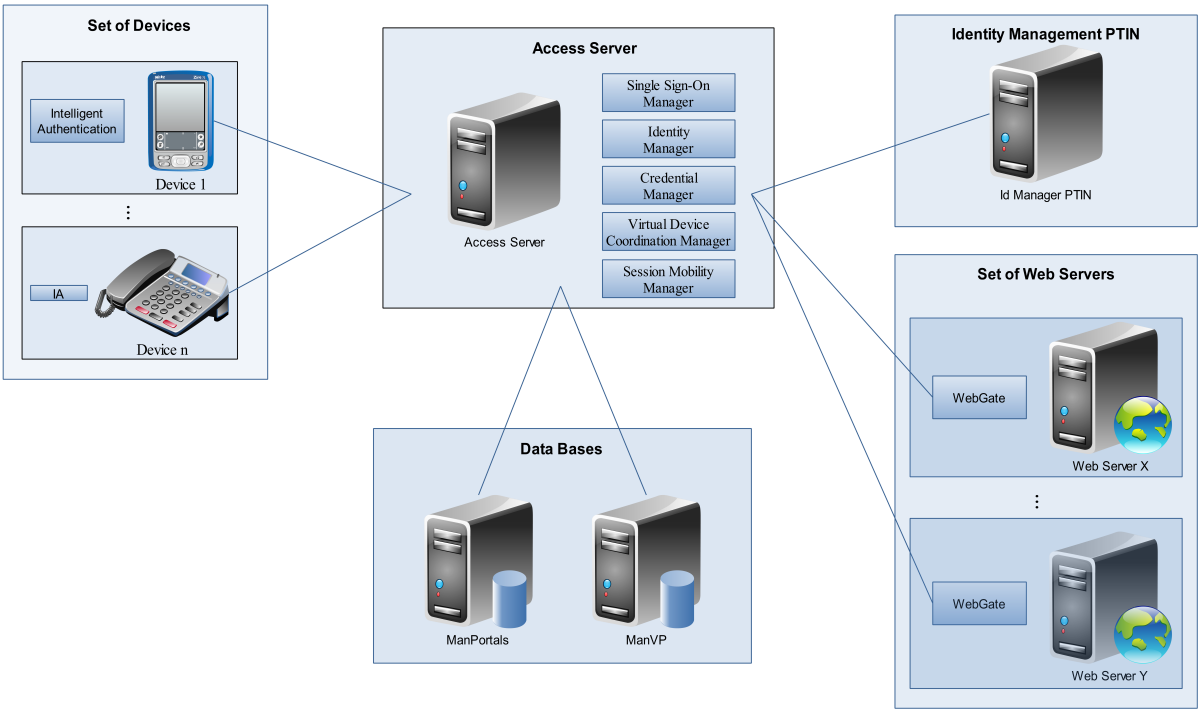

Figura 3.3: Arquitectura do Terminal Virtual (Versão Final)

A implementação desta arquitectura, no que diz respeito ao Access Server, foca-se essencialmente nas componentes relacionadas com a identidade dos utilizadores, mais concretamente o gestor de SSO, o gestor de identidades e o gestor de credenciais. Pelo que as outras duas componentes (gestor de coordenação dos dispositivos virtuais e gestor de mobilidade de sessão) não foram consideradas nesta implementação.

3.3 Integração com o gestor de identidades da PT Inovação

Actualmente, existe na PT Inovação um gestor de identidades que permite a gestão de identidades e privacidade, centrada no utilizador, num cenário de multi-fornecedores de serviços. Este sistema permite aos utilizadores gerir, de uma forma simples e eficaz, as diferentes contas em diferentes serviços das quais são donos, permitindo assim:

- Associar contas de serviços diferentes sobre uma mesma identidade virtual;
- Gerir regras de autorização que permitam a gestão do acesso por terceiros (ex. serviços web2.0) a partes ou totalidade das suas identidades.

Ao garantir um local único para gestão de toda a sua informação digital, é assegurado ao utilizador uma visão unificada dos diferentes serviços contratados e das regras de autorização que gerem o acesso à informação de cada utilizador.
Do ponto de vista do negócio, estes mecanismos potenciam não só a existência de serviços autorizados pelo utilizador, como também garantem o acesso à sua informação por terceiros, possibilitando cenários de convergência de serviços. O utilizador do sistema será, em qualquer dos casos, o ponto de decisão.
A existência de um elemento centralizado, em que o utilizador gere as suas identidades virtuais, não significa que o sistema de gestão de identidades detém a informação referente às identidades dos utilizador, nem as credenciais de acesso a cada um dos serviços individuais. Pelo contrário, ao utilizar tecnologia/standards apropriados - OAuth e SAML - o gestor de identidades é capaz de manter um mapeamento e gerir as regras de autorização tendo acesso apenas a uma pequena parte da informação do utilizador. As credenciais de acesso do utilizador aos diferentes serviços nunca são conhecidas pelo gestor de identidades, nem sequer durante o mapeamento. O gestor de identidades guarda apenas apontadores para a informação privada dos utilizador, continuando esta guardada em cada um dos serviços envolvidos.

É da responsabilidade do gestor de identidades a criação e manutenção das identidades virtuais (*Virtual Personas*). Uma *Virtual Persona* representa um mapeamento entre duas ou mais contas do utilizador. Um utilizador pode ter várias *Virtual Personas*, as quais pode definir se estão activas ou desactivas num determinado dispositivo.

No GIPTIN, não existe uma forma de agrupar as *Virtual Personas* de um determinado utilizador, ou seja, as *Virtual Personas* são independentes entre si. Para tal, foi criada uma nova estrutura de forma a poder agrupar as *Virtual Personas* de um determinado utilizador.

A Figura 3.4 representa a estrutura que relaciona as *Virtual Personas* com os utilizadores (Perso-

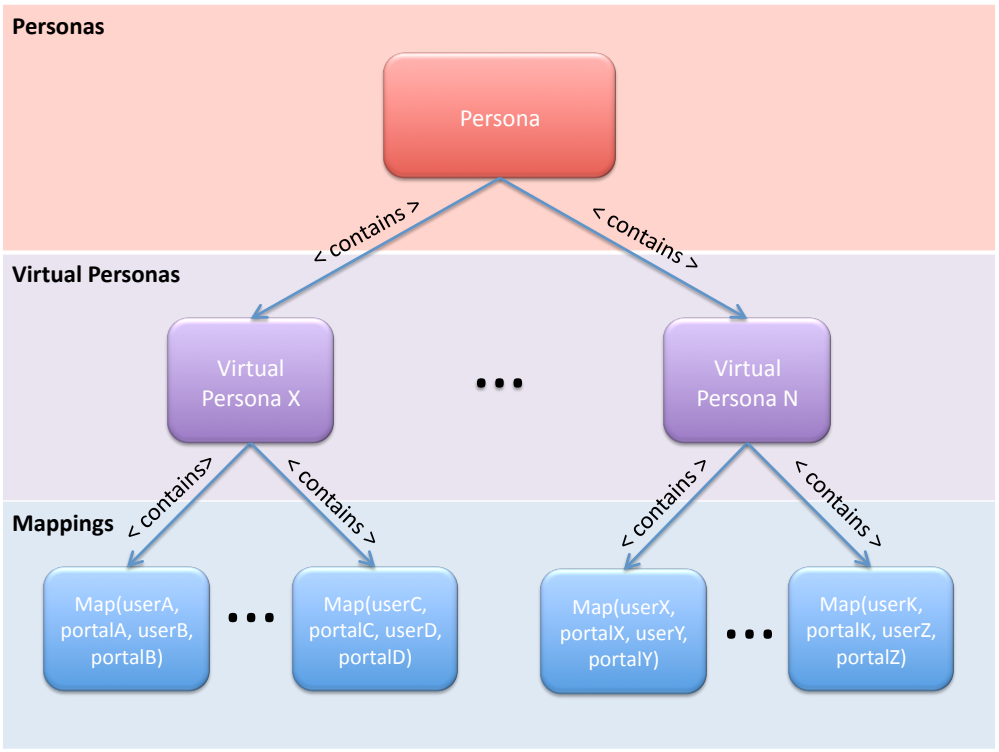

Figura 3.4: Hierarquia de Identidades

nas). Esta estrutura está dividida em três componentes: Personas, Virtual Personas e Mappings. Uma Persona é uma representação do utilizador como pessoa e cada pessoa pode ter várias *Virtual Personas*. Desta forma, um utilizador tem um identificador único que ajudará na gestão das suas *Virtual Personas*.

As *Virtual Personas* são uma representação dos mapeamentos das contas dos utilizadores. Uma *Virtual Persona* pode agrupar várias contas do utilizador, de forma a permitir uma gestão adequada de cada uma delas e definir regras para a sua utilização.

Por fim, os Mappings são os próprios mapeamentos entre os diversos *usernames* das contas dos utilizadores. Um mapping disponibiliza informação como por exemplo: o "carlosmatos"no portal A é o "cmatos"no portal B.
De forma a efectuar a integração da arquitectura do Terminal Virtual com o GIPTIN, são usados *web services* que o gestor de identidades disponibiliza e que fornecem informação acerca dos utilizadores, das suas *Virtual Personas* e dos mapeamentos das contas.
Com esta arquitectura é possível efectuar a mobilidade de identidades, de forma a permitir SSO entre portais dentro do mesmo dispositivo e entre dispositivos diferentes.

3.4 Modelo de Dados

Para que a arquitectura proposta na Secção 3.2, representada na Figura 3.3, pudesse ser implementada, foi necessário a criação de dois modelos de dados. Um para suporte da informação relativa aos portais e outro para suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*.

3.4.1 Modelo de dados para suporte da informação relativa aos portais

Para cada portal é necessário definir diversa informação, como as base de dados para autenticação, endereço do portal, atributo que identifica um utilizador, entre outros.
O modelo de dados para suporte da informação relativa aos portais, representado na Figura 3.5,

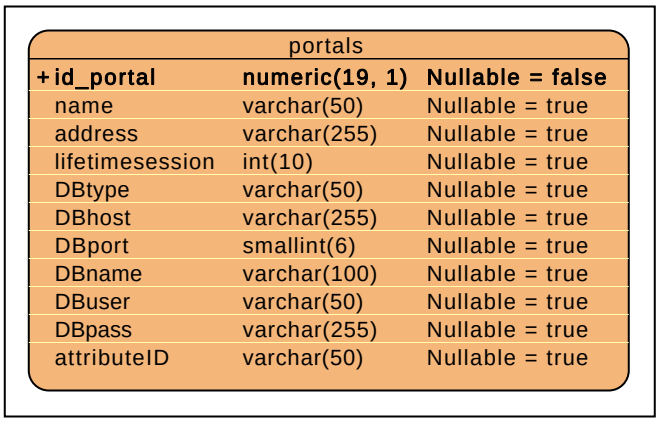

Figura 3.5: Modelo de dados para suporte da informação relativa aos portais

é constituído apenas por uma entidade denominada por *portals*. Esta é composta pelos atributos

IDportal que é o identificador do portal, *name*, pelo *address* que é o endereço do portal e pelo *lifetimesession* que indica qual o tempo que uma sessão pode estar activa. Faz parte também da entidade *portals* a informação relativa à base de dados de autenticação de cada portal. São então guardadas no atributo *DBtype* as informações acerca do tipo de base de dados que é utilizada e a sua localização, como sendo o endereço (*DBhost*), porta (*DBport*) e nome da base de dados (*DBname*). O *username* (*DBuser*) e *password* (*DBpass*) da base de dados também são necessários. É também guardado um atributo muito importante (*attributeID*), que identifica qual é o campo da base de dados que fornece o identificador de cada utilizador do portal. Este identificador será passado às páginas dos portais, para que estes possam disponibilizar a informação relativa a cada utilizador.
O modelo de dados para suporte da informação relativa aos portais apresentado, apenas necessita que os portais associados disponibilizem informação referente à autenticação dos utilizadores e os identificadores correspondestes.

3.4.2 Modelo de dados para suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*

A fim de se poder fazer a gestão e manutenção das identidades dos utilizadores, surgiu a necessidade de criar uma estrutura de suporte que contemplasse as informações mínimas mas estritamente necessárias à identificação de um utilizador.
Foi também indispensável a inclusão de informações relativas aos dispositivos dos utilizadores neste estrutura de suporte, uma vez que no GIPTIN não existe qualquer informação acerca dos dispositivos dos utilizadores. Isto deve-se ao facto do GIPTIN focar-se, essencialmente, na gestão e manutenção de identidades. Portanto, para que a arquitectura do Terminal Virtual proposta na Secção 3.2 e representada na Figura 3.3 pudesse ser implementada, foi necessário criar uma estrutura de suporte para poder efectuar também a gestão e manutenção dos dispositivos dos utilizadores.
A estrutura referida levou à elaboração do modelo de dados para suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*, tendo por base a Hierarquia de Identidades, definida na Secção 3.3 e representada na Figura 3.4 e as informações dos dispositivos dos utilizadores.
A construção deste modelo de dados teve em consideração a informação relativa às *Virtual Personas* que o GIPTIN disponibiliza através dos seus *web services*, permitindo assim tirar partido da informação já existente no GIPTIN. O modelo de dados aqui apresentado, focou-se no agru-

pamento das *Virtual Personas*, por forma a ser possível identificar quais são as *Virtual Personas* de um determinado utilizador. Faz também parte deste modelo, os dispositivos dos utilizadores e as características correspondentes a cada um deles (como o tipo de dispositivo e quais as suas interfaces de rede activas). Por fim, mas não menos importante, permite guardar a forma como as *Virtual Personas* de um determinado utilizador estão associadas aos seus dispositivos, e assim definir regras para gestão das *Virtual Personas* de forma a que, automaticamente, num determinado dia da semana, a uma determinada hora, um dispositivo particular do utilizador tenha uma identidade associada.

O modelo de dados resultante é constituído por sete entidades e está representado na Figura 3.6. A entidade *personas*, tal como na Hierarquia de Identidades (Figura 3.4), representa os utiliza-

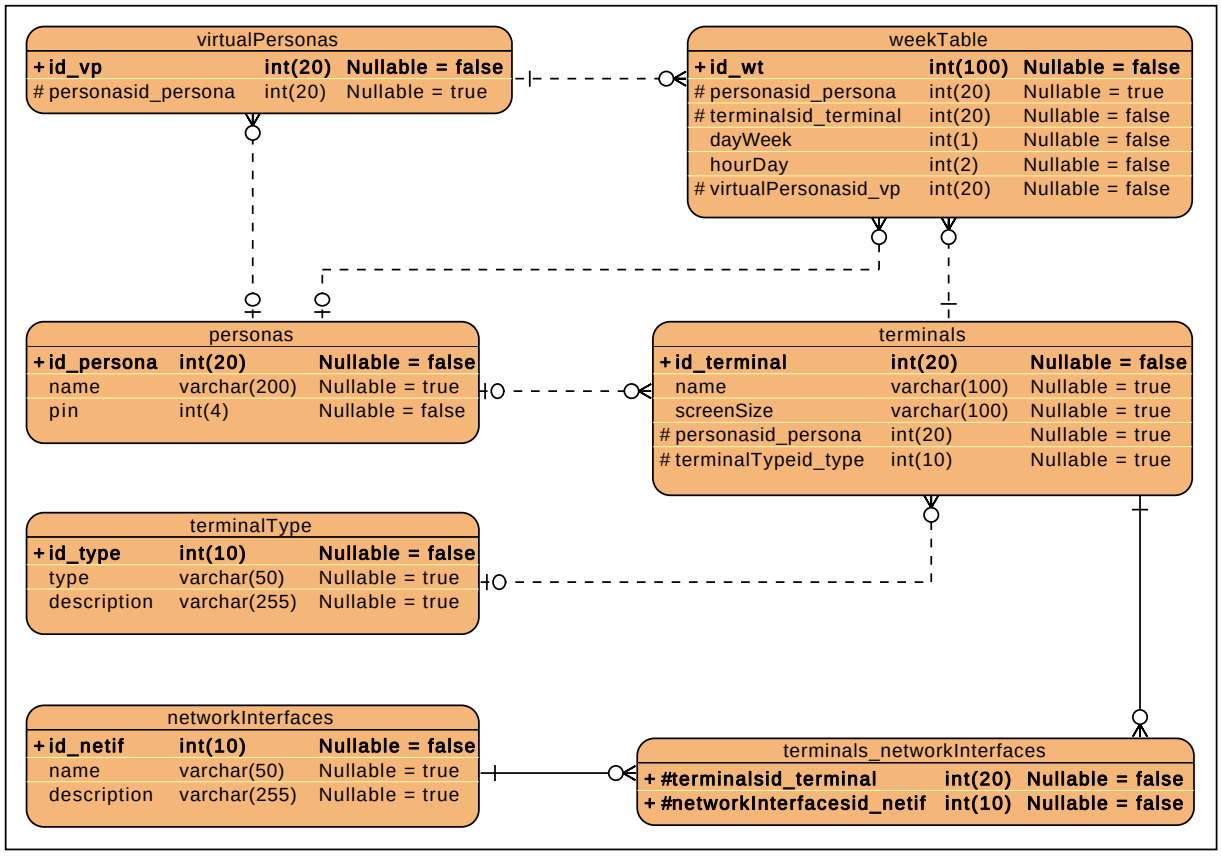

Figura 3.6: Modelo de dados para suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*

dores como pessoa. Esta entidade é composta apenas por três atributos, o *id_persona* que é um identificador único do utilizador (pessoa real), o *name* e o *pin* que em conjunto com o *id_persona* são as credenciais para acesso ao portal de gestão de multi-dispositivos.

Cada utilizador pode ser várias *Virtual Personas*. Essas *Virtual Personas*, que são concedidas pelo GIPTIN, são guardadas na entidade *virtualPersonas*. Nesta entidade existe apenas o atributo *id_vp* que é o identificador da *Virtual Persona* e o *personasid_persona* que diz quem é o utilizador a quem pertence a *Virtual Persona*.
É natural um utilizador ter mais do que um dispositivo. A informação desses dispositivos é armazenada na entidade *terminals*. Esta entidade contempla diversa informação acerca dos dispositivos como o *id_terminal*, o *name* e o *screenSize*, que representa as dimensões do ecrã do dispositivo. Faz também parte desta entidade o atributo *personasid_persona* que relaciona o dispositivo com o utilizador a quem este pertence. Como existem vários tipos de dispositivos (ex: telemóvel, computador, entre outros), foi criada uma entidade denominada por *terminalType* que regista todos os tipos de dispositivos, permitindo assim associa a cada dispositivo o tipo correspondente.
Cada vez mais, é habitual os dispositivos terem várias interfaces de rede. Este facto levou à criação da entidade *networkInterfaces* na qual são guardados todos os interfaces de redes. Como cada dispositivo pode ter associado várias interfaces de rede e cada interface de rede pode estar associada a vários dispositivos (ou seja, relacionamento *n* para *m*), foi necessário criar uma entidade (*terminals_networkInterfaces*) para relacionar as duas entidades.
A entidade responsável por registar a informação relativa às *Virtual Personas* que estão activas num determinado dispositivo, é a entidade *weekTable*. Esta entidade é constituída por seis atributos e está relacionada com as entidades *virtualPersonas*, *personas* e *terminals*. Os atributos são *id_wt*, *personasid_persona* que identifica o utilizador, *terminalsid_terminal* que identifica o terminal, *dayWeek* que indica o dia da semana, *hourDay* que indica a hora do dia e por fim *virtualPersonasid_vp* que regista a *Virtual Persona*. Desta forma, é possível definir regras para a associação das *Virtual Personas* a cada um dos diferentes dispositivos do utilizador.

3.5 Implementação

Depois da elaboração final da arquitectura do Terminal Virtual (Figura 3.3) e da definição dos modelos de dados utilizados e da integração do GIPTIN com a arquitectura, é agora apresentada a sua implementação.
Seguindo a arquitectura proposta, a implementação do Terminal Virtual divide-se na implementação de quatro componentes: Access Server, Intelligent Authentication, WebGate e MultiManager. A forma como estas componentes estão relacionadas entre si e o tipo de comunicação usado entre elas, está representado na Figura 3.7.

A componente Access Server é um servidor que tem a responsabilidade de verificar as creden-

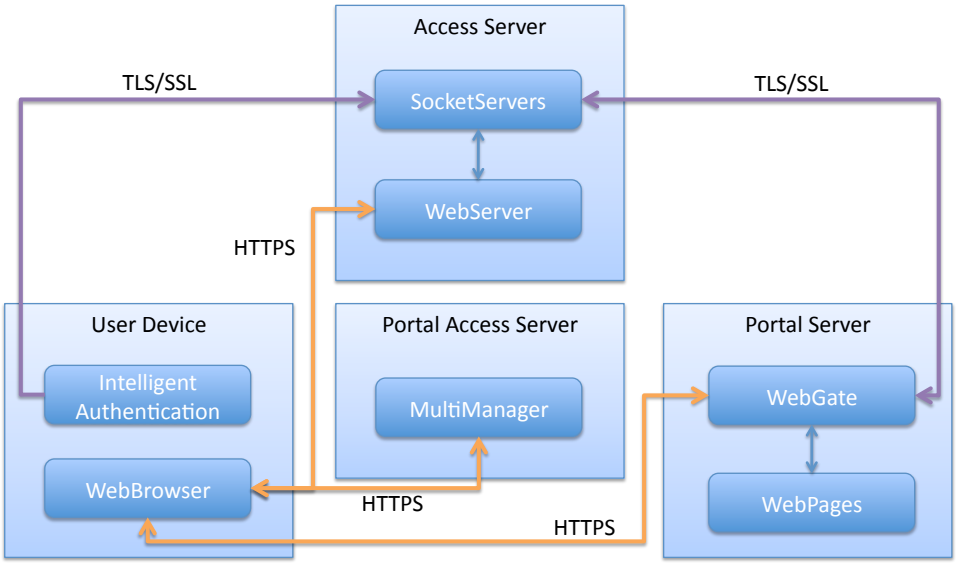

Figura 3.7: Componentes do Terminal Virtual

ciais dos utilizadores, gerir o SSO, entre outras. Estas verificações são efectuadas na subcomponente *WebServer*, através de *redirects* efectuados pela componente WebGate dos pedidos HTTP [30]. Esta comunicação é efectuada via *web*, utilizando ligações HTTPS [31]. A subcomponente *SocketServers* inclui as componentes que fazem a ligação, através de *sockets* SSL, às componentes Intelligent Authentication e WebGate. O servidor Access Server é independente dos servidores dos portais e dos dispositivos dos utilizadores. Já a componente Intelligent Authentication está integrada nos dispositivos (*User Device*) que os utilizadores usam para efectuar os pedidos HTTP. Estes pedidos podem ser efectuados através de um *WebBrowser* por exemplo.

A componente Intelligent Authentication é responsável por enviar para o Access Server a informação necessária, de forma a identificar do utilizador e do terminal que este está a utilizar. Por fim, a componente WebGate é responsável por verificar e determinar se um pedido de acesso a um determinado portal é permitido ou negado e por enviar para o Access Server as sessões que se pretenderem invalidar.

A componente WebGate está integrada nos servidores dos portais (*Portal Server*) onde estão as páginas *web* (*WebPages*). Comporta-se como um filtro à entrada dos respectivos portais. Tanto a componente Intelligent Authentication como a WebGate utilizam *sockets* SSL para se ligarem ao Access Server.

Através da componente MultiManager, os utilizadores podem gerir os seus dispositivos e asso-

ciar a cada um deles *Virtual Personas*. Com esta componente é disponibilizada ao utilizador uma vista global de todos os dispositivos de todas as suas *Virtual Personas*. Esta componente não está directamente ligada a nenhuma das componentes referidas anteriormente, está apenas ligada às bases de dados e ao GIPTIN, que não estão representados neste diagrama.
Para a implementação da arquitectura do Terminal Virtual foi utilizada uma linguagem de programação orientada a objectos, mais concretamente a linguagem de programação Java. Esta implementação faz uso de IDs de sessão, *cookies*, IDs de utilizadores (*personas*) e *Virtual Personas* que são partilhados entre as três componentes, a fim de gerir as identidades dos utilizadores e permitir o acesso aos portais que o utilizador pretende aceder. De forma a auxiliar na descrição da implementação do Terminal Virtual, são utilizados alguns diagramas de classe.

3.5.1 Implementação do Access Server

O Access Server é um servidor que é responsável pela gestão do SSO e gestão de identidades e credenciais do utilizador. É o elemento central de toda a arquitectura.
A implementação do Access Server foca-se essencialmente em *Virtual Personas*, *cookies*, IDs de sessão e redireccionamentos de pedidos HTTP para os portais que foram requeridos. A Figura 3.8 representa as componentes que constituem o Access Server e a relação entre cada uma delas.
A componente *SocketServers* aglomera as componentes que são responsáveis por tratar os pedidos via *socket*. No caso da componente *WebServer* reúne as componentes que tratam os pedidos HTTP. A componente principal é a *CheckServer* que é responsável por verificar os *cookies* e utilizadores, e por fazer o redireccionamento dos pedidos HTTP para os respectivos portais, com a informação necessária para que estes permitam ou neguem o acesso a um determinado utilizador.
A componente *ServerLogout* é responsável por enviar para as WebGates instaladas nos portais, os IDs de sessão que se pretende invalidar. É da responsabilidade da componente *ServerReceiveUID* receber os identificadores dos utilizadores e identificação do dispositivo que está a ser utilizado, por forma a poder ser efectuado a autenticação automática, com base na forma como o utilizador definiu o uso das suas *Virtual Personas* para esse dispositivo.
O Access Server inclui também a componente *ConnectLDAP* que efectua a validação das credenciais dos utilizadores nos serviços de directorias LDAP. Da mesma forma mas para validação das credencias em bases de dados SQL, existe a componente *ConnectSQL*. A componente *ReadParams* é responsável por efectuar a leitura dos parâmetros de configuração do Access Server.
Devido ao facto de haver partilha de parâmetros entre o Access Server e as componentes Intelligent Authentication e WebGate e estes poderem estar visíveis na rede e para os utilizadores

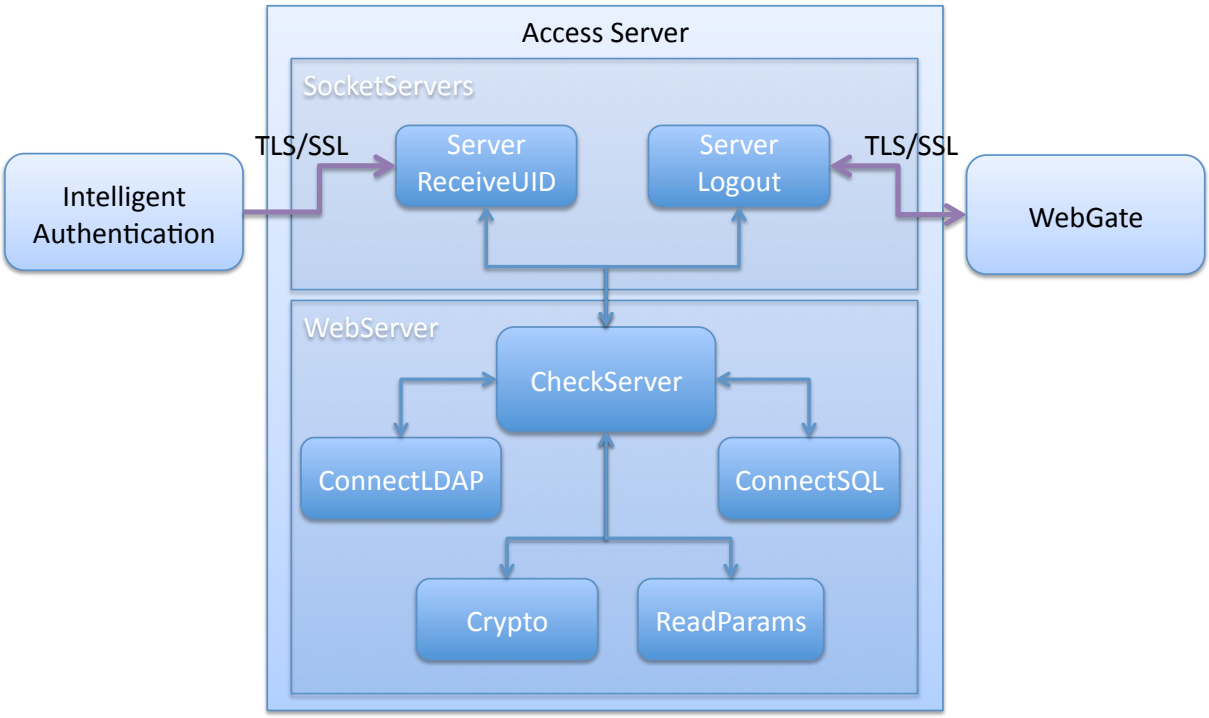

Figura 3.8: Componentes do Access Server

comuns, foi introduzida mais uma componente denominada de *Crypto* que é responsável por cifrar e decifrar todos os parâmetros que são trocados entre as componentes que fazem parte do Terminal Virtual.

A implementação do Access Server foi elaborado seguindo esta separação de componentes e construindo novos diagramas de classes consoante as necessidade. Posto isto, foi criada a classe *CheckServer* (devido às suas interligações será detalhada só depois da apresentação do diagrama de classes final) e a classe *MapSession_PersonaTerminal*. Esta última é a responsável pelo mapeamento entre os IDs de sessão e o par (*persona, terminal*). Esta classe é instanciada na classe *CheckServer*, que a passa para a classe *ServerReceiveUID* de forma a que seja possível a partilha da informação guardada. É de salientar que todos os métodos da classe *MapSession_PersonaTerminal* são *synchronized*, garantindo assim que as instruções são executadas por ordem de chegada, como uma fila.

A classe *ServerReceiveUID* cria um *ServerSocket*, que fica à espera de receber ligações da parte das componentes instaladas nos dispositivos dos utilizadores, as Intelligent Authentication. Todos os *sockets* criados pela classe *ServerReceiveUID* são *sockets* SSL. Para o estabelecimento destas ligações, são necessários certificados digitais tanto do lado do servidor como do lado dos

"clientes". Esta implementação parte do princípio de que estes certificados são previamente trocados entre o Access Server e as componentes Intelligent Authentication e WebGate.
Quando é efectuada uma ligação, a classe *ServerReceiveUID* cria uma nova *Thread* com uma instância da classe *ReceiveUserId*, à qual envia a *MapSession_PersonaTerminal* que recebeu da classe *CheckServer* e fica a aguardar novas ligações. A classe *ReceiveUserId* recebe pelo *socket*, passado pela classe *ServerReceiveUID*, o ID de sessão, a *persona* e o terminal relativo a um pedido de autenticação por parte de um utilizador. Esta informação é guarda na classe *MapSession_PersonaTerminal* para poder ser lida posteriormente pela classe *CheckServer*, originando assim o diagrama de classes representado na Figura 3.9.

Ao longo da implementação das diversas componentes, foram criadas estruturas que registam

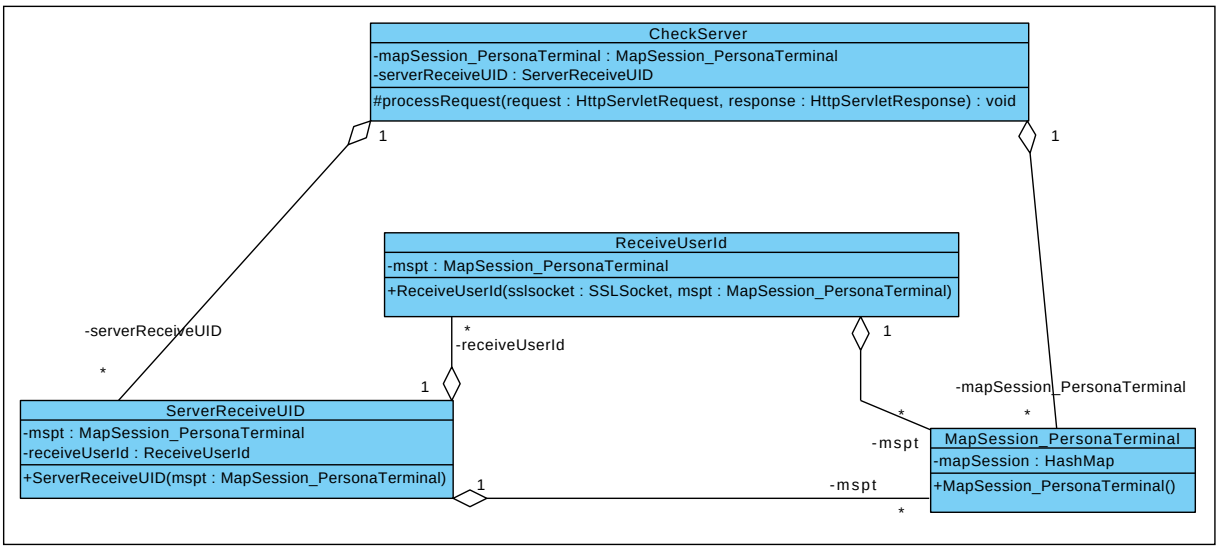

Figura 3.9: Diagrama de Classes do Access Server (Versão 1)

e mapeiam as diversas informações relevantes. Essas estruturas são, na sua maioria, *HashMaps*. Uma *HashMap* é uma estrutura de dados que utiliza uma função de *hash* para associar chaves de pesquisa a valores. A função de *hash* é usada para transformar a chave num índice (o *hash*) de uma matriz de elementos onde o valor correspondente deve ser procurada.
Na classe *CheckServer* existe uma *HashMap* chamada *mapPortalSocketLogout*, que é responsável por mapear a identificação dos portais e o respectivo *socket* pelo qual é enviado as sessões a invalidar. A *HashMap mapPortalSocketLogout* é passada pela classe *CheckServer* à classe *ServerLogout*. Esta classe cria um *ServerSocket* que fica a aguardar ligações das componentes WebGates instaladas nos servidores dos portais. Quando uma ligação é criada, a classe *ServerLogout* recebe a identificação do portal que fez a ligação, guarda essa informação em conjunto

com a respectivo *socket* na *HashMap mapPortalSocketLogout* e aguarda novamente por outras ligações.

Sempre que é necessário efectuar a invalidação de sessões de um determinado portal, a classe *CheckServer* cria uma nova *Thread* com uma instância da classe *SendSessionIdToInvalidate*, à qual passa o *socket* anteriormente guardada na *HashMap mapPortalSocketLogout* e o conjunto de sessões a invalidar. A classe *SendSessionIdToInvalidate* envia pelo *socket* aos respectivos portais, as sessões que se pretende invalidar.

Com a criação destas novas classes, o diagrama de classe representado na Figura 3.9 evoluiu para o diagrama de classes representado na Figura 3.10.

A classe *CheckServer* é também responsável, quando necessário, por efectuar a validação das

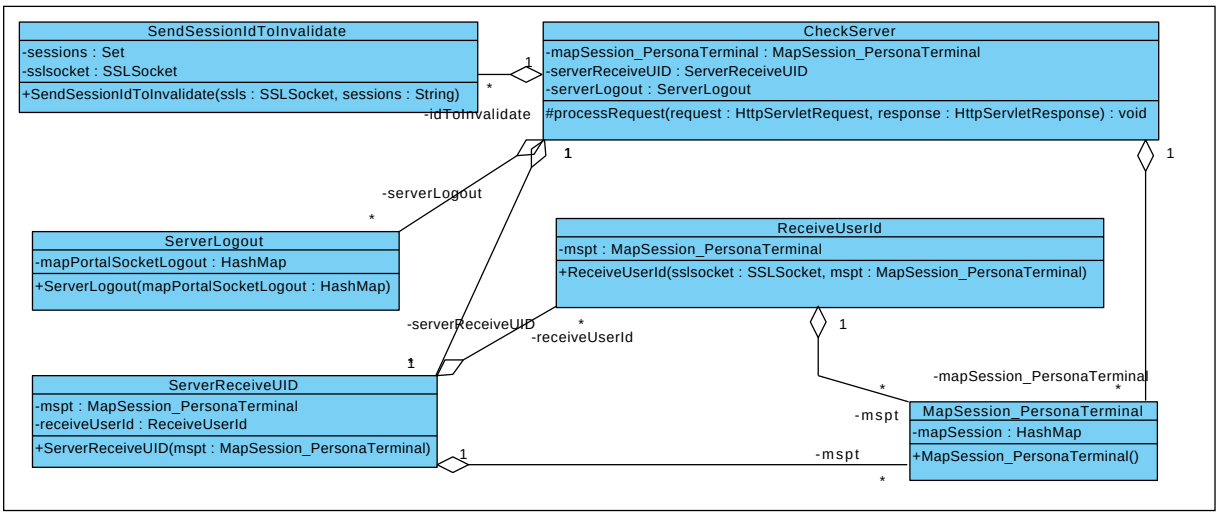

Figura 3.10: Diagrama de Classes do Access Server (Versão 2)

credenciais dos utilizadores (ex: *username* e *password*) nas bases de dados dos portais. Para tal, foram criadas duas classes para aceder às bases de dados (classe ConnectSQL) e ao serviço de directorias LDAP (classe ConnectLDAP), para que a classe *CheckServer* efectue a validação. Com a inclusão destas duas classes, foi elaborado o diagrama de classes que está representado na Figura 3.11.

No Access Server existe um ficheiro de configuração (*configBDMapping.xml*) no qual é possível configurar, entre outros parâmetros, a base de dados para suporte da informação relativa aos portais e o ficheiro que contém a chave secreta que será utilizada para a comunicação entre o Access Server e as restantes componentes. Para efectuar a leitura destes parâmetros por parte da classe *CheckServer*, foi criada a classe *ReadParams*.

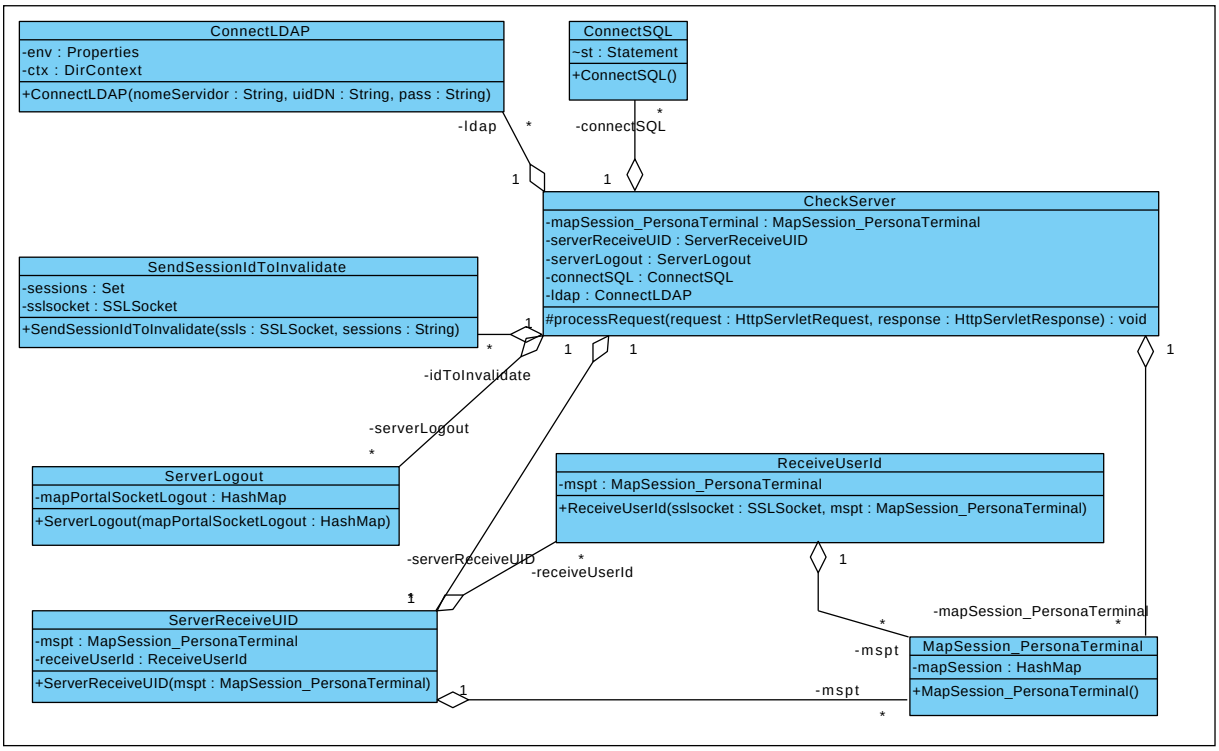

Figura 3.11: Diagrama de Classes do Access Server (Versão 3)

Devido ao facto de alguns parâmetros serem passadas do Access Server para a componente WebGate através do URL [32] , surgiu a necessidade de cifrar esses parâmetros. A classe responsável pela cifra e decifra dessa informação é a classe *Crypto*. Esta classe utiliza o algoritmo AES [33] e a chave secreta definida no ficheiro de configuração *configBDMapping.xml* e que é trocada entre os diversos servidores de tempos a tempos, para cifrar e decifrar os parâmetros trocados entre as várias componentes, dando origem ao diagrama de classes final do Access Server representado na Figura 3.12.

Finalizada a elaboração do diagrama de classes final, é agora apresentada a implementação da classe *CheckServer*. Quando esta é instanciada, são criadas duas *Threads* com um instância da classe *ServerReceiveUId* e outra da classe *ServerLogout*, por forma a que a classe *CheckServer* tenha acesso à informação de qual o utilizador que está a fazer a autenticação e também enviar informação das sessões que se pretende invalidar.

A implementação do Access Server assenta na arquitectura do Terminal Virtual e na forma de comunicação entre o Access Server e as WebGates. Esta comunicação é efectuada à base de *redirects* de pedidos HTTP entre o Access Server e as WebGates. Quando um pedido HTTP é efectuado ao Access Server, através de um *redirect* vindo de uma WebGate, a classe *CheckSer-*

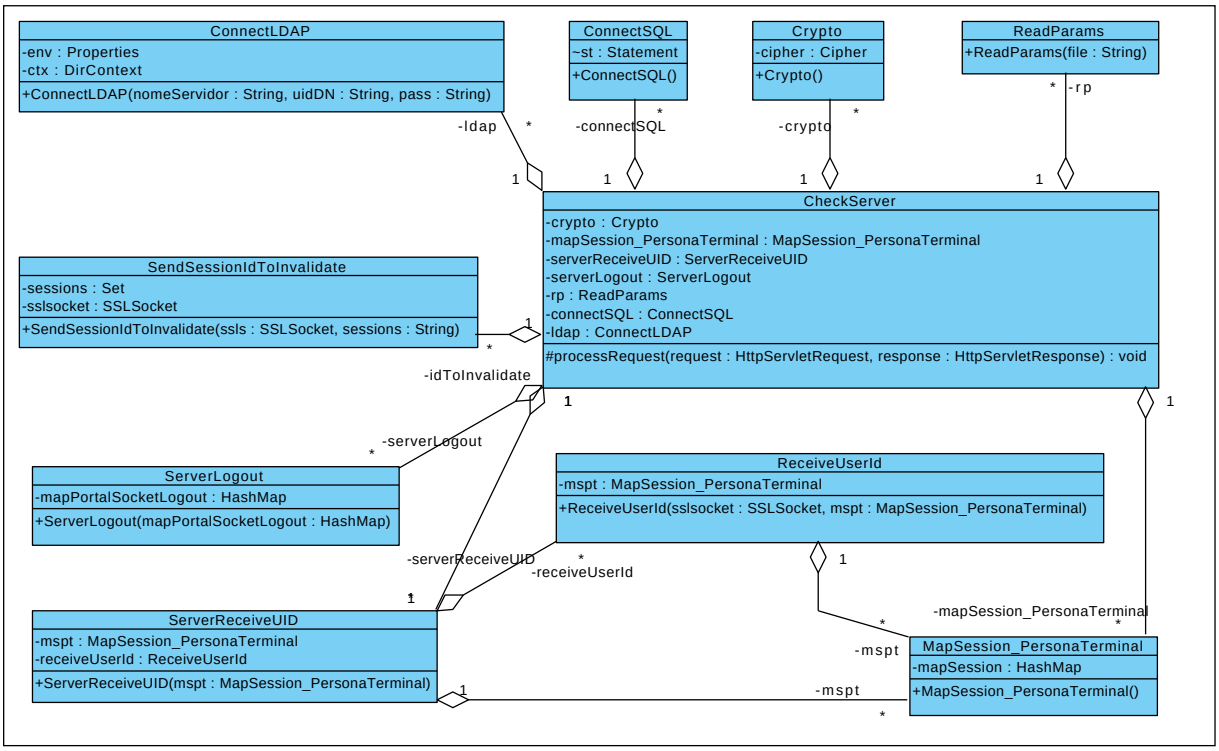

Figura 3.12: Diagrama de Classes do Access Server (Versão Final)

ver verifica se esse pedido contém *cookies*, em particular um *cookie* chamado SSO_PT, e caso contenha verifica se o *cookie* é válido. Faz sempre parte do pedido HTTP vindo das WebGates, um parâmetro chamado *url* que é fundamental para que o Access Server volte a fazer *redirect* para a página que o utilizador pediu.

O *cookie* SSO_PT contém, de forma cifrada, um ID do próprio *cookie* e o ID da sessão activa entre o utilizador e o Access Server. Existe também na classe Access Server uma *HashMap* chamada *mapCookieVP*, que é responsável por guardar o mapeamento entre os *cookies* e a *Virtual Persona* associada. Para que o *cookie* SSO_PT seja válido, é necessário que este esteja na *HashMap mapCookieVP* e, por outro lado, que seja de facto pertencente à sessão existente entre o utilizador e o Access Server.

Uma *Virtual Persona* pode estar autenticada em vários portais e em cada portal pode ter mais do que uma sessão activa e autenticada. Para guardar e gerir esta informação, foi criada uma nova *HashMap* chamada *mapVPurlSession* que para cada *Virtual Persona* guarda os portais autenticados e os respectivos IDs de sessão.

Depois da validação do *cookie* SSO_PT, verifica-se se a *Virtual Persona* já tem alguma sessão autenticada no portal pretendido. Caso se verifique, a nova sessão é adicionada à *HashMap*

mapVPurlSession e é efectuado um *redirect* para a página que o utilizador pediu, com um parâmetro *auth* a indicar o sucesso da autenticação e um parâmetro *claim* com o ID do utilizador e o ID da sessão que fez o pedido de *login*. No caso da *Virtual Persona* não ter ainda nenhuma sessão autenticada nesse portal, é pedido através de um *Web Service* ao GIPTIN as contas mapeadas na *Virtual Persona* respeitantes ao portal requerido.
Se existir alguma conta mapeada, o procedimento é igual ao apresentado anteriormente, ou seja, a nova sessão é adicionada à *HashMap mapVPurlSession* e é efectuado um *redirect* com os parâmetros *auth* e *claim*. Se não existir uma conta mapeada à *Virtual Persona*, é efectuado um *redirect* para a página que o utilizador pediu apenas o parâmetro *auth* a indicar que é necessário pedir ao utilizador que apresente as suas credenciais.
Sempre que a validação do *cookie* SSO_PT não tenha sucesso, é efectuado um *redirect* para a página que o utilizador pediu com o *cookie* SSO_PT com validade igual a zero (para que o *cookie* seja eliminado) e apenas um parâmetro *auth* com a indicação de que a autenticação falhou.
Quando o Access Server recebe um pedido HTTP que não contém o *cookie* SSO_PT mas contém um parâmetro *login*, com a indicação de que se pretende efectuar um acesso a um determinado portal, e com outros dois parâmetros *user* e *pass* (que são as credenciais do utilizador), o Access Server efectua a verificação das credenciais fornecidas nas bases da dados do portal solicitado e no caso destas serem válidas, cria um novo *cookie* SSO_PT com o ID do *cookie* e com o ID de sessão entre o utilizador e o Access Server. Este *cookie* é enviado ao utilizador através de um *redirect* para a página que o utilizador pediu, com um parâmetro *auth* a indicar o sucesso da autenticação e um parâmetro *claim* com o ID do utilizador.
É também pedido através de um *Web Service* ao GIPTIN, a *Virtual Persona* que contém o mapeamento da conta do utilizador no respectivo portal. De seguida é guardado na *HashMap mapCookieVP* o mapeamento do *cookie* com a respectiva *Virtual Persona*. É também guardado na *HashMap mapVPurlSession* a *Virtual Persona*, o portal e respectivo ID de sessão. Nesta fase é introduzida uma nova *HashMap* que guarda as *Virtual Personas* e os respectivos *cookies* activos, para auxiliar a invalidação de sessões, mais concretamente, para efectuar o Single sign-off (SSOff). Esta *HashMap* é denominada de *mapVPsetCookies* e na qual é guardado também a *Virtual Persona* e o *cookie* correspondente.
Se for efectuado um pedido HTTP com um *cookie* válido mas o pedido efectuado contiver um parâmetro *logout*, isto significa que o utilizador pretende fazer *logout* do sistema de autenticação, ou seja, efectuar o SSOff. Posto isto, o Access Server procede à verificação de todos os portais que a *Virtual Persona* está autenticada e para cada um deles, cria uma nova *Thread* com uma instância da classe *SendSessionIdToInvalidate* à qual passa o *socket* que liga o Access Server e

ao portal e as respectivas sessões que se pretende invalidar. De seguida elimina da *HashMap mapCookieVP* todos os *cookies* associados à *Virtual Persona* e elimina das *HashMaps mapVPsetCookies* e *mapVPurlSession* a *Virtual Persona* associada. Por fim, é efectuado um *redirect* para o portal com o parâmetro *auth* com a indicação de que foi efectuado o SSOff.

Sempre que o Access Server recebe um pedido HTTP sem qualquer *cookie* e sem o parâmetro *login*, é verificado se existe na instância da classe *MapSession_PersonaTerminal* denominada de *mapSession_PersonaTerminal*, o ID da sessão criada entre o utilizador e o Access Server. O ID de sessão só aparece na *mapSession_PersonaTerminal* quando a componente Intelligent Authentication (Secção 3.5.2) envia para o *ServerReceiveUID* o respectivo ID de sessão. Se tal não se verificar, é aguardado 500 milissegundos e volta-se a verificar novamente. Este procedimento repete-se até existir o ID de sessão na *mapSession_PersonaTerminal* ou ser atingido um tempo de espera definido.

Quando a existência do ID de sessão se verifica, a *mapSession_PersonaTerminal* fornece também a *persona* e o *terminal* que estão a efectuar o pedido. Com esta informação, é possível aceder, de forma automatizada, à base de dados de suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*, definida na Secção 3.4.2, e extrair a *Virtual Persona* que o utilizador atribuiu a esse *terminal*, nesse dia, a nessa hora. Tendo a *Virtual Persona*, o procedimento a seguir é idêntico ao já descrito anteriormente, ou seja, é pedido através de um *Web Service* ao GIPTIN as contas mapeadas na *Virtual Persona*, se existir alguma conta de portal requerido é criado um novo *cookie* SSO_PT com o ID do *cookie* e com o ID de sessão entre o utilizador e o Access Server. Este *cookie* é enviado ao utilizador através de um *redirect* para a página que o utilizador pediu, com um parâmetro *auth* a indicar o sucesso da autenticação e um parâmetro *claim* com o ID do utilizador.

É também guardado na *HashMap mapCookieVP* o mapeamento do *cookie* com a respectiva *Virtual Persona* e é guardado na *HashMap mapVPurlSession* a *Virtual Persona*, o portal e respectivo ID de sessão. Por fim, é registado na *HashMap mapVPsetCookies* a *Virtual Persona* e o *cookie* correspondente.

Se não existir qualquer ID de sessão na *mapSession_PersonaTerminal* ou não existir uma *Virtual Persona* definido ou, ainda, a *Virtual Persona* não contenha um mapeamento para o portal requerido, é efectuado um *redirect* para a página que o utilizador solicitou apenas um parâmetro *auth* a indicar que é necessário pedir ao utilizador que apresente as suas credenciais.

O Access Server implementa também um *Web Service* que permita à componente Intelligent Authentication verificar o PIN do utilizador e outro que fornece à mesma componente o endereço do Access Server.

3.5.2 Implementação do Intelligent Authentication

O Intelligent Authentication é um serviço (aplicação) que está presente nos dispositivos do utilizador, por forma a que quando seja necessário efectuar a autenticação do utilizador, esta seja feita automaticamente.
A componente Intelligent Authentication baseia-se na captura do tráfego que é originado na comunicação do Intelligent Authentication e o Access Server. Como está representado na Figura 3.13, a componente Intelligent Authentication é constituída por cinco componentes.

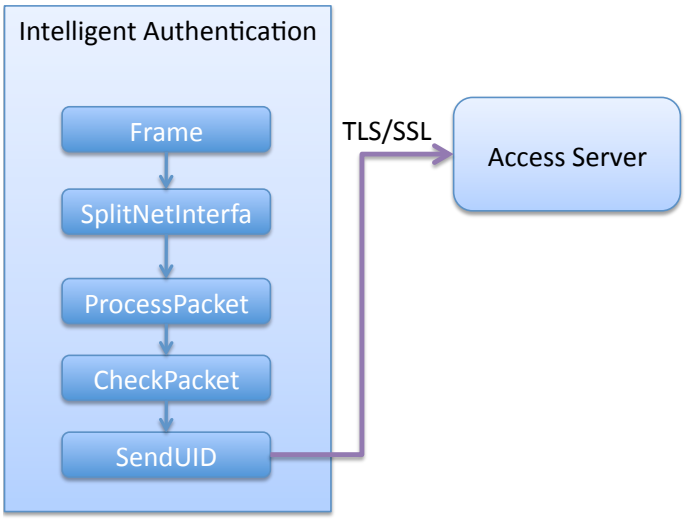

Figura 3.13: Componentes do Intelligent Authentication

A componente *Frame* é a interface gráfica, representada na Figura 3.14, que está disponível para o utilizador e onde este introduz o seu PIN para validação.

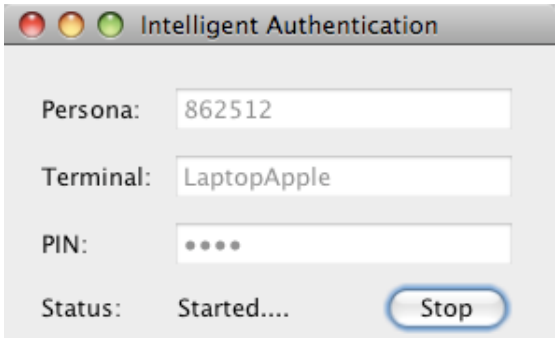

Figura 3.14: Interface do Intelligent Authentication

Sempre que o utilizador pretender fazer uso deste serviço, terá de o iniciar introduzindo o PIN, por forma a que este seja validado e, então, o serviço fica activo. A validação do PIN é efectuada através de um *Web Service* que o Access Server disponibiliza.
É da responsabilidade da componente *SplitNetworkInterface* verificar quais as interfaces de rede activas e por tratar cada uma delas. Para complementar esta componente, existe a componente *CheckPacket* que efectua a verificação dos pacotes e quando intercepta uma comunicação com o Access Server envia esses dados para a componente *ProcessPacket*. Esta componente é responsável por processar os pacotes enviados pela componente *CheckPacket*. Se se tratar de informação relativa à autenticação do utilizador, envia essa informação para a componente *SendUID*. Esta componente é responsável por enviar os dados necessários para o Access Server para este proceder à autenticação do utilizador. Tendo por base as componentes definidas, foi elaborado o diagrama de classes representado na Figura 3.15, que representa a estrutura e a relação entre as classes que fazem parte da implementação do Intelligent Authentication.

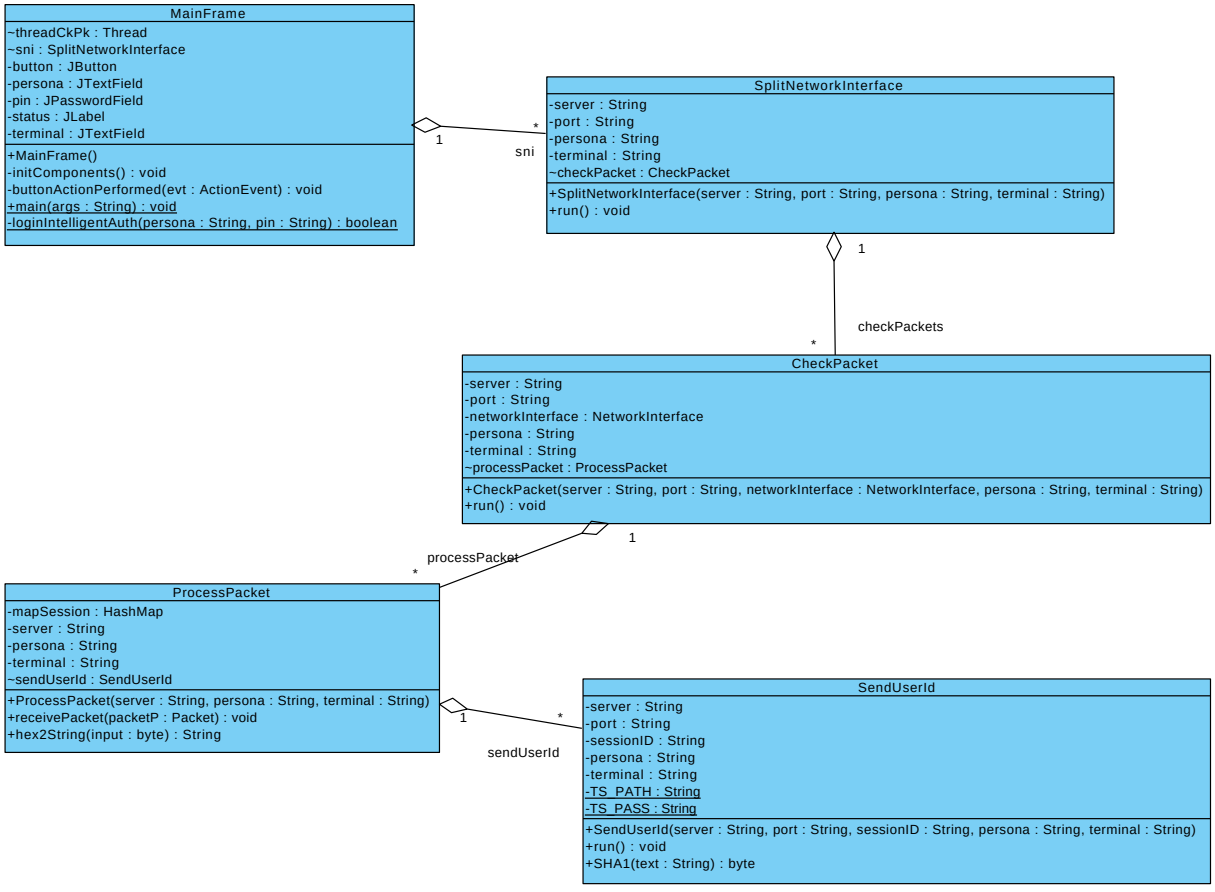

Figura 3.15: Diagrama de Classes do Intelligent Authentication

Quando o Intelligent Authentication é activado, é criada uma *Thread* com uma instância da classe *SplitNetworkInterface*. Esta classe é responsável por criar para cada interface de rede, uma nova *Thread*. As novas *Threads* criadas são instâncias da classe *CheckPacket*. Esta classe, fazendo uso da API da biblioteca *Jpcap*, tem a responsabilidade de criar um filtro composto pelo endereço IP e porta do Access Server, e por iniciar a captura de pacotes. A informação relativa ao Access Server (IP e porta) é dada por um *Web Service* que o próprio Access Server disponibiliza. A classe *CheckPacket* depois de definir o filtro, invoca o método *loopPacket* (disponibilizado pela API da biblioteca *Jpcap*) com uma instância da classe *ProcessPacket*. O método *loopPacket* fica a aguardar a captura de um pacote válido, com base no filtro definido, e quando tal acontece, processa o pacote com invocando a instância da classe que lhe foi passada e fica de novo a aguardar novas capturas.
A classe *ProcessPacket* verifica se o pacote capturado é válido e processa-o, caso contrário, simplesmente descarta-o. É esta classe que efectua a recolha do ID da sessão criada entre o Intelligent Authentication e o Access Server. Quando tal suceder, a classe *ProcessPacket* cria uma nova *Thread* com uma instância da classe *SendUserId*. A classe *SendUserId* liga-se através de um *socket* ao Access Server, mais concretamente à instância da classe *ServerReceiveUID*, a fim de lhe enviar o ID de sessão, a *persona* e *terminal* que estão a efectuar o pedido de autenticação. Desta forma, e se todas as condições se verificarem do lado do Access Server, o utilizador não necessita de introduzir as suas credenciais para se autenticar, pois a autenticação é efectuada automaticamente. Dependo da forma como o utilizador definiu as suas *Virtual Personas*, será usada a identidade por ele definida.
Para efectuar as capturas de tráfego, foi utilizado uma biblioteca Java de captura de pacotes denominada de *Jpcap*.

3.5.3 Implementação da WebGate

Para que a integração da arquitectura definida na Secção 3.2 com os portais *web* já existentes fosse possível, tal implicaria algumas mudanças de fundo nesses portais. De modo a simplificar a integração, foi implementada a componente WebGate para que as alterações a efectuar nos portais fossem as mínimas indispensáveis. Assim, nos portais apenas é necessário introduzir a componente WebGate e deixar que esta seja a responsável por permitir ou negar o acesso ao portal. A WebGate é uma componente que fica à entrada de cada portal a filtrar todos os pedidos efectuados a esse portal. Consequentemente, passa a ser responsável por verificar as credenciais dos utilizadores e gerir as sessões autenticadas entre o utilizador e o portal.

A WebGate foca-se essencialmente nas sessões HTTP e em *redirects* para o Access Server. Com base nisto, a WebGate é composta por cinco componentes. A Figura 3.16 representa as componentes que fazem parte da WebGate.

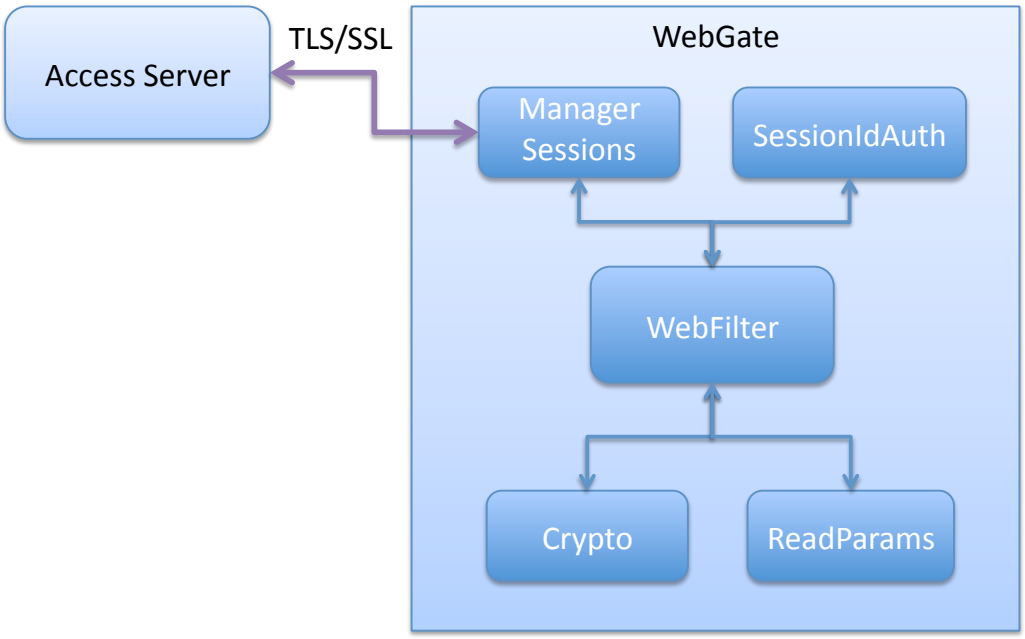

Figura 3.16: Componentes da WebGate

A componente principal denominada por *WebFilter* é responsável por fazer a triagem de todos os pedidos direccionados ao portal onde a esta está implementada, de forma a avaliar se permite ou nega o acesso ao portal. A componente *SessionIdAuth* é responsável por guardar todas as sessões autenticadas entre os utilizadores e o portal. É da responsabilidade da componente *ManagerSessions* gerir as sessões autenticadas. Esta componente está directamente ligada ao Access Server por forma a que quando for necessário invalidar uma sessão, a componente *ManagerSessions* receba a informação necessária e efectue a invalidação. Do mesmo modo que no Access Server, a WebGate inclui uma componente *ReadParams*, que é responsável pela leitura e configuração dos parâmetros necessários para a WebGate poder operar e, por fim, a componente *Crypto*, que tem a responsabilidade de cifrar e decifrar os parâmetros que são trocados entre a WebGate e o Access Server.
A implementação da WebGate seguiu a forma como as componentes foram separadas, dando origem ao diagrama de classe representado na Figura 3.17.

Uma vez introduzida nos servidores dos portais, a WebGate instância a classe *WebFilter*.

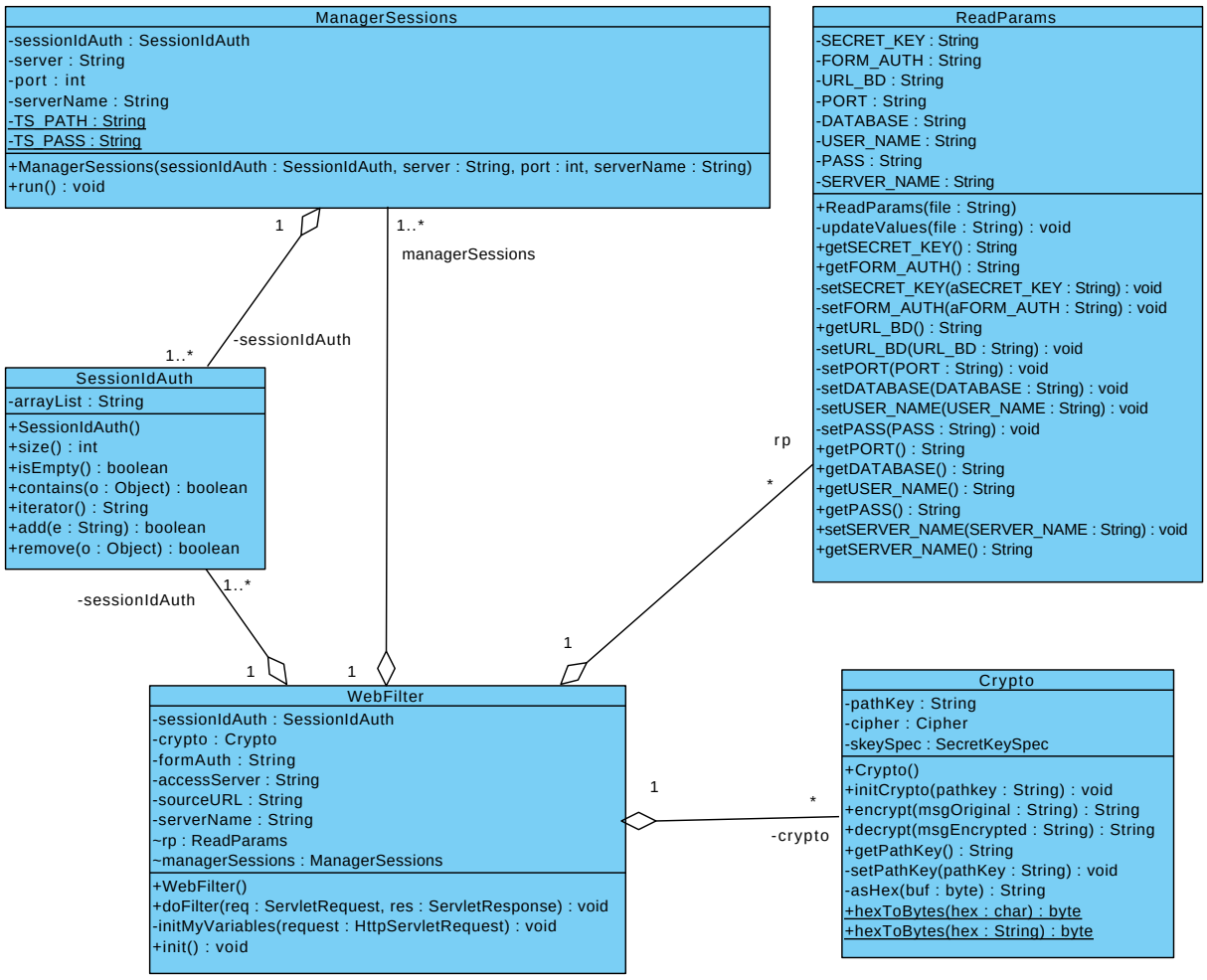

Figura 3.17: Diagrama de Classes da WebGate

Esta classe começa por configurar os parâmetros definidos no ficheiro *configBDServer.xml* (ex: endereço do portal, chave secreta, entre outros), com a ajuda da classe *ReadParams*. De seguida cria uma *Thread* com uma instância da classe *ManagerSessions* e fica a aguardar por pedidos HTTP direccionados ao portal onde esta está introduzida. A classe *SessionIdAuth* é responsável por guardar a lista de todas as sessões autenticadas no portal. Para que a informação referente às sessões autenticadas seja partilhada entre a classe *WebFilter* e a classe *ManagerSessions*, a classe *SessionIdAuth* é instanciada na classe *WebFilter* que a passa como parâmetro à classe *ManagerSessions* e todos os seus métodos são *synchronized*.

A classe *ManagerSessions* é responsável pela gestão das sessões autenticadas no portal. Mais especificamente, é responsável por invalidar sessões. Para tal, a classe *ManagerSessions* liga-se através de um *socket* SSL ao Access Server, em particular à instância da classe *ServerLogout*.

Depois da ligação ser efectuada com sucesso, envia ao Access Server o nome do portal e fica a aguarda que sejam enviado pelo mesmo canal sessões para invalidar. Quando recebe uma sessão, a classe *ServerLogout* invalida a sessão e fica de novo à espera de novas sessões para invalidar. Para invalidar uma sessão que foi enviada pelo Access Server, basta retira-la da lista da classe *SessionIdAuth*.

Quando a WebGate recebe um pedido HTTP, a classe *WebFilter* verifica se a sessão está autenticada. Para isso verifica se a sessão está contida na instância da classe *SessionIdAuth*. Se tal não suceder, é verificado se se trata de um pedido de *login*. Caso seja, são lidos os parâmetros *username* e *password* e se estes forem válidos, é efectuado um *redirect* para o Access Server com os parâmetros *url* que indica qual é a página pretendida, *login* que indica tratar-se da tentativa de autenticação, *user* e *pass* com o *username* e *password* respectivamente e *session* com o ID da sessão estabelecida entre com a WebGate. Devido ao facto destes parâmetro estarem no URL, foi necessário a criação da classe *Crypto* para cifrar e decifrar todos os parâmetros trocados entre a WebGate e o Access Server.

No caso da sessão não estar autenticada e do parâmetro *login* não estar definido, é verificado se existe um outro parâmetro que é enviado pelo Access Server, o parâmetro *auth*. Se este parâmetro existir e contiver informação que indica o sucesso da autenticação, contém também o ID do utilizador e ID da sessão que se pretende autenticar. Se o ID de sessão recebido for igual ao ID de sessão do pedido actual, é enviado para o portal o ID do utilizador e permitido assim o acesso ao portal. Se o parâmetro *auth* contiver a indicação de que a autenticação não teve sucesso, é imediatamente solicitado ao utilizador que faça a autenticação. Dependendo da forma como cada portal pretender fazer a autenticação, esta pode ser efectuada com *username* e *password*, *CardSpace*, certificados, entre outras.

Sempre que forem recebidos pedidos HTTP sem sessão autenticada e sem qualquer parâmetro ou com parâmetros inválidos, é efectuado um *redirect* para o Access Server com os parâmetros *url* e *session* apenas.

Quando é efectuado um pedido HTTP e este tem uma sessão autenticada, ou seja, o ID de sessão faz parte da lista de sessão autenticadas definido na classe *SessionIdAuth*, é verificado se este pedido contém um parâmetro *logout*. Se tal não se verificar, é dado acesso ao utilizador para entrar no portal. Se o parâmetro *logout* for válido, a sessão actual é invalidada e é efectuado um *redirect* para o Access Server com os parâmetros *url* e *logout* que indica que o utilizador pretende abandonar o portal.

3.5.4 Implementação do MultiManager

De forma a oferecer ao utilizador uma visão global dos seus dispositivos e das suas *Virtual Personas*, foi implementada a componente MultiManager, que também permite a gestão dos mesmos dispositivos e *Virtual Personas*. Esta componente foi implementada sob a forma de uma interface *web*, de modo a que o utilizador possa efectuar a gestão dos dispositivos e das *Virtual Personas* em qualquer dispositivo, em qualquer lugar e a qualquer hora.

A implementação desta componente, representada na Figura 3.18, divide-se em duas compo-

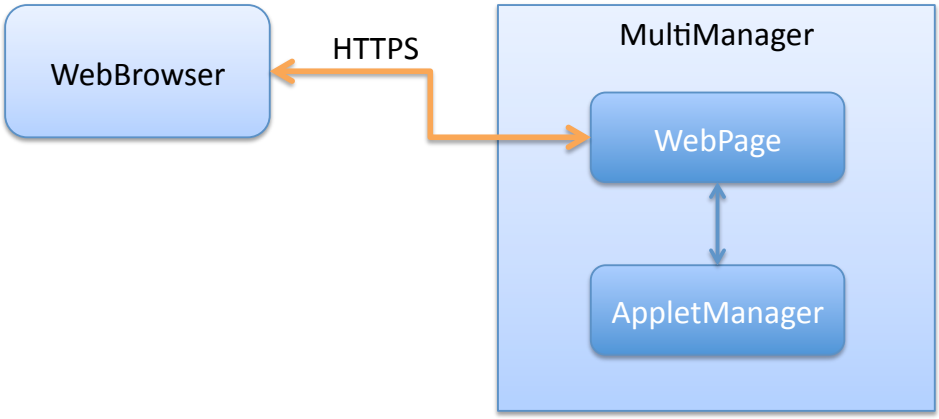

Figura 3.18: Componentes do MultiManager

nente, a *WebPages* que contém as páginas JSP criadas e a *AppletManager* que contém uma *applet* que foi criada para auxiliar a gestão dos dispositivos e das *Virtual Personas*.

A componente *WebPages* é constituída por uma página *default* do portal e por uma outra página que tem embebido a componente *AppletManager* que inclui a *applet* para a gestão. A interface *web* para a gestão está representada na Figura 3.19. A *applet* embebida nesta interface, inclui um sistema de *login* para os utilizadores se poderem autenticar e, posteriormente, gerir os seus dispositivo e *Virtual Personas*.

A gestão é disponibilizada ao utilizador, através de uma interface onde estão listados os seus dispositivos, as suas *Virtual Personas* e os mapeamentos de cada uma das *Virtual Personas*.

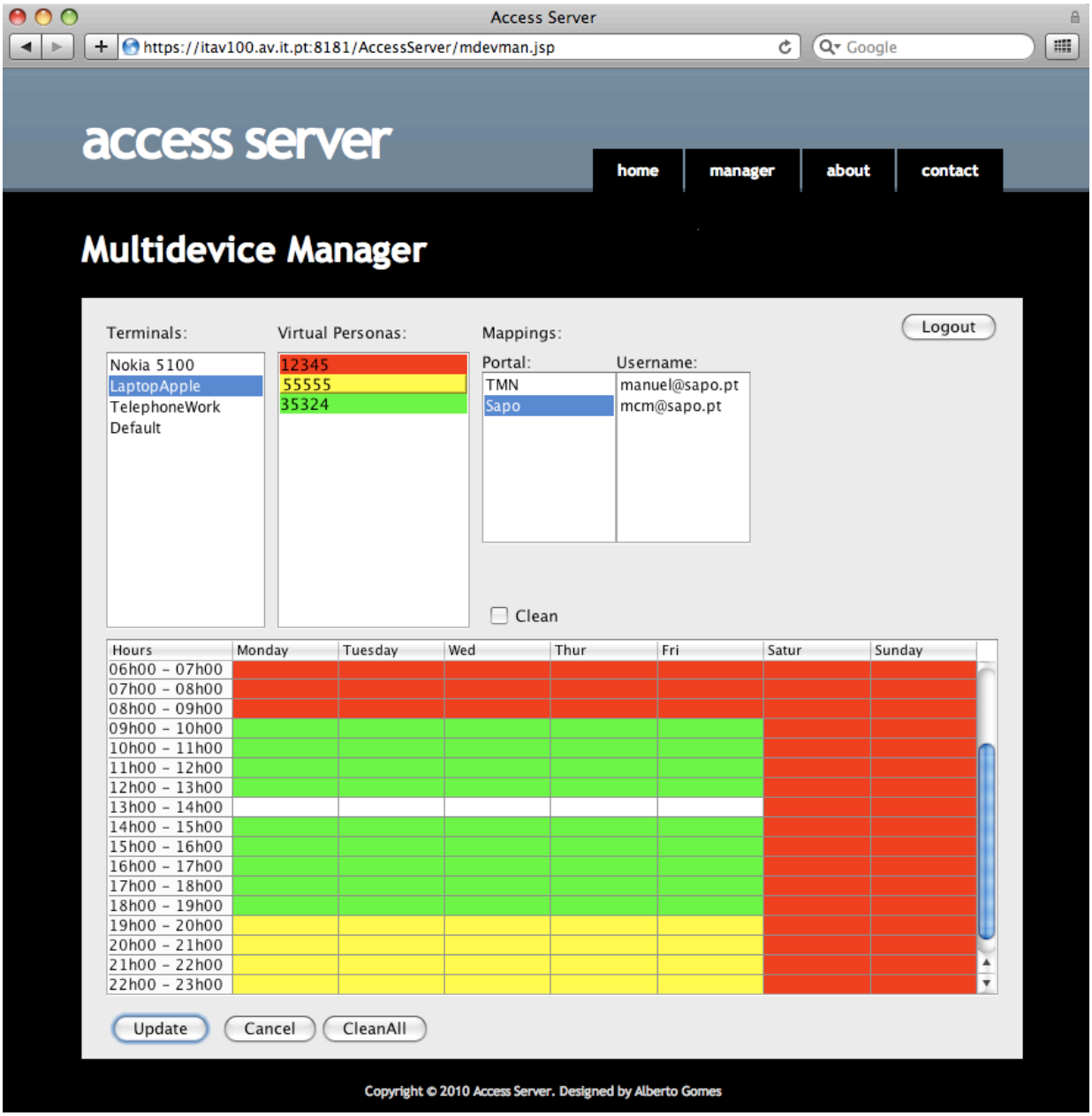

Figura 3.19: Interface *web* do gestor de dispositivos e *Virtual Personas*

Está também disponível nesta *applet*, uma tabela em formato de horário, onde estão representadas através de diferentes cores, as *Virtual Personas* associadas a cada dispositivo. Através da selecção de um dispositivo na lista de dispositivos e da selecção de uma *Virtual Persona* é possível seleccionar na tabela quais os horários em que essa *Virtual Persona* está associada ao dispositivo seleccionado.

Através da componente MultiManager, é disponibilizado ao utilizador uma forma simples de visualizar todos os seus dispositivos e *Virtual Personas*, por forma a possibilitar a gestão de cada dispositivo e respectivas *Virtual Personas* associadas.

3.6 Sumário

Neste capítulo foi apresentado a arquitectura do Terminal Virtual, onde foram especificadas as etapas desde a primeira abordagem até à elaboração da versão final da arquitectura do Terminal Virtual proposta. Apresentou-se também neste capítulo a integração do gestor de identidades da PT Inovação com a arquitectura do Terminal Virtual. Para efectuar esta integração, foi necessário a criação de uma nova estrutura, por forma a poder usar as informações disponibilizadas pelo GIPTIN mais eficazmente. Para a implementação da arquitectura do Terminal Virtual, foram também definidos neste capítulo os modelos de dados para suporte das informações relativas aos portais e relativas aos dispositivos e *Virtual Personas* pertencentes ao utilizador. Finalmente neste capítulo foi apresentada a implementação da arquitectura do Terminal Virtual, usando para isso diagramas de componentes, diagramas de classes e a apresentação de algumas interfaces.

Capítulo 4

Cenários de utilização

4.1 Introdução

Com a implementação da arquitectura do Terminal Virtual, os mecanismos de autenticação dos utilizadores nos portais são agora diferentes. De forma a mostrar os novos mecanismos usados, são apresentados neste capítulo alguns cenários de utilização muito próximos dos cenários reais, que vão desde a "simples"autenticação num portal até à invalidação de todas as sessões autenticadas de um determinado utilizador nos respectivos portais (SSOff). São também apresentados cenários de autenticação entre diferentes portais dentro do mesmo dispositivo (SSO) e até mesmo em dispositivos diferentes. Para auxiliar a representação da comunicação efectuada entre as diversas componentes que compõem o Terminal Virtual, são usados diagramas de sequência. Para os cenários especificados, os utilizadores efectuam pedidos aos portais Portal A e Portal B através de um *WebBrowser* instalado nos dispositivos. Sempre que for necessário fornecer as credenciais do utilizador, estas serão enviadas por uma *form* pertencente a cada portal e sob a forma de *username* e *password*. É necessário salientar também que todos os parâmetros trocados entre as diferentes componente do Terminal Virtual e os *cookies* estão cifrados, no entanto na descrição dos cenários tal não é referido.

Neste capítulo são apresentados quatro cenários de utilização. No Cenário I o utilizador efectua a autenticação apenas num portal. No Cenário II o utilizador efectua a autenticação em dois portais distintos usando para isso o mecanismo de SSO. O Cenário III apresenta a autenticação de um utilizador no mesmo portal mas a partir de dispositivos diferentes. No último cenário, o Cenário IV, o utilizador efectua *logout* em todos os portais, isto é, efectua o SSOff.

4.2 Cenário I - Autenticação num portal

O cenário aqui apresentado é um cenário simples mas que servirá de base aos cenários seguintes. Este cenário refere-se à autenticação de um utilizador num portal, como mostra a Figura 4.1.

Neste cenário, o utilizador não tem *a priori* qualquer sessão autenticada no portal, nem qualquer

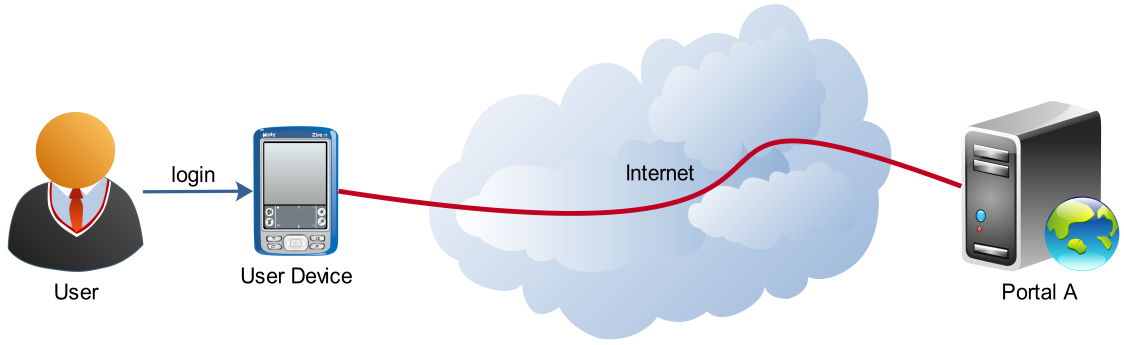

Figura 4.1: Cenário I

cookie de SSO. Pelo facto de ser possível um utilizador usar um dispositivo que não contém a componente Intelligent Authentication, essa componente também não faz parte deste cenário.

Para descrever a interação entre as várias componentes que fazem parte do Terminal Virtual e o dispositivo do utilizador, foram usados diagramas de sequência UML (Unified Modeling Language) [34]. Pelo facto destes serem extensos, serão apresentados por partes para que o seu entendimento seja mais efectivo. No entanto, a modelação completa encontra-se no Apêndice A.

O utilizador começa por efectuar um pedido de uma página através do *WebBrowser* ao Portal A, que contém a componente WebGate. Como a WebGate é responsável por verificar todos os pedidos, esta recebe o pedido e verifica se já existe alguma sessão autenticada entre o dispositivo do utilizador e o Portal A. Visto que não existe qualquer sessão autenticada, a WebGate verifica se se trata de um pedido de autenticação, ou seja, se o pedido contém um parâmetro *login* com a indicação de que se pretende efectuar um autenticação. Como tal não se verifica, a WebGate verifica a existência do parâmetro *auth* no pedido. Mais uma vez a verificação não sucede, o que faz com que a WebGate efectue um *redirect* para o Access Server. Este pedido contém os parâmetros *url* e *session* que são, respectivamente, o URL pedido e o ID da sessão entre o dispositivo do utilizador e o Portal A, como mostra o diagrama de sequência na Figura 4.2.

Quando o pedido chega ao Access Server, este começa por verificar se o pedido contém o *cookie* SSO_PT. Como o pedido não inclui o *cookie* pretendido, o Access Server passa a verificar

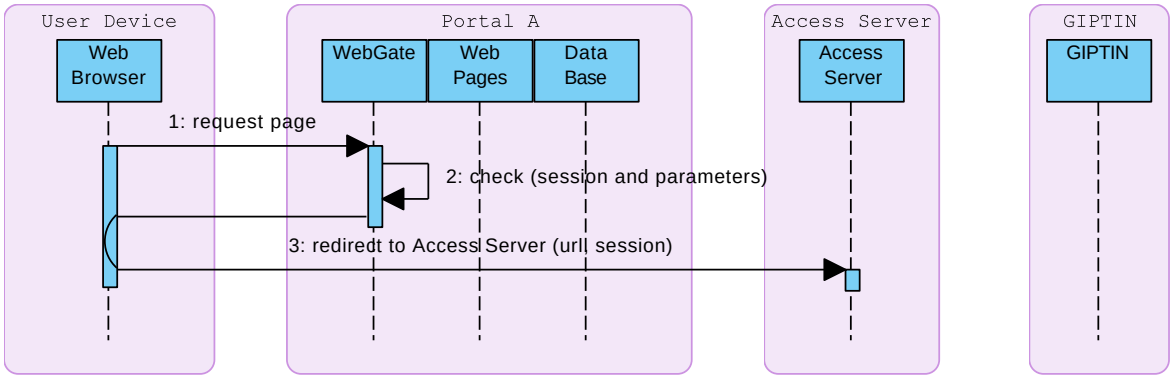

Figura 4.2: Diagrama de Sequência da autenticação (Parte 1)

os parâmetros presentes no pedido, em particular o parâmetro *login*. Visto que este parâmetro não está também contido no pedido, o Access Server efectua a última verificação, procurando na *HashMap mapSession_PersonaTerminal* se existe o ID da sessão entre o dispositivo do utilizador e o Access Server, ou seja, se a componente Intelligent Authentication enviou para o Access Server os dados necessários à identificação do utilizador. Esta procura é repetida de 500 em 500 milissegundos até ter sucesso ou ser atingido um tempo limite, definido previamente nas configurações do Access Server ou calculado durante o processo utilizando para o efeito o tempo de ida e volta de um pacote (RTT) entre o dispositivo do utilizador e o Access Server. Como o dispositivo que o utilizador está a usar não contém a componente Intelligent Authentication, a procura vai falhar e é efectuado, pelo Access Server, um novo *redirect* para a página definida no parâmetro *url*. Neste pedido é enviado um parâmetro *auth* com a indicação de que é necessário pedir as credenciais ao utilizador, representado no diagrama de sequência da Figura 4.3.

Com o pedido de volta ao Portal A, é efectuado pela WebGate todas as verificações descritas

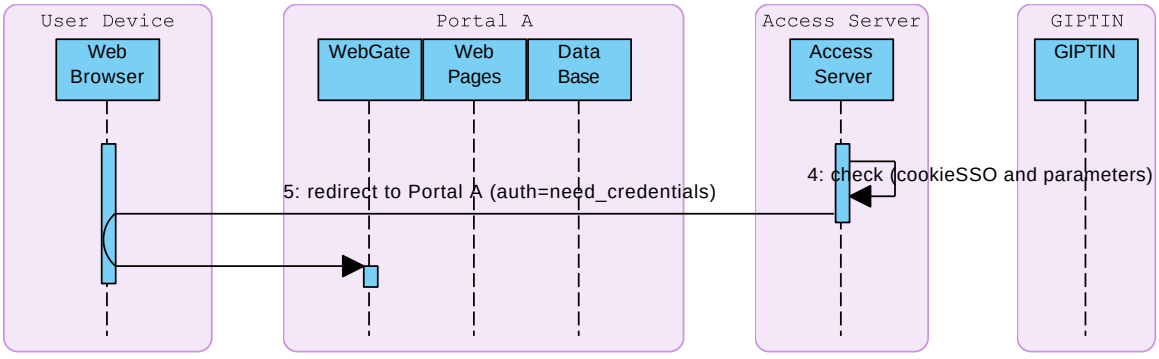

Figura 4.3: Diagrama de Sequência da autenticação (Parte 2)

anteriormente. Mesmo não tendo a sessão autenticada, este pedido que chega novamente ao Portal A já contém o parâmetro *auth*. Este parâmetro possui informação que indica à WebGate que é necessário o utilizador apresentar as suas credenciais. Então a WebGate apresenta ao utilizador a *form* onde este deve inserir as suas credenciais. Depois do utilizador inserir as respectivas credenciais, envias para o Portal A. A *form* de autenticação está desenvolvida de forma a que quando seja enviado as credenciais do utilizador para o respectivo portal, envie também o parâmetro *login* com a indicação de que se pretende efectuar uma autenticação. De novo chega à WebGate um pedido com uma sessão inválida que é verificado mas desta vez este pedido inclui o parâmetro *login*. Posto isto, a WebGate efectua novamente um *redirect* para o Access Server com os parâmetros *url* e *session*, já descritos anteriormente, e ainda com os parâmetros *user*, *pass* e *login*, que são, respectivamente, *username* e *password* enviados pelo utilizador e a indicação de que se pretende efectuar uma autenticação. A Figura 4.4 representa estas iterações.

Uma vez mais o pedido chega ao Access Server sem o *cookie* SSO_PT mas agora trás consigo o

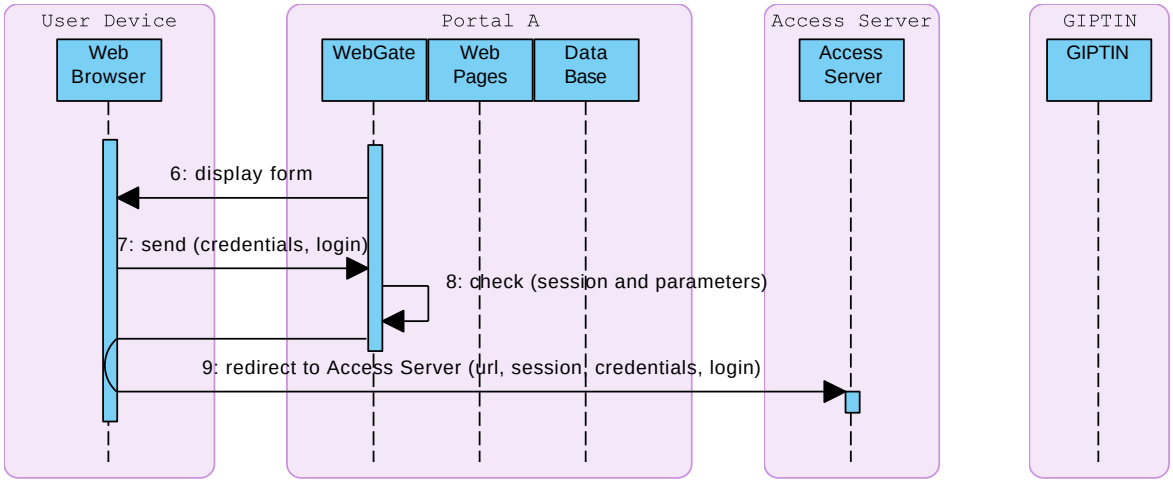

Figura 4.4: Diagrama de Sequência da autenticação (Parte 3)

parâmetro *login* e os restantes parâmetros. Desta forma o Access Server procede à validação das credenciais do utilizador na base de dados do portal requerido. Com o sucesso da validação das credenciais, o Access Server cria um *cookie* SSO_PT com o ID do próprio *cookie* e com o ID da sessão entre o dispositivo do utilizador e o Access Server. Dada a importância deste *cookie*, a inclusão do ID de sessão prende-se com o facto da possibilidade, ainda que muito remota, deste poder ser extorquido na rede e poder ser usado por outro utilizador. Desta forma, tal utilização é impossibilitada, porque a verificação do *cookie* SSO_PT por parte do Access Server inclui a verificação do ID de sessão. Depois da criação do *cookie* SSO_PT o Access Server pede também

ao GIPTIN a *Virtual Persona* que contém o mapeamento do *username* com o respectivo portal. Com estes dados, o Access Server regista nas suas estruturas internas o *cookie* SSO_PT e a *Virtual Persona* associada e regista também o portal e respectivo ID de sessão associado à *Virtual Persona* utilizada. De seguida é efectuado novo *redirect* para a página definida no parâmetro *url* com o parâmetro *auth* que contém a indicação do sucesso da autenticação e o ID da sessão criada entre o dispositivo do utilizador e o Portal A, o parâmetro *claim* com o ID do utilizador e com o *cookie* SSO_PT que será guardado no *WebBrowser* do dispositivo do utilizador, como é apresentado pelo diagrama de sequência da Figura 4.5.

Ainda com a sessão inválida, o pedido chega agora ao Portal A com o parâmetro *auth*. Desta

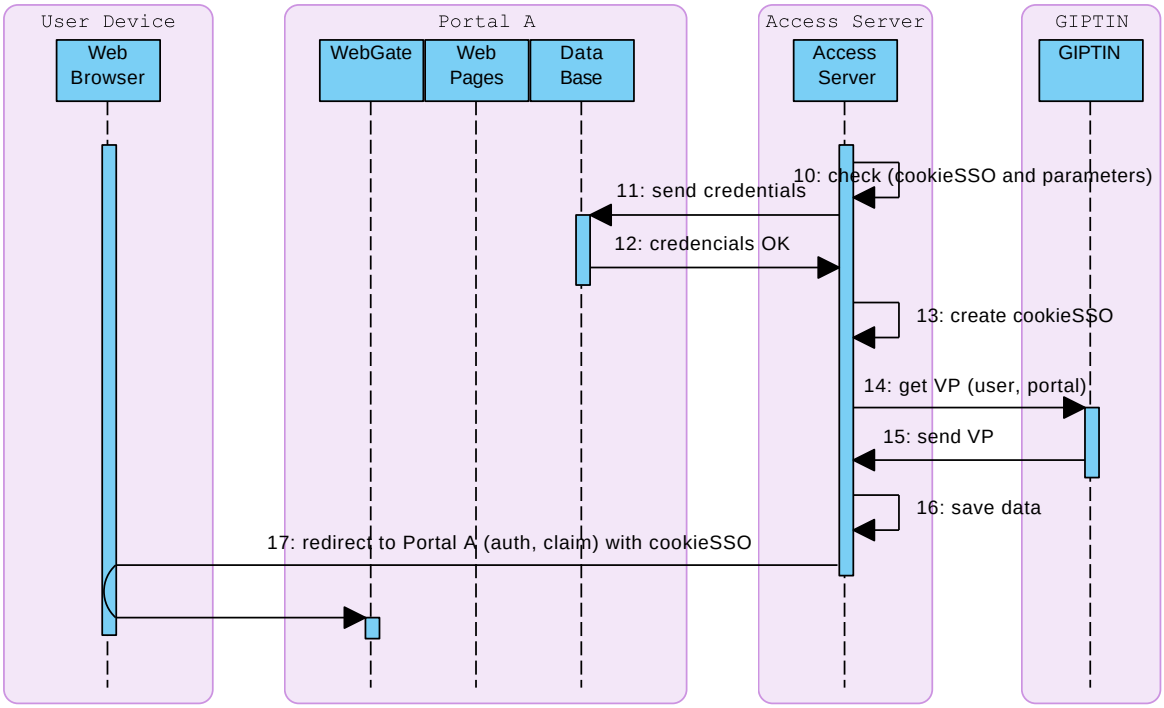

Figura 4.5: Diagrama de Sequência da autenticação (Parte 4)

vez, este parâmetro tem a indicação de que a autenticação do utilizador está concluída com sucesso no Access Server e tem também o ID de sessão. Se o ID de sessão incluído no parâmetro *auth* coincidir com o ID da sessão entre o dispositivo do utilizador e o Portal A, a autenticação é concluída com sucesso também pela WebGate que só agora autentica a sessão. De seguida envia para a página solicitada do Portal A o ID do utilizador por forma a que esta possa identificar quem é o utilizador e posteriormente apresentar a página ao utilizador. A Figura 4.6 representa o diagrama de sequência das últimas iterações, para apresentar a página solicitada ao utilizador.

Desde o momento em que a sessão está autenticada, todos os pedidos efectuados ao Portal A

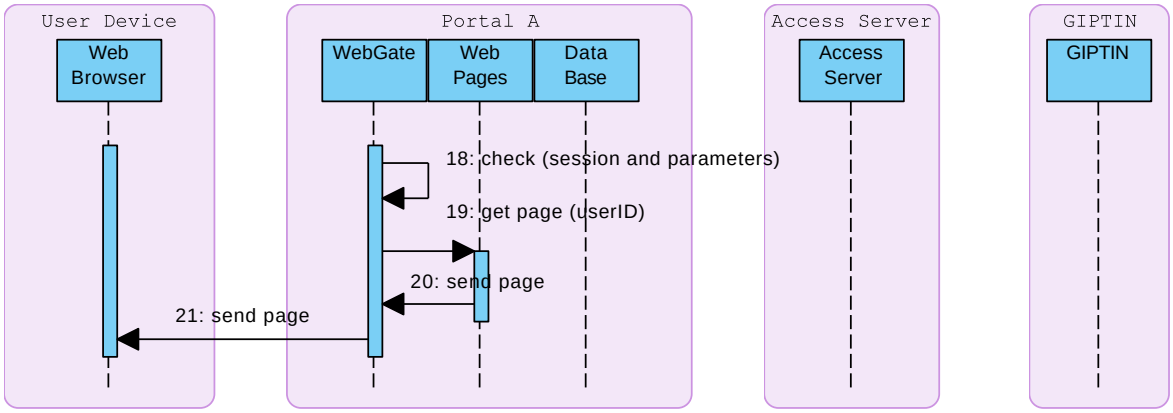

Figura 4.6: Diagrama de Sequência da autenticação (Parte 5)

continuam a passar pela WebGate mas passam agora directamente para as páginas do portal.

4.3 Cenário II - Autenticação em dois portais (SSO)

O cenário apresentado nesta secção representa a autenticação em dois portais distintos, o Portal A e o Portal B, utilizando o mesmo dispositivo do utilizador (Figura 4.7). De igual forma que o Cenário I, o dispositivo do utilizador não tem instalado a componente Intelligent Authentication.

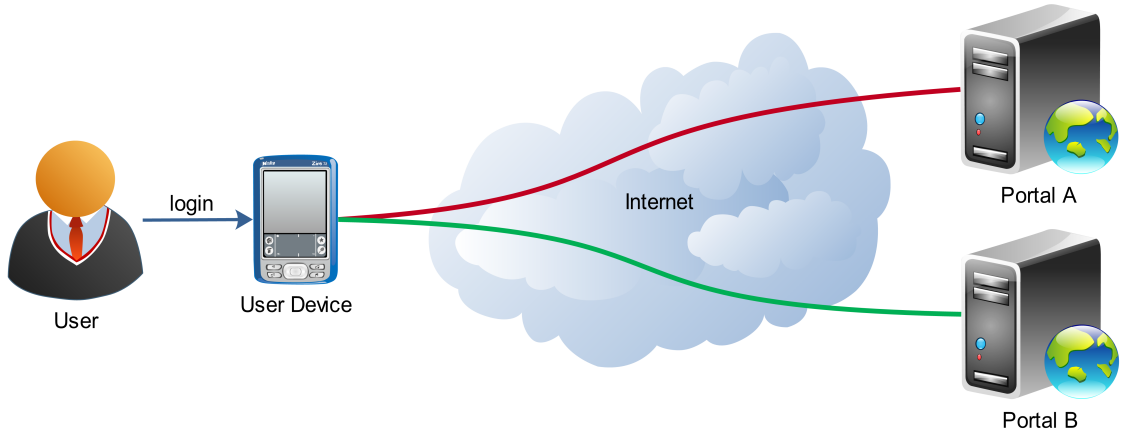

Figura 4.7: Cenário II

Partindo da autenticação no Portal A como sendo a primeira a ser efectuada, esta é executada

exactamente da forma que foi apresentada no Cenário I, o que leva a que o utilizador já tenha uma sessão autenticada no Portal A e que o *WebBrowser* contido na seu dispositivo tenha também um *cookie* SSO_PT. Posto isto, a descrição deste cenário apenas apresenta as comunicação efectuadas depois do utilizador efectuar a autenticação no Portal A e lhe ser apresentada, pelo menos, uma página desse mesmo portal.

Depois da autenticação no Portal A e da recepção da página solicitada, o utilizador efectua agora um pedido de uma página ao Portal B (que também contém a componente WebGate) através do mesmo dispositivo e do mesmo *WebBrowser*.

Quando a WebGate do Portal B recebe o pedido verifica se já existe alguma sessão autenticada entre o dispositivo do utilizador e o Portal B. Como o utilizador apenas se autenticou no Portal A, não existe ainda no Portal B qualquer sessão autenticada, o que faz com que a WebGate verifique se o pedido contém um parâmetro *login* com a indicação de que se pretende efectuar um autenticação no Portal B. Visto que o parâmetro *login* não existe, a WebGate volta a verificar os parâmetros à procura do parâmetro *auth*. Como o parâmetro *auth* não faz parte do pedido, a WebGate efectua um *redirect* para o Access Server com os parâmetros *url* que contém o URL pedido e *session* que contém o ID da sessão entre o dispositivo do utilizador e o Portal B, tal como mostra o diagrama de sequência apresentado na Figura 4.8.

Como o *redirect* efectuado pela WebGate passa pelo *WebBrowser* do dispositivo do utilizador

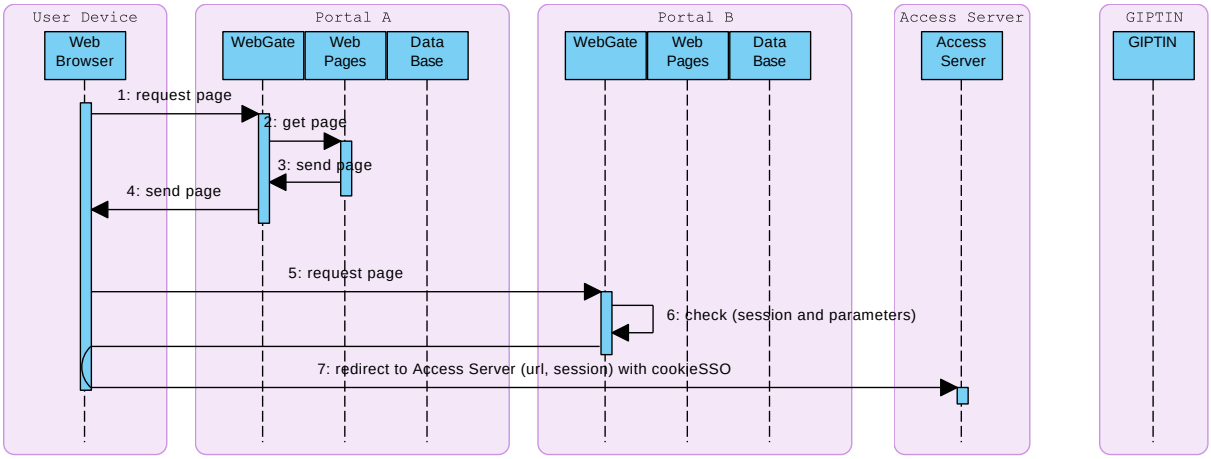

Figura 4.8: Diagrama de Sequência da autenticação em dois portais (Parte 1)

que efectuou a autenticação no Portal A, é enviado juntamente com o pedido o *cookie* SSO_PT. Este detalhe faz com quando o Access Server verificar a existência do *cookie* SSO_PT, este já exista. Assim, o Access Server verifica agora se o *cookie* é válido, isto é, se o *cookie* SSO_PT ainda faz parte da lista de *cookies* activos. O Access Server faz ainda uma verificação ao *cookie*

SSO_PT, desta vez para verificar se o *cookie* pertence à sessão estabelecida entre o dispositivo do utilizador e o Access Server. A cada *cookie* está associado uma *Virtual Persona* e a cada *Virtual Persona* está associada um lista com os portais e respectivas sessões já autenticadas. Através destes registos, o Access Server verifica se a *Virtual Persona* associada ao *cookie* SSO_PT recebido já tem alguma sessão autenticada no Portal B. Como tal não se verifica, o Access Server efectua um pedido ao GIPTIN a fim de obter dos mapeamentos da *Virtual Persona* associada ao *cookie* SSO_PT, o *username* correspondente ao Portal B. Obtido o *username* com sucesso, o Access Server acede à base de dados do Portal B para adquirir o ID do utilizador. Com esta informação, o Access Server adiciona o Portal B e respectiva sessão à lista associada à *Virtual Persona*. Depois do registo desta informação, o Access Server efectua um novo *redirect* para a página definida no parâmetro *url*. Neste *redirect* é enviado também o parâmetro *auth* que contém a indicação do sucesso da autenticação e o ID da sessão criada entre o dispositivo do utilizador e o Portal B e o parâmetro *claim* com o ID do utilizador no Portal B, como representado no diagrama de sequência da Figura 4.9.

Quando o pedido chega ao Portal B, este verifica que a sessão ainda não está autenticada e, como

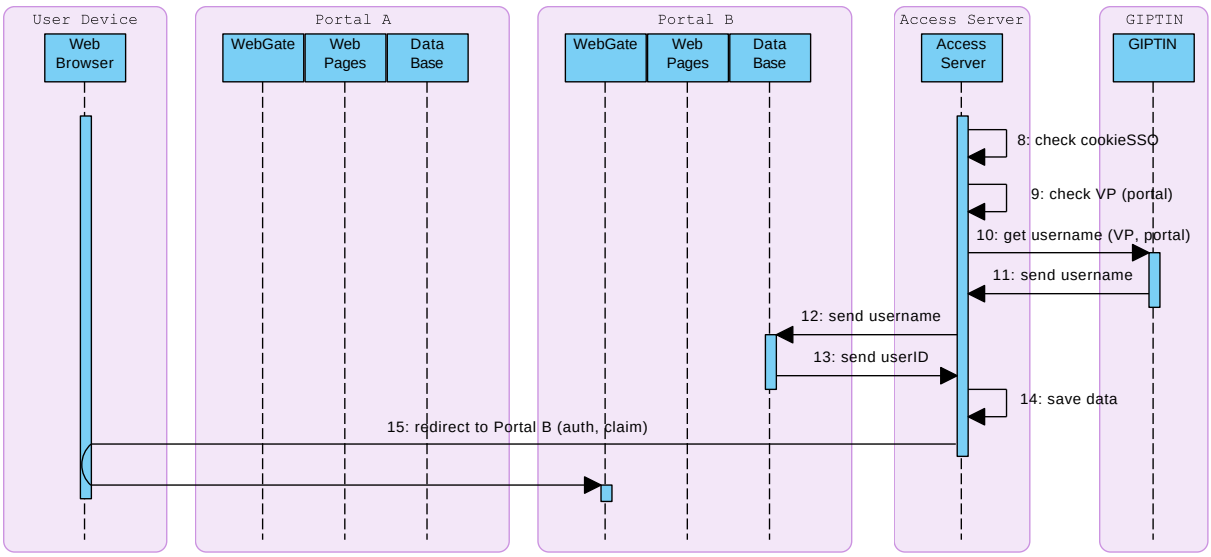

Figura 4.9: Diagrama de Sequência da autenticação em dois portais (Parte 2)

não existe nenhum parâmetro *login*, também não se trata de um pedido de autenticação. No entanto este pedido já contém o parâmetro *auth*, que contém agora a informação de que o Access Server autenticou o utilizador com sucesso e contém também o ID de sessão que foi passado ao Access Server. Se este ID de sessão corresponder ao ID da sessão criada entre o dispositivo do utilizador e o Portal B, a autenticação é concluída com sucesso e a WebGate autentica a sessão.

Depois disso, a WebGate envia para a página solicitada do Portal B o ID do utilizador, por forma a que esta possa identificar o utilizador e então apresentar a página ao utilizador. A Figura 4.10 representa o diagrama de sequência que representa estas últimas iterações.

A partir daqui, a WebGate do Portal B continua a filtrar os pedidos mas como verifica que a

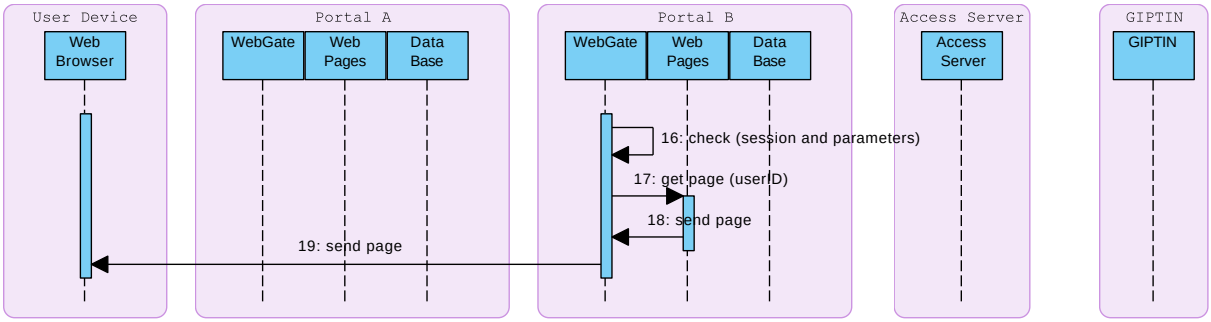

Figura 4.10: Diagrama de Sequência da autenticação em dois portais (Parte 3)

sessão está autenticada, passa os pedidos directamente para as páginas pretendidas.

O utilizador pode agora navegar nas páginas do Portal A e do Portal B sem ter de se voltar a autenticar. Este cenário refere-se apenas ao SSO entre dois portais distintos, no entanto este procedimento pode ser aplicado a qualquer outro portal, isto é, um utilizador depois de se autenticar num portal pode navegar por qualquer outro desde que a *Virtual Persona* utilizada tenha um mapeamento para o portal solicitado.

4.4 Cenário III - Autenticação em dispositivos diferentes

Até agora foram apresentados cenários em que a autenticação ou SSO é efectuada sempre no mesmo dispositivo. Com o cenário definido neste secção, pretende-se mostrar que a arquitectura do Terminal Virtual proposta permite para além disso, efectuar também o SSO em diferentes dispositivos do mesmo utilizador.

Para tal, serão utilizados dois dispositivos, o Dispositivo 1 e Dispositivo 2, previamente registados na base de dados definida na Secção 3.4.2 e apenas um portal, o Portal A. Estes dispositivos têm instalada e activada a componente Intelligent Authentication e cada um deles tem associado uma *Virtual Persona*. A associação das *Virtual Personas* foi previamente efectuada pelo utilizador, através do portal de gestão dos dispositivos do utilizador e das suas *Virtual Personas* definido na Secção 3.5.4. Este cenário está representado na Figura 4.11.

Neste cenário, o utilizador começa por autenticar-se no Portal A com o Dispositivo 1 e poste-

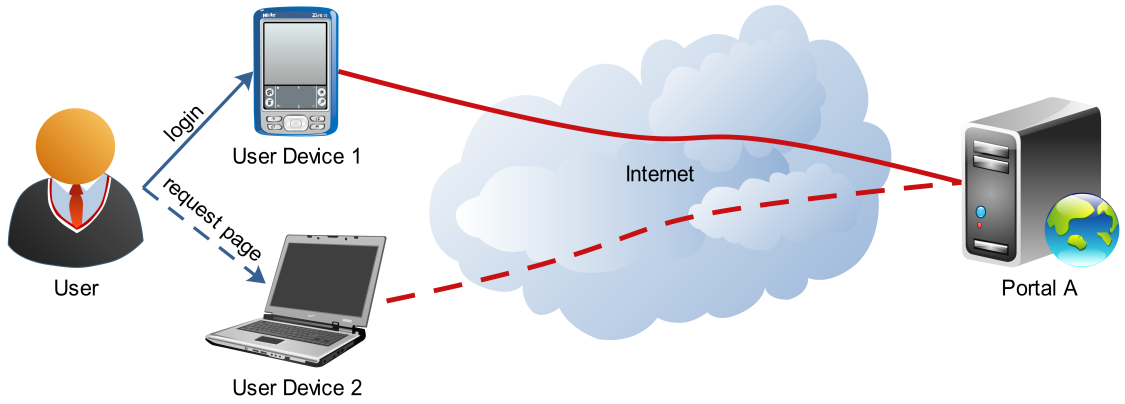

Figura 4.11: Cenário III

riormente, passa a usar o Dispositivo 2 para aceder novamente ao Portal A, ou seja, o utilizador ao mudar para o Dispositivo 2 deixa de ter a sessão autenticada e deixa também de ter o *cookie* SSO_PT que está contido no *WebBrowser* do Dispositivo 1. A forma como é efectuada a autenticação do utilizador no Portal A usando o Dispositivo 1, é muito semelhante à apresentada no Cenário I.

O utilizador começa por efectuar um pedido de uma página ao Portal A, através do *WebBrowser* instalado no Dispositivo 1. Quando a WebGate do Portal A recebe o pedido, é imediatamente verificado se a sessão criada entre o Dispositivo 1 e o Portal A está autenticada. Como a sessão não está ainda autenticada, a WebGate verifica se o pedido inclui um parâmetro *login* com a indicação de que se pretende efectuar um autenticação. Como tal não se verifica, a WebGate verifica a existência do parâmetro *auth* no pedido. Novamente a verificação não sucede e consequentemente a WebGate efectue um *redirect* para o Access Server. Este pedido contém os parâmetros *url* e *session*, que são o URL pedido e o ID da sessão entre o Dispositivo 1 e o Portal A, respectivamente.

Durante o processo de estabelecimento de uma sessão entre o Dispositivo 1 e o Access Server, a componente Intelligent Authentication instalada no Dispositivo 1 recolhe a informação relativa ao ID de sessão criado e envia-o para o Access Server em conjunto com a *persona* e *terminal* que este está a utilizar. Esta iteração e as restantes apresentadas anteriormente, estão representadas no diagrama de sequência da Figura 4.12.

Logo que o pedido efectuado pelo *redirect* da WebGate chega ao Access Server, este verifica se o pedido contém o *cookie* SSO_PT. Pelo facto do pedido não incluir o *cookie* desejado, o Access Server começa a verificar se o pedido inclui o parâmetro *login* com a indicação de que se trata de um pedido de autenticação. Visto que o parâmetro *login* não está contido no pedido, o Ac-

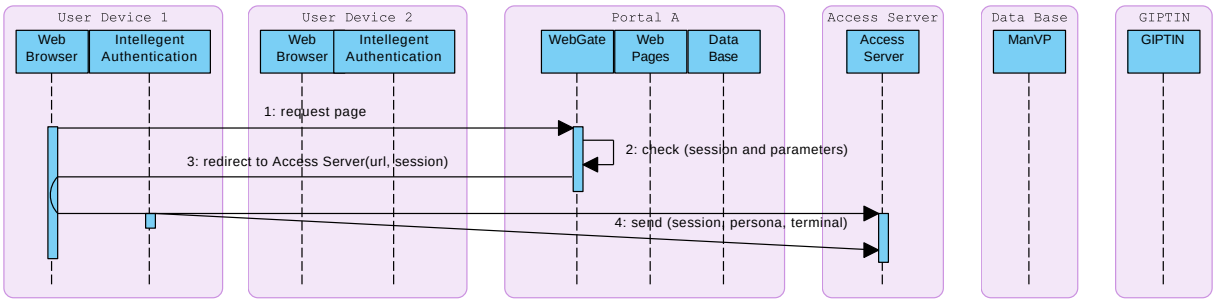

Figura 4.12: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 1)

cess Server efectua uma procura na *HashMap mapSession_PersonaTerminal* a fim de verificar se existe o ID da sessão entre o Dispositivo 1 e o Access Server, ou seja, se a componente Intelligent Authentication enviou para o Access Server os dados necessários à identificação do utilizador e do terminal o que inclui o ID de sessão criada entre o Dispositivo 1 e o Access Server. Esta procura é repetida de 500 em 500 milissegundos até ter sucesso ou ser atingido um tempo limite definido previamente nas configurações do Access Server ou calculado em tempo real, utilizando para o efeito o tempo de ida e volta de um pacote (RTT) entre o Dispositivo 1 e o Access Server. Uma vez que o dispositivo que o utilizador está a usar tem instalada e activa a componente Intelligent Authentication, e esta já enviou para o Access Server o ID da sessão, a *persona* e *terminal*, a procura tem sucesso. Posto isto, o Access Server obtém da *HashMap mapSession_PersonaTerminal* a informação relativa à *persona* e ao *terminal*, para poder obter da base de dados de suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*, a *Virtual Persona* associada ao Dispositivo 1 nesse dia, a essa hora.

Depois de obter a *Virtual Persona*, o Access Server verifica se existe algum *cookie* SSO_PT associado a esta *Virtual Persona*, isto é, verifica se a *Virtual Persona* já foi autenticada em algum portal através de um qualquer dispositivo. Esta verificação não sucede, pelo que o Access Server efectua um *redirect* para a página definida no parâmetro *url* com o parâmetro *auth* que contém a indicação de que é necessário pedir as credenciais ao utilizador, tal como mostra o diagrama de sequência da Figura 4.13.

Com o pedido de volta ao Portal A, é efectuado pela WebGate todas as verificações descritas anteriormente. Mesmo não tendo a sessão autenticada, este pedido que chega novamente ao Portal A já contém o parâmetro *auth*. Este parâmetro possui informação que indica à WebGate que é necessário o utilizador apresentar as suas credenciais. Então a WebGate apresenta ao utilizador a *form* onde este deve inserir as suas credenciais. Depois do utilizador inserir as respectivas credenciais, envias para o Portal A. A *form* de autenticação está desenvolvida de forma a que

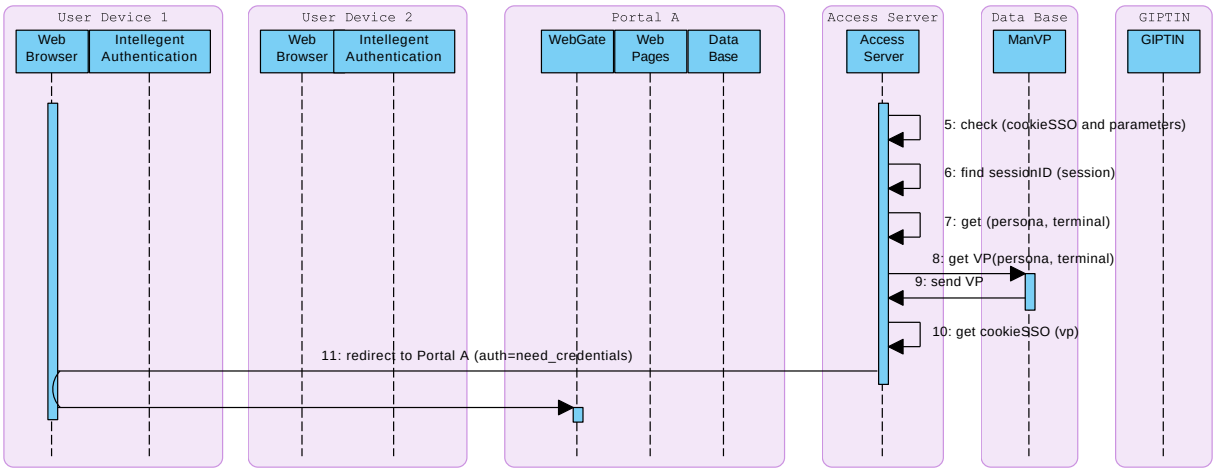

Figura 4.13: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 2)

quando seja enviado as credenciais do utilizador para o respectivo portal, envie também o parâmetro *login* com a indicação de que se pretende efectuar uma autenticação. De novo chega à WebGate um pedido com uma sessão inválida que é verificado mas desta vez este pedido inclui o parâmetro *login*. Posto isto, a WebGate efectua novamente um *redirect* para o Access Server com os parâmetros *url* e *session*, já descritos anteriormente, e ainda com os parâmetros *user*, *pass* e *login*, que são, respectivamente, *username* e *password* enviados pelo utilizador e a indicação de que se pretende efectuar uma autenticação, como pode ser visto no diagrama de sequência representado na Figura 4.14.

Uma vez mais o pedido chega ao Access Server sem o *cookie* SSO_PT mas agora trás consigo o

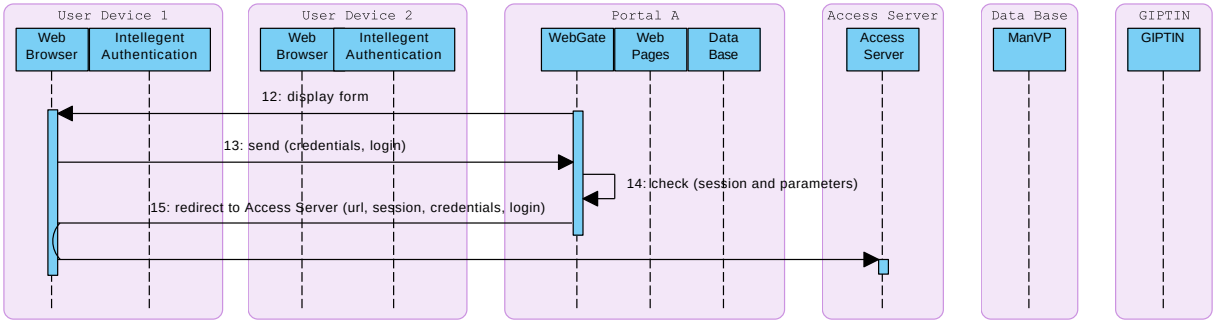

Figura 4.14: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 3)

parâmetro *login* e os restantes parâmetros. Desta forma o Access Server procede à validação das credenciais do utilizador na base de dados do portal requerido. Com o sucesso da validação das credenciais, o Access Server cria um *cookie* SSO_PT com o ID do próprio *cookie* e com o ID

da sessão entre o dispositivo do utilizador e o Access Server. Dada a importância deste *cookie*, a inclusão do ID de sessão prende-se com o facto da possibilidade, ainda que muito remota, deste poder ser extorquido na rede e poder ser usado por outro utilizador. Desta forma, tal utilização é impossibilitada, porque a verificação do *cookie* SSO_PT por parte do Access Server inclui a verificação do ID de sessão. Depois da criação do *cookie* SSO_PT o Access Server pede também ao GIPTIN a *Virtual Persona* que contém o mapeamento do *username* com o respectivo portal. Com estes dados, o Access Server regista nas suas estruturas internas o *cookie* SSO_PT e a *Virtual Persona* associada e regista também o portal e respectivo ID de sessão associado à *Virtual Persona* utilizada. De seguida é efectuado novo *redirect* para a página definida no parâmetro *url* com o parâmetro *auth* que contém a indicação do sucesso da autenticação e o ID da sessão criada entre o dispositivo do utilizador e o Portal A, o parâmetro *claim* com o ID do utilizador e com o *cookie* SSO_PT que será guardado no *WebBrowser* do dispositivo do utilizador. Todas estas verificações e interacções, estão representadas no diagrama de sequência que a Figura 4.15 apresenta.

Quando o pedido chega ao Portal A, este verifica que a sessão ainda não está autenticada e, como

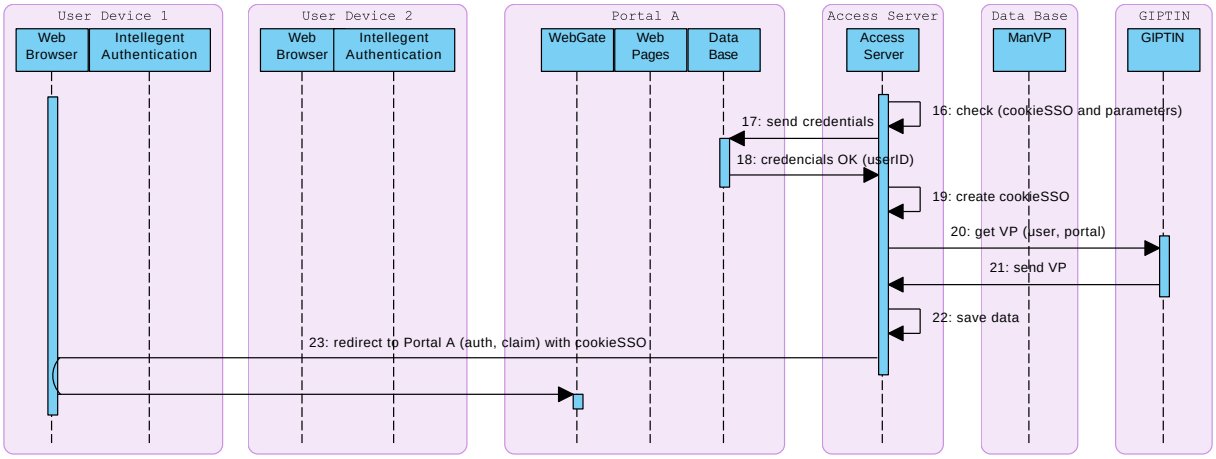

Figura 4.15: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 4)

não existe nenhum parâmetro *login*, também não se trata de um pedido de autenticação. No entanto este pedido já contém o parâmetro *auth*, que contém agora a informação de que o Access Server autenticou o utilizador com sucesso e contém também o ID de sessão que foi passado ao Access Server. Se este ID de sessão corresponder ao ID da sessão criada entre o Dispositivo 1 e o Portal A, a autenticação é concluída com sucesso e a WebGate autentica a sessão. Depois disso a WebGate envia para a página solicitada do Portal A o ID do utilizador, por forma a que esta possa identificar o utilizador e então apresentar a página, tal como é apresentado no diagrama de

sequência da Figura 4.16.

Entretanto, o utilizador efectuou um pedido ao Portal A mas agora utilizando o Dispositivo 2.

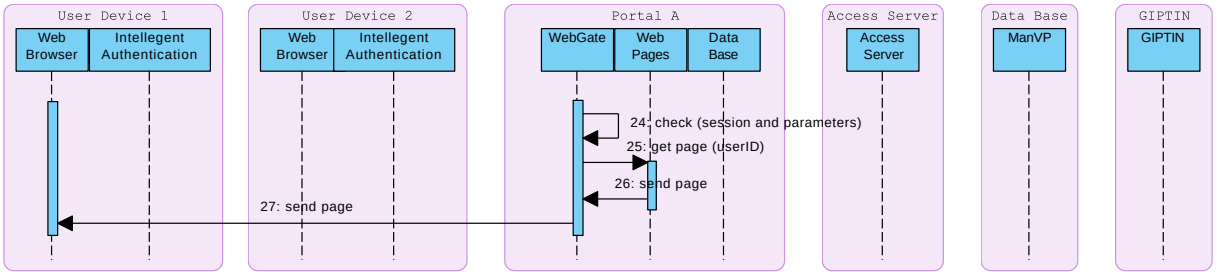

Figura 4.16: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 5)

O pedido ao chegar ao Portal A é prontamente verificado pela WebGate do mesmo portal. Esta começa por verificar se o pedido tem a sessão autenticada mas como tal não sucede, a WebGate verifica se se trata de um pedido de autenticação, ou seja, se o pedido inclui o parâmetro *login* com essa indicação. Como a verificação volta a falhar, a WebGate procura agora um outro parâmetro, o parâmetro *auth*. Este parâmetro também não faz parte do pedido, o que leva a WebGate a efectuar um *redirect* para o Access Server com o parâmetro *url* que contém o URL do pedido efectuado pelo utilizador ao Portal A e o parâmetro *session* que contém o ID da sessão entre o Dispositivo 2 e o Portal A, como é possível ver pelo diagrama de sequência representado na Figura 4.17.

Um vez que o utilizador passou a utilizar o Dispositivo 2 para efectuar os pedidos ao Portal

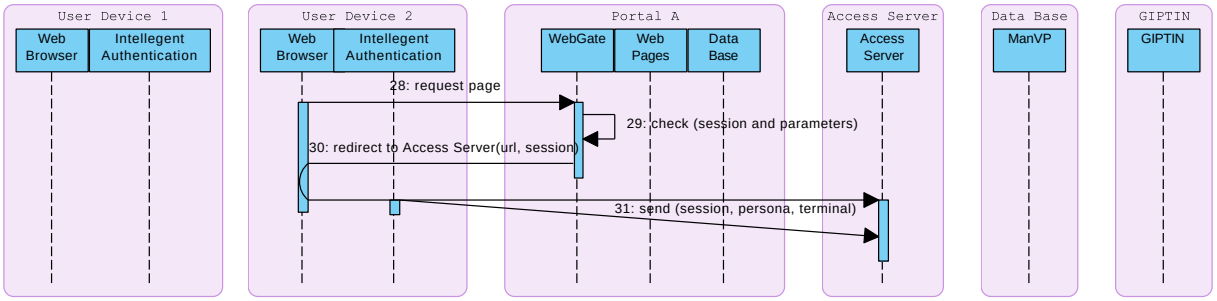

Figura 4.17: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 6)

A, a sessão que está autenticada no Dispositivo 1 é diferente da sessão que foi estabelecida entre o Dispositivo 2 e o Portal A, pelo que esta não está autenticada. Para além disso, o *cookie* SSO_PT apenas está disponível no *WebBrowser* do Dispositivo 1. No entanto, ao ser efectuado o *redirect* para o Access Server, a componente Intelligent Authentication instalada no Dispositivo

2 detecta o estabelecimento de uma sessão entre o Dispositivo 2 e o Access Server, e envia para este último a informação relativa ao ID da sessão criada, a *persona* e o *terminal* que efectuou o pedido. Esta situação faz com que quando o pedido chega ao Access Server a verificação do *cookie* SSO_PT falhe. De seguida o Access Server verifica se o pedido inclui o parâmetro *login* com a indicação de que se trata de um pedido de autenticação mas mais uma vez o parâmetro *login* não está contido no pedido. Posto isto, o Access Server efectua uma procura na *HashMap mapSession_PersonaTerminal* a fim de verificar se existe o ID da sessão estabelecida entre o Dispositivo 2 e o Access Server, ou seja, se a componente Intelligent Authentication enviou para o Access Server a informação relativa a essa sessão e a informação da *persona* e *terminal* que efectuaram o pedido. Como a componente Intelligent Authentication do Dispositivo 2, já enviou para o Access Server essas informações, a procura tem sucesso.
Posto isto, o Access Server obtém da *HashMap mapSession_PersonaTerminal* a informação relativa à *persona* e ao *terminal*, a fim de poder obter da base de dados definida na Secção 3.4.2, a *Virtual Persona* associada ao Dispositivo 2 nesse dia, nessa hora. Com a obtenção da *Virtual Persona*, o Access Server verifica se existe algum *cookie* SSO_PT associado a esta *Virtual Persona*, isto é, verifica se a *Virtual Persona* já foi autenticada em algum portal através de um qualquer dispositivo. Como esta *Virtual Persona* já tem uma sessão autenticada, esta verificação tem sucesso, o que faz com que o Access Server efectue um novo pedido ao GIPTIN para adquirir dos mapeamentos dessa *Virtual Persona*, o *username* correspondente ao Portal A. Adquirido o *username* com sucesso, o Access Server acede à base de dados do Portal A para obter o ID do utilizador no Portal A. Posteriormente, o Access Server cria um *cookie* SSO_PT com o ID do próprio *cookie* e com o ID da sessão estabelecida entre o Dispositivo 2 e o Access Server. Toda esta informação é regista pelo Access Server nas suas estruturas internas, de forma a associar ao *cookie* SSO_PT a *Virtual Persona* utilizada e associar à *Virtual Persona* o portal e respectivo ID da sessão estabelecida entre o Dispositivo 2 e o Portal A.
De seguida é efectuado pelo Access Server um novo *redirect* para a página definida no parâmetro *url* com o parâmetro *auth* a indicar o sucesso da autenticação e o ID da sessão criada entre o Dispositivo 2 e o Portal A, o parâmetro *claim* com o ID do utilizador no Portal A e, por fim, com o *cookie* SSO_PT que será armazenado no *WebBrowser* do Dispositivo 2, como se pode ver no diagrama de sequência da Figura 4.18.
O pedido que chega agora ao Portal A, ainda não contém a sessão autenticada e, como não existe nenhum parâmetro *login*, também não se trata de um pedido de autenticação. Todavia, este pedido contém o parâmetro *auth* com a informação de que o Access Server autenticou o utilizador com sucesso e contém também o ID de sessão que foi passado ao Access Server. Se este ID de

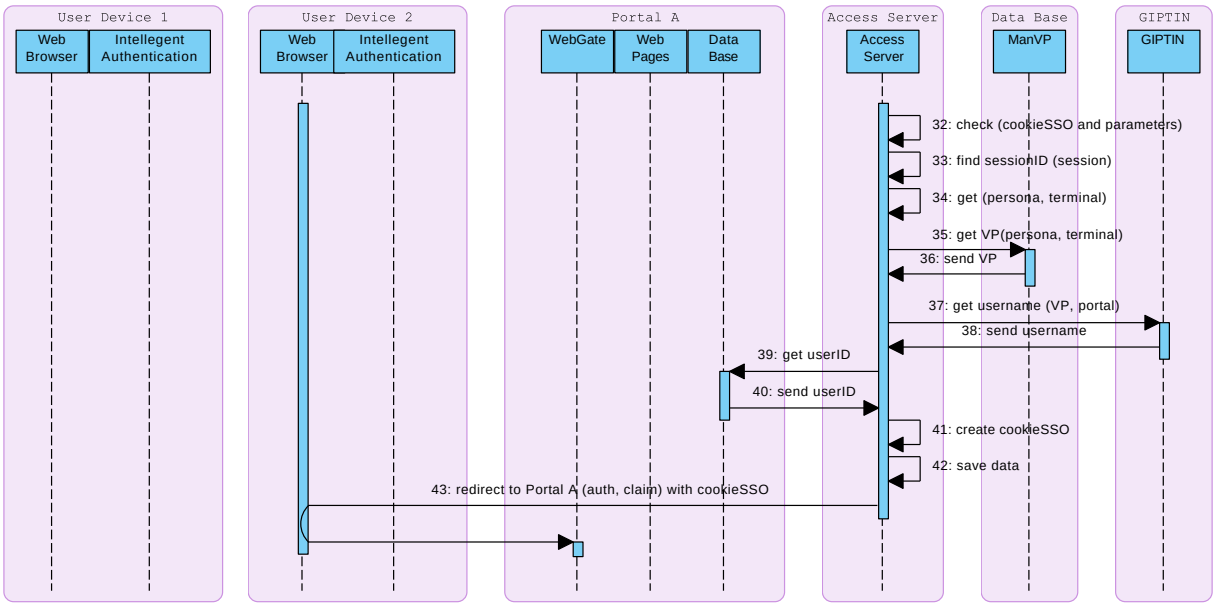

Figura 4.18: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 7)

sessão enviado pelo Access Server corresponder ao ID da sessão estabelecida entre o Dispositivo 2 e o Portal A, a autenticação é concluída com sucesso e a WebGate efectua a autenticação desta sessão. Por conseguinte, a WebGate envia para a página solicitada do Portal A o ID do utilizador, por forma a que esta possa identificar o utilizador e então proceder à apresentação da página ao utilizador, como mostra o diagrama de sequência representado na Figura 4.19. A partir deste momento, o utilizador pode navegar pelo Portal A em qualquer dos dois dispositivos.

Este procedimento apresentado pode ser aplicado a todos os dispositivos do utilizador, desde que

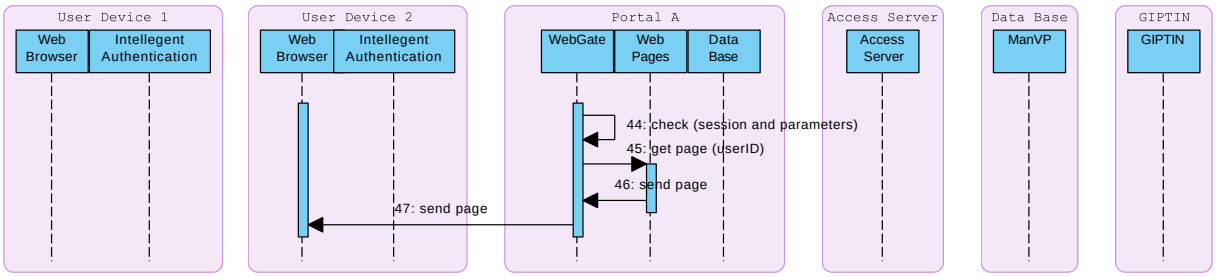

Figura 4.19: Diagrama de Sequência da autenticação em dois dispositivos diferentes (Parte 8)

a componente Intelligent Authentication esteja instalada e activada em cada um deles. É também possível o utilizador efectuar o SSO entre os diversos portais, como apresentado no Cenário II.

4.5 Cenário IV - *Logout* em todos os portais

Partindo da junção do Cenário II com o Cenário III apresentados anteriormente e da inclusão de mais um portal autenticado no Cenário II, o utilizador está agora autenticado em três portais distintos, utilizando dois dispositivos diferentes.
Neste cenário, o utilizador pretende efectuar o *logout* de todos os portais em todos os dispositivos, ou seja, efectuar o SSOff. Para proceder ao SSOff o utilizador não necessita efectuar *logout* em cada portal individualmente, nem em cada dispositivo. Basta para isso efectuar *logout* apenas num dos dispositivo e apenas num dos portais (Figura 4.20).

De modo a executar o SSOff, o utilizador começa por efectuar um pedido de *logout* no Portal

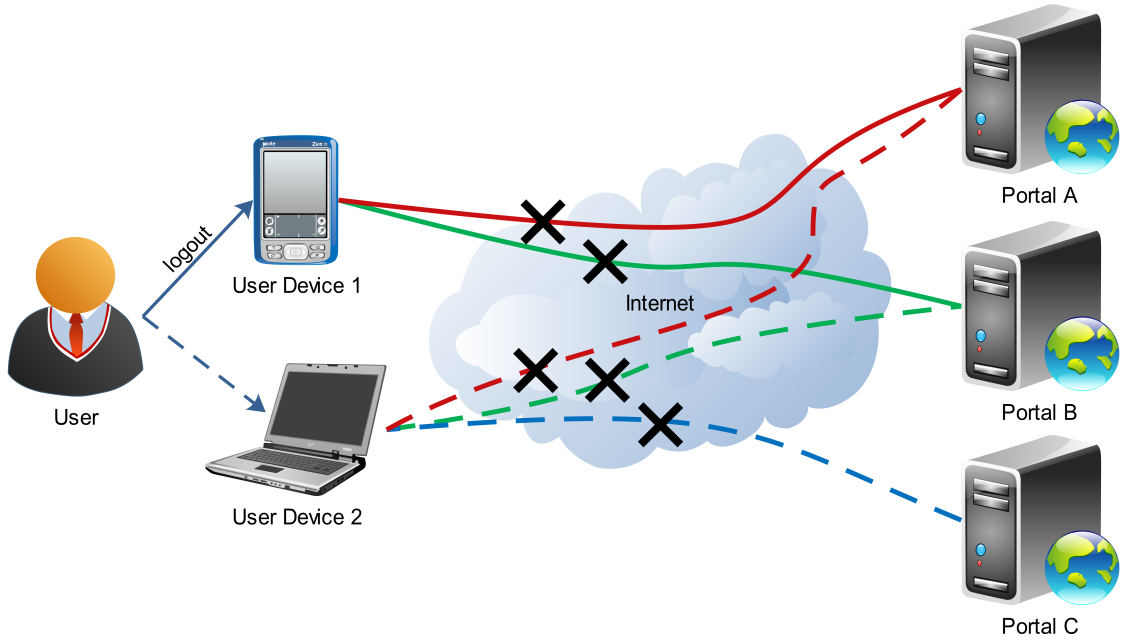

Figura 4.20: Cenário IV

A através do Dispositivo 1. Este pedido é enviado para o Portal A com um parâmetro *logout* com a indicação de que se pretende efectuar o *logout* daquela sessão. Quando o pedido chega à WebGate do Portal A, esta verifica e conclui que a sessão está autenticada. Então procede à verificação da existência do parâmetro *logout* com a informação válida de finalização da sessão. Como tal se verifica, a WebGate efectua um *redirect* para o Access Server com um parâmetro *url* e um parâmetro *logout* com a indicação de que o utilizador pretende efectuar o SSOff, tal como mostra o diagrama de sequência na Figura 4.21.

Quando o Access Server recebe o pedido efectuado pela WebGate do Portal A através do *redi-*

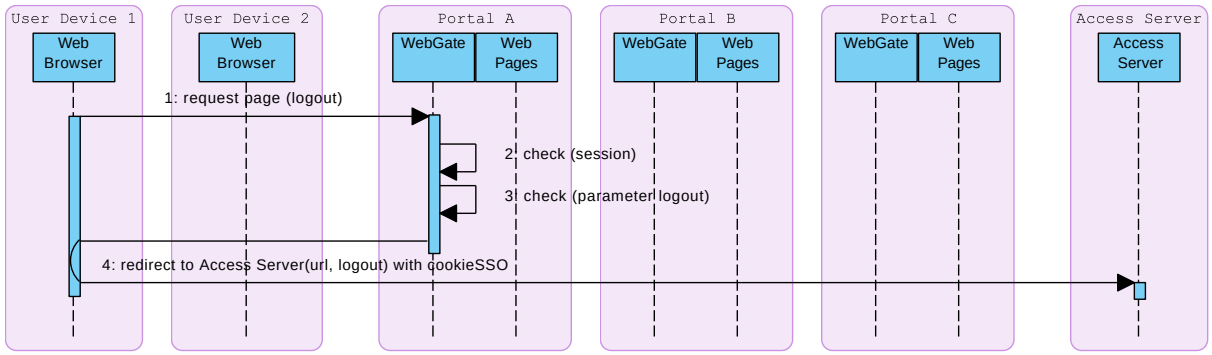

Figura 4.21: Diagrama de Sequência para efectuar o SSOff (Parte 1)

rect, este verifica se o pedido inclui o *cookie* SSO_PT. Como o pedido passa pelo *WebBrowser* do Dispositivo 1, traz também o *cookie* SSO_PT enviado anteriormente pelo Access Server e a verificação do mesmo é concluída com sucesso. Novamente é verificado pelo Access Server se o *cookie* está registado na lista de *cookies* activos. É ainda verificado se o *cookie* SSO_PT pertence à sessão que foi estabelecida entre o Dispositivo 1 e o Access Server. Visto que o *cookie* pertence de facto à sessão estabelecida, o Access Server verifica agora a existência do parâmetro *logout* com a indicação da execução do processo de SSOff. Como tal se verifica, o Access Server obtém da lista de *cookies* activos, a *Virtual Persona* associada ao respectivo *cookie*. Através da *Virtual Persona* obtida e da informação registada nas suas estruturas internas, o Access Server consegue adquirir a lista dos portais e respectivas sessões, onde a *Virtual Persona* está autenticada. Neste caso, a *Virtual Persona* tem associado o Portal A, o Portal B e o Portal C e as respectivas sessões autenticadas.

Com esta informação, o Access Server procede ao envio, via *socket*, das sessões a invalidar para o Portal A, o Portal B e para o Portal C. Em cada um destes portais, a componente *Manager Sessions* contida na WebGate respectiva, efectua a invalidação das sessões recebidas, retirando-as da lista de sessões autenticadas na respectiva WebGate.

Apesar da sessão estabelecida entre o Dispositivo 1 e o Portal A poder ser invalidada logo que chega à WebGate do Portal A o pedido de *logout*, tal não é efectuado e o Access Server envia também para o Portal A as sessões a invalidar. Este procedimento é necessário, devido ao facto de poder existir ainda sessões pertencente à *Virtual Persona* autenticadas no Portal A mas com outro dispositivo. Neste caso, existe ainda uma sessão autenticada, a sessão que foi estabelecida entre o Dispositivo 2 e o Portal A. Estas iterações estão representadas no diagrama de sequência apresentado na Figura 4.22.

Posteriormente, o Access Server procede à invalidação de todos os *cookie* associados à *Virtual*

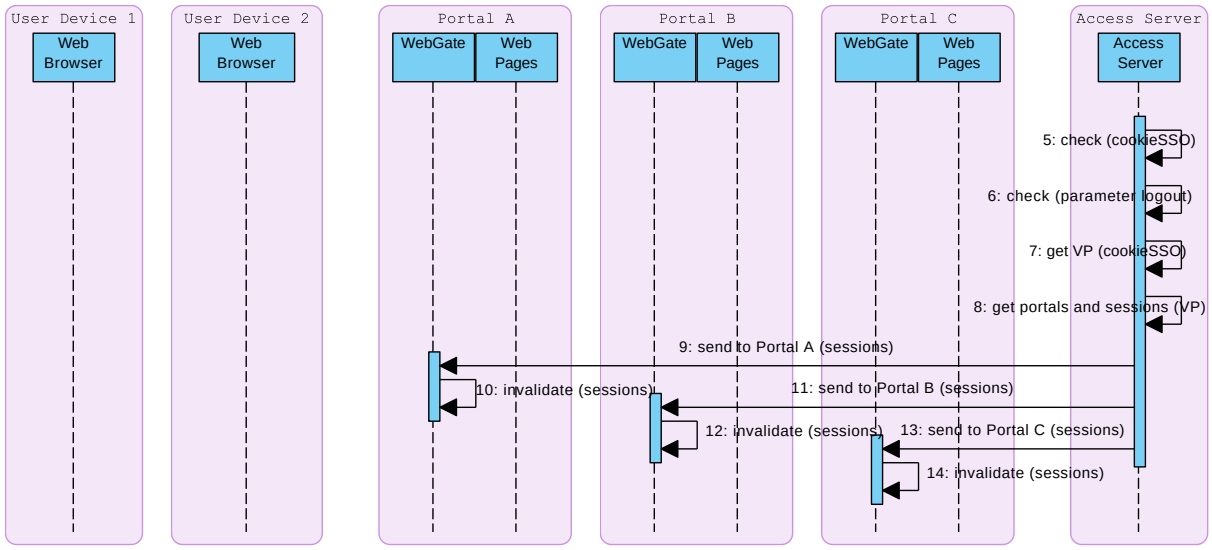

Figura 4.22: Diagrama de Sequência para efectuar o SSOff (Parte 2)

Persona. Associado à mesma *Virtual Persona* existe portais e respectivas sessões autenticadas, que são também invalidados. Finalmente, o Access Server efectua um *redirect* para a página especificada no parâmetro *url* com o parâmetro *auth* a indicar que se procedeu ao SSOff.
Este pedido que chega agora ao Portal A, é examinado pela WebGate que imediatamente verifica que a sessão não está autenticada. Como o pedido também não contém o parâmetro *login*, a WebGate verifica que existe o parâmetro *auth*. Este parâmetro tem agora a indicação de que se procedeu ao SSOff, o que faz com que a WebGate apresente ao utilizador a página de *login* ou uma outra página *default* definida previamente, como se pode ver no diagrama de sequência apresentado na Figura 4.23.

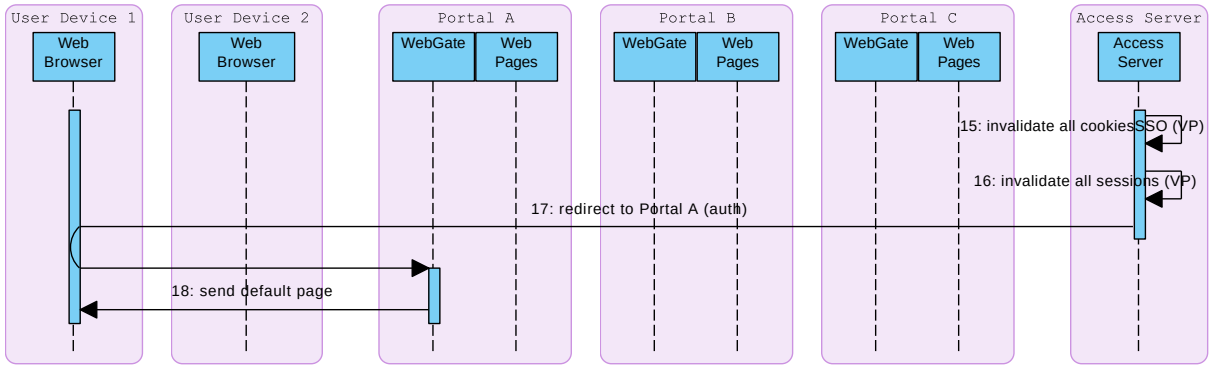

Figura 4.23: Diagrama de Sequência para efectuar o SSOff (Parte 3)

Com estes procedimentos, o utilizador invalidou todas as sessões nos respectivos portais, ao efectuar *logout* apenas num deles.

4.6 Sumário

Neste capítulo foram apresentados alguns cenários de utilização da arquitectura do Terminal Virtual. Os cenários apresentados, visam apresentar os mecanismos de autenticação utilizador através de diagramas de sequência que mostram as comunicações que são efectuadas entre as diversas componentes da arquitectura do Terminal Virtual. Através destes cenários é mostrado também como se efectua a autenticação num portal a partir de um dispositivo, até a autenticação em vários portais a partir de diferentes dispositivos sem que o utilizador necessite de introduzir as suas credenciais mais do que uma vez.

Capítulo 5

Resultados

5.1 Introdução

Depois da elaboração da arquitectura do Terminal Virtual descrita na Secção 3.2 e da implementação da arquitectura final, foram também especificados cenários concretos de utilização com diferentes dispositivos e diferentes portais genéricos. Para que implementação fosse verificada e validada foram efectuados alguns testes para verificar se o comportamento de todo o sistema seria o esperado. A fim de efectuar essas verificações e validações, foram realizador testes funcionais para verificar os cenários de utilização fundamentais. Através deste tipo de testes, foi possível conferir se as especificações do sistema foram cumpridas.
Com esta arquitectura algumas tarefas do dia-a-dia do utilizador estão agora modificadas. Um utilizador pode agora trocar de dispositivo sem ter de se preocupar com as suas credenciais. É também oferecido ao utilizador uma visão global de todos os seus dispositivos.
Neste capítulo são apresentados os teste funcionais para alguns cenários específicos e também é aqui apresentado o impacto que esta nova arquitectura produz no utilizador.

5.2 Testes Funcionais

Para efectuar os testes funcionais, foram criados três portais de teste similares aos que são utilizados pela TMN [35], MEO [36] e o Sapo Mail [37] e o portal Access Server que é o portal onde o utilizador pode gerir todos os seus dispositivos e *Virtual Personas* associadas. Apesar do nome do portal ser igual ao nome do servidor de Terminais Virtuais (Access Server), quando se

fala do portal Access Server está-se a referir ao portal que disponibiliza a interface *web* para a referida gestão e não ao servidor. Foram também usados três dispositivos diferentes como sendo pertencentes ao mesmo utilizador. Estes dispositivos estão especificados na Tabela 5.1.

A escolha destes dispositivos foi elabora por forma a poder executar os testes em diferentes

Tabela 5.1: Dispositivos utilizados nos testes funcionais

Device Name	**Type**	**Operating System**	**Network Interface**	**Web Browser**
HTC Google Nexus One	Smartphone	Android OS 2.1	3G	Android OS 2.1 Browser
Apple MacBook	Laptop	Mac OS X Version 10.5.8	Wireless	Safari Version 5.0.2
PC Work	Desktop	Ubuntu Version 9.04	Ethernet	FireFox Version 3.5.3

tipos de dispositivos (móveis ou fixos), com diferentes sistemas operativos e, por fim mas não menos importante, com diferentes ligação à Internet.

Para uma melhor execução dos testes funcionais, foram definidos quatro cenários de teste. Os três primeiros têm por base os cenários definidos no Capítulo 4, onde o utilizador acede através dos dispositivos apresentados, aos portais anteriormente referidos. O quarto e último cenário pretende mostrar e testar a gestão dos dispositivos e das *Virtual Personas* do utilizador. O utilizador aqui apresentado para efectuar os testes funcionais é chamado de Manuel Campos e o seu email (de teste) utilizado é *mcm@sapo.pt*.

- **Cenário de Teste I -** Este cenário mostra como o utilizador acede ao portal TMN e posteriormente ao portal Sapo Mail, o utilizando apenas o dispositivo *laptop* Apple MacBook.
- **Cenário de Teste II -** Neste cenário, o utilizador começa por aceder ao portal Sapo Mail através do seu *smartphone* HTC Google Nexus One. De seguida o utilizador tenta aceder de novo ao portal Sapo Mail mas agora através do seu *desktop* de trabalho PC Work.
- **Cenário de Teste III -** No cenário aqui apresentado, utilizador efectua *login* no portal TMN a partir do seu *smartphone* HTC Google Nexus One, a seguir tenta aceder ao portal Sapo Mail utilizando o seu *laptop* Apple MacBook. Ainda no seu Apple MacBook, o utilizador tenta ao portal MEO. Por fim, o utilizador efectua *logout*.
- **Cenário de Teste IV -** Neste último cenário, o utilizador acede ao portal Access Server para efectuar a gestão dos seus dispositivos e das suas *Virtual Personas*.

Através dos cenários de teste aqui apresentados, foi possível verificar todas as funcionalidades principais do Terminal Virtual, de modo a oferecer ao utilizador uma forma simples e ao mesmo tempo segura, deste se autentica em qualquer portal. É também disponibilizado ao utilizador uma vista global de todos os seus dispositivos e *Virtual Personas* e, uma forma simples e inovadora, efectuar associações das *Virtual Personas* aos dispositivos.

5.2.1 Cenário de Teste I

Neste cenário o utilizador usa o dispositivo *laptop* Apple MacBook para efectuar *login* no portal TMN. Quando o utilizador tenta aceder ao portal TMN, esse pedido chega à WebGate do portal que verifica que a sessão não está autenticada e redirecciona o pedido para o Access Server com o ID de sessão e o URL do pedido efectuado ao portal TMN incluídos. Como o dispositivo utilizado ainda não contém nenhum *cookie* SSO_PT e o utilizador não tem ainda qualquer *Virtual Persona* autenticada, o pedido é redireccionado novamente para o URL recebido, com o ID de sessão passado e com a informação de que é necessário pedir ao utilizador as suas credenciais. Para tal, é apresentado ao utilizador a página de *login*, como mostra a Figura 5.1, onde o utilizador tem de colocar as suas credenciais.

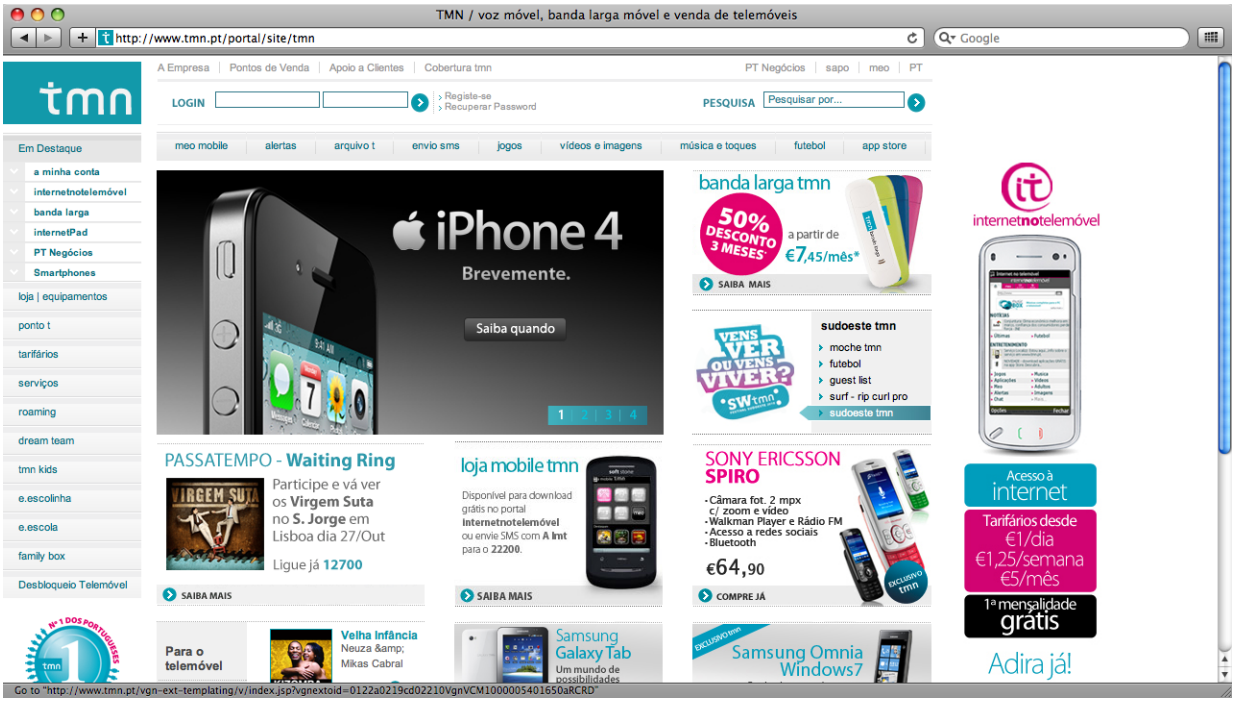

Figura 5.1: Página de *login* do portal TMN

De seguida, o utilizador introduz as suas credenciais e tenta aceder de novo. Este pedido chega agora à WebGate do portal TMN com as credenciais do utilizador. Como a sessão ainda não está autenticada e a WebGate verifica tratar-se de um pedido de autenticação, através das credenciais e da informação correspondente, a WebGate efectua um redireccionamento para Access Server com o ID de sessão, o URL do pedido, com a informação de que se trata da tentativa de autenticação e com as credenciais do utilizador contidas no pedido.
Com estas informações e como o pedido não contém o *cookie* SSO_PT, o Access Server verifica as credenciais do utilizador que lhe foram passadas. Com o sucesso deste verificação, o Access Server recebe também o ID do utilizador associado e cria um *cookie* SSO_PT que será enviado para o dispositivo do utilizador Apple MacBook através de um redireccionamento que será efectuado a seguir. Verifica também qual é a *Virtual Persona* associada às credenciais recebidas e depois de registar essas informações, efectua o redireccionamento de novo para o URL passado anteriormente, com o ID de sessão passado e com a informação de que o utilizador foi autenticado com sucesso.
Quando este pedido chega à WebGate do portal TMN, esta verifica que o utilizador foi autenticado correctamente, autentica a sessão e apresenta a página ao utilizador, como mostra a Figura 5.2. Com a apresentação desta página, é verificado assim que o utilizador autenticou-se correcta-

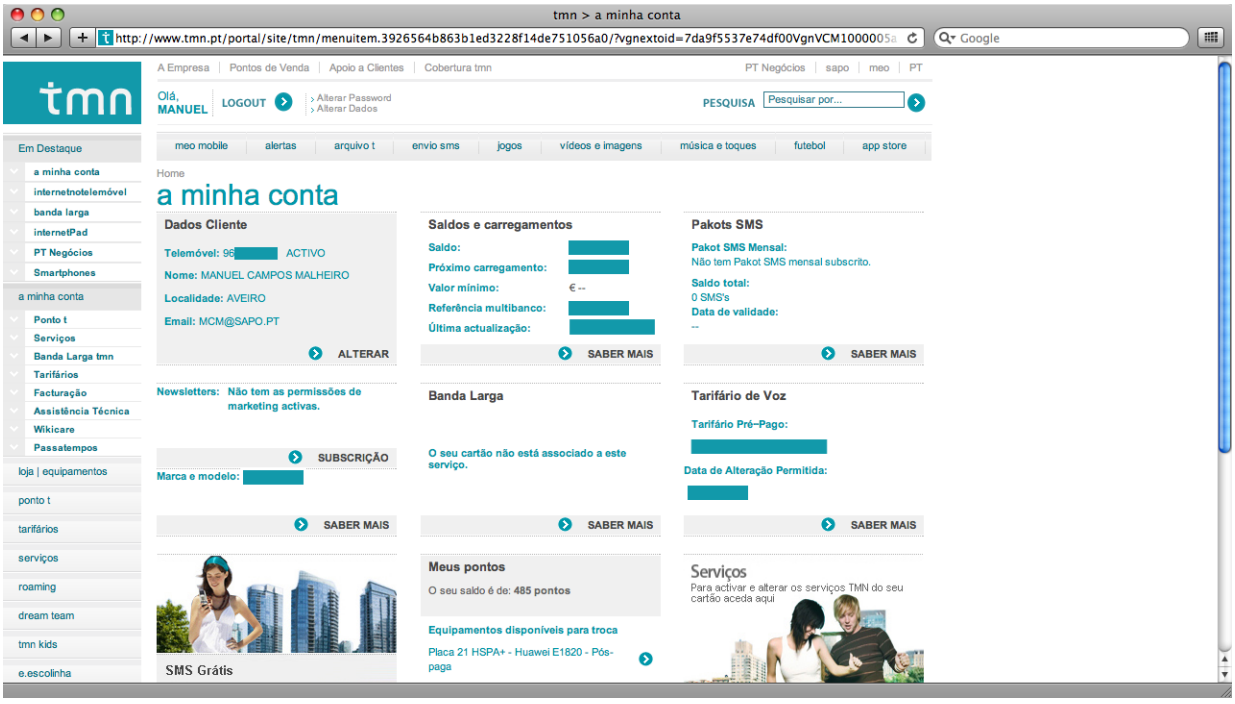

Figura 5.2: Página do portal TMN requerida pelo utilizador

mente e o sistema comportou-se como esperado, ao autenticar o utilizador e apresentar a página com os dados pessoais do utilizador.
Depois de navegar pelo portal TMN e não tendo feito *logout*, o utilizador pretende agora aceder ao portal Sapo Mail. Quando o utilizador efectua o pedido de acesso ao portal Sapo Mail, a WebGate deste portal verifica que ainda a sessão criada entre o dispositivo Apple MacBook e o portal Sapo Mail, não está autenticada. Como tal, a WebGate efectua um redireccionamento do pedido para o Access Server apenas com o ID de sessão e com o URL do pedido.
Como o utilizador já se autenticou no portal TMN, já contém no *browser* do *laptop* Apple MacBook o *cookie* SSO_PT. Como o pedido foi redireccionado para o Access Server, passa pelo *browser* do *laptop* Apple MacBook, que inclui também no pedido o *cookie* SSO_PT. Ao chegar ao Access Server, é verificado que o pedido inclui um *cookie* SSO_PT válida e ao qual já está associado uma *Virtual Persona*. Como o utilizador ainda se autenticou apenas no portal TMN, não existe ainda registado no Access Server qualquer relação entre esta *Virtual Persona* e o portal Sapo Mail. Posto isto, o Access Server efectua um pedido ao GIPTIN para adquirir o *username* que o utilizador tem no portal Sapo Mail e que está mapeado na *Virtual Persona* associada ao *cookie* SSO_PT. Neste caso, o *username* que está mapeado na *Virtual Persona* é "mcm@sapo.pt". Depois de adquirido o *username*, o Access Server acede à base de dados do portal Sapo Mail para obter o ID do utilizador. Depois de registar estas informações, o Access Server efectua um redireccionamento para o URL recebido anteriormente com a informação de que o utilizador já está autenticado, como ID de sessão e com o ID do utilizar.
Quando o pedido redireccionado pelo Access Server chega ao portal Sapo Mail, a WebGate do portal verifica que o utilizador já está autenticado, autentica a sessão e, finalmente, apresenta ao utilizador a página requerida, tal como mostra a Figura 5.3. Nesta figura é possível ver que o utilizar está autenticado correctamente, visto que a página foi apresentada e o *username* também aparece correctamente.

5.2.2 Cenário de Teste II

Neste cenário é usado um dispositivo do utilizador que está ligado à Internet através de uma ligação 3G. Este dispositivo é o *smartphone* HTC Google Nexus One. Neste caso, o utilizador começa por efectuar um pedido de acesso ao portal Sapo Mail através do *smartphone* HTC Google Nexus One. Como ainda não existe qualquer autenticação efectuada por parte deste utilizador, o pedido quando chega ao portal Sapo Mail, a WebGate do portal verifica que a sessão estabelecida entre o *smartphone* HTC Google Nexus One e o portal Sapo Mail não está auten-

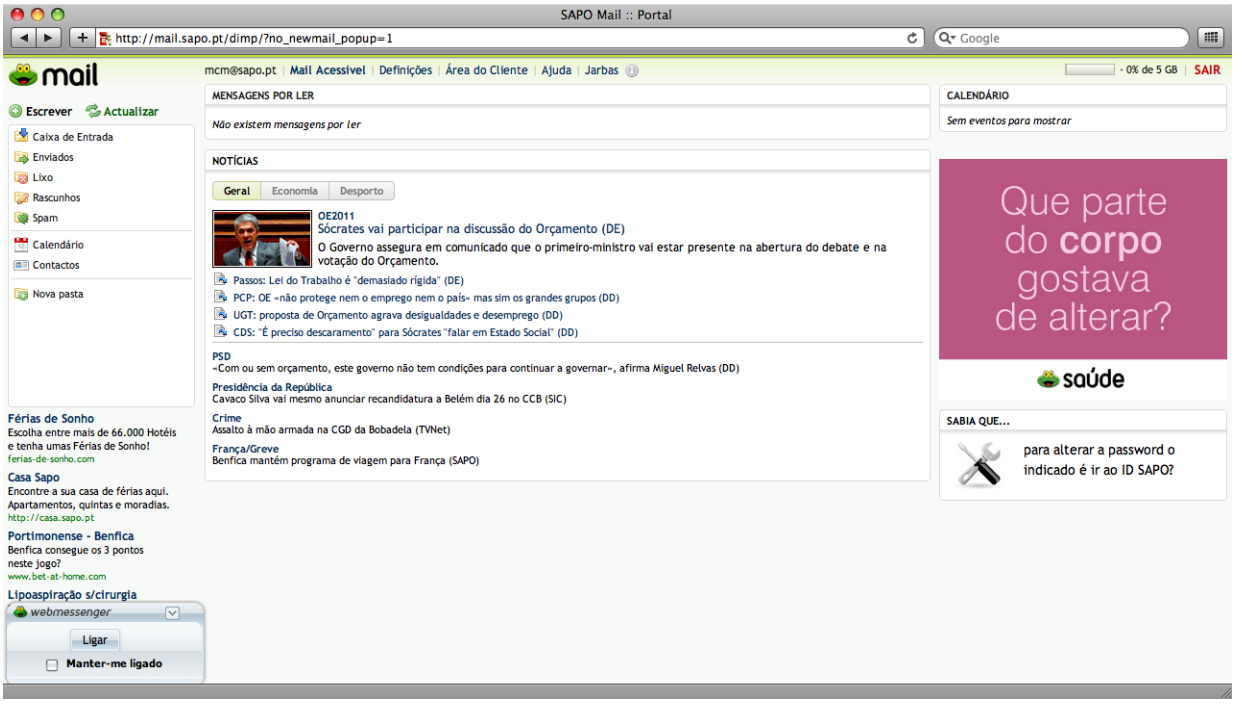

Figura 5.3: Página do portal Sapo Mail requerida pelo utilizador

ticada. Como tal, a WebGate efectua um redireccionamento para o Access Server, com o ID da sessão criada e o URL do pedido efectuado ao portal.

Logo que o pedido do redireccionamento efectuado pela WebGate chega ao Access Server, este efectua a verificação da existência do *cookie* SSO_PT. Como utilizador ainda não se autenticou em nenhum portal que tem implementado este sistema de autenticação, não inclui no *browser* do *smartphone* HTC Google Nexus One este *cookie*, pelo que a verificação não sucede. Visto que o pedido não contém o *cookie* SSO_PT e não existe no Access Server qualquer registo do sessão criada entre o *smartphone* HTC Google Nexus One e o Access Server, este último efectua um redireccionamento do pedido para o URL que lhe foi enviado pela WebGate do portal Sapo Mail. Este pedido contém o ID de sessão que a mesma WebGate enviou e a informação do insucesso da autenticação e a consequente necessidade do utilizador apresentar as suas credenciais.

Ao chegar ao portal Sapo Mail, o pedido é igualmente analisado pela WebGate do portal, a qual verifica que é necessário apresentar ao utilizador a página de *login*, tal como apresentado na Figura 5.4. Posteriormente à introdução as suas credenciais e da sua submissão, por parte do utilizador, o novo pedido que chega ao portal Sapo Mail, com a informação de que se pretende autenticar o utilizador com as credenciais que estão contidas nesse pedido. Apesar de autenticação da sessão ainda não estar efectuada, a WebGate do portal Sapo Mail verifica tratar-se de um

Figura 5.4: Página de *login* do portal Sapo Mail no *smartphone* HTC Google Nexus One

pedido de autenticação e efectua, mais uma vez, um redireccionamento do pedido para o Access Server com a indicação de que se pretende efectuar a validação das credenciais enviadas para a WebGate e que agora são enviadas para o Access Server e com a informação relativa ao ID de sessão e ao URL do pedido que chegou ao portal Sapo Mail.
Quando o pedido chega ao Access Server ainda sem o *cookie* SSO_PT, este verifica tratar-se de um pedido de autenticação. Para tal, o Access Server verifica as credenciais enviados no pedido, na base de dados do portal Sapo Mail. As credenciais são verificadas com sucesso e o Access Server recebe o ID do utilizador no respectivo portal. De seguida o Access Server cria um *cookie* SSO_PT que juntamente com o ID do utilizador recebido, com o ID de sessão contido no pedido recebido e com a indicação de que a autenticação foi efectuada com sucesso, são enviados através de um redireccionamento do pedido. Antes de efectuar o redireccionamento, o Access Server faz um pedido ao GIPTIN da *Virtual Persona* que inclui o mapeamento do *username* com o respectivo portal e também regista todas estas novas informações. Finalmente, o redireccionamento do pedido é efectuado para o URL recebido anteriormente.
Uma vez chegado o pedido ao portal Sapo Mail, a WebGate deste verifica que a sessão não está autenticada. No entanto, verifica que recebeu a informação de que a autenticação do utilizador foi efectuada com sucesso e que deve dar acesso à página pedida. Desta forma, a WebGate autentica a sessão e envia para o dispositivo do utilizador, o *smartphone* HTC Google Nexus One, a página solicitada, tal como mostra a Figura 5.5. Depois de navegar no portal Sapo Mail através do *smartphone* HTC Google Nexus One e sem efectuar *logout* deste portal, o utilizador tenta aceder novamente ao portal mas desta vez a partir do dispositivo fixo que o utilizador tem acesso

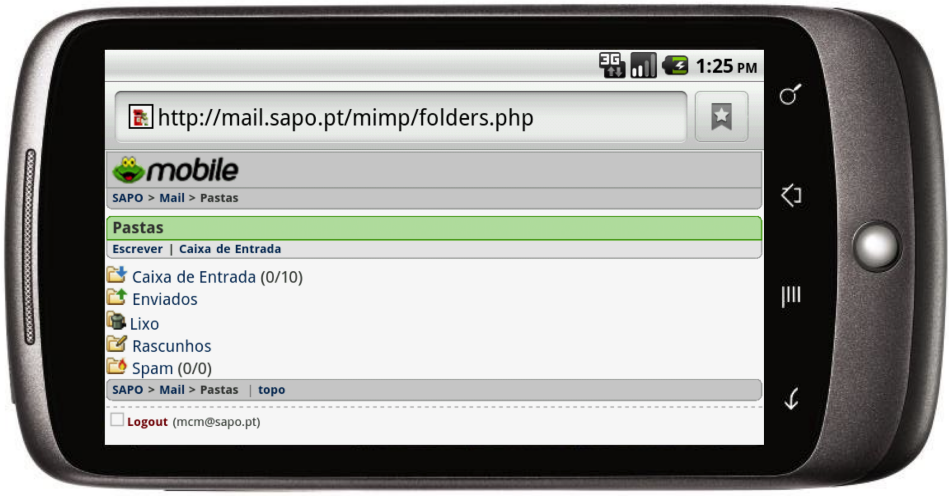

Figura 5.5: Página pedida ao portal Sapo Mail no *smartphone* HTC Google Nexus One

no seu local de trabalho. Este dispositivo é um *desktop* denominado por PC Work.

Ao efectuar o novo pedido de acesso ao portal Sapo Mail através do *desktop* PC Work, é criada uma nova sessão entre o *desktop* PC Work e o portal do Sapo Mail. Posto isto, a WebGate deste portal verifica que a sessão não está autenticada, pelo efectua um redireccionamento do pedido para o Access Server com a informação relativa ao ID da sessão criada e ao URL do pedido que chegou ao portal.

Como o utilizador ainda não efectuou qualquer autenticação num dos portais que usam este sistema de autenticação, não contém ainda o *cookie* SSO_PT no *browser* do *desktop* PC Work. Esta situação leva a que o pedido efectuado pelo redireccionamento da WebGate do portal Sapo Mail, não contém o referido *cookie*. No entanto, este dispositivo tem instalado e activado a componente Intelligent Authentication, que no momento em que o pedido passa pelo *desktop* PC Work, é responsável por enviar para o Access Server a informação relativa ao ID da sessão criada entre o *desktop* PC Work e o Access Server, relativa à *persona* e, por fim, relativa ao dispositivo que está a efectuar o pedido. Assim, quando o pedido chega ao Access Server, este verifica que não existe o *cookie* SSO_PT mas existe informação relativa à sessão criada entre o *desktop* PC Work e o Access Server.

Com esta informação, o Access Server obtém a *persona* e o *terminal* para poder aceder à base de dados de suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*, definida na Secção 3.4.2, a fim de adquirir a *Virtual Persona* associada ao *desktop* PC Work. De seguida, o Access Server verifica se já existe algum *cookie* associado à *Virtual Persona* adquirida. Como o utilizador já está autenticado no portal Sapo Mail noutro dispositivo mas a

Virtual Persona é precisamente a mesma, o Access Server conclui tratar-se do mesmo utilizador e desta forma autentica-lo. Tal só acontece porque o Access Server efectua um pedido ao GIP-TIN a fim de receber o *username* que a *Virtual Persona* tem mapeada com o portal Sapo Mail. Posteriormente, o Access Server obtém da base de dados do portal Sapo Mail, o ID do utilizador no respectivo portal. Com todas estas informações, o Access Server cria um novo *cookie* SSO_PT que, depois de registar estar informações nas suas estruturas internas, envia para o *browser* do *desktop* PC Work através do redireccionamento do pedido para o URL envia pela WebGate. Em conjunto com o *cookie* SSO_PT, vão ainda informações que indica que o utilizador já está autenticado, informações relativas ao ID de sessão recebido e ao ID do utilizador. Embora a sessão ainda não esteja autenticada quando o pedido chega ao portal Sapo Mail, a WebGate deste portal verifica que o utilizador já está autenticado e que deve dar acesso aos pedidos vindos desse dispositivo. Posto isto, a WebGate autentica a sessão e envia para o *desktop* PC Work a página solicitada, tal como mostra a Figura 5.6.

Figura 5.6: Página pedida ao portal Sapo Mail através do *desktop* PC Work

5.2.3 Cenário de Teste III

O cenário aqui apresentado começa por ser idêntico ao Cenário de Teste II anteriormente apresentado. Neste cenário de teste, o utilizador efectua *login* na página do portal TMN através do *smartphone* HTC Google Nexus One. Quando o pedido chega ao portal TMN, a WebGate do portal verifica que a sessão estabelecida entre o *smartphone* HTC Google Nexus One e o portal TMN não está ainda autenticada. Então, a WebGate realiza um redireccionamento do pedido para o Access Server, com o ID da sessão criada e o URL do pedido solicitado ao portal.
Quando o pedido chega ao Access Server, este verifica que o *cookie* SSO_PT não está contido no pedido, pelo que a verificação falha. Como também não existe no Access Server qualquer registo do sessão criada entre o *smartphone* HTC Google Nexus One e o Access Server, este último executa um novo redireccionamento do pedido para o URL que lhe foi enviado pela WebGate do portal TMN, com a informação respeitante ao ID de sessão que a mesma WebGate enviou e ao insucesso da autenticação e a consequente necessidade do utilizador apresentar as suas credenciais.
Logo que o pedido chega ao portal TMN, é analisado pela WebGate do portal que verifica que a sessão não está autenticado e que é necessário apresentar ao utilizador a página de *login*, demonstrada na Figura 5.7, onde este pode introduzir as suas credenciais. Depois do utilizador

Figura 5.7: Página de *login* do portal TMN no *smartphone* HTC Google Nexus One

introduzir e submeter as credenciais para o portal TMN, a WebGate do portal confirma que a sessão do pedido não está ainda autentica mas verifica tratar-se de uma tentativa de autenticação das credenciais contidas neste pedido. Pelo que, a WebGate efectua um redireccionamento do

pedido para o Access Server, com as credenciais recebidas, com o ID da sessão criada e com o URL do pedido e também com a informação de que se pretende efectuar a verificação dessas credenciais.

Quando o pedido chega ao Access Server, este verifica que o *cookie* SSO_PT não está presente no pedido, no entanto confirma tratar-se de um pedido de autenticação. Assim, o Access Server verifica na base de dados do portal TMN, as credenciais contidas no pedido. Como as credenciais estão correctas, o Access Server obtém o ID do utilizador no portal TMN. Com a obtenção deste ID, o Access Server cria um *cookie* SSO_PT que é enviado para o *smartphone* HTC Google Nexus One através de um redireccionamento do pedido para o URL recebido anteriormente. Para além do *cookie* SSO_PT é também enviada o ID do utilizador recebido, o ID de sessão contido no pedido recebido anteriormente e a indicação de que a autenticação foi efectuada com correctamente. Antes ainda de efectuar o redireccionamento, o Access Server faz um pedido da *Virtual Persona* que inclui o mapeamento do *username* com o respectivo portal ao GIPTIN e regista todas estas novas informações nas suas estruturas internas.

Ao chegar ao portal TMN, a WebGate deste verifica que a sessão do pedido não está autenticada. Contudo, recebe a informação de que a autenticação do utilizador foi efectuada com correctamente. Isto faz com que a WebGate autentique a sessão e envie para o dispositivo do utilizador, o *smartphone* HTC Google Nexus One, a página solicitada assim como mostra a Figura 5.8. De

Figura 5.8: Página solicitado ao portal TMN no *smartphone* HTC Google Nexus One

seguida e sem efectuar *logout* do portal TMN, o utilizador efectua um pedido de acesso ao portal Sapo Mail a partir do seu *laptop* Apple MacBook. Ao chegar ao portal Sapo Mail, a WebGate verifica que o pedido não tem a sessão activa, pelo que, efectua um redireccionamento do pedido,

com a informação do ID da sessão criada entre o *laptop* Apple MacBook e o portal Sapo Mail e com o URL de pedido recebido, para o Access Server.
Pelo facto do utilizador ainda não ter efectuado nenhuma autenticação num dos portais com o sistema de autenticação proposto, o *browser* do *laptop* Apple MacBook não contém ainda o *cookie* SSO_PT. Esta facto faz com que o pedido efectuado pelo redireccionamento da WebGate do portal Sapo Mail, não inclua o *cookie* SSO_PT. Todavia, o *laptop* Apple MacBook tem instalado e activado a componente Intelligent Authentication, que se encarrega, no momento em que o pedido redireccionado pela WebGate do portal Sapo Mail passa pela seu dispositivo, de enviar ao Access Server a informação relativa ao ID da sessão criada entre o *laptop* Apple MacBook e o Access Server, assim como, a *persona* e *terminal* que estão a efectuar o pedido. Desta forma, o Access Server verifica que não existe o *cookie* SSO_PT mas já existe informação relativa à sessão criada entre o *laptop* Apple MacBook e o Access Server.
Através desta informação, o Access Server obtém a *persona* e o *terminal* relacionado para poder aceder à base de dados de suporte definida na Secção 3.4.2, e adquirir a *Virtual Persona* associada ao *laptop* Apple MacBook. De seguida, o Access Server verifica se já existe algum *cookie* associado à *Virtual Persona* adquirida e como tal se verifica, o Access Server conclui tratar-se do mesmo utilizador. Assim, o Access Server efectua um pedido ao GIPTIN a fim de obter o *username* que a *Virtual Persona* tem mapeada com o portal Sapo Mail.
Posteriormente, o Access Server adquire da base de dados do portal Sapo Mail, o ID do utilizador nesse portal. Com base nestas informações, o Access Server cria um novo *cookie* SSO_PT. Depois de registar estar informações nas estruturas internas, o Access Server efectua um redireccionamento do pedido para o URL enviado pela WebGate, no qual inclui a indicação de que o utilizador já está autenticado, informações relativas ao ID de sessão recebido e ao ID do utilizador e também o *cookie* SSO_PT que será guardado no *browser* do *laptop* Apple MacBook.
Apesar da sessão ainda não estar autenticada, quando o pedido chega ao portal Sapo Mail a WebGate deste verifica que o utilizador já foi autenticado e, desta forma, deve dar acesso aos pedidos vindos do *laptop* Apple MacBook. Por conseguinte, a WebGate autentica a sessão e envia para o *laptop* Apple MacBook a página solicitada, tal como mostra a Figura 5.9. Ainda no *laptop* Apple MacBook e sem efectuar *logout* em qualquer um dos portais anteriormente autenticados, o utilizador efectua agora um pedido de acesso ao portal MEO. Quando o pedido chega ao portal MEO, a WebGate deste portal verifica que ainda a sessão criada entre o *laptop* Apple MacBook e o portal MEO, não está ainda autenticada. Pelo que, a WebGate efectua um redireccionamento do pedido para o Access Server com o ID dessa sessão e com o URL do pedido recebido.
Como o utilizador já se autenticou através do *laptop* Apple MacBook no portal Sapo Mail, este

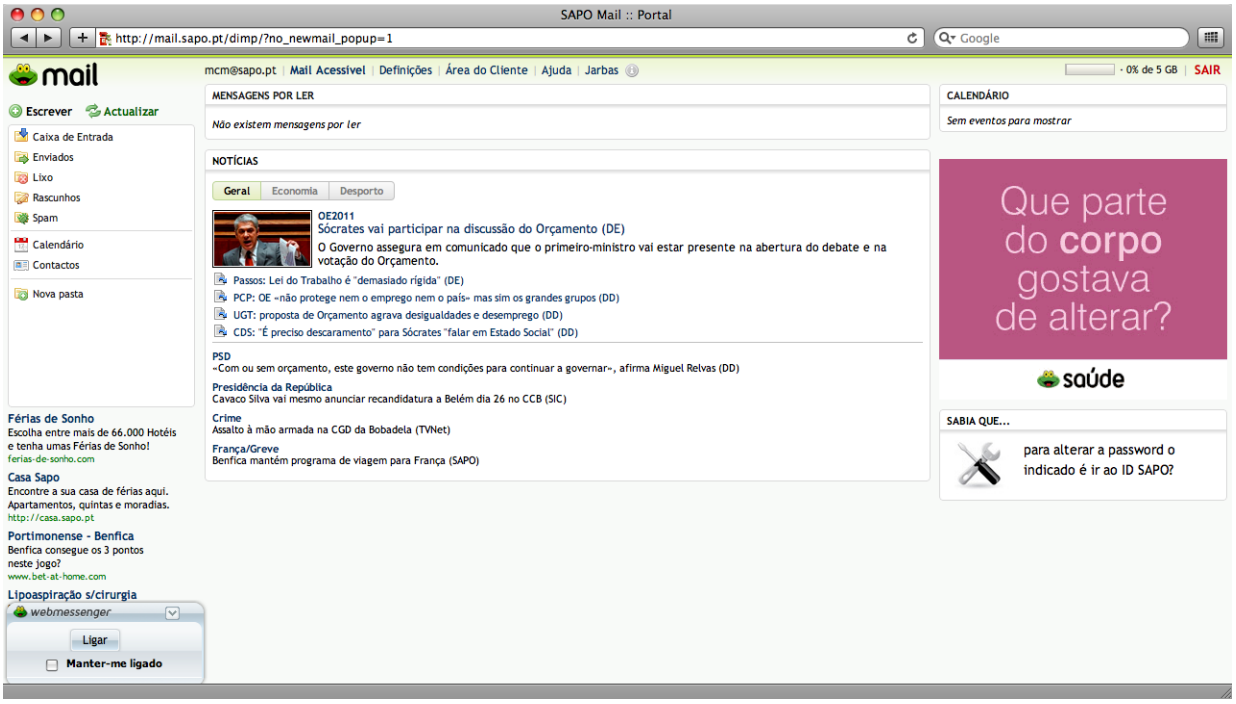

Figura 5.9: Página solicitada ao portal Sapo Mail através do *laptop* Apple MacBook

já contém no seu *browser* o *cookie* SSO_PT. O que faz com que o pedido ao chegar ao Access Server já inclua esse o *cookie*. Assim, o Access Server verifica que o *cookie* SSO_PT é válido e também verifica que este está associado uma *Virtual Persona*. No entanto, como o utilizador ainda não se autenticou no portal MEO, não existe registados no Access Server de qualquer relação entre esta *Virtual Persona* e o portal MEO. Consequentemente, o Access Server efectua um pedido ao GIPTIN para adquirir o *username* mapeado na *Virtual Persona* associada ao *cookie* SSO_PT relativo ao portal MEO.

Depois de adquirido o *username*, o Access Server acede à base de dados do portal MEO, a fim de obter o ID do utilizador correspondente. Posteriormente regista as informações que acabou de adquirir e efectua um redireccionamento para o URL recebido anteriormente, com a informação de que o utilizador já está autenticado, como ID de sessão que recebeu também anteriormente e com o ID do utilizar.

Logo que o pedido redireccionado pelo Access Server chega ao portal MEO, a WebGate do portal verifica que a sessão não está autenticada, todavia confirma que o utilizador já está autenticado e que deve proceder ao envio da página solicitada ao utilizador. Para tal, a WebGate autentica a sessão e apresenta ao utilizador a página, como se pode verificar pela Figura 5.10. Neste momento, o utilizador está autenticado em três portais diferentes (portal TMN, portal Sapo Mail

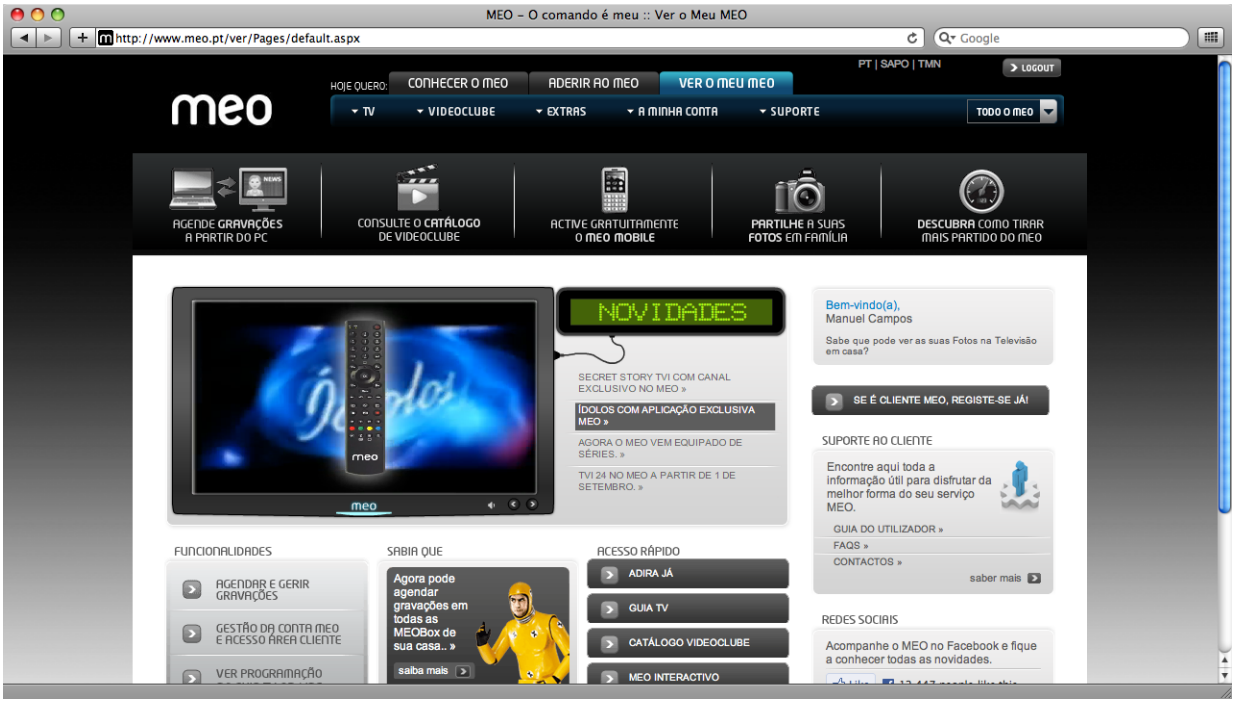

Figura 5.10: Página do portal MEO requerida através do *laptop* Apple MacBook

e portal MEO) e através de dois dispositivos distintos (*smartphone* HTC Google Nexus One e *laptop* Apple MacBook). Desta forma, o utilizador pode navegar em qualquer um desses portais sem necessitar de efectuar qualquer tipo de autenticação explícita, ou seja, pode aceder aos portais sem ter de introduzir novamente as suas credenciais.

Por fim, o utilizador pretende efectuar *logout* de todos os portais autenticados em qualquer um dos dispositivos cuja autenticação já foi efectuada. Para isso, o utilizador efectua *logout* no portal MEO através do *laptop* Apple MacBook. Quando o pedido de *logout* chega ao portal MEO a WebGate do portal verifica que a sessão já está autenticada e ao confirmar tratar-se de um pedido de *logout* válido, efectua um redireccionamento do pedido para o Access Server, com a informação de que o utilizador pretende efectuar *logout* das sessões autenticadas e o URL do pedido recebido.

O pedido quando chega ao Access Server é verificado que contém o *cookie* SSO_PT e que se trata de um pedido de *logout*. Para tal, o Access Server adquire a *Virtual Persona* associada ao *cookie* e através dela obtém os portais e respectivos IDs de sessão, onde esta *Virtual Persona* foi autenticada. Posto isto, o Access Server envia para os portais cuja *Virtual Persona* está autenticada, ou seja, para o portal TMN, o portal Sapo Mail e para o portal MEO, os respectivos IDs de sessão associados. De seguida, o Access Server invalida todos os *cookies* SSO_PT e as

sessões associadas à *Virtual Persona* autenticada. Desta forma, os *cookies* SSO_PT que estão no *smartphone* HTC Google Nexus One e *laptop* Apple MacBook são a partir deste momento inválidos, isto é, quando forem efectuados redireccionamentos dos pedidos para o Access Server e estes contenham algum *cookie* SSO_PT, os *cookies* não serão validados o que faz com que o utilizador tenha de se autenticar novamente. Depois de executar a invalidação dos *cookies* e das sessões, o Access Server efectua um redireccionamento do pedido para o URL que recebeu anteriormente, com a indicação de que já foi efectuado o *logout* de todos os portais.
Quando esta informação chega ao portal MEO, a WebGate do portal verifica que a sessão já não está autenticada e como foi efectuada o *logout* com sucesso deve apresentar ao utilizador a página *default* do portal MEO que neste caso é a página de *login*, como se pode ver na Figura 5.11. Desta forma, o utilizador procedeu ao *logout* de todos os portais que estava autenticado,

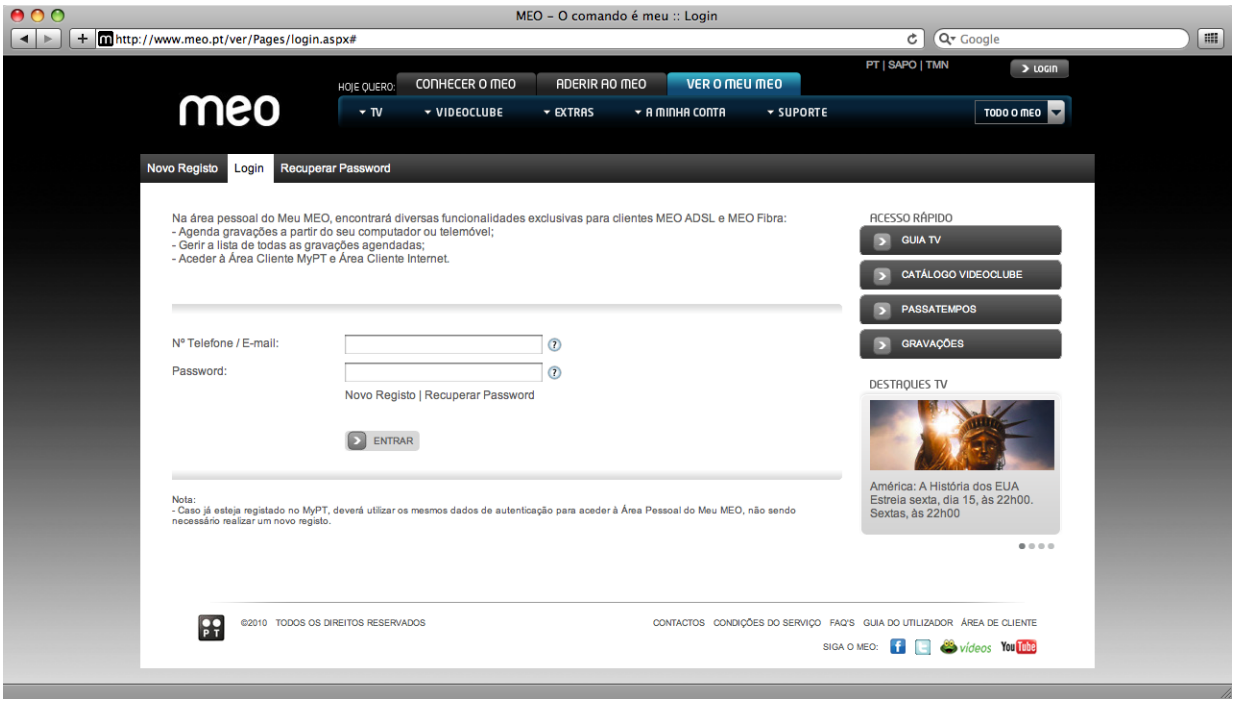

Figura 5.11: Página redireccionada pelo portal MEO depois do ser efectuado o *logout*

independentemente do dispositivo que usou para se autenticar, ou seja, efectuou o SSOff.
Para verificar se de facto o SSOff foi efectuado com sucesso, o utilizador tenta aceder novamente ao portal Sapo Mail através do *laptop* Apple MacBook. Como seria de esperar, o acesso é recusado e é apresentada a página de *login* do portal Sapo Mail. Da mesma forma, o utilizador tenta aceder ao portal da TMN, desta vez a partir do *smartphone* HTC Google Nexus One mas mais, uma vez, o acesso é recusado e é apresentada a página de *login* do portal TMN.

5.2.4 Cenário de Teste IV

Com o último cenário de teste apresentado, pretende mostrar que o gestor de múltiplos dispositivos e de *Virtual Personas* efectua, de facto, as especificações inicialmente propostas. A gestão é efectuado por cada utilizador, através do portal Access Sever que disponibiliza uma interface bastante intuitiva que possibilita ao utilizador uma vista global de todos os seus dispositivos e *Virtual Personas*, bem como os mapeamentos contidos em cada uma das *Virtual Personas* do utilizador. É também apresentado nesta interface, a forma como as *Virtual Personas* estão associadas a cada dispositivo.

Para aceder ao gestor, o utilizador acede à página do portal Access Server e escolhe a aba *manager*. Como o utilizador ainda não está autenticado, é apresentado a página de *login* do portal Access Server. As credencias que o utilizador tem de inserir para ter acesso ao gestor de múltiplos dispositivos e de *Virtual Personas*, são a *persona*, apresentada na Secção 3.3 e o PIN correspondente.

Depois das credenciais serem verificadas e validadas, é apresentado ao utilizador a interface *web* para a gestão dos seus terminais e das suas *Virtual Personas*. Esta interface *web* é apresentada na Figura 5.12. Através desta interface é oferecido ao utilizador, como referido anteriormente,

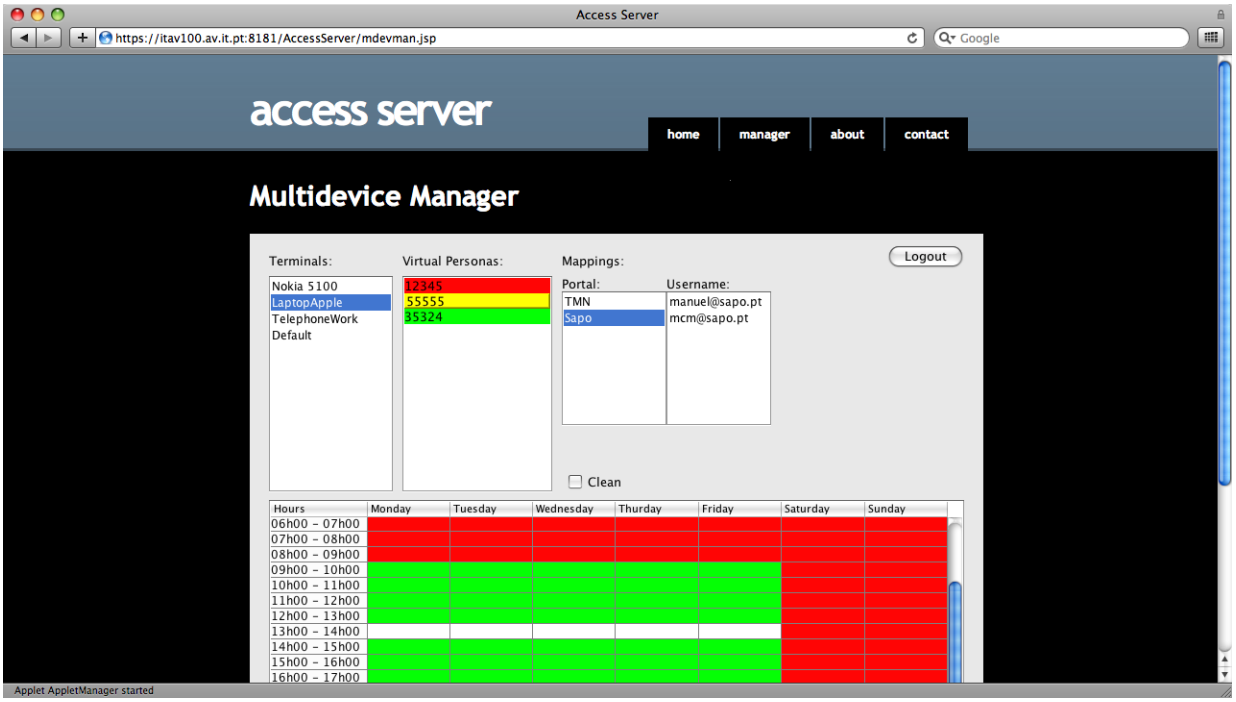

Figura 5.12: Página com a interface *web* do portal Access Server para efectuar a gestão de múltiplos dispositivos e de *Virtual Personas*

uma vista global de todos os seus dispositivos e das suas *Virtual Personas*, entre outras coisa. Especificando um pouco, é apresentada uma lista com todos os dispositivos do utilizador e outra com as *Virtual Personas*. A lista das *Virtual Personas* do utilizador têm cores diferentes, para que seja mais fácil verificar qual é a *Virtual Persona* associada a um determinado dispositivo. Ao seleccionar uma *Virtual Persona*, é imediatamente apresentado ao utilizador, numa lista ao lado, os mapeamentos contidos nessa *Virtual Persona*, ou seja, são apresentados os portais e respectivos *usernames* associados.

Quando o utilizador selecciona um dispositivo na lista de dispositivos, é apresentado através de cores numa tabela em formato de horário, as *Virtual Personas* associadas ao dispositivo seleccionada. Desta forma, o utilizador pode, para cada hora do dia e para cada dia da semana, escolher a *Virtual Persona* que pretende associar ao dispositivo seleccionado. Para isso, basta ter seleccionado o dispositivo pretendido, seleccionar a *Virtual Persona* que pretende associar e de seguida seleccionar na tabela as células correspondentes às horas e dias em que pretende associar a *Virtual Persona* ao dispositivo.

É importante realçar que a lista dos dispositivos do utilizador contém um dispositivo denominado por "Default". Na realidade não se trata de nenhum dispositivo específico ou real mas o utilizador pode associar também as suas *Virtual Persona* a este dispositivo fictício. O utilizador não é obrigado a associar a cada dispositivo *Virtual Personas* para todos os dias e para todas as horas, pelo que se isso acontecer, vão existir horas em que esses determinado dispositivos não têm nenhuma *Virtual Persona* associada. Quando o utilizador efectua um pedido de acesso a um portal e nesse momento não está qualquer *Virtual Persona* associada, o *Virtual Persona* que é usada é a que estiver definida no dispositivo Default.

Depois de efectuar a associação das *Virtual Persona* através da selecção efectuada no tabela, o utilizador pode gravar essas associações, carregando no botão "Update" ou simplesmente cancelar e voltar às associações já efectuadas anteriormente carregando no botão "Cancel". Se o utilizador pretender limpar todas as associações já efectuadas e voltar a criar novas, basta pressionar o botão "CleanAll". No entanto, se desejar apenas apagar algumas associações, basta seleccionar a opção "Clean" seleccionar na tabela as associações que quer apagar.

Quando o utilizador tiver concluído todas as associações pretendidas, pode sair do gestor de múltiplos dispositivos e de *Virtual Personas*, carregando no botão "Logout". Uma vez carregado neste botão, é apresentado de novo a página de autenticação.

5.3 Impacto no Utilizador

A implementação da arquitectura do Terminal Virtual leva a que determinadas tarefas que o utilizador efectua no seu dia-a-dia sejam, no mínimo, alteradas. Actualmente um utilizador para aceder a um determinada portal, cuja autenticação é exigida, introduz as suas credenciais e, se estas forem validadas, efectua *login* no portal. Uma grande parte dos portais que existem hoje na Internet, têm uma forma de autenticação idêntica, usando apenas como credenciais de acesso um *username* e a respectiva *password*.

Devido à facilidade da criação de contas em diversos portais, cada vez mais os utilizador aumentam o número de contas de que são donos. A criação dessas contas o utilizador têm de introduzir, para além de alguns dados pessoais, uma *password*, pois, ao contrário de algumas excepções, os portais não partilham entre si as credencias do utilizador. Um exemplo de uma dessas excepções, são os portais da Google Inc., em que o utilizador depois de criar uma conta num desses portais, apenas necessita autentica-se num para estar autenticado nos restantes. No entanto, esta solução apenas funciona no mesmo dispositivo e em portais que estejam directamente ligados à Google Inc.

Como a implementação da arquitectura do Terminal Virtual nos portais já existentes é relativamente fácil, a mobilidade de identidade entre os diversos portais é uma mais valia para o utilizador, que deixa de necessitar de um número enorme de *passwords* para aceder aos diversos portais. Assim, o utilizador apenas tem de executar uma única autenticação para poder aceder a qualquer portal que esteja previamente registado, sem que seja necessário voltar a autenticar-se, independentemente do dispositivo que está a usar.

Para que o utilizador possa usufruir destas vantagens, necessita apenas de criar *Virtual Personas* e associar a cada um dos seus dispositivo essas *Virtual Personas*. Com a criação das *Virtual Personas*, o utilizador pode fazer uma separação de todas as suas contas, isto é, o utilizador pode criar *Virtual Personas* que representam grupos, como por exemplo, a *Virtual Persona* de trabalho, onde estão mapeadas todas as contas relacionadas com o seu trabalho, com a sua vida profissional, a *Virtual Persona* governamental, onde estão mapeadas as contas que dizem respeito ao portal das Finanças, ao portal da Segurança Social, entre outra.

As associações das *Virtual Personas* aos dispositivos do utilizador, faz com este posso ter diferentes contas no mesmo portal mas, dependendo do dispositivo, do dia de semana e da hora do dia, usar a que mais lhe convém. Através destes mecanismos, aumentam o nível de segurança que o utilizador tem actualmente, devido ao facto, do utilizador apenas ter de introduzir uma única *password* de uma das contas mapeada na *Virtual Persona* que este pretende usar, para aceder a

qualquer outra conta mapeada na mesma *Virtual Persona*, reduzindo assim, significativamente, o número de *passwords* do utilizador que circulam pela Internet.
O mecanismo de autenticação implementada pela arquitectura do Terminal Virtual, introduz algum atraso comparativo aos mecanismos utilizados pelo portais que efectuam a verificação das credenciais directamente nas suas bases de dados. Apesar de todas as verificações que são feitas pelas diversas componentes que constituem o Terminal Virtual, como os redireccionamentos das WebGates para o Access Server e vice-versa e verificações de sessões e *cookies*, o atraso introduzido não é significativo. A rapidez com que todas as verificações e redireccionamentos são efectuados, faz com que do posto de vista do utilizador não seja notado qualquer latência.
As outras funcionalidades que a arquitectura do Terminal Virtual propõe são a mobilidade de sessão e a coordenação dos dispositivos do utilizador. Apesar da sua implementação não ter sido efectuada, com estas funcionalidades, o utilizador pode a qualquer momento transferir uma sessão de um dispositivo para o outro. Assim o utilizador pode transferir uma chamada de voz do seu telemóvel para o seu computador, sem que para isso tenha de efectuar uma nova chamada. Através da coordenação dos dispositivos, e voltando ao exemplo do chamada de voz, quando é efectuada uma chamada de voz para o utilizador, este recebe a chamada no dispositivo que estiver em melhores condições de estabelecer a comunicação.

5.4 Sumário

Neste capítulo foram apresentados os testes funcionais efectuados à implementação da arquitectura do Terminal Virtual, elaborando cenários de teste com base os cenários de utilização apresentados no Capítulo 4. Para efectuar estes testes foram escolhidos diferentes dispositivos reais, por forma a mostrar que a solução elaborado cumpre as especificações inicialmente propostas, independentemente do tipo de dispositivo e da plataforma que estes utilizam.
Foi também abordado neste capítulo o impacto que esta nova solução produz na experiência de utilização.

Capítulo 6

Conclusões

Neste capítulo, é efectuada uma síntese do trabalho realizado e das principais contribuições que este trabalho acrescenta, bem como uma análise crítica ao trabalho que foi efectuado. Por fim, é apresentado o trabalho futuro que poderá ser efectuado, partindo das contribuições que a elaboração deste trabalho apresenta.

6.1 Introdução

O desenvolvimento deste dissertação centrou-se na gestão e manutenção dos dispositivos que um utilizador possui. De forma a proporcionar uma melhor gestão do conjunto dos dispositivos de cada utilizador, tornou-se necessária a implementação da mobilidade de terminal, mobilidade de identidade e mobilidade de sessão. Paralelamente, verificou-se que estes tipos de mobilidade levantam questões relacionadas com a identidade e privacidade dos utilizadores, o que levou à necessidade da criação de identidades parciais. Este problema foi resolvido, através da criação das *Virtual Personas*. Em cenários futuros, a introdução do conceito de identidades virtuais, que permite seleccionar e criar diferentes identidades digitais para diferentes contextos, vai aumentar a complexidade para o utilizador em termos de gestão dos seus dispositivos e identidades. Partindo do princípio de que nem todas as identidades podem ser usadas em todos os dispositivos, a usabilidade para o utilizador diminui. De forma a proporcionar uma gestão eficaz de todos os dispositivos de que um utilizador é dono, procedeu-se à elaboração e posterior implementação da arquitectura do Terminal Virtual que proporciona ao utilizador uma nova forma de gerir, facilmente, todos os seus dispositivos físicos.

6.2 Trabalho Realizado

Uma vez que não existe hoje em dia uma forma simples e eficaz dos utilizadores gerirem os seus dispositivos, foi estudada e elaborada a arquitectura do Terminal Virtual, com o intuito de oferecer ao utilizador uma forma simplificada de gerir os seus dispositivos e identidades e também disponibilizar uma visão global de todos os esses dispositivos e identidades. Para tal, foi estudado o conceito de mobilidade, no qual estão contidas a mobilidade de objecto e a mobilidade de serviço. Apesar destes dois tipos de mobilidade serem importantes, neste trabalho foi dado mais ênfase à mobilidade de objecto. Este tipo de mobilidade permite que o utilizador use um serviço enquanto está em movimento através da mobilidade de terminal, permite que o utilizador use a mesma identidade, independentemente do dispositivo que está a utilizar através da mobilidade de identidade e, por fim, permite ao utilizador transferir uma sessão de um dispositivo para outro através da mobilidade de sessão.

Tendo como objectivo a implementação da mobilidade de sessão, mobilidade de terminal e mobilidade de identidade, foram estudadas vários tecnologias. Através do protocolo SIP pode ser proporcionado a mobilidade de sessão, pois o SIP é um protocolo de controlo da camada da aplicação que pode, por exemplo, estabelecer, modificar e terminar sessões multimédia. Já o protocolo HIP é um protocolo que tem por base a criação de uma nova camada na pilha protocolar entre a camada de transporte e a camada de rede. Desta forma, é fornecido um método de separar o papel de identificação do papel de localização, que hoje são desempenhados os dois pelo endereço IP. Através desta separação o protocolo HIP proporciona a mobilidade de terminal. Para efectuar a mobilidade de terminal foi também estudado o protocolo PMIPv6 que é um protocolo de gestão de mobilidade baseado na rede. Este protocolo deixa a responsabilidade da gestão de mobilidade para a rede.

Por forma a proporcionar a mobilidade de identidade, foi estudado a tecnologia SAML que é uma estrutura baseada em XML para comunicar a autenticação, autorização e atributos de um utilizador, que permite que as entidades de negócios façam afirmações sobre a identidade, atributos e autorizações de um utilizador para outra entidade. Para proporcionar o mesmo tipo de mobilidade foi também estudado a tecnologia OpenID. Esta tecnologia permite ao utilizador usar uma conta já existente para entrar em múltiplos portais, sem que seja necessário criar novas *passwords*. Por fim, ainda para a mobilidade de identidade foi estudado o protocolo OAuth é um protocolo aberto que permite o acesso autorizado a aplicações através da utilização de uma API segura.

A escolha destes protocolos e tecnologia foi efectuada com base no estudo de uma grande va-

riedade de abordagens tecnológicas que são hoje conhecidas. No entanto, cedo se concluiu que seriam estes protocolos e tecnologias apresentados que mais contribuiriam para uma melhor elaboração da arquitectura do Terminal Virtual. Apesar de se ter estudado estas tecnologias, não foi utilizado nenhuma deles em particular, contudo este estudo contribuiu em muito com os diversos princípios e especificações que cada tecnologia oferece.

Uma vez que os tipos de mobilidade aqui referidos levanta questões relacionadas com a identidade e privacidade dos utilizadores, foram então estudadas essas questões, fazendo com que a elaboração da arquitectura do Terminal Virtual contempla-se uma forma do utilizador gerir eficazmente as suas identidades, proporcionando assim segurança e privacidade das identidades do utilizador. Tal foi conseguido, disponibilizando ao utilizador as *Virtual Personas* permitindo assim que este efectue uma separação eficaz das suas identidades.

Finalmente foi abordado conceito do Terminal Virtual proposto por Marc Barisch, onde foi descrito o Terminal Virtual, quais os papeis que este tem e qual o contexto dos dispositivos dentro do Terminal Virtual e o conceito de sessão. Esta abordagem constituiu a base para a elaboração a arquitectura do Terminal Virtual.

A elaboração da arquitectura do Terminal Virtual foi efectuada em três fases. Inicialmente, a arquitectura elaborada era muito semelhante à arquitectura proposta por Marc Barisch, era uma arquitectura distribuída em que qualquer um dos dispositivos do utilizador poderia ser o Access Server. Esta arquitectura elaborada apenas tinha mais uma componente inserida em cada dispositivo, a componente de gestão de mobilidade de sessão. Por forma a simplificar as componentes que ficariam nos dispositivos do utilizador, a arquitectura elabora foi modificada, tornando-se numa arquitectura centrada no Access Server, que passou a ser um servidor independente, ou seja, este servidor deixou de ser um dispositivo do utilizador. Assim, apenas seria necessário a inclusão da componente Intelligent Authentication nos dispositivos do utilizador, que permite que estes usem os dispositivos facultando a sua identidade ou simplesmente usando-os de forma anónima. Foi também nesta fase que se introduziu na arquitectura o gestor de identidades da PT Inovação, para fazer uso da informação já contemplada por este gestor. Finalmente foi elaborada a arquitectura do Terminal Virtual final, enriquecendo a elaborada anteriormente, na qual foi introduzida uma nova componente intitulada de WebGate. Esta nova componente ficará instalada nos servidores dos portais onde a arquitectura for implementada.

Depois da elaboração da arquitectura do Terminal Virtual e efectuada a integração deste arquitectura com o GIPTIN, foram também criados modelos de dados para suporte da arquitectura proposta e, finalmente, foi efectuada a implementação da arquitectura do Terminal Virtual final. Na implementação desta arquitectura, o gestor de coordenação dos dispositivos virtuais e o ges-

tor de mobilidade de sessão, componentes que fazem parte do Access Server. De notar que a implementação da arquitectura do Terminal Virtual começou por ser realizada recorrendo ao uso de certificados digitais para a identificação do utilizador. Este procedimento usava apenas o certificado digital fornecido ao utilizador para colocar nos seus dispositivo e sempre que este fazia um pedido o certificado era enviado. No entanto, essa abordagem foi posta de parte pelo facto de não oferecer ao utilizador uma forma simples de controlar a sua identificação, não garantindo assim, por exemplo, o anonimato.
Terminada a implementação foram apresentados cenários de utilização expectáveis, para assim mostrar mais concretamente o funcionamento da arquitectura do Terminal Virtual, bem como as comunicações e iterações que são efectuadas entre as diversas componentes que fazem parte da arquitectura do Terminal Virtual.
Por fim, e partindo dos cenários de utilização, foram apresentados os resultados obtidos sob a forma de testes funcionais e de uma abordagem do impacto que esta nova solução induz no utilizador. Através dos testes funcionais, foi possível verificar que as especificações inicialmente estipuladas, foram cumpridas. Como se pode constatar pela realização dos diversos testes, qualquer um dos cenários é viável. A funcionalidade e viabilidade dos diversos cenários apresentados foi assim comprovada, sendo constatado que toda a arquitectura comportou-se como seria expectável. É também importante realçar que através dos testes efectuados foi possível verificar que a arquitectura cumpre as suas especificações previstas, independentemente do dispositivo que está a ser usado.
Com a apresentação do impacto no utilizador que esta solução produz, foi verificado que tal impacto é mínimo, uma vez que toda a arquitectura se comporta de modo a que as verificações e registos sejam efectuados de uma forma rápida e eficaz. Desta forma o utilizador apenas necessita fazer um registo dos seus dispositivos, efectuar a criação de *Virtual Personas* à sua medida e, por fim, associar a cada dispositivo as *Virtual Personas* que desejar.

6.3 Trabalho Futuro

Dada a dimensão que a implementação da arquitectura do Terminal Virtual tem, existem alguns pontos que se pode ter em consideração para trabalho futuro e que contribuem para o enriquecimento da solução aqui apresentada, nomeadamente:

- Implementação da componente de gestão da coordenação dos dispositivos virtuais. Com esta componente, seria oferecido ao utilizador uma forma deste usufruir sempre do melhor

serviço, no melhor dispositivo. Desta forma os dispositivos seriam utilizador de forma ainda mais eficaz.

- Implementação da componente de gestão da mobilidade de sessão. Através desta componente, o utilizador passaria a poder efectuar a transferência de uma sessão de um dispositivo para outro, sem perder o estado actual e sem quebras de serviço.

Através destes pontos, é oferecido ao utilizador uma solução completa, capaz de responder eficazmente às necessidades e desafios actuais e futuros que possa vir a enfrentar.

Bibliografia

[1] Sebastian Clauß, Dogan Kesdogan, and Tobias Kölsch. Privacy enhancing identity management: protection against re-identification and profiling. In *Digital Identity Management*, pages 84–93. ACM, 2005.

[2] Ricardo Azevedo, Stefaan Grégoir, Peter Scholta, Thorsten Raatz, Marc Barisch, Alfredo Matos, Alejandro Mendez, Ronald Marx, Hariharan Rajasekaran, and João Girão. Swift deliverable 403: Swift mobility architecture, 2009.

[3] Anneke G. Kleppe, Jos Warmer, and Wim Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

[4] Pt inovação. http://www.ptinovacao.pt.

[5] ITU-T. *ITU-T Recommendation Q.1706/Y.2801, Mobility management requirements for NGN*, November 2006.

[6] Marc Barisch, Jochen Kögel, and Sebastian Meier. A flexible framework for complete session mobility and its implementation. Barcelona, September 7-9 2009. EUNICE.

[7] Henning Schulzrinne and Elin Wedlund. *Application-Layer Mobility Using SIP*, 2000.

[8] Eranga Perera, Vijay Sivaraman, and Aruna Seneviratne. Survey on network mobility support. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(2):7–19, April 2004.

[9] J. Rosenberg, Henning Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: session initiation protocol. RFC 3261, Internet Engineering Task Force, June 2002.

[10] M. Handley, V. Jacobson, and C. Perkins. SDP: Session description protocol. RFC 4566, Internet Engineering Task Force, July 2006.

[11] J. Postel. Internet protocol. RFC 791, Internet Engineering Task Force, September 1981.

[12] J. Postel. Domain name system structure and delegation. RFC 1591, Internet Engineering Task Force, March 1994.

[13] R. Moskowitz and P. Nikander. Host identity protocol (hip) architecture. RFC 4423, Internet Engineering Task Force, May 2006.

[14] C. Kaufman, P. Hoffman, Y. Nir, and P. Eronen. Internet key exchange protocol version 2 (IKEv2). RFC 5996, Internet Engineering Task Force, September 2010.

[15] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host identity protocol. RFC 5201, Internet Engineering Task Force, April 2008.

[16] Andrei Gurtov. *Host Identity Protocol (HIP): Towards the Secure Mobile Internet*. Wiley, June 2008.

[17] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil. Proxy mobile ipv6. RFC 5213, Internet Engineering Task Force, August 2008.

[18] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775, Internet Engineering Task Force, June 2004.

[19] Oasis security services (saml) tc. http://www.oasis-open.org.

[20] Saml specifications. http://saml.xml.org/saml-specifications.

[21] Openid. http://openid.net/.

[22] Oauth. http://oauth.net.

[23] Mark Wuthnow, Jerry Shih, and Matthew Stafford. *IMS: A New Model for Blending Applications*. Auerbach, July 2009.

[24] J. Peterson. A privacy mechanism for the session initiation protocol (SIP). RFC 3323, Internet Engineering Task Force, November 2002.

[25] Gonzalo Camarillo and Miguel A. García-Martín. *The 3G IP Multimedia Subsystem (IMS): Merging the Internet and the Cellular Worlds*. Wiley, August 2004.

[26] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology. Technical report, February 2008. Version 0.31.

[27] Marc Barisch and Alfredo Matos. *Integrating User Identity Management Systems with the Host Identity Protocol*. ISCC, Sousse, Tunisia, July 2009.

[28] Antonio F. Gómez-Skarmeta, Gabriel López, Óscar Cánovas, Alejandro Pérez, and Elena María Torroglosa. Swift deliverable 302: Specification of general identity-centric security model that supports user control of privacy, 2009.

[29] M. Brown and R. Housley. Transport layer security (TLS) authorization extensions. RFC 5878, Internet Engineering Task Force, May 2010.

[30] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – HTTP/1.1. RFC 2616, Internet Engineering Task Force, June 1999.

[31] E. Rescorla. Http over tls. RFC 2818, Internet Engineering Task Force, May 2000.

[32] T. Berners-Lee, L. Masinter, and M. McCahill. Uniform resource locators (URL). RFC 1738, Internet Engineering Task Force, December 1994.

[33] J. Schaad and R. Housley. Advanced encryption standard (AES) key wrap algorithm. RFC 3394, Internet Engineering Task Force, September 2002.

[34] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2005.

[35] Tmn. http://www.tmn.pt.

[36] Meo. http://www.meo.pt.

[37] Sapo mail. http://mail.sapo.pt.

Apêndice A

Modelação UML

Diagramas de Modelo Relacional

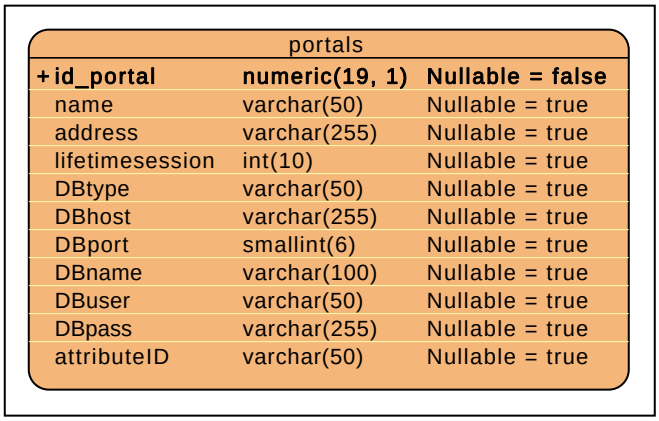

Figura A.1: Modelo de dados para suporte da informação relativa aos portais

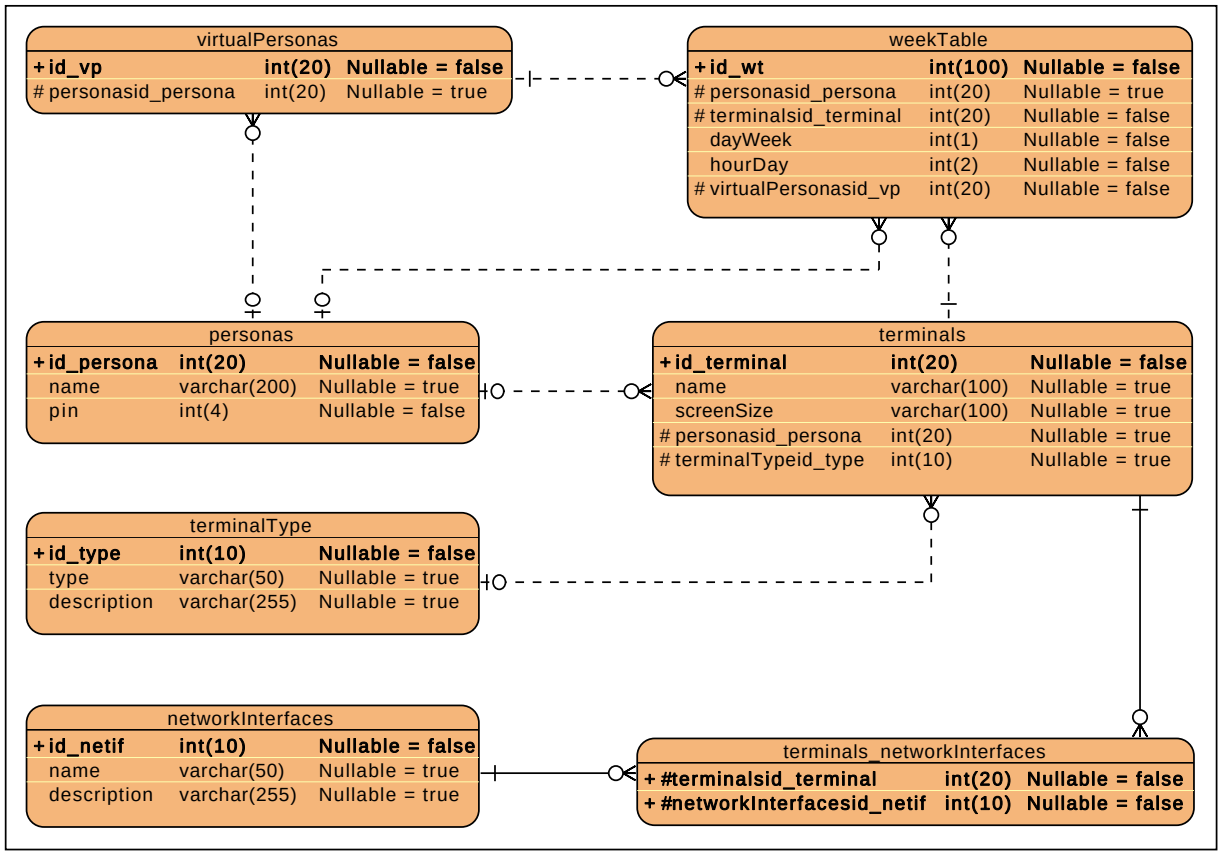

Figura A.2: Modelo de dados para suporte da informação de gestão dos dispositivos do utilizador e das suas *Virtual Personas*

Diagramas de Classe

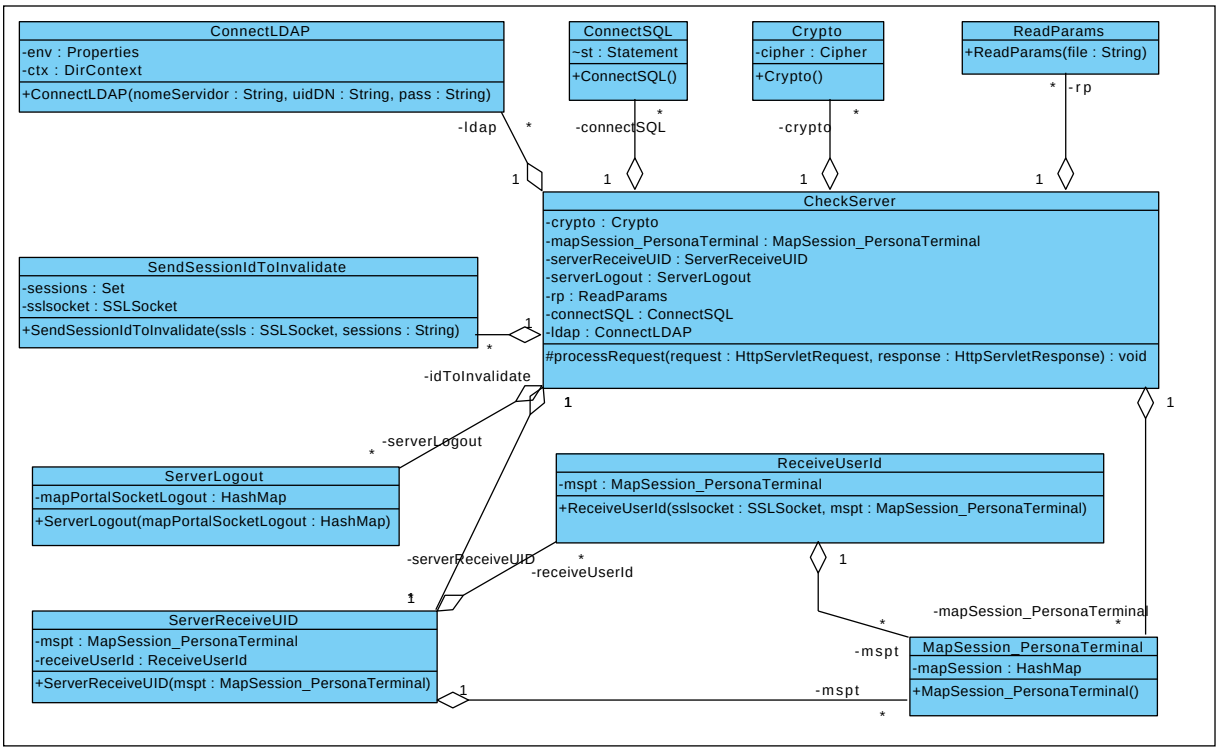

Figura A.3: Diagrama de Classes do Access Server

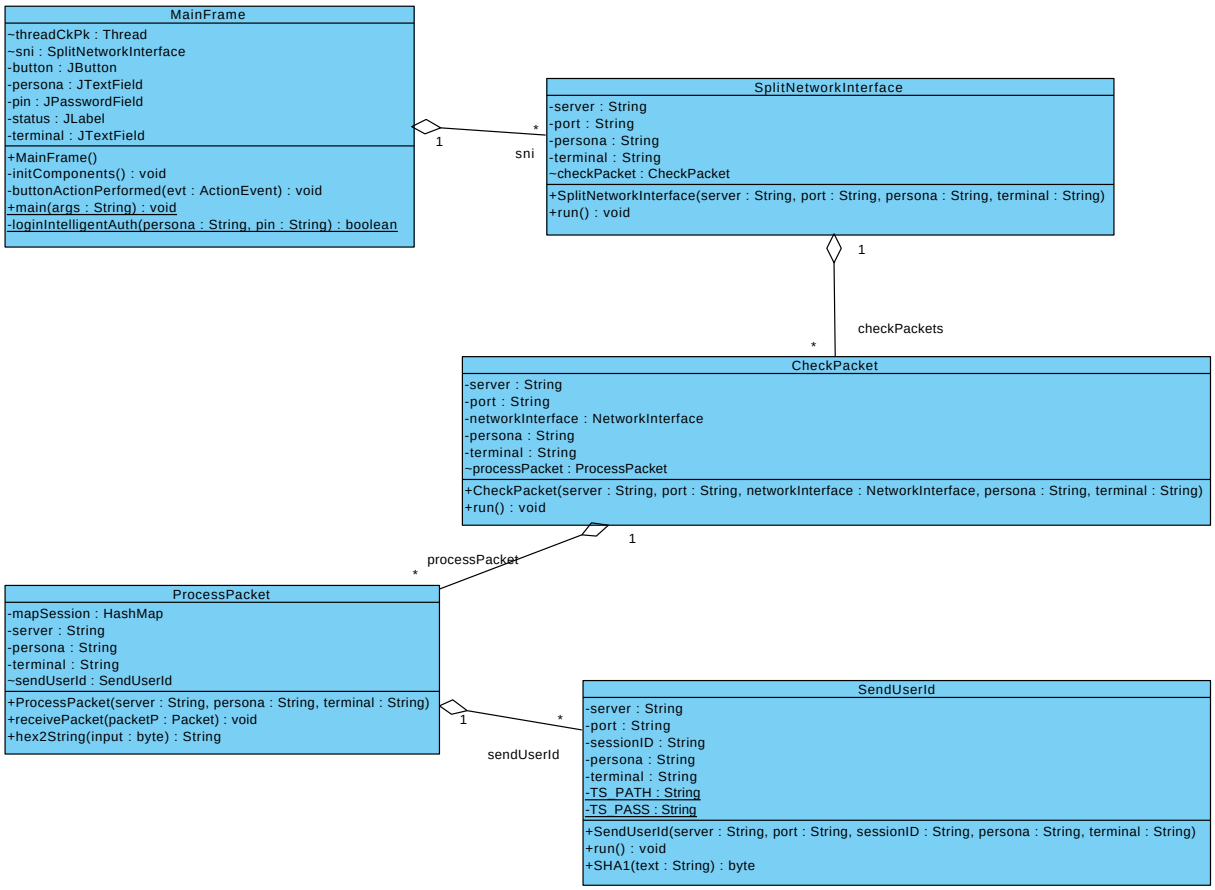

Figura A.4: Diagrama de Classes do Intelligent Authentication

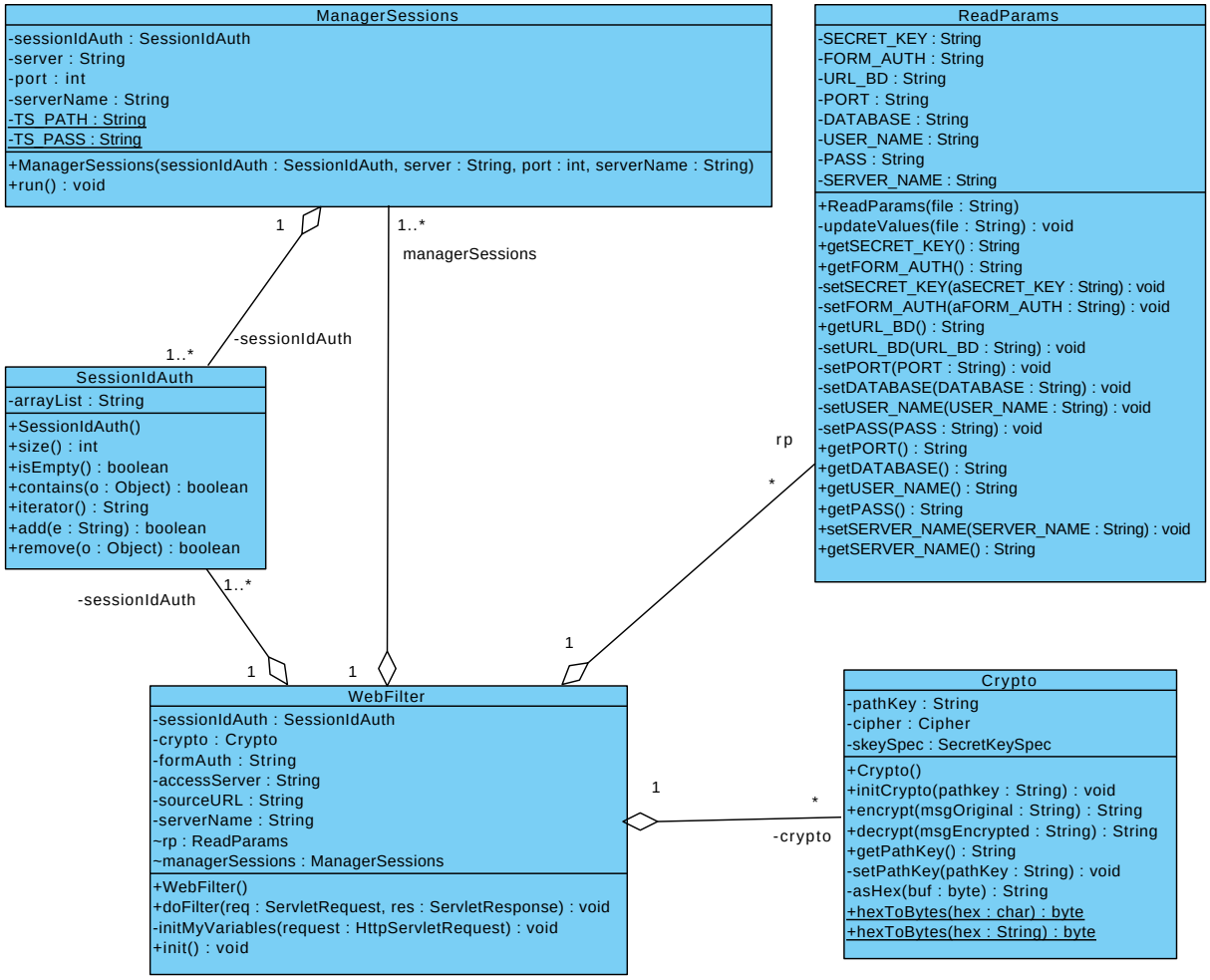

Figura A.5: Diagrama de Classes da WebGate

Diagramas de Sequência

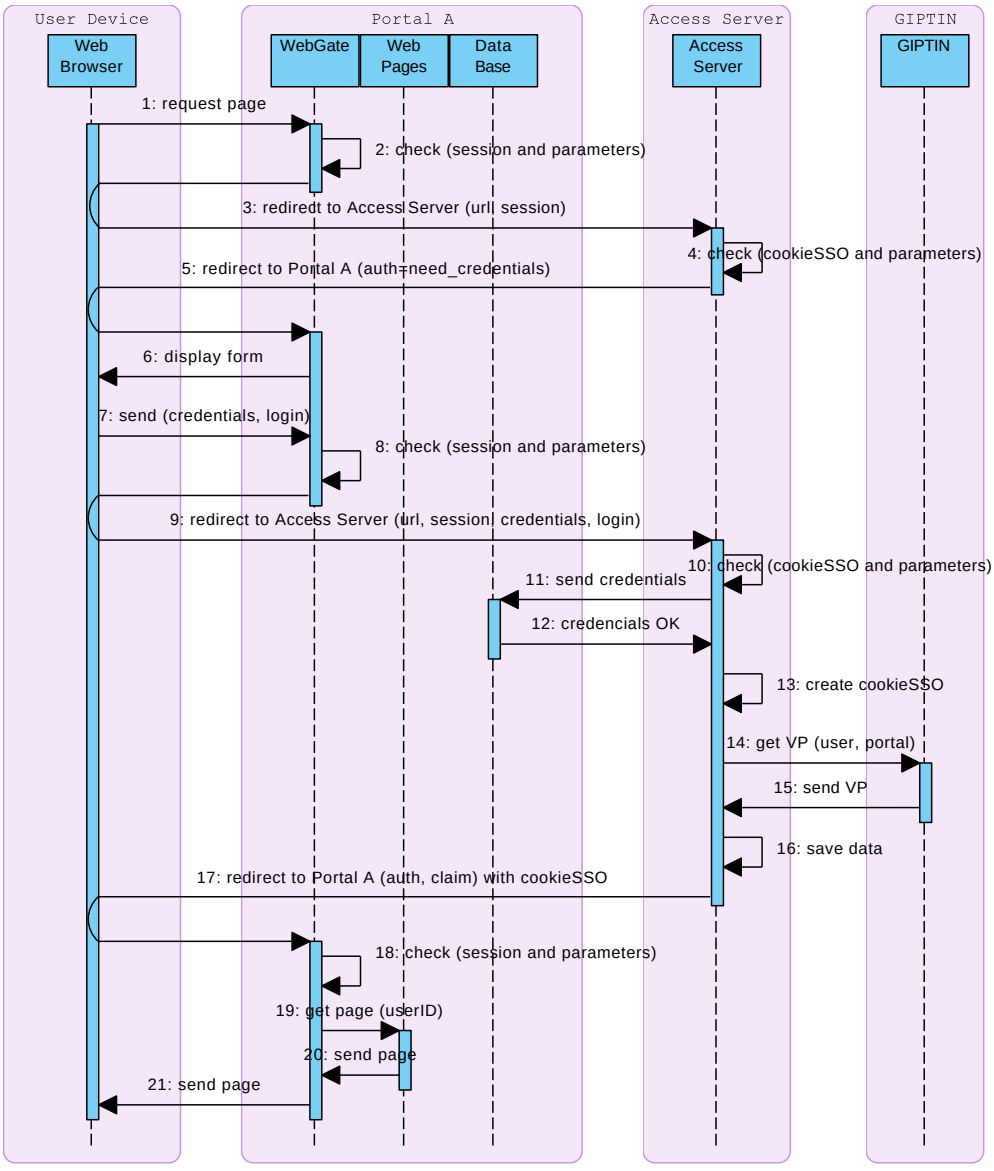

Figura A.6: Diagrama de Sequência da autenticação

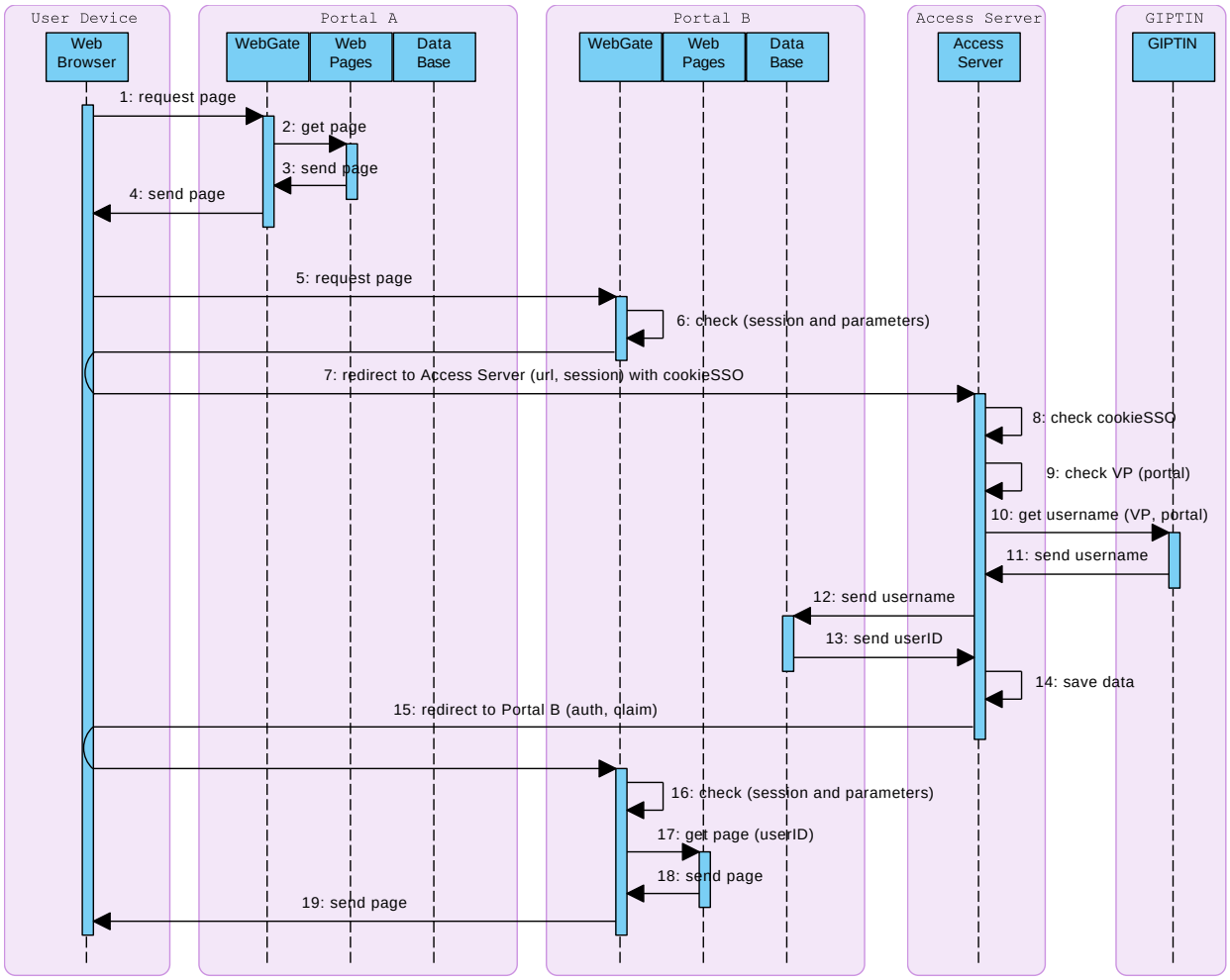

Figura A.7: Diagrama de Sequência da autenticação em dois portais

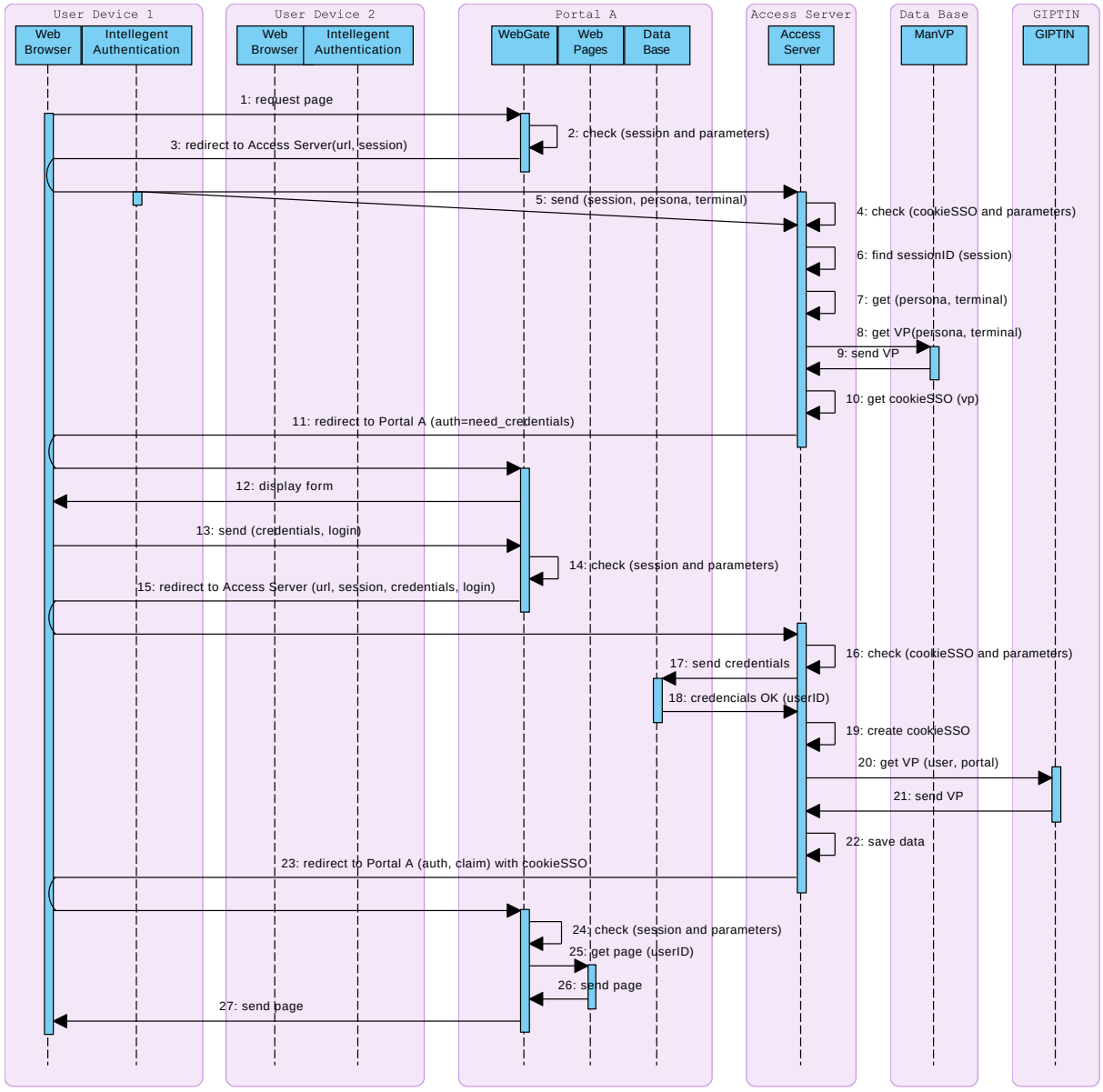

Figura A.8: Diagrama de Sequência da autenticação em dois dispositivos diferentes (1)

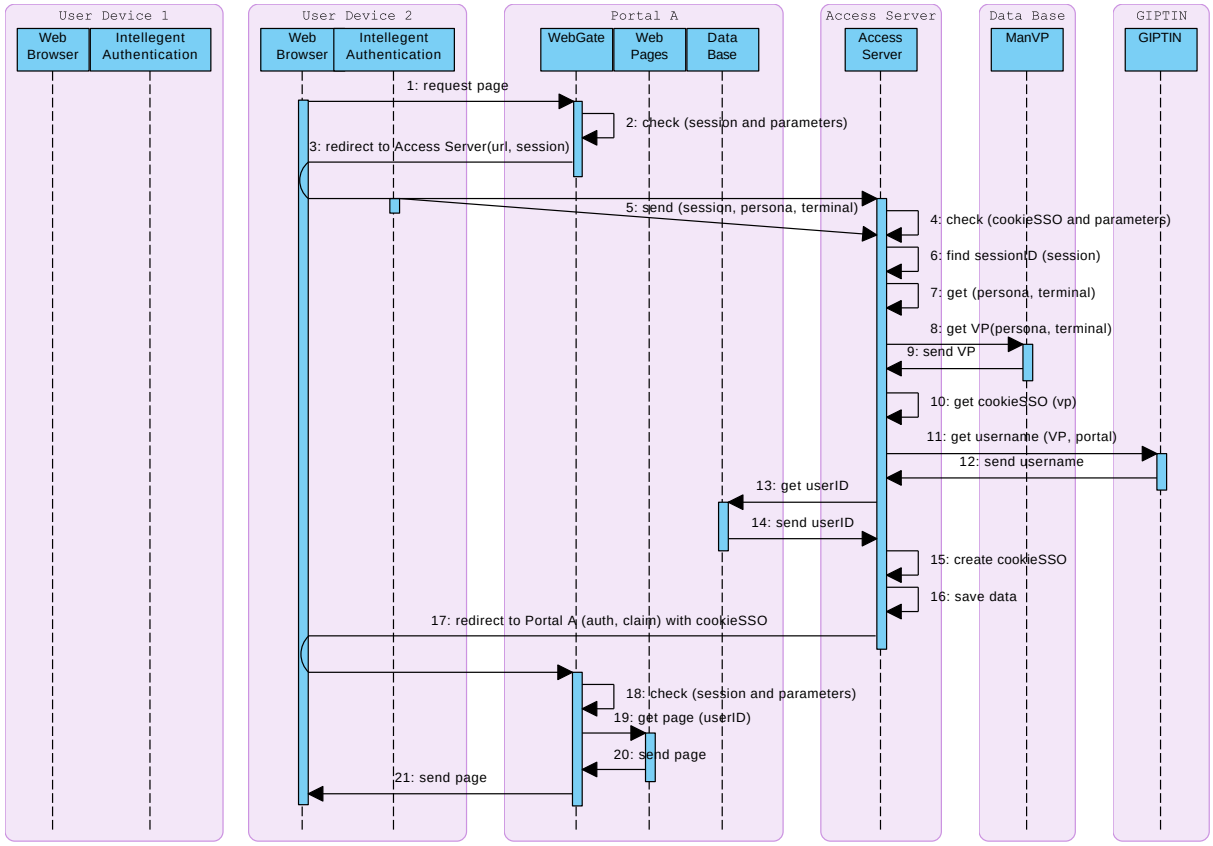

Figura A.9: Diagrama de Sequência da autenticação em dois dispositivos diferentes (2)

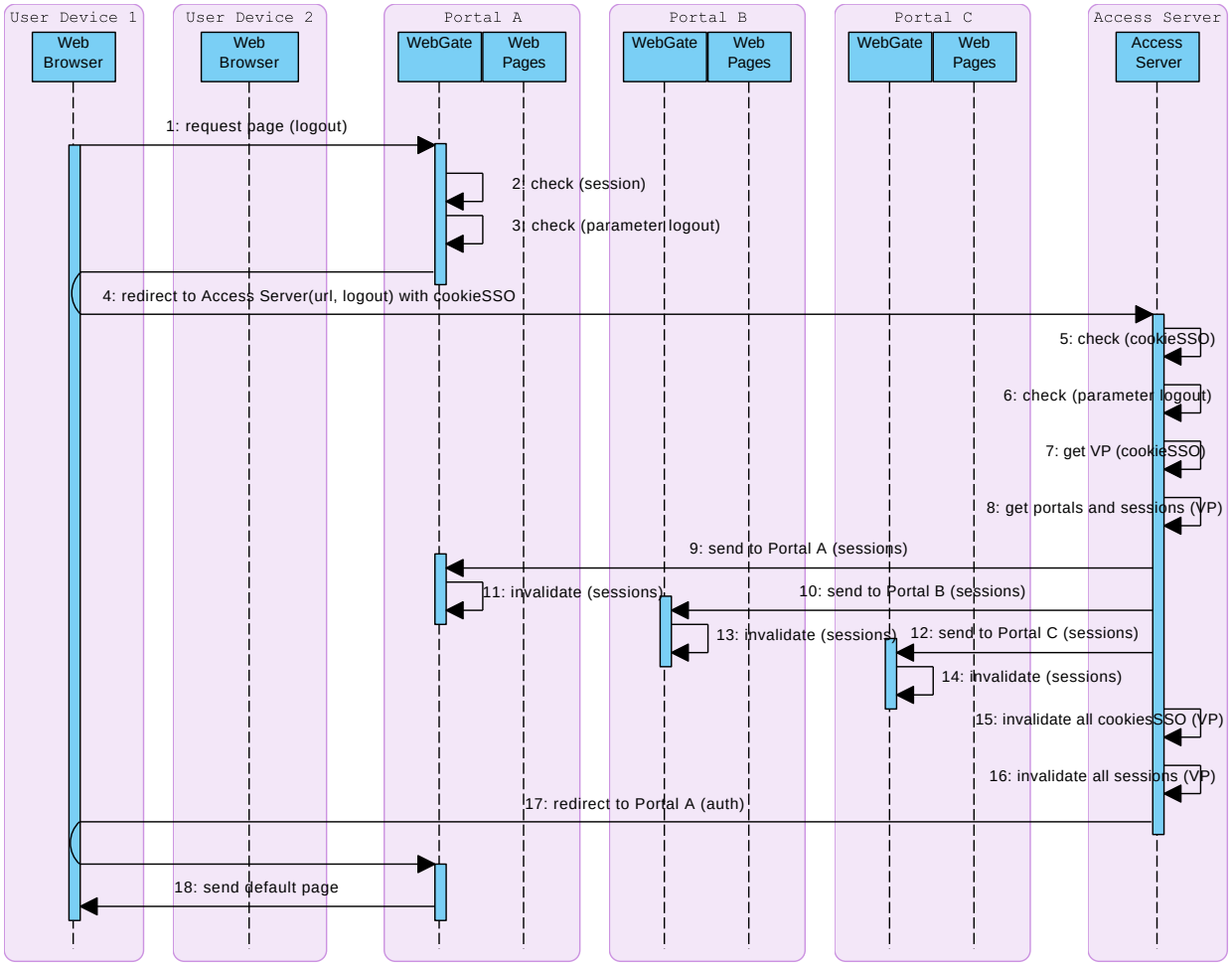

Figura A.10: Diagrama de Sequência para efectuar o SSOff