UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

NÉLISSON ADRIEL DA SILVA CAVALHEIRO

SharpGym: Integração mobile x cloud, aplicada ao treinamento personalizado

Trabalho de Graduação.

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, janeiro de 2013.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por ser o guia nessa jornada do início ao fim e por conceder-me vida e saúde em abundância.

A minha família que sempre me apoiou nos momentos que foram necessários. Especialmente aos meus pais que se dedicaram o quanto possível para que eu pudesse realizar o sonho de me formar em um curso superior. Pela compreensão e ajuda nos momentos fáceis e difíceis nesses anos de faculdade.

Ao Prof. Cláudio Geyer pela orientação e revisão nesse trabalho.

A Gisele Fredes da Silveira pelo amor, paciência, dedicação e companhia para me ajudar a enxergar as coisas lindas da vida. Obrigado por sempre estar disposta a me ouvir e me ajudar.

Por fim, agradeço a UFRGS pelo ensino gratuito e de qualidade que ainda vai me render muitas alegrias.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
AVC	Acidente Vascular Cerebral
BLL	Business Logic Layer
BLOB	Binary Large Object
CEO	Chief Executive Officer
CES	Consumer Electronics Show
CGI	Computer-generated Imagery
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CRUD	Create, Read, Update, Delete
CSS	Cascading Style Sheets
DCOM	Distributed Component Object Model
EJB	Enterprise JavaBeans
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IMC	Índice de Massa Corporal
IDE	Integrated Development Environment
IHRSA	International Health, Racquet & Sportsclub Association
MIT	Massachusetts Institute of Technology
MSDN	Microsoft Developer Network
MS-DOS	Microsoft Disk Operating System
MSMQ	Message Queuing
NAT	Network Address Translation
NPD	National Purchase Diary
OASIS	Organization for the Advancement of Structured Information Standards

PaaS	Platform as a Service
PC	Personal Computer
PHP	Hypertext Preprocessor
POC	Programação Orientada a Componentes
POO	Programação Orientada a Objetos
REST	Representational State Transfer
SaaS	Software as a Service
SDK	Software Development Kit
SGBD	Sistema Gerenciador de Banco de Dados
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SQL	Structured Query Language
TCP	Transmission Control Protocol
TFS	Team Foundation Server
TI	Tecnologia da Informação
UI	User Interface
URI	Uniform Resource Identifier
WCF	Windows Communication Foundation
WSDL	Web Service Definition Language
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O SharpGym é um sistema de controle a ser utilizado por profissionais do treinamento personalizado (personal trainer) para organizar informações referentes ao treino de seus alunos: anamnese (entrevista inicial), cadastro, avaliação física, registro diário de atividades e relatório de desempenho. Através dele, o professor tem acesso a todas essas informações de forma rápida e fácil a partir de seu dispositivo móvel.

O objetivo deste projeto é proporcionar ao professor um atendimento flexível e específico para cada aluno, atendendo às necessidades individuais de cada treino. Desse modo, o professor pode fazer observações durante as atividades físicas para potencializar os resultados.

O sistema será desenvolvido utilizando a plataforma .Net da Microsoft, contando com um módulo web hospedado na Plataforma Windows Azure, o qual promove alta disponibilidade e confiabilidade para os serviços, e um módulo móvel, usando o sistema operacional Windows 8 que proporciona agilidade e mobilidade para o profissional do treinamento personalizado. A camada de acesso entre a nuvem e os dispositivos se dará através de serviços implementados em WCF, que é a plataforma para desenvolver aplicações orientadas a serviços da Microsoft.

Palavras-Chave: Computação em Nuvem, Windows 8, Arquitetura Orientada a Serviços, Treinamento Personalizado.

SharpGym: mobile x cloud integration, applying to personal training

ABSTRACT

The SharpGym is a control system for use by personal trainer to organize information relating to the training of their students: initial interview, registration, physical evaluation, daily log of activities and performance report. Through him, the teacher has access to all this information quickly and easily from your mobile device.

The objective of this project is to provide the teacher a flexible and specific care for each student, meeting the individual needs of each workout. Thus, the teacher can make observations during physical activities to enhance the results.

The system will be developed using the Microsoft .Net Framework, with a web module hosted in Windows Azure Platform that promotes high availability and reliability for services, and a mobile module, using the Windows 8 operating system that provides agility and mobility for personal trainer. The access layer between the Cloud and the devices will be through service implemented in WCF, which is a Microsoft's platform for developing service-oriented applications.

Keywords: Cloud Computing, Windows 8, Service-Oriented Architecture, Personal Training.

1 INTRODUÇÃO

1.1 Motivação

No Brasil, cresce o número de academias para satisfazer a demanda de pessoas que as procuram segundo pesquisa da IHRSA Latin America (NEGOCIO, 2012). O Brasil é o segundo país com maior número de academias conforme Tabela.

Tabela 1.1: Ranking dos 10 maiores países em número de academias

País	**Número de Academias**
Estados Unidos	29960
Brasil	23400
México	7826
Alemanha	7304
Itália	7000
Coréia do Sul	6800
Argentina	6632
Canadá	6242
Reino Unido	5852
Espanha	5630

Fonte: NEGOCIO, 2012.

Pereira (1996) e Guiseline (2001) mostram que nos últimos anos tem sido cada vez maior o número de pessoas que procuram as academias em busca da prática de atividade física. Anteriormente, acreditava-se ser um modismo passageiro, mas o tempo mostrou o contrário: existe um interesse pela melhoria da saúde e da qualidade de vida.

O aumento do número de academias ocorre devido ao maior interesse dos brasileiros em frequentá-las, decorrente de diversos fatores, sendo o principal a estética/emagrecimento conforme Tabela.

Tabela 1.2: Motivos de Aderência

Motivos de aderência	Número Total	Frequência de incidência de respostas
Estético	198	29%
Saúde	159	23%
Bem-estar físico	147	21%
Bem-estar mental	71	10%
Porque gosta	66	10%
Convívio social	45	7%
Questão de múltipla escolha. N=686		

Fonte: ZANETTE, 2003. p. 95.

O estudo de Zanette (2003) também mostra os motivos de aderência à academia relacionados ao fator sexo conforme Figura 1.1.

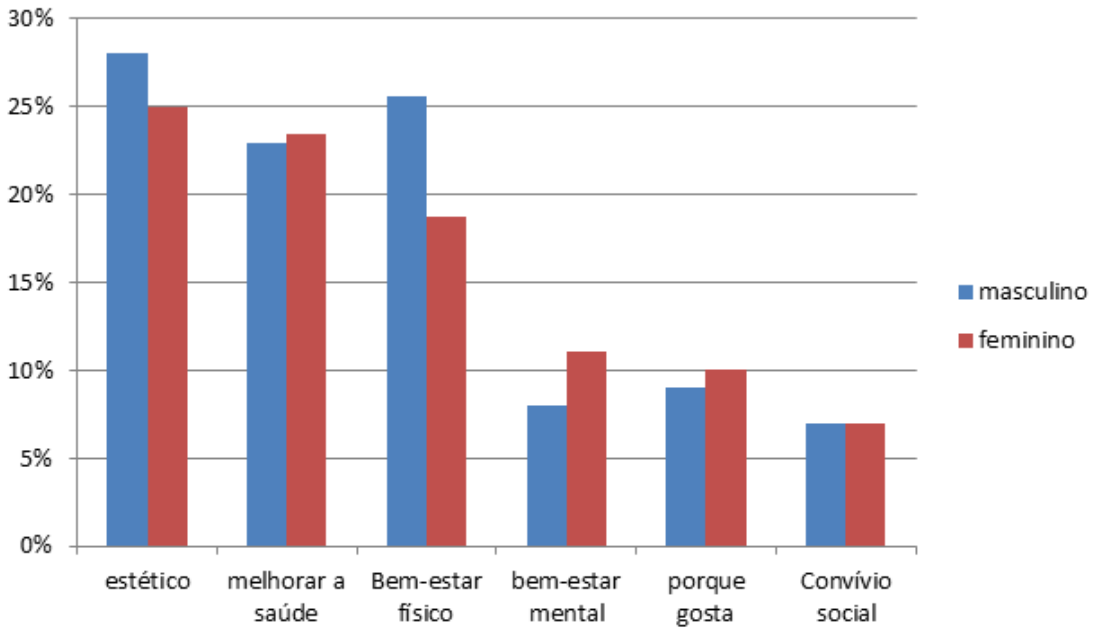

Figura 1.1: Motivos de aderência relacionados ao fator sexo (ZANETTE, 2003. p. 96)

No Brasil, 48,5% da população está acima do peso conforme estudos da Vigitel Brasil (PORTAL, 2012). Porto Alegre lidera o ranking como a capital com o maior percentual de obesos. De acordo com o Ministério da Saúde, pessoas obesas também têm mais chance de sofrer com doenças cardiovasculares. A obesidade é um dos fatores que levam à procura por profissionais de educação física. Segundo Matsuda e Matsuda (2000), a atividade física é importante no controle do peso e gordura corporal, podendo

contribuir na prevenção e controle de algumas condições clinicas associadas a estes fatores, como doenças cardiovasculares, diabetes, hipertensão, AVC, artrite, apneia do sono, prejuízo da mobilidade.

Para as pessoas que precisam de resultados mais rápidos e não dispõem de muito tempo para se exercitar, existe a opção da procura por um personal trainer. Esse profissional pode trabalhar de maneira específica as necessidades de seus alunos, seja perder peso, condicionamento físico, hipertrofia (aumento de massa muscular), entre outros.

Os dispositivos móveis tendem a facilitar o trabalho do personal trainer, o qual pode ter em mãos todas as informações e observações acerca do desempenho de seus alunos, substituindo pastas e arquivos. O atendimento aos seus alunos será mais eficiente e dinâmico, pois o profissional poderá fazer constantes observações em seu aparelho móvel.

Segundo o NPD DisplaySearch, na pesquisa Quarterly Mobile PC Shipment and Forecast Report, a venda de Tablets cresce de maneira a superar a venda de Notebooks até o ano de 2017. Dentre os fatores que influenciam esse acontecimento, estão a longa duração das baterias, portabilidade e o uso instantâneo, no qual o usuário pode ligar e usar rapidamente (DISPLAYSEARCH, 2012). A projeção para a venda de tablets é representada na Figura 1.2.

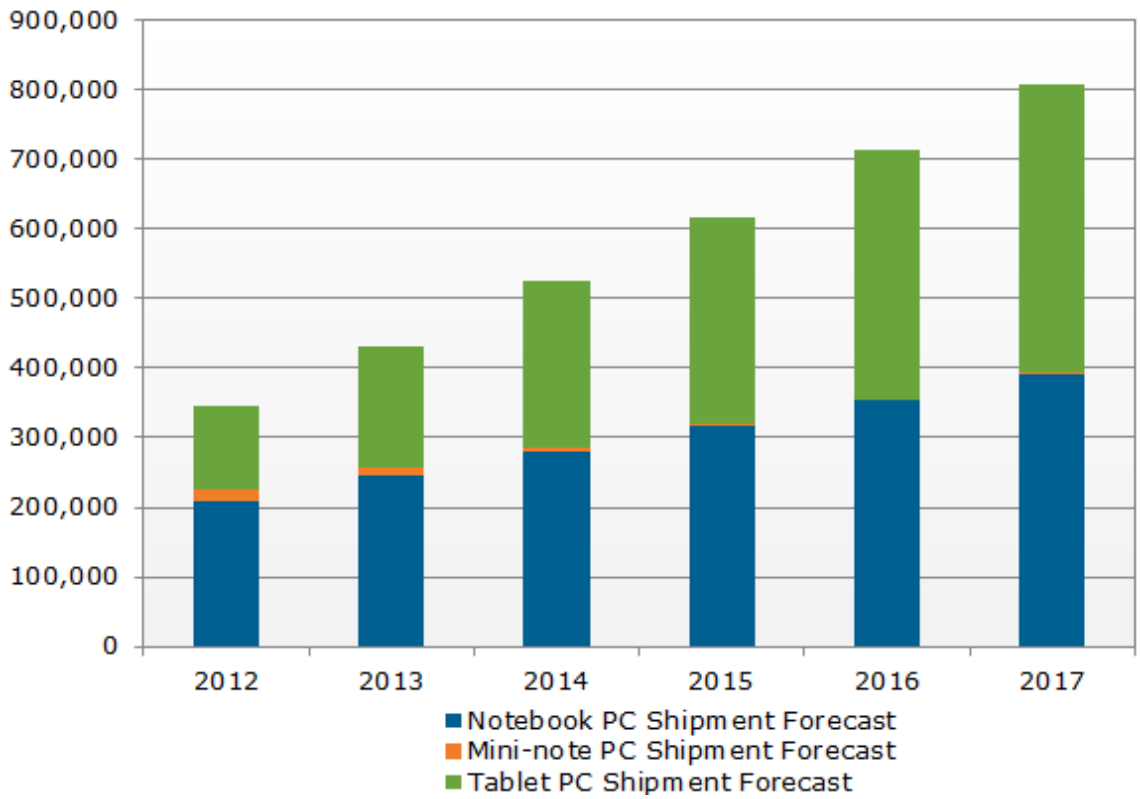

Figura 1.2: Previsão mundial de vendas para dispositivos móveis
(DISPLAYSEARCH, 2012)

Com a tendência de expansão, cada vez maior, dos tablets e a necessidade de acessar as informações pessoais a partir de qualquer lugar, a Computação em Nuvem proporciona um servidor para o armazenamento das informações e compartilhamento para os dispositivos, sem a necessidade de adquirir um servidor próprio para o estabelecimento. A Computação em Nuvem fornece alta disponibilidade e recursos sob demanda, o que é uma vantagem não só do ponto de vista do desempenho no processamento, mas também uma questão econômica, pois só se pagam os recursos que são utilizados no período determinado pelo fornecedor do serviço.

1.2 Objetivo

Este projeto visa facilitar o trabalho do personal trainer, já que este precisa ter em mãos todas as observações referentes a cada aluno para planejar cada atividade física de acordo com o desempenho e progressos de cada treino. Assim, o professor vai dispor de um recurso que organiza e facilita suas anotações, dispensando novas pilhas de papéis e até mesmo arquivos. O projeto também estará facilitando e contribuindo para que o aluno alcance seus objetivos de forma eficiente e precisa.

1.3 Organização do texto

Os próximos capítulos que estão organizados da seguinte forma:

- Capítulo 2 – Abordará as principais características e funcionalidades da Computação em Nuvem. Os valores aplicados para cobrança dos serviços. Também serão descritos os tipos de serviços oferecidos;
- Capítulo 3 – Descreverá os fundamentos da Arquitetura Orientada a Serviços (SOA) e como esses fundamentos se aplicam ao WCF;
- Capítulo 4 – Introduzirá o Windows 8 e sua arquitetura, novo sistema operacional da Microsoft projetado para tablets;
- Capítulo 5 – Detalhes da modelagem do sistema SharpGym serão apresentados neste capítulo;
- Capítulo 6 – Um protótipo para o sistema será descrito e detalhes no decorrer deste capítulo;
- Capítulo 7 – Por fim, esse capítulo mostra os aspectos positivos do desenvolvimento do sistema, análise por objetivos alcançados e sugestões para trabalhos futuros.

2 COMPUTAÇÃO EM NUVEM

Neste capítulo, será abordada a Computação em Nuvem, assim como seus benefícios. Serão apresentados as suas características e os serviços oferecidos. A plataforma Windows Azure será introduzida como um provedor de serviços de Computação em Nuvem e suas características serão definidas.

2.1 Definições

Segundo Mell e Grance (2011), Computação em Nuvem é um modelo para possibilitar o acesso via rede a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo, redes, servidores, armazenamento, aplicativos e serviços) que podem ser rapidamente fornecidos e liberados com o mínimo de esforço de gerenciamento ou interação com o provedor do serviço, de uma maneira ubíqua, conveniente e sob demanda.

Nuvem é uma metáfora da internet derivada da representação nos diagramas de rede, nos quais a internet é desenhada como uma nuvem, exemplificada na Figura 2.1. Logo, Computação em Nuvem é um tipo de computação cujos recursos estão na internet.

A Computação em Nuvem é um dos resultados da chamada Computação Utilitária, termo cunhado por John McCarthy em um discurso durante as comemorações do Centenário do MIT em 1961.

> If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility... The computer utility could become the basis of a new important industry. (GARFINKEL, 1999)

A Computação Utilitária parte do princípio de que o acesso aos recursos computacionais deveria ser pago pelo quanto e quando fosse usado, assim como se paga pelos serviços de água, luz e telefone. Fazendo uma analogia com o serviço de água: se não houvesse fornecimento residencial de água, cada casa deveria ser responsável por escavar um poço ou desviar um córrego e tratar a água para o consumo potável. Porém existem empresas que se responsabilizam pela captação e manutenção do fornecimento de água, assim os contratantes dos serviços pagam apenas pelo consumido e supõe-se que haja água o quanto seja necessário.

A Computação em Nuvem segue esse princípio e dispõe os recursos solicitados pelo usuário sob demanda e apenas cobra pelo que foi propriamente gasto de acordo com métricas definidas pelo prestador do serviço. Essa é a realização de um sonho de longa data na computação (KATZ, RABKIN,STOICA, 2012), os conceitos da Computação em Nuvem podem não ser novidades para o desenvolvimento de software, mas o termo funciona como uma marca comercial que identifica a criação de uma indústria, formada

por grandes provedores, que abastece e mantém serviços em Nuvem de qualidade e confiabilidade.

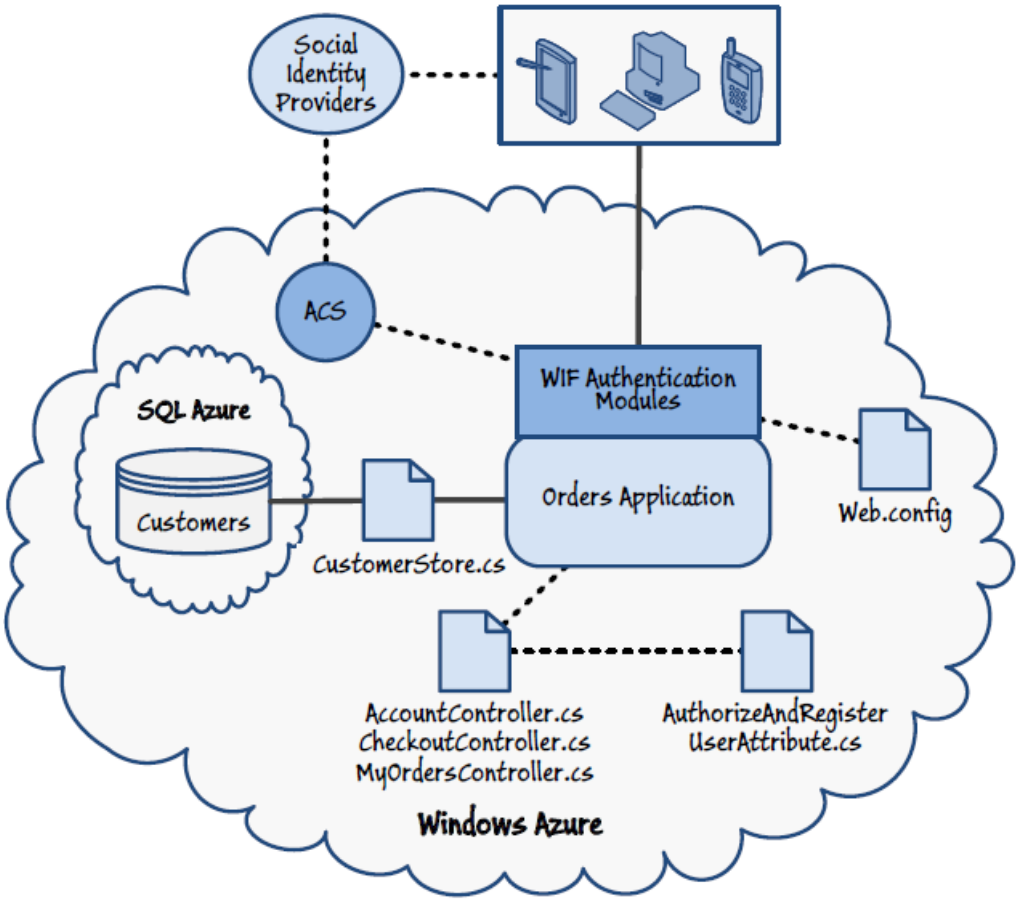

Figura 2.1: Exemplo de diagrama de rede (DENSMORE, 2012, p. 49)

2.2 Características

Uma plataforma comum de Computação em Nuvem tipicamente incorpora as seguintes características conforme classificação de Krishnan (2010):

2.2.1 Ilusão de recursos infinitos:

O usuário não precisa se preocupar com o quanto de recursos serão consumidos durante a computação, tais recursos devem ser sempre suficientes, tanto capacidade de armazenamento, processamento ou infraestrutura computacional. É importante enfatizar que a ideia de recursos infinitos é uma "ilusão", pois de fato existe um limite, mas ele deve ser tão alto quanto possível, ou seja, que nunca seja aparente ao usuário.

2.2.2 Escalabilidade sob demanda

O usuário pode acrescentar recursos sempre que forem necessários e de forma instantânea, abstraindo totalmente a alocação física dos recursos requeridos.

2.2.3 Pay-for-Play (Pague para Jogar)

Não há necessidade de fazer investimentos iniciais, reservas ou contratos de longo prazo. Paga-se apenas pelo software e hardware utilizado, salvo condições especiais de pagamento que favoreçam o usuário ao comprar, por exemplo, uma quantidade fixa de recursos por mês, mas sempre considerando as características 2.2.1 e 2.2.2 que asseguram que podem ser solicitados sempre mais recursos.

2.2.4 Alta disponibilidade, garantida por um SLA (Service Level Agreement - Acordo de nível de serviço)

O provedor do serviço de nuvem deve garantir um nível mínimo de qualidade do serviço e deve incluir um sistema de ressarcimento caso não seja possível entregar o mínimo concordado.

2.2.5 Data Centers distribuídos geograficamente

Questões como balanceamento de carga, latência das conexões de internet, interesses legais, geopolítica, entre outras, devem ser levadas em consideração pelo provedor da nuvem quando existem clientes espalhados ao redor do mundo que precisam do serviço quando e o quanto for necessário e com a qualidade exigida pela característica2.2.4.

2.3 Tipos de serviços em nuvem

Os provedores de serviços em nuvem, como o Windows Azure, oferecem vários tipos de serviços para os seus usuários. De acordo com Mackenzie (2011), serviços de nuvem são tipicamente classificados como IaaS, PaaS e SaaS.

2.3.1 Infrastructure-as-a-Service (IaaS)

Nesse modelo, o núcleo do serviço fornecido é uma máquina virtual com um sistema operacional. Ainda há outros elementos oferecidos, como armazenamento em nuvem, componentes de infraestrutura de redes e firewall.

O usuário é responsável por tudo nesse sistema operacional, incluindo instalação de novos programas. Entretanto, há a vantagem de não necessitar que ele mesmo compre um servidor, por se tratar de um serviço totalmente terceirizado de infraestrutura computacional sob demanda.

2.3.2 Plataform-as-a-Service (PaaS)

O provedor de serviços de nuvem oferece uma plataforma ao desenvolvimento, ao teste, à implantação e à manutenção de aplicações, no qual o usuário é responsável somente pelo serviço injetado nesse ambiente.

2.3.3 Software-as-a-Service (SaaS)

Disponibilização de um serviço totalmente via internet, no qual o cliente meramente faz uso do serviço. O usuário apenas paga pela utilização do serviço, enquanto hardware, infraestrutura e até mesmo software são de responsabilidade do provedor, incluindo atualizações e suporte. É a forma mais comum de serviço, por exemplo, provedores de e-mail e documentos online.

2.4 Windows Azure Platform

De acordo com Hay e Prince (2011), o Windows Azure é um sistema operacional que proporciona a habilidade de executar aplicativos de uma maneira altamente escalável e gerenciável. Esse sistema operacional executa no Data Center (Centro de Dados) Microsoft, o qual é um ambiente para Computação em Nuvem. Assim, o Windows Azure é um "sistema operacional em nuvem".

A Plataforma Windows Azure é o serviço de nuvem provido pela Microsoft e representa um grupo de tecnologias em nuvem formado por algumas partes principais, são elas: Azure AppFabric, SQL Azure e Windows Azure, como representados na Figura 2.2.

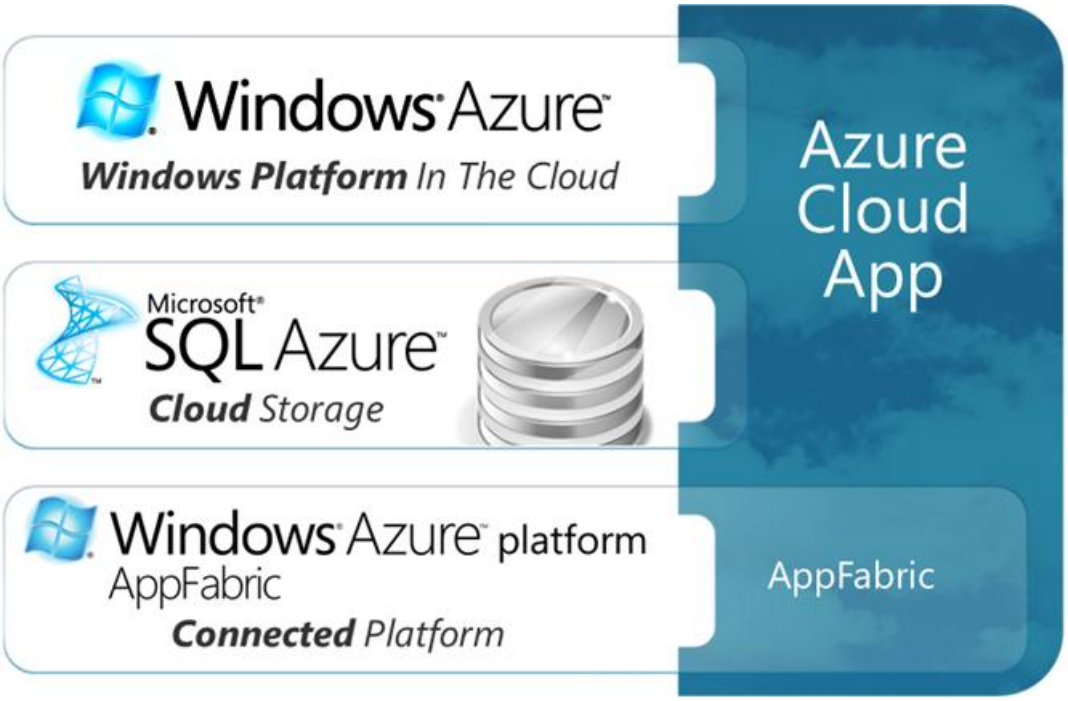

Figura 2.2: Componentes da Plataforma Windows Azure. (FLORIN, 2012)

2.4.1 Azure AppFabric

O Azure AppFabric existe como uma extensão do Windows Azure, que facilita o acesso aos serviços prestados por meio de uma API pública REST baseada em chamadas HTTP, ou seja, é o responsável pela conexão entre as demais partes da nuvem e com o ambiente externo. É formado por dois componentes: Service Bus e Access Control.

2.4.1.1Service Bus

Atua para contornar algumas barreiras existentes na comunicação em rede, como firewalls e dispositivos NAT, tendo como único propósito a transmissão de mensagens através da nuvem para os serviços fora da nuvem, por exemplo, os que executam em máquinas locais, também chamadas de “on-premise”.

2.4.1.2Access Control

Proporciona mecanismos de autenticação para integrar provedores de autenticação, como Windows Live, Google ou Facebook, com os usuários do serviço de nuvem. Esses mecanismos evitam que o usuário tenha que desenvolver seus próprios métodos de autenticação e ainda pode contar com a confiabilidade e segurança de provedores bem estabelecidos no mercado ou simplesmente por uma questão de preferência pessoal.

2.4.2 SQL Azure

SQL Azure é um Servidor SQL hospedado em nuvem. Possui as características de um banco de dados relacional, mas com o adicional de funcionar em uma plataforma de alta disponibilidade, escalável e com controle do balanceamento de carga.

A maior vantagem do SQL Azure sobre um Servidor SQL hospedado em um ambiente local é não haver a necessidade de adquirir hardware para hospedar o banco de dados nem mesmo instalar a infraestrutura de software para o Sistema Gerenciador de Banco de Dados (SGBD), o qual pode ser pago, como o Microsoft SQL Server e Oracle, ou gratuito, como MySQL.

Além de fornecer um sistema completo para armazenar o banco de dados, o SQL Azure também se responsabiliza por replicar dados, tratar de falhas de hardware e escalar o banco de dados para mais ou para menos, dependendo do utilizado pelo usuário. Sem tais preocupações, o usuário pode dedicar mais tempo para planejar e codificar seu projeto de banco de dados.

2.4.3 Windows Azure

O Windows Azure é o sistema operacional de fato, a plataforma na qual as aplicações serão implantadas e executadas, de maneira que a computação e armazenamento são realizados sob demanda e a configuração e manutenção são feitas através dos Data Centers da Microsoft.

2.5 Principais características do Windows Azure

Segundo Krishnan (2010) e Hay e Prince (2011), o Windows Azure possui algumas características principais que definem as funcionalidades do serviço de Computação em Nuvem oferecido, as quais serão descritas a seguir.

2.5.1 Hospedagem de Serviços

Aplicações do tipo servidor, websites e outros tipos podem ser hospedados no Windows Azure, respeitando algumas limitações, por exemplo, na versão atual não é permitido operações que exijam privilégios administrativos. Essas aplicações não precisam de desenvolvimento especial, pois o sistema operacional hospedeiro é o Windows Server, então um aplicativo desenvolvido para executar em ambiente Windows irá funcionar no Windows Azure.

Para aplicativos que utilizam o .Net Framework, a versão instalada por padrão nos servidores é a 3.5 com o Service Pack 1, propriamente atualizado.

Outras linguagens de programação que utilizam FastCGI também podem ser usadas para serem implantadas no Windows Azure, como PHP, Ruby on Rails e Python.

2.5.2 Ferramentas de desenvolvimentos

Como a maioria da plataforma Microsoft, o Windows Azure tem várias ferramentas que podem fazer o desenvolvimento mais fácil. Por exemplo, existe uma API pela qual é permitido enviar relatório de erros e atualizar serviços e arquivos de configuração. Também existe um SDK para que o desenvolvedor possa implantar sua aplicação em um simulador de nuvem, executando em seu computador pessoal, o qual permite testar se as técnicas utilizadas para programar são adequadas e eficientes para o ambiente de nuvem.

2.5.3 Virtualização

Dispor de um computador para cada instância solicitada no Windows Azure é extremamente custoso e, se considerar que o usuário pode solicitar uma nova instância para hospedar um aplicativo a qualquer momento, isso pode se tornar praticamente impossível.

Nesse âmbito, a virtualização é uma opção muito vantajosa e o Data Center a utiliza massivamente, representando a base da pilha do Windows Azure, pois a alocação de novas instâncias é feita de maneira virtual. A criação de uma nova máquina virtual é muito mais simples que adquirir e instalar um novo servidor.

2.5.4 The Fabric Controller

Ao modelar um sistema para ser implantando num servidor de nuvem, é necessário definir quantas máquinas serão usadas, qual configuração de rede é a ideal, quais softwares devem ser instalados, enfim, toda a infraestrutura que se faça necessária. Não bastando o planejamento, ainda é preciso adquirir todo o material e dedicar um tempo à instalação e à configuração do sistema. O Fabric Controller automatiza várias dessas atividades, responsabilizando-se por encontrar o hardware certo, implantar os softwares de infraestrutura necessários e aplicar configurações de rede. É um software distribuído que executa através de todos os nodos do Windows Azure e monitora o estado de cada nodo, em caso de um aplicativo encerrar o seu funcionamento de maneira anormal, o Fabric Controller pode reinicializá-lo ou até mesmo iniciá-lo em outro nodo.

O Fabric Controller gerencia o hardware do Data Center, monitorando o fornecimento de energia e tomando atitudes corretivas quando algo está fora de funcionamento ou executando de forma incorreta. Quando é detectado um problema e não é possível realizar uma atitude corretiva, o hardware é desativado e um operador técnico humano é solicitado pra investigar o problema. Igualmente há o cuidado para que o sistema operacional que executa nos nodos do Windows Azure também se mantenha funcionando ininterruptamente.

2.5.5 Armazenamento

O armazenamento em nuvem do Windows Azure é feito de forma redundante para proteção em caso de falhas de hardware ou software. Além de redundante, o armazenamento é consistente, ou seja, uma escrita é instantaneamente visível por todas as subsequentes leituras, o que pode diminuir um pouco o desempenho devido aos controles de exclusão mútua para resolver os conflitos de leitura/escrita. Se a performance é uma necessidade maior do que a garantia de uma leitura consistente, como arquivos de log e trace, o Windows Azure fornece a opção “optimistic concurrency” (concorrência otimista).

A forma de cobrança obedece ao conceito de Pay-for-Play, ou seja, é cobrado apenas o espaço que é utilizado de fato.

São três os tipos principais de armazenamento em Windows Azure: Binary Large Object (BLOB), tabelas semiestruturadas e filas de mensagens. A seguir uma breve explicação sobre cada um.

2.5.5.1Blob

O serviço de armazenamento blob oferece um sistema simples para armazenar arquivos de até 1tb de tamanho, quase não há limite para o número de arquivos que podem ser armazenados ou o total de espaço disponível, também é possível dividir arquivos em pequenas porções para fazer o upload de grandes arquivos de forma mais fácil e permitir a retomada posterior do envio, não precisando fazê-lo de uma só vez.

2.5.5.2Filas de mensagens

Filas de mensagens proveem um serviço confiável de armazenamento e entrega de mensagens entre as aplicações ou componentes implantados no Windows Azure. Esse serviço evita que cada usuário tenha que criar seu próprio sistema de mensagens, contando com uma opção que suporta um número ilimitado de mensagens e com garantia de entrega. Também é possível controlar o tempo de vida das mensagens, controlando exatamente quando será feito o processamento da mensagem e remoção da fila.

2.5.5.3Tabelas semiestruturadas

Praticamente todas as aplicações precisam de algum tipo armazenamento de dados. Atualmente isso é obtido através do sistema de banco de dados relacional, como Oracle, SQL Server, MySQL e outros. Logo, o serviço de tabelas do Windows Azure tem o potencial de ser o mais interessante e útil para a maioria dos aplicativos. São permitidas operações de manipulação de dados CRUD (Create, Read, Update e Delete) para quaisquer tipos de dados que o usuário possa querer.

O serviço de tabelas semiestruturadas é um banco de dados escalável, altamente confiável, mas sem as características de um banco relacional, ou seja, não há os conceitos de chaves estrangeiras, de normalização de tabelas, de suporte à sintaxe SQL, de realizar junções entre tabelas, entre outras funcionalidades que foram descartadas para que o sistema de tabelas se adequasse ao ambiente de nuvem, minimizando os problemas de um banco de dados tradicional.

2.6 Detalhes de preços

Para novos usuários, a Microsoft oferece um período de testes de três meses para experimentar os serviços do Windows Azure, porém com alguns limites. Caso algum limite seja ultrapassado, o serviço é desativado, podendo ser reativado através do pagamento do valor referente à quantidade de recurso excedido. Todos os detalhes podem ser consultados em (WINDOWS, 2012).

Para fins de exemplificação, serão descritos os detalhes de preço na Tabela 2.1 para Máquinas Virtuais executando o sistema operacional Windows Server, cuja definição referente ao tipo de serviço disponível para contratação é a seguinte:

> As Máquinas Virtuais do Windows Azure permitem que você implante uma imagem personalizada do Windows Server ou Linux no Windows Azure. As Máquinas Virtuais proporcionam aos desenvolvedores controle completo do ambiente do aplicativo e permitem fácil migração dos aplicativos existentes para a nuvem. (WINDOWS, 2012)

Os valores cobrados são dados em referência à hora de uma instância Pequena, sendo que uma hora de uma instância Extra pequena equivale a 1/6 de hora da instância Pequena, a hora da instância Média equivale a 2 horas, a Grande equivale a 4 horas e a Extragrande equivale a 8 horas.

Supondo que o usuário tenha instanciado uma máquina Grande e utilizou o serviço durante 3 horas, o valor pago é calculado na equação abaixo:

$$3\ horas\ de\ uso\ \times 4\ horas\ de\ equivalência\ \times \$\ 0{,}46\ preço/hora = \$\ 5{,}52$$

Deve-se tomar o cuidado de observar os valores finais para averiguar se o serviço contratado realmente está de acordo com as necessidades, para que o usuário não pague por uma instância muito maior do que precise. Caso seja percebido um desperdício de recursos, pode-se diminuir o tamanho da instância ou remover instâncias não utilizadas.

Tabela 2.1: Detalhes de preços para Máquinas Virtuais Executando Windows Server

Tamanho da instância de computação	**Núcleos de CPU**	**Memória**	**Preço/Hora**
Extra pequena	Compartilhado entre outras instâncias do mesmo tipo	768 MB	$ 0,02
Pequena	1	1,75 GB	$ 0,115
Média	2	3,5 GB	$ 0,23
Grande	4	7 GB	$ 0,46
Extragrande	8	14 GB	$ 0,92

Fonte: WINDOWS, 2012

3 ARQUITETURA ORIENTADA A SERVIÇOS

Esse capítulo tem por objetivo mostrar os conceitos e princípios fundamentais da Arquitetura Orientada a Serviços. A plataforma que implementa os princípios da Arquitetura Orientada a Serviços escolhida é o WCF. Serão apresentados seus fundamentos e suas funcionalidades para a compreensão de aspectos da modelagem do sistema.

3.1 Definições

De acordo com Sharp (2010), a Arquitetura Orientada a Serviços (SOA) consiste em um conjunto de recursos distribuídos que estão disponíveis como serviços independentes e podem ser acessados sem o conhecimento de como esses serviços são implementados. O resultado disso é a possibilidade de desenvolver sistemas cada vez mais complexos, nos quais a informação pode estar distribuída através de vários computadores e até mesmo ao redor do mundo, exigindo que o acesso a esses dados esteja disponível de uma maneira simples.

Dessa forma, uma aplicação consiste de um ou mais serviços que são invocados por um módulo cliente, esses serviços podem ainda invocar outros serviços, como representado na Figura 3.1, na qual o círculo escuro representa o módulo cliente e os demais são serviços. Como os serviços trabalham em conjunto, fornecendo resultados para as chamadas do cliente, a união desses elementos forma a aplicação como um todo.

3.2 Arquitetura Orientada a Serviços (SOA)

Com o objetivo de promover o reuso e o encapsulamento de código, a SOA segue o padrão “caixa preta” (CIBRARO, 2010), no qual o serviço recebe uma série de parâmetros de entrada e gera uma saída sem exibir o seu funcionamento interno. Esse objetivo torna o sistema fracamente acoplado, pois alterações no serviço não devem ser propagadas aos clientes nem devem alterar a maneira com que o cliente chama o serviço.

De acordo com a Organização para o Avanço dos Padrões de Informação Estruturada (OASIS, Organization for the Advancement of Structured Information Standards): “SOA é um paradigma para organizar e utilizar de recursos que podem ser distribuídos sob o controle de domínios proprietários diferentes” (OASIS, 2012).

Segundo Bustamante (2007), o paradigma SOA é a evolução da Programação Orientada a Objetos (POO) e da Programação Orientada a Componentes (POC), pois incorpora as funcionalidades de ambos e ainda adiciona novas.

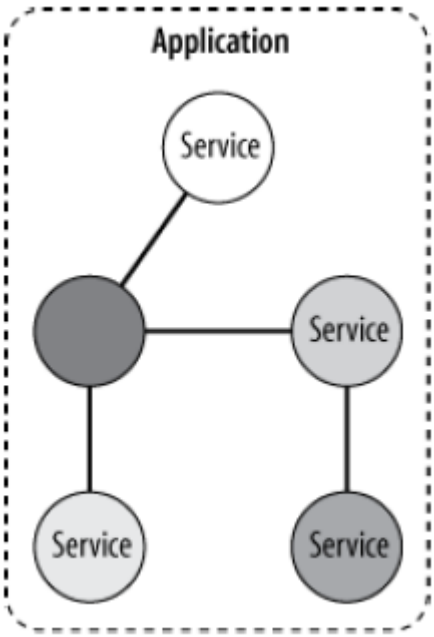

Figura 3.1: Uma aplicação orientada a serviços (LÖWY, 2010)

Na POO, classes encapsulam funcionalidades e promovem reuso de código. Para compartilhar classes entre aplicações distintas, entretanto, deve-se copiar o código entre as classes, como representado na Figura 3.2.

POC introduziu o conceito de compartilhar binários que encapsulam classes reusáveis. Inicialmente esse método era limitado à mesma máquina até que a distribuição foi possível por tecnologias como COM, DCOM, CORBA e, mais tarde, EJBs e .NET Remoting. Embora a distribuição seja feita de maneiras diferentes, o resultado é o mesmo - componentes binários são ativados como processos nas máquinas hospedeiras.

A utilização de um componente localizado remotamente e que possui classes compartilhadas entre duas ou mais aplicações pode ser visualizada na Figura 3.3.

POC tem muitas limitações, mas a mais crítica é o acoplamento a uma tecnologia específica. Não sendo possível o acesso a recursos construídos sob diferentes plataformas, a não ser que existam adaptadores para transformar mensagens de uma tecnologia para outra, então, quando uma mensagem é recebida, ela precisa ser traduzida e processada. Essa abordagem é pesada e pode introduzir múltiplas transformações entre clientes e componentes, o que às vezes nem é possível.

SOA resolve o problema herdado da POC por introduzir, via Web Services, o conceito de contratos, políticas e interoperabilidade. A respeito disso, os aplicativos podem se comunicar com os serviços de outro sem se preocupar com a tecnologia que cada um implanta.

A interoperabilidade permite situações tais como mostrada na Figura 3.4, na qual aplicações implementadas em diferentes plataformas, .Net e Java, compartilham componentes disponíveis através de serviços.

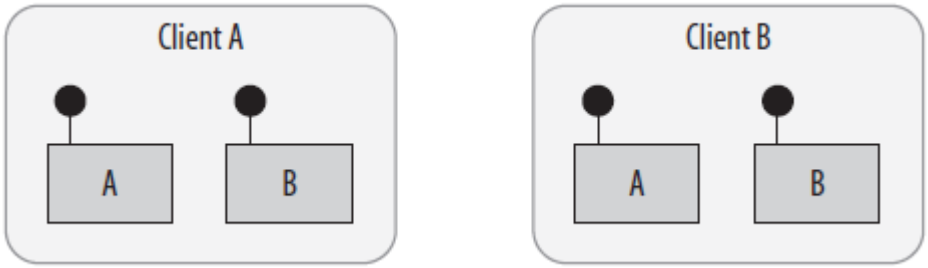

Figura 3.2: Duas aplicações, Client A e Client B, compartilhando classes, A e B (BUSTAMANTE, 2007)

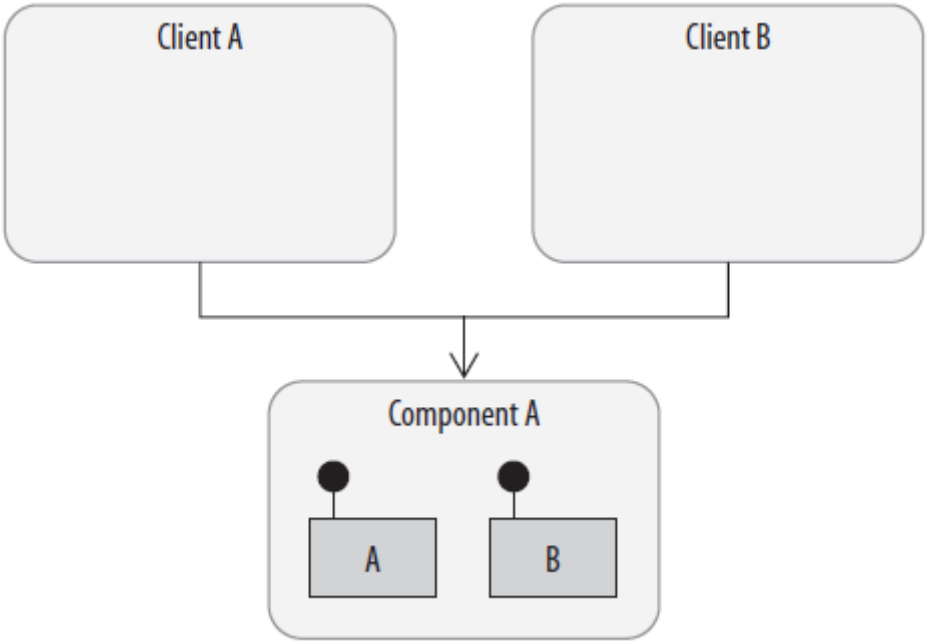

Figura 3.3: Duas aplicações, Client A e Client B, compartilhando um mesmo componente remoto, Component A (BUSTAMANTE, 2007)

3.3 Os princípios de SOA

Embora não haja um padrão oficial para SOA conforme Löwy (2010), Sharp (2010), Bustamante (2007) e Cibrari (2010), existem quatro princípios básicos que guiam os princípios para alcançar a SOA, os quais serão descritos nas próximas subseções.

3.3.1 Limites do serviço são explícitos

Os serviços são responsáveis por expor um conjunto específico de funcionalidades através de um contrato bem definido. O contrato descreve uma série de operações concretas e mensagens suportadas para o serviço, além de definir claramente que tipo de resposta será gerada pelo serviço.

O serviço encapsula completamente os detalhes de implementação por trás dos serviços, assim, qualquer plataforma de tecnologia pode ser para o seu desenvolvimento

sem impactar o cliente, além disso, a localização do serviço não é importante para o cliente.

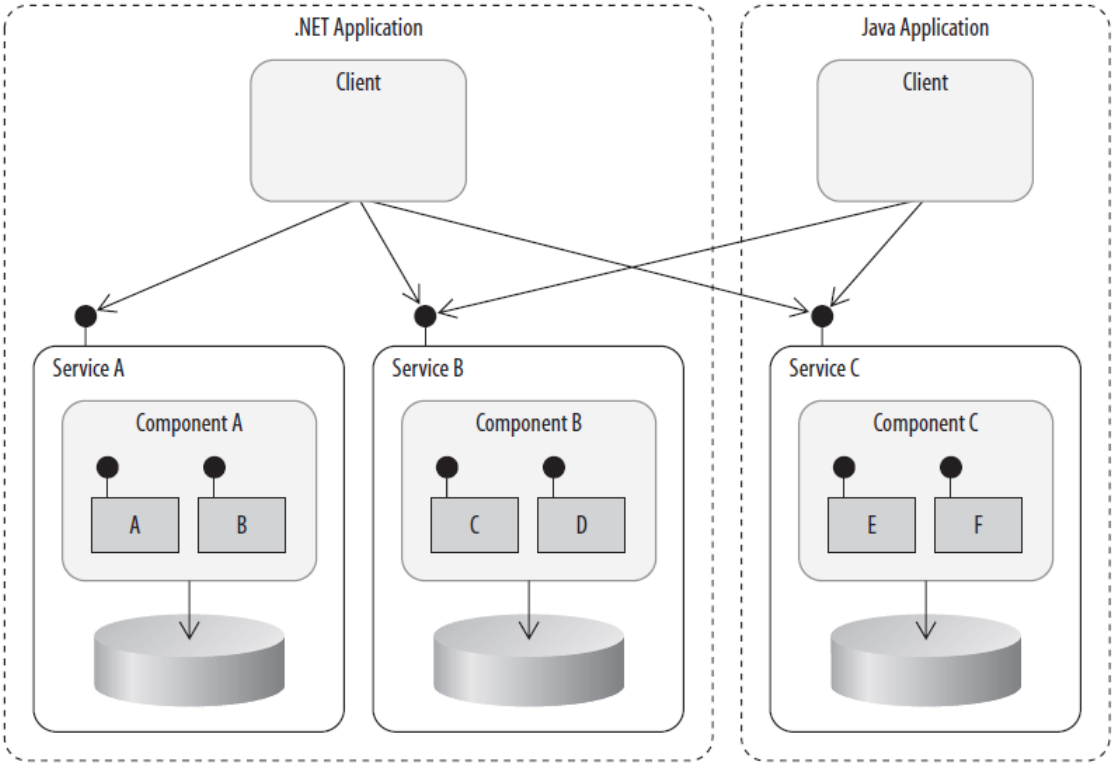

Figura 3.4 Aplicações e serviços desenvolvidos em plataformas distintas compartilhando componentes (BUSTAMANTE, 2007)

3.3.2 Os serviços são autônomos

Um serviço deve operar independentemente, sendo, assim, móvel ou até mesmo substituível sem impactar outros serviços ou funcionalidades.

Serviços são peças autônomas que não dependem do comportamento de outros serviços. Eles são considerados disponíveis sem precisar explicitamente os instanciar.

O serviço representa um conjunto maior de funcionalidades que manipulam seus próprios componentes, acesso a dados e armazenamento, se aplicável, e deve ser capaz de realizar as suas funções principais sem dependências externar.

3.3.3 Clientes e serviços compartilham contratos, não código

O contrato não deve ser modificado uma vez que publicado, ou se precisar, pelo menos o mínimo de compatibilidade com versões anteriores deve ser garantido para que o contrato não deixe de ser explícito.

Em teoria os contratos não estão presos a uma tecnologia particular ou plataforma, mas esse não é um requisito oficial de SOA atualmente, somente uma forte tendência.

3.3.4 A compatibilidade é baseada em políticas

Políticas descrevem quais condições são necessárias para processar uma mensagem, tais condições são usadas para negociar elementos da comunicação, como protocolos de

comunicação, requisitos de segurança e requisitos de confiabilidade, indicando a semântica do serviço.

Políticas são atualmente uma extensão do WSDL, assim os clientes podem ser avisados no momento que eles invocam os serviços as questões envolvidas no acesso ao serviço de uma maneira compatível.

3.4 Windows Communication Foundation (WCF)

O WCF é a plataforma Microsoft para desenvolver aplicações que implementam a arquitetura orientada a serviços (SOA), pensada especialmente para sistemas distribuídos. Proporciona baixo acoplamento entre o serviço e as funcionalidades expostas, escolha de protocolo, formato de codificação das mensagens e ambiente de hospedagem.

A Microsoft desenvolveu o WCF para integrar suas tecnologias anteriores de plataformas SOA: Enterprise Services, .Net Remoting e ASP.NET Web Services (ASMX), com a vantagem de que a nova plataforma eliminou as limitações de linguagem de programação existente em seus antecessores, tornando-o interoperável, ou seja, a linguagem de programação em que os aplicativos clientes são escritos não se restringe à mesma em que o serviço foi escrito e, no caso dos serviços Microsoft, alguma linguagem disponível pelo .Net Framework.

A "anatomia" de um serviço WCF é ilustrada na Figura 3.5, na qual se pode visualizar o serviço em seu hospedeiro, o canal de comunicação que transporta as mensagens através de um protocolo e as políticas, os esquemas e os contratos que definem qual o formato das mensagens e se há algum processamento especial para que elas sejam enviadas ao solicitante. Cada parte será descrita em mais detalhes na subseção 3.4.1.

Os serviços podem ser acessados usando-se vários tipos de protocolos suportados, incluindo TCP, HTTP e MSMQ.

3.4.1 Conceitos fundamentais de WCF

Alguns conceitos são fundamentais para compreender o processo de transmissão de mensagens entre clientes e serviços. A seguir, serão apresentados esses fundamentos que, segundo Cibraro (2010) e Bustamante (2007), são os mais importantes.

Na Figura 3.6, é mostrado o processo detalhado de transmissão de mensagens entre cliente e serviço.

3.4.1.1Serialização de mensagens

Aplicações distribuídas necessitam, em algum momento, fazer chamadas remotas através de processos ou máquinas diferentes, isso pode ser assegurado mandando mensagens entre aplicativos. Para alcançar interoperabilidade, o sistema depende de um formato padrão para as mensagens que seja entendido por ambos os lados da comunicação. WCF troca mensagens em formato XML e através de um protocolo previamente concordado entre as aplicações.

3.4.1.2Hospedagem

O serviço precisa ser hospedado antes que os clientes possam chamar as operações do serviço. As funcionalidades de serviços se tornam disponíveis em tempo de execução através do processo hospedeiro, ele é responsável por inicializar a comunicação dos

canais que entregam as mensagens ao serviço e por gerenciar o ciclo de vida dos canais de comunicação dos serviços.

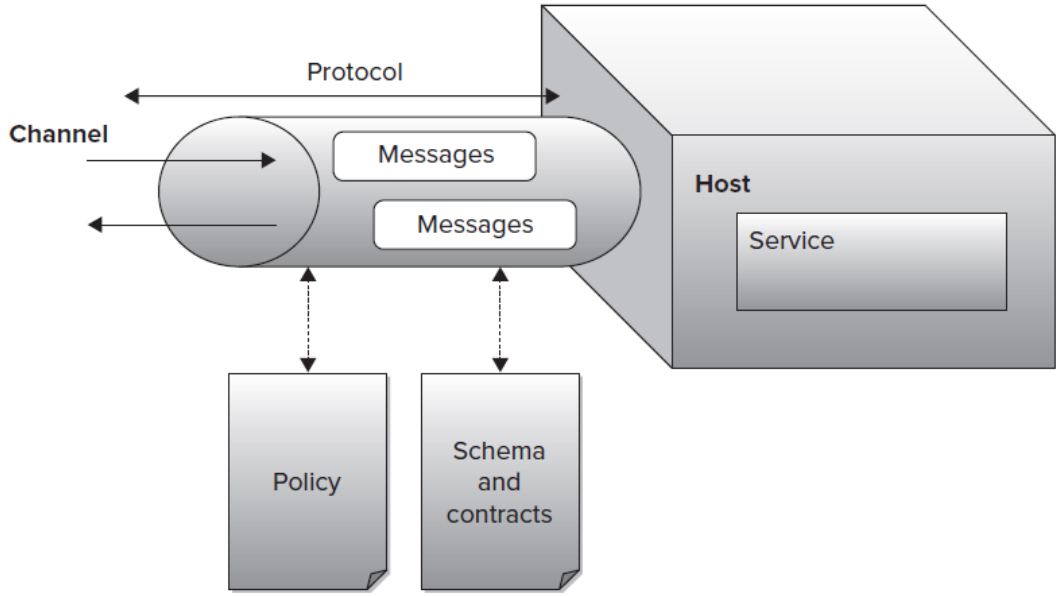

Figura 3.5: Anatomia de um serviço WCF (CIBRARO, 2010)

3.4.1.3Endpoints (pontos de entrada)

Quando o serviço hospedeiro abre um canal de comunicação para um serviço hospedado, há a possibilidade de expor pelo menos um endpoint por serviço. Isso é necessário para que os clientes possam invocar as operações. De fato, os endpoints são a chave da funcionalidade para invocar serviços, descrevendo onde o serviço pode ser encontrado, como ele pode ser encontrado e quais operações podem ser encontradas.

Um endpoint possui três partes:

- Endereço: URI para onde a mensagem é enviada;
- Binding: esquema dos protocolos suportados para o envio;
- Contrato: lista das operações disponíveis;

3.4.1.4Endereços

Cada endpoint é associado a um endereço e identificado por uma URI e esse endereço tem o a seguinte estrutura esquema://domínio:porta/caminho, onde cada parte representa o seguinte:

- Esquema: protocolo utilizado para a comunicação;
- Domínio: endereço de rede ou website no qual o serviço está hospedado;
- Porta: porta específica pela qual o processo está ouvindo e aguardando por mensagens;
- Caminho: localização utilizada para desambiguação entre serviços hospedados no mesmo endereço e porta;

Quando um endpoint é adicionado a um serviço, é preciso especificar um endereço único para cada endpoint, isso quer dizer que é necessário variar pelo um dos parâmetros da URI.

3.4.1.5Binding

O binding descreve os protocolos suportados por um endpoint em particular, especificamente o protocolo de transporte, o formato da mensagem, outro protocolo de comunicação que pode ter questões de segurança e confiabilidade relacionadas ou qualquer outro protocolo que possa afetar o conteúdo da mensagem serializada.

Existe um número de bindings pré-definidos no WCF que representam os protocolos mais comuns para comunicação.

3.4.1.6Metadados

Uma vez que o serviço hospedeiro é configurado para executar um ou mais endpoints e o canal de comunicação está aberto, as operações podem ser invocadas em cada endpoint, respeitando os protocolos suportados em cada endpoint. Os clientes invocam as operações dos serviços em um endpoint particular, para isso são necessárias informações sobre o endpoint, incluindo o endereço, binding e contrato de serviço.

Essas informações são parte dos metadados para um serviço particular. Esses metadados são essenciais para que os clientes possam gerar proxies e invocar o serviço.

3.4.1.7Proxy

Clientes se comunicam com os serviços através de proxies. O proxy é um tipo que expõe operações representativas do serviço e esconde o tratamento da serialização do cliente quando invoca as operações de serviço.

Para aplicações WCF, proxy são baseados no contrato de serviço, então uma vez que se tem acesso ao contrato de serviço, é possível criar um proxy, instanciar e invocar o serviço associado.

Antes de instanciar o proxy e poder chamar as operações do serviço, ele precisa ser provido de informações sobre um dos endpoints expostos pelo contrato de serviço, estabelecendo uma relação 1 pra 1 entre proxy e endpoint.

3.4.1.8Canais

Canais facilitam a comunicação entre clientes e serviços em WCF. O serviço hospedeiro cria um canal que fica escutando a comunicação para cada endpoint, o qual gera um canal de comunicação. O proxy, no lado cliente, cria uma "fábrica de canais", que gera um canal de comunicação para cada instância do serviço no cliente. Ambos os canais de comunicação precisam ser compatíveis entre si para que as mensagens sejam processadas efetivamente.

De fato, o canal de comunicação contém uma pilha de canais - cada canal na pilha é responsável por realizar uma atividade particular enquanto processa uma mensagem. A pilha de canais inclui um canal de transporte, um canal de codificação de mensagens, e outros canais de processamento de mensagens para segurança, confiabilidade e outras funcionalidades que sejam necessárias.

3.4.1.9Behaviors (comportamentos)

A maneira como os componentes de infraestrutura do WCF operam pode se personalizar aplicando comportamentos. Behaviors são usados para modificar como um

endpoint serializa os dados, como agrupa operações em uma transação, as credenciais específicas quando envia ou recebe mensagens e muito mais.

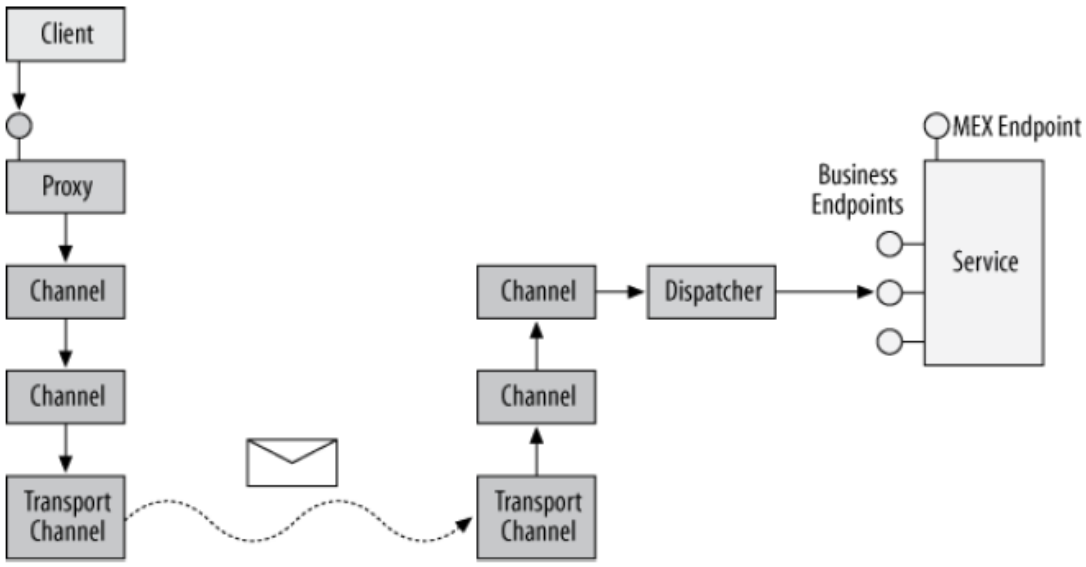

Figura 3.6: Arquitetura envolvida no envio de uma mensagem (LÖWY, 2010)

4 WINDOWS 8

Os sistemas operacionais para dispositivos móveis receberam muita atenção nos últimos anos. As grandes empresas do mundo da computação investiram pesadamente para criar suas plataformas de desenvolvimento e ganhar o mercado de aparelhos e aplicativos.

Nesse capítulo, será apresentado o sistema operacional Windows 8, o mais novo sistema operacional da Microsoft, e sua arquitetura.

4.1 Histórico

O sistema operacional Windows teve sua primeira versão lançada em 1985 como um componente extra para o MS-DOS, fornecendo-lhe uma interface gráfica que proporcionou uma melhor experiência com o usuário, substituindo o modelo de interação por linha de comando. Essas versões iniciais não eram um sistema operacional completo de fato, mas sim um complemento adicional para propiciar uma camada de interface para o MS-DOS (NOVÁK, 2012).

O sucesso das versões futuras do Windows teve forte ligação com as novidades gráficas que acompanhavam o sistema de acordo com o portal W3School, que coleta informações sobre o uso de sistemas operacionais. O Windows XP e Windows 7 lideram o mercado com quase 80% de uso conforme Tabela 4.1.

Durante 27 anos, cada nova versão que chegava ao mercado trazia um novo conjunto de funcionalidades, fazendo com que o sistema estivesse em um processo de evolução contínua e incremental. Mas, para o Windows 8, a Microsoft afirma que o sistema foi “reimaginado”, como disse o CEO Steve Ballmer em entrevista na edição 2012 do CES (Consumer Electronics Show): “with Windows 8 we've reimagined Windows” (GUARDIAN,2012). O Windows 8 não é uma simples atualização da versão anterior ou adição de características que estão na moda.

4.2 Um novo paradigma

Embora o Windows tenha surgido numa era em que computadores pessoais se tornaram parte do dia-a-dia, alguns conceitos como arquivo, diretório, registro de sistema e instalação de aplicativos são necessários para utilizar o sistema operacional, de tal modo que o consumidor doméstico tem certa dificuldade para utilizar inicialmente o sistema, como se o Windows fosse criado com foco em empresas e profissionais da Tecnologia da Informação.

Tabela 4.1: Utilização de sistemas operacionais

2012	Win7	Vista	NT*	WinXP	Linux	Mac	Mobile
November	56.5%	3.0%	3.0%	20.8%	4.8%	9.4%	2.0%
October	56.8%	3.0%	1.8%	22.1%	4.8%	9.2%	1.8%
September	55.7%	3.1%	1.5%	23.6%	4.7%	8.9%	1.8%
August	54.5%	3.2%	1.3%	24.8%	5.0%	8.7%	1.8%
July	53.8%	3.4%	1.2%	26.1%	4.9%	8.2%	1.7%
June	53.2%	3.7%	1.1%	26.2%	5.0%	8.6%	1.6%
May	52.3%	3.9%	1.1%	26.8%	4.9%	9.0%	1.6%
April	51.3%	4.2%	1.0%	27.3%	4.9%	9.3%	1.5%
March	49.9%	4.3%	1.0%	28.9%	4.9%	8.9%	1.4%
February	48.7%	4.5%	0.8%	30.0%	5.0%	9.1%	1.3%
January	47.1%	4.8%	0.9%	31.4%	4.9%	9.0%	1.3%

FONTE: (W3SCHOOLS, 2012)

Empresas como a Apple, ao lançar seu dispositivo telefônico iPhone, trouxeram um novo paradigma centralizado no consumidor, abstraindo os conceitos supracitados. Não é mais necessário que o usuário acompanhe o passo-a-passo da instalação de um aplicativo ou precise gerenciar diretórios e arquivos, resta apenas a responsabilidade de escolher um programa, solicitar a instalação e usá-lo. Do mesmo modo, a remoção da aplicação é simples, sem a preocupação de buscar vestígios residuais que possam ficar escondidos por algum diretório.

A primeira experiência da Microsoft com um sistema operacional voltado para o usuário foi o Windows Phone 7, que trouxe o conceito de Metro Style, uma nova forma de desenvolver interfaces para o usuário, um design inovador e ao mesmo tempo simples e minimalista.

O primeiro anúncio público do Windows Phone ocorreu no Mobile World Congress 2010, mas o anúncio oficial foi durante o MIX 2010 segundo James (2012).

Em Janeiro de 2012, foi anunciado o Windows 8 (GUARDIAN, 2012), o qual traz muitos conceitos novos para o sistema operacional Windows tradicional, entre eles o Store Style para desenvolver aplicações que possuem a característica de serem aplicativos que ocupam totalmente a tela do dispositivo, ou seja, apenas um aplicativo é executado por vez e sem elementos chrome (botões de fechar, minimizar e maximizar). Como o novo sistema operacional executa tanto em máquinas desktop quanto tablets, a experiência com o usuário é a mesma, não importa o dispositivo, tornando a interação muito mais intuitiva e orgânica.

A título de curiosidade, o termo “Metro Style” teve de ser alterado por decisão jurídica movida pela empresa alemã Metro AG. Por algum tempo, a Microsoft utilizou o termo “Modern UI” para se referir ao “Metro Style”. Ainda foi utilizado o termo “Windows 8 Style” e, finalmente, o definitivo “Store Style”.

4.2.1 Um novo ciclo de vida

O novo sistema operacional Windows 8 foi desenvolvido para se adequar aos dispositivos móveis nos quais venha a ser instalado. Uma dessas adaptações é a necessidade de um novo ciclo de vida para os aplicativos. Supondo que um aplicativo está em execução e um novo é aberto, como apenas um é visualizado por vez na tela, o antigo precisa ser substituído e receber o tratamento adequado. Esse tratamento é controlado automaticamente pelo sistema operacional e está representado na Figura 4.1.

Os aplicativos precisam estar ocupados e ativos apenas quando são visíveis ao usuário, uma vez que não mais o são, deve-se averiguar se há necessidade de mantê-los em funcionamento em plano de fundo, pois estarão consumindo recursos que poderiam ser usados pela aplicação ativa. Esse ciclo de vida é pensado para conservar a bateria e otimizar a vida da bateria e o uso de recursos.

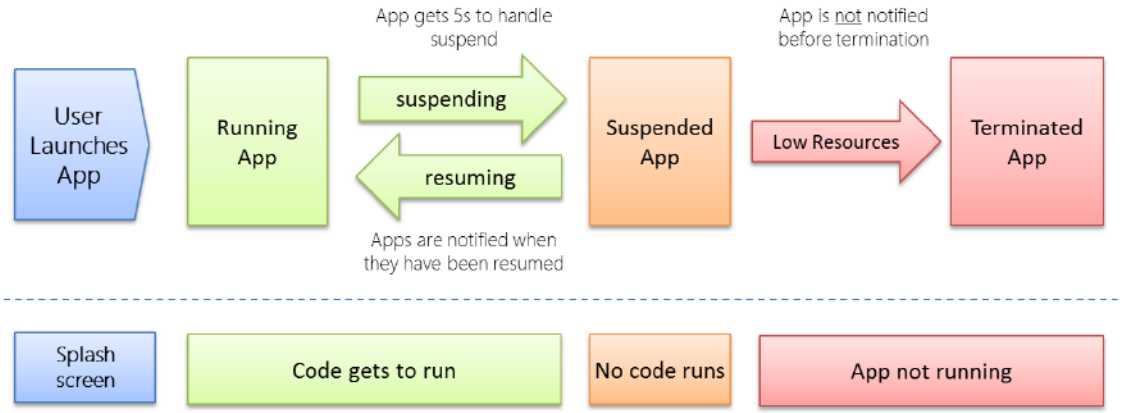

Figura 4.1: Ciclo de vida de uma aplicação Windows Store Style

(BROCKSCHMIDT, 2012)

De acordo com Brockschmidt (2012), quando o programa é posto em plano de fundo, o sistema operacional suspende seu funcionamento por um período de aproximadamente 5 segundos, mas não o remove da memória. Durante esse período, são enviadas notificações e eventos para o aplicativo informando que ele poderá ser desativado. A aplicação deve tratar esses eventos para salvar o valor atual de suas variáveis e qualquer outra informação sensível que precise ter seu estado salvo. Se for reconhecida uma falta de recursos pelo sistema operacional, o aplicativo em plano de fundo é finalizado, sem que sejam lançados eventos ou notificações.

Se o usuário trouxer o aplicativo de volta ao funcionamento, há duas possibilidades:

- Se o programa está suspenso, são enviados eventos indicando que será retomado e deve carregar as informações salvas durante a suspensão;
- Se o programa foi terminado, não está mais alocado em memória e deve iniciar todos os seus processos do início.

Dessa forma, o uso do processador e memória são otimizados para que o aplicativo ativo possa usá-los de maneira adequada e suficiente.

4.3 Arquitetura de desenvolvimento Windows 8

Ao desenvolver aplicações, normalmente se usa um conjunto de tecnologias que trabalham juntas. A plataforma de desenvolvimento Windows disponibiliza muitas ferramentas, técnicas e tecnologias, para criar aplicativos.

Para criar programas para Windows 8, existe um novo modelo de desenvolvimento, um novo tipo de aplicação: Windows Store Style App. Esse novo modelo suporta muitas linguagens de programação (C++, C#, Visual Basic, Javascript) e ainda mantém o número de componentes das tecnologias relacionadas baixo.

Os arquitetos responsáveis pelo Windows 8 tomaram uma decisão importante quando criaram uma arquitetura totalmente nova, disponibilizando duas formas de desenvolvimento que executam lado a lado, Desktop e Windows Store Style. Contudo, tanto Windows Store Style quanto Desktop usam o mesmo núcleo de sistema operacional como informado na Figura 4.2.

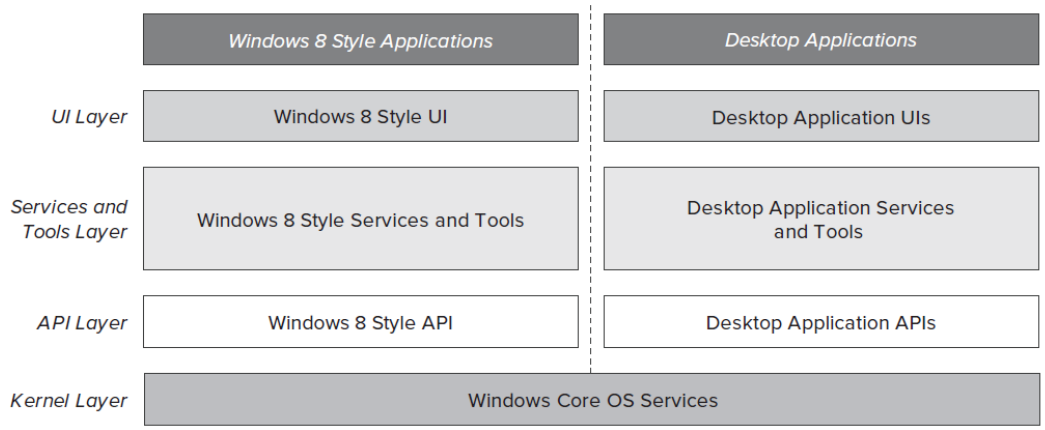

Figura 4.2: Pilhas de componentes coexistindo lado a lado (NOVÁK, 2012)

4.3.1 Windows Store Style Application

A escolha de criar uma nova arquitetura para desenvolver aplicativos Windows 8 está envolta em desafios: criar aplicações com interfaces responsivas, intuitivas e multitoque é apenas um desafio, o maior deles é estabelecer uma plataforma para desenvolvimento de maneira correta, usando tecnologias e ferramentas conhecidas e, o mais importante, ser uma plataforma que proporcione produtividade aos desenvolvedores.

A Microsoft tem uma forte comunidade que desenvolve usando a tecnologia Windows. Um grande número de desenvolvedores experimentou a preocupação de desenvolver TI voltada para o usuário com as aplicações web. Entretanto, poucos utilizam a plataforma Windows Phone, a qual é uma plataforma voltada para o usuário da Microsoft até o lançamento do Windows 8.

De acordo com WMPoweruser (2012) e Winphone (2012), a quantidade de aparelhos telefônicos que executam o Windows Phone, que é 2% do Marketshare mundial e 14,9% no Brasil, ainda é baixa para incentivar o desenvolvimento de aplicações em massa, enquanto muitos já desenvolvem para plataformas Android e iOS.

A Microsoft percebe que chegou ao mercado muito tarde na disputa para ganhar consumidores. Criar dispositivos e seu próprio sistema operacional é um importante passo para entrar na competição, mas sem uma plataforma de desenvolvimento e estratégia, não é possível garantir aplicativos bons o suficiente para fortalecer o Windows e oferecer possibilidades de qualidade para os consumidores.

Microsoft oferece uma grande IDE, o Visual Studio, que está constantemente evoluindo, e também a opção de várias linguagens de programação para desenvolver, diferente da Google e Apple que só oferecem uma linguagem. Essa é uma maneira de conseguir uma melhor posição de mercado e atrair novos clientes e desenvolvedores.

Criando uma pilha independente de componentes em camadas para aplicações Windows Store Style, a Microsoft introduziu um novo conceito que soluciona os problemas do desenvolvimento tradicional para aplicações Desktop e reimaginou a ideia da API Windows e linguagens em tempo de execução. Esse novo conceito adota o suporte para múltiplas linguagens sobre uma única API de programação como na figura abaixo.

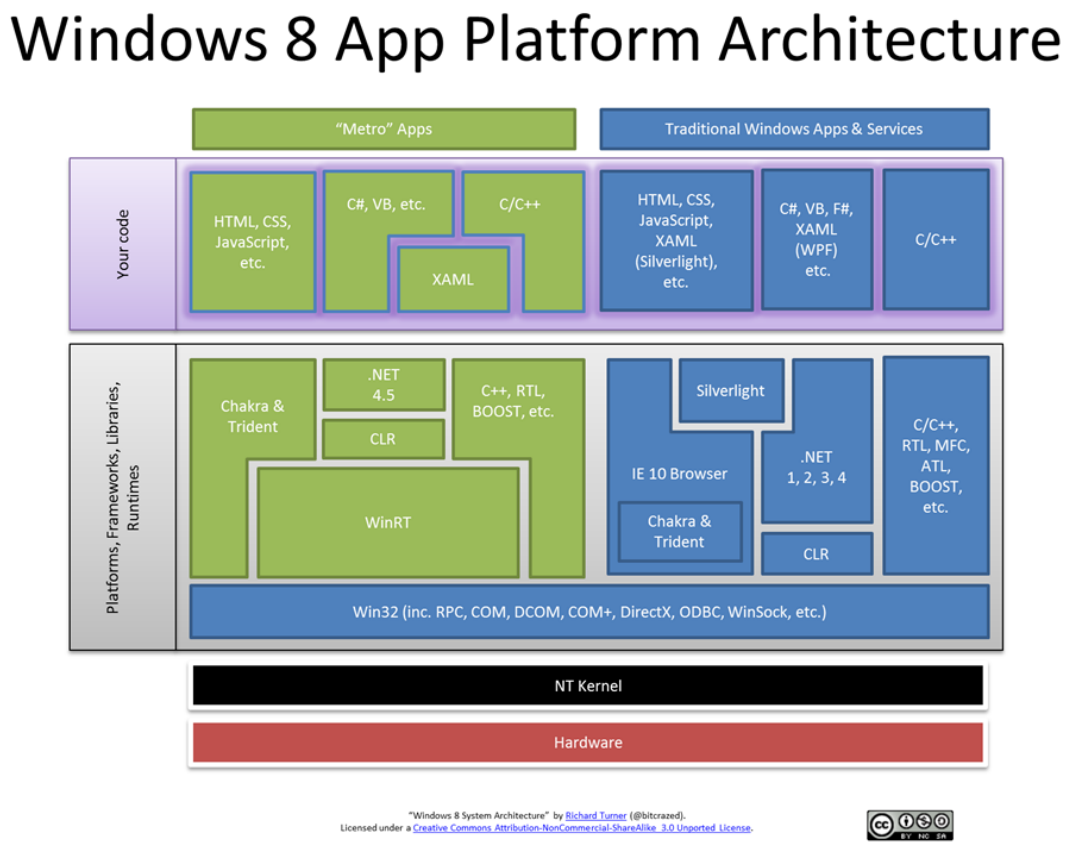

Figura 4.3: Arquitetura das aplicações no Windows 8 (XAML, 2012)

Algumas considerações importantes podem ser derivadas da Figura 4.3. Todas as aplicações Windows Store Style possuem uma única camada de API - Windows Runtime - para se comunicar com os serviços de núcleo do sistema operacional Windows. Independentemente da linguagem de programação usada, todos os serviços estão disponíveis sem limitação alguma, então a API pode ser acessada por C++, C#, Visual Basic ou JavaScript.

HTML e CSS são usados por milhões de desenvolvedores ao redor do mundo para criar páginas web. Esses desenvolvedores usam JavaScript para adicionar lógica de interface para suas páginas. No modelo de aplicação do Windows 8, o potencial do JavaScript foi aumentado ao ter acesso completo a API oferecida pelo Windows Runtime, usando a camada Chakra (que processa as chamadas JavaScript) e Trident (que renderiza o HTML).

A preferência por uma linguagem de programação apenas determina qual tecnologia escolher para criar interfaces. Enquanto JavaScript está associada a HTML, as outras linguagens (C++,C# e Visual Basic) estão ligadas a XAML. É possível utilizar as tecnologias de UI lado a lado como, por exemplo, um controle escrito em XAML pode ter uma página HTML embutida e totalmente funcional.

4.4 Windows Runtime

Um problema comum envolvendo interfaces gráficas é o "travamento", o qual normalmente ocorre quando uma operação de processamento muito demorada está executando em plano de fundo ou um acesso a recursos, como disco, é muito intenso. Segundo Novák (2012), para resolver esse problema, o Windows Runtime tem uma regra principal: "Todas as operações que levem mais do que cinquenta milissegundos para executar são implementadas como operações assíncronas", ou seja, a interface não deve ser bloqueada ao ponto de ser percebida pelo usuário, criando uma experiência de interação muito mais fluída e dinâmica.

Dentre as principais características do Windows Runtime, pode-se destacar:

- O acesso ao hardware como, GPS, sensores e câmera, é muito facilitado para os desenvolvedores, podendo realizar tarefas com poucas linhas de código. Na Figura 4.4 é mostrado um exemplo em C# de como capturar uma foto usando a câmera do dispositivo e mostrá-la em um controle *Image*, um dos muitos controles disponíveis ao desenvolvedor;

```
async void TirarFoto(object s, RoutedEventArgs e)
{
    var interfaceDeCaptura = new Windows.Media.Capture.CameraCaptureUI();

    var arquivoDeFoto = await interfaceDeCaptura
        .CaptureFileAsync(Windows.Media.Capture.CameraCaptureUIMode.Photo);

    AreaDeFoto.Source = new BitmapImage(new Uri(arquivoDeFoto.Path));
}
```

Figura 4.4: Exemplo de método para utilizar a câmera

- As aplicações executam em um ambiente seguro. Qualquer acesso a recursos do dispositivo deve ser permitido explicitamente pelo usuário, essa permissão é adquirida por meio de caixas de diálogo que solicita a confirmação do usuário;
- O Windows Runtime faz parte integrante do sistema operacional, tornando-o mais estável e melhorando o gerenciamento de memória, diferente de seu antecessor Win32 API que era uma camada acima do sistema operacional;
- Interfaces responsivas exigem operações assíncronas, essas são suportadas nativamente pelo Windows Runtime.

5 MODELAGEM DO SISTEMA

Este capítulo focará a modelagem proposta para o sistema de apoio ao personal trainer, chamado SharpGym.

5.1 Identificação do problema

De acordo com Negócio (2012), o Brasil ocupa a 2ª colocação no ranking dos países com o maior número de academias do mundo, fato que demonstra que a procura por esse tipo de atividade é muito grande no país.

O profissional de treinamento personalizado, também chamado de professor, oferece um trabalho muito mais focado nos resultados de seus clientes, alunos, procurando suprir as necessidades individuais e acompanhar de perto o rendimento de seu treino.

Para facilitar e agilitar o trabalho do professor é necessário automatizar alguns processos internos das academias, como o processo de avaliação e registro de atividades. Além de automatizar, também é importante que esses dados estejam facilmente acessíveis aos profissionais e com alta disponibilidade. Dessa forma, o tempo gasto em preencher planilhas e fichas de papel, o espaço para armazená-las e o tempo que o professor precisa dedicar para calcular e verificar o desempenho de seus alunos pode ser substituído pelo sistema SharpGym, proporcionando mais tempo disponível para planejar o treino e acompanhar o resultado dos alunos.

Inicialmente o sistema foi projetado para um profissional específico, Márcio Soares CREF - 4528 G/RS, personal trainer desde 2002 (MÁRCIO, 2012). Assim, as definições do sistema foram criadas para se adequar à realidade de sua academia, mas as definições são comuns entre a maioria dos estabelecimentos e poderiam ser adaptados de acordo com a necessidade.

5.2 Requisitos do sistema

Os requisitos a seguir foram formados a partir de entrevistas com o profissional Márcio Soares. Tais entrevistas eram realizadas semanalmente, durante um período de dois meses, nas quais houve o levantamento dos principais processos e elementos envolvidos no modelo de negócio exercido no estabelecimento. O aperfeiçoamento dos requisitos se deu por meio de observações durante o treino de alguns alunos voluntários, de tal forma que as atividades físicas eram realizadas e a maneira com que o professor fazia anotações puderam ser notadas em seu ambiente natural e cooperaram para o desenvolvimento do sistema.

5.2.1 Requisitos funcionais

O sistema deve automatizar uma série de atividades comuns para o professor, tais atividades formam os requisitos básicos do sistema e serão descritos a seguir:

- Armazenar informações cadastrais, incluindo anamnese, dos alunos;
- Realizar o controle dos horários de entrada e saída dos treinos;
- Registrar os tipos de atividades disponíveis na academia;
- Registrar desempenho em atividades aeróbicas;
- Registrar quantidade de carga, número de séries e repetições dos exercícios de musculação, quando aplicável;
- Registrar informações de frequência cardíaca e pressão arterial durante o treinamento de cada aluno;
- Calcular os índices indicativos de saúde física, como IMC e percentual de gordura, baseados nas medidas corporais coletadas nos alunos;
- Demonstrar a evolução do treinamento com auxílio de gráficos e tabelas;
- Registrar o número de aulas efetivamente presentes dos alunos, para fins de cobrança e permitir que o aluno acesse essa informação;
- Gerar boletim de desempenho, que contém as avaliações físicas realizadas e evolução do treinamento de cada aluno, e permitir ao aluno acessar o boletim.

5.2.1 Requisitos não funcionais

Aqui serão listados alguns requisitos não funcionais que, apesar de não serem essenciais para o funcionamento do sistema, devem ser considerados para a melhor interação com o usuário:

- Usabilidade: o sistema deve apresentar as informações da maneira mais simples possível e de forma que o profissional possa editar, incluir e remover informações agilmente, mais conhecidas como operações CRUD;
- Desempenho: um dos compromissos do Windows8 é proporcionar um sistema altamente responsivo à interação do usuário. Segundo Novák (2012), todas as operações que levem mais de cinquenta milissegundos devem ser realizadas de forma assíncrona, logo a experiência do usuário é algo sem interrupções, proporcionando a impressão de alto desempenho;
- Mobilidade: atualmente existem poucos dispositivos móveis no mercado que possam utilizar o Windows8 em sua totalidade. Um desktop pode ter o Windows8 instalado e usar o sistema. Os requisitos de hardware exigidos para a execução do sistema proposto são os mesmos exigidos para a instalação do Windows8, ou seja, um dispositivo móvel ou desktop que puder executar o sistema operacional Windows 8 também poderá executar o SharpGym;
- Software: para o total funcionamento e exploração das capacidades do sistema, basta ter o Windows8 instalado no dispositivo e mantê-lo atualizado;
- Alta disponibilidade: para enviar e receber informações da nuvem é necessária uma conexão com internet. Esse não é um requisito forte, pois é apenas para fins de sincronização. Se apenas foi possível sincronizar uma vez durante o período deum dia de atividades, isso deve ser suficiente para manter o sistema atualizado tanto no lado cliente quanto no lado servidor.

5.3 Características do sistema

5.3.1 Origem do nome

O nome SharpGym foi criado pensando no núcleo do desenvolvimento do sistema: a linguagem C# (lê-se “C Sharp”) e o seu objetivo de auxiliar o profissional de

treinamento personalizado e o personal trainer que possui uma academia, em inglês “Gym”.

Além de ser o nome da linguagem de programação, o termo “Sharp” pode ser traduzido como “forte”, “afiado”, “exato”, “definido”, entre outras possíveis traduções, e esse é o objetivo principal de quem procura uma academia, ficar com músculos fortes e definidos, ter um condicionamento físico afiado e exato.

Nessa junção de termos e significados, surge o nome SharpGym, cuja identidade visual é mostrada pela Figura 5.1.

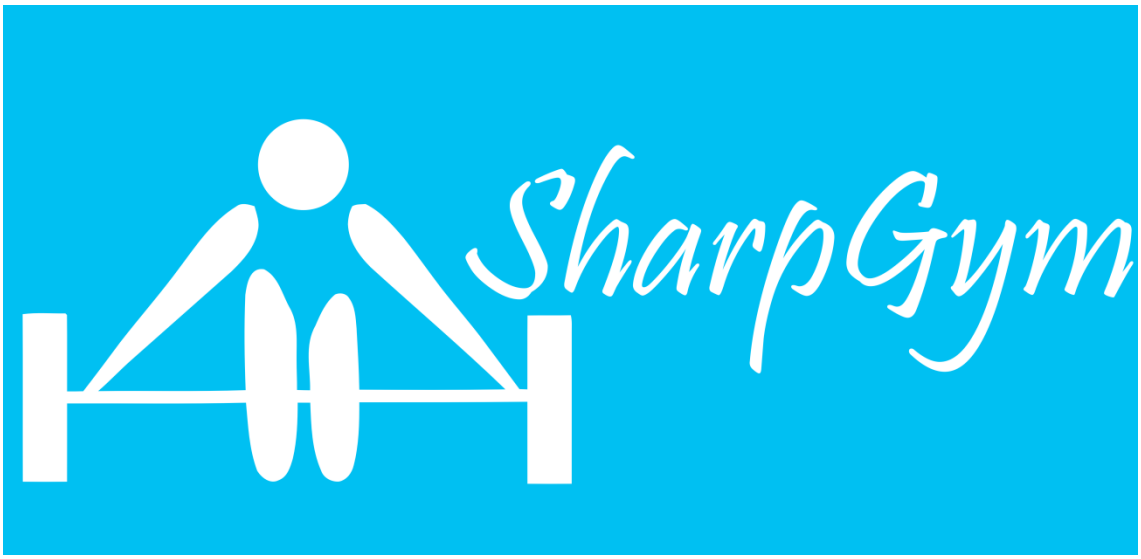

Figura 5.1: Identidade visual do sistema SharpGym

5.3.2 Definições

As definições do sistema são resultado de uma série de entrevistas com um personal trainer de forma a se adaptar às suas necessidades e ações dentro de seu ambiente de trabalho. Esse é um sistema altamente personalizável e que deve se adaptar às necessidades de novos clientes.

O professor é o responsável total pela organização e orientação do treino de seus alunos, que por sua vez devem realizar uma série de exercícios, aeróbicos e de musculação, de acordo com o seu objetivo previamente definido, como emagrecimento, hipertrofia e condicionamento físico. Esse objetivo é modificável conforme o desempenho nas avaliações que serão desenvolvidas durante o treinamento.

Cada academia tem o seu conjunto de aparelhos de ginástica e isso deve ser levado em consideração, portanto, o conjunto apresentado pelo sistema deve ser dinâmico para se adaptar também a possíveis modificações, por exemplo, dois aparelhos que foram substituídos por outro que une as funções de ambos.

Devem-se considerar os métodos avaliativos existentes para verificar o rendimento do aluno, nesse sistema o Protocolo de Avaliação Faulkner (1968) é o escolhido.

Faulkner (1968) utiliza a medida de gordura verificada em quatro dobras cutâneas e aproxima a taxa de gordura corporal, também aproximando a taxa de massa magra

(musculatura). O cálculo a seguir demonstra como calcular o percentual de gordura segundo Faulkner, no qual DC é a sigla para Dobra Cutânea, medida em milímetros com o auxílio de um adipômetro.

$$PercentualDeGordura = (TrícepsDC + EscapularDC + SupraIlíacaDC + AbdominalDC) \times 0.153 + 5.783$$

O treino só é completo se tiver como foco principal a saúde do aluno, então também são salvas informações sobre pressão arterial e frequência cardíaca, que também podem mostrar indícios de problemas cardíacos.

O aluno recebe o resultado de sua avaliação como forma de incentivo para prosseguir com o treinamento. Essa avaliação também pode representar um aviso para que se tomem medidas de cuidado á saúde, por exemplo, se o percentual de gordura estiver acima de 26% para homens ou 33% para mulheres (considerados muito ruins de acordo com Pollock e Wilmore (1993)).

5.3.1 Segurança

O banco de dados do Windows Azure garante que apenas os serviços WCF hospedados na nuvem tenham acesso aos dados devido às permissões dos usuários que são configuradas diretamente no SQL Azure.

Como o acesso aos serviços só é disponível através do endereço do WCF, se esse endereço não for conhecido, não há como invocar os serviços e, assim, acessar aos dados.

Maiores dispositivos de segurança não foram exigidos para essa modelagem, então não serão abordados nesse trabalho.

5.4 Modelagem e arquitetura do sistema

A modelagem e a arquitetura têm o objetivo de apresentar uma abstração dos componentes do sistema, o conjunto de funcionalidades que são encapsuladas e que assumem algum tipo de responsabilidade dentro do sistema e seus conectores. Esses conetores são responsáveis pela coordenação e comunicação entre os componentes.

5.4.1 Arquitetura geral do sistema

O sistema é definido em uma arquitetura cliente-servidor. O cliente poderá ser acessado por meio de um dispositivo que execute o sistema operacional Windows 8 ou ainda através de uma interface online. Ambos os clientes podem gerenciar todas as informações referentes ao treino dos alunos na academia. O servidor será responsável por armazenar o banco de dados do sistema e fornecer serviços que possam acessar e modificar as tabelas específicas para cada tipo utilizado na modelagem do sistema. Uma análise mais detalhada do banco de dados será apresentada em breve.

Na Figura 5.2, podem-se visualizar os componentes principais do sistema e como eles interagem. Para facilitar o entendimento da arquitetura, ela será dividida em módulo cliente e módulo servidor.

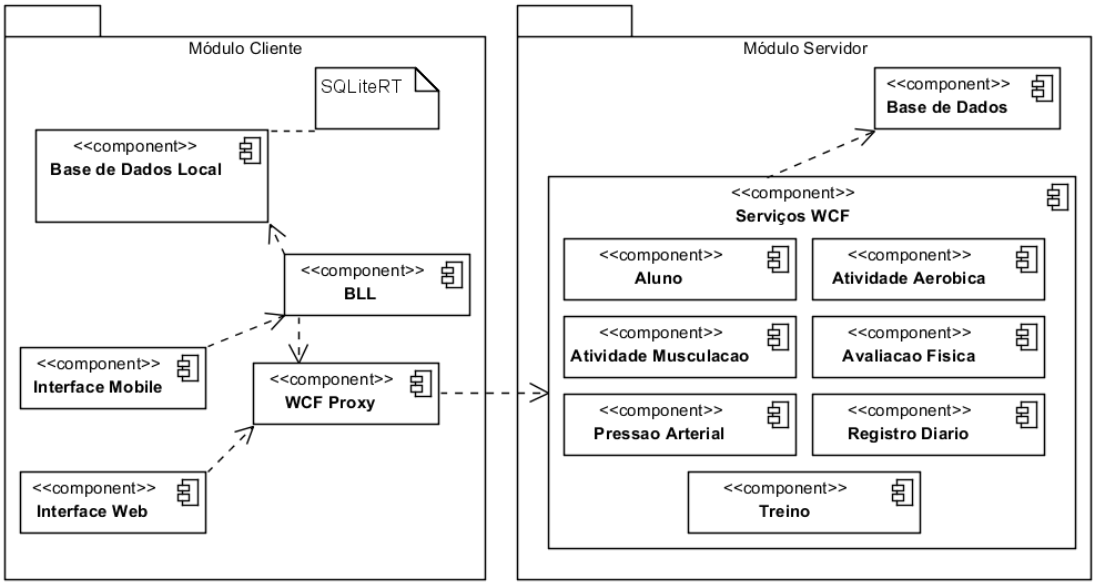

Figura 5.2: Arquitetura geral do sistema

5.4.1.1Módulo Cliente

A partir da Figura 5.2 será explicado cada um dos componentes do módulo cliente:

- Interface mobile: interface fornecida pelo sistema operacional Windows 8, que está disponível para tablets e desktop;
- Interface web: alternativa com as mesmas funcionalidades da interface mobile;
- Base de dados local: base de dados utilizada pela interface mobile para persistir as informações no dispositivo;
- WCF Proxy: componente necessário para comunicação entre cliente e serviços conforme explicitado na subseção 3.4.1;
- BLL: componente que reúne todas as operações disponíveis pelo proxy do serviço, mas com o adicional de testar a conectividade do dispositivo, dividindo as operações em dois casos:
  - Caso não haja conectividade, a BLL realiza a modificação no banco de dados local e indica que houve alteração por uma coluna do banco de dados com valor booleano verdadeiro em conjunto com a data da última alteração;
  - Caso haja conectividade, a BLL realiza a modificação localmente e imediatamente chama o serviço responsável por atualizar o valor no banco de dados do servidor.

5.4.1.2Módulo Servidor

- Base de dados: Responsável por controlar o acesso à base de dados, realizar consultas e inserções. Cada tabela no banco de dados representa um tipo do sistema. A base de dados utiliza o SQL Azure como provedor de serviço;
- Serviços: os serviços estão hospedados no servidor de Computação em Nuvem Windows Azure.

5.4.2 Funcionalidades

Todas as funcionalidades disponíveis no sistema estão representadas na Figura 5.3.

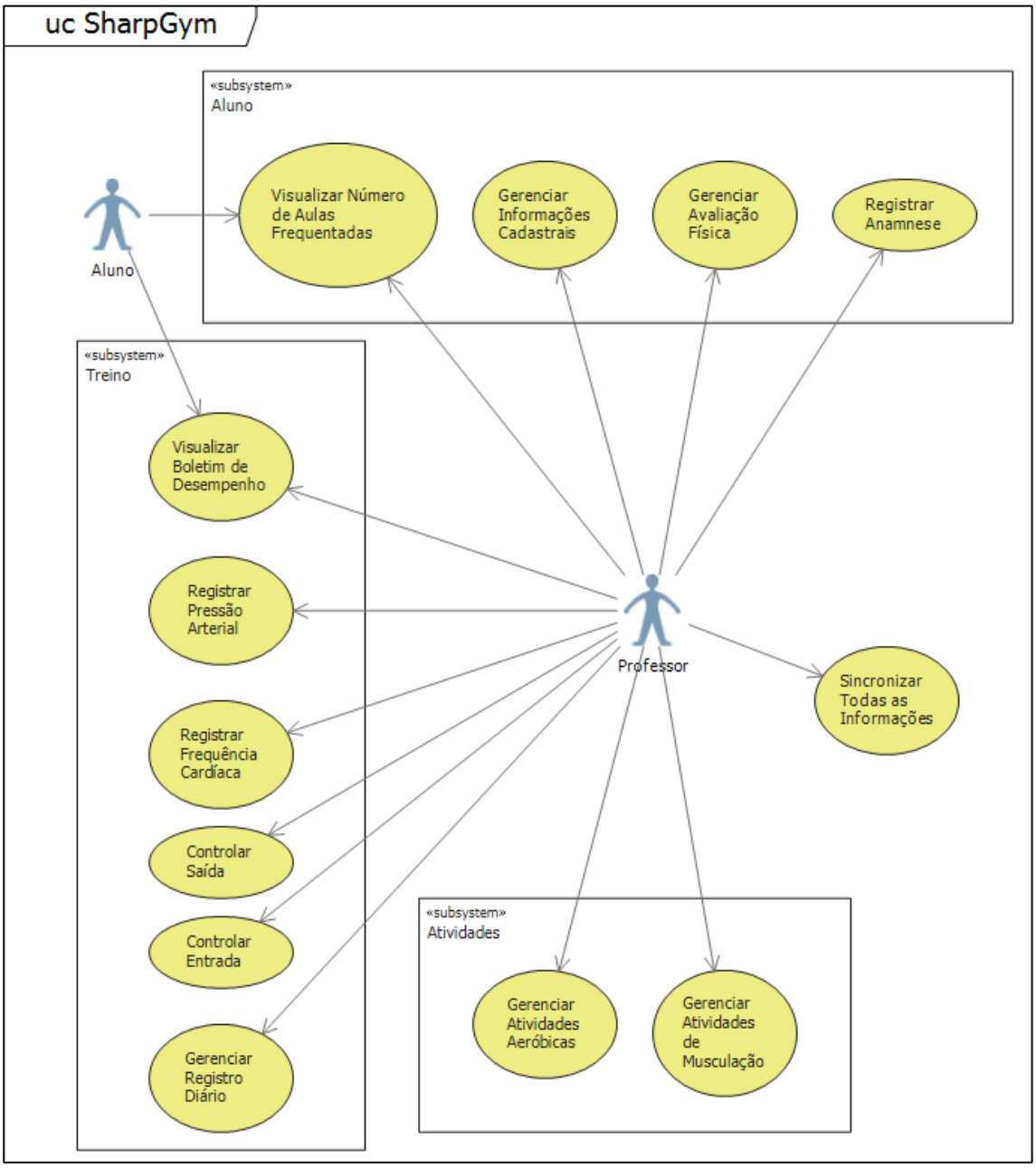

Figura 5.3: Diagrama de Caso de uso para o sistema

Como pode ser notado, o professor representa o centro de todas as operações do sistema, pois ele é o operador do dispositivo e tem acesso total às funcionalidades disponíveis. Alguns casos de uso são compartilhados com o aluno também, mas o seu acesso às funções é muito limitado.

A seguir será feita uma pequena descrição de cada um dos casos de uso do sistema, que foram reunidos em subsistemas para melhor organizar a visualização.

Controlar Entrada

O professor está preocupado com o tempo disponível para cada aluno, então é importante registrar a hora de entrada dos alunos por meio de um controle específico. Para o módulo cliente, esse controle também serve para filtrar os alunos que estão presentes e encontrar as informações do treino individual. Esse caso de uso é executado quando um aluno entra na academia e se prepara para começar o treino.

Controlar Saída

Da mesma forma que o controle de entrada é importante, o controle de saída serve para diminuir a lista de alunos presentes e auxiliar o professor a escalonar melhor os alunos dentro do horário disponível. Esse caso de uso é executado quando o aluno já concluiu o seu treino e está pronto pra deixar o estabelecimento.

Registrar Anamnese

A anamnese, entrevista inicial, é um elemento importante para a definição do tipo de treino que é mais adequado para um indivíduo. Nela se encontram perguntas relacionadas ao histórico de lesões ou fraturas (que necessitam de cuidados especiais para não sobrecarregar e causar dano), histórico familiar de doenças cardíacas (informação que auxilia na detecção de algum nível de deficiência cardíaca), nome, data de nascimento (para cadastro) e outros dados relevantes para cada profissional.

Registrar Frequência Cardíaca

Para uma verificação do condicionamento físico do aluno, é necessário realizar medição da frequência cardíaca e anotá-las propriamente. Cada pessoa tem uma faixa de frequência cardíaca na qual a queima de calorias é máxima, essa faixa deve ser buscada durante o treinamento para maximizar a perda de gordura corporal.

Registrar Pressão Arterial

Ao chegar e ao sair da academia, a pressão arterial é aferida para que não se torne um fator de risco ao aluno, pois iniciar o treino com a pressão muito elevada pode causar danos à saúde. Também a pressão muito baixa pode diminuir o desempenho durante as atividades, pois o aluno sente seu corpo mais pesado e menos disposto para efetuar suas tarefas.

Sincronizar Todas as Informações

Caso o dispositivo fique por muito tempo sem conectividade e forem realizadas muitas alterações na base de dados local, é necessário que haja uma sincronização de todas as informações do dispositivo com o banco de dados do servidor, pois não deve haver conflitos de informações. Se houverem conflitos, esses deverão ser resolvidos manualmente pelo professor, mas com o auxílio do sistema.

Visualizar Boletim de Desempenho

Para que seja gerado um boletim de desempenho, as informações referentes às avaliações físicas e ao registro diário são coletadas e processadas para formar um relatório com a evolução do aluno nas suas atividades e se os seus objetivos estão sendo alcançados. Esse é um dos casos de uso que estão acessíveis ao aluno.

Visualizar Número de Aulas Frequentadas

O número de aulas efetivamente frequentadas tem dois objetivos. Para o aluno, serve para fins de simples conferência. Para o professor, tem como objetivo calcular o valor da mensalidade que será pago, pois nessa modelagem proposta o valor da mensalidade é proporcional ao número de treinos realizados.

Os casos de uso a seguir são basicamente operações CRUD, as quais permitem manipular as tabelas das bases de dados.

Gerenciar Atividades Aeróbicas

Como atividade aeróbica são consideradas aquelas atividades que aumentam os batimentos cardíacos e exijam que o aluno respire mais vezes devido à necessidade de maior oferta de oxigênio na corrente sanguínea causada pela respiração celular. Alguns exemplos de atividades aeróbicas são a corrida na esteira e bicicleta. Caso um aparelho seja substituído ou um novo seja incluído, como um aparelho de remo, é necessário gerenciar as atividades aeróbicas disponíveis.

Gerenciar Atividades de Musculação

Diferente das atividades aeróbicas, as atividades de musculação têm por objetivo definir ou aumentar (hipertrofiar) um grupo específico de músculos, como por exemplo, bíceps e tríceps. Essas atividades sofrem o mesmo tipo de alteração das atividades aeróbicas, sempre que novos aparelhos ou modalidades forem incluídos, o sistema deve refletir essa mudança.

Gerenciar Avaliação Física

O professor deve definir um período de tempo entre uma avaliação física e outra para que se notem os resultados do treinamento. Nessa avaliação física são realizadas várias medidas corporais no aluno, incluindo peso, altura, perímetros musculares e gordura localizada. A partir dessas medidas, são calculados índices indicativos de saúde, como IMC, taxa de gordura e relação cintura/quadril, esses índices norteiam os próximos treinos e demonstram o rendimento do aluno.

Gerenciar Informações Cadastrais

Durante a anamnese, são coletadas informações cadastrais que são salvas no dispositivo: nome, telefone, data de nascimento e, opcionalmente, uma foto para personalizar a visualização dos alunos no sistema.

Gerenciar Registro Diário

O registro diário é uma ferramenta na qual o professor pode anotar a frequência cardíaca inicial e final do aluno, também pode registrar a pressão arterial em vários momentos do treino.

5.4.3 Banco de dados

O diagrama do banco de dados completo está ilustrado na Figura 5.4 e a descrição de cada tabela e campo pode ser encontrada no Anexo A.

Para essa modelagem não há uma tabela para o professor, pois o modelo proposto conta com apenas um personal trainer que administra sua academia. Para ambientes com vários profissionais, deve-se criar uma tabela Professor e criar uma associação com Aluno. A cardinalidade da associação dependerá do modelo de trabalho da academia, ou seja, se treino de um aluno é responsabilidade de um ou de mais professores.

A representação de cada tabela pode ser vista na Tabela 5.1.

Tabela 5.1 Tabelas do banco de dados e seus significados

Tabela	Significado
Aluno	Cada linha dessa tabela representa um aluno. A linha contém colunas que são os dados cadastrais do aluno. Um lado pode ter um ou mais avaliações físicas, registros diário ou treinos.
AtivMusculacao	Tabela que representa as atividades de musculação disponíveis pela academia. Contém nome, carga, número de séries e repetições do exercício.
AtivAerobica	Tabela que representa as atividades aeróbicas disponíveis pela academia. Contém nome, distância, duração e frequência cardíaca máxima durante o exercício.
AvaliacaoFisica	A maior tabela do sistema contém campos para todas as medidas feitas pelo professor durante a avaliação, além de um espaço para observações sobre desempenho.
PressaoArterial	A pressão arterial é composta por três colunas: diastólica, sistólica e horário. Por

	ser um tipo composto, necessita de uma tabela própria de acordo com a normalização de tabelas.
RegistroDiario	O registro diário contém a frequência cardíaca inicial e final do aluno, horário de entrada e saída e um ou mais registros de pressão arterial.
Treino	Cada treino tem um campo para observações, a data do treino e uma ou mais atividades de musculação ou aeróbica de acordo com a orientação do professor.

5.5 Casos de uso

A seguir alguns casos de uso na forma descritiva para exemplificar melhor a interação do usuário com o sistema.

5.5.1 Controlar Entrada

Nome: Controlar Entrada.

Atores: Professor.

Finalidade: Registrar a entrada de um aluno na academia e adicioná-lo à lista de alunos presentes.

Visão geral: Esse caso de uso se inicia quando um novo aluno chega à academia e o professor solicita o controle da entrada.

Tipo: Essencial.

Tabela 5.2: Caso de uso Controlar Entrada

5.5.2 Sequência de eventos

Ação do ator	Resposta do sistema
1. O professor, ao notar a entrada de um aluno, solicita o controle de entrada de alunos.	
	2. O sistema mostra a lista de alunos e um campo para filtrar por nome
3. O professor digita o nome do aluno desejado	4. O sistema filtra os alunos listados enquanto o nome é digitado
5. O professor encontra o aluno desejado	

6. O professor seleciona o aluno	
	7. O sistema marca o aluno com “aluno presente” e adiciona-o à lista de alunos presentes.

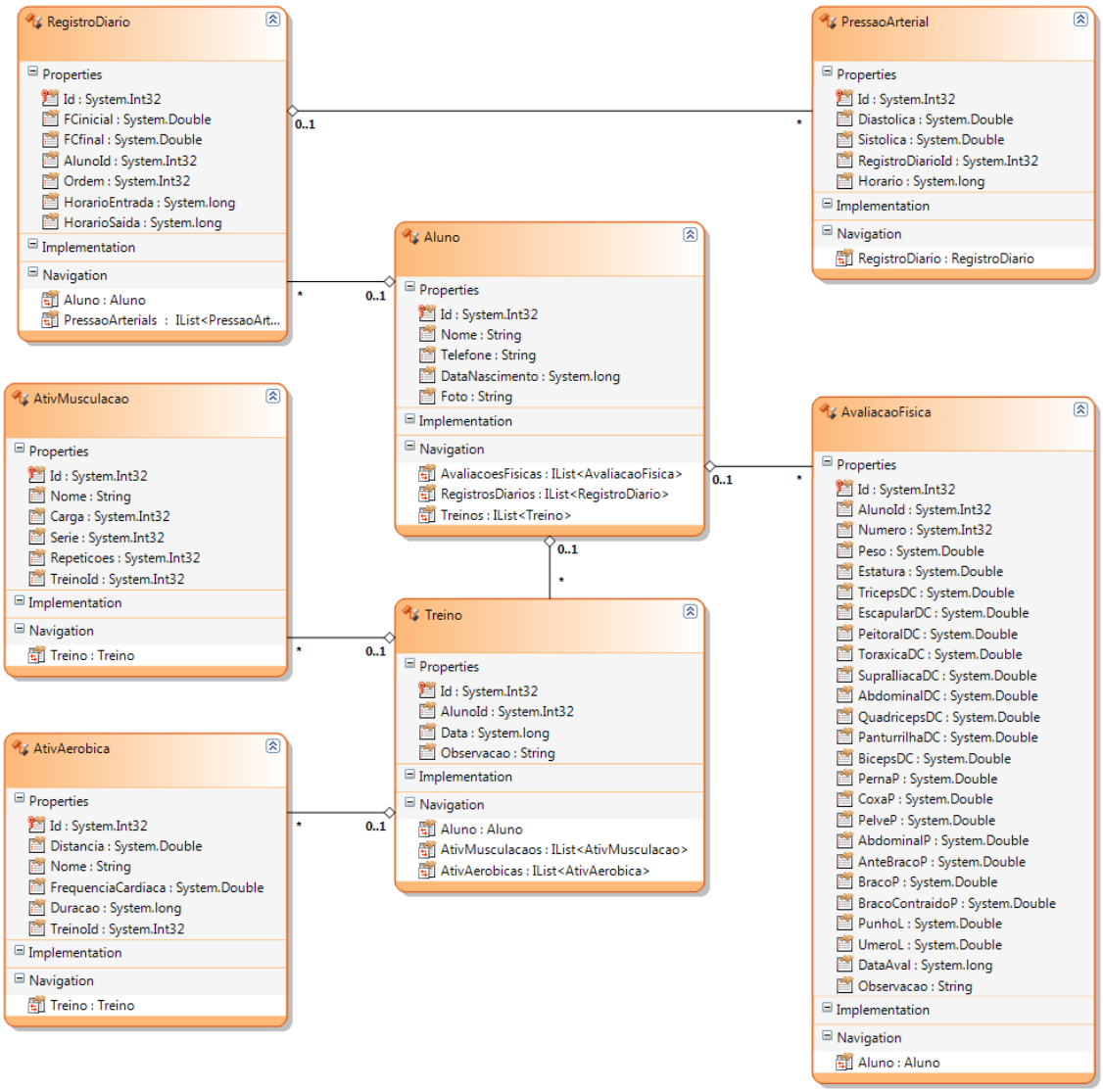

Figura 5.4: Diagrama de Banco de Dados

5.5.3 Sincronizar Todas as Informações

Nome: Sincronizar Todas as Informações.

Atores: Professor.

Finalidade: Atualizar o banco de dados local com possíveis alterações do servidor e enviar ao servidor as alterações locais.

Visão geral: Esse caso de uso se inicia quando o professor solicita a opção sincronizar na tela inicial.

Tipo: Essencial.

Tabela 5.3: Caso de uso Sincronizar Todas as Informações

5.5.4 Sequência de eventos	
Ação do ator	**Resposta do sistema**
1. O professor solicita a sincronização na tela inicial do sistema.	
	2. O sistema verifica a conectividade.
	2.1. Caso não haja conexão, informar ao professor e terminar a sincronização, voltando à tela inicial. 2.2. Caso haja conexão, prossegue normalmente.
	3. O sistema gera uma lista com todos os objetos que foram alterados no banco de dados local.
	4. O sistema envia a listagem para o servidor.
	5. O servidor compara a data de modificação dos objetos vindos do dispositivo com os armazenados no servidor.
	5.1. Caso a data do objeto do dispositivo seja mais atual que a do servidor, substitui no servidor. 5.2. Caso contrário, a alteração do dispositivo é descartada.
	6. O sistema gera uma lista com os objetos alterados no servidor, excetos os enviados pelo dispositivo.
	7. O sistema envia a listagem para o dispositivo.
	8. O dispositivo substitui localmente seus objetos pelos respectivos recebidos.

	9. O sistema mostra um aviso de que a sincronização foi um sucesso.
10. O usuário confirma a leitura do aviso.	
	11. O sistema mostra a tela inicial.

5.5.5 Registrar Frequência Cardíaca

Nome: Registrar Frequência Cardíaca.

Atores: Professor.

Finalidade: Registrar uma medição de frequência cardíaca durante o treino.

Visão geral: Esse caso de uso se inicia quando o professor solicita o registro de frequência cardíaca na tela “Treino do Aluno <nome do aluno>”.

Tipo: Essencial.

Tabela 5.4: Caso de Uso Registrar Frequência Cardíaca

5.5.6 Sequência de eventos

Ação do ator	Resposta do sistema
1. O professor solicita o registro de frequência cardíaca na tela “Treino do Aluno <nome do aluno>”.	
	2. Mostra campo específico para inserir a frequência cardíaca medida e o horário da medição.
3. O professor preenche os campos.	
	4. O sistema valida os campos para os tipos permitidos.
	4.1. Caso algum campo esteja preenchido incorretamente, mostra mensagem de erro. 4.2. Caso contrário, prossegue normalmente.
5. O professor termina de preencher os campos.	
6. O professor seleciona a opção “Salvar” na tela.	
	7. O sistema verifica a conectividade do dispositivo.
	7.1. Caso não haja conexão, o objeto é somente salvo

	localmente e sinalizado que foi alterado. 7.2. Caso contrário, salva localmente e invoca o serviço do servidor responsável por adicionar frequência cardíaca. 7.2.1 O servidor recebe a nova frequência cardíaca. 7.2.2 O servidor salva o objeto no banco de dados. 7.2.3 O servidor envia uma resposta confirmando que o objeto foi adicionado ao dispositivo.
	8. Dispositivo recebe a confirmação.
	9. Sistema apresenta uma mensagem de confirmação ao professor.
10. O professor confirma a leitura da mensagem.	
	11. Sistema mostra a tela “Treino do Aluno <nome do aluno>”.

6 PROTOTIPAÇÃO

A partir da modelagem feita no capítulo 5, foi implementado um protótipo com algumas das principais funcionalidades a fim de validar parte do modelo proposto. Para isso, serão apresentados neste capítulo a implementação parcial dos módulos cliente e servidor, com descrição de funcionamento, detalhando alguns pontos mais complexos, assim como as tecnologias utilizadas e os testes realizados.

6.1 Escopo do protótipo

Este protótipo é a demonstração de uma parte do sistema, portanto, é implementada a funcionalidade de Gerenciamento de Informações Cadastrais, que trata da edição, da remoção e da visualização das informações de alunos cadastrados, bem como da inclusão de novos alunos. Foram codificados os módulos cliente e servidor, contemplando aspectos como hospedagem em Nuvem, serviços web, banco de dados e interface com usuário, tanto web quanto mobile.

Os demais casos de uso foram omitidos nesse protótipo, pois uma análise deve ser feita sobre como as informações devem ser mostradas para que a experiência com o usuário seja mais agradável. Entretanto, o Gerenciamento de Informações Cadastrais é semelhante a outros casos de uso, como o Gerenciamento de Registro Diário, dessa forma, os casos de uso relacionados a gerenciamento e registro de informações podem ser validados pelo protótipo. Outros casos de uso envolvem, basicamente, a coleta e processamento de informações, logo, o modelo também pode ser validado.

6.2 Ambiente de desenvolvimento

Para o desenvolvimento do protótipo, foi utilizada a IDE Visual Studio 2012 (ou simplesmente VS2012), ambiente integrado de desenvolvimento fornecido pela Microsoft. O VS2102 possui um conjunto de ferramentas que permite codificar, implantar e testar aplicações feitas para o .Net Framework 4.5.

C# 5.0 foi escolhido como linguagem de programação. Essa linguagem possui uma grande coleção de bibliotecas e documentações oferecidas pelo MSDN.

O módulo servidor de serviços foi implementado utilizando o WCF, plataforma de desenvolvimento para aplicações orientadas a serviços.

A plataforma de Computação em Nuvem selecionada para hospedar os módulos servidores foi a Windows Azure Platform, incluindo o banco de dados que utiliza o SQL Azure.

A interface mobile do módulo cliente foi desenvolvida através do .Net Framework 4.5, seguindo os princípios de *design* do Windows 8, sistema operacional que é pré-requisito para a execução do módulo cliente.

Por fim, a interface web do módulo cliente foi criada em Silverlight5, que é uma extensão para web do .Net Framework. Esse módulo também foi hospedado no Windows Azure para fins de facilitar o controle de acesso de usuários.

6.3 Apresentando o protótipo

Esta seção apresentará a funcionalidade Gerenciamento de Informações Cadastrais. Os aspectos relacionados nesse caso de uso foram testados e codificados integralmente.

6.3.1 Interface web do cliente

Na tela inicial da aplicação web, apresentada na Figura 6.1, temos a apresentação do sistema e a exibição de um menu navegacional, o qual dispõe das opções disponíveis para o gerenciamento de todas as informações fornecidas pelo sistema.

O professor, após acessar a interface web, deve selecionar a opção “alunos”, a qual o levará à página de Gerenciamento de Informações Cadastrais.

Ao navegar para a página, a lista de alunos é carregada e mostrada na coluna à esquerda. Na primeira vez que o serviço de alunos é invocado, é necessário inicializar os canais e proxies que serão utilizados para a comunicação. Enquanto os dados são carregados, é mostrada uma caixa de diálogo indicando a sincronização conforme ilustrado na Figura 6.2.

Um detalhe interessante sobre a chamada dos serviços é que ela é feita de maneira assíncrona, assim, a interface não é bloqueada pela operação, tanto que o professor pode entrar em outras páginas da aplicação que a chamada será completada em plano de fundo e, ao voltar para a página de alunos, a lista estará carregada.

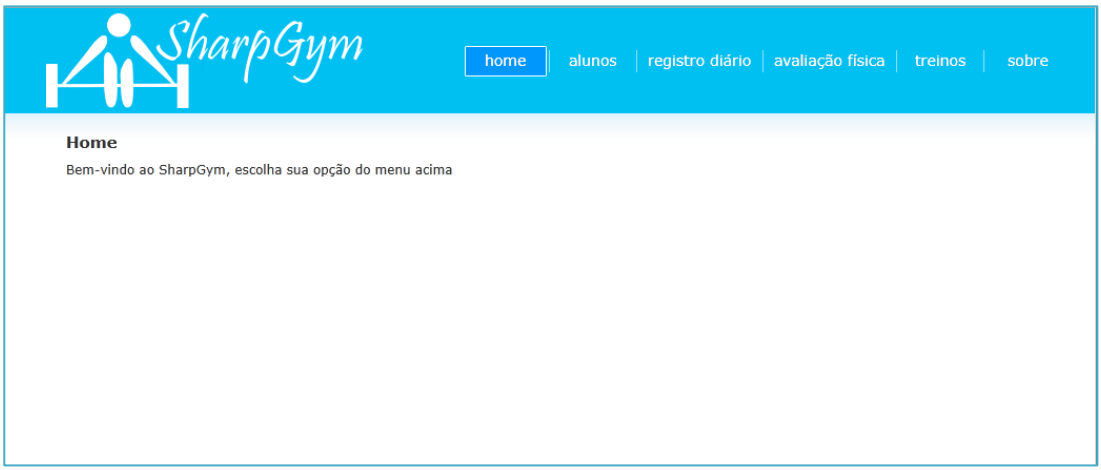

Figura 6.1: Tela inicial da interface web no módulo cliente

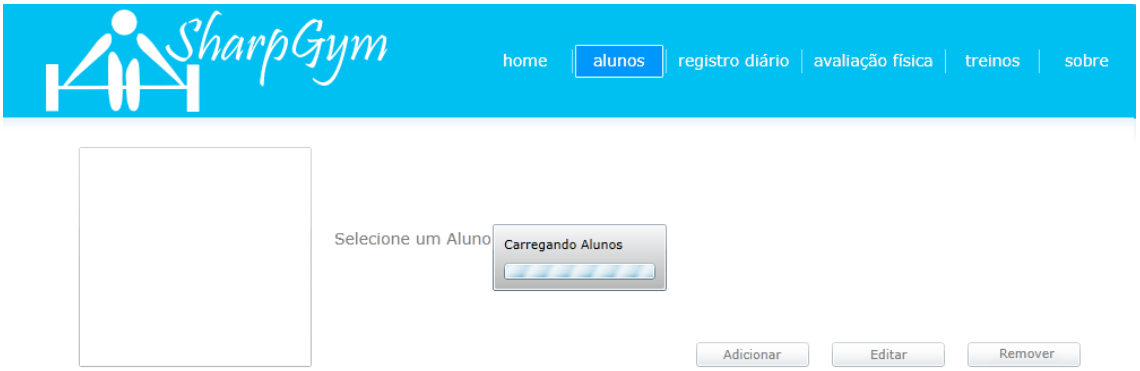

Figura 6.2: Indicativo de sincronização

Para fins de demonstração da funcionalidade em destaque, a barra de menu será ocultada nas próximas figuras.

A lista carregada é semelhante à lista mostrada na Figura 6.3. Quando um aluno é selecionado, automaticamente suas informações cadastrais são visualizadas ao lado da lista como ilustrado na Figura 6.4.

Com um usuário selecionado, duas operações são permitidas: removê-lo da lista ou editar suas informações.

No caso da remoção, é apresentada uma caixa de diálogo para confirmar a exclusão e, caso não haja nenhum aluno selecionado, é apresentada uma caixa de aviso para que um usuário seja selecionado antes da remoção. Se o professor confirmar a exclusão, uma mensagem de operação completada é mostrada e a listagem é atualizada.

Para a edição, uma tela com os campos disponíveis é visualizada para modificar todas as informações do aluno, esses campos vêm preenchidos com as informações atuais conforme exemplificado na Figura 6.5.

A inserção de novos alunos tem uma tela semelhante à tela de edição, porém com os campos em branco para ser adicionado um novo aluno.

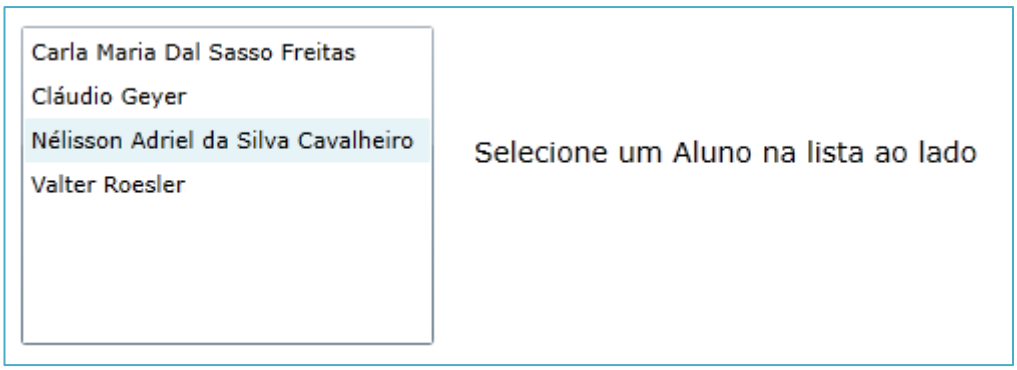

Figura 6.3: Lista de alunos

Figura 6.4: Informações cadastrais do aluno selecionado

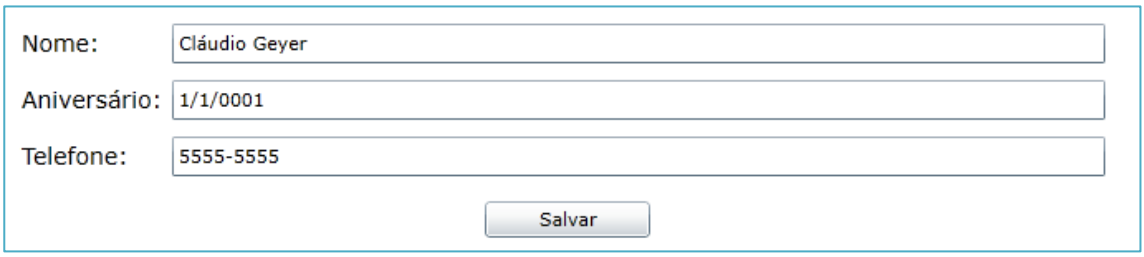

Figura 6.5: Edição das informações cadastrais do aluno selecionado

6.3.2 Interface mobile do cliente

As diretrizes de interface propostas para aplicativos que seguem o Windows Store Style dizem que o conteúdo é o elemento mais importante de uma aplicação, então, diferente do módulo web, que possui uma tela inicial, a interface mobile apresenta os alunos na própria tela inicial de acordo com a Figura 6.6. Os outros menus (treinos, avaliação física e etc.) serão apresentados como grupos ao lado do grupo de alunos, mas para esse protótipo essas funcionalidades foram omitidas.

Uma funcionalidade muito interessante dos aplicativos para Windows 8 é a AppBar, uma barra que pode ser mostrada na parte superior ou inferior da tela que apresenta as funções disponíveis à página atual. Para esse protótipo, a AppBar usada é a ilustrada na Figura 6.7, a qual possui 3 botões: Adicionar, Editar e Remover, assim como na interface web.

Ao selecionar um aluno, a aplicação mostra a tela de detalhes do aluno, na qual as informações aparecem muito mais claras e destacadas comparada à tela de grupo. Nas laterais da tela, são vistas flechas que realizam a navegação entre as páginas de detalhes dos alunos registrados. Essa tela também possui uma AppBar que permite remover ou editar, semelhantemente à interface web, a edição é feita com campos editáveis e depois salvos por um botão “Salvar” ao lado desses campos. Essas características, exceto a edição, são visualizadas na Figura 6.8.

Pode também ser notada uma barra de rolagem na parte inferior da tela, essa barra demonstra que é possível mostrar mais informações à direita da imagem, como descrição do treino e detalhes da última avaliação, entretanto esses detalhes foram ocultados nesse protótipo.

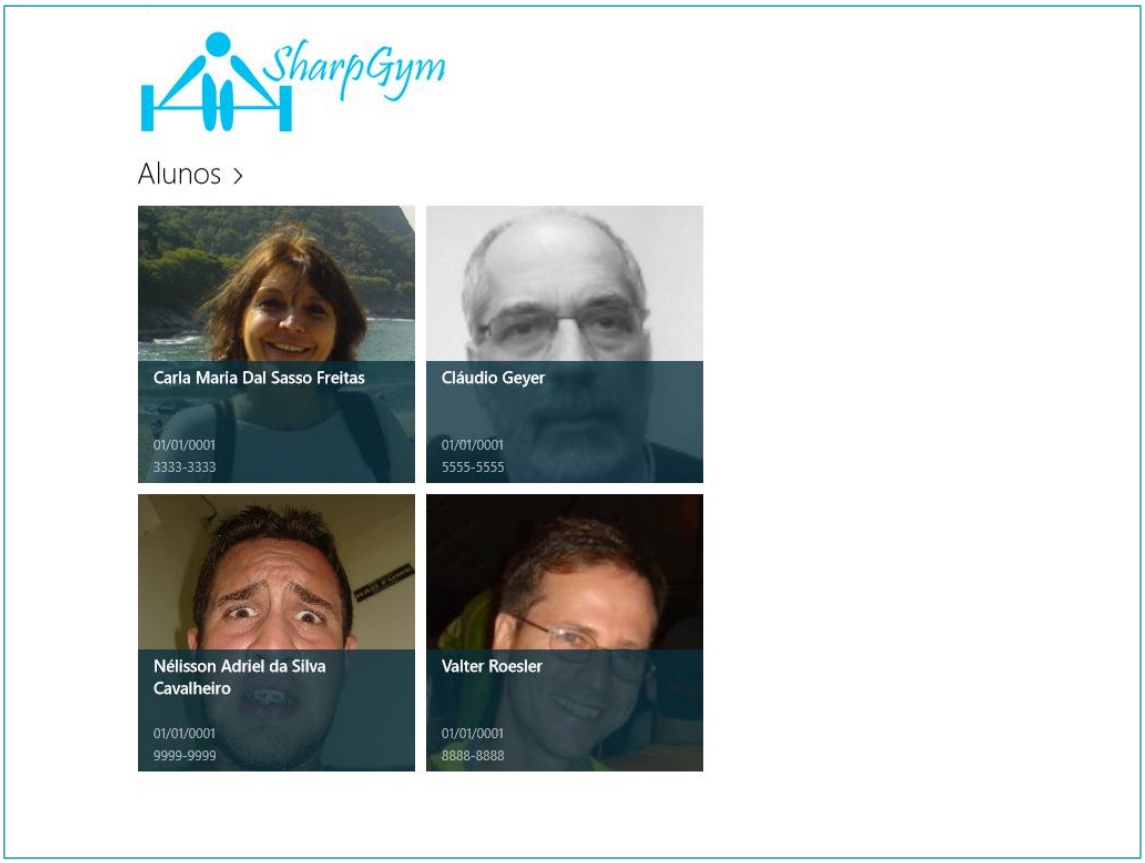

Figura 6.6: Tela inicial da interface mobile

Figura 6.7: AppBar na parte inferior da aplicação

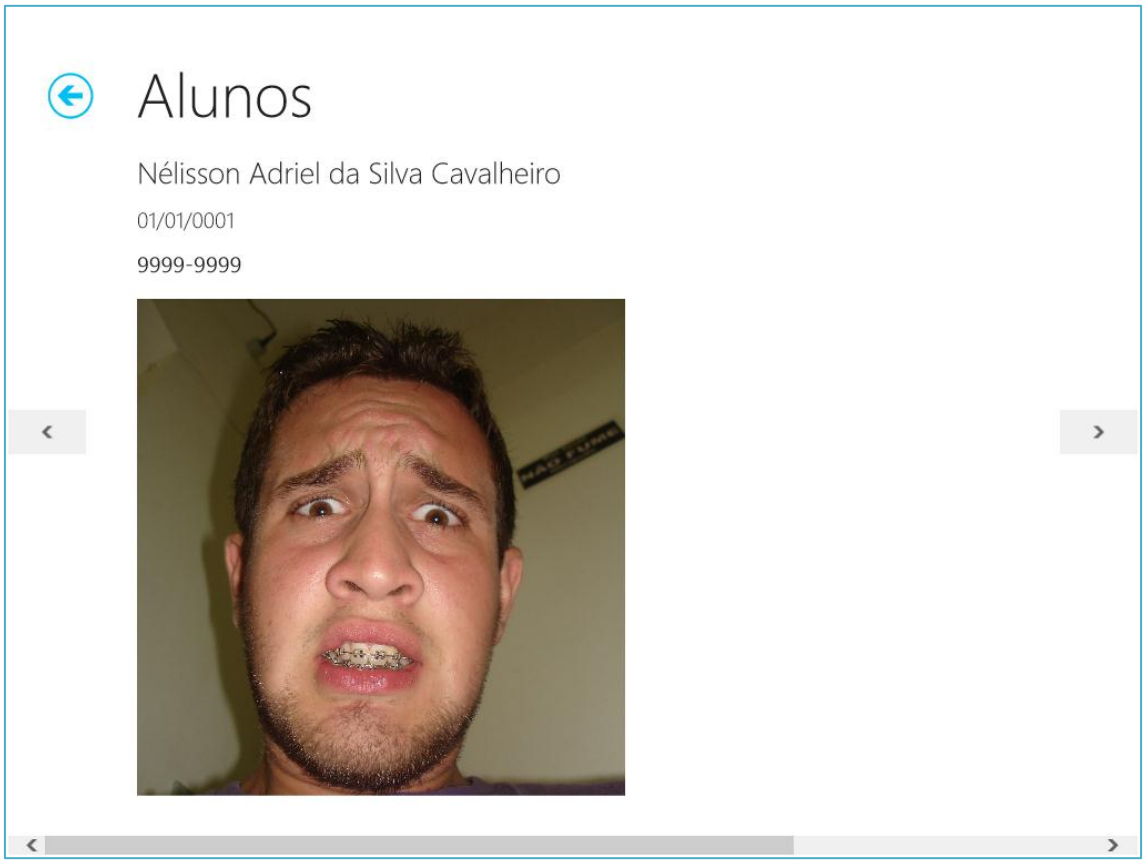

Figura 6.8: Tela de detalhes do Aluno

6.4 Comunicação

A comunicação entre clientes e servidor é feita por um WCF Proxy, esse proxy realiza uma operação de POST para um endereço que contém o serviço e recebe a resposta serializada como XML.

Um exemplo de cabeçalho da chamada POST para o serviço é dado a seguir.

POST /SharpService/SharpGymContextService.svc HTTP/1.1
Host: nelisson.zapto.org
Connection: keep-alive
Content-Length: 136
Origin: http://nelisson.zapto.org
soap action: "http://tempuri.org/ISharpGymContextService/ReadAlunos"
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) Apple WebKit/537.11 (KHTML, like Gecko) Chrome/23.0.1271.97 Safari/537.11
content-type: text/xml; charset=utf-8
Accept: */*
Referer: http://nelisson.zapto.org/SharpGym/ClientBin/SharpGym.Silverlight.xap
Accept-Encoding: gzip, deflate, sdch
Accept-Language: pt-BR, pt; q=0.8, en-US; q=0.6, en; q=0.4
Accept-Charset: ISO-8859-1, utf-8; q=0.7,*; q=0.3

Nessa chamada, algumas linhas foram postas em **negrito**, pois representam, sequencialmente, o endereço do serviço no hospedeiro, o endereço do hospedeiro e a ação (método) que será invocada no serviço.

Para essa chamada, a resposta foi a seguinte:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
    <s:Header>
        <ActivityId CorrelationId="a987384a-24d3-46b7-a365-0d2044755429" xmlns=
        "http://schemas.microsoft.com/2004/09/ServiceModel/Diagnostics">
        09e2c147-b881-452b-804e-8d2bbcdf07fe
        </ActivityId>
    </s:Header>
    <s:Body>
        <ReadAlunosResponse xmlns="http://tempuri.org/">
            <ReadAlunosResult xmlns:a="http://schemas.datacontract.org/2004/07/SharpGym.WCF.Dto" xmlns:i=
            "http://www.w3.org/2001/XMLSchema-instance">
                <a:AlunoDto z:Id="i1" xmlns:z="http://schemas.microsoft.com/2003/10/Serialization/">
                    <a:AvaliacoesFisicas i:nil="true"/>
                    <a:DataNascimento>24/11/1989</a:DataNascimento>
                    <a:DtoKey>Id=1</a:DtoKey>
                    <a:Foto i:nil="true"/>
                    <a:Id>1</a:Id>
                    <a:Idade>23</a:Idade>
                    <a:Nome>Nélisson Adriel da Silva Cavalheiro</a:Nome>
                    <a:RegistrosDiarios i:nil="true"/>
                    <a:Telefone>3054-3848</a:Telefone>
                    <a:Treinos i:nil="true"/>
                </a:AlunoDto>
            </ReadAlunosResult>
        </ReadAlunosResponse>
    </s:Body>
</s:Envelope>
```

Figura 6.9: Resposta do serviço ReadAlunos

Essa resposta segue o padrão envelope de soap e o elemento mais interessante de se analisar é o s:Body, o qual contém a resposta à chamada do método.

Nesse caso, a chamada retornou apenas um aluno, pois é a quantidade registrada no banco de dados. O objeto Aluno foi serializado de uma forma que foi definida por meio do contrato de serviço e é conhecida em ambos os lados da comunicação.

6.5 Hospedagem em Nuvem

O serviço de Computação em Nuvem utilizado é o Windows Azure. Para implantar a aplicação no servidor, está disponível o Windows Azure SDK, o qual é uma extensão para o VS2012 com todas as ferramentas necessárias para implantação e para o teste do módulo servidor.

No painel de controle do Windows Azure foi criado uma conta de armazenamento, um serviço de Nuvem (para hospedagem da interface web e serviços) e um banco de dados SQL conforme visto na Figura 6.10. O VS2012 facilita o trabalho de criar esses serviços, pois, na primeira implantação que o usuário realizar, todos os serviços que não existirem até o momento são criados e inicializados.

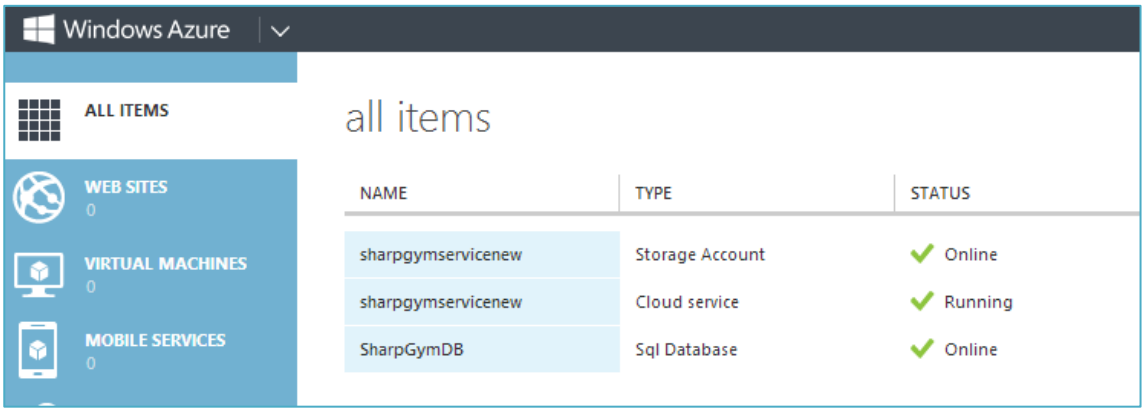

Figura 6.10: Serviços criados para executar o sistema

6.6 Bibliotecas utilizadas

Para a modelagem do banco de dados do servidor, foi utilizada a biblioteca OpenAccess ORM, disponível em http://www.telerik.com/products/orm.aspx. Essa biblioteca permite a construção de diagramas, como no Anexo A, que automaticamente geram código com as propriedades e campos para cada tabela desenhada. Ainda tem a funcionalidade de criar os serviços WCF necessários para a comunicação entre clientes e servidor.

No desenvolvimento da interface web, o Silverlight Toolkit, disponível em http://silverlight.codeplex.com/, foi utilizado para adicionar elementos gráficos à aplicação, como a caixa de diálogo que é mostrada enquanto o sistema sincroniza os dados.

Por fim, para o armazenamento local do dispositivo, foi utilizada a SQLite for Windows Runtime, disponível em http://www.sqlite.org/download.html.

6.7 Testes realizados

Dado o escopo apresentado no início desse capítulo, testes envolvendo inclusão, exclusão e edição de usuários foram realizados, simulando condições de falta de conexão, edição com valores impróprios para os campos e interrupção do serviço caso haja falha da conexão durante a chamada ou o serviço terminar repentinamente no servidor. Todos os testes foram satisfatórios e o sistema foi tolerante a essas falhas, apresentando informações ao usuário sempre que necessário.

7 CONCLUSÃO

Neste capítulo, são apresentadas as conclusões sobre esse trabalho com base nos resultados obtidos. Também são apresentados os aspectos relacionados a ajustes e melhorias para o sistema.

7.1 Análise geral

Os capítulos anteriores trataram de:

- Apresentar as principais motivações que levam as pessoas a procurarem academias e personal trainers;
- Explicar que o trabalho do profissional de treinamento personalizado deve ser ágil e eficaz para atender as necessidades dos seus alunos;
- Os principais conceitos da Computação em Nuvem, da Arquitetura Orientada a Serviços e do Windows 8;
- Descrever o modelo no qual o SharpGym é baseado;
- Criar um protótipo para validar as funcionalidades básicas do sistema.

Dessa forma, a utilidade de uma ferramenta como o SharpGym foi mostrada. As tecnologias utilizadas para a implementação do modelo foram escolhidas levando em consideração as características descritas nos capítulos conceituais desse trabalho.

7.2 Resultados

O sistema não apresentou todas as funcionalidades necessárias para implementar completamente a modelagem do sistema. Da maneira e da quantidade que foram descritas as funcionalidades, se tornou inviável a codificação de todos os casos de uso.

Apesar de o módulo cliente estar incompleto, ou seja, as interfaces não apresentam todas as funcionalidades definidas, o servidor está preparado com o necessário para receber as solicitações dos clientes e responder corretamente a elas. Todos os componentes de infraestrutura do servidor, banco de dados e serviços, foram implantados e estão disponíveis para o acesso dos dispositivos.

Com a prototipação, conclui-se que a aplicação é muito atrativa do ponto de vista do usuário, apresentando um design inovador e bonito. Vale ressaltar que no momento do planejamento desse sistema não havia outros aplicativos que possuíam a mesma funcionalidade para Windows 8, então pode-se considerar que é um sistema inovador dentro dessa tecnologia.

Do ponto de vista de desenvolvimento, houve desafios relacionados com o aprendizado das diretrizes de design exigidas para uma aplicação ser aceita pela Microsoft, pois são muito diferentes do estilo conhecido para desktop tradicional. Um

grande esforço foi feito para escolher a melhor forma de integrar os serviços com o banco de dados em Nuvem, o que só foi obtido graças às ferramentas do Visual Studio 2012.

7.3 Melhorias e novas funcionalidades

No conjunto de melhorias para o sistema estão incluídas todas as funcionalidades omitidas no protótipo apresentado. Para que essas funcionalidades sejam aplicadas, é necessário o estudo de como apresentá-las da melhor forma possível nos dispositivos que executam Windows 8, de forma que a interação com o usuário seja a mais fluída e agradável possível.

Dentre as novas funcionalidades que uma próxima versão do sistema possa ter, duas podem ter destaque na interface:

- Criação de uma versão do aplicativo para os alunos, mostrando apenas as informações relevantes e não permitindo a edição de nenhuma delas;
- Utilizar reconhecimento facial para realizar o controle de entrada e saída dos alunos, evitando a busca pela listagem total, caso seja muito grande.

Essas funcionalidades devem ser estudadas junto ao cliente para verificar se são necessárias para o modelo de trabalho utilizado na academia e se há o interesse do investimento para desenvolver esses módulos.

Também há a possibilidade de remodelar a arquitetura do sistema, substituindo o módulo servidor pelo armazenamento em Nuvem do arquivo de banco de dados do cliente. Esse arquivo contém todas as informações salvas durante as atividades e poderia ser salvo em Nuvem com o auxílio de ferramentas como o SkyDrive, Dropbox ou Google Drive. Assim, a complexidade dos serviços em nuvem seria substituída pela sincronização que essas ferramentas de armazenamento proveem.

REFERÊNCIAS

BROCKSCHMIDT, K. **Windows 8 Apps Programming with HTML, CSS, and JavaScript.** 1st ed.Redmond: Microsoft Press, 2012.

BUSTAMANTE, M. L. **Learning WCF.** 1st ed. Sebastopol: O'Reilly Media, Inc., 2007.

CIBRARO, P. et al. **Professional WCF 4.** 1st ed. Indianapolis: Wiley Publishing, Inc. 2010.

DENSMORE, S. et al. **Building Hybrid Applications in the Cloud on Windows Azure**.1st ed. Redmond: Microsoft, 2012.

DISPLAYSEARCH em 2016 tablets ultrapassarão notebooks em número de vendas. Disponível em <http://macmagazine.com.br/2012/07/03/displaysearch-em-2016-tablets-ultrapassarao-notebooks-em-numeros-de-vendas/>. Acesso em: out. 2012.

FAULKNER, J. A. **Physiology of swimming and diving.** Baltimore: Academic Press, 1968.

FLORIN Toader - Windows Azure Overview. Disponível em: <http://florintoader.net/2012/05/24/windows-azure-overview/>. Acesso em: nov. 2012.

GARFINKEL, S. L. **Architects of the Information Society**. 1st ed. Cambridge: MIT Press, 1999.

GUARDIAN, The - Ballmer's parting message to CES: just Windows, Windows, Windows. Disponível em <http://www.guardian.co.uk/technology/blog/2012/jan/10/microsoft-ballmer-ces-2012-keynote>. Acesso em: set. 2012

GUISELINI, M. **Total Fitness: força, resistência e flexibilidade**. 2 ed. São Paulo: Phorte Editora, 2001.

HAY, C; PRINCE, B. H. **Azure in Action.** 1st ed. Stamford: Manning Publications Co., 2011.

JAMES O'Neill's blog - Windows Phone announcement this afternoon. Disponível em:<http://blogs.technet.com/b/jamesone/archive/2010/02/15/windows-phone-announcement-expected-this-afternoon.aspx>. Acesso em: out. 2012.

KATZ, R. H.;RABKIN, A.; STOICA,A Berkeley View of Cloud Com-UCB/EECS-2009-28. Disponível em: <http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html>.

KRISHNAN, S. **Programming Windows Azure.**1st ed. Sebastopol: O'Reilly Media, Inc., 2010.

LÖWY, J. **Programming WCF Services.**3st ed. Sebastopol: O'Reilly Media, Inc., 2010.

MACKENZIE, N. **Microsoft Windows Azure Development Cookbook.**1st ed. Birmingham: Packt Publishing, 2011.

MÁRCIO Soares - Professor Márcio Soares. Disponível em: <http://www.marciosoares.com.br/profissional.htm>. Acesso em: set. 2012

MATSUDA, Sandra Mahecha; MATSUDA, Victor Keihan Rodrigues. **Efeitos benéficos da atividade física na aptidão física e saúde mental durante o processo de envelhecimento**. Revista Brasileira de atividade física e saúde, São Caetano do sul Vs nº2, 2000.

MELL, P., GRANCE, T. **The NIST definition of cloud computing (draft)**.NIST special publication, 800, 145.(2011)

NEGOCIO & Fitness – Ranking dos 10 países com Maior Número de Academias no Mundo. Disponível em <http://www.negociofitness.com.br/gestao-de-academias/ranking-10-paises-maior-numero-academias>. Acesso em: out. 2012

NOVÁK, I. et al. **Beginning Windows® 8 Application Development.** Indianopolis: John Wiley & Sons, Inc., 2012.

OASIS SOA Reference Model Documentation. Disponível em <http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=soa-rm>. Acesso em: nov. 2012.

PEREIRA, M. M. F. **Academia! Estrutura Técnica e Administrativa**. Rio de Janeiro: Sprint, 1996.

POLLOCK, M.L., WILMORE, J.H. **Exercícios na Saúde e na Doença : Avaliação e Prescrição para Prevenção e Reabilitação**. [S.l.]: MEDSI Editora Médica e Científica Ltda., 1993.

PORTAL da Saúde, Ministério da saúde – Quase metade da população brasileira está acima do peso. Disponível em:<http://portalsaude.saude.gov.br/portalsaude/noticia/4718/162/quase-metade-da-populacao-brasileira-esta-acima-do-peso.html>. Acesso em: out. 2012

SHARP, J. **Windows Communication Foundation 4 Step by Step**. 1st ed. Sebastopol: O'Reilly Media, Inc., 2010.

W3SCHOOLS - OS Platform Statistics. Disponível em: <http://www.w3schools.com/browsers/browsers_os.asp>. Acesso em: dez. 2012.

WINDOWS Azure - Detalhes de preço. Disponível em <http://www.windowsazure.com/pt-br/pricing/details>. Acesso em: nov. 2012.

WINPHONE Brasil - Windows Phone cresce 10% em um ano e já é o terceiro colocado em Marketshare no Brasil. Disponível em <http://winphonebrasil.com.br/14206/windows-phone-cresce-10-em-um-ano-e-ja-e-o-terceiro-colocado-em-marketshare-no-brasil>. Acesso em: out. 2012.

WMPOWERUSER - Windows Phone had 2% market share in Q3 2012, still the fastest growing mobile OS. Disponível em <http://wmpoweruser.com/windows-phone-

had-2-market-share-in-q3-2012-still-the-fastest-growing-mobile-os/>. Acesso em: nov. 2012.

XAML Wonderland - Windows 8 Platform Architecture diagram reimagined. Disponível em <http://blog.wpfwonderland.com/2012/03/19/windows-8-platform-architecture-diagram-reimagined/>. Acesso em: nov. 2012.

ZANETTE, E. T. **Análise do Perfil dos Clientes de Academia de Ginástica: o primeiro passo para o planejamento estratégico**. 2003. Dissertação (Mestrado Profissionalizante em Engenharia) – Escola de Engenharia, UFRGS, Porto Alegre.

ANEXO A ESPECIFICAÇÃO DOS CAMPOS DO BANCO DE DADOS

Tabela	Atributo	Tipo	Tam	Descrição
Aluno	Id(PK)	int		Código de identificação única do Aluno
Aluno	Nome	varchar	255	Nome completo do Aluno
Aluno	Telefone	varchar	255	Número telefônico do Aluno
Aluno	DataNascimento	long		Data de nascimento do Aluno
Aluno	Foto	varchar	Max	Foto do Aluno (base64)
Treino	Id(PK)	int		Código de identificação única do Treino
Treino	AlunoId(FK)	int		Código de Aluno que o Treino está relacionado
Treino	Data	long		Data de realização do Treino
Treino	Observacao	varchar	255	Observações/Anotações feitas durante/após o Treino
AtivAerobica	Id(PK)	Int		Código de identificação única da Atividade Aeróbica
AtivAerobica	Nome	varchar	255	Nome da Atividade Aeróbica
AtivAerobica	FrequenciaCardiaca	double		Frequência Cardíaca verificada durante a Atividade Aeróbica
AtivAerobica	Duracao	long		Tempo de Duração da Atividade Aeróbica
AtivAerobica	Distancia	double		Distância percorrida (quando houver) da

				Atividade Aeróbica
AtivAerobica	TreinoId(FK)	int		Código de Treino que a Atividade Aeróbica está relacionada
AtivMusculacao	Id(PK)	int		Código de identificação única da Atividade de Musculação
AtivMusculacao	Nome	varchar	255	Nome da Atividade de Musculação
AtivMusculacao	Carga	int		Carga/peso utilizado na Atividade de Musculação
AtivMusculacao	Serie	int		Quantidade de Repetições realizadas na Atividade de Musculação
AtivMusculacao	Repeticoes	int		Número de vezes que a Atividade de Musculação é feita por série
AtivMusculacao	TreinoId(FK)	int		Código de Treino que a Atividade de Musculação está relacionada
RegistroDiario	Id(PK)	int		Código de identificação única do Registro Diário
RegistroDiario	FCinicial	double		Frequência Cardíaca verificada ao Aluno iniciar o Registro Diário
RegistroDiario	FCfinal	double		Frequência Cardíaca verificada ao Aluno encerrar o Registro Diário
RegistroDiario	AlunoId(FK)	int		Código do Aluno que o Registro Diário está relacionado
RegistroDiario	Ordem	int		Número de sequência do Registro Diário
RegistroDiario	HorarioEntrada	long		Horário de abertura do Registro Diário
RegistroDiario	HorarioSaida	long		Horário de encerramento do Registro Diário

PressaoArterial	Id(PK)	int		Código de identificação única da Pressão Arterial
PressaoArterial	Diastolica	double		Pressão Arterial máxima do ciclo cardíaco
PressaoArterial	Sistolica	double		Pressão Arterial mínima do ciclo cardíaco
PressaoArterial	RegistroDiarioId(FK)	int		Código do Registro Diário que a Pressão Arterial está relacionada
PressaoArterial	Horario	long		Horário da aferição da Pressão Arterial
AvaliacaoFisica	Id(PK)	int		Código de identificação única da Avaliação Física
AvaliacaoFisica	AlunoId(FK)	int		Código do Aluno que a Avaliação Física está relacionada
AvaliacaoFisica	Numero	int		Número de sequência da Avaliação Física
AvaliacaoFisica	Peso	double		Peso medido do Aluno durante a Avaliação Física
AvaliacaoFisica	Estatura	double		Estatura (altura) medida do Aluno durante a Avaliação Física
AvaliacaoFisica	TricepsDC	double		Medição da Dobra Cutânea do Tríceps durante a Avaliação Física
AvaliacaoFisica	EscapularDC	double		Medição da Dobra Cutânea Escapular durante a Avaliação Física
AvaliacaoFisica	PeitoralDC	double		Medição da Dobra Cutânea Peitoral durante a Avaliação Física
AvaliacaoFisica	ToraxicaDC	double		Medição da Dobra Cutânea Torácica durante a Avaliação Física
AvaliacaoFisica	SupraIliacaDC	double		Medição da Dobra Cutânea Supra Ilíaca durante a Avaliação

				Física
AvaliacaoFisica	AbdominalDC	double		Medição da Dobra Cutânea Abdominal durante a Avaliação Física
AvaliacaoFisica	QuadricepsDC	double		Medição da Dobra Cutânea do Quadríceps durante a Avaliação Física
AvaliacaoFisica	PanturrilhaDC	double		Medição da Dobra Cutânea da Panturrilha durante a Avaliação Física
AvaliacaoFisica	BicepsDC	double		Medição da Dobra Cutânea do Bíceps durante a Avaliação Física
AvaliacaoFisica	PernaP	double		Perímetro medido da Perna durante a Avaliação Física
AvaliacaoFisica	CoxaP	double		Perímetro medido da Coxa durante a Avaliação Física
AvaliacaoFisica	PelveP	double		Perímetro medido da Pelve durante a Avaliação Física
AvaliacaoFisica	AbdominalP	double		Perímetro Abdominal medido durante a Avaliação Física
AvaliacaoFisica	AnteBracoP	double		Perímetro medido do Ante Braço durante a Avaliação Física
AvaliacaoFisica	BracoP	double		Perímetro medido do Braço durante a Avaliação Física
AvaliacaoFisica	BracoContraidoP	double		Perímetro medido do Braço Contraído durante a Avaliação Física
AvaliacaoFisica	PunhoL	double		Largura do Punho medida durante a Avaliação Física
AvaliacaoFisica	UmeroL	double		Largura do Úmero medida durante a

				Avaliação Física
AvaliacaoFisica	DataAval	long		Data da Avaliação Física
AvaliacaoFisica	Observacao	string	255	Observações/Anotações feitas durante/após a Avaliação Física