**CENTRO UNIVERSITÁRIO CARIOCA – UNICARIOCA**

**JOÃO CARLOS MARINS JUNIOR**

**JULIO CESAR DIAS CASIMIRO**

**THIAGO DA COSTA GIECHT**

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA CONSUMO DE DADOS ARMAZENADOS NA NUVEM.**

**RIO DE JANEIRO**

**2017**

**JOÃO CARLOS MARINS JUNIOR**

**JULIO CESAR DIAS CASIMIRO**

**THIAGO DA COSTA GIECHT**

**DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA CONSUMO DE DADOS ARMAZENADOS NA NUVEM.**

Trabalho de Conclusão de Curso Apresentado ao Curso de Ciência da Computação do Centro Universitário Carioca, como requisito à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Alberto Tavares da Silva

**RIO DE JANEIRO**

**2017**

**Resumo**

Este trabalho de Conclusão de Curso visa o desenvolvimento de uma aplicação C# através da interface de desenvolvimento *Visual Studio*, focada em nuvem *Microsoft Azure,* com interface *Android* e acesso a dados através do ambiente *Microsoft SQL,* para armazenamento e busca de prestadores de serviços por pessoas interessadas.

**Palavras chave:**

*Microsoft Azure, Xamarin, C#. Coss-Platform, SQL Server, Android, Visual Studio*.

***ABSTRACT***

*This course completion work aims to development of a C # application through the Visual Studio development interface, focused on Microsoft Azure cloud, with Android interface and data access through the Microsoft SQL environment, for storage and search of service providers by interested people.*

**Key words:**

*Microsoft Azure, Xamarin, C#. Coss-Platform, SQL Server, Android, Visual Studio*.

**Sumário**

[1 INTRODUÇÃO 5](#_Toc498979450)

[1.1 Contexto 5](#_Toc498979451)

[1.2 Objetivo 5](#_Toc498979452)

[1.3 Justificativa 6](#_Toc498979453)

[1.4 Estrutura do trabalho 6](#_Toc498979454)

[2 COMPUTAÇÃO EM NUVEM 7](#_Toc498979455)

[2.1 Modelos de implantação 8](#_Toc498979456)

[2.1.1 Nuvem Pública 8](#_Toc498979457)

[2.1.2 Nuvem Privada 8](#_Toc498979458)

[2.1.3 Nuvem Comunitária 8](#_Toc498979459)

[2.1.4 Nuvem Híbrida 8](#_Toc498979460)

[2.2 Características de computação em nuvem 9](#_Toc498979461)

[2.2.1 Auto-Serviço Sob Demanda 9](#_Toc498979462)

[2.2.2 Elasticidade Rápida 10](#_Toc498979463)

[2.2.3 *Pool* de Recursos 10](#_Toc498979464)

[2.2.4 Mobilidade 10](#_Toc498979465)

[2.2.5 Monitoramento 10](#_Toc498979466)

[2.3 Modelos de serviços 11](#_Toc498979467)

[*2.3.1* *Software as a Service (SaaS)* 11](#_Toc498979468)

[*2.3.2* *Platform as a Service (PaaS)* 11](#_Toc498979469)

[2.3.3 *Infrastructure as a Service (IaaS)* 12](#_Toc498979470)

[3 MICROSOFT AZURE 13](#_Toc498979471)

[3.1 Uso comercial 13](#_Toc498979472)

[3.2 Serviços 14](#_Toc498979473)

[4 TECNOLOGIAS MÓVEIS 15](#_Toc498979474)

[4.1 Sistemas Operacionais Móveis 15](#_Toc498979475)

[*4.1.1* *IOS* 16](#_Toc498979476)

[*4.1.2* *WINDOWS PHONE* 16](#_Toc498979477)

[*4.1.3* *ANDROID* 17](#_Toc498979478)

[*4.2* *User Interface e User Experience* 17](#_Toc498979479)

[*4.2.1* *User Interface* 17](#_Toc498979480)

[*4.2.2* *User Experience* 18](#_Toc498979481)

[*4.3* Desenvolvimento *Móvel e Cross-plataform* 19](#_Toc498979482)

[4.4 Xamarin 19](#_Toc498979483)

[4.5 Model-View-ViewModel (MVVM) 22](#_Toc498979484)

[*4.5.1* *Model* 23](#_Toc498979485)

[*4.5.2* *View* 23](#_Toc498979486)

[*4.5.3* *ViewModel* 24](#_Toc498979487)

[5 SONAR DE SERVIÇOS 25](#_Toc498979488)

[5.1 Mini mundo do projeto 25](#_Toc498979489)

[5.2 Levantamento de requisitos 27](#_Toc498979490)

[5.2.1 Requisitos Funcionais 27](#_Toc498979491)

[5.2.2 Requisitos não Funcionais 28](#_Toc498979492)

[5.2.3 Regras de Negócio 28](#_Toc498979493)

[5.3 Modelo de Casos de Uso 29](#_Toc498979494)

[5.4 Modelo de Classes de domínio 30](#_Toc498979495)

[5.5 Modelo Conceitual do Banco de Dados 31](#_Toc498979496)

[5.6 Modelo Lógico de Banco de Dados 32](#_Toc498979497)

[5.7 Modelo Físico do Banco de Dados 33](#_Toc498979498)

[5.7.1 Tabela USUARIO 33](#_Toc498979499)

[5.7.2 Tabela CLIENTE 34](#_Toc498979500)

[5.7.3 Tabela SERVICO 34](#_Toc498979501)

[5.7.4 Tabela PRESTADOR\_SERVICO 35](#_Toc498979502)

[5.7.5 Tabela INTERESSE 36](#_Toc498979503)

[5.7.6 Criação de Índices 37](#_Toc498979504)

[5.8 Interface do Sistema 38](#_Toc498979505)

[5.8.1 Tela de Abertura 38](#_Toc498979506)

[5.8.2 Tela Inicial 39](#_Toc498979507)

[5.8.3 Cadastro de Usuários 40](#_Toc498979508)

[5.8.4 Tela inicial para usuários autenticados 41](#_Toc498979509)

[5.8.5 Telas de Atividades como Cliente 42](#_Toc498979510)

[5.8.6 Telas de Registro como Prestador de Serviço 43](#_Toc498979511)

[5.8.7 Tela de Atividades como Prestador de Serviços 44](#_Toc498979512)

[5.8.8 Exibindo resultados de uma busca 45](#_Toc498979513)

[6 CONCLUSÃO 47](#_Toc498979514)

[7 Bibliografia 49](#_Toc498979515)

# INTRODUÇÃO

## Contexto

Pensar em um mundo sem internet hoje seria um exercício mental quase impossível, pois a facilidade que esta tecnologia nos proporciona foi tomando cada vez mais espaço no cenário da sociedade atual, passando de um bem de consumo para uma necessidade diária. Planejar suas viagens, realizar compras, evitar filas bancárias, entre muitas coisas são facilitadas graças aos serviços oferecidos através da rede mundial de computadores. Hoje com o advento da computação em nuvem, ficou cada vez mais fácil para empresas e pessoas desenvolverem suas soluções sem se preocupar com infraestrutura e disponibilidade do serviço, delegando esta responsabilidade a empresas que se prontificam a realizar este trabalho. A proposta deste trabalho é conhecer um pouco mais da computação em nuvem, de seus aspectos e desenvolver uma aplicação a partir dela.

## Objetivo

O objetivo deste trabalho é o desenvolvimento de uma aplicação para cadastro de clientes e prestadores de serviços de mecânica e borracharia, focada em computação em nuvem *Microsoft® Azure*, no qual o cliente efetua uma busca baseada em sua localização e a aplicação exibe os prestadores dos serviços cobertos em um raio inicial de cinco quilômetros, permitindo aumento pelo usuário.

A aplicação utilizará cadastro de interesses para manter controle de acessos por local, além de contar com uma modalidade de avaliação para que clientes possam avaliar prestadores de serviços e os prestadores de serviços replicarem a avaliação do cliente.

## Justificativa

A idéia surgiu da observação de que, para alguns tipos de trabalhos, ainda falta um ambiente de concentração de seus profissionais. Já existem aplicações que fazem o mesmo com transporte e alimentação, porém basta furar um pneu no meio da estrada, que descobrimos que alguns serviços não carecem desta facilidade.

## Estrutura do trabalho

O capítulo 1 é a introdução do trabalho, onde são descritos contexto, os objetivos, as justificativas e a estrutura do trabalho que está sendo apresentado.

O capítulo 2 trata do conceito de computação em nuvem, seu modelo de implantação, suas características e seu modelo de serviços.

O capítulo 3 aborda a ferramenta *Microsoft Azure,* bem como seu uso em ambiente comercial e seus serviços.

O capítulo 4 trata de tecnologias móveis, discorrendo sobre sistemas operacionais, modelos de interface, desenvolvimento *cross-platform,* além de introduzir-nos a ferramentas para desenvolvimento *cross-platform*, o *Xamarin,* e seu modelo estrutural, realizando a divisão de um projeto em modelo, visão e modelo da visão, o padrão *Model-View-ModelView (MVVM)*.

O capítulo 5 apresenta o aplicativo Sonar de Serviços, que é o objetivo deste trabalho. O processo de desenvolvimento da ferramenta inclui, o levantamento de requisitos funcionais, requisitos não funcionais e regras de negócio; diagramação de casos de uso; diagramação de classes de domínio; modelagem lógica do banco de dados da aplicação, além de um levantamento de ideias a serem usadas em novas versões da aplicação.

O capítulo 6 conclui o trabalho, aborda conceitos adquiridos pelo grupo a partir do desenvolvimento do documento e da aplicação.

# COMPUTAÇÃO EM NUVEM

Computação em nuvem, do inglês *cloud computing* é o termo usado para definir um modelo de compartilhamento de recursos acessíveis de qualquer lugar, que possam ser ajustados conforme a demanda, com mínimo esforço de manutenção por parte do provedor [NIST, 2011]. Plataformas de desenvolvimento de *software* e banco de dados, armazenamento de arquivos, infraestruturas de rede, servidores de impressão, máquinas virtuais, entre outros, são oferecidos ao usuário como serviços, como mostra a figura 1, bastando apenas o usuário acessá-los e operá-los de qualquer computador com um navegador e acesso à internet, sem necessidade de programas ou recursos adicionais.

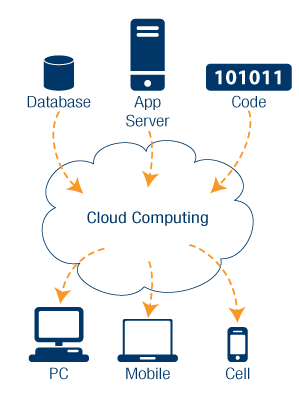


Figura 1: Modelo de computação em nuvem

Fonte: www.infoescola.com

## Modelos de implantação

A computação em nuvem possui modelos de implantação, que predizem a localização dos recursos e quem terá controle e acesso sobre eles, como é mostrado na figura 2. São eles, a nuvem pública, nuvem privada, nuvem híbrida e nuvem comunitária, e serão abordados a seguir.

### Nuvem Pública

Modelo onde os recursos são compartilhados e geralmente são oferecidos por provedores que fornecem soluções para empresas e pessoas, com restrição a recursos de acordo com privilégios de usuário, por isso, geralmente a nuvem pública está em conjunto com um sistema de autorização. A nuvem pública tem como vantagem os recursos compartilhados, o que pode proporcionar um custo menor a empresas com necessidades baixas de recursos.

### Nuvem Privada

Ao contrário da nuvem pública, o modelo de nuvem privada não é compartilhado, ou seja, é constituído pelo uso de servidores dedicados a empresas ou pessoas físicas, situando-se em instalação própria ou provedor de terceiros. A vantagem do modelo de nuvem privada é que o controle do crescimento é do próprio usuário, porém pode significar existência de recursos em demasia, dependendo da demanda da empresa ou pessoa.

### Nuvem Comunitária

No modelo de nuvem comunitária, organizações com interesses em comum se unem para compartilhar a infraestrutura. Sua administração ocorre por meio da própria comunidade ou terceiros existentes, sejam dentro ou fora da comunidade.

### Nuvem Híbrida

No modelo de nuvem híbrida existe a junção de dois ou mais modelos de implantação de nuvem, não sendo somente pública ou privada ou comunitária, mas mesclando características desses modelos. Nuvens híbridas são agrupadas por tecnologia padrão ou proprietária, permitindo portabilidade de aplicativos e dados, porém ainda são vistas como entidades únicas.

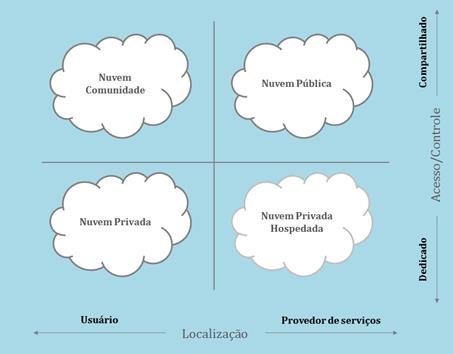


Figura 2: Modelos de implantação de nuvem

Fonte: teleco.com.br

## Características de computação em nuvem

Dentre as características da computação em nuvem, encontram-se o auto-serviço sob demanda, a elasticidade rápida, o *pool* de recursos, a mobilidade, o amplo acesso e o monitoramento de serviços.

### Auto-Serviço Sob Demanda

O auto-serviço sob demanda, do inglês *On-Demand Self-Service* é a capacidade dos recursos serem adicionados ou removidos conforme necessidade do cliente, sem que para isso ocorra a interação humana com o provedor.

### Elasticidade Rápida

A elasticidade rápida, do inglês *Rapid Elasticity,* é característica de alocar mais ou menos recursos instantaneamente conforme seja necessário, dando a impressão ao cliente de que a nuvem é um espaço infinito de processamento e armazenamento.

### *Pool* de Recursos

O *Pool* de recursos, do inglês *Resource Pooling,* é a característica de divisão de recursos computacionais do provedor para múltiplos consumidores, usando um modelo *multi-tenant*, descrito no item 2.3, habilitando e reabilitando de acordo com a demanda do cliente. O cliente não possui controle de onde exatamente estão alocados seus recursos, porém pode especificar o país, estado ou centro de dados em alto nível de abstração.

### Mobilidade

A mobilidade é a capacidade dos serviços da nuvem serem acessados de qualquer lugar, bastando somente uma conexão de banda larga e um navegador de internet. Por não ser necessário um sistema ou aplicação própria, o cliente pode acessar os serviços através de *smartphones*, *tablets, lan houses,* televisores *smart,* entre outros dispositivos que possuam acesso à internet.

### Monitoramento

O Monitoramento de serviços é a capacidade de controlar a utilização dos recursos da nuvem, medindo a utilização de forma automática e controlando-os de forma transparente, ou seja, o fornecedor e o cliente possuem total conhecimento da utilização dos recursos.

## Modelos de serviços

Os modelos de serviços definem a divisão de responsabilidades entre cliente e provedor, delimitando onde iniciam e terminam as responsabilidades de cada um dos envolvidos. Conforme apresentado na figura 3, cada modelo atende a um tipo diferente de propósito, sendo três os modelos de serviços definidos pela *National Institude of Standarts and Technology - NIST* (2011), o PaaS, o Saas e o IaaS, que serão descritos a seguir.

### *Software as a Service (SaaS)*

Software como um serviço é um modelo de disponibilização de sistemas de propósitos específicos armazenados e executados sobre a infraestrutura do provedor, sendo acessíveis através de um navegador *WEB* ou por uma interface de programação da aplicação (API – *Application Program Interface*). No SaaS, o usuário não possui controle sobre a infraestrutura, exceto configurações específicas de domínio do usuário. Sistemas operacionais, armazenamento, rede, entre outros, são de controle do provedor do serviço. Exemplos de serviços neste modelo são o *Google Docs* e o *Office Web Apps.*

### *Platform as a Service (PaaS)*

Plataforma como um serviço é um modelo de disponibilização de ambientes de programação, testes e hospedagem de aplicações desenvolvidas pelo usuário na infraestrutura do provedor, onde o usuário possui um ambiente propício para desenvolver suas aplicações, sem a preocupação com infraestrutura e *softwares de desenvolvimento e gestão de banco de dados. A* plataforma *Microsoft Azure* é um exemplo deste modelo (*Microsoft*, 2012).

### *Infrastructure as a Service (IaaS)*

Infraestrutura como um serviço é um modelo de disponibilização de recursos de infraestrutura fundamentais, tais como servidores, rede, entre outros, onde o usuário pode instalar sistemas operacionais e aplicativos de forma arbitrária. O IaaS possui uma interface única para administração da infraestrutura, API para interação com *hosts, routers e switchs,* adição de novos equipamentos de forma simples. O controle da infraestrutura no modelo IaaS não fica nas mãos do usuário, porém o mesmo possui controle sobre os sistemas operacionais, aplicativos e armazenamento implantados, também a seleção eventual de componentes de rede, porém o usuário é responsável pela escalabilidade no modelo, ao invés do provedor.

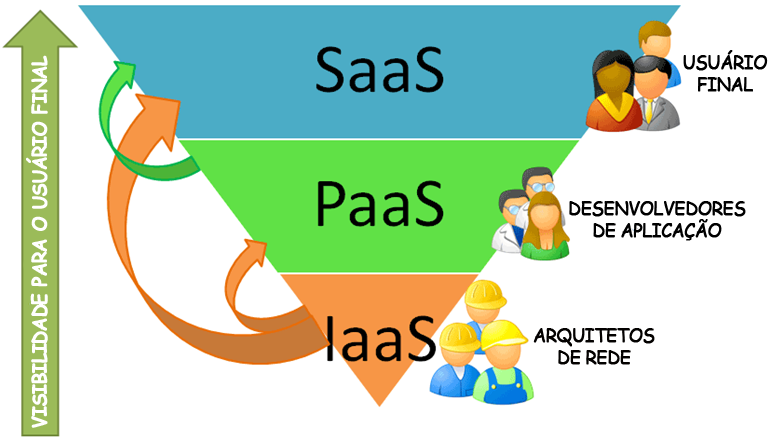


Figura 3: Serviços da nuvem

Fonte: vitormeriat.com.br

# MICROSOFT AZURE

A *Microsoft Azure, cuja logo é exibida na figura 4,* é uma plataforma destinada à execução de aplicativos e serviços, sendo baseada nos conceitos da computação em nuvem. Trabalha com o *Windows Azure* que é uma espécie de sistema operacional que gerencia todo e qualquer recurso empregado juntamente com seu banco SQL *Azure* e seu *middler* de integração *AppFabric*.



Figura 4: Microsoft Azure

Fonte: <https://www.persistent.com>

## Uso comercial

Hoje em dia, a *Microsoft Azure* é utilizada para fins comerciais por empresas que buscam serviços em nuvem tanto para demandas internas quanto para demandas externas, pelo fato de possuir grande abrangência de sistemas que podem ser desenvolvidos, assim como linguagens que a plataforma aceita em seus servidores. O sistema conta com um gerenciador, que é o próprio *Microsoft Azure*, conforme supracitado, funcionando com máquinas virtuais, onde pode aplicar qualquer sistema operacional. Desta forma, permite que o usuário migre para a plataforma Azure utilizando os meios que ele acha mais seguro para manter seu sistema.

## Serviços

A *Microsoft Azure* trabalha hoje com serviços de *Cloud Computing* e *Data Centers* que conseguem atender os clientes, com suporte aos principais *frameworks* de desenvolvimento, assim como os principais bancos de dados para hospedagem em seus servidores. São exemplos de linguagens suportadas pela *Microsoft Azure*: C#, *Visual Basic* .*NET*, *ASP .NET,* *Java*, *Phyton*, PHP e *Node*.*js*. O sistema conta também com máquinas virtuais em *Windows* e *Linux*, conforme supracitado para atender as plataformas que forem solicitadas ao seu serviço. Possui escalabilidade da demanda para facilitar a compra dos serviços de máquinas em *cloud*, por exemplo: Se você possui um servidor com certo limite de memória e precisar aumentar sua capacidade, você poderá ficar despreocupado que o sistema *Microsoft Azure* faz o ajuste em cima do gasto de processamento que você poderá vir a ter.

A *Microsoft Azure*, hoje em dia, oferece soluções de hospedagem e gerenciamento de software para pequenos e grandes aplicativos, em outras palavras, a *Microsoft Azure* consegue lidar com grande parte dos aplicativos que possam ser hospedados em seu serviço. A *Azure* também possui recursos para serviços com distribuição global, como é o caso do seu BD *Cosmos*. O Serviço também conta com uma função *Backup* que oferece hospedagem dos *backups* em servidores online e também, *backups* em fitas conforme necessidade de algumas empresas de guardar dados durante anos. A figura 5 mostra os serviços disponibilizados através da plataforma *Azure.*

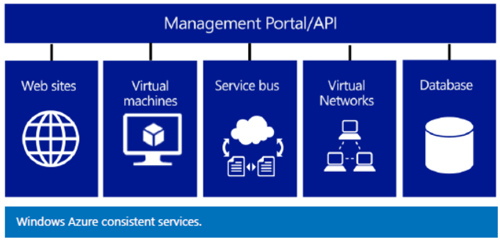


Figura 5: Serviços Azure.

Fonte: i-msdn.sec.s-msft.com

# TECNOLOGIAS MÓVEIS

Tecnologias móveis são aquelas que permitem a utilização durante a movimentação de seu usuário. O conceito foi apresentado na década de 40, mas devido às limitações da época não teve nenhum avanço prático, limitando-se à criação de teorias. Em 1973, por intermédio das primeiras ligações efetuadas por um dispositivo móvel e o surgimento dos primeiros conceitos sobre o celular, as possibilidades começaram a ser discutidas novamente.

Os primeiros aparelhos de uso pessoal eram limitados, possibilitando somente ligações. Foram as gerações seguintes que começaram a apresentar uma gama de serviços e novas possibilidades, tais como, envio de mensagens de texto (SMS), telas coloridas e a agregação de serviços multimídia.

O próprio conceito de "celular" foi abandonado e por sua vez substituído pela alcunha de *smartphones*. Os aparelhos antes de uso limitado se destacam agora pela robustez de hardware integrado e largo uso de software, vindo a possuir até variações de sistemas operacionais, agregando muito mais possibilidades para os usuários e desenvolvedores de conteúdo.

As tecnologias móveis são abrangentes. Muitos setores já investem no mercado móvel, não só na parte de telefonia, mas também em eletrodomésticos e em carros.

## Sistemas Operacionais Móveis

O sistema operacional é o mais importante componente de dispositivos móveis. Sendo o responsável pela integração entre as vontades do usuário e os recursos de hardware, para que um determinado aplicativo realize uma tarefa, ou que múltiplos serviços estejam em uso simultâneo. Suas diferenças começam já nos layouts iniciais, como mostra a figura 6. Cada sistema possui seus pontos fortes e fracos, pode ser projetado para uso em hardwares específicos, possuir uma gama maior e mais versátil de aplicativos ou focar na simplicidade e uso mais ágil. Serão apresentados agora, alguns dos principais sistemas utilizados em *smartphones*.



Figura 6: Layout Android, IOS e Windows Phone

Fonte: firstpost.com

### *IOS*

Lançado em 2007 pela Apple, sendo criado para os aparelhos da linha *iPhone*, integrando os *tablets* *iPad* posteriormente. Possui muita fluidez, fruto do aproveitamento de *hardware* dedicado. Seus aplicativos nativos e de terceiros são baixados e instalados somente pela loja oficial, a *Apple Store*, conferindo certa segurança aos mesmos. O visual é semelhante às versões *desktop* do sistema operacional. Atualmente está se encaminhando para a versão 11.

### *WINDOWS PHONE*

Sistema idealizado pela Microsoft e baseado no *Kernel* do *Windows* CE6. Foi lançado em 2010, sucedendo o *Windows Mobile*. Possui alta compatibilidade com os computadores *Windows,* sendo empregado em sua maioria em aparelhos da própria Microsoft. Também possui loja oficial para aplicativos, atualmente na versão 10.

### *ANDROID*

Criado pela *Android Inc* e baseado no *Kernel Linux*, com diversos componentes e bibliotecas, sendo usado e posteriormente adquirido pela Google. É disponibilizado sobre licença de código aberto, mas devido à facilidade de personalização é geralmente combinado com *software* privado, possibilitando interfaces diferentes, algumas vistas na figura 7. Possui a maior gama de aplicativos em sua loja e possibilita também a instalação de programas de terceiros. Atualmente é o mais difundido entre usuários e desenvolvedores, chegando à versão 8.



Figura 7 - Diferentes personalizações de tela do Android.

Fonte: techtudo.com.br

## *User Interface e User Experience*

### *User Interface*

A interface com o usuário basicamente inclui a aparência do sistema, seus elementos visuais, botões e ícones, ou seja, tudo o que é utilizado para interagir com um dispositivo. A figura 8 mostra as diferentes interfaces de acesso inicial dos três principais sistemas.



Figura 8: Interfaces de acesso IOS, Android e Windows Phone.

Fonte: firstpost.com

### *User Experience*

A experiência do usuário trata dos aspectos da interação a um nível de satisfação com os serviços oferecidos. Foca o processo de resolução de um problema ou solicitação. A figura 9 da uma idéia inicial dos recursos de interação com as opções de configuração iniciais dos sistemas móveis, fator primordial para melhor aproveitamento no uso.

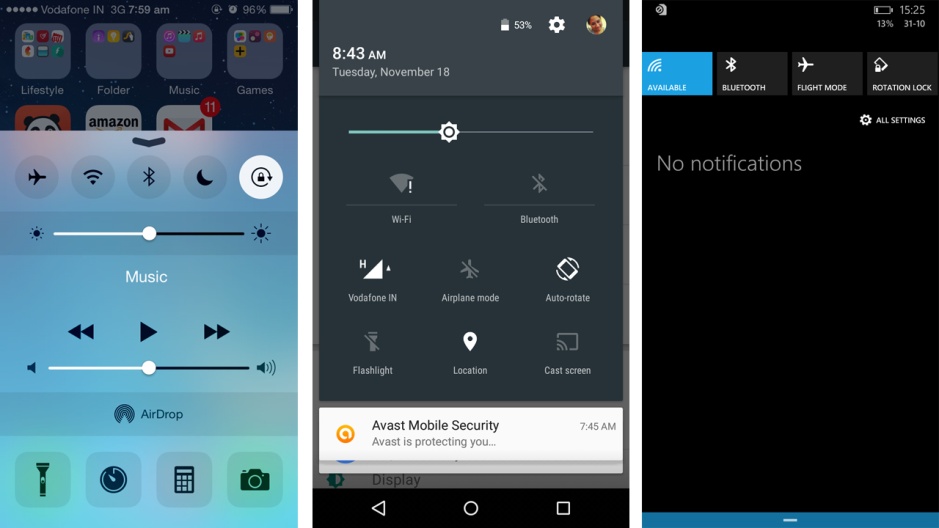


Figura 9: Configurações para IOS, Android e Windows Phone.

Fonte: firstpost.com

## Desenvolvimento *Móvel e Cross-plataform*

Dadas as estruturas de concepção, a construção e arquitetura de um sistema operacional móvel possuem diferentes recursos para o desenvolvimento de aplicações. Com tantas particularidades o ambiente móvel se mostrou promissor para a integração de *cross-plataform*. A compatibilidade gerada através dessas ferramentas permite a flexibilidade no desenvolvimento de aplicativos para várias plataformas ao mesmo tempo. Uma aplicação pode ter versões para várias plataformas, como mostra a figura 10 e o desenvolvimento pode ocorrer ao mesmo tempo.

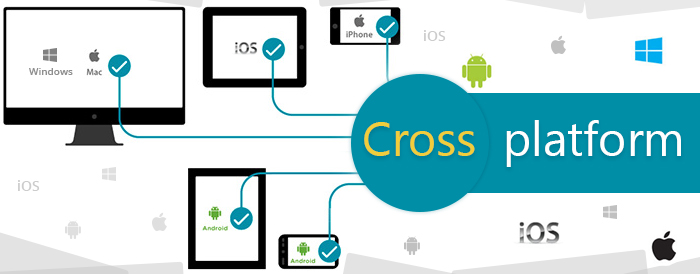


Figura 10: Ilustração do conceito de Cross Plataform.

Fonte: purelogics.net

## Xamarin

É uma suíte de produtos voltada para o desenvolvimento de aplicativos móveis. Com a plataforma *Xamarin* é possível criar aplicações para *IOS*, *Android* e para a plataforma universal do *Windows*, o UWP, a partir das linguagens C# e F#.

É importante destacar que uma aplicação criada a partir do *Xamarin* é 100% nativa. Através do C#, a plataforma consegue ter acesso total as interfaces de programação dos sistemas operacionais móveis, retirando do programador a tarefa de saber as linguagens de programação para cada plataforma móvel a qual irá desenvolver sua aplicação, necessitando apenas conhecer a linguagem trabalhada no *Xamarin*, seja C# ou F#, como mostra a figura 11.

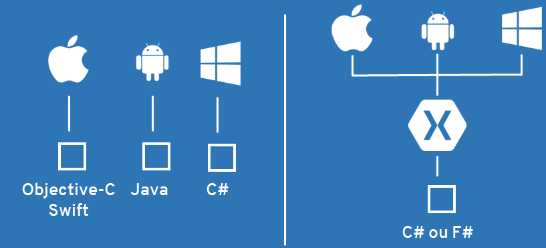


Figura 11: centralização da tarefa de programação *Xamarin*

Fonte: www.mundotibrasil.com.br

Através das *portable class libraries,* o *Xamarin* aproveita uma grande vantagem do desenvolvimento *cross-platform*, que é o compartilhamento de código. Ao desenvolver para o *Xamarin* estamos desenvolvendo um código compartilhado, que será convertido em código nativo das plataformas *Android, IOS e UWP* entre 75% e 100% do todo, mas também é possível desenvolver funções específicas para uma das plataformas, pois um projeto *cross-platfomr* no *Xamarin* é dividido em um de código compartilhado e um projeto para cada uma das plataformas suportadas, como mostra a figura 12.

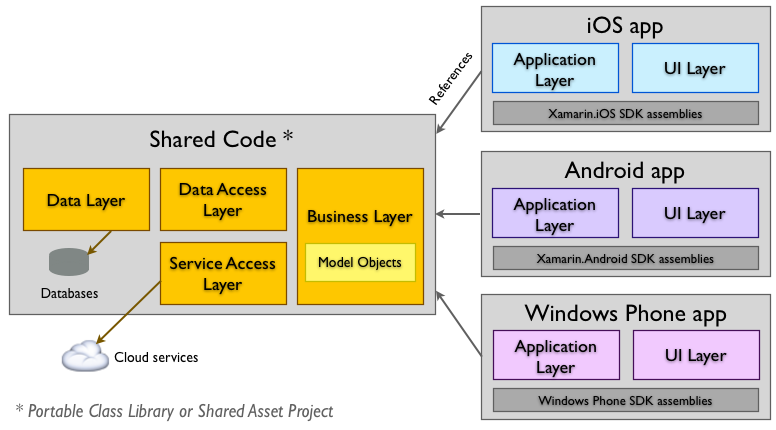


Figura 12 Arquitetura de um projeto *cross-platform em Xamarin*

Fonte www.lambda3.com.br

O *Xamarin* possui uma interface de desenvolvimento própria chamada de *Xamarin Studio,* mas também pode estar integrada ao Microsoft Visual Studio, tendo acesso total às funcionalidades dos kits de desenvolvimento de softwares oficial de cada dispositivo, através de chamadas C#.

Para desenvolvimento de interfaces o *Xamarin.Forms* permite a utilização de código C# ou da *Extensible Application Markup Language* (XAML), que permite desenvolver utilizando mapeamento de itens comuns presentes nas plataformas envolvidas. Recursos específicos para cada plataforma devem ser desenvolvidos nos projetos específicos. A figura 13 mostra uma representação do desenvolvimento *Xamarin.Forms* e como sua implementação torna o código desenvolvido em *Xamarin* cada vez mais compartilhado entre plataformas diferentes.

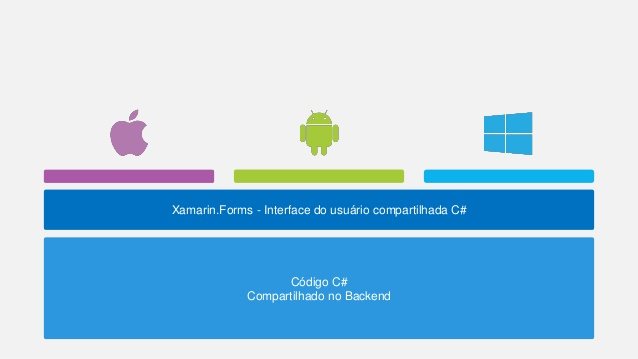


Figura 13 Representação de compatibilidade do código C#

Fonte: williamsrz.azurewebsites.net

## Model-View-ViewModel (MVVM)

É um padrão de projetos criado em 2005, por John Gossman, um arquiteto de projetos da *Microsoft.*

O modelo MVVM procura separar as responsabilidades de uma aplicação em diferentes camadas, desacoplando interface, lógica de apresentação e dados. As classes de negócio e serviços externos, como acesso a banco de dados, são separados dos componentes de interface com o usuário. Não existe conhecimento entre as camadas que usam o mecanismo de vinculação de dados, ou *data bindings* para sua comunicação.

O *data binding* realiza a interação entre os componentes da *View* com a *ViewModel* notificando eventos e também disparando comandos específicos. Por sua vez a *ViewModel* é capaz de responder à essas notificações realizando algum procedimento no modelo da aplicação. A mecânica dessa interação é mostrada na figura 14.

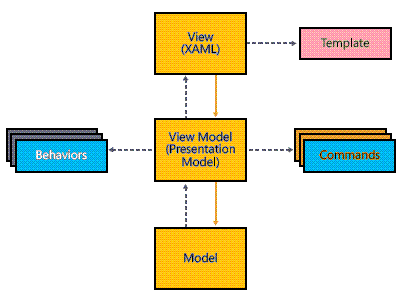


Figura 14 Estrutura básica de aplicação MVVM

Fonte: www.macoratti.net

### *Model*

É onde toda a lógica da aplicação e os dados estão encapsulados. O modelo não tem como referenciar diretamente a *View* ou *ViewModel*, mas pode interagir e providenciar mudanças de estado através de interfaces. É o responsável pela validação de dados e conexão com serviços externos.

### *View*

Responsável por definir interface com o usuário. A conexão da *View*, com sua *ViewModel* específica, se dá através da propriedade *DataContext*. Deve possuir somente código que manipule elementos visuais ou chamados a métodos de inicialização de componentes dentro de um construtor. Estes métodos irão disponibilizar os dados adequadamente para as outras camadas.

### *ViewModel*

É a camada que coordena a interação entra as camadas *View* e *Model*, sendo ela responsável por notificar outras camadas sobre alterações de dados ocorridas. Essas classes podem fazer uso de comandos que possibilitam a passagem de dados para preenchimento de controles de exibição ou recebendo dados definidos em uma *View ou para* operações da *Model*, como inserção ou retorno de dados. A figura 15 apresenta um exemplo de interação *ViewModel* com as outras camadas do padrão MVVM.



Figura 15 Interação entre camadas MVVM

Fonte: https://msdn.microsoft.com

.

# SONAR DE SERVIÇOS

Sonar de serviços será uma aplicação ambientada em nuvem, que realizará o cadastro de prestadores de serviços, que poderão ser buscados por clientes cadastrados no sistema ou não.

## Mini mundo do projeto

A proposta é a criação de um aplicativo de execução em ambiente de nuvem que permita que prestadores de serviços de mecânica e borracharia se cadastrem como pessoa física ou jurídica, permitindo que clientes que busquem pelos serviços oferecidos nas proximidades os encontrem e os contatem.

O sistema deve contar com um formulário para cadastro de prestadores de serviços, que colete informações como Nome ou Razão Social, CPF ou CNPJ, endereço residencial / comercial, telefones, e-mail, valor do orçamento para clientes nas proximidades, serviços prestados, informações complementares, além de *login* e senha do sistema.

O sistema também deve contar com um formulário para cadastro de clientes, que colete informações como Nome ou razão social, CPF / CNPJ, endereço, telefones, e-mail, nome de usuário do sistema e senha de acesso.

Deverão ser aceitos clientes sem cadastro, porém um usuário não cadastrado só poderá fazer pesquisas em um raio de 5 (cinco) quilômetros, sendo necessário um registro para alterar o raio de busca.

O cliente poderá realizar uma busca digitando um endereço ou utilizando coordenadas de GPS do *smartphone*.

Deverá haver uma funcionalidade que permita que clientes avaliem serviços feitos por prestadores, além de réplica opcional pelo prestador de serviços. O sistema de avaliação contará com uma nota de 1 (hum) até 5 (cinco), além de uma descrição da avaliação, opcional pelo cliente.

O telefone e endereço do prestador de serviço buscado deverão ficar ocultos, até que o cliente clique em um botão que revele os dados, que crie um registro de interesse do cliente ao prestador, para estatísticas do aplicativo, que poderá ser desenvolvido em versões futuras.

O sistema deve permitir ao cliente visualizar prestadores de serviço por pontos no mapa ou por uma lista, em ordem de proximidade.

## Levantamento de requisitos

O levantamento de requisitos do sistema contará com requisitos funcionais, requisitos não funcionais e regras de negócios, demonstrados a seguir.

### Requisitos Funcionais

|  |  |
| --- | --- |
| RF01 | O cliente pode realizar o cadastro no sistema |
| RF02 | RF02 – O cliente pode efetuar o *login* no sistema. |
| RF03 | O cliente poderá avaliar o serviço executado por um prestador que demonstrou interesse. |
| RF04 | O cliente que possui cadastro como prestador de serviço pode alterar entre modos sem a necessidade de sair do sistema. |
| RF05 | O Usuário deve poder ver os prestadores como uma lista. |
| RF06 | O Usuário deve poder visualizar prestadores de serviço como pontos no mapa. |
| RF07 | O prestador de serviço poderá responder a uma avaliação feita por um cliente. |
| RF08 | O GPS deve poder obter a localização do usuário. |
| RF09 | O usuário poderá digitar sua localização para uma busca. |

### Requisitos não Funcionais

|  |  |
| --- | --- |
| RNF01 | O sistema será hospedado em nuvem *Microsoft® azure.* |
| RNF02 | O banco de dados será hospedado em nuvem *Microsoft® azure.* |
| RNF03 | O banco de dados será desenvolvido a partir da plataforma *SQL Server.* |
| RNF04 | O sistema será desenvolvido a partir da linguagem de programação C#. |
| RNF05 | A interface com o sistema será desenvolvida para dispositivos *Android.* |
| RNF06 | Os dispositivos de interface deverão armazenar uma cópia do banco de dados da nuvem pelo internamente pelo *SQLite.* |

### Regras de Negócio

|  |  |
| --- | --- |
| RN01 | Serão permitidos somente cadastros com CPF ou CNPJ válidos. |
| RN02 | Somente clientes cadastrados podem aumentar um raio de pesquisa. |
| RN03 | O raio de busca inicial é de cinco quilômetros. |
| RN04 | Somente clientes cadastrados podem avaliar prestadores de serviços. |

## Modelo de Casos de Uso

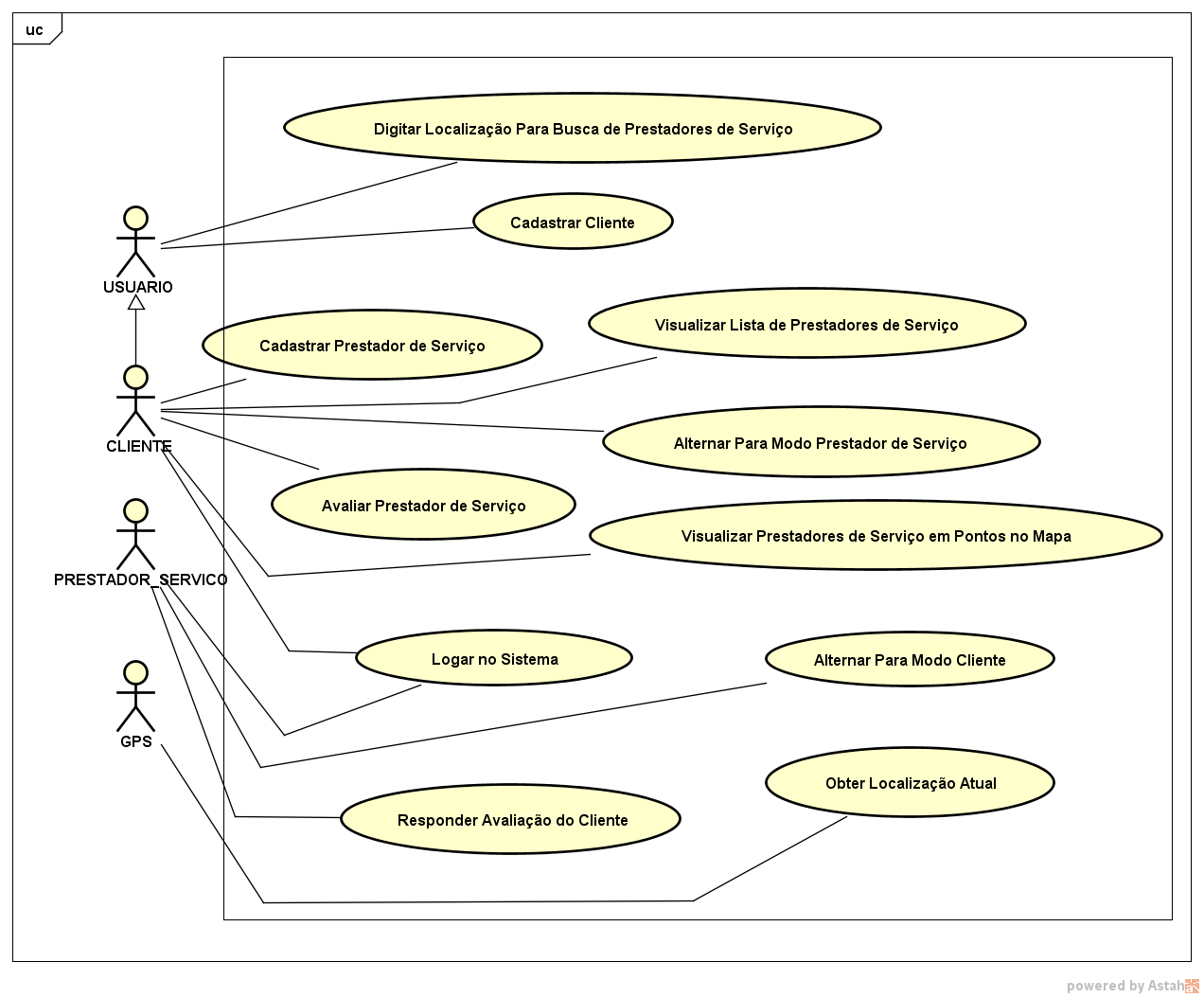
 São descritos na figura 16, os atores que desempenham a realização de todo o uso de funcionalidades apresentados na aplicação Sonar de Serviços. Usuários comuns, não cadastrados, que podem realizar consultas básicas. Clientes, habilitados à fazer consultas com mais recursos, podem também visualizar a lista de prestadores de serviço, cadastrarem-se como prestadores e alternar entre estes modos. Prestador de serviço, que responder a uma avaliação do cliente, além de alternar seu modo de acesso para o modo cliente, a fim de usufruir do aplicativo. GPS, o responsável por obter a localização atual de seu usuário, seja ele cliente cadastrado ou prestador de serviço.

Figura 16: Descrição dos Atores e Funcionalidades

## Modelo de Classes de domínio

O modelo de classes de domínio é constituído pelas classes que compõe a lógica da aplicação.

A classe de usuário contém a definição do usuário do sistema, contendo seu nome de usuário, sua senha, o cliente a qual está atrelado e um *hash* de identificação.

A classe de cliente tem os dados pessoais do cliente, assim como nome, endereço e telefones.

A classe de prestador de serviços estende a classe cliente e possui uma lista de serviços que presta na aplicação.

A classe de interesse possui uma referência do cliente interessado, do prestador interessante e do serviço que foi buscado, além de uma referência do endereço em que a busca foi feita.

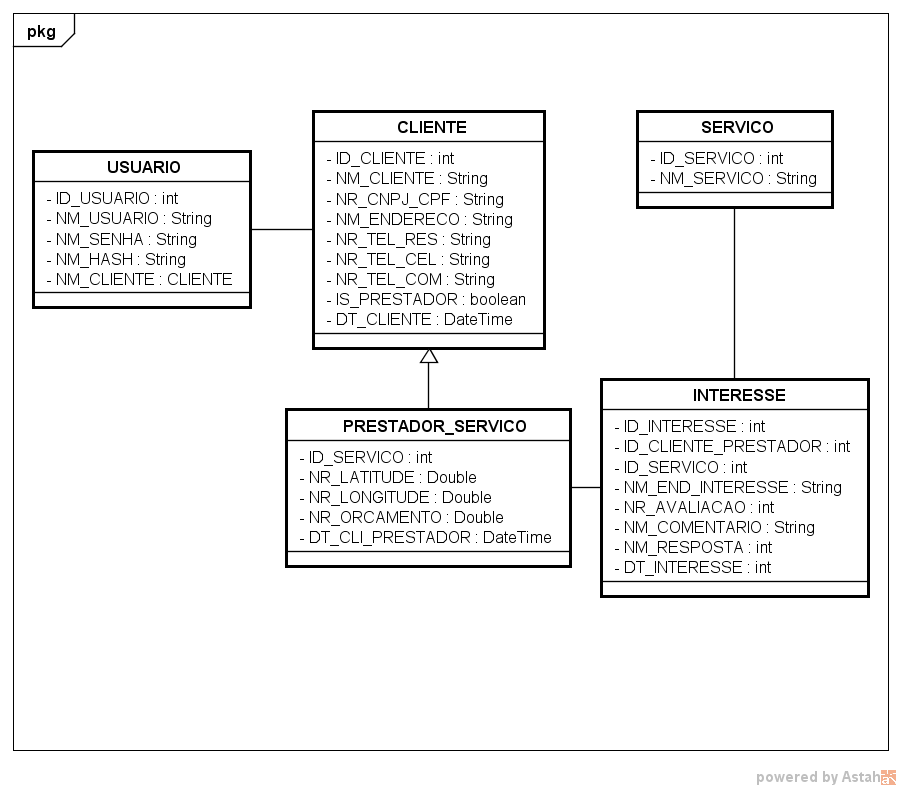
A classe de serviço possui uma identificação única e o nome do serviço.

Figura 17: Modelo de Classes de Domínio

## Modelo Conceitual do Banco de Dados

O modelo conceitual do banco de dados, mostrado na figura 18, mostra o conceito usado para se elaborar o banco de dados da aplicação Sonar de Serviços.

Neste modelo vemos que as entidades, seus relacionamentos e papéis no modelo.

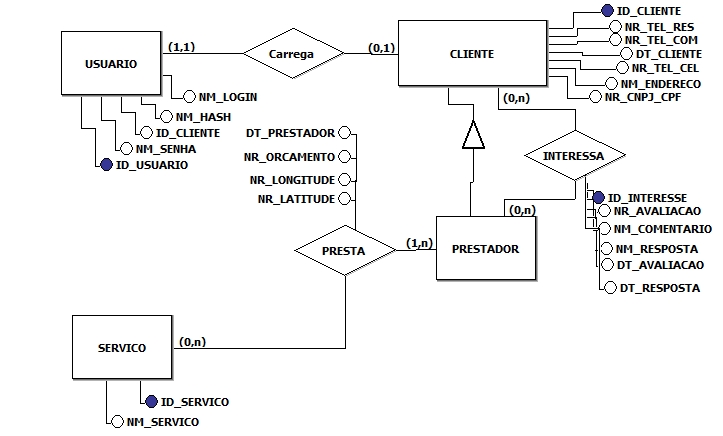


Figura 18 - Modelo Conceitual de Banco de Dados

## Modelo Lógico de Banco de Dados

O modelo lógico do banco de dados da aplicação, que pode ser visto na figura 19, exibe as entidades do banco de dados do sistema e suas respectivas relações. A relação entre um usuário e um cliente acontece por meio do cadastro de um cliente no sistema, portanto temos uma chave estrangeira nula e que será preenchida após o cadastro do cliente pelo usuário.

A relação de interesse ocorre quando um cliente visualiza os dados de um prestador por um serviço específico do mesmo, a partir de uma busca. Este interesse fica disponível na tela principal do usuário, para posterior avaliação.

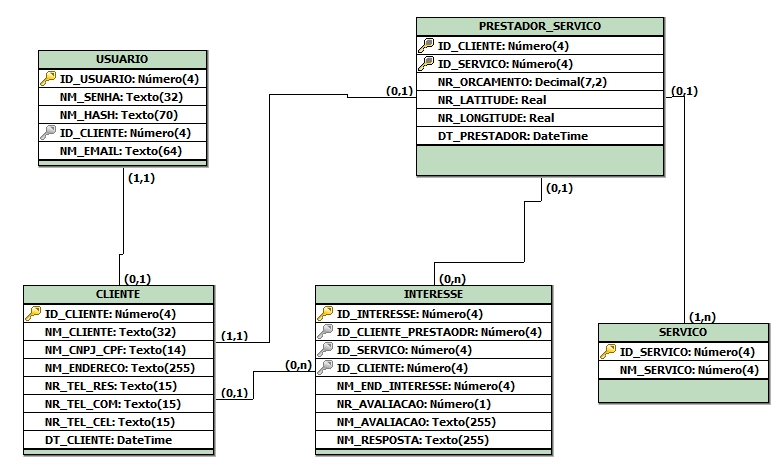


Figura 19 - Modelo lógico do banco de dados

## Modelo Físico do Banco de Dados

### Tabela USUARIO

|  |
| --- |
| CREATE TABLE USUARIO(  ID\_USUARIO INT NOT NULL PRIMARY KEY IDENTITY,  NM\_EMAIL VARCHAR(64) NOT NULL UNIQUE,  NM\_USUARIO VARCHAR(32) NOT NULL UNIQUE,  NM\_SENHA VARCHAR(32) NOT NULL,  NM\_HASH VARCHAR(70),  ID\_CLIENTE INT,  CONSTRAINT FK\_USUARIO\_CLIENTE  FOREIGN KEY (ID\_CLIENTE)  REFERENCES CLIENTE (ID\_CLIENTE)  ON DELETE CASCADE  ON UPDATE NO ACTION  ); |

### Tabela CLIENTE

|  |
| --- |
| CREATE TABLE CLIENTE(  ID\_CLIENTE INT NOT NULL PRIMARY KEY IDENTITY,  NM\_CLIENTE VARCHAR(80) NOT NULL,  NR\_CNPJ\_CPF VARCHAR(14) NOT NULL UNIQUE,  NM\_ENDERECO VARCHAR(255) NOT NULL,  NR\_TEL\_CEL VARCHAR(15),  NR\_TEL\_COM VARCHAR(15),  NR\_TEL\_RES VARCHAR(15),  IS\_PRESTADOR VARCHAR(1) NOT NULL DEFAULT 'F',  DT\_CLIENTE DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP  ); |

### Tabela SERVICO

|  |
| --- |
| CREATE TABLE SERVICO(  ID\_SERVICO INT NOT NULL PRIMARY KEY IDENTITY,  NM\_SERVICO VARCHAR(100) NOT NULL  ); |

### Tabela PRESTADOR\_SERVICO

|  |
| --- |
| CREATE TABLE PRESTADOR\_SERVICO(  ID\_CLIENTE INT NOT NULL,  ID\_SERVICO INT NOT NULL,  NR\_ORCAMENTO DECIMAL(7,2) NOT NULL DEFAULT 0.00,  NR\_LATITUDE REAL NOT NULL,  NR\_LONGITUDE REAL NOT NULL,  DT\_PRESTADOR DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY (ID\_CLIENTE, ID\_SERVICO),  CONSTRAINT FK\_CLI\_PRESTADOR  FOREIGN KEY (ID\_CLIENTE)  REFERENCES CLIENTE (ID\_CLIENTE)  ON DELETE CASCADE  ON UPDATE CASCADE,  CONSTRAINT FK\_SER\_PRESTADOR  FOREIGN KEY (ID\_SERVICO)  REFERENCES SERVICO (ID\_SERVICO)  ON DELETE CASCADE  ON UPDATE CASCADE  ); |

### Tabela INTERESSE

|  |
| --- |
| CREATE TABLE INTERESSE(    ID\_INTERESSE INT NOT NULL IDENTITY,  ID\_CLIENTE\_PRESTADOR INT NOT NULL,  ID\_SERVICO INT NOT NULL,  ID\_CLIENTE INT,  DT\_INTERESSE  DATETIME NOT NULL DEFAULT CURRENT\_TIMESTAMP,  NM\_END\_INTERESSE VARCHAR(256) NOT NULL,  NR\_AVALIACAO INT NOT NULL DEFAULT 0,  NM\_COMENTARIO VARCHAR(256),  NM\_RESPOSTA VARCHAR(256),  DT\_AVALIACAO DATETIME,  PRIMARY KEY (ID\_INTERESSE),  CONSTRAINT FK\_CLI\_INTERSSE  FOREIGN KEY (ID\_CLIENTE)  REFERENCES CLIENTE(ID\_CLIENTE)  ON DELETE CASCADE  ON UPDATE NO ACTION,  CONSTRAINT FK\_SER\_INTERESSE  FOREIGN KEY (ID\_CLIENTE\_PRESTADOR, ID\_SERVICO)  REFERENCES PRESTADOR\_SERVICO(ID\_CLIENTE, ID\_SERVICO)  ON DELETE NO ACTION  ON UPDATE NO ACTION  ); |

### Criação de Índices

|  |
| --- |
| CREATE INDEX IDX\_INTERESSE ON INTERESSE(NR\_AVALIACAO);  CREATE INDEX IDX\_PRESTADOR ON INTERESSE(ID\_CLIENTE\_PRESTADOR, ID\_SERVICO, ID\_CLIENTE);  CREATE INDEX IDX\_PRESTADOR ON CLIENTE(IS\_PRESTADOR);  CREATE INDEX IDX\_CLI\_PREST ON PRESTADOR\_SERVICO(ID\_CLIENTE, ID\_SERVICO);  CREATE INDEX IDX\_USUARIO ON USUARIO(NM\_USUARIO);  CREATE INDEX IDX\_HASH ON USUARIO(NM\_HASH);  CREATE INDEX IDX\_CLIENTE\_EMAIL ON USUARIO (NM\_EMAIL);  CREATE INDEX IDX\_NM\_SERVICO ON SERVICO(NM\_SERVICO);  CREATE INDEX IDX\_CPF\_CNPJ ON CLIENTE(NR\_CNPJ\_CPF);  CREATE INDEX IDX\_CLIENTE\_ENDERECO ON CLIENTE (NM\_ENDERECO); |

## Interface do Sistema

A interface da aplicação Sonar de Serviços será feita através de um dispositivo *Android,* o qual o protótipo será descrito nesta seção.

### Tela de Abertura

Ao iniciarmos a aplicação é mostrada a tela de apresentação, mostrada na figura 20. A tela do presente protótipo conta com uma imagem de logotipo, o nome da aplicação o logotipo da instituição a qual se destina o presente trabalho.



Figura 20 Tela de Abertura do Sonar de Serviços.

### Tela Inicial

Após a tela de abertura, se o usuário não for validado no sistema, o mesmo se deparará com a tela presente na figura 21, que conta com duas abas, sendo uma para busca e outra para *login*, onde o usuário possui acesso ao cadastro de usuário.

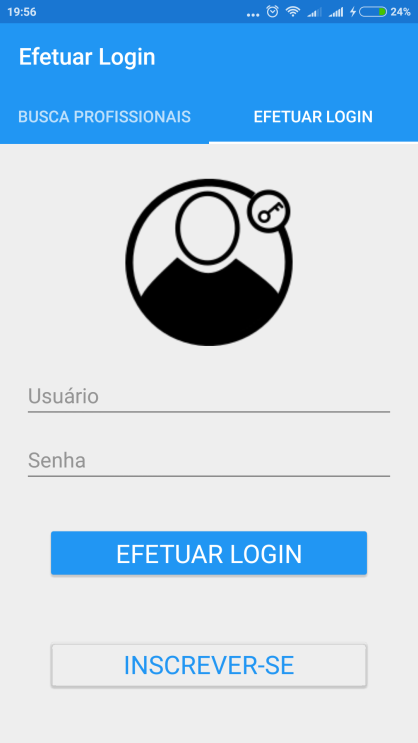


Figura 21 Telas de início para usuários não autenticados.

### Cadastro de Usuários

Caso o usuário clique no botão inscrever-se, presente na tela de *login* da figura 21, o mesmo será levado à tela para cadastro de usuário, que coleta informações de email, nome de usuário e senha, como mostrado na figura 22.

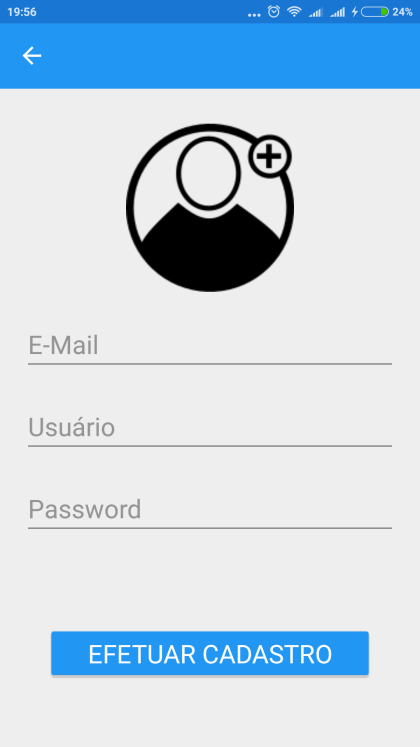


Figura 22 Tela de Cadastro de usuário do sistema

### Tela inicial para usuários autenticados

Após a autenticação ou após o registro de um nome de usuário será exibida a tela inicial para usuários registrados, mostrada na figura 23, que permite ao usuário o aumento do raio de uma busca, o acesso à sua atividade como cliente e permite que o mesmo se habilite como prestador de um dos serviços prestados pela aplicação.

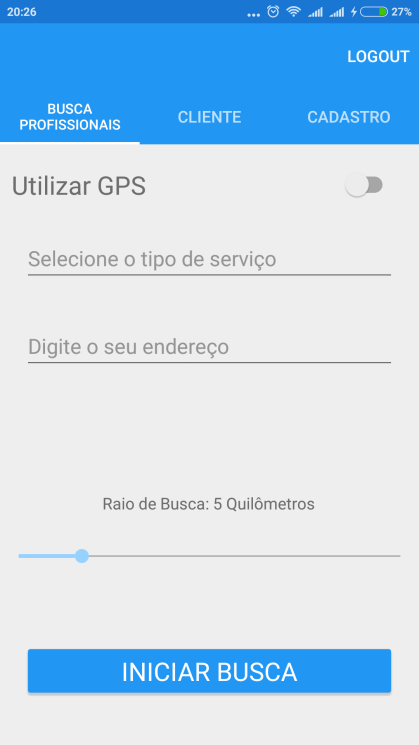


Figura 23 - Tela inicial para usuários autenticados

### Telas de Atividades como Cliente

A aba de clientes apresentada na figura 24 mostra as cinco últimas visualizações de prestadores de serviços feitas pelo cliente e permite o mesmo efetuar avaliações de prestadores de serviços aos quais interagiu, escolhendo uma nota entre um e cinco, além de um comentário curto.

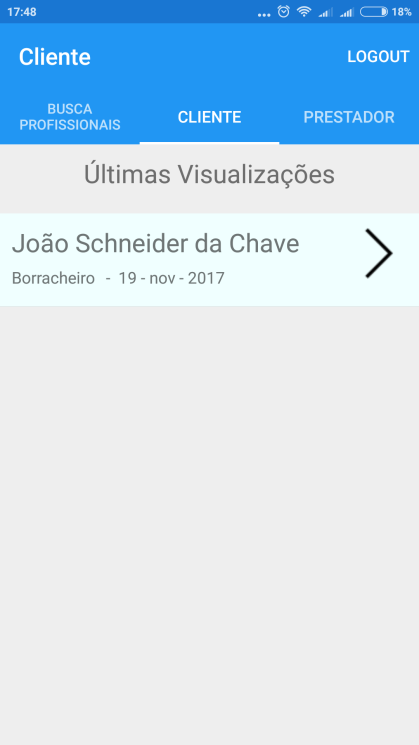


Figura 24 - Telas de atividades do cliente e avaliação

### Telas de Registro como Prestador de Serviço

Caso o usuário pretenda se tornar um prestador de serviços ele pode acessar a tela de cadastro mostrada na figura 25, que conta com a lista de serviços oferecidos pelo sistema, onde ele pode escolher os que deseja prestar.

Ao marcar um serviço o usuário é levado à tela que permite informar um endereço e um valor base de orçamento, e ao clicar em concluir ele retorna para a tela de cadastro, onde pode clicar no botão de efetuar cadastro, tornando-se um prestador dos serviços escolhidos.

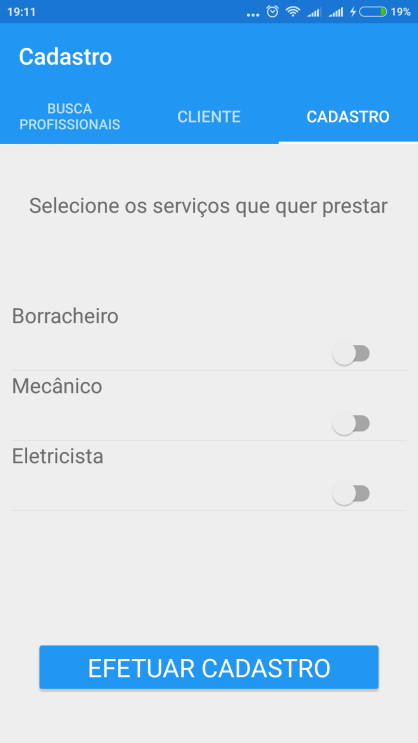


Figura 25 Telas de Registro de Prestador de Serviço

### Tela de Atividades como Prestador de Serviços

Um prestador registrado tem acesso à tela de atividades como prestador de serviços, mostrada na figura 26, na qual ele pode visualizar os últimos cinco clientes que o avaliara e pode responder a esta avaliação com um comentário curto.

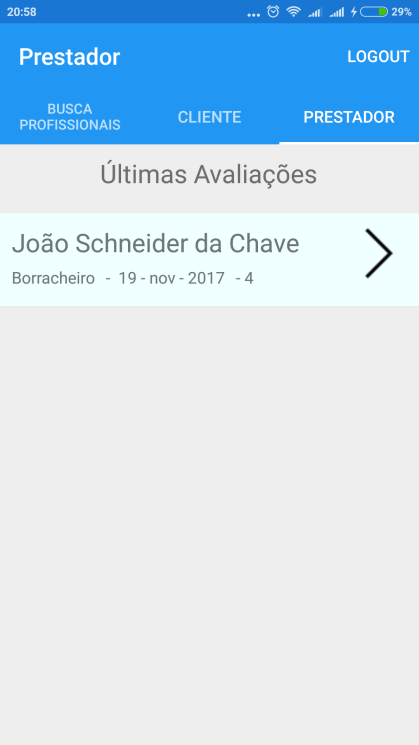


Figura 26 - Tela de atividades como prestador de serviços

### Exibindo resultados de uma busca

Após uma busca bem sucedida por prestadores de serviço, é exibido ao usuário a tela inicial de resultados da figura 27, que conta permite uma visualização de resultados em formato lista ou em pontos no mapa.

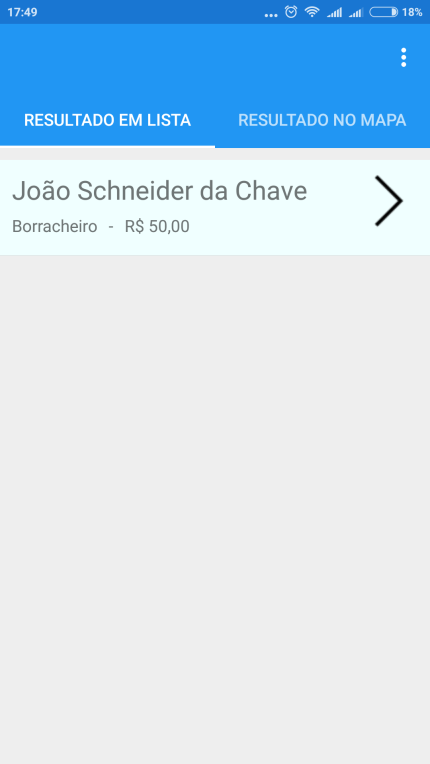


Figura 27 Resultados de uma busca de profissionais

Caso o usuário toque em qualquer item da lista ou na informação que é exibida quando ele clica em um ponto do mapa, ele é levado a uma janela de detalhes desse usuário, que contém informações como nome, números de telefone, endereço, atividade e valor base de orçamento do mesmo, conforme mostra a figura 28.



Figura 28 Visualização de prestadores de serviço por lista e por mapa

# CONCLUSÃO

A computação em nuvem torna o desenvolvimento de soluções de software bem mais fácil, disponibilizando toda uma infraestrutura e plataformas que poupam o investimento inicial do projeto. A plataforma *Microsoft Azure* possui uma interface intuitiva que facilita a criação de aplicações menos complexas, até ambientes completos, por meio de seu serviço de máquinas virtuais.

A integração entre a *Microsoft Visual Studio* e a plataforma *Azure* permite o desenvolvimento local de soluções remotas, além de integração com o *SQL Server*, poupando o desenvolvedor do trabalho de dominar diferentes plataformas, de mudar de plataforma e de implantar a solução manualmente na nuvem. A interface de desenvolvimento se encarrega de todo o trabalho de implantação do software no ambiente de nuvem, incluindo suas configurações, sem a interação do usuário no processo.

A opção por utilizar um desenvolvimento *Xamarin* melhora a produtividade e facilita a criação, tendo visto que o foco é voltado para a aplicação compartilhada, sem a necessidade de aprendizado de novas linguagens e utilização de interfaces de desenvolvimento diversas, bastando apenas o aprendizado da linguagem C#, da linguagem de marcação XAML e da interface de desenvolvimento *Visual Studio*, que possuem vasta documentação centralizada na interne, seja pela própria Microsoft, seja pelos fóruns com foco no assunto.

O Projeto Sonar de Serviços foi sem dúvidas um desafio prazeroso, pois exigiu dedicação e aprofundamento nas disciplinas envolvidas no projeto, desde a criação de modelos ao seu primeiro protótipo, porém muitas coisas podem ser adicionadas ao mesmo para aprimorá-lo, seja adicionando novas funcionalidades, seja modificando funcionalidades existentes.

Nesta versão inicial, o aplicativo não apresenta um mensageiro para que clientes e prestadores de serviço se comuniquem em um interesse, não possui uma funcionalidade de adição de fotos de serviços realizados por prestadores a clientes cadastrados, e estes itens podem ser adotados em versões posteriores, para complementar as funcionalidades da aplicação.

Por mais que tenha sido adotado um modelo *cross-platform* de desenvolvimento com o *Xamarin,* que permite desenvolver uma aplicação que gera executáveis distintos para *IOS, Android* e *Windows Universal Platform*, esta versão conta somente com interface *Android,* por conta do tempo de projeto e da necessidade de customizar a visualização por mapas, alteração que deve ser feita para cada plataforma, fugindo do código compartilhado e interferindo na portabilidade. Para o futuro do projeto, a geração de executáveis para as outras plataformas se faz possível, aumentando assim o nicho da aplicação.

# Bibliografia

DXC Technology. (17 de 08 de 2017). *O Que é O Cloud Computing? | DXC Technology*. Acesso em 21 de 08 de 2017, disponível em DXC Technology: http://www.dxc.technology/pt/offerings/63346-o\_que\_%C3%A9\_o\_cloud\_computing

HAUTSCH, O. (13 de Novembro de 2008). *Windows Azure: seu computador vai às nuvens - TecMundo*. Acesso em 24 de Agosto de 2017, disponível em TechMundo: https://www.tecmundo.com.br/web/943-windows-azure-seu-computador-vai-as-nuvens.htm

Imamura, D. (19 de Maio de 2011). *Windows Azure: o que é?* Acesso em 24 de Agosto de 2017, disponível em Oficina da Net: https://www.oficinadanet.com.br/artigo/windows/windows-azure-o-que-e

Microsoft. (01 de Janeiro de 2017 ). *Microsoft Azure: Plataforma e serviços de computação em nuvem*. Acesso em 24 de Agosto de 2017, disponível em Microsoft: https://azure.microsoft.com/pt-br/

Pinheiro, F. (2010). *Cloud Computing.* Acesso em 21 de 08 de 2017, disponível em Universidade Federal do Rio de Janeiro: https://www.gta.ufrj.br/ensino/eel879/trabalhos\_vf\_2010\_2/fernando/index.html

The NIST Definition of Cloud Computing. (10 de 2011). *National Institute of Standards and Technology Special Publication 800-145* , p. 7.

.NET Coders. (20/09/2015). Introdução ao MVVM (Model-View-ViewModel). Acesso em 12 de Novembro de 2017, disponível em .NET Coders: http://netcoders.com.br/introducao-ao-mvvm/

iMasters. (18/11/2010). Entendendo o Pattern Model-View-ViewModel (MVVM). Acesso em 12 de Novembro de 2017, disponível em iMasters: https://imasters.com.br/artigo/18900/desenvolvimento/entendendo-o-pattern-model-view-viewmodel-mvvm?trace=1519021197&source=single

Lambda3. (10/10/2016). O que é Xamarin?. Acesso em 13 de Novembro de 2017, disponível em Lambda3: https://www.lambda3.com.br/2016/10/o-que-e-xamarin/

Rodrigo Mourão. (11/06/2015). Desenvolvimento Mobile Cross Platform. Acesso em 15 de Setembro de 2017, disponível em Rodrigo Mourão: http://www.rodrigomourao.com.br/desenvolvimento-mobile-cross-platform/

TechTudo. (05/12/2012). Android ou IOS? Entenda as diferenças entre os sistemas mobile. Acesso em 10 de Setembro de 2017, disponível em TechTudo: http://www.techtudo.com.br/artigos/noticia/2011/12/iphone-ou-android-entenda-diferenca-entre-os-sistemas-mobile.html

DevMedia. Android: Plataforma Android - Java. Acesso em 10 de Setembro de 2017, disponível em DevMedia: https://www.devmedia.com.br/android-plataforma-android-java/17840

UseMobile. (12/04/2017). Sistema Android e iOS: quais suas principais diferenças?. Acessado em 11 de Setembro de 2017, disponível em UseMobile: http://usemobile.com.br/sistema-android-ios/

OlharDigital. (26/09/2016). Android, iOS, e Windows Phone: saiba tudo sobre os sistemas operacionais. Acessado em 11 de Setembro de 2017, disponível em OlharDigital: https://olhardigital.com.br/lu-explica/noticia/android-ios-e-windows-phone-saiba-tudo-sobre-os-sistemas-operacionais/62473