Centro Universitário de Brasília - UniCEUB
Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS

FILLIPE DE SOUSA MOURA

SOLUÇÃO DE VISTORIA PREDIAL BASEADO EM LEITURA DE QR CODE

Brasília, DF – BRASIL
NOVEMBRO DE 2016

FILLIPE DE SOUSA MOURA

Solução de Vistoria Predial baseado em leitura de QR Code

Trabalho de Conclusão de Curso apresentado à Banca examinadora do curso de Engenharia da computação da FATECS – Faculdade de Tecnologia e Ciências Sociais Aplicadas – Centro Universitário de Brasília como requisito para obtenção do título de Bacharel em Engenharia da Computação.

Orientador: Prof. MsC Francisco Javier de Obaldia Diaz

BRASÍLIA, DF – BRASIL

NOVEMBRO DE 2016

FILLIPE DE SOUSA MOURA

Solução de Vistoria Predial baseado em leitura de QR Code

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Este trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Dr. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof. MS Francisco Javier de Obaldía Díaz.
Orientador

_____

Prof. MS Layany Damázio
Membro da banca

_____

Prof. MS Ivandro Ribeiro
Membro da banca

AGRADECIMENTOS

Primeiramente quero agradecer a Deus por guiar minha vida e sempre estar à frente dos meus passos. Com Ele no comando, conquistarei o meu sonho de se tornar engenheiro. Também desejo prestar meus agradecimentos a minha família, amigos e professores que me apoiaram durante todo esse período na universidade.

“O sábio antevê o perigo e protege-se, mas os imprudentes passam e sofrem as consequências.” (Provérbios: 22:3).

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E SÍMBOLOS

AOT - Ahead-Of-Time
API – Aplication Programming Interface
BD – Banco de Dados
BI - Business Intelligence
CBMDF - Corpo de Bombeiros Militar do Distrito Federal
C# - C Sharp
CIL - Common Intermediate Language
CLI - Common Language Infrastructure
CLR - Common Language Runtime
Código QR - Quick Response
CPU - Central Processing Unit
DTUs - Database Throughput Units
EF - Entity Framework
FCL - Framework Class Library
FK – Foreign Key
HTTP – Hypertext Transfer Protocol
IDC - International Data Corporation
IDE - Integrated Development Environment
IL - Intermediate Language
ILDasm - Intermediate Language Disassembler
IN – Instrução Normativa
Intel® HAXM - Intel® Hardware Accelerated Execution Manager
JIT - Just In Time
LINQ - Language Integrated Query
JSON – JavaScript Object Notation
MSDN - Microsoft Developer Network
MSIL - Microsoft Intermediate Language
OLTP - Online Transaction Processing
OS – Operating System
QR Code - Quick Response Code
RSIP-DF - Regulamento de Segurança Contra Incêndio e Pânico do Distrito Federal
SGBD - Sistema Gerenciador de Banco de Dados
SDK – Software Development KIT
SO – Sistema Operacional
SSMS - SQL Server Management Studio
T-SQL – Transact SQL
UI - User Interface

RESUMO

O propósito desta monografia é desenvolver um aplicativo para smartphones que possa ser instalado em sistema operacional Android e o seu back-end seja armazenado na nuvem da Microsoft, o Azure. O aplicativo deve ser capaz de realizar vistorias em edificações, buscando ter um controle exato de tudo a ser vistoriado e não deixar esquecido nenhum local ou equipamento. A segurança de um local privado que possua visitação constante do público, é de inteira responsabilidade dos seus proprietários. Consequentemente, sendo também sua responsabilidade a segurança do local de trabalho dos seus funcionários.

Cada local ou equipamento a ser vistoriado possuirá etiquetado um código QR, conhecido como QR Code, desta forma, cada um terá sua identidade própria, evitando que haja fraudes em uma vistoria. O aplicativo confirma que o local foi vistoriado somente após a análise do QR Code feita pela câmera do smartphone. Em caso de inconformidades encontradas durante a vistoria, o aplicativo apresentará a opção de tirar uma foto, a qual será armazenada no banco de dados do sistema e associada a vistoria realizada. Ao fim de cada vistoria o aplicativo deverá realizar a validação da mesma, enviando por e-mail um resumo da vistoria.

O sistema será composto por 7 componentes básicos, sendo eles, o dispositivo móvel, a sincronização offline, o tipo de comunicação sem fio, a nuvem Azure, o serviço de aplicativo, o gerenciamento de API e o banco de dados SQL. Com o desenvolver do código do sistema, todos os componentes irão trabalhar de forma transparente para o usuário final, mascarando o local aonde cada requisição será processada.

Palavras-chave: Vistoria, Segurança no Trabalho, Nuvem, Azure Android, QR Code e Xamarin.

ABSTRACT

The purpose of this monograph is to develop a smartphone application that can be installed on the Android operating system and your back-end is stored in the Microsoft cloud, Azure. The application must be able to make inspections in buildings, seeking to have an exact control of everything to be inspected and not to leave forgotten any place or equipment. The security of a private place that has constant public visitation, is the responsibility of its owners. Consequently, it is also their responsibility to ensure the safety of their employees' workplaces.

Each place or equipment to be inspected will have a QR Code tagged, in this way, each one will have its own identity, avoiding that there will be fraud in an inspection. The application confirms that the equipment was scanned only after QR Code analysis by the smartphone camera. In case of nonconformities found during the survey, the application will have the option of taking a photo, which will be stored in the system database and associated with the survey performed. At the end of each survey, the application must validate it by sending a summary of the inspection by e-mail.

The system will consist of 7 basic components, such as the mobile device, offline synchronization, wireless communication type, Azure cloud, application service, API management and SQL database. With the development of system code, all components will work seamlessly to the end user, masking the location where each request will be processed.

Keywords: Inspect, Work Safety, Cloud, Azure, Android, QR Code and Xamarin.

CAPÍTULO 1 – INTRODUÇÃO

1.1 Apresentação do Problema

Edificações construídas em território brasileiro devem proporcionar um nível mínimo de condições de segurança contra incêndio e pânico para o seu pleno funcionamento. No caso de Brasília, tais exigências são estabelecidas na INSTRUÇÃO NORMATIVA Nº 002/2016 - DIVIS/DESEG, previstos no Regulamento de Segurança Contra Incêndio e Pânico do Distrito Federal (RSIP-DF).

Agentes Fiscalizadores do Corpo de Bombeiros Militar do Distrito Federal (CBMDF) utilizam os procedimentos básicos estabelecidos nesta IN para realizar vistorias técnicas, analisando as condições de segurança de todos estabelecimentos.

Visando manter segura a vida dos usuários do local, proprietários das edificações realizam vistorias programadas por conta própria. Habilitando um de seus funcionários para verificar todas as áreas e equipamento que auxiliem no controle de riscos de cada edificação. O controle de risco tem por objetivo tornar os possíveis riscos abaixo de valores tolerados, evitando que quaisquer riscos se tornem fatos reais de emergência.

A falta de um sistema inteligente que faça a gestão de tais vistorias em edificações de grande porte, como shoppings, escolas e aeroportos, aumenta as chances de ocorrer uma falha em um dos equipamentos e ser identificado somente em casos de emergência. O método mais utilizado hoje nas vistorias utiliza o apoio de tabelas criadas no Microsoft Excel, onde o funcionário apenas assina se comprometendo que vistoriou todos aqueles locais.

Um sistema que possa informar quais locais devem ser vistoriados, como deve ser feita essa verificação, a pessoa autorizada para realizar tal vistoria, realizar uma validação eletrônica que o vistoriador esteve no local, como são tratadas as inconformidades quando encontradas ou até mesmo realizar auditoria sobre essas vistorias, trariam grande conforto e segurança aos gestores de grandes empreendimentos.

Com a criação dessa tecnologia a tendência é minimizar os riscos e mantê-los abaixo dos valores tolerados. Embora já sejam feitas vistorias, não são feitas da maneira mais adequada.

Será que com o desenvolvimento de um sistema provendo uma real confirmação de que todos os locais a serem vistoriados foram realmente vistoriados e que o relatório emitido por ele, pode servir como objeto de validação do processo?

1.2 Objetivos do Trabalho:

1.2.1 Objetivo geral

Desenvolver um sistema de apoio a vistorias de todo um complexo predial. Visando manter o complexo sempre em conformidade contra falhas elétricas, falhas hidráulicas, falhas de segurança e incêndios.

1.2.2 Objetivos específicos

- Criar mecanismo de identificação Quick Response Code (QR Code) para itens a serem vistoriados;
- Realizar auditoria digital dos locais a serem vistoriados, validando a vistoria por meio de leitura do QR Code em cada um dos locais;
- Desenvolver o código em linguagem C Sharp para o aplicativo de smartphone;
- Desenvolver o código em linguagem C Sharp para o sistema web;
- Promover a comunicação entre o aplicativo e o sistema web;
- Disponibilizar relatório com resultados da vistoria;

1.3 Justificativa e Importância do Trabalho

Quando se trata de empresas de grande porte, o número de pessoas que visitam diariamente as suas edificações é bem expressivo. Portanto, um único erro que ocorra na segurança predial pode causar um incêndio, exposição à perigos externos e até mesmo a danos a saúde das pessoas.

De acordo com o Departamento do trabalho dos Estados Unidos, foram feitas pesquisas, chamadas de Revisions to the 2014 Census of Fatal Occupational Injuries (Revisão do Censo de Acidentes Fatais no Trabalho de 2014) (United States Departament of Labor, 2016):

> The final count of fatal work injuries in the United States in 2014 was **4,821**, up from the preliminary count of 4,679 reported in September 2015 and the highest annual total since 2008. The overall fatal work injury rate for the United States in 2014 was 3.4 fatal injuries per 100,000 full-time equivalent (FTE) workers, slightly higher than the final rate of 3.3 reported for 2013. The higher overall rate in 2014 is the first increase in the national fatal injury rate since 2010.

A solução proposta irá gerar um relatório com o status de segurança de cada objeto vistoriado das edificações, demonstrando o quão seguro o ambiente está ou não está. Com foco em quais medidas devem ser tomadas para melhorar a segurança predial.

1.4 Metodologia

A primeira modalidade de pesquisa adotada foi a pesquisa de campo, realizando entrevistas com profissionais da área de Segurança predial e Manutenção predial. Segundo a visão deles, que tem como principal função proporcionar um nível básico de segurança aos seus clientes, a vistoria de todo o complexo deverá ser feita cotidianamente e agendada, cumprindo todas as atividades e horários determinados. Especificações de atividades e horários serão definidas pelos gestores de cada empreendimento, por tanto, sendo de forma padronizável no sistema.

A jurisprudência utilizada como base para pesquisa foi criada pelo Governo do Distrito Federal e o Corpo de Bombeiros Militar do Distrito Federal. As exigências para um correto funcionamento do complexo predial, a fim de se evitar incêndio e pânico, estão contidas na INSTRUÇÃO NORMATIVA Nº 002/2016 do Departamento de Segurança Contra Incêndio, da Diretoria de Vistorias.

Esta vistoria visa verificar as condições necessárias para garantir o correto funcionamento dos equipamentos de combate a incêndio e demais necessidades de segurança do local, bem como a segurança das pessoas.

Cada equipamento ou área que fizer parte da vistoria, deverá ter um QR Code próprio gerado pelo sistema. Este QR Code deverá ser impresso e colado ao equipamento, portanto, cada equipamento terá sua ‘identidade’ própria.

Dessa forma, a vistoria será feita por um funcionário do condomínio portando um smartphone. O smartphone terá instalado um aplicativo, o qual será desenvolvido como parte desse projeto com o software Visual Studio utilizando as ferramentas do software Xamarin. O aplicativo terá uma tela de inicio com autenticação para usuários, com isso, cada usuário terá sua própria lista de vistorias a fazer. O aplicativo fará a leitura do QR Code e identificação do equipamento vistoriado fazendo uma busca em banco de dados, logo após mostrará os requisitos que deverão ser vistoriados no equipamento.

O aplicativo fará a transmissão dos dados para o aplicativo web de acordo com um Job em seu código, mantendo sempre a base de dados atualizada.

O sistema deverá emitir um relatório que demonstre todos os componentes vistoriados e status da vistoria. O relatório será enviado para os gestores do condomínio e para seus condôminos por um servidor de e-mail. Irregularidades encontradas na vistoria serão fotografadas e também farão parte do relatório, podendo ser estabelecidas formas de punição aos condôminos.

1.5 Resultados Esperados

O sistema será capaz de:

- Realizar vistorias em todo o complexo predial por meio do aplicativo para smartphone:
    - Combate e prevenção de incêndios;
    - Controle de inspeção em condôminos;
    - Controle de inspeção em condôminos de alimentação;

- Geração de relatório demonstrando os resultados das vistorias;
- Enviar automaticamente para o e-mail dos administradores do sistema e condôminos, os relatórios gerados pelo sistema;

1.6 Estrutura do Trabalho

O conteúdo deste trabalho está subdividido em 5 capítulos, cada um deles com o objetivo de explanar uma parte do projeto, sendo eles:

No capítulo 1 foi desenvolvida a introdução. O capítulo 2 demonstra quais foram os embasamentos teóricos para a realização do trabalho. O capítulo 3 apresenta o protótipo da solução. O capítulo 4 traz os resultados da solução e quais foram os testes realizados. E por fim, o capítulo 5 apresenta as conclusões finais.

CAPÍTULO 2 – REFERÊNCIAL TEÓRICO

2.1 Segurança no Trabalho

“A segurança é uma condição ou estado do que está livre de danos ou riscos. Situação em que nada há a temer.” (MICHAELIS, 2016)

Todos os trabalhadores de uma organização têm direito a trabalhar em ambientes em que os riscos para a sua saúde e segurança são devidamente controlados. Perante a lei, a responsabilidade de segurança dos locais é dos empregadores, eles devem obedecer a um conjunto de medidas e normas que previnem o acontecimento de acidentes no trabalho.

Exercer a segurança no local de trabalho tem o intuito de reduzir a zero a incidência de acidentes. Portanto, proporcionar um ambiente seguro requer um estudo na prevenção de acidentes. É importante citar que a prevenção de acidentes leva em consideração os danos que causam agravos ao patrimônio, destruição ao meio ambiente e a perdas de pessoas. Para Cardella (2014, p. 70)

> I - Nas organizações e sociedades, o acidente é um fenômeno de natureza multifacetada, que resulta de interações complexas entre fatores físicos, biológicos, psicológicos, sociais e culturais.
> II - Todos os acidentes podem ser evitados.
> III -“Os acidentes ocorrem porque a mente se envolve com o trabalho e esquece do corpo.”
> IV - Um indivíduo não consegue, sozinho, controlar os riscos de sua atividade.

A prevenção de acidentes prevê que pelo menos uma pessoa seja responsável pela segurança do local em um meio onde se queira manter seguro, tal pessoa deve ser responsabilizado pelos erros cometidos ali. Nenhum sistema eletrônico é cem por cento confiável, caso ele venha a apresentar uma falha, a condução da emergência depende da interferência humana. A condução de uma liderança no processo de segurança no trabalho desenvolve um facilitador em exercer duas funções, a primeira é controlar riscos e a segunda é controlar emergências.

O dever de manter um local seguro não é exclusivamente de uma única pessoa, podendo congregar várias equipes de uma instituição, como, equipe de seguranças, equipe de brigadistas, equipe de limpeza, equipe de manutenção e etc. Os funcionários têm o dever

de cuidar de sua própria saúde e segurança de outras pessoas que possam ser afetadas por suas ações no trabalho. Os trabalhadores devem cooperar com os empregadores e colegas de trabalho para um ajudar ao outro.

Caso um empregado venha a ter dúvidas ou preocupações relativas à saúde e segurança no seu local de trabalho, tais questões deveram ser tratadas com seu empregador.

2.2 Legislação de Segurança e Saúde no Trabalho

Em empresas privadas e públicas, a Engenharia de Segurança e Medicina do trabalho se tornou item de suma importância na construção de edifícios a partir da década de 70. Decorrente da criação da Lei Nº 6.514, de 23 de Dezembro de 1977, escrito no Diário Oficial da União:

> "Altera o Capítulo V do Título II da Consolidação das Leis do Trabalho, relativo à Segurança e Medicina do Trabalho.
> O Presidente da República. Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:
> Art. 1o O Capítulo V do Título II da Consolidação das Leis do Trabalho, aprovada pelo Decreto-lei no 5.452, de 1o de maio de 1943, passa a vigorar com a seguinte redação:
> Capítulo V
> DA SEGURANÇA E DA MEDICINA DO TRABALHO [...]"

No ano seguinte, na data de 8 de junho de 1978, é publicado no Diário Oficial da União pelo Gabinete do Ministro do Ministério do Trabalho, nova portaria especificando as Normas Regulamentadoras da Consolidação das Leis do Trabalho, relativas à Segurança e Medicina do Trabalho. Normas Regulamentadoras são documentos, adotados por uma autoridade com poder legal para tanto, que contêm regras de caráter obrigatório para as empresas (NUNES, Flávio Oliveira; 2014). Estas estão atualmente em vigor. Vide abaixo:

> "MINISTÉRIO DO TRABALHO GABINETE DO MINISTRO
> PORTARIA No 3.214, DE 8 DE JUNHO DE 1978[1]
> Aprova as Normas Regulamentadoras – NR – do Capítulo V, Título II, da Consolidação das Leis do Trabalho, relativas à Segurança e Medicina do Trabalho.

O MINISTRO DO ESTADO, no uso de suas atribuições legais, considerando o disposto no artigo 200, da Consolidação das Leis do Trabalho, com redação dada pela Lei no 6.514, de 22 de dezembro de 1977,

RESOLVE:

Art. 1o Aprovar as Normas Regulamentadoras – NR – do Capítulo V, Título II, da Consolidação das Leis do Trabalho, relativas à Segurança e Medicina do Trabalho:

NORMAS REGULAMENTADORAS

NR-1 – Disposições gerais

NR-2 – Inspeção prévia

NR-3 – Embargo e interdição

NR-4 – Serviço especializado em segurança e medicina do trabalho – SSMT

NR-5 – Comissão interna de prevenção de acidentes – CIPA

NR-6 – Equipamento de proteção individual – EPI

NR-7 – Exames médicos

NR-8 – Edificações

NR-9 – Riscos ambientais

NR-10 – Segurança em instalações e serviços de eletricidade[2]

NR-11 – Transporte, movimentação, armazenagem e manuseio de materiais

NR-12 – Máquinas e equipamentos

NR-13 – Vasos sob pressão

NR-14 – Fornos

NR-15 – Atividades e operações insalubres

NR-16 – Atividades e operações perigosas

NR-17 – Ergonomia

NR-18 – Obras de construção, demolição e reparos

NR-19 – Explosivos

NR-20 – Combustíveis líquidos e inflamáveis

NR-21 – Trabalho a céu aberto

NR-22 – Trabalhos subterrâneos

NR-23 – Proteção contra incêndios

NR-24 – Condições sanitárias dos locais de trabalho

NR-25 – Resíduos industriais

NR-26 – Sinalização de segurança

NR-27 – Registro de profissionais

NR-28 – Fiscalização e penalidades”

[1] *DOU* de 6-7-1978 (Suplemento).

[2] Redação dada pela Portaria nº 598, de 7-12-2004. *DOU* de 8-12-2004.

Com o decorrer dos anos fez-se necessária criação de portarias para a complementação da Portaria Nº 3.214. Sendo estas também publicadas no Diário Oficial da União.

> NR-29 – Segurança e saúde no trabalho portuário[1]
> NR-30 – Segurança e saúde no trabalho aquaviário[2]
> NR-31 – Segurança e saúde no trabalho na agricultura, pecuária, silvicultura, exploração florestal e arquicultura[3].
> NR-32 – Segurança e Saúde no Trabalho em Serviços de Saúde[4]
> NR-33 – Segurança e Saúde nos trabalhos em espaços confinados[5]
> NR-34 – Condições e meio ambiente de trabalho na indústria da construção e reparação naval[6]
> NR-35 – Segurança e saúde no trabalho em altura[7]
> NR-36 – Segurança e saúde no trabalho em empresas de abate e processamento de carnes e derivados[8]

[1] Redação dada pela Portaria nº 158, de 10-4-2006. *DOU* de 17-4-2006.
[2] Redação dada pela Portaria nº 34, de 4-12-2002. *DOU* de 9-12-2002.
[3] Redação dada pela Portaria nº 86, de 3-3-2005. *DOU* de 4-3-2005.
[4] Redação dada pela Portaria nº 485, de 11-11-2005. *DOU* de 16-11-2005.
[5] Redação dada pela Portaria nº 202, de 22-12-2006. *DOU* de 27-12-2006.
[6] Redação dada pela Portaria nº 200, de 20-1-2011. *DOU* de 21-1-2011.
[7] Redação dada pela Portaria nº 313, de 23-3-2012. *DOU* de 27-3-2012.
[8] Redação dada pela Portaria nº 555, de 18-4-2013. *DOU* de 19-4-2013

2.3 Função Controle de Riscos e Função Controle de Emergências

Ambas são funções auxiliares da Função Segurança, consequentemente uma abrange a outra. Quando é desenvolvido um sistema que faça o controle de emergências, este também está controlando os riscos.

A função de controle de riscos é uma função necessária que procura manter os riscos do trabalho abaixo das ameaças toleradas. A execução do controle de riscos é composta por várias ferramentas que a organização usa para realizar o planejamento de atividades, a operação das atividades e o modelo de controle de suas atividades.

Para a execução de um controle eficiente deve-se identificar os perigos e saber lidar com cada um deles de forma segregada, tendo uma ação proativa de correção que lide com qualquer perigo separadamente.

A Figura 2.1 mostra o fluxo do processo de gestão de riscos descrita no texto acima.

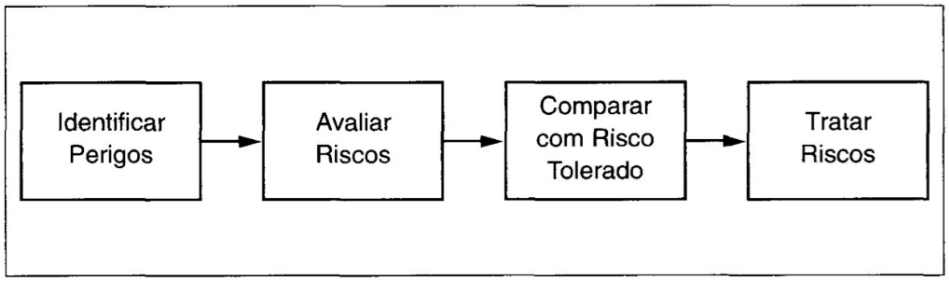


Figura 2.1 - Processo de gestão de riscos
Fonte: Segurança no Trabalho e Prevenção de Acidentes

Já a função de controle de emergências, só é exercida após que um risco previsto ou não, comece a mostrar-se como um evento real. Caso um evento real de emergência esteja ocorrendo, o ser humano apresenta grandes chances de cometer erros. Portanto, os funcionários que agem na função de controle de emergências devem estar devidamente treinados para agir.

A política de gestão de emergências é determinada por gestores de cada organização, não tolerando dúvidas em relação ao comportamento esperado dos responsáveis por emergências. As ações a serem tomadas devem ser ponderadas antecipadamente e dado o acontecimento, tais ações deverão ser executadas de modo automático. Para Cardella (2014, p. 80):

> I - Atuar prioritariamente no sentido de proteger e não colocar em risco a integridade das pessoas, inclusive a dos próprios componentes da organização para controle de emergências.
> II - Toda informação sobre anormalidades externas à organização, mas que possam estar relacionadas com suas atividades deve ser prontamente averiguado.
> III - Como forma de colaboração, a organização pode prestar apoio à comunidade em emergências não relacionadas com suas atividades, desde que isso não prejudique sua própria segurança.
> IV - As emergências com potencial para afetar áreas externas devem ser prontamente comunicadas aos órgãos públicos (órgãos ambientais, Defesa Civil).

A determinação de funções para cada pessoa que irá atuar em emergências é crucial para uma ação de brigada com sucesso. Tendo como ferramentas fundamentais: uma boa organização de procedimentos, definição de recursos, simulações de casos e treinamento de pessoal.

Vale ressaltar que a alta gestão da organização sempre deverá ter uma análise completa de todos os riscos diretos e riscos agregados que podem ocorrer no empreendimento. Realizando um diagnóstico de cada elemento que compõe um todo, identificando os perigos de forma dissolvida para obter uma inspeção mais detalhada.

2.4 Teoria das Falhas

Uma falha ocorre quando algum elemento do sistema não executa ou executa de forma errônea uma função previamente dada a ele. Está falha pode ser causada por erro humano ou por erro de um equipamento. Grande parte das falhas é de responsabilidade humana, pois se fosse feita uma análise mais profunda dos erros causados por equipamentos, podem-se atribuir, em sua maioria, à erros humanos no desenvolvimento destes equipamentos.

Ambos os tipos de falhas, podem se dar de cinco formas distintas: executar uma ação incompleta, executar a ação de forma errônea, executar uma ação que não deveria ter sido executada, podem executar ações prevista em ordens diferentes e por fim, podem executar ações previstas em um momento que não deveria executar. (CARDELLA, 1999)

A teoria das falhas se torna necessária quando um sistema tem que manter um nível mínimo de segurança para seus usuários presentes, fazendo uma inspeção completa de todos os eventos considerados incomuns que podem causar uma falha. “A análise completa consiste em identificar o modo e o tipo da falha, os agentes promotores e inibidores, a fase do ciclo de vida do componente ou o sistema em que a falha ocorreu, e a fase geradora.” (CARDELLA, 2014, p. 184)

Utilizando das técnicas da Teoria das Falhas, qualquer sistema tende a reduzir a quantidade de falhas à zero. Vale ressaltar que nenhum equipamento ou pessoa está sujeita a acertar 100% das vezes que realizar aquela ação, por mais que já tenha realizado inúmeras vezes de forma correta.

2.5 Linguagens de programação

Os itens a seguir apresentam as principais características das linguagens de programação necessárias ao desenvolvimento da solução aqui proposta. Cada subitem abordando um tema de forma mais superficial, que será mais aprofundado no capítulo 3, o qual é o capítulo do desenvolvimento.

2.5.1 Microsoft .NET Framework

Microsoft .NET Framework é uma plataforma de desenvolvimento de software que foi criada pela Microsoft em fevereiro de 2002. Mundialmente conhecida por possibilitar a criação de aplicativos em qualquer Sistema Operacional ou dispositivo, pois a plataforma provê a facilidade de poder desenvolver com várias linguagens de programação. O .NET é composto pelo Common Language Runtime (CLR) e o Framework Class Library (FCL).

O FCL é um conjunto de bibliotecas com funcionalidades comuns entre todas as linguagens definidas por Common Language Infrastructure (CLI). Essas bibliotecas de interfaces, classes e tipos de valores provêm acesso às funcionalidades do sistema.

O CLR é o componente da máquina virtual do .NET Framework que gerencia a execução dos programas, realizando a conversão de código em formato Common Intermediate Language (CIL) em instruções de máquina, para serem executadas por qualquer Central Processing Unit (CPU), como pode-se observar na Figura 2.2. Todas as linguagens de programação utilizadas no desenvolvimento de um único sistema .NET são compiladas em linguagem CIL, gerando um código único, chamado de “Bytecode”. Isto resulta na interoperabilidade, fazendo que todo o sistema funcione com uma única linguagem. Dessa forma, quando for realizar a execução do programa, ele é compilado novamente, agora pelo compilador Just In Time (JIT), onde o código de máquina assembly é gerado.

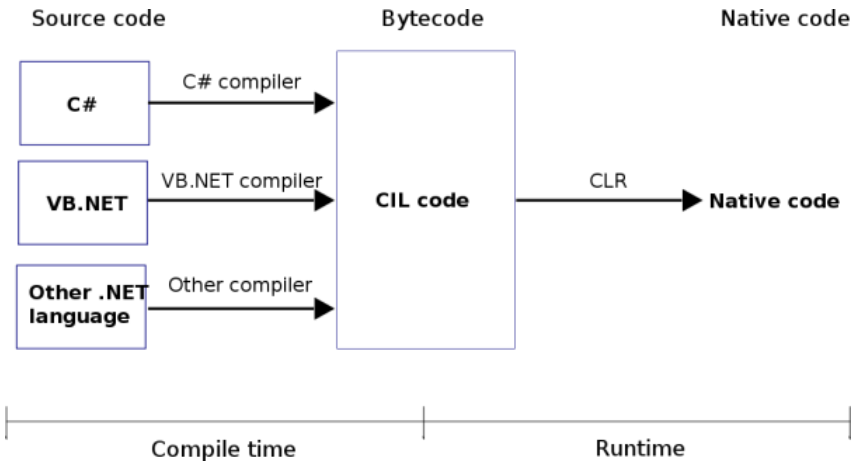


Figura 2.2 - Diagrama CLR
Fonte: (Storset, 2007)

2.5.2 Linguagem C Sharp

A linguagem de programação C Sharp (C#) foi desenvolvida pela Microsoft juntamente com a criação do .NET Framework em meados dos anos 2000. O C# é uma linguagem de programação orientada a objetos e foi desenvolvida para trabalhar na plataforma .NET. Sua sintaxe é muito similar a linguagem C, C++ e Java. Desenvolvedores familiarizados com essas tecnologias estarão hábios de programar em C# rapidamente. (Microsoft Developer Network, 2016)

O código fonte de C# é compilado em um Intermediate Language (IL), chamado de Common Intermediate Language, que está em conforme com as especificações da CLI.

Para diminuir o tempo gasto no desenvolvimento de códigos, o C# retira algumas preocupações de baixo nível dos programadores, como problemas com segurança, construções de bibliotecas e gerenciamento de memória. Fazendo com que o seu esforço seja gasto trabalhando em suas aplicações e não em regras lógicas. Uma das grandes funcionalidades do C# na plataforma .NET Framework é o Garbage Collector, em português livre ‘coletor de lixo’. Ele gerencia liberação e a alocação de memória para sua aplicação, ou seja, possui uma tecnologia que determina o melhor momento para realizar uma limpeza na memória com base nas alocações que estão sendo feitas na hora. O Garbage Collector é sempre iniciado pelo CLR.

Por ser uma linguagem orientada a objetos, a linguagem C# também é baseada em herança, polimorfismo e encapsulamento. Todos os métodos e variáveis são encapsuladas em definições de classes, por exemplo, o método *Main*. E uma regra básica das heranças de classe é que uma classe só pode herdar de uma classe pai. Por esse este e outros motivos facilitadores, o C# será utilizado como linguagem de programação para todo o desenvolvimento.

Ponto importante a ser comentado sobre essa linguagem é a forma de consulta Language Integrated Query (LINQ). Conforme a definição a seguir, feita pela Microsoft.

> O LINQ oferece um modelo consistente para trabalhar com dados em vários tipos de fontes de dados e formatos. Em uma consulta LINQ, você está sempre trabalhando com objetos. Você usa os mesmos padrões de codificação básicos para consultar e transformar dados em documentos XML, bancos de dados SQL, Datasets ADO.NET, coleções do .NET e qualquer outro formato para o qual uma consulta LINQ está disponível.

Todas as operações de consultas LINQ consistem em três ações distintas: Obter a fonte de dados, criar a consulta e executar a consulta. (Visual Studio, 2015)

2.5.3 Xamarin

A Xamarin é uma empresa de software criada em 2011, mas foi comprada pela Microsoft em 2016. Seu principal produto é o Xamarin Platform, plataforma de desenvolvimento baseada em linguagem C#, utilizada para escrever aplicativos para o iOS, Android e Windows Phone.

No desenvolvimento de um aplicativo com a plataforma Xamarin, a maioria dos aplicativos possui alguma exigência para salvar os dados no dispositivo localmente. A menos que a quantidade de dados seja pequena, isso geralmente requer um banco de dados e uma camada de dados no aplicativo para gerenciar o acesso de banco de dados. Grande parte das funcionalidades de acesso aos dados são iguais para o Android e o iOS, que suportam as APIs ADO.NET framework e SQLite-NET 3rd party library. Onde dois modelos são discutidos, o acesso básico aos dados e o acesso avançado aos dados. (Xamarin Data Access, 2016)

Uma das grandes vantagens da plataforma Xamarin é a aplicabilidade dos Shared Projects. É está funcionalidade que permite que um único código seja compartilhado por aplicações em Android, Windows ou iOS. Os projetos Portable Class Library (PCL), no português, Biblioteca de Classes Portátil que fazem a implementação dos Shared Projects. Este código é desenvolvido em C#, linguagem suportada pelo .NET Framework, depois ele pode ser compilado separadamente por cada uma das aplicações, como indicado na Figura 2.3.

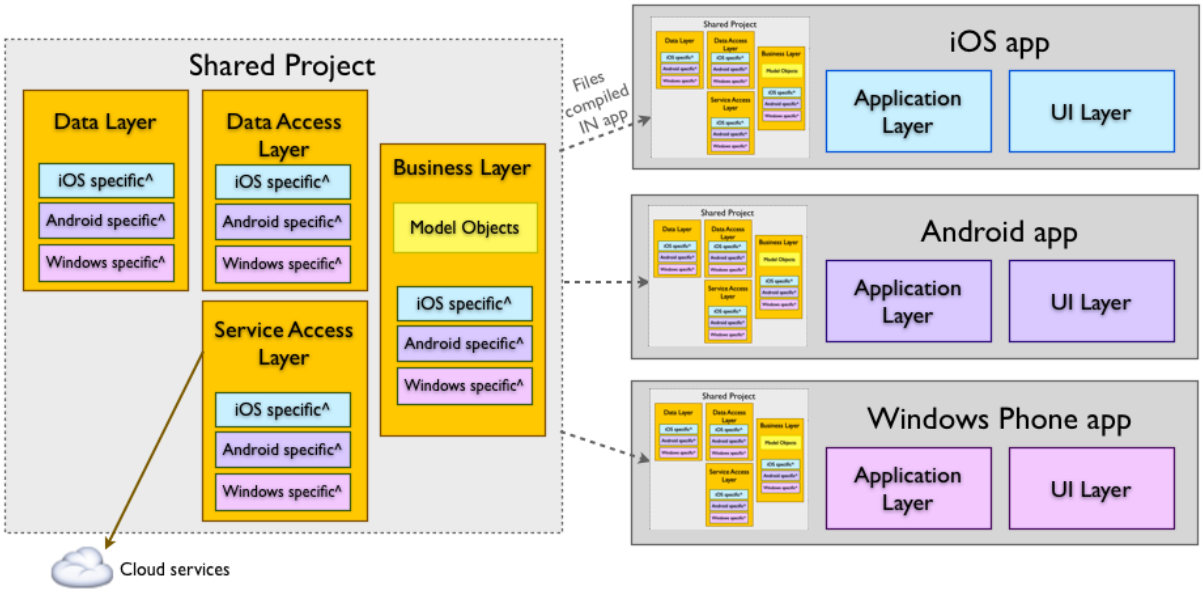


Figura 2.3 - Shared Projects Xamarin
Fonte: (Site Oficial Xamarin, 2016)

Apesar da plataforma compartilhar o mesmo código para iOS, Android e Windows, a implementação é diferente em cada uma das tecnologias, mesmo que o código seja escrito em C#. A compilação do código em C# ocorre dentro de cada um dos projetos de maneira diferente, como descrito em 2016 no site da Xamarin:

> iOS – C# is ahead-of-time (AOT) compiled to ARM assembly language. The .NET framework is included, with unused classes being stripped out during linking to reduce the application size.
>
> Android – C# is compiled to IL and packaged with MonoVM + JIT'ing. Unused classes in the framework are stripped out during linking. The application runs side-by-side with Java/ART (Android runtime) and interacts with the native types via JNI.

> Windows – C# is compiled to IL and executed by the built-in runtime, and does not require Xamarin tools. Designing Windows applications following Xamarin's guidance makes it simpler to re-use the code on iOS and Android.

O Integrated Development Environment (IDE) escolhido para desenvolver o sistema do trabalho foi o Xamarin Visual Studio para Windows. Para desenvolver em IOS, Windows e Android é necessário que seja ele. O Xamarin Studio for Mac OS X é capaz de desenvolver somente para IOS e Android, sendo que para isso é necessário um computador da marca Apple tendo seu sistema operacional nativo, ou seja, o Mac OS.

Sobre a User Interface (UI), é usada uma tecnologia proprietária da Xamarin, chamada de Xamarin.Forms. O qual possui funcionalidades semelhantes com o Share Projects, ou seja, é possível criar somente um layout da interface do usuário para o Windows Phone, iOS e Android.

2.6 Android

O Android é um Sistema Operacional para dispositivos móveis, como smartphones, tablets, televisões, câmeras, smartwatches, satélites da NASA, consoles de jogos, dispositivos residenciais, câmeras e muitos outros. Ele é baseado em Linux e possui seu código-fonte aberto e gratuito. Diferentemente do seu grande concorrente, o iOS, que possui o código fechado e só roda em dispositivos Apple. A Google Play Store é a loja virtual onde estão disponíveis todos os aplicativos para Android. Local onde se pode encontrar milhões de aplicativos, para realizar qualquer função desejada. Caso não possua lá um aplicativo para realizar o que deseja, então programe-o, pois, outras pessoas também devem estar precisando. (Android, 2016)

A linguagem de programação dos seus aplicativos nativos é em Java, linguagem que ocupa o segundo lugar em Junho de 2016 na lista de linguagens mais utilizadas no mundo. Perdendo apenas para a sua "irmã" JavaScript. Em terceiro lugar temos o PHP, em quarto o Python e em quinto C#. [1]

Alguns dados demonstram a superioridade da Android no mundo, no segundo semestre de 2015, a Android possuía 82,8% do mercado global de Sistemas Operacionais para Smartphones, seguida do iOS com 13,9%, Windows Phone com 2,6%, BlackBerry OS com 0,3% e todo os outros com 0,4%.[2] Como se pode observar na Figura 2.4, dados

apresentados pelo IDC em agosto de 2015, a Android sempre teve a grande fatia do mercado global e o esperado para 2016 é que esteja maior.

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Figura 2.4 - Mercado de SO de Smartphones
Fonte: (IDC, 2º Trimestre de 2015)

[1] http://redmonk.com/sogrady/category/programming-languages (Data de acesso: 15/09/2016)

[2] http://www.idc.com/prodserv/smartphone-os-market-share.jsp. (Data de acesso: 15/09/2016)

A escolha de desenvolver para Android o aplicativo do sistema de vistorias se explica com esses números acima. Mas o motivo de se desenvolver o aplicativo em C# é muito simples, mesmo que os números demonstrem o contrário, desenvolver o aplicativo em C# e utilizando o compilador do Xamarin for Visual Studio, em um futuro breve, a escrita do aplicativo para o iOS está basicamente pronta. Lembrando que ambas as linguagens, Java e C#, são orientadas ao objeto.

Todos os pré-requisitos do Android, Java e Xamarin Tool necessários para a utilização do Xamarin Studio fazem parte do seu instalador. Basta a instalação do software e o computador já está com as SDKs do Java e Android. O que proveem a compilação, emulação e outras funcionalidades para criar, desenvolver e testar. O Xamarin.Android já provê o suporte para o Android 7.0.(XAMARIN, 2016)

Para o acesso as informações no Android, deve-se ter alguns entendimentos breves, conforme descrição do livro Android 6 para programadores, 2016.

> As informações são armazenadas em um banco de dados SQLite. De acordo com o site www.sqlite.org, SQLite é um dos mecanismos de banco de dados mais ampla- mente distribuídos do mundo. Você vai usar uma subclasse de SQLite-OpenHelper (pacote android.database.sqlite) para simplificar a criação do banco de dados e obter um objeto SQLiteDatabase (pacote android.database.sqlite) para manipular o conteúdo do banco de dados. As consultas de banco de dados são realizadas com a linguagem SQL

> (Structured Query Language). Os resultados da consulta são gerenciados por meio de um Cursor (pacote android.database).

Outra forma de acesso aos dados é o acesso assíncrono, operações realizadas durante as threads que não bloqueoa a execução do programa até que esse acesso termine. Esta técnica é usada para manter a velocidade de resposta do aplicativo bem rápida.

A biblioteca ZXing.Net.Mobile será utilizada no aplicativo para realizar a leitura de código de barras ou QR Code. É uma biblioteca com o código aberto e baseada em C# e .NET, tendo total compatibilidade com o Xamarin.Android, Xamarin.iOS, Windows Phone (Silverlight) and Windows Universal. (Xamarin, 2016)

2.7 ASP.NET Web API

ASP.NET Web API é um framework que torna mais fácil construir serviços HTTP que chegam a uma ampla gama de clientes, incluindo navegadores e dispositivos móveis. ASP.NET Web API é uma plataforma ideal para a criação de aplicativos RESTful no .NET Framework. Realiza uma comunicação através da troca de mensagens com requisições e respostas HTTP nos formatos XML ou JSON. (Microsoft ASP.NET, 2016)

A tecnologia conhecida como RESTful API, é uma interface de programação de aplicação que faz requisições HTTP para realizar com o dado as funções GET, DELETE, POST, PUT.

A Figura 2.5 mostra o cenário mais comum nos aplicativos instalados em smartphones para as requisições de dados. Quando o usuário digita seu login e senha no aplicativo solicitando acesso, esta requisição é enviada para o servidor de autenticação verificar se estão válidas as informações de acesso, caso esteja, retorna para o cliente um Token. Para o aplicativo no smartphone realizar a consulta de qualquer informação ao banco de dados ou no servidor central da aplicação, podendo ser o Instagram, Facebook ou qualquer outro, ele realiza essa requisição enviando o token e o servidor central da aplicação retorna o dado requisitado.

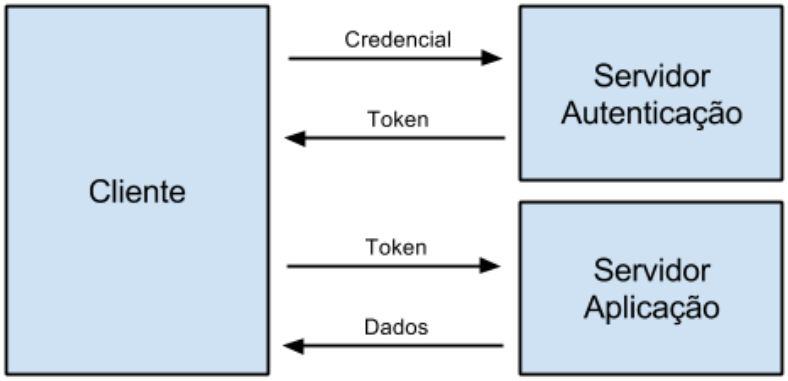


Figura 2.5 - Autenticação Web API
Fonte: (C# Brasil, 2015)

No trabalho será utilizado para desenvolver códigos de Web API dentro do Visual Studio. Utilizando templates pré-configurados para facilitar o desenvolvimento, apenas realizando alguns ajustes e atribuindo os parâmetros corretos.

2.8 Entity Framework

Entity Framework (EF) é um conjunto de tecnologias em ADO.NET que suportam o desenvolvimento de aplicativos de software orientadas a dados. É uma ferramenta da Microsoft que faz o mapeamento de objetos-relacionais para permitir aos desenvolvedores .NET que trabalhem com dados relacionais usando objetos específicos de domínio. Ele elimina a necessidade dos desenvolvedores precisarem escrever a maioria do código que faz o acesso a dados. (MSDN, 2016)

O EF será utilizado no trabalho para ter um nível mais alto de abstração de dados, sem ter que se preocupar com as tabelas de banco de dados todas as horas, mas sim com objetos.

2.9 Microsoft SQL Server

O SQL Server é um Sistema Gerenciador de Banco de Dados (SGBD) relacional desenvolvido pela Microsoft em 1988 em parceria com a Sybase. O SQL Server realiza inúmeras funções, mas sua função primordial de banco de dados é armazenar e restaurar dados estruturados, além de possuir ferramentas, como, Business Intelligence (BI), Online Transaction Processing (OLTP) e análise avançada dos dados operacionais. (Microsoft, 2016)

Estudo realizado em outubro de 2015 pela empresa de consultoria Gartner, demonstra a superioridade do SQL Server em relação a outras ferramentas de banco de dados, o que justifica a escolha no desenvolvimento do trabalho, como descrito no trecho abaixo e mostrado na Figura 2.6.

> A Gartner reconheceu novamente a Microsoft como Líder no Quadrante Mágico de 2015 para Sistemas de gerenciamento de bancos de dados operacionais, posicionando-a mais à direita no eixo que representa a abrangência de visão e mais ao alto no eixo referente à capacidade de atuação.

Figura 2.6 - Quadrante Mágico de Sistemas de gerenciamento de banco de dados operacionais
Fonte: (Gartner, 2015)

O Banco de Dados SQL de Nuvem do Microsoft Azure é o serviço de banco de dados de nuvem do desenvolvedor. (Microsoft Azure, 2016). Este modelo de banco de dados será utilizado no trabalho haja vista suas características de total integração com o SQL Server. Além disso, conforme o site da Microsoft em 2016, é confiável por estar na nuvem do mesmo fabricante, apresenta total compatibilidade com o Visual Studio, possui tecnologia de multilocatário que permite um banco de dados atender N clientes remotos ao mesmo tempo, ter fácil gerenciamento e escalabilidade.

2.10 Microsoft Azure

O Microsoft Azure é uma coleção de serviços de nuvem integrados, como máquinas virtuais, IaaS, PaaS, linguagens de programação, serviços de dispositivos móveis, armazenamento de dados, serviço de análises, banco de dados, entre outros. Contam com mais de 600 serviços disponíveis em sua plataforma. (Microsoft Azure, 2016)

O Azure é executado em centros de processamentos de dados em 30 regiões pelo mundo, sendo todos gerenciados pela Microsoft. Com um Data Center sempre perto do cliente, a Azure permite entregar um maior desempenho ao acesso. O Data Center do Brasil fica no estado de São Paulo, estando disponível 24 horas por dia, 7 dias da semana, 365 dias no ano.

A confiança na Azure atingiu um número superior a 66% das empresas listadas na revista Fortune 500 em 2015. Isto mostra quão eficiente e confiável são os serviços prestados pela plataforma. Exemplos de clientes são: Heineken, 3M e Real Madrid.

O seu custo é relativamente caro, mas tem a grande vantagem de se poder adiquirir apenas o que se deseja utilizar. No desenvolvimento do trabalho, foi feita a inscrição e utilização da plataforma para um início grátis. Como bônus, o crédito de R$ 750,00 reais para utilizar com o que for necessário, no caso, utilização de máquinas virtuais, banco de dados SQL, autenticação do Azure Active Directory, hospedar o aplicativo web e realizar monitoramento de desempenho. Acredita-se, com isto, ser suficiente até o fim da pesquisa e desenvolvimento do sistema. Um dos recursos de aplicativos móveis importante para o desenvolvimento móvel em nuvem é o acesso aos dados descrito na documentação do Azure, em 2016.

> “Os Aplicativos Móveis do Azure fornecem uma fonte de dados OData v3 compatível com dispositivos móveis vinculada ao SQL Azure ou a um SQL Server local. Esse serviço pode ser baseado no Entity Framework, permitindo que você integre facilmente com outros provedores de dados NoSQL e SQL [...]”

O Azure possui SDKs que estabelecem um tipo de implementação padrão para armazenamento local, baseada em SQLite.

O código presente no aplicativo controla em qual momento esta sincronização será realizada para que sejam salvas as alterações que foram realizadas localmente. Os dados só serão enviados ao back-end do Azure após ser realizada uma chamada para alterações locais de PUSH. Por outro lado, quando é realizada uma chamada para dados de PULL, o repositório local é preenchido com novos dados. Segue definição do site oficial da Azure, em outubro de 2016.

> “Push: push é uma operação no contexto de sincronização e envia todas as alterações de CUD desde o último envio por push. Observe que não é possível enviar apenas alterações de uma tabela individual, pois as operações poderiam ser enviadas fora da ordem. O envio por push executa uma série de chamadas REST ao back-end do seu aplicativo móvel do Azure, que por sua vez, modificará o banco de dados do servidor.
>
> Pull: o pull é executado por tabela e pode ser personalizado com uma consulta para recuperar apenas um subconjunto dos dados do servidor. Os SDKs de cliente móvel do Azure inserem então os dados resultantes no armazenamento local.”

2.11 QR Code

O Quick Response Code (QR Code) é uma imagem bidirecional composta por quadrados pretos que formam um código de barras 2D, tal código pode ser interpretada por smartphones ou tablets que possuam câmera e um software scanner de QR codes. O objetivo desta imagem é ser um caminho para informações. Para a substituição de longos códigos de barras e vários números que compõem um serial number, uma divisão do Grupo Toyota desenvolveu em 1994 este facilitador. (Vieria, Liliana. 2013)

A codificação realizada na geração da imagem proporciona que sejam armazenadas a quantidade de 7.089 caracteres numéricos, 4.296 alfa-numéricos, 2.953 binários (8 bits) e 1.817 Kanji/Kana (alfabeto japonês).

Sobre a taxa de reparação de informações e a correçãos dos erros dos códigos, é sua maior vantagem, fica em torno de 7% (Nível L) a 30% (Nível H) de melhoria. Um dos principais fatores de sua facilidade é a leitura não necessitar de um scanner com infravermelho e poder ser feita em qualquer posição. Como pode-se observar na Figura 2.7, são adicionados dados tanto na vertical, quanto na horizontal, diferentemente do código de barras convencional que armazena dados somente na horizontal. Para a identificação da posição da imagem, são usados 3 quadrados maiores. Hoje exitem alguns padrões de codificação, o mais atual foi criado em 1 de setembro de 2006, o ISO/IEC 18004:2006.

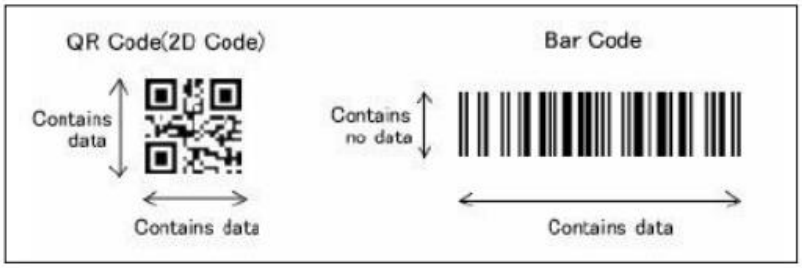


Figura 2.7 – QR Code
Fonte: Susono & Shimomura, 2006

Por provê uma grande quantidade de informação, ele é utilizado para inúmeras funções hoje em dia, como direcionar a uma URL, mostrar o cartão de visita de uma empresa, ser direcionado para outros aplicativos, assistir o trailer de um filme, abrir a página da rede social de uma pessoa, pode ser o controle de um inventário de uma fábrica e outros. A criação desta imagem pode ser feita a partir de inúmeros sites, como <http://br.qr-code-generator.com/> e <http://qrcode.kaywa.com/>.

CAPÍTULO 3 – DESENVOLVIMENTO DO PROTÓTIPO

3.1 Visão Geral do Projeto

A topologia adotada para a implantação do projeto está mostrada na Figura 3.1. Como pode-se observar, o protótipo é composto por 2 tecnologias, sendo elas, um smartphone e um back-end do sistema que será hospedado em nuvem. Essas 2 tecnologias podem ser subdivididas em 7 componentes principais: o dispositivo móvel, a sincronização offline, o tipo de comunicação sem fio, a nuvem Azure, o serviço de aplicativo, o gerenciamento de API e o banco de dados SQL.

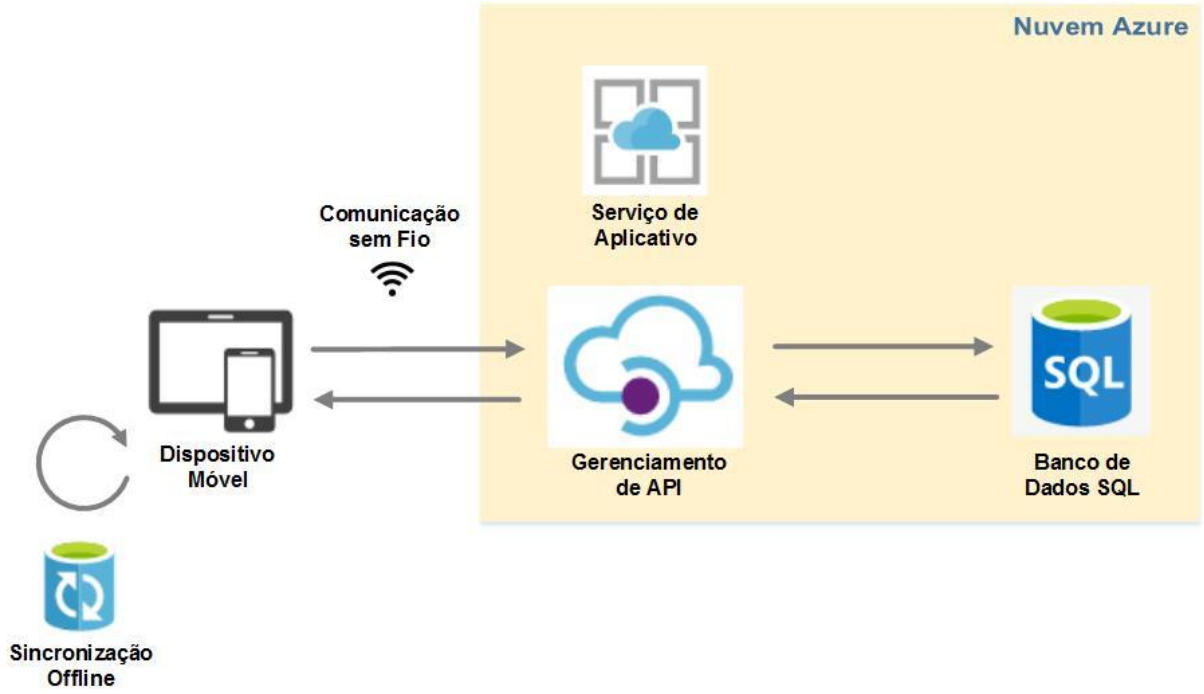


Figura 3.1 – Topologia do projeto
Fonte: Autor

Neste capítulo será abordado o desenvolvimento do protótipo proposto, contemplando como foi realizado o levantamento dos requisitos de vistorias, todas as tecnologias utilizadas e as 3 etapas de desenvolvimento. As 5 etapas serão esclarecidas em cada detalhe sendo divididas em subitens.

3.2 Requisitos levantados para realizar a vistoria

Os requisitos foram estabelecidos como forma de demonstração de locais e o que deve ser vistoriado em um Shopping de grande porte. A metodologia utilizada para determinar quais seriam os requisitos a serem vistoriados, foi a entrevista com dois gerentes do shopping Iguatemi Brasília: o gerente de segurança e logística e o gerente da manutenção.

Esta vistoria visa verificar as condições necessárias para a proteção contra incêndio e demais necessidades de segurança do local e garantir o correto funcionamento dos equipamentos de combate a incêndio, bem como a segurança das pessoas, atendendo o prescrito na legislação vigente.

Podemos observar nas Tabelas 3.1, 3.2 e 3.3, o quão detalhado devem ser as vistorias de segurança de um local onde existe uma circulação de pessoas. A preocupação em manter o local seguro vai desde os corredores do empreendimento, até o interior de todos as lojas. Sendo responsabilidade de todos os proprietários cuidar e zelar pela segurança de cada local, tornando um shopping um lugar que seja proveitoso para pessoas de todos os estilos e idades.

As vistorias são separadas por tipo de localidade, sendo divididas em 3 grupos. A primeira a ser abordada é a vistoria nas operações âncoras e satélites, como pode-se observar na Tabela 3.1, que prevê vistorias em lojas grandes de departamento. Também incluindo vistorias em lojas menores, denominadas lojas satélites.

Tabela 3-1 – Vistorias nas operações âncoras e satélites

VISTORIAS NAS OPERAÇÕES ÂNCORAS E SATÉLITES:
Ao realizar a vistoria nas operações deverão observar os seguintes itens:
Inspeção dos Extintores:
Carga;
Vencimento;
Lacre;
Acessibilidade;
Sinalização;
Distribuição e cobertura do local;
Adequação da utilização pelo tipo de risco.

Fonte: Shopping Center em Brasília

Tabela 3-2 – Vistorias nas operações âncoras e satélites (continuação)

Hidrantes:
Esguicho;
Chaves;
Sinalização;
Quantidade e tamanho de mangueiras.
Área de mezanino:
Excesso de mercadorias;
Armazenamento;
As mercadorias deverão estar a 30 cm da luminária;
Arranjo fisíco;
Mercadorias fora do alcance da proteção de sprinklers;
Condições da estrutura do mesmo;
Adequação da reder de sprinklers ao layout do mezanino;
Corredores entre as prateleiras desobstruídos.
Instalações elétricas:
Quadro de energia;
Quadro sinalizado;
Quadro desobstruído;
Existência de contra tampa evitando contatos acidentais nos barramentos;
Fiação exposta;
Sobrecarga nas tomadas (utilização de “benjamim”).
Locais de segurança:
Rota de fuga livre;
Porta de saída de emergência sinalizada e livre;
Escada de acesso ao mezanino em condições adequadas para deslocamento em caso de sinistro.
Iluminação de emergência.

Fonte: Shopping Center em Brasília

Outro grupo de vistoria, considerado de risco maior, é o grupo das vistorias nas operações de alimentação. O qual é classificado como mais importante por possuir equipamentos com maior fator de incêndio, citado na Tabela 3.2, como fontes de calor, materias combustíveis, coifas e dutos. Lembrando que o cuidado com a rede elétrica deve ser diferenciado, pois com a instalação de equipamentos que provê um aquecimento (fontes de calor), a distância e o local que devem estar as instalações elétricas devem sem diferentes de uma loja satélite por exemplo. Todo cuidado deve ser tomado e as vistorias nesses ambientes devem ser de maior frequência, por mais que todos os funcionários de uma loja de

alimentação tomem todo o cuidado necessário, erros humanos podem ocorrer e acidentes acontecerão. Por isso, todos devem estar sempre atentos e ter responsabilidade de vistoriar o próprio trabalho e o trabalho de seu colega no dia-a-dia. O papel do funcionário contratado pelo condomínio para vistoriar tais ambientes, também é de auxiliar com melhores práticas de ações a serem tomadas em caso de irregularidades encontradas.

Tabela 3-3 - Vistorias nas operações de alimentação

VISTORIAS NAS OPERAÇÕES DE ALIMENTAÇÃO:
Ao realizar a vistoria nas operações deverão observar os seguintes itens:
Inspeção dos Extintores:
Carga;
Vencimento;
Lacre;
Acessibilidade;
Sinalização;
Distribuição e cobertura do local;
Adequação da utilização pelo tipo de risco.
Instalações elétricas:
Todas as caixas de passagem devem estar fechadas;
Todos os reatores deverão ser fixados sobre materiais incombustíveis;
O quadro elétrico deverá possuir contra-tampa visando evitar contatos acidentais nos barramentos;
As tomadas elétricas devem ser instaladas fora do fluxo gasoso proveniente dos equipamentos de cocção;
Devem estar totalmente protegidas por eletrodutos.
Fontes de calor:
Fogões, fornos, churrasqueiras e similares devem estar no interior de compartimentos com piso, paredes e cobertura incombustíveis.
Saídas:
As saídas devem ser mantidas livres e desimpedidas, de acesso facilitado, de forma que os ocupantes não tenham dificuldade em abandonar a edificação em caso de sinistro.
Materiais combustíveis próximos as coifas:
Óleos;
Papeis e panos;
Armazenamento de caixas com produtos diversos.

Fonte: Shopping Center em Brasília

Tabela 3-4 - Vistorias nas operações de alimentação (continuação)

Coifas e dutos:
Limpeza (incrustação, damper, etc);
A rede de dutos de exaustão em nenhum trecho pode passar em compartimentos com medidores ou botijões de gás combustível, em instalações fixas;
Dampers corta-fogo com acionamento eletromecânico e devem atender aos seguintes requisitos:
1 - Tempo de resposta ao fechamento dever ser imediato;
2 - Estanqueidade a líquidos, chamas e fumaças;
3 - Temperatura da superfície na face não exposta à chama inferior à temperatura de fulgor de óleos e gorduras;
4 - Sistema de proteção contra incêndio internamente.
Rede de Sprinklers
Todos produtos/materias devem ser estocado abaixo da linha dos bicos de sprinklers;
Os produtos / materiais devem estar afastados dos bicos de sprinklers no mínimo 30 cm;
A rede de sprinklers deverá estar adequada ao lay-out do mezanino (prateleiras);
É proibido a utilização da rede sprinklers para armazenar roupas (como cabides, etc).

Fonte: Shopping Center em Brasília

Por fim e não menos importante, está o grupo 3, as vistorias em depósitos de uso privativo e administração. Tal vistoria é de suma importância para manter organizado o local onde não é visto pelo público. Por motivo de não ter que ser um local bonito, digamos, não é a vitrine e não é um local que faça um retorno direto de investimento, costuma estar sempre muito cheio e mal organizado. Em função disso, a vistoria passa a ser obrigatória para todos, tanto para depósitos de lojas ou restaurantes, como depósitos da própria administração do shopping.

Tabela 3-5 – Vistorias em depósitos de uso privativo e administração

VISTORIAS EM DEPÓSITOS DE USO PRIVATIVO E ADMINISTRAÇÃO:
Ao realizar a vistoria nos depósitos deverão observar os seguintes itens:
Inspeção dos Extintores:
Carga;
Vencimento;
Lacre;
Acessibilidade;
Sinalização;
Distribuição e cobertura do local;
Adequação da utilização pelo tipo de risco.
Hidrantes:
Esguicho;
Chaves;
Sinalização;
Quantidade e tamanho de mangueiras.
Condições gerais:
Excesso de mercadorias;
Armazenamento;
As mercadorias deverão estar a 30 cm das luminárias;
Arranjo fisíco;
Mercadorias fora do alcance da proteção de sprinklers;
Corredores entre as prateleiras desobstruídos.
Instalações elétricas:
Quadro de energia;
Quadro sinalizado;
Quadro obstruido;
Existência de contra tampa evitando contatos acidentais nos barramentos
Fiação exposta;
Sobrecarga nas tomadas (utilização de “benjamim”);

Fonte: Shopping Center em Brasília

Como forma de demonstração no protótipo, será feita uma breve vistoria apenas de extintores e hidrantes, como pode-se observar na Tabela 3.6, com a criação de QR Codes para ambos. Dessa forma será possível visualizar a execução e funcionamento de todo o sistema e como se aplica a todos os itens. Desde a autenticação de usuário, a criação de QR Codes, até a saída do relatório.

Tabela 3-6 – Inspeção demonstrativa

INSPEÇÃO DEMONSTRATIVA
Inspeção dos Extintores:
Carga;
Vencimento;
Lacre;
Acessibilidade;
Sinalização;
Distribuição e cobertura do local;
Adequação da utilização pelo tipo de risco.
Inspeção de Hidrantes:
Esguicho;
Chaves;
Sinalização;
Quantidade e tamanho de mangueiras.

Fonte: Autor

3.3 Primeira Etapa: Alocar recursos na nuvem Azure e configuração do Visual Studio

A forma de entregar uma solução de infraestrutura de dados varia bastante de acordo com a demanda de cada operação. Há clientes que hospedam seus serviços em nuvem Pública, outros que implementam sua nuvem privada e a maior porção possui seu próprio Data Center. A diversidade de designs demonstra como a infraestrutura em particular de cada empresa tem seus desafios para obter suficientes ganhos de escalabilidade. Balancear a relação custo x benefício é a proposta ideal que visa o melhor Retorno do Investimento (ROI).

A ideia de possuir uma nuvem pública é inovadora e traz facilidades na gestão, na segurança e na disponibilidade, variando proporcionalmente o custo da solução. Quanto maior a sua necessidade, maior será o investimento. Se almejado o mesmo nível de tecnologia (funcionalidades) e disponibilidade que se pode conseguir em um ambiente de Data Center local, o custo tende a ser superior em relação a este.

O investimento na aquisição de um Data Center local se justifica há médio e longo prazo. Com os produtos líderes de mercado a gestão é cada vez mais simplificada, otimizando de forma relevante os gastos operacionais e facilitando os processos de automatização de

serviço. Dessa forma, seu custo operacional é menor do que o valor pago mensalmente ao serviço de nuvem pública.

O uso da nuvem pública é dependente de conectividade com o ambiente onde se hospeda seus serviços. Mesmo que a empresa possua uma redundância de link, eles estão vulneráveis a problemas como o baixo desempenho ou indisponibilidade. Essa dependência não ocorre quando a empresa possui uma infraestrutura local, todos os serviços internos se mantêm operantes e com pouco ou nenhum impacto em caso de perda de comunicação com a rede externa (link de internet).

Quando aborda-se o custo para contratação de nuvem pública, embora em um primeiro momento seja muito semelhante à aquisição de uma infraestrutura própria, ao final do contrato o cliente é surpreendido com a necessidade de uma nova contratação custando 100% do valor anterior de acordo com a Microsoft. Por outro lado, manter o seu ambiente já adquirido uma vez terá o custo somente de renovação do contrato de suporte e garantia.

O custo para se manter o ambiente Azure está relacionado abaixo na Tabela 3.7. Como pode-se observar abaixo, foi realizada uma cotação de preço no próprio portal da Azure, no qual é possível escolher quais serviços você deseja adquirir. O método de cobrança do Azure é pelo mês de utilização.

Tabela 3-7 - Custo serviços Azure

Serviço	Descrição	Custo	Período/Utilização
BD SQL Server 2016	5 DTUs, 2GB de Armazenamento	R$ 22,60	Total de 744 horas (1 mês)
Máquina Virtual	1 core, 1.75GB de RAM, HD de 40GB	R$ 178,56	Total de 744 horas (1 mês)
DNS	1 Zona DNS, 1Mi de consultas DNS	R$ 3,38	Total de 744 horas (1 mês)
TOTAL		R$ 204,54 mensalmente	

Fonte: Autor realizou uma cotação no próprio portal da Azure, novembro de 2016

Os Database Throughput Units (DTUs) são Unidades de transação do banco de dados. De acordo com a Azure em 2016, uma DTU é uma medida que junta CPU, memória e dados de E/S, e E/S de log de transações, em uma proporção determinada por uma carga de trabalho OLTP de parâmetro projetada para ser igual às cargas de trabalho OLTP reais.

Admitindo baixo crescimento na base de dados do BD e que essa solução suportaria todo o ambiente por 1 ano. O custo ao fim desse período de 12 meses seria de R$ 2.454,48. Já com a aquisição de toda a solução própria, o custo do software SQL Server é grátis e um

Desktop que suportasse o BD e tivesse garantia por 1 ano é de R$ 1.889,00. Estas informações constam na Tabela 3.8 a seguir:

Tabela 3-8 - Custo de aquisição de hardware e software

Equipamento	Descrição	Custo	Período/Utilização
BD SQL Server 2016 Express	Até 10GB de Armazenamento, 1 core físico, 1GB de RAM	Grátis	Eterno
Desktop	Lenovo 63, 1 core, 4GB de RAM, HD de 500GB	R$ 1.889,00	Eterno, mas com 1 ano de garantia.
TOTAL		Única aquisição de 1.889,00	

Fonte: Cotação de preço realizada no próprio site da SQL e Lenovo

Considerando todos os pontos abordados anteriormente, em relação a comercialização do sistema em um futuro próximo, adquirir um mini Data Center próprio com um servidor e o software do BD, que faça o papel da Azure neste sistema é a melhor opção da atual necessidade. Visando manter o processamento de aplicações e os dados armazenados compartilhados internamente, com segurança, alto desempenho, alta disponibilidade e excelente custo-benefício.

A escolha da alocação de recursos na nuvem foi feita por questões de não possuir um servidor para a resolução do trabalho. Em contrapartida, a Azure oferecer um pacote de 750 créditos para iniciantes e goza de compatibilidade com todos as tecnologias utilizadas. O trabalho está sendo desenvolvido para utilizar suas tecnologias de processamento e armazenamento em nuvem. A data de criação do Banco de Dados SQL Server na Versão 12 foi no dia 8/10/2016.

A Azure é responsável pelo "*back-end*" (sistema responsável pelas regras de negócio e webservice) de todo o aplicativo, a sua principal função é manter sempre online o banco de dados, que é a alma do sistema. Para isso é necessário que ele possua o processamento do Serviço de Aplicativo e o armazenamento do Banco de Dados SQL. Lá são gerenciados os acessos do aplicativo pelas credenciais permitidas, a forma que é transmitida o dado, as APIs que tratam o dado para ser armazenado no BD e o banco de dados SQL.

O Serviço de Aplicativo é um Aplicativo Web que está instalado no Azure. É responsável por manter a comunicação das instâncias do aplicativo que está armazenado no Azure com o aplicativo que está em funcionamento no smartphone. No Serviço de Aplicativo

que são criadas as APIs responsáveis em popular as informações de forma correta e eficiente no BD da solução.

Para ter uma conta Azure, basta entrar nesse link <https://azure.microsoft.com/pt-br/free/> e seguir o passo a passo. É bastante intuitivo, apenas se atentar a forma que será cobrado após o vencimento da assinatura gratuita.

Para baixar o Visual Studio, basta acessar o site oficial do fabricante neste link <https://code.visualstudio.com/download> e seguir o passo a passo. Logo após a instalação de todo o software, que é bastante grande e demorar muito tempo para ser instalado, será realizada a configuração das suas credenciais. No canto superior direito da página inicial, selecione a opção “Sign in”. Entre com as credenciais cadastradas no site oficial da Azure, dessa forma, um único cliente terá o mesmo acesso a todos os portais da Microsoft.

A instalação do software Xamarin é dentro do próprio Visual Studio. Na tela principal do Visual Studio, na aba “Tools”, selecione a opção “Extensions and Updates”. Serão mostradas todas as extensões e atualizações que podem ser realizadas no Visual Studio. Deverá ser utilizado o campo de pesquisa para encontrar o software “Get Xamarin”, selecione a opção “Install” e após alguns minutos o software estará pronto.

3.4 Segunda etapa: Desenvolvimento e Implementação do BD SQL Server e do SSMS

3.4.1 Estruturação do Banco de dados

Como já citado, o Banco de Dados (BD) escolhido para o desenvolvimento do protótipo foi o SQL Server, que tem total compatibilidade com o Visual Studio, a hospedagem no Azure, o DBDesigner e a criação com o SQL Server Management Studio.

O DBDesigner é um sistema de construção visual de BD, que provê em um único ambiente capacidades de desenvolvimento, modelagem e criação. Este ambiente combina funcionalidades profissionais com uma simples e limpa interface para o usuário, oferecendo um método extremamente eficiente para conceber seus objetivos do BD. (FabFORCE, 2016)

A versão utilizado para modelar o BD do sistema foi o DBDesiner 4. O software é de download gratuito na página do seu fabricante, o FabFORCE. O link onde ele está disponível

é <http://fabforce.net/downloads.php>, o tamanho do executável para Windows é de apenas 5,6 MB. Os pré-requisitos de sistema são atendidos com facilidade, o software é relativamente leve e não utiliza de muita memória e processamento.

Na Tabela 3.9 é mostrada o tipo de cada uma das relações feitas entre as tabelas do BD, a grande maioria das relações é do tipo “1:n”. Isto significa que uma linha na tabela de origem corresponde várias linhas na tabela de destino. Somente a Rel_09 possui a relação do tipo “1:1”, que significa que uma linha na tabela de origem corresponde a uma linha na tabela de destino.

Tabela 3-9 - Relação entre tabelas do BD

Relação	Tipo	Tabela Origem	Tabela Destino
Rel_01	1:n	Usuario	Usuario_has_GrupoUsuario
Rel_02	1:n	GrupoUsuario	Usuario_has_GrupoUsuario
Rel_03	1:n	GrupoUsuario	Verificacao
Rel_04	1:n	TipoVerificacao	Verificacao
Rel_05	1:n	Produto	Verificacao
Rel_06	1:n	Setor	Produto
Rel_07	1:n	Verificacao	ProblemaVerificacao
Rel_08	1:n	Verificacao	HistoricoVerificacao
Rel_09	1:1	HistoricoVerificacao	Usuario

Fonte: Autor

A modelagem do BD no DBDesigner foi realizado como pode-se observar na Figura 3.2. Foram criadas 9 tabelas, sendo elas: Usuario, Usuario_has_GrupoUsuario, GrupoUsuario, Verificacao, TipoVerificacao, Produto, Setor, ProblemaVerificacao e HistoricoVerificacao.

Serão descritas todas as tabelas do banco de dados como forma de explicação da sua estrutura. Como mostrado na Figura 3.2, a tabela Usuario possui uma chave primária, chamada de idUsuario, representando cada usuario com um número, o tipo de dado é “integer”, ou seja, um número inteiro. Possui as seguintes linhas:

- nmUsuario - representando o nome do usuário, o tipo do dado é “varchar”, ou seja, dados compostos.
- coSenha - representando a senha que possuirá cada usuário, o tipo do dado é “varchar”.
- icAdministrador - representando se o usuário tem permissões de administrador, o tipo do dado é “bool”, ou seja, saída ‘sim’ ou ‘não’.

- edEmail - representando o endereço de e-mail cada usuário, o tipo do dado é “varchar”.

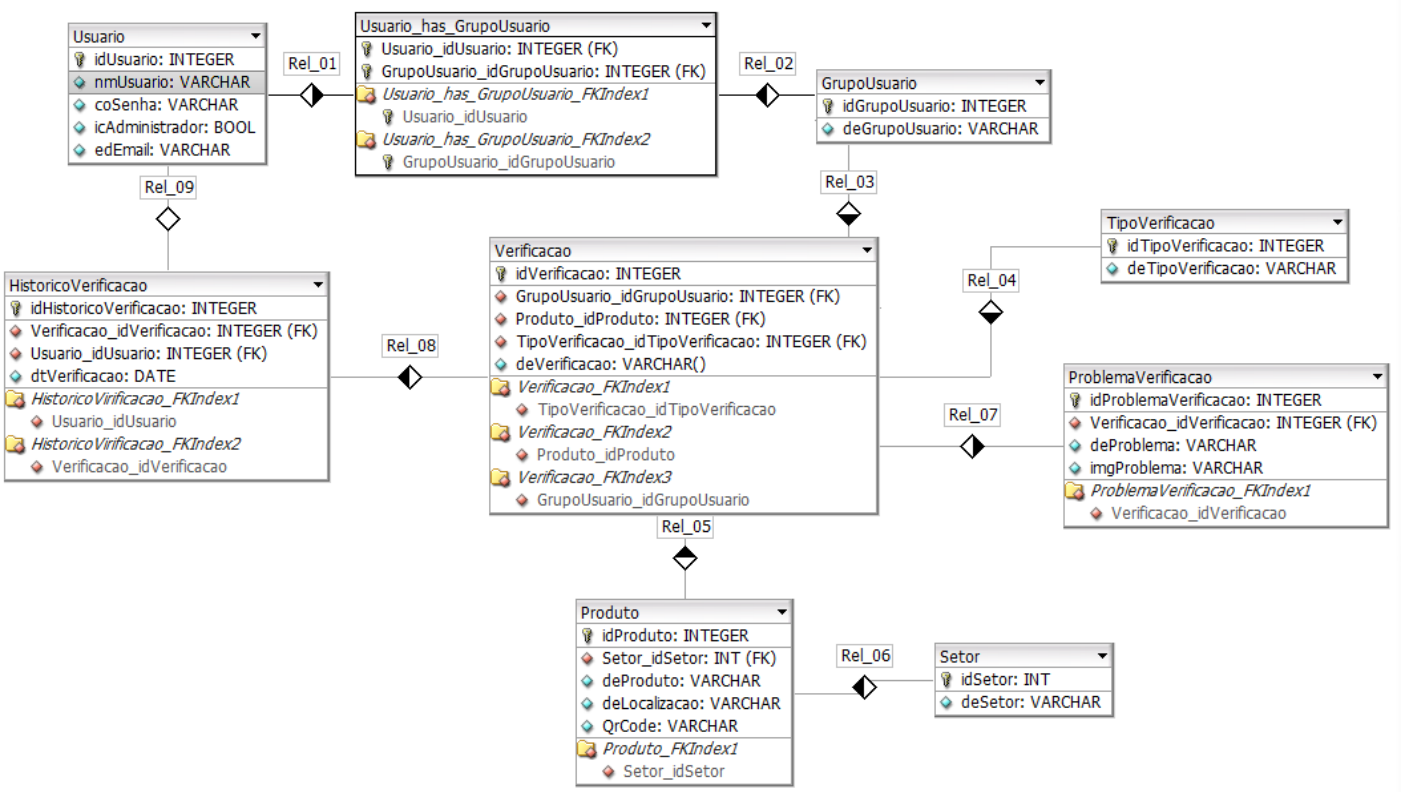


Figura 3.2 – Modelagem do Banco de Dados
Fonte: Autor

A tabela Usuario_has_GrupoUsuario é uma tabela de ligação entre a Usuario e a GrupoUsuario. Ela é necessária porque a relação estabelecida entre as tabelas Usuario e GrupoUsuario é de muitos para muitos (N:N). Ela possui duas chaves primárias, a Usuario_idUsuario e GrupoUsuario_idGrupoUsuario, ambas são chaves estrangeiras (Foreign Keys) e o tipo de dado é “integer”. Um campo da tabela definido como Foreign Key (FK) é utilizada para estabelecer o relacionamento entre duas tabelas.

A tabela GrupoUsuario possui uma chave primária, chamada de idGrupoUsuario, representando cada grupo de usuários com um número, por isso o tipo de dado é “integer”. Possui a seguinte linha:

- deGrupoUsuario - representando a descrição do grupo de usuários, o tipo do dado é “varchar”.

A tabela TipoVerificacao possui uma chave primária, chamada de idTipoVerificacao, representando cada verificação com um número, por isso o tipo de dado é “integer”. Ela possui a seguinte linha:

- deTipoVerificacao - representando a descrição do tipo de verificacao, o tipo do dado é “varchar”.

A tabela Setor possui uma chave primária, chamada de idSetor, representando cada setor do shopping com um número, por isso o tipo de dado é “integer”. Ela possui a seguinte linha:

- deSetor - representando a descrição do setor que será feito a verificacao, o tipo do dado é “varchar”.

A tabela Produto possui uma chave primária, chamada de idProduto, representando cada produto que será verificado com um número, o tipo de dado é “integer”. Possui as seguintes linhas:

- Setor_idSetor – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela Setor, o tipo do dado é “int”.
- deProduto - representando a descrição de cada produto, o tipo do dado é “varchar”.
- deLocalizacao - representando o local aonde está fisicamente instalado cada produto, o tipo do dado é “varchar”.
- QrCode - representando a string do QR Code, o tipo do dado é “varchar(max)”.

A tabela Verificacao é a tabela central do banco de dados, ela que faz o relacionamento e cruza as informações com todas as outras tabelas. Ela possui uma chave primária, chamada de idVerificacao, representando cada vistoria com um número, o tipo de dado é “integer”. Esta tabela possui as seguintes linhas:

- GrupoUsuario_idGrupoUsuario – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela GrupoUsuario, o tipo do dado é “integer”.
- Produto_idProduto – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela Produto, o tipo do dado é “integer”.

- TipoVerificacao_idTipoVerificacao – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela TipoVerificacao, o tipo do dado é “integer”.
- deVerificacao - representando a descrição da vistoria, o tipo do dado é “varchar”.

A tabela ProblemaVerificacao somente será populada de dados caso venha a ocorrer um problema enquando estiver realizando uma vistoria. Ela possui uma chave primária, chamada de idProblemaVerificacao, representando cada problema com um número, o tipo de dado é “integer”. Possui as seguintes linhas:

- Verificacao_idVerificacao – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela Verificacao, o tipo do dado é “integer”.
- deProblema - representando a descrição de cada problema ocorrido, o tipo do dado é “varchar”.
- imgProblema – será armazanada uma imagem de cada problema encontrado, o tipo do dado é “VARBINARY(MAX)” por razão de uma imagem ser formada por um grupo de vários “0” e “1” e de tamanho variável.

A tabela HistoricoVerificacao possui uma chave primária, chamada de idHistoricoVerificacao, representando cada histórico de verificações com um número, o tipo de dado é “integer”. Possui as seguintes linhas:

- Verificacao_idVerificacao – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela Verificacao, o tipo do dado é “integer”.
- Usuario_idUsuario – é uma FK, tem o papel de representar a mesma linha que é a chave primária da tabela Usuario, o tipo do dado é “integer”.
- dtVerificacao - representando a data que foi realizada a verificação, o tipo do dado é “date”, ou seja, uma data.

3.4.2 Exportação do Banco de dados

O DBDesigner possui uma opção de exportar o datagrama criado no formato de criação de um SQL Script, já com todas as suas relações. Para realizar estes comandos é necessário seguir o caminho “File -> Export -> SQL Create Script”. Em seguida será aberto

um Pop-Up “Export SQl Scrip”, onde deve ser selecionada a opção “Copy Script to Clipboard”. Com isto feito, o script pode ser copiado para o NotePad para realizar algumas correções antes de ser copiado para o SSMS.

Deverá ser retirado do script todas as palavras “Unsigned”, por motivo de não ser mais necessário na versão mais atual do SQL, a versão 12.

A propriedade “auto_increment” deverá ser substituída pela propriedade “identity”, que criará na tabela em questão uma coluna de identidade, tal propriedade é utilizada juntamente com a instrução CREATE TABLE. Sintaxe também utilizada a partir do SQL Server 2008.

Na tabela Usuario, é necessário que seja substituída a referência “Bool” por “Bit”, a versão 12 do SQL não entende Bool.

Na tabela Produto na coluna QrCode é necessário que seja substituída a referência “VARCHAR” por “VARBINARY”, a versão 12 do SQL necessita desse tipo de dados para imagem.

Na tabela ProblemaVerificacap na coluna ImgProblema é necessário que seja substituída a referência “VARCHAR” por “VARBINARY”, a versão 12 do SQL necessita desse tipo de dados para imagem.

O script final que implementa todo o banco de dados está descrito abaixo:

```
CREATE TABLE Setor (                        /*Criar tabela de Setor*/
  idSetor INT  NOT NULL IDENTITY,
  deSetor VARCHAR NULL,
  PRIMARY KEY(idSetor)
);

CREATE TABLE TipoVerificacao (              /*Criar tabela do Tipo de
Verificição*/
  idTipoVerificacao INTEGER  NOT NULL IDENTITY,
  deTipoVerificacao VARCHAR NULL,
  PRIMARY KEY(idTipoVerificacao)
);

CREATE TABLE Usuario (                      /*Criar tabela de Usuário*/
  idUsuario INTEGER  NOT NULL IDENTITY,
  nmUsuario VARCHAR NULL,
  coSenha VARCHAR NULL,
  icAdministrador BIT NULL,
  edEmail VARCHAR NULL,
  PRIMARY KEY(idUsuario)
);

CREATE TABLE GrupoUsuario (                 /*Criar tabela de Grupo de Usuário*/
  idGrupoUsuario INTEGER  NOT NULL IDENTITY,
```

```
  deGrupoUsuario VARCHAR NULL,
  PRIMARY KEY(idGrupoUsuario)
);

CREATE TABLE Produto (                    /*Criar tabela de Produto*/
  idProduto INTEGER  NOT NULL IDENTITY,
  idSetor INTEGER  NOT NULL,
  deProduto VARCHAR NULL,
  deLocalizacao VARCHAR NULL,
  QR Code VARBINAY NULL,
  PRIMARY KEY(idProduto),
  INDEX Produto_FKIndex1(idSetor),
  FOREIGN KEY(idSetor)
    REFERENCES Setor(idSetor)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE UsuarioGrupoUsuario (      /*Criar tabela de ligação de Usuário e
Geupo de Usuário*/
  idUsuario INTEGER  NOT NULL,
  idGrupoUsuario INTEGER  NOT NULL,
  PRIMARY KEY(idUsuario, idGrupoUsuario),
  INDEX Usuario_has_GrupoUsuario_FKIndex1(idUsuario),
  INDEX Usuario_has_GrupoUsuario_FKIndex2(idGrupoUsuario),
  FOREIGN KEY(idUsuario)
    REFERENCES Usuario(idUsuario)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(idGrupoUsuario)
    REFERENCES GrupoUsuario(idGrupoUsuario)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE Verificacao (              /*Criar tabela de Verificação*/
  idVerificacao INTEGER  NOT NULL IDENTITY,
  idGrupoUsuario INTEGER  NOT NULL,
  idProduto INTEGER  NOT NULL,
  idTipoVerificacao INTEGER  NOT NULL,
  deVerifcacao VARCHAR NULL,
  PRIMARY KEY(idVerificacao),
  INDEX Verificacao_FKIndex1(idTipoVerificacao),
  INDEX Verificacao_FKIndex2(idProduto),
  INDEX Verificacao_FKIndex3(idGrupoUsuario),
  FOREIGN KEY(idTipoVerificacao)
    REFERENCES TipoVerificacao(idTipoVerificacao)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(idProduto)
    REFERENCES Produto(idProduto)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(idGrupoUsuario)
    REFERENCES GrupoUsuario(idGrupoUsuario)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE ProblemaVerificacao (      /*Criar tabela de Problema de
Verificação*/
```

```
  idProblemaVerificacao INTEGER NOT NULL Identity,
  idVerificacao INTEGER NOT NULL,
  deProblemaVerificacao VARCHAR NULL,
  imgProblema VARBINARY NULL,
  PRIMARY KEY(idProblemaVerificacao),
  INDEX ProblemaVerificacao_FKIndex1(idVerificacao),
  FOREIGN KEY(idVerificacao)
    REFERENCES Verificacao(idVerificacao)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);

CREATE TABLE HistoricoVirificacao (     /*Criar tabela de Histórico de
Verificição*/
  idHistoricoVirificacao INTEGER  NOT NULL IDENTITY,
  idVerificacao INTEGER  NOT NULL,
  idUsuario INTEGER  NOT NULL,
  dtVerificacao DATE NULL,
  PRIMARY KEY(idHistoricoVirificacao),
  INDEX HistoricoVirificacao_FKIndex1(idUsuario),
  INDEX HistoricoVirificacao_FKIndex2(idVerificacao),
  FOREIGN KEY(idUsuario)
    REFERENCES Usuario(idUsuario)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION,
  FOREIGN KEY(idVerificacao)
    REFERENCES Verificacao(idVerificacao)
      ON DELETE NO ACTION
      ON UPDATE NO ACTION
);
```

3.4.3 Instalação do Banco de dados

O Banco de Dados SQL Server será processado em uma máquina virtual criada no servidor da Azure. Com o script gerado e corrigido, o SQL Server Management Studio é responsável por criar todo o banco, as tabelas e os relacionamentos. O SSMS está conectado ao Azure com as mesmas credenciais, permitindo assim que qualquer alteração feito no SSMS seja atualizada instantaneamente no BD armazenado no Azure. Deixando também a possibilidade de mais de uma pessoa poder fazer alterações pelo SSMS.

O BD foi criado para armazenar os dados a partir da sincronização do smartphone com com o back-end do sistema na nuvem Azure. Toda a comunicação e movimentação de dados é realiza por APIs presentes no código do apicativo e APIs no código do serviço processado no Azure.

A implantação do banco de dados no Azure é feita da seguinte forma:

1. Cadastramento no portal da Azure.

2. Deverão ser preenchidos os seguintes campos abaixo:

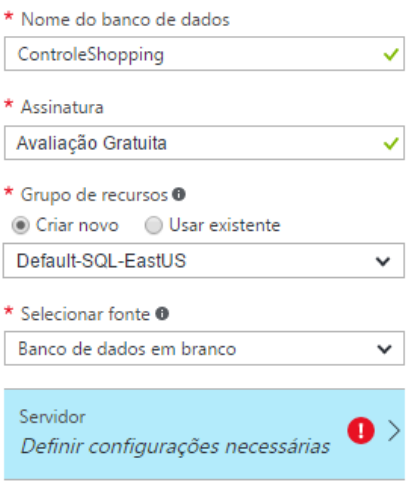


Figura 3.3 – Preenchimento de dados do Banco de Dados
Fonte: Autor

3. Logo abaixo, deverá selecionar a opção “Servidor” que está com um alerta.
4. Deverá selecionar a opção “Criar um novo servidor”.
5. Os campos foram preenchidos de acordo com a Figura 3.4 abaixo:

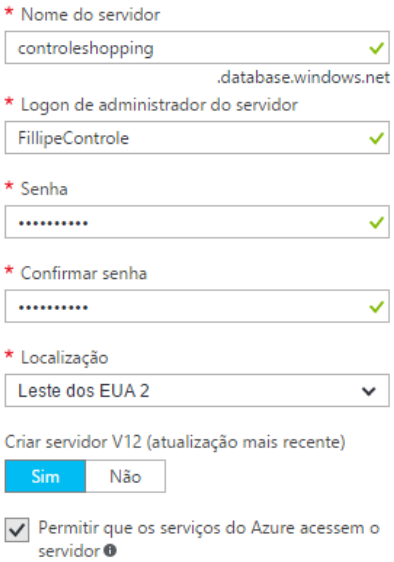


Figura 3.4 – Preenchimento de dados do Servidor
Fonte: Autor

6. As credenciais escolhida de acordo com o critérios. Logon de administrador: FestaDuraBD. Senha: Brasil@)20.
7. Após preenchidos com tais informações, deverá ser selecionado a opção “Selecionar”.
8. No campo “Deseja usar o pool elástico SQL?”, deverá ser selecionada a opção “Agora não”.
9. No campo “Tipo de preço”, deverá ser selecionada a opção “Básico”.
10. No campo “Agrupamento”, não foi alterado o que foi sugerido, continuou “SQL_Latin1_General_CP1_CI_AS”.
11. Deverá ser selecionado a opção “Criar”.
12. Após alguns minutos o Banco de Dados SQL v12 já está pronto.

Após a implantação do BD no Azure, deverão ser configurados no Firewall do BD os IPs externos que poderão ter acesso ao BD. No caso do protótipo, foram configurados os IPs do computador que está com o SSMS instalado. A configuração foi realizada da seguinte forma:

1. No azure, selecionar o BD ControleShopping que foi criado.
2. Selecionar a opção “Definir firewall do servidor”.
3. Deverá ser configurado da seguinte forma:

Figura 3.5 – Configuração de Firewall
Fonte: Autor

4. Após o preenchimento dos IPs Inicial e Final, deverá ser selecionada a opção “Salvar”.
5. Na página da Visão geral do BD, em Fundamentos, deverá ser selecionada a opção “Mostrar cadeias de conexão do baco de dados”. Como mostrado na Figura 3.6.

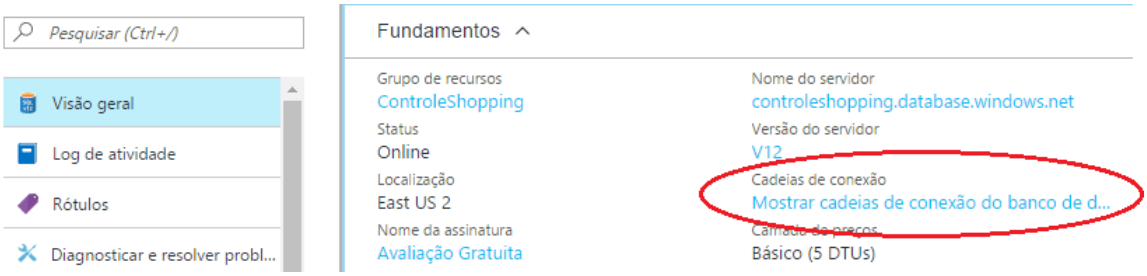


Figura 3.6 – Cadeias de Conexão
Fonte: Autor

6. Na página de Cadeias de conexão do banco de dados, deverá sem copiado o seguinte caminho destacado em azul “ses22h08rw.database.windows.net,1433”. Onde se encontra explicitamente o número de porta e o protocolo com o nome do servidor.

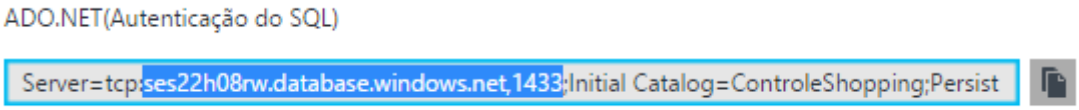


Figura 3.7 – Autenticação do SQL
Fonte: Autor

Após de realizar todas as etapas acima, deverá ser configurada a comunicação do Azure com o SSMS instalado no computador. Segue abaixo a configuração a ser realizada:

1. Realizar o download do software no link < https://msdn.microsoft.com/pt-br/library/mt238290.aspx> e instalar o software.
2. Abrir o SSMS.
3. Será aberta a seguinte janela:

Figura 3.8 – Conectando o SSMS ao Azure
Fonte: Autor

4. No campo “Server name:”, deverá ser copiado o campo que foi destacado em azul em “Cadeias de conexão do banco de dados”. Que se refere a “ses22h08rw.database.windows.net,1433”.
5. No campo “Authentication:”, deverá ser selecionada a opção de “SQL Server Authentication”.
6. As credenciais a serem usadas devem ser as que foram criadas no Azure no momento de criação do BD.
7. Após o preenchimento dos dados, ficará como pode-se observar na Figura 3.9. Deve-se selecionar a opção “Connect”.

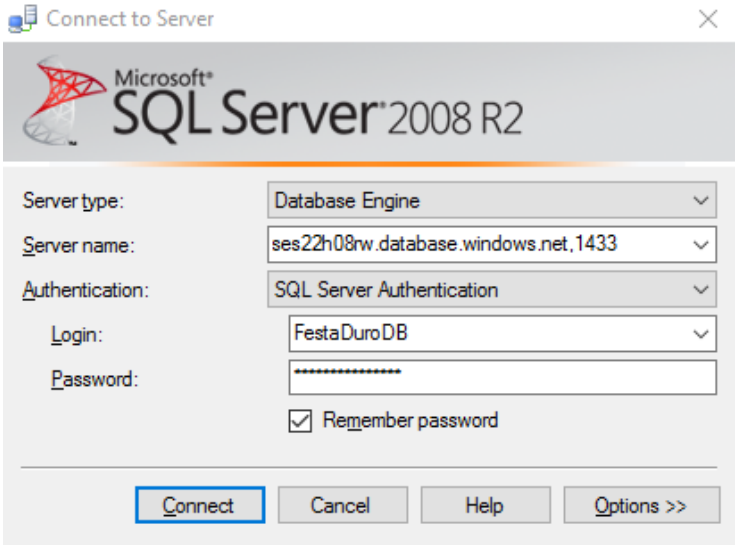


Figura 3.9 – Dados de conexão preenchidos do SSMS
Fonte: Autor

8. Será apresentado em tela, todos os BDs criados no Azure conectado.
9. Selecionar o BD ControleShopping.
10. Pressionar as teclas “Ctrl + N” para criar novas tabelas no BD.
11. Copiar o Script Final gerado pelo DBDesigner para o SSMS.
12. Por fim, salvar o arquivo.

Feitos todos esses passos, o BD do SSMS está conectado ao Azure. Todas alterações realizadas no BD serão atualizadas automaticamente no Azure. Os últimos passos a serem feitos tem a finalidade de conectar o BD do Azure ao Visual Studio, para que o sistema utilize o BD armazenado no Azure como sua base de dados. É muito semelhante a conexão feita do SSMS com o Azure, uma vez que o propósito é colocar no Visual Studio as Cadeias de conexão do banco de dados do Azure. Segue abaixo a configuração a ser realizada.

1. Seguir o caminha mostrado na Figura 3.10:

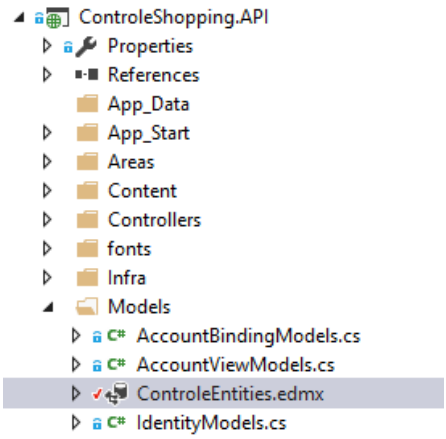


Figura 3.10 – Local do Banco de Dados no Visual Studio
Fonte: Autor

2. No Visual Studio, selecionar o “namespace: ControleShopping.API”.
3. Abrir a pasta “Models”.
4. Clicar no “ControleEntities.edmx”.
5. Apertar com o botão direito na página em branco que irá abrir.
6. Selecionar a opção “Update Model from Database...”.
7. Será aberta a página “Update Wizard” como pode-se observar na Figura 3.11.

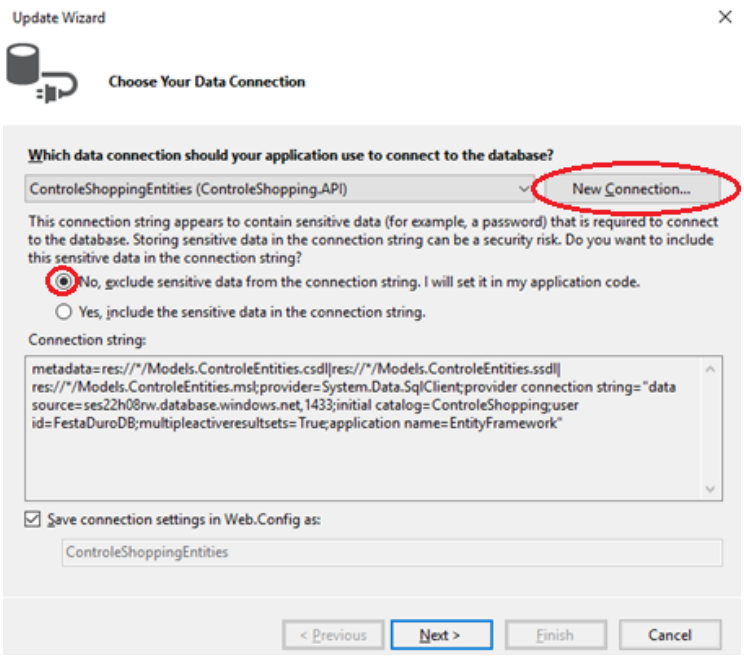


Figura 3.11 – Conexão do BD do Azure com o Visual Studio
Fonte: Autor

8. Deverá ser selecionada a opção “No...” como resposta a pergunta.
9. Deverá selecionar o botão “New Connection...”.
10. Abrirá outra página chamda de “Connection properties”, como pode-se observar na Figura 3.12.

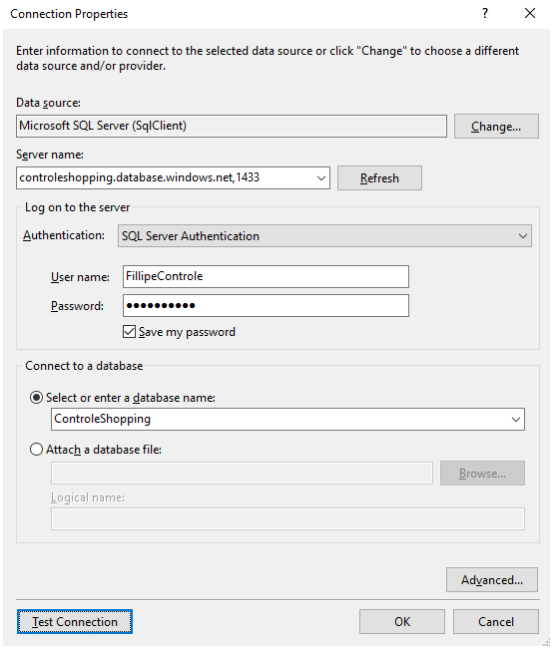


Figura 3.12 – Preenchimento de dados de conexão com o BD
Fonte: Autor

11. Está página deverá ser preenchida da forma que está demonstrada na Figura 3.12.
12. Após preenchidos todos os campos da devida forma, selecionar o botão “OK”.
13. Finalizar o procedimento na página “Update wizard”, selecionando o botão “Finish”.

3.4.4 Inserir dados no Banco de Dados

Com o BD hospedado na nuvem Azure, podendo ser gerenciado pelo SSMS e conectado ao sistema no Visual Studio, ainda terá uma ação a ser tomada, acrescentar novos dados ao banco de dados.

Para inserir dados no BD, será utilizado o SSMS. Será mostrado aqui um breve código para realizar esta função na tabela Usuario do banco de dados ControleShopping:

```
INSERT INTO [ControleShopping].[dbo].[Usuario]
        ([idUsuario]
        ,[nmUsuario]
        ,[coSenha]
        ,[icAdministrador]
        ,[edEmail])
    VALUES
        (1
        ,'Fillipe de Sousa Moura'
        ,'cannondale'
        ,1
        ,'fillipemoura2@gmail.com')
GO
```

Também pode ser realizado o comando ‘INSERT INTO’ para inserção de dados em todas tabelas do banco.

3.4.5 Executar consulta no Banco de dados

A consulta ao banco de dados que será realizada é a Transact-SQL (T-SQL) que segue um breve passo a passo.

1. Na página “Object Explore”, clique o botão direito do mouse sobre o BD.

2. Selecione a opção “New Query”.
3. Na página de Consulta, copie o código abaixo:

```
“Select
   idUsuario
   ,nmUsuario
   ,coSenha
   ,icAdministrador
   ,edEmail
FROM dbo.Usuario”
```

4. O resultado da pesquisa aparecerá na janela “Resultados”. Como pode-se observar na Figura 3.13.

Results | Messages

	idUsuario	nmUsuario	coSenha	icAdministrador	edEmail
1	1	Fillipe de Sousa Moura	cannondale	1	fillipemoura2@gmail.com
2	2	Professor Francisco Javier	uniceub	0	francisco.javier@uniceub.br
3	3	Carina Maldi	12345	0	carinamaldi.arq@gmail.com
4	4	Edneide Moura	edmoura	0	edneidemoura@yahoo.com.br
5	5	Teste	123	0	teste
6	6	Anderson Patriota	capitaoamerica	1	apatriota@iguatemi.com.br
7	7	Segurança 1 TESTE	00000	0	seguranca1teste@gmail.com
8	8	Brigadista 1 TESTE	00000	0	brigadista1teste@gmail.com

Figura 3.13 – Resultados do Select na tabela Usuario
Fonte: Autor

3.5 Terceira etapa: Desenvolvimento do código

3.5.1 Fluxo do sistema

A terceira etapa do desenvolvimento do protótipo é o desenvolvimento do código. O desenvolvimento foi feito na plataforma Visual Studio na linguagem C#. Como pode-se observar na Figura 3.14, o fluxograma da aplicação mostra todas as etapas de funcionamento.

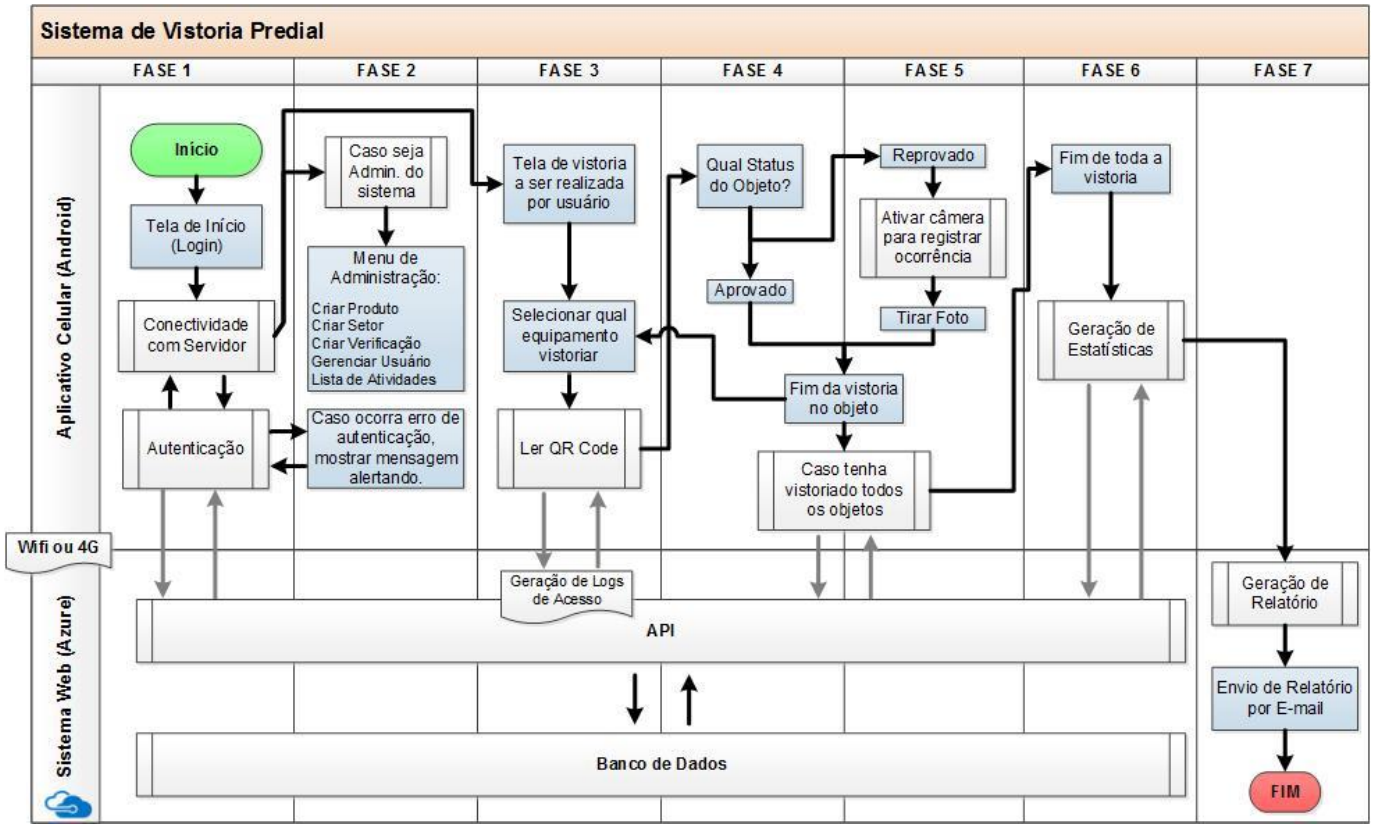


Figura 3.14 – Fluxograma do sistema
Fonte: Autor

O fluxograma mostra todo o fluxo do sistema de vistoria. Esta vistoria tem o objetivo de verificar o estado de equipamentos e locais de combate a incêndio e demais necessidades de segurança básicas de uma edificação, visando a segurança das pessoas que lhe frequentam.

O funcionamento do sistema foi dividido em 7 Fases, havendo cada uma delas sua função primordial. A grande parte do sistema é processado localmente no aplicativo do celular, mas a base de dados fica alocada no sistema web hospedado no Azure, com isso, em todas

as 7 fases do sistema estabelece comunicação entre o aplicativo e o sistema web para transferência de dados. Podendo ser uma conexão para leitura de dados, escrita de novos dados ou alteração de dados existentes.

Para a comunicação entre o aplicativo e o sistema web, é necessário que tenha uma conexão Wifi ou 4G no smartphone.

Todo acesso do aplicativo ao banco de dados SQL é realizado pelo sistema web que utiliza uma API. O sistema web possui um serviço de API, o qual gerencia toda a troca de informação, chamado de “ControleShopping.API”. O sistema possui outros projetos com funções distintas que são requisitados por essa API, como o projeto que resolve a forma de tratar o dado, resolução HTTP, os parâmetros utilizados e entre outros. Entretanto, no fluxograma está descrito basicamente como ‘API’. A explicação detalhada de cada um dos projetos da API será abordado no próximo subitem que descreve componentes do código.

Com a API fazendo o tratamento dos dados, a troca de informação fica transparente para toda a aplicação. Independente de qual tabela esteja, qual seja o objeto pesquisado ou aonde quer atribuí-lo.

O sistema se inicia com uma tela de Login. Onde é possível ser preenchido os campos de usuário e senha, como pode-se observar na Figura 3.15.

Figura 3.15 – Tela de Login
Fonte: Autor

Esta tela de início é a fase 1 do fluxograma. Ao clicar o botão “Login”, o aplicativo estabelecerá conexão com a nuvem e fará a autenticação do usuário ali aplicado. Caso o usuário esteja cadastrado no BD, passará para a próxima fase. Caso não esteja, o aplicativo mostrará uma mensagem de erro.

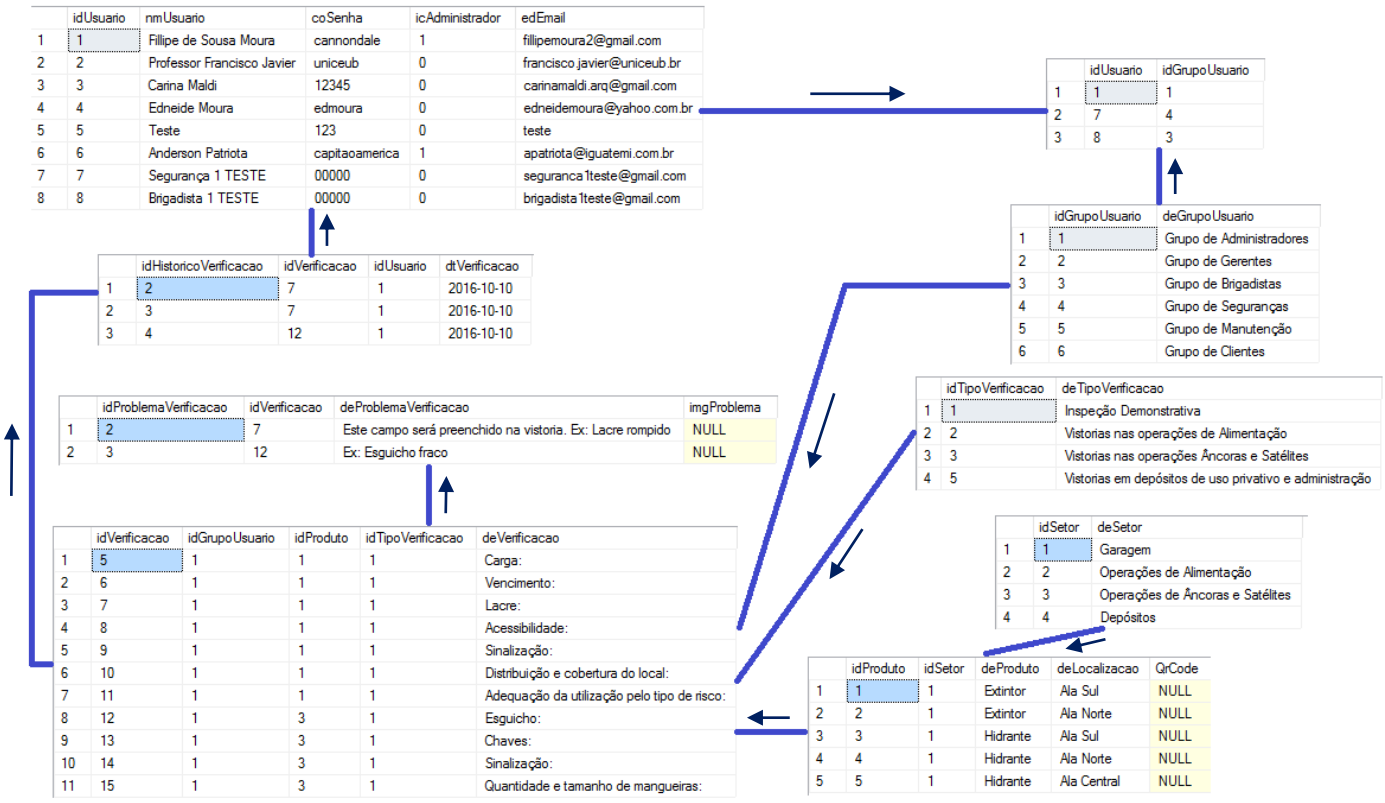
Na Fase 2, o sistema faz uma busca no BD na tabela dbo.Usuario na coluna icAdministrador. Caso tenha o valor “1”, ele é um administrador. Caso tenha o valor “0”, ele é um usuário comum. Sabendo qual o grau de privilégio do usuário, o sistema define se ele acessará a tela de “Menu” caso seja administrador ou se ele vai para a Fase 3 caso seja um usuário comum. Ainda na Fase 2, a tela de Menu mostrada na Figura 3.16, possui as opções de gerenciamento do sistema, como, Gerar QR Code, Modificar Usuário, Modificar Grupo de Usuário, Modificar Produtos, Modificar Setores, Modificar Verificações e Modificar Tipos de Verificação. Qualquer alteração realizada na tela Menu, altera diretamente o banco de dados do sistema. É neste local que são gerados os QR Codes dos produtos. Dessa maneira cada equipamento ou área que será vistoriado, possuirá um QR Code próprio gerado pelo sistema. Consequentemente cada objeto terá sua ‘identidade’ própria, não havendo maneira de fraudar a vistoria.

Figura 3.16 – Tela de Administração
Fonte: Autor

Para a geração de novas etiquetas de QR Code, o usuário administrador necessita selecionar a opção “Gerar QR Code”. Será demonstrada em tela a nova etiqueta, que deve ser impressa e colada ao objeto.

Na Fase 3 é onde começa a vistoria. A primeira ação do aplicativo nesta fase é mostrar qual será a vistoria que o usuário deve realizar. Com a autenticação do usuário na fase 1, cada usuário possui um Grupo e cada Grupo de Usuário está correlacionado a um Tipo de Verificação. Dessa forma, cada grupo de usuário possui um tipo de verificação diferente para realizar, por exemplo, o usuário Fillipe Moura está no grupo de Administradores, tal grupo está correlacionado ao tipo de verificação “Inspeção Demonstrativa”. Na tabela ‘Verificacao’ são correlacionados quais serão as verificações que cada grupo de usuário pode realizar, também atribuindo tal verificação à um tipo de verificação.

A Figura 3.17 mostra de forma gráfica os dados que foram inseridos no banco de dados e auxilia na interpretação do parágrafo acima.



	idUsuario	nmUsuario	coSenha	icAdministrador	edEmail
1	1	Fillipe de Sousa Moura	cannondale	1	fillipemoura2@gmail.com
2	2	Professor Francisco Javier	uniceub	0	francisco.javier@uniceub.br
3	3	Carina Maldi	12345	0	carinamaldi.arq@gmail.com
4	4	Edneide Moura	edmoura	0	edneidemoura@yahoo.com.br
5	5	Teste	123	0	teste
6	6	Anderson Patriota	capitaoamerica	1	apatriota@iguatemi.com.br
7	7	Segurança 1 TESTE	00000	0	seguranca1teste@gmail.com
8	8	Brigadista 1 TESTE	00000	0	brigadista1teste@gmail.com

	idUsuario	idGrupoUsuario
1	1	1
2	7	4
3	8	3

	idGrupoUsuario	deGrupoUsuario
1	1	Grupo de Administradores
2	2	Grupo de Gerentes
3	3	Grupo de Brigadistas
4	4	Grupo de Seguranças
5	5	Grupo de Manutenção
6	6	Grupo de Clientes

	idHistoricoVerificacao	idVerificacao	idUsuario	dtVerificacao
1	2	7	1	2016-10-10
2	3	7	1	2016-10-10
3	4	12	1	2016-10-10

	idProblemaVerificacao	idVerificacao	deProblemaVerificacao	imgProblema
1	2	7	Este campo será preenchido na vistoria. Ex: Lacre rompido	NULL
2	3	12	Ex: Esguicho fraco	NULL

	idTipoVerificacao	deTipoVerificacao
1	1	Inspeção Demonstrativa
2	2	Vistorias nas operações de Alimentação
3	3	Vistorias nas operações Âncoras e Satélites
4	5	Vistorias em depósitos de uso privativo e administração

	idVerificacao	idGrupoUsuario	idProduto	idTipoVerificacao	deVerificacao
1	5	1	1	1	Carga:
2	6	1	1	1	Vencimento:
3	7	1	1	1	Lacre:
4	8	1	1	1	Acessibilidade:
5	9	1	1	1	Sinalização:
6	10	1	1	1	Distribuição e cobertura do local:
7	11	1	1	1	Adequação da utilização pelo tipo de risco:
8	12	1	3	1	Esguicho:
9	13	1	3	1	Chaves:
10	14	1	3	1	Sinalização:
11	15	1	3	1	Quantidade e tamanho de mangueiras:

	idSetor	deSetor
1	1	Garagem
2	2	Operações de Alimentação
3	3	Operações de Âncoras e Satélites
4	4	Depósitos

	idProduto	idSetor	deProduto	deLocalizacao	QrCode
1	1	1	Extintor	Ala Sul	NULL
2	2	1	Extintor	Ala Norte	NULL
3	3	1	Hidrante	Ala Sul	NULL
4	4	1	Hidrante	Ala Norte	NULL
5	5	1	Hidrante	Ala Central	NULL

Figura 3.17 – Banco de dados populado
Fonte: Autor

Ainda na Fase 3, na tela da lista de equipamentos que devem ser vistoriados, o usuário deverá selecionar qual equipamento ele deseja vistoriar. Estes equipamentos demonstrados estão representando os produtos da coluna ‘deProduto’ como pode-se observar na Figura 3.17. Após selecionar o objeto, o aplicativo abrirá a câmera do smartphone para fazer a leitura do QR Code. Neste momento o aplicativo realiza uma consulta no BD. Caso não reconheça o QR Code, lhe mostrará um erro de identificação não realizada. As consultas realizadas com sucesso geram um Log de Acesso, buscando manter um controle maior por parte do administrador. Se um dia ele desejar fazer uma auditoria na vistoria, poderá.

A Fase 4 é responsável por atribuir o status do objeto vistoriado. Após o aplicativo mostrar o que deve ser vistoriado no equipamento, como pode-se observar na Figura 3.18, abrirá o reconhecimento do QR Code, e perguntará qual é o Status do Objeto, lhe apresentando duas opções: Aprovado ou Reprovado. Se o objeto apresentar tudo em conformidade com o que deve ser vistoriado, a opção a ser marcada deverá ser “Aprovado”, desse modo, o fim da vistoria no objeto acaba e aplicativo redireciona o usuário para a tela de lista de equipamentos a serem vistoriados novamente e começa o processo de vistoria.

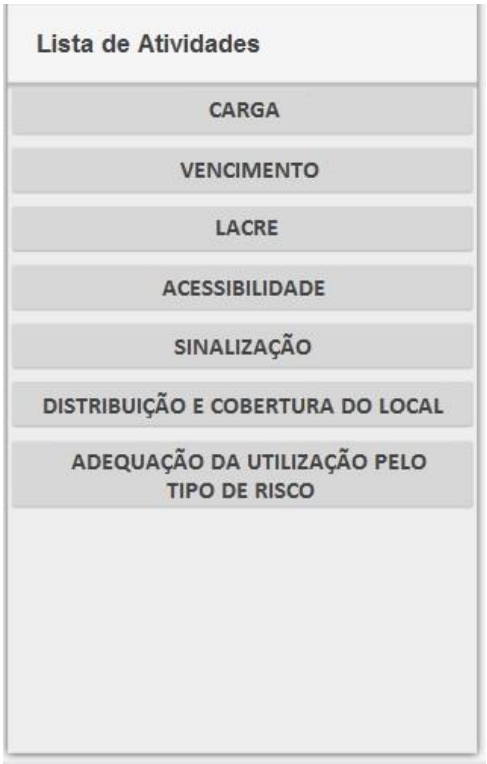


Figura 3.18 – Tela de Lista de atividades do Produto Extintor
Fonte: Autor

Ao perguntar o status do objeto, se o mesmo apresentar alguma inconformidade, a opção a ser macada deverá ser “Reprovado”. O aplicativo começa a Fase 5 neste momento, devendo o usuário preencher o campo de descrição do problema e selecionando o botão “Confirmar”, com isso a câmera do smartphone é ativada novamente para fotografar a inconformidade encontrada. Ao finalizar este procedimento, a foto é armazenada no BD na tabela ProblemaVerificaca na coluna ‘imgProblema’.

Com estas etapas terminadas, ainda na Fase 5, caso tenha sido feita a vistoria de todos os equipamentos desta verificação, o aplicativo é direcionado para a Fase 6, onde finaliza toda a vistoria e gera um histórico de verificação. O próprio aplicativo faz uma análise de como foi a vistoria e gera estatísticas, como, quantos equipamentos foram vistoriados, quantas verificações estavam de acordo com as regras e quantas existiam problemas.

A Fase 7 é a representação dos resultados da vistoria. Em sua última etapa que é possível ter a visualização dos resultados e que se encontra o momento em que o aplicativo finaliza seus processos. A Figura 3.19 é um exemplo do Relatório de vistoria. De agora em diante, todos os dados já estão salvos no BD e o processo de geração de relatório é feito pelo Sistema Web.

Relatório do dia 05/12/16 Caixa de entrada x

fillipemoura2@gmail.com 18:59 (há 1 minuto)

para mim

Produto	Verificação	Data da Verificação	Existe problema	Nome do Usuário
Hidrante 1	Esguicho	05/12/2016	Sim	Fillipe Moura
Hidrante 1	Esguicho	05/12/2016	Sim	Fillipe Moura
Hidrante 1	Chaves	05/12/2016	Não	Fillipe Moura
Hidrante 1	Sinalização	05/12/2016	Não	Fillipe Moura
Hidrante 1	Quantidade e tamanho de mangueiras	05/12/2016	Não	Fillipe Moura
Extintor 1	Carga	05/12/2016	Não	Fillipe Moura
Extintor 1	Vencimento	05/12/2016	Sim	Fillipe Moura
Extintor 1	Lacre	05/12/2016	Não	Fillipe Moura
Extintor 1	Acessibilidade	05/12/2016	Sim	Fillipe Moura
Extintor 1	Sinalização	05/12/2016	Sim	Fillipe Moura
Extintor 1	Distribuição e cobertura do local	05/12/2016	Sim	Fillipe Moura
Extintor 1	Adequação da utilização pelo tipo de risco	05/12/2016	Não	Fillipe Moura

Figura 3.19 – Relatório da vistoria
Fonte: Autor

Como abordado no Capítulo 1, o sistema emitirá um relatório para demonstrar produtos vistoriados, componentes vistoriados, data da verificação, status dos componentes vistoriados e o usuário. O objetivo do relatório é para que gestores do condomínio predial e

para que seus condôminos tenham controle da segurança do empreendimento. Estes receberam do servidor de e-mail feito em Arduino, o relatório produzido pelo sistema.

3.5.2 Componentes do código

O nome dado para o projeto foi ControleShopping2.0. O código do sistema é composto por 7 projetos, chamados de API.Core, API.DATA.DTO, API.HttpClient, API.Parameters, API.Services, ControleShopping.AndroidApp e ControleShopping API. Como pode-se observar na Figura 3.20, possui 5 projetos no formato Portable, 1 aplicativo Android e 1 API central de controle.

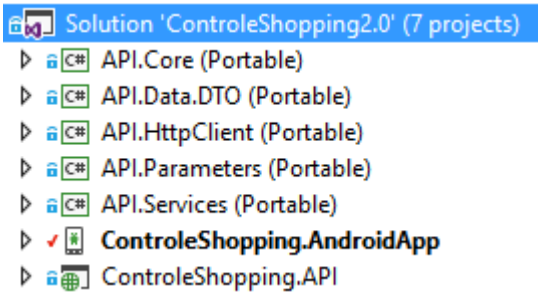


Figura 3.20 – Projetos do código
Fonte: Autor

A seguir será feita uma breve explicação de cada um dos projetos, mas antes disso é necessário saber o porquê do sistema estar dividido por projetos.

Os projetos Portable Class Library (PCL) que é o grande truque utilizado para programar em c# no Visual Studio e poder utilizar grande parte do código para no futuro desenvolver um aplicativo para iOS ou Windows Phone. A partir desde conceito de PCL que foi definida a estrutura de todo o código. Os projetos criados em formato PCL são necessários quando se desenvolve para um dispositivo portátil. Todas estes projetos são formas que a API principal (ControleShopping.API) utiliza para realizar todas as suas ações, isto significa que se o programador quiser desenvolver um aplicativo para o iOS, ele utilizaria todos os projetos protáteis e a API principal, só necessitaria desenvolver outro aplicativo. Porque todo o esqueleto do projeto já está pronto, a única coisa a se fazer é desenvolver um projeto aplicativo no formato que é requisitado pelo sistema operacional no smartphone.

Todo projeto PCL gera uma saída DLL. O .dll é um tipo de arquivo de execução de assembly. É a menor unidade na implantação de um aplicativo. Quando se realiza a instalação do aplicativo no smartphone, o arquivo .dll fica armazenado na pasta raiz do aplicativo. O método de utilização de arquivos .dll no projeto tornará as PCLs a forma mais ideal de disponibilizar os componentes e bibliotecas do projeto para a plataforna escolhida, o Android. Uma DLL pode ser usada para referenciar outra DLL.

Como pode-se observar no ‘Apêndice A’ anexado ao trabalho, as classes de um mesmo projeto PCL possui semelhanças em grande parte do código. Em razão de todas as classes pertecentes a um projeto estarem realizando funções para o mesmo namespace, alterando basicamente a atividade que realizará.

O projeto do aplicativo (ControleShopping.Android) fica instalado no próprio smartphone e o projeto ControleShopping.API fica localizado no sistema Web. A configuração de todo o sistema é pelo Visual studio, o qual está conectado com todos os serviços da nuvem Azure.

3.5.3 ControleShopping.AndroidApp

O aplicativo para o Android é o projeto “ControleShopping.AndroidApp”, nele que é estruturada como cada tela do aplicativo irá se comportar. Cada um das classes mostradas na Figura 3.21 representam uma ação que o aplicativo deve realizar. Os nomes dados para cada uma das classes descreve a sua principal função.

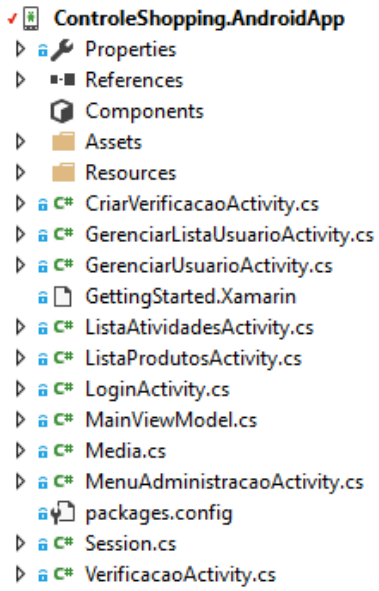


Figura 3.21 – Projeto do aplicativo ControleShopping.AndroidApp
Fonte: Autor

Será mostrado na Figura 3.22 como foi desenvolvido parte do código da Classe “LoginActivity.cs”. O objetivo de tal demonstração é deixar claro como é feita a estruturação de uma das ações desta classe, a ação de clicar o botão de login (btnLogin_Click), mas para isso foi necessário explicar o funcionamento do método OnCreate. Esta classe ainda possui outros métodos, são eles: btnRegistrar_Click e SetAlerta.

```
namespace ControleShopping.AndroidApp
{
    [Activity(Label = "ControleShopping.Android", MainLauncher = true, Icon = "@drawable/icon")]
    0 references
    public class LoginActivity : Activity
    {
        // Declarar as variáveis globais dos editores de texto de E-mail e Senha
        private EditText edEmail;
        private EditText coSenha;
        private UsuarioDTO usuario;
        private TextView alertMessageLogin;

        9 references
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);

            // Definir a visão de Login de acordo com a constante Login do Layout do Resource.
            SetContentView(Resource.Layout.Login);

            // Construir os botões para registrar as ações dos botões
            Button btnLogin = FindViewById<Button>(Resource.Id.btnLogin);
            Button btnRegistrar = FindViewById<Button>(Resource.Id.btnRegistrar);

            // Registrar as ações dos botões
            btnLogin.Click += btnLogin_Click;
            btnRegistrar.Click += btnRegistrar_Click;


        }

        protected void btnLogin_Click(object sender, EventArgs e)
        {
            //Registrar os editores de texto
            edEmail = FindViewById<EditText>(Resource.Id.edEmail);
            coSenha = FindViewById<EditText>(Resource.Id.coSenha);

            // Atribuir ao usuario um novo serviço (UsuarioAPIServices, descrito no projeto de API.Services) no formato descrito.
            usuario = new UsuarioAPIServices(Operacao.GetByEmailSenha, null, edEmail.Text, coSenha.Text).Get();

            // Se o idUsuario for maior que 0, ir para a classe ListaAtividades.cs do projeto LoginActivity.
            if (usuario.idUsuario > 0)
            {
                StartActivity(typeof(ListaAtividades));
            }
            else
            {
                SetAlerta();
            }
        }
```

Figura 3.22 – Classe LoginActivity do projeto ControleShopping.AndroidApp
Fonte: Autor

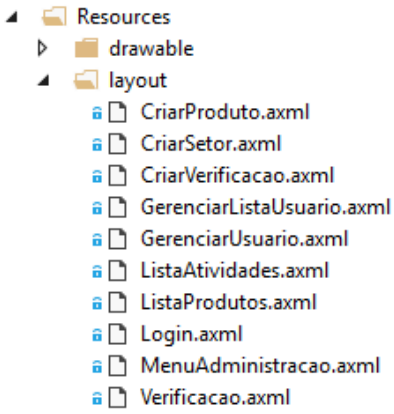


Figura 3.23 – Layouts do projeto ControleShopping.AndroidApp
Fonte: Autor

Também é de grande importância explicar do projeto ControleShopping.AndroidApp as telas de layout. São nelas que atribuí-se uma imagem em uma função ou classe feita em outro momento do código. Como pode-se observar na Figura 3.23, os layout criados no Xamarin foram: CriarProduto, CriarSetor, CriarVerificacao, GerenciarListaUsuario, GerenciarUsuario, ListaAtividades, ListaProdutos, Login, MenuAdministracao e Verificacao. A criação de cada um dos layouts foi realizada no Visual Studio.

3.5.4 ControleShopping.API:

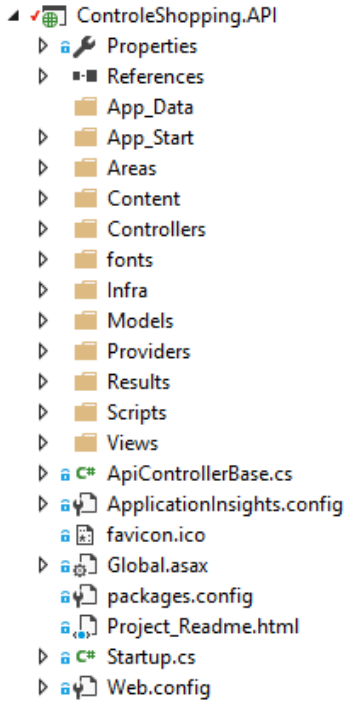


Figura 3.24 – Projeto da API principal ControleShopping.API
Fonte: Autor

Este projeto possui várias sub-pastas em sua estrutura, sendo que cada uma delas realiza uma função específica. A grande maioria são estruturas que já vem predefinidas ao se criar um projeto com características de um aplicativo para Android. Uma das mais importante é a pasta das Controllers, como pode-se observar na Figura 3.25. As classes controllers fazem o controle de cada uma das tabelas do banco de dados, enviando as mensagens de resposta HTTP e os tipos que operações que podem ser realizadas, como o Get e o GetList.

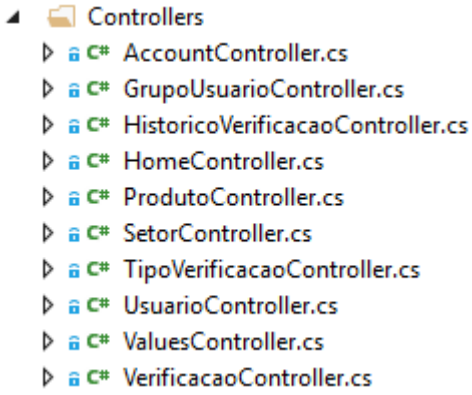


Figura 3.25 – Classes Controllers do projeto ControleShopping.API
Fonte: Autor

O exemplo utilizado para demonstrar a estruturação do código das Controllers será da classe GrupoUsuarioController.cs. Observe a Figura 3.26.

```
public class GrupoUsuarioController : ApiControllerBase
{
    private ControleShoppingEntities db = new ControleShoppingEntities();

    [HttpGet]
    // Realiza Get da fromUri, que retorna 'parameters' com os paramêtros de GrupoUsuarioParameters.
    0 references | 0 requests | 0 exceptions
    public HttpResponseMessage Get([FromUri]GrupoUsuarioParameter parameters)
    {
        object result = null;
        int nuTotalRegistros = 0;
        switch (parameters.idOperacao)
        {
            case (int)GrupoUsuarioParameter.Operacao.Get: // Está atribuindo quais os tipo de operações do Get
                {
                    var GrupoUsuario = db.GrupoUsuario.Where(tipoVeri => tipoVeri.idGrupoUsuario == parameters.idGrupoUsuario).FirstOrDefault();
                    result = GrupoUsuario;
                    nuTotalRegistros = 1;
                }
                break;
            case (int)GrupoUsuarioParameter.Operacao.GetList: //atribuindo oq é a operação getlist
                {
                    var listaGrupoUsuario = db.GrupoUsuario.Select(tipoVeri => tipoVeri.idGrupoUsuario > 0);
                    result = listaGrupoUsuario;
                    nuTotalRegistros = 1;
                }
                break;
        }
        return GetOkResponse(result, nuTotalRegistros);
    }
```

Figura 3.26 – Classe GrupoUsuarioController
Fonte: Autor

Outro ponto necessário de ser comentado no projeto ControleShopping.API é a pasta Models, que possui o arquivo ControleEntities.edmx, como mostrado na Figura 3.27. Este arquivo é responsável por armazenar na API todas as informações de estrutura do banco de dados. É nele que é configurada a sincronização com o banco de dados hospedado no Azure, como explicado no item 3.4.2.

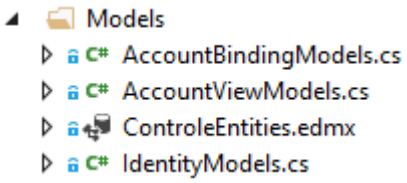


Figura 3.27 – Pasta Models do projeto ControleShopping.API
Fonte: Autor

Na Figura 3.28 está demonstrado como o banco está configurado na API. Caso deseje fazer alterações na estrutura do BD, não é recomendado realizar diretamente no arquivo.edmx. A melhor opção para evitar corromper os dados, é fazer as alterações no SSMS e após finalizado, realizar uma nova sincronização do BD com o Visual, como explicado no item 3.4.2.

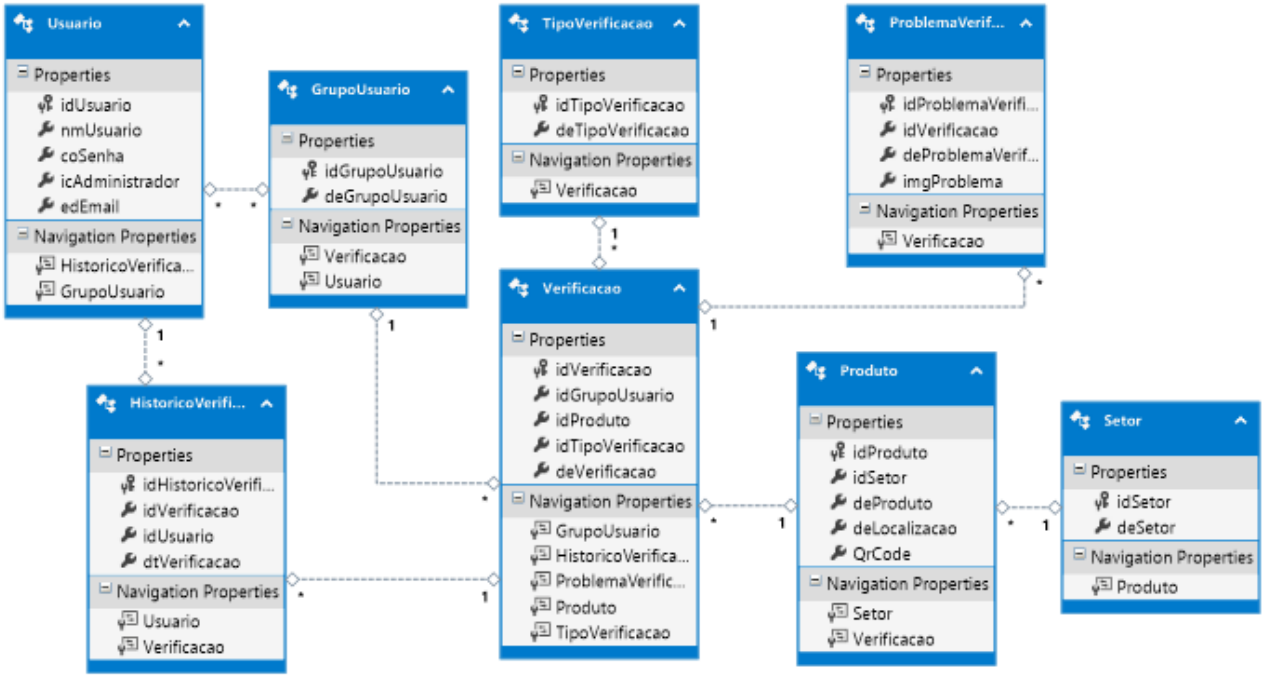


Figura 3.28 – Arquivo .edmx – Estrutura do BD
Fonte: Autor

3.5.5 API.Core:

É um namespace utilizado para criar as regras de negócio do sistema. A grande vantagem de utilizar este namespace é de poder referenciar ao mesmo tempo tanto no ControleShopping.AndroidApp, quanto no ControleShopping.API. Ao invés de ter que fazer como será a comunicação nos dois, é desenvolvido esse código para dizer como será feita a troca de informação.

Uma das regras de como o sistema vai se comunicar é que o aplicativo não referencia o API.Core diretamente, o aplicativo vai sempre referenciar o API.HttpClient para estabelecer está comunicação. Dessa forma, o API.HttpClient que faz a maioria das chamadas das funções presentes na API.Core.

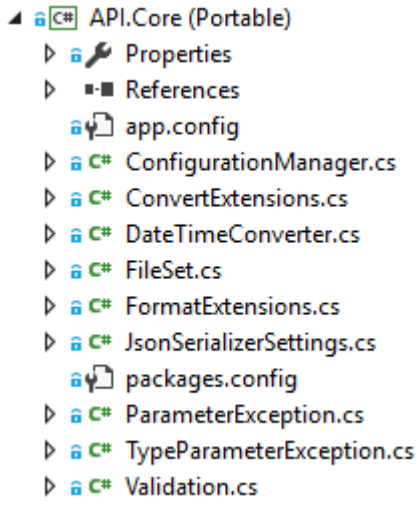


Figura 3.29 – Projeto API.Core
Fonte: Autor

A classe ConfigurationManager.cs é uma classe que tem o objetivo de atribuir strings, cada uma com um assessor obter (get) e um assessor definir (set). Caso ocorra um erro no sistema que venha a mensagem de ‘ConfigurationManager’, a mensagem de erro será representando uma forma errada de atribuição dos assessores. Por exemplo: ‘O valor está forado padrão’ ou ‘A verificação está com a data errada ou antiga.”.

Como mostrado na Figura 3.30, nesta classe foram atribuídas algumas variáveis strings, porém somente 2 possuem valores estáticos para o assessor Set.

```
{
    1 reference
    public class ConfigurationManager
    {
        0 references
        public static string MaxRequestLenght { get; set; }
        8 references
        public static class AppSettings
        {
            1 reference
            public static string AccountName { get; set; }
            1 reference
            public static string TokenAccess { get; set; }

            // Variável global que não pode ser alterada: 'Set' estático.
            0 references
            public static string BusinessRulesPath { get { return "/App_Data/"; } }

            // Variável global que não pode ser alterada: 'Set' estático.
            6 references
            public static string ApiUrl {get { return "http://controleshopping.azurewebsites.net"; }
                }
        }
    }
}
```

Figura 3.30 – Classe ConfigurationManager do projeto API.Core
Fonte: Autor

3.5.6 API.Data.DTO:

Toda a regra de negócio de BD é gerada na API principal, na pasta Models, no arquivo ‘ControleEntities.edmx’. Então todas as tabelas estão somente no projeto ControleShopping.API, ou seja, as classes de objetos: Produto, Verificacao, Usuario e outras. Porém, a necessidade do sistema é que o projeto do aplicativo também saiba lidar com os objetos do BD e também o que cada um significa.

Como o arquivo .edmx está no ControleShopping.API, somente ele sabe como funciona o BD, se não for criada uma classe para cada uma das tabelas no próprio aplicativo, o aplicativo não saberia como tratar os objetos. No caso do protótipo, foi desenvolvido um namespace com o nome API.Data.DTO somente para possuir classes que representassem cada uma das tabelas do BD, como mostrado na Figura 3.31.

Isto tudo é um padrão do Entity Framework. O API.Data.DTO é um projeto desenvolvido para poder comunicar pelo JSON. Entretando o JSON não consegue entender objetos nulos, porém o BD aceita objetos nulos. Se no BD tem algum campo que ele aceita

nulo, no .edmx vai ficar como nulo, mas no API.Data.DTO não é aceito. Então o API.Data.DTO serve também para poder criar objetos primitivos, ou seja, objetos não nulos, como uma String. Isto é aproveitado porque as PCLs não conseguem criar arquivos .edmx.

Resumindo, são classes cópias das tabelas, mas é uma cópia abstrata. Se possui um paramêtro de um campo da tabela que é opcional, no API.Data.DTO ele vai estar como normal, não vai estar como nulo.

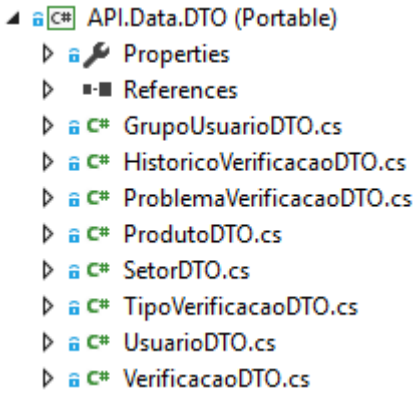


Figura 3.31 – Projeto API.Data.DTO
Fonte: Autor

Na Figura 3.32 pode-se observar como foi desenvolvida de forma simples a classe ProblemaVerificacaoDTO, que representa a tabela ProblemaVerificacao do BD. O qual possui variáveis representando cada coluna da tabela.

```
namespace API.Data.DTO
{
    11 references
    public class ProblemaVerificacaoDTO
    {
        0 references
        public int idProblemaVerificacao { get; set; }
        1 reference
        public int idVerificacao { get; set; }
        1 reference
        public string deProblemaVerificacao { get; set; }
        1 reference
        public byte[] imgProblema { get; set; }

        0 references
        public virtual VerificacaoDTO VerificacaoDTO { get; set; }
    }
}
```

Figura 3.32 – Classe ProblemaVerificacaoDTO do projeto API.Data.DTO
Fonte: Autor

3.5.7 API.Parameters:

Este namespace é reservado somente para atibuir os parâmetros de operações. Foi criada uma classe para cada uma das tabelas do BD como pode-se observar na Figura 3.33. Onde cada classe será responsável pelos parâmetros e operações de cada uma das tabelas.

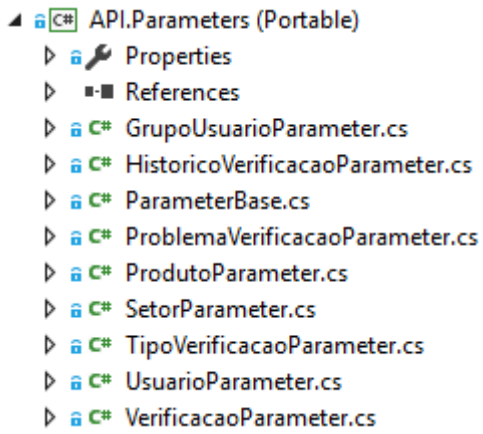


Figura 3.33 – Projeto API.Parameters
Fonte: Autor

A Figura 3.34 representa como são atribuídos os assessores Get e Set. Também mostrando quais são as operações que podem ser realizadas por esta classe; tais operações são igualadas a um número para deixar o aplicativo mais legível. Dessa forma não terá que trabalhar com város tipos de nomes de operações.

Como as configurações de ParameterBase serão herdadas para a GrupoUsuarioParameter, não é necessário que repita dentro da classe GrupoUsuarioParameter todas os parâmetros da ParameterBase novamente.

```
namespace API.Parameters
{
    // Criação dos paramêtros de GrupoUsuarioParameter, sempre irá herdar da ParameterBase
    12 references
    public class GrupoUsuarioParameter : ParameterBase
    {
        1 reference | 0 exceptions
        public int idGrupoUsuario { get; set; } // Objeto que pode tanto atribuir quanto setar o valor
        0 references | 0 exceptions
        public int idUsuario { get; set; }
        1 reference | 0 exceptions
        public string GrupoUsuarioJSON { get; set; }

        5 references
        public enum Operacao // Tipos de operações que pode realizar
        {
            Get = 1 ,
            GetList = 2,
            GetListByUsuario = 3,
            Create  = 15
        }
    }
```

Figura 3.34 – Classe GrupoUsuarioParameter do projeto API.Parameters
Fonte: Autor

3.5.8 API.Services:

Este namespace tem o objetivo de realizar os serviços para cada uma das tabelas do banco. Foi criada uma classe para cada uma das tabelas como pode-se observar na Figura 3.35. Onde cada classe será responsável por proceder requisições, utilizando interfaces pré-configuradas, classes da API.Data.Dto e parâmetros das classes API.Parameters.

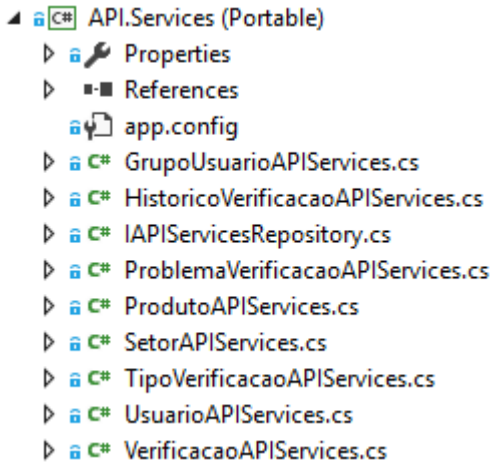


Figura 3.35 – Projeto API.Services
Fonte: Autor

A Figura 3.36 mostra como uma classe é definida e como são realizadas as suas principais funções. Na figura somente foi mostrada a função Delete, entretando neste serviço GrupoUsuarioAPIServices ainda exitem as funções de GetList, Post e Update.

```
namespace API.Services
{
    // A requisição do GrupoUsuario herda de HttpclientBase, utilizando a interface IAPIServiceRepository
    2 references
    public class GrupoUsuarioAPIServices : HttpClientBase, IAPIServicesRepository<GrupoUsuarioDTO, GrupoUsuarioParameter>
    {
        1 reference
        public GrupoUsuarioAPIServices
            (

                Operacao operacao, //Lista de operações que equivalem um número definido em API.Parameter.
                int? idGrupoUsuario = null, //Insere interrogação nos valores primitivos para atribuir valores nulos.
                int? idUsuario = null, //Insere interrogação nos valores primitivos para atribuir valores nulos.
                string includeProperties = "" //Insere a variável em formato string que representa um texto.
            ) : base("GrupoUsuario") // Atribui ao GrupoUsuarioAPIServices a base GrupoUguario. Utilizar caso queira herdar.
        {
            Parameters.idOperacao = (int)operacao; //Atribui na variável operacao os parâmetros de idOperacao
            Parameters.idGrupoUsuario = idGrupoUsuario;
            Parameters.idUsuario = idUsuario;
            Parameters.includeProperties = includeProperties;
        }

        //Sempre necessita ter todas (delete, GetList, post e update) essas funções abaixo quando está utilizando uma interface.
        8 references
        public bool Delete()
        {
            throw new NotImplementedException();
        }
```

Figura 3.36 – Parte da classe GrupoUsuarioAPIServices do projeto API.Services
Fonte: Autor

3.6 Quarta etapa: Configuração do dispositivo móvel

O Dispositivo Móvel é um smartphone que possui o sistema operacional Android. Após a postagem da primeira versão, ele ficará disponível na Google Play Store para ser baixado gratuitamente. Apesar de que o acesso a sua console principal será liberado apenas aqueles que tiverem um usuário e senha cadastrados em seu banco de dados. Caso não possua as credenciais, o cidadão que baixar o aplicativo poderá visualizar somente a tela inicial de login.

O nome do aplicativo é “Solução de vistoria predial”. Escolhido assim por ter como principal função a vistoria de todo um complexo predial.

Os pré-requisitos de instalação e funcionamento do aplicativo são:

- Sistema operacional Android com no mínimo a versão 4.1 (API Leve 16 – Jelly Bean);
- Possui qualquer uma das arquiteturas abaixo:
  - x86;
  - x86_64;
  - arm64-v8a;
  - armeabi-v7a.
- Conexão com a internet, podendo ser 4G ou Wifi;
- Câmera fotográfica.

A versão mínima do Android e a arquitetura do SO são definidas no Visual Studio. A versão mínima escolhida foi definida por ser uma versão criada em 2012 e já possuir smartphones capazes de processar o aplicativo.

O Wireless ou 4G tem o objetivo de estabelecer a comunicação sem fio do celular com a internet. As formas que podem ser realizadas estas conexões serão abordadas no Cápitulo 4. O qual abordará todo o conjunto dos testes de conexão entre o aplicativo e a nuvem do azure, com o dispositivo móvel estando conectado a internet utilizando rede 3G, 4G e Wi-Fi. Também sendo realizados teste de perda de conexão e baixa taxa de comunicação.

A necessidade de uma câmera no smartphone vem da função de QR Code, ela é feita através da análise que a câmera fotográfica realiza em uma imagem que possua o código de barras.

O aplicativo também pode ser adicionado em um smartphone diretamente pelo computador, mas para isso é necessário que tenha todo o conjunto de arquivos gerados pelo programador. Após instalar o aplicativo, podendo ser de qualquer uma das duas formas, pela Google Play Store ou diretamente, basta inicia-lo e manter a conexão com internet estável.

3.7 Quinta etapa: Configuração do Servidor de E-mail

O ambiente do servidor de e-mail será hospedado em um dos 7 componentes do sistema, no Serviço do Aplicativo. Por tal serviço ficar localizado na nuvem, o envio do e-mail será feito diretamente do próprio serviço, onde todas as configurações de firewall e segurança serão gerenciadas pela Azure. Foi utilizado o SMTP da Google com o e-mail fillipemoura2@gmail.com.

Abaixo segue o código utilizado para configurar o servidor de e-mail na classe JobRelatorio.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Threading;
using System.Net.Mail;
using System.Net;
using ControleShopping.API.Models;
using System.Text;

namespace ControleShopping.API
{
    public class JobRelatorio
    {
        private ControleShoppingEntities db = new ControleShoppingEntities();
        private System.Threading.Timer timer;
        private bool icJaFoi;
        public void SetUpTimer(TimeSpan alertTime)
        {
            DateTime current = DateTime.Now;
            TimeSpan timeToGo = alertTime - current.TimeOfDay;

            EnviarRelatorio();
        }
        private void EnviarRelatorio()
        {
            try
            {
                var client = new SmtpClient("smtp.gmail.com", 587)
                {
                    UseDefaultCredentials = false,
```

```
                    Credentials = new NetworkCredential("Fillipemoura2@gmail.com",
"Fillipe@)20"),
                    EnableSsl = true,
                    DeliveryMethod = SmtpDeliveryMethod.Network,
                    Port = 587
                };
                MailMessage message = new MailMessage();
                message.From = new MailAddress("Fillipemoura2@gmail.com");
                message.To.Add("Fillipemoura2@gmail.com");
                message.Subject = "Relatório do dia " +
DateTime.Now.ToString("dd/MM/yy");
                message.Body = GetTextRelatorio();
                message.IsBodyHtml = true;
                client.Send(message);
            }catch(SmtpException ex)
            {
                throw ex;
            }
            catch(Exception ex)
            {
                throw ex;
            }
        }
        private string GetTextRelatorio()
        {
            var ontem = DateTime.Now.AddDays(-1);
            var historico = db.HistoricoVerificacao.Where(histo => histo.dtVerificacao
>= ontem);
            var texto = new StringBuilder();
            texto.Append("<table>");
            texto.Append("<tr>");
            texto.Append("<th>Produto</th>");
            texto.Append("<th>Verificação</th>");
            texto.Append("<th>Data da Verificação</th>");
            texto.Append("<th>Existe problema</th>");
            texto.Append("<th>Nome do Usuário</th>");
            texto.Append("</tr>");
            foreach (var verificacao in historico)
            {
              texto.Append("<tr>");
              texto.Append("<td>" + verificacao.Verificacao.Produto.deProduto + "</td>");
              texto.Append("<td>" + verificacao.Verificacao.deVerificacao + "</td>");
              texto.Append("<td>" +
verificacao.dtVerificacao.GetValueOrDefault().ToString("dd/MM/yyyy") + "</td>");
               texto.Append("<td>" +
(verificacao.Verificacao.ProblemaVerificacao.Any(problema =>
problema.dtProblemaVerificacao >= ontem) ? "Sim" : "Não" ) + "</td>");
              texto.Append("<td>" + verificacao.Usuario.nmUsuario + "</td>");
              texto.Append("</tr>");
            }
            texto.Append("</table>");
            return texto.ToString();
        }
    }
}
```

Com isto, encerram-se as etapas na implementação da solução, estando pronta para aplicação. Antes, serão feitos alguns testes e demonstração de funcionamento com resultados, como será visto no próximo capítulo.

CAPÍTULO 4 – TESTES E RESULTADOS

O capítulo 4 irá descrever e explicar quais foram os testes realizados no aplicativo, tanto como demonstrar quais foram os resultados encontrados. Também será explicitado quais foram as técnicas utilizadas para evitar falhas e erros no sistema, buscando manter o sistema sempre em funcionamento. Ao fim deste capítulo serão apresentadas as dificuldades encontradas no decorrer do todo o processo de desenvolvimento do aplicativo.

4.1 Emulador Intel HAXM

Para a realização dos primeiros testes de utilização do aplicativo foi utilizado o emulador Intel® Hardware Accelerated Execution Manager (Intel® HAXM). O Intel® HAXM é um hypervisor que utiliza a tecnologia de virtualização da Intel para rodar aplicativos Android em uma máquina física. O software está disponível para download no seguinte site <https://software.intel.com/en-us/android/articles/intel-hardware-accelerated-execution-manager>.

Nele foi possível rodar as primeiras versões do aplicativo, de uma forma fácil e transparente. Em razão de poder configurar todo um smartphone da maneira que necessita. Por exemplo, qual plataforma deseja emular, quais sensores deseja (GPS, Acelerômetro, giroscópio e sensor de próximidade), qual CPU, capacidade de memória RAM, arquitetura do sistema, resolução da tela e entre outros. Ainda existe a opção de apenas utilizar alguns templates já criados, como do Nexus 5, Nexus 9, Nexus 10 e outros.

Após realizada a instalação no computador, a inicialização do HAXM é feita pelo próprio Visual Studio, basta seguir este caminho na barra inicial do Visual Studio: Tools -> Android -> Android Emulator Manager. Com a interface principal do HAXM aberta, selecione a opção “Create...” para realizar a criação de um smartphone virtual. Será aberta a janela “Edit Android Virtual Device (AVD)”, nela deverão ser preenchidos os campos da forma que desejas. Para a instalação do aplicativo foi configurado um emulador conforme a Figura 4.1.

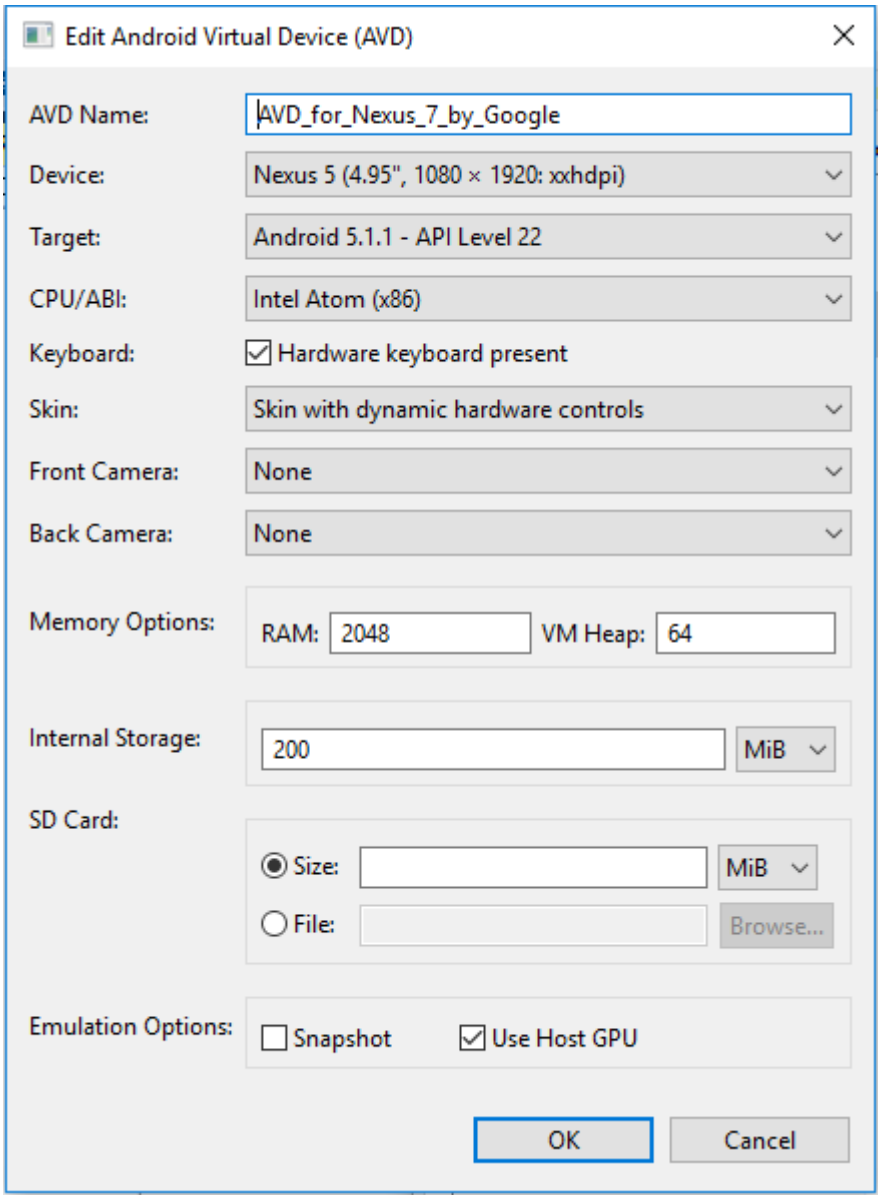


Figura 4.1 – Criação do smartphone virtual no emulador
Fonte: Autor

Para a realização dos testes no emulador foram utilizadas configurações de um smartphone de acordo com a Figura 4.1. Foram feitos inúmeros testes para cada ação criada e desenvolvida no código. Entretanto os testes realizados de utilização após o aplicativo estar totalmente pronto foram realizados em um Galaxy S3 gt-i9300. O resultado de cada um dos testes foi somado e dividido por 3. Por exemplo: 1º teste (7,7 segundos) + 2º teste (7,4 segundos) + 3º teste (9,3 segundos) / 3 = 8,13 segundos de média.

Os testes representaram resultados em segundos, como mostra a Tabela 4.1.

Tabela 4-1 – Tabela de testes

	Vistoria com 3G (Segundos)	Vistoria com 4G (Segundos)	Vistoria com WiFi (Segundos)	Tempo MÉDIO (Segundos)
Iniciar Aplicativo (variação de acordo com smartphone utilizado)	5,96	5,96	5,96	5,96
Realizar Login	12,73	8,49	5,96	9,06
Abertura de Lista de Produtos	5,88	3,47	1,83	3,73
Abertura de Lista de Atividades	6,33	4,02	2,52	4,29
Reconhecer QR Code e atribuir ao BD	3,74	3,18	2,37	3,10
Enviar Status do Objeto para BD	13,21	11,95	7,85	11,00
Enviar foto de inconformidade para BD	128,43	87,51	48,19	88,04

Fonte: Autor

4.2 Formas de evitar Falhas e Erros: Solução para dificuldades de Transparência em Sistemas Distribuídos

Existem alguns desafios que devem ser sanados e um dos maiores de uma aplicação mobile com um sistema remoto (como um servidor na “cloud”), é a sincronização do aplicativo com esse sistema remoto. Ou seja, é a comunicação do aplicativo com as informações do sistema. Essas comunicações podem ser separadas em dois tipos: “Uploading” e “Downloading”. Serão tratadas as dificuldades de transparência de localização e falha.

O grande desafio é o de ter que se levar em conta a disponibilidade da rede, a informação mais recente, a interface do usuário, entre outros. Por isso, a proposta é aplicar um padrão para a atualização das informações e o recebimento das mesmas. Assim, evitando que o usuário fique com a sua interface travada e permitir também um evento de sincronização de informação sem a necessidade da interface do usuário.

O sistema possui um JOB do aplicativo que fica verificando o banco de dados local, caso tenha algo de novo, o aplicativo estabelece comunicação com as APIs do aplicativo web no Azure e envia os dados. Caso venha a ocorrer uma falha na comunicação entre o aplicativo do smartphone e o aplicativo web na Azure, o sistema local cria um banco de dados (SQLite), a partir daí que realiza a sincronização offline explicada no referencial teórico. Com a comunicação estabelecida entre o celular e a nuvem novamente, os dados são recebidos pelas APIs do sistema web, o qual é responsável por armazenar no banco de dados SQL.

A Sincronização Offline é utilizada para que o aplicativo se mantenha disponível mesmo que esteja ocorrendo problemas na conexão de rede. Caso o aplicativo esteja em modo Offline, o usuário que está logado no aplicativo ainda poderá modificar e criar novos dados. Uma vez que sejam alterados ou criados dados, eles serão armazanados em um repositório local. Assim que o dispositivo móvel estabelecer conexão com a internet e o aplicativo fique online outra vez, ele fará a sincronização dos dados que estão armazenados localmente com a base de dados armazanada no Azure.

4.3 Transparência de Localização:

A Transparência de Localização é responsável por ocultar o local que se encontra o recurso utilizado pelo sistema. Desta forma o usuário não saberá dizer qual é o computador ou rede que o sistema está sendo processado, podendo estar no próprio smartphone, em um centro de processamento de dados no Vale do Silício, ser acessado via um endereço IP ou até mesmo por um link como http://www.vistoriapredial.uniceub.br.

A dificuldade de transparência de localização foi resolvida com a utilização da nuvem Azure e com o processamento local do sistema. Da forma que foi desenvolvido o sistema, ele está sendo processado localmente, por não ter o código muito extenso e não utilizar de muito processamento do smartphone. Porém, a sua base de dados está armazenada na Azure, dessa forma o smartphone necessita estabelecer uma conexão quase constante com a nuvem para manter a sua base de dados atualizada.

O aplicativo é responsável por todo o sistema, mas para apresentar o que cada usuário deve vistoriar, popular o banco de dados com novas informações ou alterar o que já estava salvo, será necessário que o aplicativo trabalhe em conjunto com o aplicativo web que está sendo processado na Azure. Mas para o cliente, independe de onde estão os dados, ele só quer que suas informações estejam salvas em um local seguro. Também independe de onde está sendo processado, ele só quer que o aplicativo funcione normalmente sem travar em momento algum. Fatos que foram determinantes na escolhe desta arquitetura.

Isto provê mais performance ao usuário final, aparentando estar trabalhando com dados locais, mas na verdade está sincronizado e trabalhando com dados que estão armazenados em um servidor físico no centro de processamento de dados da Azure no Leste dos Estados.

Foi possível observar como o desafio de transparência de localização deve sempre ser abordado, possibilitando que o sistema se torne o mais simples possível para aquele que o utiliza, buscando tornar o usuário final mais produtivo e focado na atividade fim.

4.4 Transparência de falha:

Segundo Leslie Lamport “Você sabe que tem um sistema distribuído quando a falha de um computador do qual você nunca ouviu falar faz com que você pare completamente de trabalhar." Vendo essa citação podemos conferir o quão complicado é mascarar uma falha ao usuário. Porém mesmo um sistema distribuído sendo construído a partir de um conjunto de outros componentes, é possível que os serviços construídos sejam confiáveis a partir de componentes que exibem falhas, como por exemplo o uso de vários servidores que contêm réplicas de dados para continuar a fornecer o serviço sem que haja perdas.

A Transparência de Falha é responsável por ocultar todas as falhas e a recuperação de um recurso sem que o usuário final perceba que isso ocorreu no sistema. O que foi tratado neste trabalho foi uma falha de comunicação entre o smartphone e o serviço de aplicativo na nuvem. De maneira que mesmo o aplicativo perdendo a conexão com o banco de dados armazenado no Azure, o usuário continue a trabalhar normalmente sem a interrupção do aplicativo.

Tal solução foi definida para que o usuário acreditasse que todo o sistema estivesse sendo processado no smartphone, entretanto por questões de segurança da informação, caso o celular viesse a quebrar, apresentar uma falha ou algo similar, a grande massa de dados estaria salva em um local seguro. Independentemente de onde estivesse sendo processada a informação, qualquer escrita ou alteração dos dados no aplicativo do celular é feita em um banco de dados SQLite local. Deste banco de dados que será realizada a leitura dos dados ou até mesmo salvo novas informações por APIs do aplicativo.

Ao atualizar o registro de uma verificação, é feito um *call-back* (retorno) caso ocorra um erro na hora de fazer a atualização. Para que isso ocorra é encapsulado a função de update do banco dentro do método try, para que caso aconteça um erro, é acionado a função call-back (catch), essa que é responsável em salvar tal verificação em um banco de dados local (SQLite) e o usuário continue a usar o aplicativo sem que veja o erro que aconteceu (transparência de falha).

```
try
{
    var verifica = await new API.Services.VerificacaoAPIServices(API.Parameters.VerificacaoParameter.Operacao.Update).Update(new VerificacaoParameter()
    {
        VerificacaoJSON = JsonConvert.SerializeObject(_verificacao),
        idOperacao = (int)VerificacaoParameter.Operacao.Update
    });
}
catch
{
    string folder = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
    folder = System.IO.Path.Combine(folder, "verificacoes.db");
    var  retornoTable = await new CreateSQLiteTable().createDatabase(folder);
    if(retornoTable == "Database created")
    {
        var criada = await new CreateSQLiteTable().insertUpdateDataVerificacao(_verificacao, folder);
    }
}
```

Figura 4.2 – Tentativa de inserção
Fonte: Autor

Quando o aplicativo inicia a classe responsável por mostrar para o usuário o menu de opções, é inicializada e colocada em fila (agendado) uma tarefa que será responsável por verificar no banco de dados local se existe algum dado gravado. Essa tarefa será executada de 10 em 10s caso exista qualquer conexão com a internet e é mostrado uma mensagem para o usuário que foi startado o Job.

```
serviceComponent = new ComponentName(this, Java.Lang.Class.FromType(typeof(TaskVerificarSQLite)));

var job = new JobInfo.Builder(12, serviceComponent);
job.SetMinimumLatency(10000);
job.SetRequiredNetworkType(NetworkType.Any);
job.SetRequiresDeviceIdle(true);
_taskService.ScheduleJob(job.Build());
Toast.MakeText(this, "Job Startado.",
        ToastLength.Short).Show();
```

Figura 4.3 – Criação do JOB
Fonte: Autor

Repare que na Figura 4.4 existe uma referência da classe TaskVerificarSSQLite, ela herda da classe JobService que é a classe que contém todos as funções abstratos que devem ser sobrepostos. Essas funções sobrepostas serão as responsáveis pelas ações da tarefa criada para verificar o banco de dados local. Assim a função sobreposta (override) OnStarJob será acionada quando o Job entrar em execução pela pilha de tarefas do OS do Android.

```
[Service(Exported =true,Permission ="android.permission.BIND_JOB_SERVICE")]

public class TaskVerificarSQLite : JobService
{
    private static string Tag = "SyncService";
    ListaAtividades owner;
    private InformacoesSistema infoSistema;
    /** Send job to the JobScheduler. */

    public void ScheduleJob(JobInfo t)
    {
        var tm = (JobSchedulerType)GetSystemService(Context.JobSchedulerService);
        var status = tm.Schedule(t);
        Log.Info(Tag, "Scheduling job: " + (status == JobSchedulerType.ResultSuccess ? "Success" : "Failure"));
    }


    public override bool OnStartJob(JobParameters args)
    {
        VerificaDadosSQLite(infoSistema.pathSQLite);
        Log.Info(Tag, "on start job: " + args.JobId);
        return true;
    }
```

Figura 4.4 – Criação da classe TaskVerificarSQLite
Fonte: Autor

Ainda sobre a transparência de falha, deve-se enobrecer a sua utilidade, pois é ela que torna possível o cliente acreditar que o sistema está livre de falhas e erros, mesmo que eles estejam acontecendo em backplane.

4.5 Dificuldades encontradas

As dificuldades encontradas foram um grande atraso no desenvolvimento do sistema, levando em consideração que não foi encontrado nenhum livro para troubleshooting de aplicativo desenvolvido em C# no Xamarin. A maioria das resoluções dos problemas foi baseada em pesquisas no site oficial da Microsoft para desenvolvedores, o <https://msdn.microsoft.com/pt-br/default.aspx>. O local onde foi grande fonte de pesquisa para desenvolvimento deste sistema.

1. A primeira dificuldade encontrada foi em criar um servidor local e ter de gerenciar acessos externos. Para se desenvolver um sevidor DMZ é necessário grande conhecimento na área de segurança, caso contrário, o sistema teria falhas de seguranças.

a. Resolução: Mudar infraestrutura para Azure. Inúmeros benefícios, já citados no Item 3.3.

2. Dificuldades no desenvolvimento do código:
    a. Criação da estrutura do sistema.
        i. Resolução: Seguir exemplos com a utilização de Portable Class Library demonstrados no site da Microsoft. Também realizar a criação de classes apartir de templates do Visual Studio.

    b. Criação de classes portáveis (Portable Class Library).
        i. Resolução: Não criar em modelo tradicional e depois transformar em portable, também foram encontradas dificuldades nesse processo. Desenvolver classes Portable desde o princípio.

    c. Criação de uma PCL para realizar a comunicação HTTP. Onde foram encontrados vários erros de difícil resolução.
        i. Resolução: Realizar toda essa comunicação dentro do código do próprio aplicativo.

    d. Criação do QR Code.
        i. Resolução: Utilização da biblioteca ZXing.Net.Mobile para a criação do QR Code.

    e. Envio de imagem para a API no Serviço de aplicativo..
        i. Resolução: Mudar o formato da variável e permitir o recebimento de uma quantidade grande de Bytes pela API.

3. Dificuldades com o emulador HAXM.
    a. O emulador simula o processador da Intel. Entretando, o computador que foi realizado os primeiros testes era AMD.
        i. Resolução: Utilizar um PC com processador Intel e também testar em smartphone.

4.6 Avaliação da solução

O projeto entregou o que lhe foi proposto na ideia principal. Portanto as soluções utilizadas que foram estabelecidas como as melhores opções para realização deste, se justificaram entregando um aplicativo funcional e seguro.

Não se esperava tamanha dificuldade em estabelecer a comunicação entre o aplicativo e o sistema Web, onde foram encontrados inúmeros erros e vários momentos de pesquisa para solucionar tais problemas. Em contrapartida, a utilização da linguagem C# para o desenvolvimento do aplicativo e do sistema web facilitou o desenvolvimento de classes e estrutura de API.

A tentativa de desenvolvimento de um código que criasse um QR Code foi basicamente tempo gasto de forma errônea. Pois havia mais de uma biblioteca com o código fonte pronto em sites confiáveis na internet, como o próprio site da Microsoft. Onde foi encontrada a estrutura utilizada no protótipo.

O custo em reais para o desenvolvimento do protótipo foi zero. A utilização de uma conta para iniciantes no Azure permite que o usuário goze de todas as funcionalidades sem ter nenhum custo. Caso tornasse realidade a utilização do aplicativo em ambientes funcionais, seria necessário a aquisição de licenças na nuvem da Azure ou aquisição de equipamentos físicos, como abordado o valor de investimento no item 3.3. O único fator gasto no desenvolvimentodo protótipo foi o tempo de seu autor.

A recomendação de uso do aplicativo ficou bem abrangente. Podendo ser utilizado em shoppings, edifícios comercias, aeroportos, faculdades, hipermercados e outros. Para a utilização em outros meios além do shopping, basta a alteração de nomes que especificam localidades, ou seja, algumas alterações no banco de dados. Para a realização destas alterações é necessário apenas os requisitos de vistoria de cada empreendimento, desta forma o autor necessitaria realizar *updates* (atualizações) em algumas tabelas do banco. Ação simples que não gastaria muito tempo.

CAPÍTULO 5 – CONCLUSÃO

Todas essas questões, devidamente ponderadas, levantam dúvidas de como se evitar um acidente, pois sempre haverá brechas para erros. Mas a maioria dessas dúvidas pode ser respondida com uma simples resposta: Todos devemos ser responsáveis em manter a segurança do local onde estamos, nunca sendo conivente com a falta de cuidado de ninguém.

A utilização do sistema de vistoria pode ser a saída para evitar inúmeros acidentes que ocorrem no dia-a-dia. Servindo como lembrete para os esquecidos e um grande aliado dos gestores de condomínios. As pessoas só compreendem as reais consequências de uma emergência ou acidente, quando passam por tais problemas, antes disso ficam levianamente distraídos não dando a real importância para vistorias prediais.

O sistema teve um grau de complexidade muito maior do que o esperado, em razão da utilização das PCLs e possuir um *back-end* na nuvem. Foi necessário aprender uma vasta gama de novos conceitos, formas de transferência de arquivos e maneiras de programar a comunicação. Caso ele fosse desenvolvido em uma linguagem de programação como o Java e a IDE de escrita fosse o Android Studio, o sistema teria menos linhas de código, nenhuma PCL e baixa complexidade. Entretanto, dessa forma o conceito de todo o projeto não faria sentido e para programar o aplicativo para o sistema operacional iOS, seria necessário que começasse o desenvolvimento de um novo código desde o princípio.

Os testes realizados com o uso do QR Code foram necessários para aprimorar o código, onde foram encontradas diversas vezes erro ao gerar o QR Code e atribuí-lo a um produto. Entretanto, ao fim, funcionou adequadamente como pode-se observar nos testes realizados. Com isso pode-se confirmar a execução de forma correta do aplicativo.

O desenvolvimento do trabalho deixou um vasto campo de pesquisa, trazendo um interesse por aplicar no sistema o desafio de sistemas distribuídos de Heterogeneidade. Principalmente por estar trabalhando com o aplicativo em Android e parte do processamento estar em nuvem. Também aplicar o desafio de Segurança, que foi resolvido exclusivamente por ter credenciais Azure.

Com o desenvolver do sistema, foram surgindo várias ideias de acrescentar funcionalidades ao projeto. Sendo a maioria delas, necessidades reais para empreendimentos como um Shopping Center. Sugestões para trabalhos futuros:

- Acrescentar no sistema todas as vistorias das Tabelas 3.1 e 3.2. O qual entregaria uma vistoria completa de todo o empreendimento.

- Estudar forma de melhorar a performance da atividade de enviar a foto de uma inconformidade para a Nuvem.

- Realizar estudo de mercado para a comercialização do aplicativo.

- Desenvolvimento do sistema para smartphones que utilizam o sistema operacional iOS. Seria aproveitado grande parte do código do sistema, melhor dizendo, todas as Portable Class Library.

- Migração da parte do sistema localizado na nuvem Azure para um servidor local, onde deveria ter no mínimo a mesma confidencialidade, integridade e disponibilidade, resiliência, confiabilidade e performance.

- Desenvolver parte do sistema para realizar o controle de abertura e fechamento de todas as portas do empreendimento, como por exemplo as portas de todas as lojas de um shopping center. Entregando os hórarios de abertura e fechamento das lojas em um relatório enviado diariamente para seus proprietários, o qual utilizaria para gerenciar melhor seus funcionários.

- Criação de um ‘Botão de emergência’. Onde cada loja de um shopping center possuíria e seria acionado em caso de emergências. Também podendo ser gerenciado pelo sistema.

- Criação de um sistema para controle da garagem de um shopping center. Onde seriam demonstrados os andares que possuem vagas livres, quantidade de vagas livres em cada andar, localidade da vaga livre, luz de indicação da vaga livre e relatório de entrada e saída de carros.

- Menu do aplicativo onde os clientes poderiam pagar a saída do estacionamento.

REFERÊNCIAS

1. Antão, Igor; Carvalho, Rosa; Rabelo, Ricardo; Santos, Tina. SISTEMA DE CONTROLE DE ACESSO VEICULAR GERENCIADO POR QR CODE, ano 2013, 4 p.

2. Atlas, Equipe. Segurança e Medicina do Trabalho, 77ª edição. Atlas, 02/2016.

3. Cardella, Benedito. Segurança no Trabalho e Prevenção de Acidentes. Atlas, 08/1999. VitalSource Bookshelf Online.

4. COULOURIS, George, DOLLIMORE, Jean, KINDBERG, Tim, BLAIR, Gordon. Sistemas Distribuídos: Conceitos e Projeto, 5th edição. Bookman, 08/2013. VitalSource Bookshelf Online.

5. DEITEL, Paul, DEITEL, Harvey, WALD, Alexander. Android 6 para Programadores: Uma Abordagem Baseada em Aplicativos, 3rd edição. Bookman, 01/01/2016.

6. MANZANO, José Augusto G. Programação de Computadores com C#. Érica, 06/2014. 144 p.

7. MONK, Simon. Programação com Arduino: Começando com Sketches - Série Tekne. AMGH, 03/2013. VitalSource Bookshelf Online.

8. Vieira, Liliana; Coutinho, Clara. Artigo MOBILE LEARNING: PERSPETIVANDO O POTENCIAL DOS CÓDIGOS QR NA EDUCAÇÃO, ano 2013, 18 p.

9. SHARP, John. Microsoft Visual C# 2013 - Série Passo a Passo. Bookman, Janeiro de 2015.

10. Site de explicação do que se trata os desafios de sistemas distribuídos <http://passeiucp.com.br/arquivos/33a29de29ac127796ccf471c84931eca.pdf> . Acesso em 19 de outubro de 2016.

11. Site oficial da FabFORCE, proprietária do DBDesiner, Disponível em <http://fabforce.net/dbdesigner4/>. Acesso em 15 de outubro de 2016.

12. Site oficial da Microsoft Azure, Disponível em <https://azure.microsoft.com/pt-br/documentation/articles/app-service-mobile-offline-data-sync/ >. Acesso em 9 de outubro de 2016.

13. Site oficial da Microsoft Azure, sobre Web sites .Net. Disponível em https://azure.microsoft.com/pt-br/documentation/articles/web-sites-dotnet-get-started/>. Acesso em 28 de Julho.

14. Site oficial da Microsoft para desenvolvedores explicando o funcionamento de PCLs, Disponível em: https://msdn.microsoft.com/pt-br/library/gg597391(v=vs.110).aspx. Acesso em 15 de Setembro.

15. Site oficial da Microsoft para desenvolvedores, Disponível em: https://msdn.microsoft.com/en-us/library/. Acesso em 03 de setembro de 2016.

16. Site oficial do .NET Framework, Disponível em: <https://docs.microsoft.com/pt-br/dotnet/>. Acesso em 11 de setembro de 2016.

17. Site oficial do Arduino, Disponível em http://forum.arduino.cc/index.php?topic=279996>. Acesso em 10 de agosto de 2016.

18. Site oficial do Corpo de Bombeiros Militar do Distrito Federal, Disponível em <https://www.cbm.df.gov.br/2012-11-12-17-41-39/2012-11-12-18-51-53?view=category&id=407>. Acesso em 04 de setembro de 2016.

19. Site oficial do Departamento de trabalho dos Estados Unidos da América, Disponível em <http://www.bls.gov/iif/cfoi_revised14.html>. Acesso em 03 de setembro de 2016.

20. Site oficial do QR CODE, Disponível em: <http://www.QR_Code.com/en>. Acesso em 5 de Agosto de 2016.

21. Site oficial do SQL Server Management Studio, propriedade do SQL Server, Disponível em <https://azure.microsoft.com/pt-br/documentation/articles/sql-database-connect-query-ssms/>. Acesso em 28 de outubro de 2016.

22. Site oficial do SQL Server Management Studio (SSMS), Disponível em: <https://msdn.microsoft.com/pt-br/library/mt238290.aspx>. Acesso em 27 de Setembro de 2016.

23. Site oficial do Xamarin, Disponível em: <https://developer.xamarin.com/>. Acesso em 8 de Junho de 2016.

24. Site oficial do Xamarin sobre Cross-Plataform, Disponível em: https://developer.xamarin.com/guides/cross-platform/application_fundamentals/>. Acesso em 28 de Junho.

25. Weir, Michel. QR Codes & Mobile Marketing for the Small Bussiness Owner. 2010, 49 p.

26. Waters, Joe. QR Codes for Dummies. 2012 por John Wiley & Sons. 107 p.

APÊNDICE A – CÓDIGO DO SISTEMA DE VISTORIAS

A Figura 3.20 mostra a disposição dos projetos da solução ControleShopping2.0. O resto do código fonte que não está no Apêndice A são estruturas próprias do próprio Visual Studio com o Xamarin para trabalhar com Android, Azure, Web Aplication, SQL Server 2012 e API.

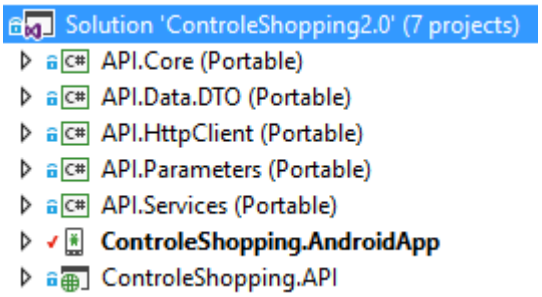


Figura 3.20 – Projetos do código
Fonte: Autor

- Projeto API.Core:
    - Classe ConfigurationManager;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Core
{
   public static class ConfigurationManager
   {
      public static string MaxRequestLenght { get; set; }
      public static class AppSettings
      {
         public static string AccountName { get; set; }
         public static string TokenAccess { get; set; }
         public static string BusinessRulesPath { get { return "/App_Data/"; } }
         public static string ApiUrl {get { return "http://controleshopping.azurewebsites.net"; } //
variavel global que não pode ser alterada. no set
            }
      }
   }
}
```

- Projeto API.Core:
    - Classe ConvertExtensions;

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using System.Linq;

namespace API.Core
{
    public static class ConvertExtensions
    {
        public static double ToDouble(this object value)
        {
            if (value == null)
                return 0.0;
            Double res;
            Double.TryParse(value.ToString(), out res);
            return res;
        }

        public static decimal ToDecimal(this object value)
        {
            if (value == null)
                return 0;
            Decimal res;
            Decimal.TryParse(value.ToString(), out res);
            return res;
        }

        public static byte ToByte(this object value)
        {
            byte res = 0;
            if (value != null)
                byte.TryParse(value.ToString(), out res);
            return res;
        }

        public static int ToInt32(this object value)
        {
            int res = 0;
            if (value != null)
                int.TryParse(value.ToString(), out res);
            return res;
        }

        public static long ToInt64(this object value)
        {
            long res = 0;
            if (value != null)
                long.TryParse(value.ToString(), out res);
            return res;
        }

        public static DateTime? ToDate(this object value)
        {
            try
            {
                return DateTime.Parse(value.ToString());
            }
            catch
            {
                return null;
            }
```

```
        }

        public static bool ToBoolean(this object value)
        {
            bool result = false;
            if (value != null)
            {
                bool.TryParse(value.ToString(), out result);
                return result;
            }
            return false;

        }

        public static IEnumerable<IEnumerable<T>> Split<T>(this T[] array, int size)
        {
            for (var i = 0; i < (float)array.Length / size; i++)
            {
                yield return array.Skip(i * size).Take(size);
            }
        }
    }
}
```

- Projeto API.Core:
    - Classe DataTimeConvert;

```
using Newtonsoft.Json;
using Newtonsoft.Json.Converters;
using System;
using System.Globalization;
using System.Net;

namespace API.Core
{
    public class DateTimeConverter : DateTimeConverterBase
    {
        public override object ReadJson(JsonReader reader, Type objectType, object existingValue,
JsonSerializer serializer)
        {
            string value = reader.Value.ToString();

            /* Verifica se a data foi passada com Html Encode e valida o formato.
             * Exemplo: 21%2F09%2F1977
             */
            if (value.IndexOf(@"%2F") > -1)
            {
                value = WebUtility.UrlDecode(value);
            }

            DateTime date;

            try
            {
                //DateTime.TryParse(value, CultureInfo.InvariantCulture, DateTimeStyles.AssumeLocal,
out date);
```

```
            date = DateTime.Parse(value,CultureInfo.CurrentCulture,
DateTimeStyles.AssumeLocal);
         }
         catch (Exception)
         {
            date = new DateTime();
         }

         return date;
      }

      public override void WriteJson(JsonWriter writer, object value, JsonSerializer serializer)
      {
         writer.WriteValue(((DateTime)value).ToString("dd/MM/yyyy HH:mm:ss",
CultureInfo.InvariantCulture));
      }
   }

}
```

- Projeto API.Core:
    - Classe FileSet;

```
using System.Collections.Generic;
using System.Dynamic;
using Newtonsoft.Json;

namespace API.Core
{
   public class FileSet
   {
      private dynamic _parameters = new ExpandoObject();

      public string Name { get; set; }

      public dynamic Parameters
      {
         get { return _parameters; }
         set { _parameters = value; }
      }

      [JsonIgnore]
      public IList<File> Files { get; set; }
   }

   public class File
   {
      public string Name { get; set; }

      public string MimeType { get; set; }

      public byte[] Content { get; set; }
   }
}
```

- Projeto API.Core:
    - Classe FormatExtensions;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Core
{
   public static class FormatExtensions
   {
      /// <summary>
      /// Format date in dd/MM/yyyy (pt-BR)
      /// </summary>
      /// <param name="value">Date</param>
      /// <returns>Formated date</returns>
      public static string FormatPT(this DateTime value)
      {
         return ((DateTime)value).ToString("dd/MM/yyyy");
      }

      public static DateTime ParseDateTime(this string date)
      {
         string dayOfWeek = date.Substring(0, 3).Trim();
         string month = date.Substring(4, 3).Trim();
         string dayInMonth = date.Substring(8, 2).Trim();
         string time = date.Substring(11, 9).Trim();
         string offset = date.Substring(20, 5).Trim();
         string year = date.Substring(25, 5).Trim();
         string dateTime = string.Format("{0}-{1}-{2} {3}", dayInMonth, month, year, time);
         DateTime ret = DateTime.Parse(dateTime);
         return ret;
      }

      public static string Cut(this string value, int maxLenght)
      {
         if (value == null)
            return null;

         if (value.Length > maxLenght)
            return value.Substring(0, maxLenght - 1);
         else
            return value;
      }

      public static string StripTagsCharArray(this string source)
      {
         char[] array = new char[source.Length];
         int arrayIndex = 0;
         bool inside = false;

         for (int i = 0; i < source.Length; i++)
         {
            char let = source[i];
            if (let == '<')
            {
```

```
                inside = true;
                continue;
            }
            if (let == '>')
            {
                inside = false;
                continue;
            }
            if (!inside)
            {
                array[arrayIndex] = let;
                arrayIndex++;
            }
        }
        return new string(array, 0, arrayIndex);
    }
  }
}
```

- Projeto API.Core:
    - Classe JsonSettings;

```
using Newtonsoft.Json;

namespace API.Core
{
  public class JsonSettings
  {
    public static JsonSerializerSettings GetSerializerSettings()
    {
      var settings = new JsonSerializerSettings();

      settings.Formatting = Formatting.Indented;
      settings.DateTimeZoneHandling = DateTimeZoneHandling.Utc;

      settings.Converters.Add(new DateTimeConverter());

      return settings;
    }
  }
}
```

- Projeto API.Core:
    - Classe ParameterException;

```
using System;

namespace API.Core
```

```
{
   public class ParameterException : Exception
   {
      public ParameterException(string parameterName, object parameterValue,
TypeParameterException type)
         : base(FormatMessage(parameterName, parameterValue, type))
      {
      }

      public ParameterException(string parameterName, object parameterValue)
         : base(FormatMessage(parameterName, parameterValue,
TypeParameterException.Invalid))
      {
      }

      public ParameterException(string parameterName, TypeParameterException type)
         : base(FormatMessage(parameterName, null, type))
      {
      }

      public ParameterException(string message)
         : base(message)
      {
      }

      private static string FormatMessage(string parameterName, object parameterValue,
TypeParameterException type)
      {
         switch (type)
         {
            case TypeParameterException.IsNullOrEmpty:
               return string.Concat("O parâmetro \"{0}\" não pode ser {1}.", parameterName,
parameterValue == null ? "nulo" : "vazio");
            case TypeParameterException.IsNull:
               return string.Concat("O parâmetro \"{0}\" não pode ser nulo.", parameterName);
            case TypeParameterException.Invalid:
               return string.Format("O parâmetro \"{0}\" com valor \"{1}\" é inválido.",
parameterName, parameterValue);
            default:
               throw new Exception("Invalid type of parameter exception.");
         }
      }
   }
}
```

- Projeto API.Core:
    - Declaração de Enumeração TypeParameterException;

```
namespace API.Core
{
   public enum TypeParameterException
   {
      IsNullOrEmpty,
      Invalid,
      IsNull
   }
```

```
}
```

- Projeto API.Data.DTO:
    - Classe GrupoUsuario;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
   public class GrupoUsuario
   {
      public GrupoUsuario()
      {
         this.Verificacao = new HashSet<Verificacao>();
         this.Usuario = new HashSet<Usuario>();
      }

      public int idGrupoUsuario { get; set; }
      public string deGrupoUsuario { get; set; }

      public virtual ICollection<Verificacao> Verificacao { get; set; }
      public virtual ICollection<Usuario> Usuario { get; set; }
   }
}
```

- Projeto API.Data.DTO:
    - Classe HistoricoVerificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
   public class HistoricoVerificacao
   {
      public int idHistoricoVirificacao { get; set; }
      public int idVerificacao { get; set; }
      public int idUsuario { get; set; }
      public System.DateTime dtVerificacao { get; set; }

      public virtual Usuario Usuario { get; set; }
      public virtual Verificacao Verificacao { get; set; }
   }
```

```
}
```

- Projeto API.Data.DTO:
    - Classe ProblemaVerificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class ProblemaVerificacao
    {
        public int idProblemaVerificacao { get; set; }
        public int idVerificacao { get; set; }
        public string deProblemaVerificacao { get; set; }
        public byte[] imgProblema { get; set; }
        public System.DateTime dtProblemaVerificacao { get; set; }
        public virtual Verificacao Verificacao { get; set; }
    }
}
```

- Projeto API.Data.DTO:
    - Classe Produto;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class Produto
    {
        public Produto()
        {
            this.Verificacao = new HashSet<Verificacao>();
        }

        public int idProduto { get; set; }
        public int idSetor { get; set; }
        public string deProduto { get; set; }
        public string deLocalizacao { get; set; }
        public byte[] coQrCode { get; set; }

        public virtual Setor Setor { get; set; }
        public virtual ICollection<Verificacao> Verificacao { get; set; }
```

```
    }
}
```

- Projeto API.Data.DTO:
    - Classe Setor;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class Setor
    {
        public Setor()
        {
            this.Produto = new HashSet<Produto>();
        }

        public int idSetor { get; set; }
        public string deSetor { get; set; }

        public virtual ICollection<Produto> Produto { get; set; }
    }
}
```

- Projeto API.Data.DTO:
    - Classe TipoVerificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class TipoVerificacao
    {
        public TipoVerificacao()
        {
            this.Verificacao = new HashSet<Verificacao>();
        }

        public int idTipoVerificacao { get; set; }
        public string deTipoVerificacao { get; set; }

        public virtual ICollection<Verificacao> Verificacao { get; set; }
    }
```

```
}
```

- Projeto API.Data.DTO:
    - Classe Usuario;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class Usuario
    {
        public int idUsuario { get; set; }
        public string nmUsuario { get; set; }
        public string coSenha { get; set; }
        public bool icAdministrador { get; set; }
        public string edEmail { get; set; }

        public virtual ICollection<HistoricoVerificacao> HistoricoVerificacao { get; set; }

        public virtual ICollection<GrupoUsuario> GrupoUsuario { get; set; }
    }
}
```

- Projeto API.Data.DTO:
    - Classe Verificacao;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Data.DTO
{
    public class Verificacao
    {
        public int idVerificacao { get; set; }
        public int idGrupoUsuario { get; set; }
        public int idProduto { get; set; }
        public int idTipoVerificacao { get; set; }
        public string deVerificacao { get; set; }

        public virtual GrupoUsuario GrupoUsuario { get; set; }
        //public virtual ICollection<HistoricoVirificacao> HistoricoVirificacao { get; set; }
        public virtual Produto Produto { get; set; }
        public virtual TipoVerificacao TipoVerificacao { get; set; }
    }
```

```
}
```

- Projeto API.HttpClient:
    - Classe ApiException;

```
using API.HttpClient.Proxy;
using System;

namespace API.HttpClient
{
   public class ApiException : Exception
   {
      private string _id;
      private string _description;

      public string ID {
         get { return _description; }
      }

      public string Description
      {
         get { return _description; }
      }

      public ApiException(ResponseMessageProxy message)
         : base(message.description)
      {
         _id = message.id;
         _description = message.description;
      }
   }
}
```

- Projeto API.HttpClient:
    - Classe Headers;

```
using System;
using System.Globalization;
using System.Resources;
using System.Text;
using System.Reflection;

namespace API.HttpClient
{
   public sealed class Headers
   {
      public static string GetAuthenticationString(string accountName, string tokenAccess)
      {
         return string.Concat("Basic ",
Convert.ToBase64String(Encoding.UTF8.GetBytes(string.Format("{0}:{1}", accountName,
tokenAccess))));
```

```
        }

        public static string GetTimestamp()
        {
            return DateTime.UtcNow.ToString("U", CultureInfo.InvariantCulture);
        }

        public static string DecodeAuthenticationString(string
authenticateUsingHttpBasicAuthentication)
        {
            authenticateUsingHttpBasicAuthentication =
authenticateUsingHttpBasicAuthentication.Replace("Basic ", "");

            byte[] encodedDataAsBytes =
System.Convert.FromBase64String(authenticateUsingHttpBasicAuthentication);

            string returnValue =
System.Text.Encoding.UTF8.GetString(encodedDataAsBytes,0,encodedDataAsBytes.Length);

            return returnValue;
        }
    }
}
```

- Projeto API.HttpClient:
    - Classe HttpClientBase;

```
using API.Core;
using API.HttpClient.Proxy;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Linq;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Text;
using System.Threading.Tasks;
using System.Reflection;
using System.Net;
using static API.Core.ConfigurationManager;
using Android.Widget;

namespace API.HttpClient
{
    public abstract class HttpClientBase
    {
        private System.Net.Http.HttpClient _httpClient = new System.Net.Http.HttpClient();

        private dynamic _parameters = new ExpandoObject();

        protected System.Net.Http.HttpClient Client { get { return _httpClient; } }

        protected string RequestUri { get; set; }

        protected dynamic Parameters { get { return _parameters; } }
```

```
        protected HttpClientBase(string requestUri) //sempre deve ter a string de requisição
        {
            RequestUri = string.Format("api/{0}", requestUri); // para onde ele vai aopontar na api
            Client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
        }

        protected async Task<ResponseProxy<TResult>> ExecuteRequest<T1, TResult>(T1 param,
HttpMethod method = HttpMethod.Get) //Classe assincrona task, que retorna reponse proxy
            where TResult : class
            where T1 : class
        {
            try
            {
                var parameters = new StringContent("");
                string queryString = "";
                switch (method)
                {
                    case HttpMethod.Get:
                        queryString = GetQueryString(param);
                        return await GetRequest<TResult>(queryString);

                    case HttpMethod.Delete:
                        queryString = GetQueryString(param);
                        return await DeleteRequest<TResult>(queryString);

                    case HttpMethod.Post:
                        parameters = GetStringContent(param);
                        return await PostRequest<TResult>(parameters);

                    case HttpMethod.Put:
                        parameters = GetStringContent(param);
                        return await PutRequest<TResult>(parameters);

                    default:
                        return new ResponseProxy<TResult>();
                }
            }
            catch (Exception ex)
            {
                var result = new ResponseProxy<TResult>();
                result.message.description = ex.Message;
                return result;
            }

        }

        protected ResponseProxy<T> UploadResponse<T>(FileSet fileSet, Dictionary<string, object>
queryStringParameters = null)
            where T : class
        {
            var multipartContent = new MultipartFormDataContent();

            var postedFilesJson = JsonConvert.SerializeObject(fileSet,
JsonSettings.GetSerializerSettings());

            multipartContent.Add(new StringContent(postedFilesJson, Encoding.UTF8,
"application/json"), "fileset");

            int counter = 0;

            foreach (var file in fileSet.Files)
```

```
        {
            var fileContent = new ByteArrayContent(file.Content);
            fileContent.Headers.ContentType = new MediaTypeHeaderValue(file.MimeType);
            multipartContent.Add(fileContent, "file" + counter++, file.Name);
        }

        var httpClient = new System.Net.Http.HttpClient();


        string queryString = "";

        if (!string.IsNullOrEmpty(queryString) && queryStringParameters.Count > 0)
        {
            queryString = string.Format("?{0}", GetQueryString(queryStringParameters));
        }

var uri = new Uri(string.Format("{0}/{1}{2}", AppSettings.ApiUrl, RequestUri, queryString));

        var responseMessage = httpClient.PostAsync(uri.ToString(), multipartContent).Result;

        var response = responseMessage.Content.ReadAsStringAsync().Result;

        return JsonConvert.DeserializeObject<ResponseProxy<T>>(response,
JsonSettings.GetSerializerSettings());
    }

    /// <summary>
    /// Método utilizado para enviar requisições POST para o barramento de serviço.
    /// </summary>
    /// <typeparam name="T1"></typeparam>
    /// <typeparam name="TResult"></typeparam>
    /// <param name="parameter"></param>
    /// <returns></returns>
    protected async Task<ResponseProxy<TResult>> PostRequest<TResult>(StringContent
stringContent) // PostRequest tipo de retorno T1 e TResult.
        where TResult : class
    {
        using (var client = new System.Net.Http.HttpClient())
        {


            var responseString = string.Empty; // Usar string vazio de inicio

            client.BaseAddress = new Uri(AppSettings.ApiUrl);  // vai no appsettings pegar a ApiURl
e setar como uma nova url, atribuindo a client.base Address


            client.DefaultRequestHeaders.Accept.Clear(); // formatos que o client aceita. Primeiro
limpa
            client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json")); // adiciona um novo de tal forma

            HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post,RequestUri);
            request.Content = stringContent;
            var response = await client.SendAsync(request).ConfigureAwait(false);

            if (response.IsSuccessStatusCode)
            {
                responseString = await response.Content.ReadAsStringAsync(); // retorno dos dados
em string
```

```
            }

            var responseProxy = await Task.Factory.StartNew(() => JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString, JsonSettings.GetSerializerSettings())); // explicar o funcionamento, que está deseraliazando um objeto

            return responseProxy; // retorno já no formato necessário. objeto TResult. retorna do tipo responseProxy, nele tem o data, que retornara tresult no caso
        }
    }

    /// <summary>
    /// Método utilizado para enviar requisições PUT para o barramento de serviço.
    /// </summary>
    /// <typeparam name="T1">TIpo do parametro</typeparam>
    /// <typeparam name="TResult">Tipo do retorno</typeparam>
    /// <param name="parameter">objeto parameter utilizado para </param>
    /// <returns></returns>
    protected async Task<ResponseProxy<TResult>> PutRequest<TResult>(StringContent stringContent)
        where TResult : class
    {
        using (var client = new System.Net.Http.HttpClient())
        {
            try
            {

                var responseString = string.Empty;

                client.BaseAddress = new Uri(AppSettings.ApiUrl);


                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

                HttpRequestMessage request = new HttpRequestMessage(System.Net.Http.HttpMethod.Post, RequestUri);
                request.Content = stringContent;
                var response = await client.SendAsync(request).ConfigureAwait(false);
                if (response.IsSuccessStatusCode)
                {
                    responseString = await response.Content.ReadAsStringAsync();
                }

                var responseProxy = await Task.Factory.StartNew(() => JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString, JsonSettings.GetSerializerSettings()));

                return responseProxy;
            }
            catch (HttpRequestException EX)
            {
                throw new ApiException(new ResponseMessageProxy
                {
                    description = EX.Message,
                    id = EX.Source.ToString()
                });
            }
        }
    }
```

```
        /// <summary>
        /// Método utilizado para enviar requisições GET para o barramento de serviço.
        /// </summary>
        /// <typeparam name="TResult">Tipo do retorno</typeparam>
        /// <param name="parameter">objeto parameter utilizado para </param>
        /// <returns></returns>
        protected async Task<ResponseProxy<TResult>> GetRequest<TResult>(string QueryString)
            where TResult : class
        {
            var queryString = string.Format("?{0}", QueryString);
            var responseString = "";
            using (var client = new System.Net.Http.HttpClient())
            {
                client.BaseAddress = new Uri(string.Format("{0}/{1}{2}", AppSettings.ApiUrl, RequestUri,
queryString));

                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));

                var response = await client.GetAsync(client.BaseAddress);
                if (response.IsSuccessStatusCode)
                {
                    responseString = await response.Content.ReadAsStringAsync();
                }
                var responseProxy = new ResponseProxy<TResult>();
                try
                {

                    responseProxy                                                                    =
JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString,
JsonSettings.GetSerializerSettings());
                }catch(JsonException Jex)
                {
                    throw Jex;
                }
                return responseProxy;
            }
        }

        /// <summary>
        /// Método utilizado para enviar requisições DELETE para o barramento de serviço.
        /// </summary>
        /// <typeparam name="TResult">Tipo do retorno</typeparam>
        /// <param name="parameter">objeto parameter utilizado para </param>
        /// <returns></returns>
        protected async Task<ResponseProxy<TResult>> DeleteRequest<TResult>(string
parameter)
            where TResult : class
        {
            var queryString = string.Format("?{0}", parameter);
            var responseString = "";
            using (var client = new System.Net.Http.HttpClient())
            {
                client.BaseAddress = new Uri(string.Format("{0}/{1}{2}", AppSettings.ApiUrl, RequestUri,
queryString));

                client.DefaultRequestHeaders.Accept.Clear();
                client.DefaultRequestHeaders.Accept.Add(new
MediaTypeWithQualityHeaderValue("application/json"));
```

```
            var response = await client.DeleteAsync(client.BaseAddress).ConfigureAwait(false);
            if (response.IsSuccessStatusCode)
            {
               responseString = await response.Content.ReadAsStringAsync();
            }

            var responseProxy = await Task.Factory.StartNew(() => JsonConvert.DeserializeObject<ResponseProxy<TResult>>(responseString, JsonSettings.GetSerializerSettings()));

            return responseProxy;
         }
      }
      #region Construtores
      private StringContent GetStringContent(object obj)
      {
         var jsonObject = JsonConvert.SerializeObject(obj);
         StringContent result = new StringContent(jsonObject);
         return result;
      }

      public string GetQueryString(object obj)
      {
         try
         {
            var properties = from p in obj.GetType().GetRuntimeProperties()
                             where p.GetValue(obj, null) != null
                             select p.Name + "=" + WebUtility.UrlEncode(p.GetValue(obj, null).ToString());

            return String.Join("&", properties.ToArray());
         }catch(Exception e)
         { throw e; }
      }
      #endregion
   }


}
```

- Projeto API.Parameters:
    - Classe GrupoUsuarioParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
   public class GrupoUsuarioParameter : ParameterBase // Sempre é pasado o parametro dessa forma para a API.
   {
      public int idGrupoUsuario { get; set; } // Objeto que pode tanto atribuir quanto setar o valor
      public int idUsuario { get; set; }
      public string GrupoUsuarioJSON { get; set; }

      public enum Operacao //tipos de operações que pode ter
```

```
        {
            Get = 1 ,
            GetList = 2,
            GetListByUsuario = 3,
            Create  = 15
        }
    }
}
```

- Projeto API.Parameters:
    - Classe HistoricoVerificacaoParameter;

```
using API.Data.DTO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class HistoricoVerificacaoParameter : ParameterBase
    {
        public int idHistoricoVerificacao { get; set; }
        public string HistoricoVerificacaoJSON { get; set; }

        public enum Operacao
        {
            Get = 1,
            GetList = 2,
            GetListByUsuario = 3,
            Create = 15
        }
    }
}
```

- Projeto API.Parameters:
    - Classe ParameterBase;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class ParameterBase
    {
        public string includeProperties { get; set; }
        public int nuPagina { get; set; }
```

```
        public int nuResultados { get; set; }
        public int idOperacao { get; set; }
    }
}
```

- Projeto API.Parameters:
    - Classe ProblemaVerificacaoParameter;

```
using API.Data.DTO;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class ProblemaVerificacaoParameter : ParameterBase
    {
        public int idProblemaVerificacao { get; set; }
        public ProblemaVerificacao ProblemaVerificacao { get; set; }
        public string ProblemaVerificacaoJSON { get; set; }

        public enum Operacao
        {
            Get = 1,
            GetList = 2,
            GetListByVerificacao = 3,
            Create = 15,

        }
    }
}
```

- Projeto API.Parameters:
    - Classe ProdutoParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class ProdutoParameter:ParameterBase
    {
        public int idProduto { get; set; }
        public string ProdutoJSON { get; set; }

        public enum Operacao
```

```
        {
            Get = 1,
            GetList =2,
            Create = 15
        }
    }
}
```

- Projeto API.Parameters:
    - Classe SetorParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class SetorParameter : ParameterBase
    {
        public int idSetor { get; set; }
        public string SetorJSON { get; set; }
        public enum Operacao
        {
            Get = 1,
            GetList = 2,
            Create = 15
        }
    }
}
```

- Projeto API.Parameters:
    - Classe TipoVerificacaoParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
    public class TipoVerificacaoParameter : ParameterBase
    {
        public int idTipoVerificacao { get; set; }
        public string TipoVerificacaoJSON { get; set; }
        public enum Operacao
        {
            Get = 1,
            GetList = 2,
```

```
        Create = 15
      }
   }
}
```

- Projeto API.Parameters:
    - Classe UsuarioParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;


namespace API.Parameters
{
   public class UsuarioParameter : ParameterBase
   {
      public int idUsuario { get; set; }
      public string edEmail { get; set; }
      public string nmUsuario { get; set; }
      public string coSenha { get; set; }
      public string UsuarioJSON { get; set; }
      public enum Operacao
      {
         Get = 1,
         GetByEmailSenha = 2,
         GetList = 3,
         IncluirUsuario = 20,
         UpdateUsuario = 21
      }
   }
}
```

- Projeto API.Parameters:
    - Classe VerificacaoParameter;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Parameters
{
   public class VerificacaoParameter : ParameterBase
   {
      public int idVerificacao { get; set; }
      public int idGrupoUsuario { get; set; }
      public string VerificacaoJSON { get; set; }
```

```
        public enum Operacao
        {
            Get = 1,
            GetList = 2,
            GetListByGrupoUsuario = 3,
            Create = 20,
            Update = 21
        }
    }
}
```

- Projeto API.Services:
    - Classe GrupoUsuarioAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.GrupoUsuarioParameter;

namespace API.Services
{
    public class GrupoUsuarioAPIServices : HttpClientBase, IAPIServicesRepository<GrupoUsuario, GrupoUsuarioParameter> // A requisição do GrupoUsuario herda de HttpclientBase, utilizando a interface IAPIServiceRepository, utilizando os dois tipos de retorno.
    {
        public GrupoUsuarioAPIServices
            (
                Operacao operacao, //lista de nome que equivalem um número. deixa o aplicativo mais legivel. Para não ter que trabalhar com vairos tipos de nomes de operações.
                int? idGrupoUsuario = null, //colocar interrogação nos valores primitivos para atribuir valores nulos.
                int? idUsuario = null, //colocar interrogação nos valores primitivos para atribuir valores nulos.
                string includeProperties = ""
            ) : base("GrupoUsuario") // atribuir GrupoUsuarioAPIServices a base grupoUguario. Se quiser herdar qlqr coisa utilizar o BASe
        {
            Parameters.idOperacao = (int)operacao;
            Parameters.idGrupoUsuario = idGrupoUsuario;
            Parameters.idUsuario = idUsuario;
            Parameters.includeProperties = includeProperties;
        }


        //Sempre necessita ter todas essas funções abaixo quando está utilizando uma interface.
        public bool Delete()
        {
            throw new NotImplementedException();
        }
```

```
        public bool Delete(out object data)
        {
            throw new NotImplementedException();
        }

        public async Task<GrupoUsuario> Get(GrupoUsuarioParameter parameter)
        {
            var result = await ExecuteRequest<GrupoUsuarioParameter, GrupoUsuario>(parameter,
HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<IList<GrupoUsuario>> GetList(GrupoUsuarioParameter parameter)
        {
            var result = await ExecuteRequest<GrupoUsuarioParameter,
List<GrupoUsuario>>(parameter,HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<GrupoUsuario> Post(GrupoUsuarioParameter parameter)
        {
            var result = await ExecuteRequest<GrupoUsuarioParameter,
GrupoUsuario>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<GrupoUsuario> Update(GrupoUsuarioParameter parameter)
        {
            var result = await ExecuteRequest<GrupoUsuarioParameter, GrupoUsuario>(parameter,
HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:
    - Classe HistoricoVerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.HistoricoVerificacaoParameter;

namespace API.Services
{
   public class HistoricoVerificacaoAPIServices : HttpClientBase, IAPIServicesRepository<HistoricoVerificacao, HistoricoVerificacaoParameter>
   {
      public HistoricoVerificacaoAPIServices
         (
            Operacao operacao,
            int? idHistoricoVerificacao = null
         ) : base("HistoricoVerificacao")
      {
         Parameters.idOperacao = (int)operacao;

         Parameters.idHistoricoVerificacao = idHistoricoVerificacao;
      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<HistoricoVerificacao> Get(HistoricoVerificacaoParameter parameters)
      {
         var result = await ExecuteRequest<HistoricoVerificacaoParameter, HistoricoVerificacao>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<IList<HistoricoVerificacao>> GetList(HistoricoVerificacaoParameter parameters)
      {
         var result = await ExecuteRequest<HistoricoVerificacaoParameter, List<HistoricoVerificacao>>(parameters, HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
```

```
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<HistoricoVerificacao> Post(HistoricoVerificacaoParameter parameter)
        {
            var result = await ExecuteRequest<HistoricoVerificacaoParameter,
HistoricoVerificacao>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<HistoricoVerificacao> Update(HistoricoVerificacaoParameter parameter)
        {
            var result = await ExecuteRequest<HistoricoVerificacaoParameter,
HistoricoVerificacao>(parameter,HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
throw new Exception(result.message.description);
}
}
}
```

- Projeto API.Services:
    - Classe IAPIServicesRepository;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace API.Services
{
    public interface IAPIServicesRepository<TResult, TParameter>
        where TResult : class
        where TParameter : class
    {
        Task<TResult> Get(TParameter parameter);
        Task<IList<TResult>> GetList(TParameter parameter);
        Task<TResult> Post(TParameter parameter);
        Task<TResult> Update(TParameter parameter);
        bool Delete();
        bool Delete(out object data);
    }
}
```

- Projeto API.Services:
    - Classe ProblemaVerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.ProblemaVerificacaoParameter;

namespace API.Services
{
   public class ProblemaVerificacaoAPIServices : HttpClientBase, IAPIServicesRepository<ProblemaVerificacao, ProblemaVerificacaoParameter>
   {
      public ProblemaVerificacaoAPIServices
         (
            Operacao operacao,
            int? idProblemaVerificacao = null
         ) : base("ProblemaVerificacao")
      {

      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<ProblemaVerificacao> Get(ProblemaVerificacaoParameter parameters)
      {
         var result = await ExecuteRequest<ProblemaVerificacaoParameter,ProblemaVerificacao>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<IList<ProblemaVerificacao>> GetList(ProblemaVerificacaoParameter parameters)
      {
```

```
        var result = await ExecuteRequest<ProblemaVerificacaoParameter,List<ProblemaVerificacao>>(parameters,HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<ProblemaVerificacao> Post(ProblemaVerificacaoParameter parameter)
    {
        var result = await ExecuteRequest<ProblemaVerificacaoParameter, ProblemaVerificacao>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<ProblemaVerificacao> Update(ProblemaVerificacaoParameter parameter)
    {
        var result = await ExecuteRequest<ProblemaVerificacaoParameter, ProblemaVerificacao>(parameter, HttpMethod.Put);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
  }
}
```

- Projeto API.Services:
    - Classe ProdutoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.ProdutoParameter;

namespace API.Services
{
  public class ProdutoAPIServices : HttpClientBase, IAPIServicesRepository<Produto, ProdutoParameter>
  {
    public ProdutoAPIServices
      (
        Operacao operacao,
```

```
            int? idProduto = null,
            string includeProperties = ""
        ) : base("Produto")
    {
        Parameters.idOperacao = (int)operacao;
        Parameters.idProduto = idProduto;
        Parameters.includeProperties = includeProperties;
    }

    public bool Delete()
    {
        throw new NotImplementedException();
    }

    public bool Delete(out object data)
    {
        throw new NotImplementedException();
    }

    public async Task<Produto> Get(ProdutoParameter parameters)
    {
        var result = await ExecuteRequest<ProdutoParameter,
Produto>(parameters,HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<IList<Produto>> GetList(ProdutoParameter parameters)
    {
        var result = await
ExecuteRequest<ProdutoParameter,List<Produto>>(parameters,HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<Produto> Post(ProdutoParameter parameter)
    {
        var result = await ExecuteRequest<ProdutoParameter,
Produto>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<Produto> Update(ProdutoParameter parameter)
    {
        var result = await ExecuteRequest<ProdutoParameter,
Produto>(parameter,HttpMethod.Put);
        if (result.status == ResponseStatus.Success)
```

```
            return result.data;
         else
            throw new Exception(result.message.description);
      }
   }
}
```

- Projeto API.Services:
    - Classe SetorAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.SetorParameter;

namespace API.Services
{
   public class SetorAPIServices : HttpClientBase, IAPIServicesRepository<Setor,
SetorParameter>
   {
      public SetorAPIServices
         (
            Operacao operacao,
            int?  idSetor = null,
            string includeProperties = ""
         ) : base("Setor")
      {
         Parameters.idOperacao = (int)operacao;
         Parameters.idSetor = idSetor;
         Parameters.includeProperties = includeProperties;
      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<Setor> Get(SetorParameter parameters)
      {
         var result = await ExecuteRequest<SetorParameter,Setor>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
```

```
            throw new Exception(result.message.description);
        }
    }

    public async Task<IList<Setor>> GetList(SetorParameter parameters)
    {
        var result = await ExecuteRequest<SetorParameter,
List<Setor>>(parameters,HttpMethod.Get);
        if (result.status == ResponseStatus.Success)
        {
            return result.data;
        }
        else
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<Setor> Post(SetorParameter parameter)
    {
        var result = await ExecuteRequest<SetorParameter, Setor>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<Setor> Update(SetorParameter parameter)
    {
        var result = await ExecuteRequest<SetorParameter, Setor>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
  }
}
```

- Projeto API.Services:
    - Classe TipoVerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.TipoVerificacaoParameter;

namespace API.Services
{
  public class TipoVerificacaoAPIServices : HttpClientBase,
IAPIServicesRepository<TipoVerificacao, TipoVerificacaoParameter>
  {
```

```
        public TipoVerificacaoAPIServices
            (
                Operacao operacao,
                int? idTipoVerificacao = null,
                string includeProperties = ""
            ) : base("TipoVerificacao")
        {
            Parameters.idOperacao = (int)operacao;
            Parameters.idTipoVerificacao = idTipoVerificacao;
            Parameters.includeProperties = includeProperties;
        }

        public bool Delete()
        {
            throw new NotImplementedException();
        }

        public bool Delete(out object data)
        {
            throw new NotImplementedException();
        }

        public async Task<TipoVerificacao> Get(TipoVerificacaoParameter parameters)
        {
            var result = await
ExecuteRequest<TipoVerificacaoParameter,TipoVerificacao>(parameters,HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<IList<TipoVerificacao>> GetList(TipoVerificacaoParameter parameters)
        {
            var result = await
ExecuteRequest<TipoVerificacaoParameter,List<TipoVerificacao>>(parameters,HttpMethod.Get)
;
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<TipoVerificacao> Post(TipoVerificacaoParameter parameter)
        {
            var result = await ExecuteRequest<TipoVerificacaoParameter,
TipoVerificacao>(parameter);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<TipoVerificacao> Update(TipoVerificacaoParameter parameter)
```

```
        {
            var result = await ExecuteRequest<TipoVerificacaoParameter,
TipoVerificacao>(parameter);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:
    - Classe UsuarioAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.UsuarioParameter;

namespace API.Services
{
    public class UsuarioAPIServices : HttpClientBase, IAPIServicesRepository<Usuario,
UsuarioParameter>
    {

        public UsuarioAPIServices
            (
                Operacao idOperacao,
                int? idUsuario = null,
                string edEmail = "",
                string coSenha = "",
                string includeProperties = ""
            ) : base("Usuario")
        {
            Parameters.edEmail = edEmail;
            Parameters.coSenha = coSenha;
            Parameters.includeProperties = includeProperties;
            Parameters.idUsuario = idUsuario;

            Parameters.idOperacao = (int)idOperacao;
        }

        public bool Delete()
        {
            throw new NotImplementedException();
        }

        public bool Delete(out object data)
        {
            throw new NotImplementedException();
        }
```

```
        public async Task<Usuario> Get(UsuarioParameter parameters)
        {
            var result = await
ExecuteRequest<UsuarioParameter,Usuario>(parameters,HttpMethod.Get);
            if(result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<IList<Usuario>> GetList(UsuarioParameter parameters)
        {
            var result = await
ExecuteRequest<UsuarioParameter,List<Usuario>>(parameters,HttpMethod.Get);
            if (result.status == ResponseStatus.Success)
            {
                return result.data;
            }
            else
            {
                throw new Exception(result.message.description);
            }
        }

        public async Task<Usuario> Post(UsuarioParameter parameter)
        {
            var result = await ExecuteRequest<UsuarioParameter,
Usuario>(parameter,HttpMethod.Post);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }

        public async Task<Usuario> Update(UsuarioParameter parameter)
        {
            var result = await ExecuteRequest<UsuarioParameter,
Usuario>(parameter,HttpMethod.Put);
            if (result.status == ResponseStatus.Success)
                return result.data;
            else
                throw new Exception(result.message.description);
        }
    }
}
```

- Projeto API.Services:
    - Classe VerificacaoAPIServices;

```
using API.Data.DTO;
using API.HttpClient;
using API.HttpClient.Proxy;
```

```
using API.Parameters;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static API.Parameters.VerificacaoParameter;

namespace API.Services
{
   public class VerificacaoAPIServices : HttpClientBase, IAPIServicesRepository<Verificacao,
VerificacaoParameter>
   {
      public VerificacaoAPIServices
         (
            Operacao operacao,
            int? idAtividade = null,
            int? idGrupoUsuario = null,
            string includeProperties = ""
         )
         :base("Verificacao")
      {
         Parameters.idOperacao = (int)operacao;
         Parameters.idAtividade = idAtividade;
         Parameters.idGrupoUsuario = idGrupoUsuario;

         Parameters.includeProperties = includeProperties;
      }

      public bool Delete()
      {
         throw new NotImplementedException();
      }

      public bool Delete(out object data)
      {
         throw new NotImplementedException();
      }

      public async Task<Verificacao> Get(VerificacaoParameter parameters)
      {
         var result = await
ExecuteRequest<VerificacaoParameter,Verificacao>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
         {
            throw new Exception(result.message.description);
         }
      }

      public async Task<IList<Verificacao>> GetList(VerificacaoParameter parameters)
      {
         var result = await ExecuteRequest<VerificacaoParameter,
List<Verificacao>>(parameters,HttpMethod.Get);
         if (result.status == ResponseStatus.Success)
         {
            return result.data;
         }
         else
```

```
        {
            throw new Exception(result.message.description);
        }
    }

    public async Task<Verificacao> Post(VerificacaoParameter parameter)
    {
        var result = await ExecuteRequest<VerificacaoParameter,
Verificacao>(parameter,HttpMethod.Post);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }

    public async Task<Verificacao> Update(VerificacaoParameter parameter)
    {
        var result = await ExecuteRequest<VerificacaoParameter,
Verificacao>(parameter,HttpMethod.Put);
        if (result.status == ResponseStatus.Success)
            return result.data;
        else
            throw new Exception(result.message.description);
    }
  }
}
```

- Projeto API.Services:
    - Classe VerificacaoAPIServices;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Reflection;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using API.Parameters;
using static API.Core.ConfigurationManager;
using API.HttpClient.Proxy;
using API.Data.DTO;

namespace App06Teste
{
  [Activity(Label = "CriarVerificacaoActivity")]
  public class CriarVerificacaoActivity : Activity
  {
    Spinner spnProduto;
    Spinner spnTipoVerificacao;
```

```
        Spinner spnGrupoUsuario;
        EditText txtdeVerificacao;
        Button btnSalvarVerificacao;
        List<GrupoUsuario> _listaGrupoUsuario;
        List<TipoVerificacao> _listaTipoVerificacao;
        List<Produto> _listaProduto;
        Verificacao _verificacao;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.CriarVerificacao);
            // Create your application here
            spnProduto = FindViewById<Spinner>(Resource.Id.spnProduto);
            spnTipoVerificacao = FindViewById<Spinner>(Resource.Id.spnTipoVerificacao);
            spnGrupoUsuario = FindViewById<Spinner>(Resource.Id.spnGrupoUsuario);
            txtdeVerificacao = FindViewById<EditText>(Resource.Id.txtdeVerificacao);

            btnSalvarVerificacao = FindViewById<Button>(Resource.Id.btnSalvarVerificacao);

            btnSalvarVerificacao.Click += BtnSalvarVerificacao_Click;
        }

        protected async override void OnResume()
        {
            base.OnResume();

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    GrupoUsuarioParameter parameter = new GrupoUsuarioParameter()
                    {
                        idOperacao = (int)GrupoUsuarioParameter.Operacao.GetList
                    };
                    var uri = new Uri(string.Format("{0}/api/GrupoUsuario/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
```

```
                {
                    await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                    {
                        if (ta.Exception == null)
                        {
                            stringResult = ta.Result;
                        }
                        else
                        {
                            SetAlerta(ta.Exception.Message);
                        };
                    })); ;
                } catch (HttpRequestException HRex)

                {
                    SetAlerta(HRex.Message + HRex.StackTrace);
                }
                catch (Exception ex)
                {
                    SetAlerta(ex.Message + ex.StackTrace);
                }
                var                                    proxyUsuario                                    =
JsonConvert.DeserializeObject<ResponseProxy<List<GrupoUsuario>>>(stringResult);
                if (proxyUsuario.status == ResponseStatus.Success)
                    _listaGrupoUsuario = proxyUsuario.data;
                else
                    SetAlerta(proxyUsuario.message.description);
            }
        }
    }
    catch (Exception ex) { SetAlerta(ex.Message); }

    try
    {
        using (HttpClient Client = new HttpClient())
        {
            Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
            Client.MaxResponseContentBufferSize = 256000;
            TipoVerificacaoParameter parameter = new TipoVerificacaoParameter()
            {
                idOperacao = (int)TipoVerificacaoParameter.Operacao.GetList
            };
            var uri = new Uri(string.Format("{0}/api/TipoVerificacao/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

            var requestMessage = new HttpRequestMessage()
            {
                RequestUri = uri,
                Method = HttpMethod.Get
            };
            var response = new HttpResponseMessage();
            try
            {
                response = await Client.GetAsync(uri).ConfigureAwait(true);
            }
            catch (HttpRequestException HRex)
            {
                SetAlerta(HRex.Message + HRex.StackTrace);
            }
            catch (Exception ex)
```

```
                {
                    SetAlerta(ex.Message + ex.StackTrace);
                }
                if (response.IsSuccessStatusCode)
                {
                    string stringResult = "";
                    try
                    {
                        await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                        {
                            if (ta.Exception == null)
                            {
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var proxyUsuario =
JsonConvert.DeserializeObject<ResponseProxy<List<TipoVerificacao>>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _listaTipoVerificacao = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                }
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }

        try
        {
            using (HttpClient Client = new HttpClient())
            {
                Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                Client.MaxResponseContentBufferSize = 256000;
                ProdutoParameter parameter = new ProdutoParameter()
                {
                    idOperacao = (int)VerificacaoParameter.Operacao.GetList
                };
                var uri = new Uri(string.Format("{0}/api/Produto/Get?{1}", AppSettings.ApiUrl,
GetQueryString(parameter)));

                var requestMessage = new HttpRequestMessage()
                {
                    RequestUri = uri,
                    Method = HttpMethod.Get
                };
                var response = new HttpResponseMessage();
                try
                {
                    response = await Client.GetAsync(uri).ConfigureAwait(true);
```

```
                }
                catch (HttpRequestException HRex)
                {
                    SetAlerta(HRex.Message + HRex.StackTrace);
                }
                catch (Exception ex)
                {
                    SetAlerta(ex.Message + ex.StackTrace);
                }
                if (response.IsSuccessStatusCode)
                {
                    string stringResult = "";
                    try
                    {
                        await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                        {
                            if (ta.Exception == null)
                            {
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var                                    proxyUsuario                                    =
JsonConvert.DeserializeObject<ResponseProxy<List<Produto>>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _listaProduto = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                }
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }

        var adapter = new ArrayAdapter(this, Resource.Layout.spinnerTextFile);
        foreach(var grupoUsu in _listaGrupoUsuario)
    {

adapter.Add(grupoUsu.deGrupoUsuario);
        }
        spnGrupoUsuario.Adapter = adapter;

        var adapterVeri = new ArrayAdapter(this, Resource.Layout.spinnerText2);
        foreach(var tipoVerificacao in _listaTipoVerificacao)
        {
            adapterVeri.Add(tipoVerificacao.deTipoVerificacao);
        }
        spnTipoVerificacao.Adapter = adapterVeri;

        var adapterProduto = new ArrayAdapter(this, Resource.Layout.spinnerText3);
```

```
            foreach( var prod in _listaProduto.Select(prod => new { prod.deProduto }).Distinct())
            {
                adapterProduto.Add(prod.deProduto);
            }
            spnProduto.Adapter = adapterProduto;
        }

        private async void BtnSalvarVerificacao_Click(object sender, EventArgs e)
        {
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new VerificacaoParameter()
                    {
                        idOperacao = (int)VerificacaoParameter.Operacao.Create,
                        VerificacaoJSON = JsonConvert.SerializeObject(new Verificacao()
                        {
                            deVerificacao = txtdeVerificacao.Text,
                            idTipoVerificacao = _listaTipoVerificacao.Where(tipoVerif =>
tipoVerif.deTipoVerificacao ==
spnTipoVerificacao.SelectedItem.ToString()).FirstOrDefault().idTipoVerificacao,
                            idProduto = _listaProduto.Where(prod => prod.deProduto ==
spnProduto.SelectedItem.ToString()).FirstOrDefault().idProduto,
                            idGrupoUsuario = _listaGrupoUsuario.Where(grupoUsu =>
grupoUsu.deGrupoUsuario ==
spnGrupoUsuario.SelectedItem.ToString()).FirstOrDefault().idGrupoUsuario
                        })
                    };
                    var uri = new Uri(string.Format("{0}/api/Verificacao/Post", AppSettings.ApiUrl));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Post
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        HttpRequestMessage request = new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, uri);
                        request.Content = GetStringContent(parameter);
                        request.Headers.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                        response = await Client.SendAsync(request).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        SetAlerta("Sucesso");
                        string stringResult = "";
                        try
```

```
                {
                    await response.Content.ReadAsStringAsync().ContinueWith((ta =>
                    {
                        if (ta.Exception == null)
                        {
                            stringResult = ta.Result;
                        }
                        else
                        {
                            SetAlerta(ta.Exception.Message);
                        };
                    })); ;
                }
                catch (HttpRequestException HRex)
                {
                    SetAlerta(HRex.Message + HRex.StackTrace);
                }
                catch (Exception ex)
                {
                    SetAlerta(ex.Message + ex.StackTrace);
                }
                var                                    proxyUsuario                                    =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                if (proxyUsuario.status == ResponseStatus.Success)
                    _verificacao = proxyUsuario.data;
                else
                    SetAlerta(proxyUsuario.message.description);
                if (_verificacao.idVerificacao > 0)
                    StartActivity(typeof(MenuAdministracaoActivity));
                else
                    SetAlerta("Ocorreu um erro.");
            }
            else
                SetAlerta(response.StatusCode.ToString());
        }
    }
    catch (Exception ex) { SetAlerta(ex.Message); }
}

private void SetAlerta(string v)
{
    Toast toast = Toast.MakeText(this, v, ToastLength.Short);
    toast.Show();
}

public string GetQueryString(object obj)
{
    var properties = from p in obj.GetType().GetRuntimeProperties()
                     where p.GetValue(obj, null) != null
                     select p.Name + "=" + WebUtility.UrlEncode(p.GetValue(obj, null).ToString());

    return String.Join("&", properties.ToArray());
}
private StringContent GetStringContent(object obj)
{
    var jsonObject = JsonConvert.SerializeObject(obj);
    StringContent result = new StringContent(jsonObject, Encoding.UTF8, "application/json");
    return result;
}
  }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp)::
    - Classe CriarVerificacaoActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Reflection;
using System.Net;
using System.Net.Http;
using Newtonsoft.Json;
using API.Parameters;
using static API.Core.ConfigurationManager;
using API.HttpClient.Proxy;
using API.Data.DTO;

namespace App06Teste
{
    [Activity(Label = "CriarVerificacaoActivity")]
    public class CriarVerificacaoActivity : Activity
    {
        Spinner spnProduto;
        Spinner spnTipoVerificacao;
        Spinner spnGrupoUsuario;
        EditText txtdeVerificacao;
        Button btnSalvarVerificacao;
        List<GrupoUsuario> _listaGrupoUsuario;
        List<TipoVerificacao> _listaTipoVerificacao;
        List<Produto> _listaProduto;
        Verificacao _verificacao;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.CriarVerificacao);
            // Create your application here
            spnProduto = FindViewById<Spinner>(Resource.Id.spnProduto);
            spnTipoVerificacao                                          =
FindViewById<Spinner>(Resource.Id.spnTipoVerificacao);
            spnGrupoUsuario                                             =
FindViewById<Spinner>(Resource.Id.spnGrupoUsuario);
            txtdeVerificacao                                            =
FindViewById<EditText>(Resource.Id.txtdeVerificacao);

            btnSalvarVerificacao                                        =
FindViewById<Button>(Resource.Id.btnSalvarVerificacao);

            btnSalvarVerificacao.Click += BtnSalvarVerificacao_Click;
        }

        protected async override void OnResume()
```

```
        {
            base.OnResume();

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    GrupoUsuarioParameter        parameter        =        new
GrupoUsuarioParameter()
                    {
                        idOperacao                                          =
(int)GrupoUsuarioParameter.Operacao.GetList
                    };
                    var                uri                =                new
Uri(string.Format("{0}/api/GrupoUsuario/Get?{1}",          AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response                    =                    await
Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
```

```
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                    proxyUsuario                    =
JsonConvert.DeserializeObject<ResponseProxy<List<GrupoUsuario>>>(stringResult)
;
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaGrupoUsuario = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    TipoVerificacaoParameter       parameter       =       new
TipoVerificacaoParameter()
                    {
                        idOperacao                                          =
(int)TipoVerificacaoParameter.Operacao.GetList
                    };
                    var                uri                =                new
Uri(string.Format("{0}/api/TipoVerificacao/Get?{1}",        AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response                    =                    await
Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
```

```
                                    {
                                        stringResult = ta.Result;
                                    }
                                 else
                                {
                                    SetAlerta(ta.Exception.Message);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                         proxyUsuario                         =
JsonConvert.DeserializeObject<ResponseProxy<List<TipoVerificacao>>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaTipoVerificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    ProdutoParameter parameter = new ProdutoParameter()
                    {
                        idOperacao = (int)VerificacaoParameter.Operacao.GetList
                    };
                    var   uri   =   new   Uri(string.Format("{0}/api/Produto/Get?{1}",
AppSettings.ApiUrl, GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
```

```
                    {
                        string stringResult = "";
                        try
                        {
                            await response.Content.ReadAsStringAsync().ContinueWith((ta
=>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                         proxyUsuario                         =
JsonConvert.DeserializeObject<ResponseProxy<List<Produto>>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaProduto = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            var adapter = new ArrayAdapter(this, Resource.Layout.spinnerTextFile);
            foreach(var grupoUsu in _listaGrupoUsuario)
            {
                adapter.Add(grupoUsu.deGrupoUsuario);
            }
            spnGrupoUsuario.Adapter = adapter;

            var adapterVeri = new ArrayAdapter(this, Resource.Layout.spinnerText2);
            foreach(var tipoVerificacao in _listaTipoVerificacao)
            {
                adapterVeri.Add(tipoVerificacao.deTipoVerificacao);
            }
            spnTipoVerificacao.Adapter = adapterVeri;

            var adapterProduto = new ArrayAdapter(this, Resource.Layout.spinnerText3);
            foreach(  var  prod  in  _listaProduto.Select(prod  =>  new  {  prod.deProduto
}).Distinct())
            {
                adapterProduto.Add(prod.deProduto);
            }
            spnProduto.Adapter = adapterProduto;
                }

private async void BtnSalvarVerificacao_Click(object sender, EventArgs e)
```

```
        {
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new VerificacaoParameter()
                    {
                        idOperacao = (int)VerificacaoParameter.Operacao.Create,
                        VerificacaoJSON = JsonConvert.SerializeObject(new Verificacao()
                        {
                            deVerificacao = txtdeVerificacao.Text,
                            idTipoVerificacao  =  _listaTipoVerificacao.Where(tipoVerif
=>                            tipoVerif.deTipoVerificacao                                  ==
spnTipoVerificacao.SelectedItem.ToString()).FirstOrDefault().idTipoVerificacao,
                            idProduto = _listaProduto.Where(prod => prod.deProduto ==
spnProduto.SelectedItem.ToString()).FirstOrDefault().idProduto,
                            idGrupoUsuario   =   _listaGrupoUsuario.Where(grupoUsu   =>
grupoUsu.deGrupoUsuario                                                                  ==
spnGrupoUsuario.SelectedItem.ToString()).FirstOrDefault().idGrupoUsuario
                        })
                    };
                    var   uri   =   new   Uri(string.Format("{0}/api/Verificacao/Post",
AppSettings.ApiUrl));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Post
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        HttpRequestMessage           request           =           new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, uri);
                        request.Content = GetStringContent(parameter);
                        request.Headers.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                        response                          =                         await
Client.SendAsync(request).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        SetAlerta("Sucesso");
                        string stringResult = "";
                        try
                        {
                            await response.Content.ReadAsStringAsync().ContinueWith((ta
=>
                            {
                                if (ta.Exception == null)
```

```
                            {
                                stringResult = ta.Result;
                            }
                            else
                            {
                                SetAlerta(ta.Exception.Message);
                            };
                        })); ;
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    var                         proxyUsuario                         =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                    if (proxyUsuario.status == ResponseStatus.Success)
                        _verificacao = proxyUsuario.data;
                    else
                        SetAlerta(proxyUsuario.message.description);
                    if (_verificacao.idVerificacao > 0)
                        StartActivity(typeof(MenuAdministracaoActivity));
                    else
                        SetAlerta("Ocorreu um erro.");
                }
                else
                    SetAlerta(response.StatusCode.ToString());
            }
        }
        catch (Exception ex) { SetAlerta(ex.Message); }
    }

    private void SetAlerta(string v)
    {
        Toast toast = Toast.MakeText(this, v, ToastLength.Short);
        toast.Show();
    }

    public string GetQueryString(object obj)
    {
        var properties = from p in obj.GetType().GetRuntimeProperties()
                         where p.GetValue(obj, null) != null
                         select p.Name + "=" + WebUtility.UrlEncode(p.GetValue(obj,
null).ToString());

        return String.Join("&", properties.ToArray());
    }
    private StringContent GetStringContent(object obj)
    {
        var jsonObject = JsonConvert.SerializeObject(obj);
        StringContent   result   =   new   StringContent(jsonObject,   Encoding.UTF8,
"application/json");
        return result;
    }
}
```

- Projeto API.Teste06:
    - Classe ListaAtividadesActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android.App;
using Android.Widget;
using Android.OS;
using System.Linq;
using System.Net.Http;
using System;
using static API.Core.ConfigurationManager;
using API.Data.DTO;
using System.Threading.Tasks;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using System.Collections.Generic;
using System.Reflection;
using API.Parameters;
using System.Net;
using Android.Content;
using Android.Graphics;

namespace App06Teste
{
    [Activity(Label = "Lista de Atividades")]
    class ListaAtividadesActivity : Activity
    {
        LinearLayout linearLayout;
        Button btnCarregarLista;
        List<Verificacao> listaVerificacao;
        List<Produto> listaProduto;
        protected async override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.ListaAtividades);
            linearLayout                                                    =
FindViewById<LinearLayout>(Resource.Id.linearLayout1);


        }

        protected async override void OnResume()
        {
            base.OnResume();

            int idProduto = 0;
            int.TryParse(Intent.GetStringExtra("idProduto"), out idProduto);


            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
```

```
                    VerificacaoParameter parameter = new VerificacaoParameter()
                    {
                        idOperacao                                              =
(int)VerificacaoParameter.Operacao.GetListByGrupoUsuario,
                        idGrupoUsuario                                          =
Session.Usuario.GrupoUsuario.FirstOrDefault().idGrupoUsuario,
                        includeProperties = "Produto"
                    };
                    var                uri                =                 new
Uri(string.Format("{0}/api/Verificacao/Get?{1}",           AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            stringResult                =                await
response.Content.ReadAsStringAsync();
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                    proxyUsuario                    =
JsonConvert.DeserializeObject<ResponseProxy<List<Verificacao>>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            listaVerificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
                foreach (var verificao in listaVerificacao)
                {
                    if (idProduto > 0)
                    {
                        if (verificao.Produto.idProduto == idProduto)
                        {
```

```
                            Button            btnNew            =            new
Button(base.ApplicationContext);
                            btnNew.Text = verificao.deVerificacao;
                            btnNew.Id = verificao.idVerificacao;
                            btnNew.TextAlignment                                =
Android.Views.TextAlignment.Center;
                            btnNew.TextSize = 35;
                            btnNew.SetTextColor(Color.White);

                            btnNew.Click   +=   (senders,   EventArgs)   =>   {
btnNew_Click(senders, EventArgs, verificao.idVerificacao); };
                            linearLayout.AddView(btnNew);
                        }
                    }
                    else
                    {
                        Button btnNew = new Button(base.ApplicationContext);
                        btnNew.Text = verificao.deVerificacao;
                        btnNew.Id = verificao.idVerificacao;
                        btnNew.TextAlignment                                    =
Android.Views.TextAlignment.Center;
                        btnNew.TextSize = 35;
                        btnNew.SetTextColor(Color.White);

                        btnNew.Click    +=    (senders,    EventArgs)    =>    {
btnNew_Click(senders, EventArgs, verificao.idVerificacao); };
                        linearLayout.AddView(btnNew);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        protected   void   btnNew_Click(object   sender,   EventArgs   e,   int
idVerificacao)
        {
            Intent intent = new Intent(this, typeof(VerificacaoActivity));
            intent.PutExtra("idVerificacao", idVerificacao);
            StartActivity(intent);
            //SetAlerta(idVerificacao.ToString());
        }

        private void SetAlerta(string v)
        {
            Toast toast = Toast.MakeText(this, v, ToastLength.Short);
            toast.Show();
        }

        public string GetQueryString(object obj)
        {
            var properties = from p in obj.GetType().GetRuntimeProperties()
                             where p.GetValue(obj, null) != null
                             select        p.Name        +        "="        +
WebUtility.UrlEncode(p.GetValue(obj, null).ToString());

            return String.Join("&", properties.ToArray());
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe ListaProdutosActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using System.Net.Http;
using API.Parameters;
using static API.Core.ConfigurationManager;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using API.Data.DTO;
using System.Net;
using System.Reflection;
using Android.Graphics;

namespace App06Teste
{
    [Activity(Label = "Lista de Produtos")]
    public class ListaProdutosActivity : Activity
    {
        LinearLayout linearLayoutListaProdutos;
        List<Verificacao> _listaVerificacao;
        List<Produto> _listaProduto;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.ListaProdutos);

            linearLayoutListaProdutos                                                    =
FindViewById<LinearLayout>(Resource.Id.linearLayoutListaProdutos);
            // Create your application here
        }

        protected async override void OnResume()
        {
            base.OnResume();

            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new VerificacaoParameter()
```

```
                    {
                        idOperacao                                                  =
(int)VerificacaoParameter.Operacao.GetListByGrupoUsuario,
                        idGrupoUsuario                                              =
Session.Usuario.GrupoUsuario.FirstOrDefault().idGrupoUsuario,
                        includeProperties = "Produto"
                    };
                    var  uri  =  new  Uri(string.Format("{0}/api/Verificacao/Get?{1}",
AppSettings.ApiUrl, GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            await response.Content.ReadAsStringAsync().ContinueWith((ta
=>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message);
                                }
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                         proxyUsuario                        =
JsonConvert.DeserializeObject<ResponseProxy<List<Verificacao>>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _listaVerificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
```

```
                }
                if (_listaProduto == null)
                    _listaProduto = new List<Produto>();
                foreach (var verificao in _listaVerificacao)
                {
                    _listaProduto.Add(verificao.Produto);
                }
                foreach (var produto in _listaProduto.Select(prod => new {
prod.idProduto, prod.deProduto}).Distinct())
                {
                    Button btnNew = new Button(base.ApplicationContext);
                    btnNew.Text = produto.deProduto;
                    btnNew.Id = produto.idProduto;
                    btnNew.TextAlignment = Android.Views.TextAlignment.Center;
                    btnNew.TextSize = 35;
                    btnNew.SetTextColor(Color.White);

                    btnNew.Click += (senders, EventArgs) => { btnNew_Click(senders,
EventArgs, produto.idProduto); };
                    linearLayoutListaProdutos.AddView(btnNew);
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        private void btnNew_Click(object senders, EventArgs eventArgs, int idProduto)
        {
            Intent intent = new Intent(this, typeof(ListaAtividadesActivity));
            intent.PutExtra("idProduto", idProduto.ToString());
            StartActivity(intent);
        }

        private void SetAlerta(string v)
        {
            Toast toast = Toast.MakeText(this, v, ToastLength.Short);
            toast.Show();
        }

        public string GetQueryString(object obj)
        {
            var properties = from p in obj.GetType().GetRuntimeProperties()
                             where p.GetValue(obj, null) != null
                             select p.Name + "=" + WebUtility.UrlEncode(p.GetValue(obj,
null).ToString());

            return String.Join("&", properties.ToArray());
        }
    }
        }
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe LoginActivity;

```
using Android.App;
using Android.Widget;
using Android.OS;
```

```
using System.Linq;
using System.Net.Http;
using System;
using static API.Core.ConfigurationManager;
using API.Data.DTO;
using System.Threading.Tasks;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using System.Collections.Generic;
using System.Reflection;
using API.Parameters;
using System.Net;
using Android.Content;

namespace App06Teste
{
    [Activity(Label = "Login", MainLauncher = true, Icon = "@drawable/icon")]
    public class LoginActivity : Activity
    {
        EditText txtedEmail;
        EditText txtcoSenha;
        Button btnLogin;
        TextView txtnmUsuario;
        TextView txtdeGrupo;
        Usuario usuario;
        Intent intent;
        protected override void OnCreate(Bundle bundle)
        {
            base.OnCreate(bundle);
            SetContentView(Resource.Layout.Login);
            btnLogin = FindViewById<Button>(Resource.Id.btnLogin);
            btnLogin.Click += BtnLogin_Click;

        }

        private async void BtnLogin_Click(object sender, System.EventArgs e)
        {
            txtedEmail = FindViewById<EditText>(Resource.Id.txtedEmail);
            txtcoSenha = FindViewById<EditText>(Resource.Id.txtcoSenha);
            txtnmUsuario = FindViewById<TextView>(Resource.Id.txtnmUsuario);
            txtdeGrupo = FindViewById<TextView>(Resource.Id.txtdeGrupoUsuario);
            if (string.IsNullOrEmpty(txtedEmail.Text))
            {
                SetAlert("É necessário preencher o e-mail.");
                return;
            }
            if (string.IsNullOrEmpty(txtcoSenha.Text))
            {
                SetAlert("É necessário preencher a senha.");
            }

            var            usuario            =            await            new
API.Services.UsuarioAPIServices(API.Parameters.UsuarioParameter.Operacao.GetBy
EmailSenha,  null,  txtedEmail.Text,  txtcoSenha.Text,  "GrupoUsuario").Get(new
API.Parameters.UsuarioParameter()
            {
                idOperacao                                                    =
(int)API.Parameters.UsuarioParameter.Operacao.GetByEmailSenha,
                edEmail = txtedEmail.Text,
                coSenha = txtcoSenha.Text,
                includeProperties = "GrupoUsuario"
```

```
            });

            if (usuario != new Usuario())
            {
                if (usuario.GrupoUsuario.Any())
                {
                    Session.Usuario = usuario;
                    txtdeGrupo.Text                                          =
usuario.GrupoUsuario.FirstOrDefault().deGrupoUsuario;
                    if (usuario.icAdministrador)
                    {
                        intent          =          new          Intent(this,
typeof(MenuAdministracaoActivity));
                    }
                    else
                    {
                        intent          =          new          Intent(this,
typeof(ListaAtividadesActivity));
                    }
                    StartActivity(intent);
                }
                else
                    txtdeGrupo.Text = "Não está em nenhum grupo";
                txtnmUsuario.Text = usuario.nmUsuario;
            }
        }

        protected void SetAlert(string alerta)
        {
            Toast toast = Toast.MakeText(this, alerta, ToastLength.Long);
            toast.Show();
            return;
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe MainViewModel;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using XLabs.Platform.Device;

namespace App06Teste
{
    public class MainViewModel : XLabs.Forms.Mvvm.ViewModel
    {
```

```
        private readonly IDevice _device;
        private string _message;

        public MainViewModel(IDevice device)
        {
            _device = device;
            Message = String.Format("Hello Xamarin Forms Labs MVVM Basics!! How
is your {0} device", device.Manufacturer);
        }

        public string Message
        {
            get { return _message; }
            set { SetProperty(ref _message, value); }
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe Media;

```
using System;
using XLabs.Platform.Services.Media;
using XLabs.Ioc;
using XLabs.Platform.Device;
using Xamarin.Forms;

namespace App06Teste
{
    public static class Media
    {
        static IMediaPicker mediaPicker = null;
        public static IMediaPicker MediaPicker
        {
            get
            {
                if (mediaPicker == null)
                {
                    var device = Resolver.Resolve<IDevice>();
                    mediaPicker   =   DependencyService.Get<IMediaPicker>()   ??
device.MediaPicker;
                    if      (mediaPicker      ==      null)      throw      new
NullReferenceException("MediaPicker DependencyService.Get error");
                }

                return mediaPicker;
            }
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):

- Classe MenuAdministracaoActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;

namespace App06Teste
{
    [Activity(Label = "MenuAdministracaoActivity")]
    public class MenuAdministracaoActivity : Activity
    {
        Button btnGerenciarUsuario;
        Button btnCriarProduto;
        Button btnCriarSetor;
        Button btnCriarVerificacao;
        Button btnListaAtividades;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.MenuAdministracao);

            // Create your application here
            btnGerenciarUsuario                                                 =
FindViewById<Button>(Resource.Id.btnGerenciarUsuario);
            btnCriarProduto                                                     =
FindViewById<Button>(Resource.Id.btnCriarProduto);
            btnCriarSetor = FindViewById<Button>(Resource.Id.btnCriarSetor);
            btnCriarVerificacao                                                 =
FindViewById<Button>(Resource.Id.btnCriarVerificacao);
            btnListaAtividades                                                  =
FindViewById<Button>(Resource.Id.btnListaAtividades);

            btnGerenciarUsuario.Click += BtnGerenciarUsuario_Click;
            btnCriarProduto.Click += BtnCriarProduto_Click;
            btnCriarSetor.Click += BtnCriarSetor_Click;
            btnCriarVerificacao.Click += BtnCriarVerificacao_Click;
            btnListaAtividades.Click += BtnListaAtividades_Click;
        }

        private void BtnCriarVerificacao_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(CriarVerificacaoActivity));
        }

        private void BtnCriarSetor_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(CriarSetorActivity));
        }

        private void BtnCriarProduto_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(CriarProdutoActivity));
        }
```

```
        private void BtnGerenciarUsuario_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(GerenciarListaUsuarioActivity));
        }

        private void BtnListaAtividades_Click(object sender, EventArgs e)
        {
            StartActivity(typeof(ListaProdutosActivity));
        }
    }
}
```

- Projeto App06Teste (Representa o ControleShopping.AndroidApp):
    - Classe VerificacaoActivity;

```
using System;
using System.Collections.Generic;
using System.Linq;
using Android.App;
using Android.Widget;
using Android.OS;
using System.Net.Http;
using static API.Core.ConfigurationManager;
using API.Data.DTO;
using System.Threading.Tasks;
using Newtonsoft.Json;
using API.HttpClient.Proxy;
using System.Reflection;
using API.Parameters;
using System.Net;
using Android.Content;
using Android.Graphics;
using Android.Views;
using Android.Provider;
using Android.Content.PM;
using Java.IO;
using System.Globalization;

namespace App06Teste
{
    [Activity(Label = "VerificacaoActivity")]
    public class VerificacaoActivity : Activity
    {
        ImageView imgView;
        Button btnScan;
        Button btnTirarFoto;
        Button btnSalvar;
        Switch switchProblema;
        private static Verificacao _verificacao;
        EditText _txtProblema;
        private static ProblemaVerificacao _problemaVerificacao;

        protected async override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
```

```
            SetContentView(Resource.Layout.Verificacao);
            _verificacao = new Verificacao();
            // Create your application here
            try
            {
                ZXing.Mobile.MobileBarcodeScanner.Initialize(Application);
                //Create your application here


                btnScan = FindViewById<Button>(Resource.Id.btnScan);
                btnTirarFoto = FindViewById<Button>(Resource.Id.btnFoto);
                btnSalvar = FindViewById<Button>(Resource.Id.btnSalvar);
                switchProblema                                                =
FindViewById<Switch>(Resource.Id.switchProblema);
                imgView                                                       =
FindViewById<ImageView>(Resource.Id.imgQRVerificacao);
                _txtProblema = FindViewById<EditText>(Resource.Id.txtProblema);

                _txtProblema.TextChanged += _txtProblema_TextChanged;

                btnScan.Click += btnScan_Click;
                btnTirarFoto.Click += btnTirarFoto_Click;
                btnSalvar.Click += btnSalvar_Click;
                switchProblema.CheckedChange += switchProblema_CheckedChange;
            }
            catch (Exception e) { SetAlerta(e.Message); }
            if (IsThereAnAppToTakePictures())
            {
                CreateDirectoryForPictures();
            }
        }

        private       void       _txtProblema_TextChanged(object       sender,
Android.Text.TextChangedEventArgs e)
        {
            btnSalvar.Enabled = true;
        }

        protected async override void OnResume()
        {
            base.OnResume();
            int idVerificacao = Intent.GetIntExtra("idVerificacao", 0);

            if (idVerificacao == 0)
            {
                base.OnBackPressed();
            }
            if (_verificacao == null)
                _verificacao = new Verificacao();
            _verificacao.idVerificacao = idVerificacao;
            if (_problemaVerificacao == null)
                _problemaVerificacao = new ProblemaVerificacao();
            _problemaVerificacao.idVerificacao = idVerificacao;
            await GetVerificacao(idVerificacao);
        }

        private      void      switchProblema_CheckedChange(object      sender,
CompoundButton.CheckedChangeEventArgs e)
        {
            switchProblema = (Switch)sender;
```

```
            if (switchProblema.Checked)
            {
                btnSalvar.Enabled = false;
                _txtProblema.Visibility = ViewStates.Visible;
                btnTirarFoto.Visibility = ViewStates.Visible;
            }
            else
            {
                btnSalvar.Enabled = true;
                _txtProblema.Visibility = ViewStates.Invisible;
                btnTirarFoto.Visibility = ViewStates.Invisible;
            }
        }

        private async void btnSalvar_Click(object sender, EventArgs e)
        {
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    HistoricoVerificacaoParameter     parameter     =     new
HistoricoVerificacaoParameter()
                    {
                        idOperacao                                          =
(int)HistoricoVerificacaoParameter.Operacao.Create,
                        HistoricoVerificacaoJSON                            =
JsonConvert.SerializeObject(new HistoricoVerificacao()
                        {
                            dtVerificacao = DateTime.Now,
                            idUsuario = Session.Usuario.idUsuario,
                            idVerificacao = _problemaVerificacao.idVerificacao
                        })
                    };
                    var                uri                =                new
Uri(string.Format("{0}/api/HistoricoVerificacao/Post", AppSettings.ApiUrl));

                    var requestMessage = new HttpRequestMessage()
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Post
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        HttpRequestMessage        request        =        new
HttpRequestMessage(System.Net.Http.HttpMethod.Post, uri);
                        request.Content = GetStringContent(parameter);
                        request.Headers.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                        response                    =                    await
Client.SendAsync(request).ConfigureAwait(true);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    } catch (Exception ex)
```

```
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        SetAlerta("Sucesso");
                        string stringResult = "";
                        try
                        {
                            await
response.Content.ReadAsStringAsync().ContinueWith((ta =>
                            {
                                if (ta.Exception == null)
                                {
                                    stringResult = ta.Result;
                                }
                                else
                                {
                                    SetAlerta(ta.Exception.Message           +
ta.Exception.StackTrace);
                                };
                            })); ;
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                    proxyUsuario                    =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _verificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                        if     (_verificacao.idVerificacao     >     0     &&
!switchProblema.Selected)
                            StartActivity(typeof(MenuAdministracaoActivity));
                    }
                    else
                        SetAlerta(response.StatusCode.ToString());
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }

            if (switchProblema.Selected)
            {
                SetAlerta("Entrou");
                try
                {
                    using (var client = new HttpClient())
                    {
                       using (var content =
                            new MultipartFormDataContent())
                        {
                            content.Add(new                    StreamContent(new
System.IO.MemoryStream(_problemaVerificacao.imgProblema)),         "bilddatei",
"upload.jpg");
```

```
                            ProblemaVerificacaoParameter   parameter   =   new
ProblemaVerificacaoParameter()
                            {
                                idOperacao                                  =
(int)ProblemaVerificacaoParameter.Operacao.Create,
                                ProblemaVerificacaoJSON                     =
JsonConvert.SerializeObject(new ProblemaVerificacao()
                                {
                                    deProblemaVerificacao                   =
_problemaVerificacao.deProblemaVerificacao,
                                    idVerificacao                           =
_problemaVerificacao.idVerificacao
                                })
                            };

                            content.Add(GetStringContent(parameter));

                            using (
                        var message =
                            await
client.PostAsync(string.Format("{0}/api/ProblemaVerificacao/Post",
AppSettings.ApiUrl), content))
                            {
                                var           input           =           await
message.Content.ReadAsStringAsync();
                                var                  _proble                =
JsonConvert.DeserializeObject<ResponseProxy<ProblemaVerificacao>>(input);
                                if (_proble.status == ResponseStatus.Success)
                                    _problemaVerificacao = _proble.data;
                                else
                                    SetAlerta(_proble.message.description);
                                if  (_problemaVerificacao.idProblemaVerificacao
> 0)

StartActivity(typeof(MenuAdministracaoActivity));
                            }
                        }
                    }
                }
                catch (Exception ex) { throw ex; }
            }
        }

        private void btnTirarFoto_Click(object sender, EventArgs e)
        {
            if (_problemaVerificacao == null)
                _problemaVerificacao = new ProblemaVerificacao();
            _problemaVerificacao.deProblemaVerificacao = _txtProblema.Text;
            try
            {

                Intent intent = new Intent(MediaStore.ActionImageCapture);
                App._file           =           new           File(App._dir,
string.Format("{1}/_verificacao.${2}_{0}.jpg",
DateTime.Now.ToString("dd/MM/yy:hhmmss"),          _verificacao.deVerificacao,
_verificacao.idVerificacao));
                intent.PutExtra(MediaStore.ExtraOutput,
Android.Net.Uri.FromFile(App._file));
                StartActivityForResult(intent, 0);
            }
            catch (TaskCanceledException)
```

```
            {
                SetAlerta("TakePhoto cancelled");
            }
            catch (Exception ex)
            {
                SetAlerta(ex.Message + "\n" + ex.StackTrace);
            }
        }

        protected async override void OnActivityResult(int requestCode, Result
resultCode, Intent data)
        {
            base.OnActivityResult(requestCode, resultCode, data);

            if (resultCode == Result.Ok)
            {
                Android.Net.Uri                selectedImage                =
Android.Net.Uri.FromFile(App._file);
                Bitmap bitmap;

                try
                {
                    int height = Resources.DisplayMetrics.HeightPixels;
                    int width = imgView.Height;
                    BitmapFactory.Options options = new BitmapFactory.Options {
InJustDecodeBounds = true };


                    int outHeight = options.OutHeight;
                    int outWidth = options.OutWidth;
                    int inSampleSize = 1;

                    if (outHeight > height || outWidth > width)
                    {
                        inSampleSize = outWidth > outHeight
                                        ? outHeight / height
                                        : outWidth / width;
                    }


                    options.InSampleSize = inSampleSize;
                    options.InJustDecodeBounds = false;
                    Bitmap                    resizedBitmap                    =
BitmapFactory.DecodeFile(App._file.Path, options);

                    if (resizedBitmap == null || resizedBitmap.ByteCount == 0)
                        SetAlerta("Bitmap Lixo");
                    else
                        SetAlerta(resizedBitmap.ByteCount.ToString());
                    byte[] bitmapData;
                    using (var stream = new System.IO.MemoryStream())
                    {
                        resizedBitmap.Compress(Bitmap.CompressFormat.Jpeg,
100, stream);
                        bitmapData = stream.ToArray();
                    }
                    if (_problemaVerificacao == null)
                        _problemaVerificacao = new ProblemaVerificacao();

                    _problemaVerificacao.imgProblema = bitmapData;
```

```
                    SetAlerta("Imagem Salva com sucesso");
                    SetAlerta(_problemaVerificacao.idVerificacao.ToString());

                    switchProblema.Visibility = ViewStates.Visible;
                    switchProblema.Selected = true;
                    btnSalvar.Enabled = true;

                    Toast.MakeText(this, selectedImage.ToString(),
                            ToastLength.Long).Show();
                }
                catch (Exception e)
                {
                    Toast.MakeText(this, e.Message, ToastLength.Long)
                            .Show();
                }
                GC.Collect();
            }
        }

        private async void btnScan_Click(object sender, EventArgs e)
        {
            try
            {

var scanner = new ZXing.Mobile.MobileBarcodeScanner();
                var result = await scanner.Scan();

                if (result != null)
                {
                    SetAlerta("Scanned Barcode: " + result.Text);
                    int idProdutoBarCode = 0;
                    int.TryParse(result.Text, out idProdutoBarCode);
                    if (_verificacao.idProduto != idProdutoBarCode)
                        base.OnBackPressed();
                    switchProblema.Visibility = ViewStates.Visible;
                    btnSalvar.Enabled = true;
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        protected async Task GetVerificacao(int idVerificacao)
        {
            try
            {
                using (HttpClient Client = new HttpClient())
                {
                    Client.DefaultRequestHeaders.Accept.Add(new
System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json"));
                    Client.MaxResponseContentBufferSize = 256000;
                    VerificacaoParameter parameter = new VerificacaoParameter()
                    {
                        idOperacao = (int)VerificacaoParameter.Operacao.Get,
                        idVerificacao = idVerificacao
                    };
                    var                uri                =                new
Uri(string.Format("{0}/api/Verificacao/Get?{1}",            AppSettings.ApiUrl,
GetQueryString(parameter)));

                    var requestMessage = new HttpRequestMessage()
```

```
                    {
                        RequestUri = uri,
                        Method = HttpMethod.Get
                    };
                    var response = new HttpResponseMessage();
                    try
                    {
                        response = await Client.GetAsync(uri);
                    }
                    catch (HttpRequestException HRex)
                    {
                        SetAlerta(HRex.Message + HRex.StackTrace);
                    }
                    catch (Exception ex)
                    {
                        SetAlerta(ex.Message + ex.StackTrace);
                    }
                    if (response.IsSuccessStatusCode)
                    {
                        string stringResult = "";
                        try
                        {
                            stringResult                =                await
response.Content.ReadAsStringAsync();
                        }
                        catch (HttpRequestException HRex)
                        {
                            SetAlerta(HRex.Message + HRex.StackTrace);
                        }
                        catch (Exception ex)
                        {
                            SetAlerta(ex.Message + ex.StackTrace);
                        }
                        var                    proxyUsuario                    =
JsonConvert.DeserializeObject<ResponseProxy<Verificacao>>(stringResult);
                        if (proxyUsuario.status == ResponseStatus.Success)
                            _verificacao = proxyUsuario.data;
                        else
                            SetAlerta(proxyUsuario.message.description);
                    }
                }
            }
            catch (Exception ex) { SetAlerta(ex.Message); }
        }

        private void SetAlerta(string message)
        {
            Toast.MakeText(this, message, ToastLength.Long).Show();
        }

        private object GetQueryString(VerificacaoParameter parameter)
        {
            var         properties         =         from         p         in
parameter.GetType().GetRuntimeProperties()
                          where p.GetValue(parameter, null) != null
                          select        p.Name        +        "="        +
WebUtility.UrlEncode(p.GetValue(parameter, null).ToString());

            return String.Join("&", properties.ToArray());
        }
        private void CreateDirectoryForPictures()
```

```
        {
            App._dir = new File(
                Android.OS.Environment.GetExternalStoragePublicDirectory(
                    Android.OS.Environment.DirectoryPictures),
"CameraAppDemo");
            if (!App._dir.Exists())
            {
                App._dir.Mkdirs();
            }
        }

        private bool IsThereAnAppToTakePictures()
        {
            Intent intent = new Intent(MediaStore.ActionImageCapture);
            IList<ResolveInfo> availableActivities =
                PackageManager.QueryIntentActivities(intent,
PackageInfoFlags.MatchDefaultOnly);
            return availableActivities != null && availableActivities.Count >
0;
        }

        public static class App
        {
            public static File _file;
            public static File _dir;
            public static Bitmap bitmap;
        }

        private StringContent GetStringContent(object obj)
        {
            var jsonObject = JsonConvert.SerializeObject(obj);
            StringContent    result    =    new    StringContent(jsonObject,
System.Text.Encoding.UTF8, "application/json");
            return result;
        }
    }
}
```

- Projeto ControleShopping.API:
    - Classe VerificacaoActivity;

```
using System;
using System.Web;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Validation;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Reflection;
using System.Web.Http;
using System.IO;
using API.HttpClient.Proxy;
using API.Core;
using log4net;
using ControleShopping.API.Infra;
using System.Web.Http.OData;
```

```
namespace ControleShopping.API
{
    public class ApiControllerBase : ApiController
    {
        protected object responseResult { get; set; }
        protected int nuQtdRegistros { get; set; }

        protected HttpResponseMessage GetBadRequestResponse()
        {
            return   Request.CreateResponse(HttpStatusCode.BadRequest,   new
ResponseProxy<ResponseMessageProxy>
            {
                status = ResponseStatus.Error,
                message = new ResponseMessageProxy
                {
                    id = "-2",
                    description = "Solicitação Inválida."
                }
            });
        }

        protected HttpResponseMessage GetNotFoundResponse()
        {
            return    Request.CreateResponse(HttpStatusCode.NotFound,    new
ResponseProxy<ResponseMessageProxy>
            {
                status = ResponseStatus.NotFound,
                message = new ResponseMessageProxy
                {
                    id = "-5",
                    description = "Não encontrado."
                }
            });
        }

        protected HttpResponseMessage GetOkResponse()
        {
            return       Request.CreateResponse(HttpStatusCode.OK,       new
ResponseProxy<NullObject>
            {
                status = ResponseStatus.Success,
                data = null,
                total = 0
            });
        }

        protected  HttpResponseMessage  GetNotFoundResponse<T>(T  dataProxy)
where T : class
        {
            return       Request.CreateResponse(HttpStatusCode.OK,       new
ResponseProxy<T>
            {
                status = ResponseStatus.NotFound,
                data = dataProxy,
                total = 0,
            });
        }

        protected  HttpResponseMessage  GetOkResponse<T>(T  dataProxy,  int
total = 0) where T : class
```

```
        {
            return      Request.CreateResponse(HttpStatusCode.OK,      new
ResponseProxy<T>
            {
                status = ResponseStatus.Success,
                data = dataProxy,
                total = total,
            });
        }

        protected HttpResponseMessage GetDeleteResponse()
        {
            return      Request.CreateResponse(HttpStatusCode.OK,      new
ResponseProxy<object>
            {
                status = ResponseStatus.Success
            });
        }

        protected string GetEventoLog(object logArguments)
        {
            var properties =
                logArguments.GetType()
                    .GetProperties(BindingFlags.Instance                    |
BindingFlags.Public | BindingFlags.InvokeMethod);

            var propertiesToJoin =
                properties.Select(
                    property =>
                        string.Format("{0}:{1}", property.Name,

logArguments.GetType().GetProperty(property.Name).GetValue(logArguments,
null)))
                    .ToList();

            return string.Join(Environment.NewLine, propertiesToJoin);
        }

        protected void LogEvento(string evento, Exception excessao)
        {
            ILog log = LogManager.GetLogger("LogApi");
            if (log.IsErrorEnabled)
            {
                var parametosRequisicao = new
                {
                    SESSION_ID                                              =
HttpContext.Current.Request.Headers.Get("Cookie") + Environment.NewLine,
                    URL      =      HttpContext.Current.Request.Url      +
Environment.NewLine,
                    QUERY_STRING  =  HttpContext.Current.Request.QueryString
+ Environment.NewLine,
                    FILE_PATH   =   HttpContext.Current.Request.FilePath   +
Environment.NewLine,
                    HOST_NAME  =  HttpContext.Current.Server.MachineName  +
Environment.NewLine,
                    USER_AGENT  =  HttpContext.Current.Request.UserAgent  +
Environment.NewLine,
                    EXCESSAO_SERVIDOR                                       =
HttpContext.Current.Server.GetLastError() + Environment.NewLine,
                    EXCESSAO_APLICACAO = excessao + Environment.NewLine,
                    PARAMETROS_PESQUIA = evento + Environment.NewLine,
```

```
                };
                log.Error(parametosRequisicao);
            }
        }
```

- Projeto ControleShopping.API:
    - Classe JobRelatorio;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Threading;
using System.Net.Mail;
using System.Net;
using ControleShopping.API.Models;
using System.Text;

namespace ControleShopping.API
{
    public class JobRelatorio
    {
        private ControleShoppingEntities db = new ControleShoppingEntities();
        private System.Threading.Timer timer;
        private bool icJaFoi;
        public void SetUpTimer(TimeSpan alertTime)
        {
            DateTime current = DateTime.Now;
            TimeSpan timeToGo = alertTime - current.TimeOfDay;
            if (timeToGo < TimeSpan.Zero)
            {
                icJaFoi = false;
                return;//time already passed
            }
            if (!icJaFoi)
            {
                this.timer = new System.Threading.Timer(x =>
                {
                    this.EnviarRelatorio();
                    icJaFoi = true;
                }, null, timeToGo, Timeout.InfiniteTimeSpan);
            }
        }

        private void EnviarRelatorio()
        {
            var client = new SmtpClient("smtp.gmail.com", 587)
            {
                Credentials = new NetworkCredential("Fillipemoura2@gmail.com",
"Fillipe@)20"),
                EnableSsl = true
            };
            client.Send("Fillipemoura2@gmail.com",   "Fillipemoura2@gmail.com",
"Relatório do dia " + DateTime.Now.ToString("dd/MM/yy"), GetTextRelatorio());

        }
```

```
        private string GetTextRelatorio()
        {
            var    historico    =    db.HistoricoVerificacao.Where(histo    =>
histo.dtVerificacao <= DateTime.Now.AddDays(-1));
            var texto = new StringBuilder();
            texto.Append("<table>");
            texto.Append("<tr>");
            texto.Append("<th>Verificação</th>");
            texto.Append("<th>Data da Verificação</th>");
            texto.Append("<th>Existe problema</th>");
            texto.Append("</tr>");
            foreach (var verificacao in historico)
            {
                texto.Append("<tr>");
                texto.Append("<td>"  +  verificacao.Verificacao.deVerificacao  +
"</td>");
                texto.Append("<td>" + verificacao.dtVerificacao + "</td>");
                texto.Append("<td>"                                              +
verificacao.Verificacao.ProblemaVerificacao.Any(problema                        =>
problema.dtProblemaVerificacao >= DateTime.Now.AddDays(-1)) + "</td>");
                texto.Append("</tr>");
            }
            texto.Append("</table>");
            return texto.ToString();
        }
    }
  }
```

- Projeto ControleShopping.API:
    - Classe AccountController;

```
using System;
using System.Collections.Generic;
using System.Net.Http;
using System.Security.Claims;
using System.Security.Cryptography;
using System.Threading.Tasks;
using System.Web;
using System.Web.Http;
using System.Web.Http.ModelBinding;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.EntityFramework;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using Microsoft.Owin.Security.Cookies;
using Microsoft.Owin.Security.OAuth;
using ControleShopping.API.Models;
using ControleShopping.API.Providers;
using ControleShopping.API.Results;

namespace ControleShopping.API.Controllers
{
    [Authorize]
    [RoutePrefix("api/Account")]
    public class AccountController : ApiController
```

```
    {
        private const string LocalLoginProvider = "Local";
        private ApplicationUserManager _userManager;

        public AccountController()
        {
        }

        public AccountController(ApplicationUserManager userManager,
            ISecureDataFormat<AuthenticationTicket> accessTokenFormat)
        {
            UserManager = userManager;
            AccessTokenFormat = accessTokenFormat;
        }

        public ApplicationUserManager UserManager
        {
            get
            {
                return                      _userManager                    ??
Request.GetOwinContext().GetUserManager<ApplicationUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        public ISecureDataFormat<AuthenticationTicket> AccessTokenFormat { get;
private set; }

        // GET api/Account/UserInfo
        [HostAuthentication(DefaultAuthenticationTypes.ExternalBearer)]
        [Route("UserInfo")]
        public UserInfoViewModel GetUserInfo()
        {
            ExternalLoginData                 externalLogin                 =
ExternalLoginData.FromIdentity(User.Identity as ClaimsIdentity);

            return new UserInfoViewModel
            {
                Email = User.Identity.GetUserName(),
                HasRegistered = externalLogin == null,
                LoginProvider     =     externalLogin     !=     null     ?
externalLogin.LoginProvider : null
            };
        }

        // POST api/Account/Logout
        [Route("Logout")]
        public IHttpActionResult Logout()
        {

Authentication.SignOut(CookieAuthenticationDefaults.AuthenticationType);
            return Ok();
        }

        // GET api/Account/ManageInfo?returnUrl=%2F&generateState=true
        [Route("ManageInfo")]
        public async Task<ManageInfoViewModel> GetManageInfo(string returnUrl,
bool generateState = false)
```

```
        {
            IdentityUser               user               =               await
UserManager.FindByIdAsync(User.Identity.GetUserId());

            if (user == null)
            {
                return null;
            }

            List<UserLoginInfoViewModel>          logins          =          new
List<UserLoginInfoViewModel>();

            foreach (IdentityUserLogin linkedAccount in user.Logins)
            {
                logins.Add(new UserLoginInfoViewModel
                {
                    LoginProvider = linkedAccount.LoginProvider,
                    ProviderKey = linkedAccount.ProviderKey
                });
            }

            if (user.PasswordHash != null)
            {
                logins.Add(new UserLoginInfoViewModel
                {
                    LoginProvider = LocalLoginProvider,
                    ProviderKey = user.UserName,
                });
            }

            return new ManageInfoViewModel
            {
                LocalLoginProvider = LocalLoginProvider,
                Email = user.UserName,
                Logins = logins,
                ExternalLoginProviders      =      GetExternalLogins(returnUrl,
generateState)
            };
        }

        // POST api/Account/ChangePassword
        [Route("ChangePassword")]
        public                   async                  Task<IHttpActionResult>
ChangePassword(ChangePasswordBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult             result             =             await
UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword,
                model.NewPassword);

            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }
```

```
        // POST api/Account/SetPassword
        [Route("SetPassword")]
        public                  async                  Task<IHttpActionResult>
SetPassword(SetPasswordBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult             result             =             await
UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);

            if (!result.Succeeded)
            {
                return GetErrorResult(result);

        }

            return Ok();
        }

        // POST api/Account/AddExternalLogin
        [Route("AddExternalLogin")]
        public                  async                  Task<IHttpActionResult>
AddExternalLogin(AddExternalLoginBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            Authentication.SignOut(DefaultAuthenticationTypes.ExternalCookie);

            AuthenticationTicket                    ticket                    =
AccessTokenFormat.Unprotect(model.ExternalAccessToken);

            if (ticket == null || ticket.Identity == null || (ticket.Properties
!= null
                && ticket.Properties.ExpiresUtc.HasValue
                && ticket.Properties.ExpiresUtc.Value < DateTimeOffset.UtcNow))
            {
                return BadRequest("External login failure.");
            }

            ExternalLoginData                  externalData                  =
ExternalLoginData.FromIdentity(ticket.Identity);

            if (externalData == null)
            {
                return BadRequest("The external login is already associated with
an account.");
            }

            IdentityResult             result             =             await
UserManager.AddLoginAsync(User.Identity.GetUserId(),
                new                  UserLoginInfo(externalData.LoginProvider,
externalData.ProviderKey));
```

```
            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // POST api/Account/RemoveLogin
        [Route("RemoveLogin")]
        public                  async                  Task<IHttpActionResult>
RemoveLogin(RemoveLoginBindingModel model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            IdentityResult result;

            if (model.LoginProvider == LocalLoginProvider)
            {
                result                          =                          await
UserManager.RemovePasswordAsync(User.Identity.GetUserId());
            }
            else
            {
                result                          =                          await
UserManager.RemoveLoginAsync(User.Identity.GetUserId(),
                    new UserLoginInfo(model.LoginProvider, model.ProviderKey));
            }

            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // GET api/Account/ExternalLogin
        [OverrideAuthentication]
        [HostAuthentication(DefaultAuthenticationTypes.ExternalCookie)]
        [AllowAnonymous]
        [Route("ExternalLogin", Name = "ExternalLogin")]
        public async Task<IHttpActionResult> GetExternalLogin(string provider,
string error = null)
        {
            if (error != null)
            {
                return     Redirect(Url.Content("~/")     +     "#error="     +
Uri.EscapeDataString(error));
            }

            if (!User.Identity.IsAuthenticated)
            {
                return new ChallengeResult(provider, this);
            }

            ExternalLoginData                  externalLogin                  =
ExternalLoginData.FromIdentity(User.Identity as ClaimsIdentity);
```

```
            if (externalLogin == null)
            {
                return InternalServerError();
            }

            if (externalLogin.LoginProvider != provider)
            {

Authentication.SignOut(DefaultAuthenticationTypes.ExternalCookie);
                return new ChallengeResult(provider, this);
            }

            ApplicationUser    user    =    await    UserManager.FindAsync(new
UserLoginInfo(externalLogin.LoginProvider,
                externalLogin.ProviderKey));

            bool hasRegistered = user != null;

            if (hasRegistered)
            {

Authentication.SignOut(DefaultAuthenticationTypes.ExternalCookie);

                 ClaimsIdentity          oAuthIdentity          =          await
user.GenerateUserIdentityAsync(UserManager,
                    OAuthDefaults.AuthenticationType);
                ClaimsIdentity          cookieIdentity          =          await
user.GenerateUserIdentityAsync(UserManager,
                    CookieAuthenticationDefaults.AuthenticationType);

                AuthenticationProperties             properties             =
ApplicationOAuthProvider.CreateProperties(user.UserName);
                Authentication.SignIn(properties,               oAuthIdentity,
cookieIdentity);
            }
            else
            {
                IEnumerable<Claim> claims = externalLogin.GetClaims();
                ClaimsIdentity    identity    =    new    ClaimsIdentity(claims,
OAuthDefaults.AuthenticationType);
                Authentication.SignIn(identity);
            }

            return Ok();
        }

        // GET api/Account/ExternalLogins?returnUrl=%2F&generateState=true
        [AllowAnonymous]
        [Route("ExternalLogins")]
        public   IEnumerable<ExternalLoginViewModel>   GetExternalLogins(string
returnUrl, bool generateState = false)
        {
            IEnumerable<AuthenticationDescription>        descriptions        =
Authentication.GetExternalAuthenticationTypes();
            List<ExternalLoginViewModel>          logins          =          new
List<ExternalLoginViewModel>();

            string state;

            if (generateState)
```

```
            {
                const int strengthInBits = 256;
                state = RandomOAuthStateGenerator.Generate(strengthInBits);
            }
            else
            {
                state = null;
            }

            foreach (AuthenticationDescription description in descriptions)
            {
                ExternalLoginViewModel login = new ExternalLoginViewModel
                {
                    Name = description.Caption,
                    Url = Url.Route("ExternalLogin", new
                    {
                        provider = description.AuthenticationType,
                        response_type = "token",
                        client_id = Startup.PublicClientId,
                        redirect_uri     =     new     Uri(Request.RequestUri,
returnUrl).AbsoluteUri,
                        state = state
                    }),
                    State = state
                };
                logins.Add(login);
            }

            return logins;
        }

        // POST api/Account/Register
        [AllowAnonymous]
        [Route("Register")]
        public   async   Task<IHttpActionResult>   Register(RegisterBindingModel
model)
        {
            if (!ModelState.IsValid)
            {
                return BadRequest(ModelState);
            }

            var user = new ApplicationUser() { UserName = model.Email, Email =
model.Email };

            IdentityResult   result   =   await   UserManager.CreateAsync(user,
model.Password);

            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            return Ok();
        }

        // POST api/Account/RegisterExternal
        [OverrideAuthentication]
        [HostAuthentication(DefaultAuthenticationTypes.ExternalBearer)]
        [Route("RegisterExternal")]
```

```
        public                  async                  Task<IHttpActionResult>
RegisterExternal(RegisterExternalBindingModel model)
        {
            if (!ModelState.IsValid)
            {

        return BadRequest(ModelState);
    }

    var info = await Authentication.GetExternalLoginInfoAsync();
    if (info == null)
    {
        return InternalServerError();
    }

            var user = new ApplicationUser() { UserName = model.Email, Email =
model.Email };

            IdentityResult result = await UserManager.CreateAsync(user);
            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }

            result = await UserManager.AddLoginAsync(user.Id, info.Login);
            if (!result.Succeeded)
            {
                return GetErrorResult(result);
            }
            return Ok();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing && _userManager != null)
            {
                _userManager.Dispose();
                _userManager = null;
            }

            base.Dispose(disposing);
        }

        #region Helpers

        private IAuthenticationManager Authentication
        {
            get { return Request.GetOwinContext().Authentication; }
        }

        private IHttpActionResult GetErrorResult(IdentityResult result)
        {
            if (result == null)
            {
                return InternalServerError();
            }

            if (!result.Succeeded)
            {
                if (result.Errors != null)
```

```
                {
                    foreach (string error in result.Errors)
                    {
                        ModelState.AddModelError("", error);
                    }
                }

                if (ModelState.IsValid)
                {
                    // No ModelState errors are available to send, so just return
an empty BadRequest.
                    return BadRequest();
                }

                return BadRequest(ModelState);
            }

            return null;
        }

        private class ExternalLoginData
        {
            public string LoginProvider { get; set; }
            public string ProviderKey { get; set; }
            public string UserName { get; set; }

            public IList<Claim> GetClaims()
            {
                IList<Claim> claims = new List<Claim>();
                claims.Add(new  Claim(ClaimTypes.NameIdentifier,  ProviderKey,
null, LoginProvider));

                if (UserName != null)
                {
                    claims.Add(new   Claim(ClaimTypes.Name,   UserName,   null,
LoginProvider));
                }

                return claims;
            }

            public   static   ExternalLoginData   FromIdentity(ClaimsIdentity
identity)
            {
                if (identity == null)
                {
                    return null;
                }

                Claim                    providerKeyClaim                    =
identity.FindFirst(ClaimTypes.NameIdentifier);

                if         (providerKeyClaim         ==         null         ||
String.IsNullOrEmpty(providerKeyClaim.Issuer)
                    || String.IsNullOrEmpty(providerKeyClaim.Value))
                {
                    return null;
                }

                if (providerKeyClaim.Issuer == ClaimsIdentity.DefaultIssuer)
                {
```

```
                    return null;
                }

                return new ExternalLoginData
                {
                    LoginProvider = providerKeyClaim.Issuer,
                    ProviderKey = providerKeyClaim.Value,
                    UserName = identity.FindFirstValue(ClaimTypes.Name)
                };
            }
        }

        private static class RandomOAuthStateGenerator
        {
            private    static    RandomNumberGenerator    _random    =    new
RNGCryptoServiceProvider();

            public static string Generate(int strengthInBits)
            {
                const int bitsPerByte = 8;

                if (strengthInBits % bitsPerByte != 0)
                {
                    throw new ArgumentException("strengthInBits must be evenly
divisible by 8.", "strengthInBits");
                }

                int strengthInBytes = strengthInBits / bitsPerByte;

                byte[] data = new byte[strengthInBytes];
                _random.GetBytes(data);
                return HttpServerUtility.UrlTokenEncode(data);
            }
        }

        #endregion
    }
}
```