

FUNDAÇÃO FACULDADE FILOSOFIA CIÊNCIAS E
LETRAS DE MANDAGUARI

RAFAEL JOSÉ MATIA DE SÁ TELES

IAMHERE
APLICATIVO DE CONTROLE DE FREQUÊNCIA

MANDAGUARI-PR

2019

RAFAEL JOSÉ MATIA DE SÁ TELES

IAMHERE
APLICATIVO DE CONTROLE DE FREQUÊNCIA

Trabalho de conclusão de curso apresentado ao departamento de Informática da Fundação Faculdade de Filosofia, Ciências e Letra de Mandaguari, como parte dos requisitos necessários para a obtenção do título de Bacharel em Ciência da Computação, sob a orientação do Prof. Itamar Solopak.

MANDAGUARI-PR

2019

RAFAEL JOSÉ MATIA DE SÁ TELES

IAMHERE

APLICATIVO DE CONTROLE DE FREQUÊNCIA

Trabalho de conclusão de curso (TCC) apresentado ao departamento de Informática da Fundação Faculdade de Filosofia, Ciências e Letra de Mandaguari, como parte dos requisitos necessários para a obtenção do título de Bacharel em Ciência da Computação, sob a orientação do Prof. Itamar Solopak.

Mandaguari-PR, 08 de Agosto de 2019.

BANCA EXAMINADORA

Prof. Itamar Solopak

Prof. Munif Gebara Junior

Prof. Fernando Celestino Paschualetto

DEDICATÓRIA

Dedico este TCC primeiramente à Deus que me deu luz e sabedoria para escrever este trabalho.

Dedico também, a toda minha família, pelo incentivo e por acreditar em mim sempre, não permitindo que eu desistisse, mesmo nos momentos de maior dificuldade. O amor que vocês têm por mim é o que me estimula a lutar e vencer todos os dias.

Dedico também, a todo o corpo docente do curso de Bacharel em Ciências da Computação, que mais do que repassar conteúdos, ajudaram na minha formação de maneira enriquecedora. Vocês foram parte fundamental desta caminhada e são exemplos que eu quero levar para minha vida pessoal e profissional.

AGRADECIMENTOS

Agradeço a Deus acima de tudo, por ter me dado saúde, inteligência e força para superar às dificuldades, e conseguido concretizar mais esta etapa da minha vida.

Agradeço também aos meus pais, Agness Matia de Sá Teles e José de Sá Teles, e meu irmão João Pedro Matia de Sá Teles, que são a base da minha essência, os quais me apoiaram quando eu não tinha mais forças para continuar.

Em particular, agradeço a minha noiva, Bianca Silva Santos, pela compreensão, incentivo e apoio em todos os fins de semana dedicado aos estudos.

Gostaria de agradecer também aos meus grandes amigos da faculdade, que sempre estavam dispostos à auxiliar um ao outro e permitir que essa caminhada fosse mais alegre.

E a todo o corpo docente do curso de Bacharel em Ciências da Computação, que sempre me indicaram o caminho certo a seguir, ajudando na minha formação de maneira enriquecedora.

Em particular, gostaria de agradecer ao Lucas Almeida, pelo apoio no desenvolvimento deste trabalho.

Ainda em particular, gostaria de agradecer ao professor Itamar Solopak, por me conceder o primeiro emprego profissional na área da programação, o qual alavancou meus conhecimentos.

“Você pode encarar um erro como uma besteira a ser esquecida, ou como um resultado que aponta uma nova direção.”

(Steve Jobs)

RESUMO

Este trabalho tem como objetivo automatizar o livro de registro de classe da instituição FAFIMAN – Fundação Faculdade Filosofia Ciências e Letras de Mandaguari, localizada na cidade de Mandaguari, Paraná. O cenário atual conta com um sistema educacional da empresa Elotech Informática e Sistemas, onde o mesmo apresenta mal funcionamento ou um grau alto de complexidade em determinadas funcionalidades necessárias ou até mesmo a falta destas funcionalidades, como a que será apresentada neste trabalho. Em uma conversa com alguns docentes da instituição, foram levantados alguns pontos-chaves onde ocorrem os maiores déficits, um destes pontos será abordado neste trabalho. O livro de registro de classe da instituição é gerado pelo atual sistema, porém os docentes veem a necessidade de obter um aplicativo móvel para que seja realizado o preenchimento deste, o que atualmente é preenchido a caneta. O projeto será realizado na linguagem Delphi, utilizando a IDE *RAD Studio 10.3 Community*, conhecida como Delphi Rio. Para a comunicação entre o aplicativo móvel e o sistema desktop será utilizado os recursos DataSnap Server / Client, Rest e JSON presentes na IDE. Para gerenciamento do banco de dados do sistema desktop será utilizado o Firebird 3.0 juntamente com a ferramenta gráfica IBExpert, para o aplicativo móvel será utilizado o banco SQLite juntamente com a ferramenta gráfica DB Browser (SQLite).

Palavras-chave: Livro de Registro De Classe, Chamada Eletrônica, Aplicativos Moveis.

ABSTRACT

This work aims to automate the FAFIMAN's class registration book - Mandaguari School of Philosophy Sciences and Letters Foundation, located in the city of Mandaguari, Paraná. The current scenario has an educational system of the company Elotech Informática e Sistemas, where it presents malfunction or a high degree of complexity in certain required features or even the lack of these features, as will be presented in this paper. In a conversation with some teachers of the institution, some key points were raised where the largest deficits occur, one of these points will be addressed in this paper. The institution's class record book is generated by the current system, but teachers see the need to obtain a mobile application to complete it, which is currently filled with the pen. The project will be done in Delphi language, using the IDE RAD Studio 10.3 Community, known as Delphi Rio. The communication between the mobile application and the desktop system will use the DataSnap Server / Client, Rest and JSON features present in the IDE. For database management of the desktop system Firebird 3.0 will be used together with the graphical tool IBExpert, for the mobile application will be used the database SQLite together with the graphical tool DB Browser (SQLite).

Keywords: Class Registration Book, Electronic Call, Mobile apps.

LISTAS DE FIGURAS

Figura 1 – Smartphone Addiction Tightens Its Global Grip	20
Figura 2 – Componentes de um sistema de banco de dados	23
Figura 3 – Sistema SGDB	24
Figura 4 – Características de Microserviços	26
Figura 5 – Exemplo de Microserviços	27
Figura 6 – Como funciona uma API	28
Figura 7 – Exemplo do padrão de utilização dos métodos HTTP em um serviço REST, para um recurso chamado Cliente	32
Figura 8 – Exemplo de objeto JSON com todos os tipos de dados básicos do JSON	34
Figura 9 – Exemplo de um objeto JSON com atributos do tipo array e um array do tipo objeto ou array	34
Figura 10 – Estrutura de representação do JSON em Objeto	35
Figura 11 – Exemplo de dados representados JSON com a estrutura de Objeto	35
Figura 12 – LiveBindings comunicação entre componentes FireMonkey e Acesso a Dados	40
Figura 13 – Controle de Notas e Frequência, pág. 1	44
Figura 14 – Controle de Notas e Frequência, pág. 2	44
Figura 15 – Diagrama de Classe	47
Figura 16 – Diagrama de Classe	48
Figura 17 – Prefixos das Tabelas do Banco de Dados	49
Figura 18 – Dicionário de Dados – Tab. Departamentos	50
Figura 19 – Dicionário de Dados – Tab. Cursos	50

Figura 20 – Dicionário de Dados – Tab. Disciplinas	50
Figura 21 – Menu Principal e as opções disponíveis – Sistema Desktop	51
Figura 22 – Tela de Pesquisa das Data de Vigência – Sistema Desktop	52
Figura 23 – Tela de Manutenção das Data de Vigência – Sistema Desktop	52
Figura 24 – Tela de Pesquisa dos Horários – Sistema Desktop	53
Figura 25 – Tela de Manutenção dos Horários – Sistema Desktop	53
Figura 26 – Menu Principal – Aplicativo	54
Figura 27 – Tela de filtro para realizar a listagem da chamada – Aplicativo	55
Figura 28 – Tela para realizar a chamada – Aplicativo	56

LISTAS DE TABELAS

Tabela 1 – Métodos do Protocolo HTTP	31
Tabela 2 – Tipos de dados básicos do JSON	34
Tabela 3 – Caso de Uso	45
Tabela 4 – Especificação de Caso de Uso	46

LISTAS DE ABREVIATURAS E SIGLAS

SMS – *Short Message Service* - “Serviço de Mensagens Curtas”.

PDA – *Personal Digital Assistants* - “Assistente Pessoal Digital”.

APP - *Application* - “Aplicativo”.

BYOD – *Bring Your Own Device* - “Traga Seu Próprio Dispositivo”.

TI – Tecnologia da Informação.

API – *Application Programming Interface* - “Interface de Programação de Aplicações”.

CEO – *Chief Executive Officer* - “Diretor Executivo”.

DLP – *Data Loss Prevention* - “Prevenção de perda de dados”.

SIEM – *Security Information and Event Management* - “Gerenciamento de eventos e informações de segurança”.

HTML – *HyperText Markup Language* - “Linguagem de Marcação de Hipertexto”.

XML – *Extensible Markup Language*.

HTTP – *HyperText Transfer Protocol* – “Protocolo de Transferência de Hipertexto”.

CSS – *Cascading Style Sheets* - “Folhas de Estilo em Cascata”.

OPEN-SOURCE – Software com Código Fonte Aberto.

JS – *JavaScript*.

SQL – *Structured Query Language* - “Linguagem de Consulta Estruturada”.

REST – *Representational State Transfer* - “Transferência de Estado Representacional”.

JSON – *JavaScript Object Notation* - “Notação de Objetos JavaScript”.

VCL - *Visual Component Library* – “Biblioteca de Componentes Visuais”.

SUMÁRIO

1. INTRODUÇÃO.....	14
2. CONCEITOS E FUNDAMENTOS TEÓRICOS.....	17
2.1 DISPOSITIVOS MÓVEIS.....	17
2.2 APLICATIVOS MÓVEIS.....	19
2.3 BYOD - BRING YOUR OWN DEVICE	20
3. FERRAMENTAS E SERVIÇOS DISPONÍVEIS PARA O DESENVOLVIMENTO DE APLICATIVO MÓVEIS	22
3.1 BANCO DE DADOS E SGBD	22
3.2 FRAMEWORK	25
3.3 MICROSERVIÇOS	25
3.4 API - APPLICATION PROGRAMMING INTERFACE.....	27
3.4.1 APIS PÚBLICAS:.....	28
3.4.2 APIS PRIVADAS:	29
3.4.3 EXEMPLOS DE APIS DISPONÍVEIS NO MERCADO:	29
3.5 REST E RESTFULL.....	30
3.6 JSON.....	33
3.7 IONIC FRAMEWORK.....	36
3.8 ANGULAR.....	36
3.9 VUE.JS.....	37
4. FERRAMENTAS E SERVIÇOS UTILIZADOS NO DESENVOLVIMENTO DO PROJETO	38
4.1 DELPHI	38
4.1.1 FIREMONKEY E LIVE BINDINGS.....	39
4.2 FIREBIRD	41
4.3 SQLITE	42
5. DESENVOLVIMENTO	43
5.1 CENÁRIO ATUAL	43
5.2 ESCOPO DO PROJETO.....	45
5.3 CASO DE USO	45
5.3.1 ESPECIFICAÇÃO DE CASO DE USO.....	46
5.4 DIAGRAMA DE CLASSE	47
5.5 DICIONÁRIO DE DADOS	49
5.6 TELAS DO SISTEMA.....	51
5.6.1 SISTEMA DESKTOP.....	51
5.6.2 APLICATIVO MOBILE	54
6. CONCLUSÃO	57
7. REFERENCIAS BIBLIOGRÁFICAS	58

1. INTRODUÇÃO

Ao longo do ano letivo são acumuladas inúmeras pilhas de papéis que não podem ser descartadas, como por exemplo: regimento, controle de frequência, projetos, avaliações, autorizações, fichas cadastrais, históricos escolares, entre outros.

De acordo com Vivian Staroski (2018), professora do Senac EAD e especialista em Supervisão, Orientação e Inspeção Escolar, um dos principais erros das instituições é a falta de organização, em relação aos documentos.

No Brasil, segundo o Instituto Brasileiro de Geografia e Estatística (IBGE), são produzidos diariamente pela população urbana cerca de 120 mil toneladas de lixo.

Conforme os dados do Panorama dos Resíduos Sólidos no Brasil, da Associação Brasileira das Empresas de Limpeza Pública e Resíduos Especiais (Abrelpe), apontam que ao longo do ano de 2014, foram gerados 387,63 kg de lixo per capita, ou seja, cada cidadão brasileiro produziu em média 1,062 kg de resíduos sólidos por dia, um aumento de 2% em relação ao ano de 2013.

Segundo o professor de matemática Henrique Martins da Escola Estadual Prefeito Nestor de Camargo, localizada em Barueri, no estado de São Paulo, eles utilizam em torno de 1.600 folhas de papel sulfites, para a realização de um “provão”, com cerca de três a quatro páginas. Multiplicando isto pelo número escolas estaduais, municipais, e instituições de ensino superior, teríamos um elevado número de folhas sulfites utilizadas.

Segundo dados levantados pela empresa *WWI-Worldwatch Institute*, sediada em Washington, desde a metade do século XX, o consumo de papel no mundo cresceu mais de seis vezes, podendo chegar a mais de 300 kg per capita ao ano em alguns países.

De acordo com os dados da Associação Brasileira Técnica de Celulose e Papel (ABTCP), o consumo de papel no Brasil é em média 6 milhões de toneladas por ano.

Segundo José Maria Gusman Ferraz, doutor em Ecologia, pesquisador da Embrapa Meio Ambiente, no Brasil, apenas 37% do papel produzido vai para a reciclagem. De todo o papel reciclado, 80% é destinado à confecção de embalagens, 18% para papéis sanitários e apenas 2% para impressão.

Na era da tecnologia, a utilização de documentos digitais vem crescendo continuamente nos últimos anos, pois cada vez mais o mundo necessita de soluções inovadoras para otimizar suas atividades e fluxo de comunicação em seus negócios, além de procurar garantir melhores resultados, competitividade, agilidade e atingir a máxima eficiência.

Segundo Rondinelli (2005 p. 116), o Gerenciamento Eletrônico de Documentos (GED) no Brasil surgiu “[...] por parte do Poder Executivo Federal, de programas voltados para a disponibilização de serviços de informações ao cidadão via internet.”.

Conforme pesquisa realizada pelo Centro Nacional de Monitoramento e Alertas de Desastres Naturais (CENADEM), para 25% dos entrevistados a redução no volume de papéis nas empresas, é o principal motivo na hora de contratar serviços de GED.

Para Baldam (2002 p.37), a utilização de ferramentas de gerenciamento de documentos digitais, tem como principais objetivos a redução do tempo no manuseio do papel, melhoria da qualidade no trabalho, aumento da satisfação do usuário, referente ao serviço prestado, acesso imediato de multiusuário a qualquer informação, além do aumento na precisão na localização de documentos, obtendo assim as respostas de forma mais rápida e eficaz.

Com base nos estudos e regimentos da instituição, a mesma, tem por obrigação armazenar o histórico escolar dos alunos, por isso, foi sugerida a implementação do aplicativo para controle de frequência dos acadêmicos.

Conforme observado durante o cotidiano pelos acadêmicos da instituição, os docentes têm dificuldade ao realizar a chamada, pois muitas das vezes os Livros de Registro de Classe estarem desatualizados ou não estarem prontos, ou até mesmo por problemas resultados ao lançarem as frequências no sistema utilizado atualmente, além de problemas causados por perda de pacote de internet. Atualmente a falta de um aplicativo para celular que funcione offline não permite a substituição digital do Livro de Registros de Classe.

A implementação do aplicativo para controle de frequência dos acadêmicos tem como objetivo, organizar, minimizar e facilitar o acesso às informações existentes, além de centralizar as informações no formato digital, com indexação e armazenamento em uma infraestrutura de banco de dados, da qual o documento poderá ser encontrado facilmente, por meio de sistemas corporativos. Além de

inúmeros benefícios, tais como, a redução da utilização de folhas de papel sulfites e de custos com a elaboração e fabricação do Livro de Registro de Classe impresso.

Este aplicativo, destinado aos professores da instituição FAFIMAN (Fundação Faculdade Filosofia Ciências e Letras de Mandaguari), reúne diversos recursos, como controle de frequência dos acadêmicos, visualização da grade de horários, entre outros. O objetivo principal deste aplicativo é centralizar as informações em uma infraestrutura de banco de dados, evitando o desperdício de recursos e folhas de sulfites, além de auxiliar o dia a dia docente, facilitando a comunicação entre a secretaria, professores, profissionais de diferentes departamentos da instituição e principalmente com os acadêmicos.

2. CONCEITOS E FUNDAMENTOS TEÓRICOS

Neste capítulo serão abordados conceitos básicos referente a dispositivos e aplicativos móveis, assim como o conceito de utilização dos mesmos no ambiente corporativo.

2.1 DISPOSITIVOS MÓVEIS

Os dispositivos móveis são tecnologias digitais que permitem que os usuários tenham acesso a diversas informações, sejam elas, pessoais e/ou corporativas, independente do momento ou de sua localização, assim facilitando a execução de diversas tarefas.

Os dispositivos móveis são considerados um computador de bolso, cuja maioria possui grande poder operacional e funcional, sendo capazes de comunicar a outros dispositivos, para obter dados, assim como fornecê-los, conectar-se a internet e rodar poderosíssimos aplicativos móveis.

“O maquinismo se converteu nos últimos tempos em eletrodoméstico ou série de aparelhos portáteis que, mais e mais, coordenam, medeiam e agenciam não só as atividades da casa, mas movimentam as ligações do sujeito com nossa civilização planetária. Os negócios, comunicações, pesquisas, lazeres e atividades profissionais, para não falar das relações de poder e dos laços de afetividade, passam agora todos por ele e, assim, formam uma rede de trocas e ações cujo sentido dominante, todavia, não é técnico, mas de ordem social, espiritual e histórica.” (RÜDIGER, 2013, p. 14-15).

O celular é um dispositivo móvel, indicado para tarefas mais simples, como realizar ligações, enviar mensagens (SMS), armazenar informações e contatos, dependendo do modelo, consegue-se tirar fotos (lógico com uma resolução menor), gravar vídeos simples e até ouvir música pelo rádio FM.

O smartphone é um celular com tecnologias avançadas e significa “telefone inteligente”, cujos mesmos possibilitam o desenvolvimento de diversos programas, chamados de aplicativos.

Um smartphone contém as características de computadores, como *hardware* e *software*, pronto para acessar a internet, redes sociais, baixar aplicativos, jogar, sincronizar dados, entre outras tarefas. Segundo Corrêa et al. (2014) a sua superioridade está ligada às funcionalidades oferecidas, pois segundo os autores:

“Ao invés de apenas armazenar informações de telefones, efetuar e receber ligações, receber e enviar mensagens de texto, os smartphones realizam tarefas mais avançadas: receber e enviar emails [sic], realizar pesquisas na Internet e executar tarefas normalmente associadas a computadores pessoais ou notebooks.”. (CORRÊA et al., 2014, p. 1-2).

Segundo Corrêa et al., a “expansão tecnológica na área da Computação Móvel, em especial, da telefonia móvel, aumentou a demanda por aplicações para os mais variados setores da sociedade” (CORRÊA et al., 2014, p. 2). Segundo Fincotto:

“Com a utilização dos aplicativos móveis aliados à automação comercial e as tecnologias existentes, as empresas podem se tornar mais competitivas diante de um mercado cada vez mais globalizado e concorrido.”. (FINCOTTO, 2014, p. 11).

Os dispositivos móveis se comunicam com aplicações em um servidor web e precisam trocar dados com a base de dados no servidor. A maioria dos dados da base de dados do servidor não devem ser baixados para o dispositivo, apenas os dados que podem ser usados pelo usuário da aplicação móvel. A programação desse envio de dados deve ser feita com cautela para não aumentar desnecessariamente o tempo de sincronização e de carga nas tabelas do dispositivo, além de diminuir o tempo de consultas e de uso do espaço de armazenamento. (IMPERIAL, 2010).

2.2 APLICATIVOS MÓVEIS

Aplicativo móvel ou aplicação móvel é um software desenvolvido para ser instalado em dispositivos eletrônico móveis, como tablets e smartphones.

Os aplicativos são normalmente conhecidos como “apps”, uma forma abreviada para “aplicação de *software*” ou um programa de *software* para um sistema operacional de computador ou telefone, cuja mesma, em 2010, foi assinalada como a “palavra do ano” pela *American Dialect Society* (Sociedade Americana de Dialeto), em sua 21ª votação anual das palavras do ano.

Segundo Appcelerator (2012, p. 3), “80% dos desenvolvedores expressaram a necessidade de estender suas aplicações em mais de um sistema operacional”, visto que “os principais sistemas operacionais móveis são: Android, iOS, Symbian, Windows Mobile e BlackBerry” (DE MENDONÇA, 2011, p. 2).

Os aplicativos podem ser instalados em aparelhos por meio de *downloads* realizados pelos usuários através de uma loja on-line, como Google Play, App Store ou Windows Phone Store. Alguns aplicativos estão disponibilizados para serem baixados de forma gratuita, enquanto outros são pagos, podendo um aplicativo, obter custos diferenciados, dependendo do dispositivo e de seu sistema operacional.

Conforme as estimativas da plataforma *Store Intelligence* da Sensor Tower, o aplicativo mais baixado no mundo, referente à download de aplicativos, realizados na App Store e o Google Play em todo o mundo entre 1º de abril de 2019 e 31 de maio de 2019, foi o WhatsApp, com mais de 64,7 milhões de instalações.

Conforme o gráfico apresentado pela empresa *Statista*, nos anos de 2016 os brasileiros possuíam a maior média de uso de smartphones do mundo, sendo 4 horas e 48 minutos.

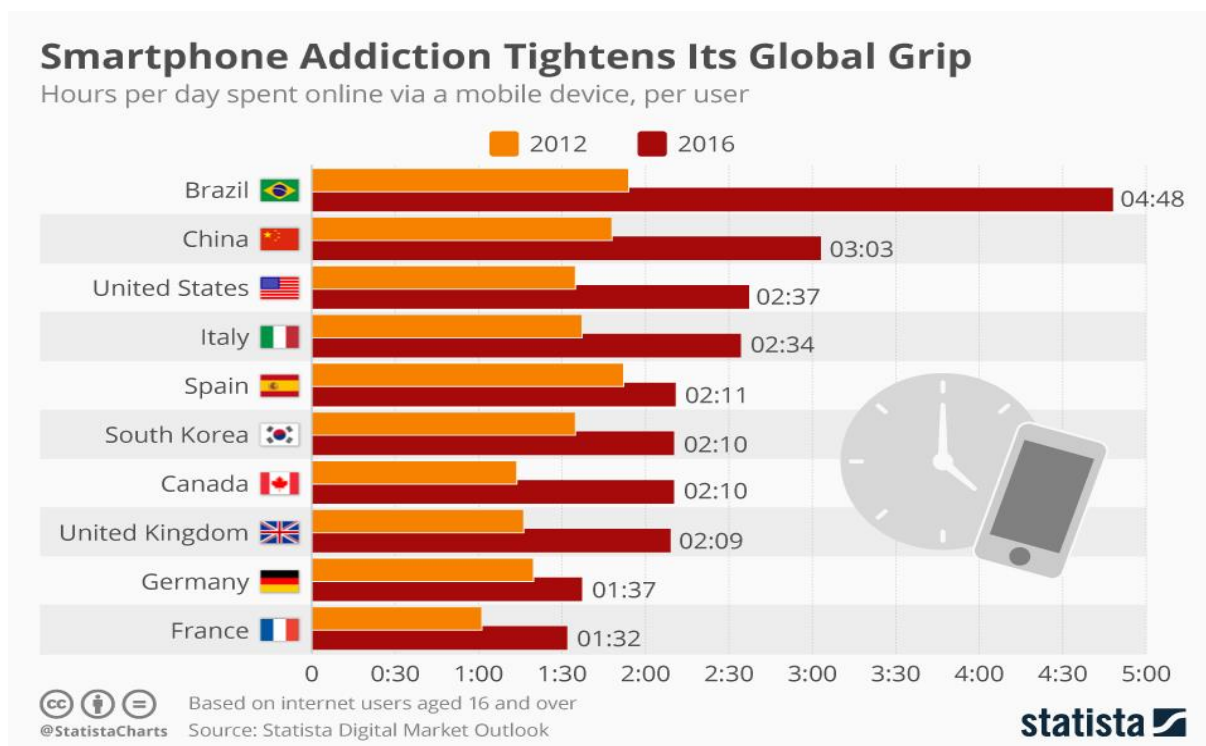


Figura 1 – Smartphone Addiction Tightens Its Global Grip

Fonte: statista, 24 de maio de 2017.

2.3 BYOD - BRING YOUR OWN DEVICE

A sigla BYOD refere-se ao termo em inglês “*Bring your own device*” ou em português “Traga seu próprio dispositivo”.

Conforme Miglioli (2007) e Leite (2015), a Tecnologia da Informação (TI) é definida como a “infraestrutura organizada de hardware, software, banco de dados e redes de telecomunicações, que permite manipular, gerar e distribuir dados e informações ao longo dos seus usuários (empresas ou pessoas)”.

Segundo Bindes (2012, p. 7-8) a TI é utilizada nas empresas por questões estratégicas, já que “a competitividade requer maior domínio das situações, aptidão para se ajustar às novas condições do mercado e maior competência para saber empregar novas tecnologias”, pois a rápida tomada de decisão possibilita “às organizações se transformarem rapidamente e levar essas mudanças ao mercado”

(BINDES, 2012, p. 11). Fincotto (2014) apresenta uma justificativa para o uso da TI nas organizações e alguns benefícios da sua utilização:

“O aumento da concorrência e do consumo faz com que as empresas busquem novas soluções para melhorar e agilizar seu atendimento, exposição de seus produtos e serviços visando aumentar sua competitividade e destaque no mercado. Tendo em vista todas essas mudanças e a constante evolução dos computadores e celulares, as empresas fazem uso cada vez mais intensivo da tecnologia da informação (TI) como principal ferramenta de apoio estratégico em seus negócios.”.
(FINCOTTO, 2014, p. 1).

BYOD surgiu no final da primeira década do século XXI, juntamente com a explosão do mundo mobile. Este termo descreve a tendência global da qual viabiliza aos profissionais que tragam seus dispositivos pessoais para desenvolverem suas atividades no ambiente corporativo, sejam eles smartphones, tablets ou notebooks, pois os gestores das empresas perceberão que muitas vezes os dispositivos de seus colaboradores eram mais modernos e eficazes que os próprios dispositivos da empresa, transformando locais formais em lugares mais flexíveis.

O presidente da Dell Brasil, Luis Gonçalves, afirma que “O uso intensivo de tecnologia no cotidiano, tem tornado os usuários cada vez mais exigentes em relação aos equipamentos e sistemas utilizados no trabalho”.

O conceito de BYOD está ganhando forças nas organizações, pois tem como objetivo a redução de custos com máquinas e aumento do conforto e produtividade do colaborador, já que o colaborador tem a liberdade de utilizar uma tecnologia que já está acostumado para acessar os dados da organização.

Não importa se o colaborador está fisicamente no ambiente de trabalho ou remotamente, ele utilizará seu próprio dispositivo, para realizar as atividades profissionais.

Muitas empresas ainda deixam de lado este conceito, pois os gestores sentem-se preocupados com assuntos como segurança da informação e o suporte oferecido pelo departamento de TI.

3. FERRAMENTAS E SERVIÇOS DISPONÍVEIS PARA O DESENVOLVIMENTO DE APLICATIVO MÓVEIS

Neste capítulo serão abordados conceitos básicos referente as ferramentas e serviços que estão disponíveis para o desenvolvimento de aplicativos móveis.

3.1 BANCO DE DADOS E SGBD

Um banco de dados é uma coleção organizada de dados (esquemas, tabelas, consultas, relatórios, exibições e outros objetos). Os bancos de dados são responsáveis pelo armazenamento de dados e pelo seu relacionamento entre si.

Um banco de dados é um projeto elaborado para o armazenamento de fatos que possuam aspectos do mundo real, do qual tem o objetivo de fornecer os dados necessários aos Sistemas de Informação para processamento e geração de informação para os usuários.

Segundo Korth, um banco de dados “é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico”. (DEVMEDIA, 2006).

“Um banco de dados representa algum aspecto do mundo real, às vezes chamado de minimundo ou de universo de discurso (UoD – Universe of Discourse).” (ELMASRI; NAVATHE, 2011).

Por exemplo, considere nomes, números telefônicos e endereços de pessoas que você conhece. Esses dados podem ter sido escritos em uma agenda de telefones ou armazenados em um computador, por meio de programas como o Microsoft Access ou Excel. Essas informações são uma coleção de dados com um significado implícito, conseqüentemente, um banco de dados. (ELMASRI; NAVATHE, 2011).

Sistemas de Gestão de Base de Dados (SGBD), oriundo do inglês “*Data Base Management System*” (DBMS), trata-se de um *software* que possui recursos capazes

de manipular as informações do banco de dados e interagir com o usuário (DEVMEDIA, 2006). Para que isso seja feito, o SGBD utiliza um tipo de linguagem. A mais conhecida é a SQL, Structured Query Language.

“Um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações.”
(ELMASRI; NAVATHE, 2011).

Para (RAMAKRISHNAN e GEHRKE, 2008) o uso de um SGBD proporciona várias vantagens. Uma delas é a possibilidade de “utilizar os recursos do SGBD para gerenciar os dados de uma forma robusta e eficiente”.

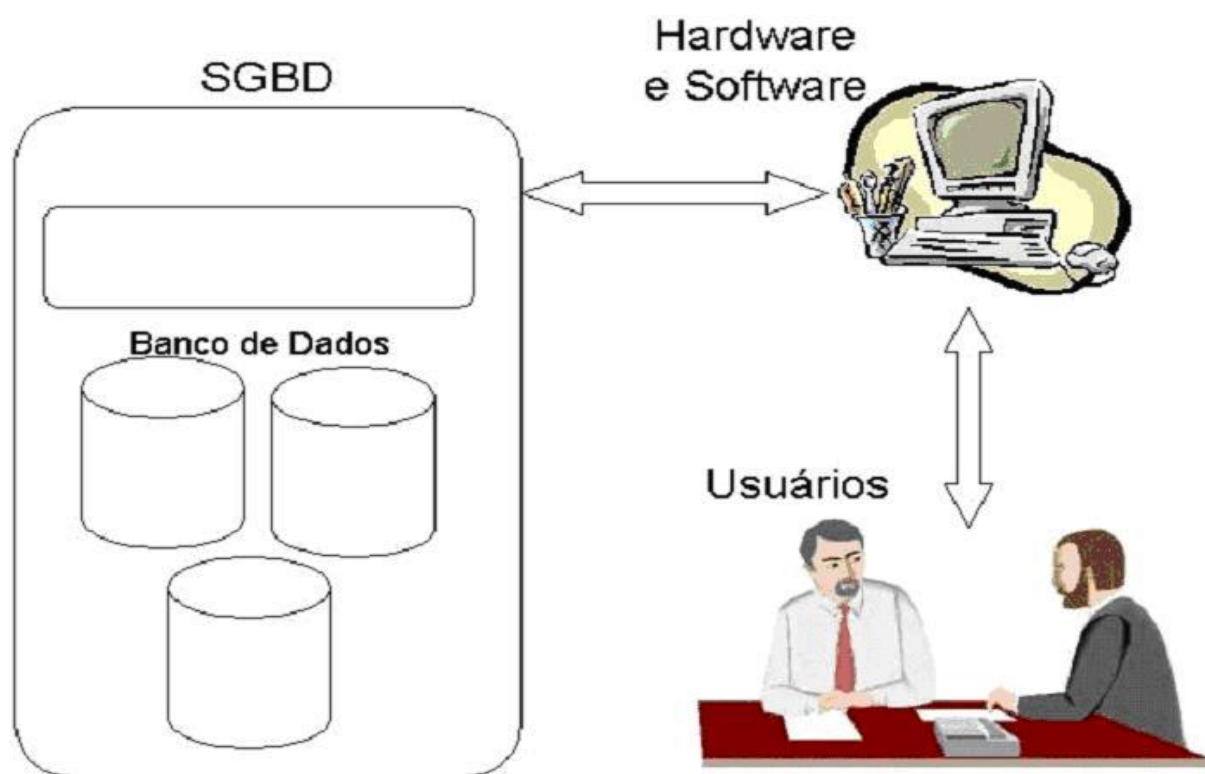


Figura 2 – Componentes de um sistema de banco de dados

Fonte: DevMedia

As interfaces dos SGDB's são conjuntos de rotinas que incluem as funcionalidades do programador vai necessitar frequentemente, exemplo de hoje em dia, a grande maioria dos programadores comunica-se com os usuários através de interfaces gráficas de janelas. (DEVMEDIA, 2011).

Exemplos de SGDBs são: Oracle, SQL Server, DB2, PostgreSQL, MySQL, o próprio Access ou Paradox, entre outros.



Figura 3 – Sistema SGDB

Fonte: DevMedia

3.2 FRAMEWORK

Os *frameworks* para desenvolvimento de software são conhecidos como um conjunto de códigos abstratos, ou seja, captura as funcionalidades comuns a várias aplicações, poupando tempo de desenvolvimento em operações básicas como validação de dados, conexão com o banco de dados, entre outros. Hartmann define os *frameworks* como:

“Um conjunto de bibliotecas, componentes de software e diretrizes de arquitetura que fornecem ao desenvolvedor um abrangente conjunto de ferramentas para construir uma aplicação móvel completa, de cima para baixo.”. (HARTMANN, 2011).

Os *frameworks* não são padrões de projetos de softwares, pois padrões possui um grau de abstração maior, já um *framework* inclui linhas de código, conjunto de classes com o objetivo de reutilização de arquitetura de *softwares*, devidamente implementadas e testadas.

“Framework é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.” (FAYAD; SCHMIDT, 2015).

3.3 MICROSERVIÇOS

O termo “microserviços” surgiu em uma conferência de arquitetos de *software*, perto de Veneza em maio de 2011.

O microserviço é um tipo de arquitetura de *software*, que permite que um *software* “grande” seja composto por unidades “pequenas” e independentes que se comunicam geralmente por meio de APIs. Cada serviço é desenvolvido para executar uma única tarefa, trabalhando de forma altamente independente. (HABIB, 2016).

Um microsserviço é um padrão utilizado para transformar aplicativos complexos em processos simples, composto por um ou mais serviços, ou seja, as aplicações são desmembradas em componentes mínimos, independentes e fracamente acoplados, tornando-se mais fáceis o desenvolvimento e manutenção de suas aplicações, além das integrações com outras aplicações.

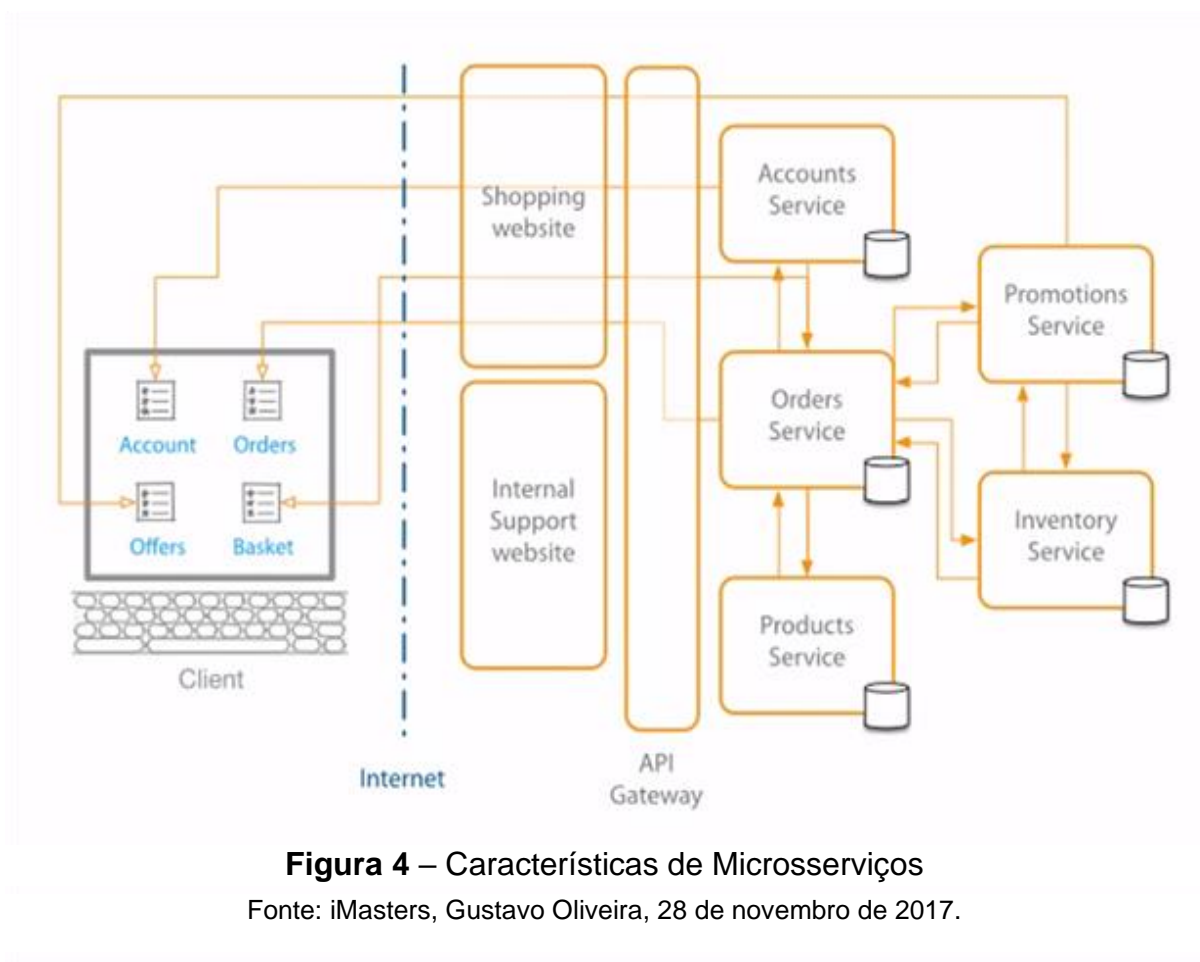


Figura 4 – Características de Microsserviços

Fonte: iMasters, Gustavo Oliveira, 28 de novembro de 2017.

Segundo Sam Newman, em seu livro “Building Microservices” (Construção de Microsserviços), microsserviços são componentes pequenos e focados, construídos para fazer uma única coisa e muito bem.

Os microsserviços tem inúmeros benefícios, um destes são a facilidade da implantação contínua e a alteração, depuração ou até mesmo substituição de certas funcionalidades de uma aplicação sem afetar as demais. (ANNENKO, 2016).

Os microsserviços podem ser implementados através de inúmeras tecnologias, alguma delas são nodeJs, Java, Python, entre outras. (EXAME, 2017).

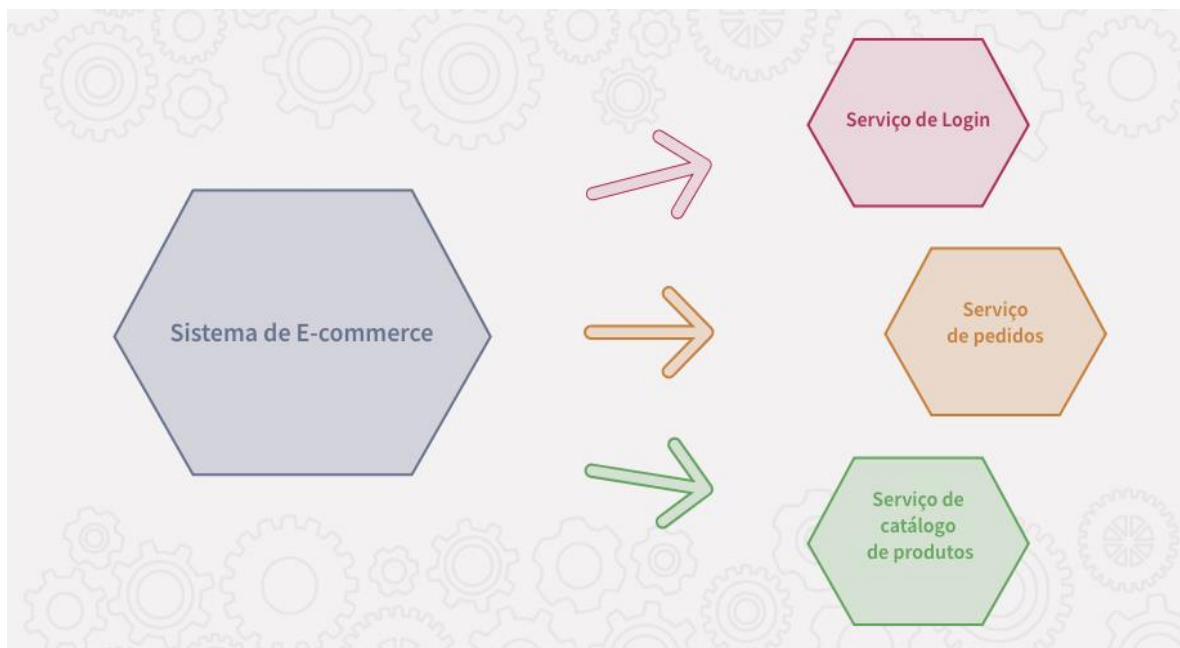


Figura 5 – Exemplo de Microsserviços
Fonte: School of Net, Victor Lima, 17 de setembro de 2018.

3.4 API - APPLICATION PROGRAMMING INTERFACE

A sigla API refere-se ao termo em inglês “Application Programming Interface”, traduzindo para o português “Interface de Programação de Aplicações”.

API é o conjunto padrões, definições, protocolos, rotinas de programação, que permitem integrar *softwares* de aplicações, podendo ser utilizadas para os mais variados tipos de negócio, por empresas de diversos nichos de mercado ou tamanho, e são invisíveis ao usuário comum, que enxerga apenas a interface dos *softwares* e aplicativos.

As APIs simplificam o desenvolvimento de aplicações, gerando economia de tempo e dinheiro para as empresas.

Uma API nada mais é que uma interface simplificada e padronizada, que fornece a vários programadores um método para a construção de aplicações que se comuniquem internamente, além de simplificar o design a administração e o uso, fornecem oportunidades de inovações.

As APIs são desenvolvidas, quando uma empresa de *software* tem a intenção de que outros criadores de *software* desenvolvam produtos associados ao seu

serviço, com um padrão aberto e uma documentação de acesso livre. Elas são uma forma de integrar sistemas, possibilitando benefícios como a segurança dos dados, facilidade no intercâmbio entre informações com diferentes linguagens de programação e a monetização de acessos.

As APIs proporcionam inúmeras possibilidades para os desenvolvedores de softwares e aplicativos, como a integração entre sistemas que possuem linguagem totalmente distintas de maneira ágil e segura, como diferentes bancos de dados, por exemplo.

Segundo Marcus Ribeiro, CEO da Pluga, em 2013, mais de 10 (dez) mil APIs tinham sido publicadas por empresas, para consumo público.

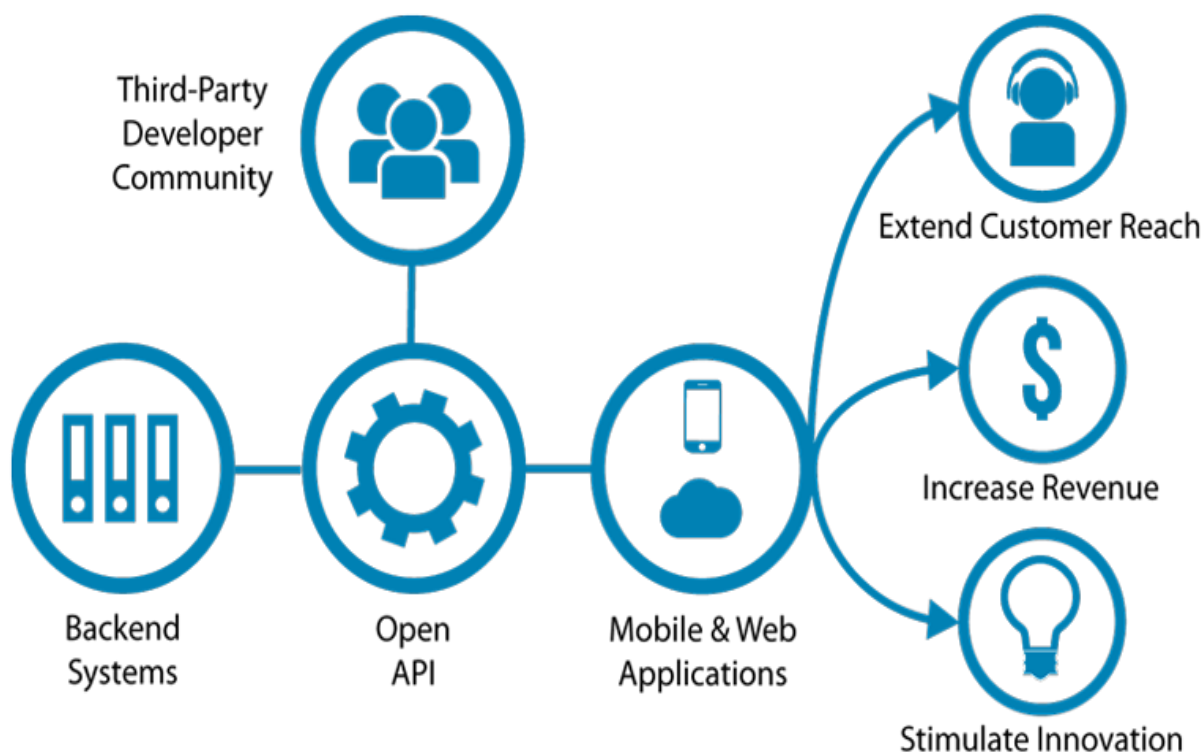


Figura 6 – Como funciona uma API

Fonte: Mind Consulting, 2 de julho de 2019.

3.4.1 APIS PÚBLICAS:

As APIs Públicas são interfaces projetadas para serem de fácil acesso pelos desenvolvedores, ou seja, elas permitem serem utilizadas por qualquer desenvolvedor externo que deseja ter acesso à interface, ou por desenvolvedores da organização que publicou a mesma API, permitindo que a organização estimule seus

desenvolvedores a desenvolverem aplicações inovadoras, até soluções inovadoras que agregam valor para seu negócio.

3.4.2 APIS PRIVADAS:

As APIs Privadas são interfaces projetadas para uso interno de uma instituição ou empresa, onde há um acesso é limitado de usuários.

Uma API privada, permite que desenvolvedores integrem facilmente com soluções internas, alinhando as necessidades do setor de TI com os softwares, para que os problemas sejam resolvidos de maneira rápida e eficiente, tornando o fluxo de trabalho mais eficaz. Além de tornar possível a expansão dos sistemas da empresa de acordo com o que for necessário, explorando novos negócios e modernizando a estrutura interna, permitindo uma integração inteligente quando forem necessários novos aplicativos para atender as demandas de diferentes mercados.

3.4.3 EXEMPLOS DE APIS DISPONÍVEIS NO MERCADO:

- **Dropbox**

A API do Dropbox permite criar, acessar e atualizar arquivos no Dropbox do usuário. Esta API contém um conjunto de soluções de segurança, conformidade e administração, como eDiscovery, Prevenção de perda de dados (DLP) e Gerenciamento de eventos e informações de segurança (SIEM).

- **Facebook**

A API do Facebook utiliza protocolo REST e para as respostas o formato XML. Algumas das funcionalidades desta API são: facilitar o acesso e o compartilhamento, criar jogos e aplicativos, pagamentos e Ads (propagandas).

- **Buscapé**

A API do Buscapé é um conjunto de ferramentas para a criação de aplicações em que há necessidade de uma base de dados de lojas, produtos, serviços e ofertas.

- **Marvel Comics**

A Marvel Comics API é uma ferramenta para ajudar os desenvolvedores em todos os lugares a criar sites e aplicativos incríveis, surpreendentes e incríveis, usando dados dos mais de 70 anos da era da Marvel dos quadrinhos.

Esta API faz uso do Swagger para a documentação e teste das funcionalidades existentes.

Para acessar os recursos da Marvel Comics API serão utilizadas uma chave pública e outra privada, além das chaves pública e privada, será necessário indicar ainda os domínios de aplicações que utilizarão a API.

- **Google Maps**

O Google Maps um dos grandes exemplos na área de APIs. Por meio de seu código original, muitos outros sites e aplicações utilizam os dados do Google Maps adaptando-o da melhor forma a fim de utilizar esse serviço.

Quando uma pessoa acessa uma página de um hotel, por exemplo, é possível visualizar dentro do próprio site o mapa do Google Maps para saber a localização do estabelecimento e verificar qual o melhor caminho para chegar até lá.

3.5 REST E RESTFULL

REST é acrônimo de *Representational State Transfer*, cujo mesmo tem como objetivo a definição de características fundamentais para a construção de aplicações Web seguindo boas práticas. (DIAS, 2016).

O REST é um modelo a ser utilizado para se projetar arquiteturas de software distribuído, baseadas em comunicação via rede, ou seja, ele pode ser considerado como um conjunto de princípios, que quando aplicados de maneira correta em uma aplicação, a beneficia com a arquitetura e padrões da própria Web. (FERREIRA, 2017).

Conforme Stefan Tilkov, líder, co-fundador, e consultor principal da comunidade de SOA da InfoQ, o REST é um conjunto de princípios que definem como

Web Standards como HTTP (“*Hypertext Transfer Protocol*”, ou em português “Protocolo de Transferência de Hipertexto”) e URIs (“*Uniform Resource Identifier*”), ou em português “Identificador niforme de Recursos”) devem ser usados.

O REST foi descrito por Roy Fielding, um dos principais criadores do protocolo HTTP, em sua tese de doutorado e que foi adotado como o modelo a ser utilizado na evolução da arquitetura do protocolo HTTP.

O elemento primordial de um REST é o recurso, que ao contrário de objetos, contem seus métodos previamente definidos pelo protocolo HTTP.

O protocolo HTTP possui diversos métodos, sendo que cada um possui uma semântica distinta, mas os principais, de maneira sucinta, são:

Método	Descrição de sua Utilização	Observações
GET	Indica a recuperação de um recurso.	Equivalente a um SELECT em uma base de dados.
POST	Indica a inserção ou criação de um novo recurso.	Equivalente a um INSERT em uma base de dados.
PUT	Indica a substituição de um determinado recurso.	Equivalente a um UPDATE em uma base de dados.
PATCH	Indica a atualização parcial de um determinado recurso.	
DELETE	Indica a exclusão de um determinado recurso.	Equivalente a um DELETE em uma base de dados.

Tabela 1 – Métodos do Protoco HTTP

Fonte: Dados retirados de: <<https://blog.caelum.com.br/rest-principios-e-boas-praticas/>> e <<https://www.treinaweb.com.br/blog/rest-nao-e-simplesmente-retornar-json-into-alem-com-apis-rest/>>

Para o modelo REST os recursos, são abstrações sobre um determinado tipo de informação que uma aplicação gerencia, além disto o mesmo diz que todos os recursos devem possuir uma identificação única.

A identificação de um recurso dentro de um REST se dá através de um conceito unificado para IDs, a URI (Uniform Resource Identifier — identificador uniforme de Recursos).

Na figura 7 abaixo, temos um exemplo do padrão de utilização dos métodos HTTP em um serviço REST, para um recurso chamado Cliente.

Método	URI	Utilização
GET	/clientes	Recuperar os dados de todos os clientes.
GET	/clientes/id	Recuperar os dados de um determinado cliente.
POST	/clientes	Criar um novo cliente.
PUT	/clientes/id	Atualizar os dados de um determinado cliente.
DELETE	/clientes/id	Excluir um determinado cliente.

Figura 7 – Exemplo do padrão de utilização dos métodos HTTP em um serviço REST, para um recurso chamado Cliente.

Fonte: Dados abstraídos de: <<https://blog.caelum.com.br/rest-principios-e-boas-praticas/>>

Existe uma certa confusão quanto aos termos REST e RESTful. Entretanto, ambos representam os mesmos princípios. Sistemas que utilizam os princípios REST são chamados de RESTful.

- REST: conjunto de princípios de arquitetura;
- RESTful: capacidade de determinado sistema aplicar os princípios de REST.

O RESTful também tem impacto sobre outras métricas de qualidade de software, por exemplo: a facilidade de uso da aplicação e o tempo necessário pelo usuário para aprender a utilizá-la.

Outra característica importante dos recursos no padrão REST é a maneira como os dados são manipulados na implementação do serviço não está vinculada ao formato da resposta a ser fornecida a uma solicitação, o que permite que os clientes peçam os dados em uma grande variedade de formatos, como JSON, HTML, XML, texto puro, PDF, JPG, entre outros. (DEV MEDIA, 2016).

3.6 JSON

O *JavaScript Object Notation*, ou seja, Notação de Objeto em Javascript, popularmente conhecido por JSON, é um formato de representação de dados derivado da linguagem de programação *Javascript*, por isto seu nome. (ALVES, 2018).

O JSON não é um protocolo de transporte de informações como o HTTP, e sim é um formato para transferência de dados entre programas. (CAMPOMORI, 2017).

“JSON é em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados.” (JSON.ORG).

O JSON apesar de muito simples, tem sido bastante utilizado por diversas aplicações, de diferentes linguagens de programação, devido a sua capacidade de estruturar informações de uma forma bem mais compacta do que a conseguida pelo modelo XML, devido ao fato de seres humanos conseguirem lê-lo e escrevê-lo facilmente, assim como capacidade ágil das máquinas conseguirem gerá-lo e interpretá-lo. O que explica o fato de o JSON ter sido adotado por empresas como Google e Yahoo, cujas aplicações precisam transmitir grandes volumes de dados. (DEVMEDIA, 2012).

Os JSON são estruturados em objetos e/ou *arrays* (ou listas). Na representação do modelo objetos, os atributos devem seguir de um caractere, dois pontos (":") e o valor do atributo, e devem ser separados por vírgulas (","), Já o *arrays* só podem ser de um determinado tipo de dados. Porém um objeto JSON pode ter atributos do tipo *array* e um *array* pode ser do tipo objeto ou *array*. Além disto tanto *array* quando objeto, podem serem vazios em JSON. (ALVES, 2018).

Os tipos de dados básicos do JSON são:

Tipo	Descrição de sua Utilização	Exemplos
STRING	Deve estar entre aspas (duplas ou simples).	a) "Brasil" b) 'Brasil'.
NÚMERO	Deve estar sem aspas (duplas ou simples). Pode ser inteiro ou real, quando for do tipo real deve-se usar o caractere ponto (.) para separar a parte inteira das casas decimais.	a) 1 (inteiro) b) 23.454 (real).
BOOLEANO	Deve estar sem aspas (duplas ou simples). Tipo lógico normal, pode assumir valores true ou false.	a) True b) False
NULL	Indica a atualização parcial de um determinado recurso.	{ "nome" : null }.

Tabela 2 – Tipos de dados básicos do JSON.

Fonte: Dados retirados de: < <https://dicasdeprogramacao.com.br/o-que-e-json/> >

A figura 8 abaixo, representa um exemplo de objeto JSON com todos os tipos de dados apresentados na tabela 2.

```
{  
  "texto" : "Brasil",  
  "numero" : 23,  
  "numeroReal" : 54.87,  
  "booleano": true,  
  "nulo": null  
}
```

Figura 8 – Exemplo de objeto JSON com todos os tipos de dados básicos do JSON.

Fonte: Dados retirados de: < <https://dicasdeprogramacao.com.br/o-que-e-json/> >

A figura 9 abaixo, representa um exemplo de um objeto JSON com atributos do tipo *array* e um *array* do tipo objeto ou *array*.

```
{  
  "atributoDoTipoArray" : [1,2,3,54]  
}  
  
[{  
  "a":1  
},{  
  "b":1  
}]
```

Figura 9 – Exemplo de um objeto JSON com atributos do tipo *array* e um *array* do tipo objeto ou *array*.

Fonte: Dados retirados de: < <https://dicasdeprogramacao.com.br/o-que-e-json/> >

A representação de objeto em JSON, contém a seguinte estrutura:

- I. Um objeto começa com chave de abertura (“{”) e termina com chave de fechamento (“}”).
- II. Cada nome é seguido por dois pontos (“:”) e os pares nome/valor são seguidos por vírgula (“,”).

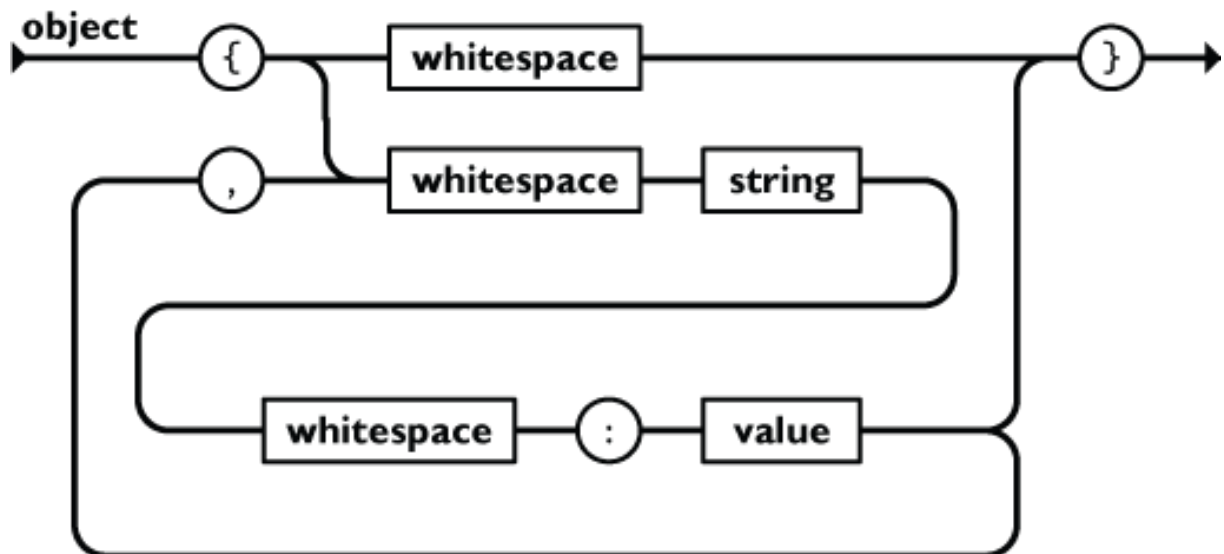


Figura 10 – Estrutura de representação do JSON em Objeto.

Fonte: Disponível em: <<https://www.json.org/json-pt.html>>

Vamos pensar no exemplo de um objeto **pessoa** com nome **Pedro** e altura 1,90. A representação deste objeto em JSON ficaria assim:

Na figura 11, temos a chave de abertura (“{”) e a chave de fechamento (“}”), os nomes dos atributos da classe (“**nome**”: e “**altura**”:) e os valores (“**Pedro**” e **1.90**).

```
{
  "nome": "Pedro",
  "altura": 1.90
}
```

Figura 11 – Exemplo de dados representados JSON com a estrutura de Objeto.

Fonte: Dados abstraído de: <<https://dicasdeprogramacao.com.br/o-que-e-json/>>

3.7 IONIC FRAMEWORK

O IONIC FRAMEWORK é um kit de ferramentas *open-source*, utilizado para o desenvolvimento de aplicativos móveis multiplataforma, com mais agilidade, adaptando e mudando a aparência dos seus componentes de acordo com o sistema operacional do dispositivo do usuário.

O IONIC foi criado por Max Lynch, Ben Sperry e Adam Bradley da Drifty Co, sediada em Madison, Wisconsin, EUA, no ano de 2012.

O IONIC consiste na utilização dos recursos do HTML, CSS, Javascript e Angular, em sua compilação, as quais possibilitam uma experiência fluída do usuário e diminuição no tempo de desenvolvimento (IONIC, 2016). Além disto o IONIC também possui integração com o Cordova e o ngCordova, recursos que simplificam ainda mais o desenvolvimento, tornando cada vez mais capaz a utilização de recursos nativos dos dispositivos.

3.8 ANGULAR

O Angular é uma plataforma e *framework* utilizado para o desenvolvimento de interfaces de aplicações utilizando HTML, CSS e JavaScript, liderada por uma equipe de desenvolvedores da Google. (AFONSO, 2018).

O Angular é um dos *frameworks* utilizado para o desenvolvimento de aplicações *client-side*, sejam elas para a web, *mobile* ou desktop, além de ser um dos *frameworks* mais amplos para o desenvolvimento de aplicações interativas do tipo “*Single-Page Applications*” (SPA), por ser orientado a objetos (OO) e utilizar a linguagem TypeScript como padrão. (DEV MEDIA).

O Angular é *open-source* e possui diversos elementos básicos, os principais são os componentes, *templates*, diretivas, roteamento, módulos, serviços, injeção de dependências e ferramentas de infraestrutura que automatizam tarefas, como a de executar os testes unitários de uma aplicação. (AFONSO, 2018).

3.9 VUE.JS

Vue.JS, VueJS ou simplesmente Vue é um *framework open-source* da linguagem JavaScript, utilizado para o desenvolvimento de interfaces do usuário. Por conter a forma incremental o Vue possibilita a facilidade na integração com projetos que utilizam bibliotecas de JavaScript, além disto o mesmo pode funcionar até como uma estrutura de aplicações web, capaz de alimentar aplicativos avançados de uma única página. O seu objetivo principal é permitir a interação uni e bidirecional dos elementos HTML com os dados e métodos definidos via JavaScript. (GALDINO, 2017).

A biblioteca central do Vue.js é focada exclusivamente na camada visual “*view layer*”, sendo fácil de ser coleta e integrada com outros projetos existentes ou a outras bibliotecas. (VUE.JS).

O Vue.js destaca-se pela simplicidade na execução das mesmas tarefas que outros *frameworks* contêm. Ele possui os mesmos conceitos que um framework reativo possui, como *data bind*, *two way*, *events*, criação de componentes, entre outros. (MENDES; SOUZA, 2018).

4. FERRAMENTAS E SERVIÇOS UTILIZADOS NO DESENVOLVIMENTO DO PROJETO

Com base no capítulo 3, o qual foram apresentadas ferramentas e serviços que estão disponíveis no mercado para o desenvolvimento de aplicativos móveis. Neste capítulo serão apresentadas as ferramentas e serviços que utilizamos para desenvolvimento do aplicativo de chamada eletrônica.

4.1 DELPHI

O Delphi é uma linguagem RAD (*Rapid Application Development* – Desenvolvimento Rápido de Aplicações) lançado no ano de 1995 pela Borland, por meio do projeto arquitetado por Anders Hejlsberg, que visava a criação de um ambiente visual para criar aplicações Windows utilizando a linguagem *Object Pascal*, que teve como origem a linguagem Pascal, que foi desenvolvida por Niklaus Wirth. (DALEPIANE, 2014).

Segundo o CEO e Co-Fundador da RM Factory, Rodrigo Mourão, o desenvolvimento de aplicativos nativos sempre foi defendido como sendo uma das melhores opções para a extração das melhores experiências visuais e gráficas dos usuários.

Em 2011, a Embacadero, disponibilizou a versão Delphi XE2, a qual então era direcionada exclusivamente à construção de aplicativos para a plataforma iOS, contemplando dispositivos Apple, tais como iPhone, iPad e iPodTouch. Por causa da expansão de dispositivos Android, foram surgindo os primeiros pedidos para que a plataforma desenvolvesse suporte nativo ao desenvolvimento de aplicações para a plataforma Android, aos quais foram atendidos na versão Delphi XE5. (KAWATA, 2014).

“O Delphi é uma plataforma completa para desenvolvimento de software, seja ele desktop, web ou mobile. Os pilares dessa tecnologia são a IDE, a linguagem e o compilador, todos batizados com o mesmo nome: Delphi.”. (CARDOSO, [20--]).

O *RAD Studio* oferece a forma mais simples e mais ágil para o desenvolvimento de aplicativos nativos multiplataforma com serviços de nuvem flexíveis e ampla conectividade IoT, além de controles avançados de *Visual Component Library* (VCL) para Windows 10, o mesmo permite o desenvolvimento com *FireMonkey* (FMX) para Windows, Mac e plataformas *mobile*. O *RAD Studio* suporta Delphi ou C++, além de tornar 5x mais rápido o desenvolvimento e a implantação entre várias plataformas de desktop, mobile, nuvem e banco de dados incluindo o Windows 10 de 32 e de 64 bits. (EMBARCADERO).

4.1.1 FIREMONKEY E LIVE BINDINGS

O Delphi passou por uma evolução, e foi criado o *framework FireMonkey* (FMX), cuja sua característica principal é gerar aplicações nativas tanto para Windows quanto para OS X, Android e iOS, ou seja, ele permite que um *software* seja compilado com código nativo para múltiplos dispositivos sem a necessidade de alterações no código fonte. (GRANATYR, 2017).

“FireMonkey é uma camada de abstração da interface gráfica que internamente é compilada de forma nativa para diversas plataformas.” (DEVMEDIA, 2016).

O *FireMonkey* é uma boa alternativa para o desenvolvimento de soluções de plataforma cruzada, pois permite aproveitar os conhecimentos e conceitos existentes no VCL (“Visual Component Library”, em português “Biblioteca de Componentes Visuais”), porém o VCL é uma estrutura somente do Windows e não pode ser usada no *FireMonkey*, logo que *FireMonkey* é compatível com Windows, Mac, IOS, Android e Linux. (FLECTH, 2012).

Conforme Rodrigo Mourão, as principais qualidades do *FireMonkey* são:

- ✓ O *FireMonkey* é a tecnologia principal da nova geração de plataformas para desenvolvimento de aplicações do Delphi;
- ✓ Genuinamente um *framework* de desenvolvimento multiplataforma;
- ✓ Consegue criar aplicações com visual e experiência de usuário tão bons quanto aplicações nativas;
- ✓ O *FireMonkey* faz uso da GPU (“*Graphics Processing Unit*”, em português “Unidade de Processamento Gráfico”) do dispositivo, o que possibilita o trabalho com gráficos avançados e efeitos de imagem de forma rápida e eficiente;
- ✓ Possui conectividade com os principais bancos de dados;
- ✓ Suporte completo por parte da Embarcadero e comunidade Delphi.

O *LiveBindings* foi lançado juntamente com o *FireMonkey* no Delphi XE2, principalmente porque esta nova plataforma de desenvolvimento não contém os controles *DataAware* tradicionais como existiam na VCL.

O *LiveBinding* é um mecanismo de *DataBinding* que tem o objetivo de fazer a ligação de fontes distintas de dados através de expressões, podendo ser unidirecional ou bidirecional. (DEVMEDIA, 2014).

No *LiveBinding* temos os objetos que são os *Source Objects* (objetos de origem) e os *Control Objects* (objetos de controle). Há a possibilidade de um mesmo objeto ser tanto de origem quanto de controle.

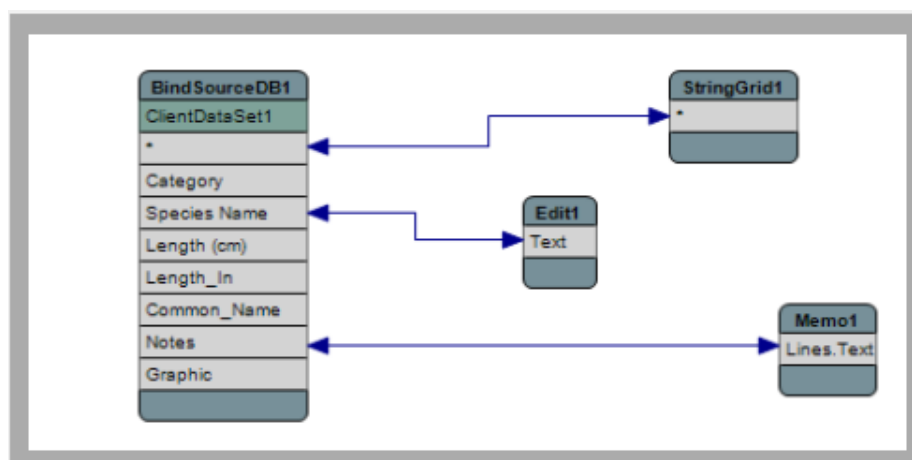


Figura 12 – LiveBindings comunicação entre componentes FireMonkey e Acesso a Dados.

Fonte: DevMedia

4.2 FIREBIRD

O *Firebird* é um sistema gerenciador de banco de dados gratuito e *open source*, que roda em diversos sistemas operacionais Linux, Windows, Mac OS, e uma variedade de plataformas Unix.

O *Firebird* nasceu nos meados de julho de 2000, nos bastidores da *Borland Software Corporation* no momento em que a mesma se pronunciou contra a continuidade do *InterBase 6.0*, assim abrindo o código fonte do mesmo, tornando-o *free e open source*.

“O Firebird é derivado do código do Borland InterBase 6.0. Ele tem o código aberto e não possui licença dupla, portanto você pode utilizá-lo em qualquer tipo de aplicação, seja ela comercial ou não, sem pagar nada por isso.” (CANTU, 2010).

O *Firebird* oferece recursos como: *procedures*, *triggers*, integridade referencial, SQL (ANSI 92/99), linguagem nativa (PSQL), UDFs (funções definidas por usuários), compatibilidade ACID (Atomicidade, Consistência, Isolamento e Durabilidade - em inglês: *Atomicity, Consistency, Isolation, Durability*), transações MVCC (Controle de concorrência multiversão), *collations*, entre outros recursos.

O acesso ao banco de dados *Firebird* pode ser feito diretamente pela API com C/C++ ou via componentes que usam acesso nativo em ferramentas RAD como: IBOjects (Delphi/C++Builder), FIBPlus (Delphi/Kylix/C++Builder), etc.

Segundo Cantu (2005), o *Firebird* conquistou vários desenvolvedores que utilizavam diversas tecnologias de programação como o Java, Delphi, PHP, .NET, entre outras. A tecnologia fez estes programadores migrarem seus bancos de dados *InterBase* para o *Firebird*, devido ser uma tecnologia em desenvolvimento e de código aberto.

O *Firebird* é um banco de dados muito leve no que diz respeito ao tamanho de servidor, sua instalação completa tem cerca de 4MB, além disto não requer praticamente qualquer tipo de configuração manual, dispensando na maioria das vezes a necessidade de um DBA.

Cantu (2005) destaca a facilidade e rapidez na instalação do *Firebird*, não é necessário um espaço definido, pois o banco de dados cresce conforme os dados são inseridos e não diminui o tamanho quando os dados são excluídos. Assim esses arquivos excluídos, são reaproveitados conforme novos dados são inseridos.

4.3 SQLITE

Por mais que os dispositivos atuais tenham boa capacidade de armazenamento, não se comparam ao poder de processamento de máquinas usadas como servidores de bancos de dados. Neste contexto, o projeto SQLite foi iniciado em meados dos anos 2000, com a intenção de oferecer apoio até o ano 2050.

O SQLite é formado por um conjunto de bibliotecas escritas em C, que implementa um banco de dados Structured Query Language (SQL). Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo *Relational Database Management System* (RDBMS) separado. O SQLite lê e grava diretamente em arquivos de disco comuns. Um banco de dados SQL completo com várias tabelas, índices, acionadores e visualizações está contido em um único arquivo de disco. (SQLite).

O SQLite é um mini-SGBD, capaz de criar um arquivo, ler e escrever diretamente nele, seu código fonte é open source, portanto, é livre para uso para qualquer finalidade, comercial ou privada.

Ganhador do prêmio *Google O'Reilly 2005 Open Source Awards Winner!*, o SQLite gera um banco de dados que pode ser entregue junto com a aplicação, excelente para aplicações pequenas, podendo ser instalado facilmente utilizando o método NNF (*Next, Next, Finish*), sem as complicações da instalação de um cliente/servidor. (DEVMEDIA, 2007).

A DLL (*Dynamic-link library*, em português Biblioteca de Vínculo Dinâmico) *Sqlite3.dll* que está disponível para download no site do SQLite, é fundamental para o desenvolvimento de aplicações com ela, seja em Delphi, Java, entre outros.


5. DESENVOLVIMENTO

Este capítulo tem como objetivo principal apresentar e demonstrar a análise, desenvolvimento e funcionamento do sistema desktop e aplicativo mobile, ambos desenvolvidos para o controle de frequência dos acadêmicos da instituição FAFIMAN.

5.1 CENÁRIO ATUAL

Atualmente a instituição FAFIMAN - Fundação Faculdade Filosofia Ciências e Letras de Mandaguari, conta com o sistema educacional da empresa Elotech Informática e Sistemas.

O maior problema enfrentado pela instituição é forma utilizada para a realização da chamada em sala de aula, pois não obtem uma plataforma que simplifique o lançamento de presença e falta dos acadêmicos, tendo assim que gerar o livro de registro de classe mensalmente, conforme as figuras 13 e 14, que demonstra como são as listas de chamadas geradas atualmente.



FAFIMAN

RUA: RENÉ TÁCCOLA, nº 152

(44) 3233-1356 / (44) 3233-
HTTP://WWW.FAFIMAN.BR

Curso: ADMINISTRAÇÃO - BACHARELADO

Período: 2019

Disciplina: CONTABILIDADE GERAL

Professor: MARILDA DA SILVA BUENO

Turma: ADM 2019

Carga: 140

Cadeira: 286151

Controle de Notas e Frequência

Referente ao:

1° Bimestre ☐

2° Bimestre ☐

3° Bimestre ☐

4° Bimestre ☐

Tipo de Aula:

Nº de aulas:

Dia:

Mês:

RA	ALUNOS													NOTAS	FALTAS
8523	ALINE CRISTINA MATTOS														
8459	BRUNA REBECA QUEIROZ														
8468	CAIO CÉSAR CRACCO SILVA														
8511	DAVID GABRIEL DA SILVA														
8476	GIOVANA MARIA DE SÁ														
8441	GIOVANNI AUGUSTO BIANCO														
8405	HIAGO FELIPE BRITO DE OLIVEIRA														
8499	JOÃO VITOR MELQUIADES DOS SANTOS FONSECA														
8455	LEONARDO ALMEIDA DE OLIVEIRA														
8424	LUANA APARECIDA DE ALMEIDA CLEMENTE														
8434	LUCAS FERNANDO RODRIGUES														
8513	MATEUS MELENDI DE CARVALHO														
8544	MYLENA APARECIDA SILVINO														
8512	OTAVIO FELIX BOA														
8486	RAYANE STEFANI DE JESUS														
8438	RENATA FERNANDES LOPES DE OLIVEIRA														

Aulas Previstas: _____ Aulas Dadas: _____

Encerrado em: ____ de ____ de ____


Assinatura do _____

Assinatura do Professor _____

Elotech Informática e Sistemas (www.elotech.com.br)

Página 1 de 2

Figura 13 – Controle de Notas e Frequência, pág. 1.
 Fonte: Imagem cedida pela secretaria da instituição FAFIMAN.



FAFIMAN

RUA: RENÉ TÁCCOLA, nº 152

(44) 3233-1356 / (44) 3233-
HTTP://WWW.FAFIMAN.BR

Curso: ADMINISTRAÇÃO - BACHARELADO

Período: 2019

Disciplina: CONTABILIDADE GERAL

Professor: MARILDA DA SILVA BUENO

Turma: ADM 2019

Carga: 140

Cadeira: 286151

Controle de Notas e Frequência

Referente ao:

1° Bimestre ☐

2° Bimestre ☐

3° Bimestre ☐

4° Bimestre ☐

Tipo de Aula:

Nº de aulas:

Dia:

Mês:

RA	ALUNOS													NOTAS	FALTAS
8481	STHEFANIE CRISTINA APARECIDA DA SILVA														
8489	SUZIANE PEREIRA DE TOLEDO														
8495	TAMARA DE JESUS FARIA TIAGO														
7931	THIAGO ANTONIO DA SILVA TONEU														

Aulas Previstas: _____ Aulas Dadas: _____

Encerrado em: ____ de ____ de ____

Assinatura do _____

Assinatura do Professor _____

Elotech Informática e Sistemas (www.elotech.com.br)

Página 2 de 2

Figura 14 – Controle de Notas e Frequência, pág. 2.
 Fonte: Imagem cedida pela secretaria da instituição FAFIMAN.

5.2 ESCOPO DO PROJETO

Desenvolver uma aplicação *desktop*, para que sejam cadastrados os alunos, professores, grade curricular, matriz curricular, turmas e horários das aulas.

Desenvolver também uma aplicação *mobile*, compatível com os sistemas operacionais Android e IOS, para que desta forma os docentes possam realizar as chamadas de forma eletrônica, utilizando os seus próprios dispositivos móveis, mesmo que não contenha acesso à internet, ou seja, de forma *offline*, dispensando a forma tradicional, o papel.

Além disto deverá ser desenvolvida uma forma onde os dados sejam sincronizados, entre as aplicações, ou seja, entre a aplicação *desktop* e aplicação *mobile*.

5.3 CASO DE USO

Identificador	Caso de Uso	Prioridade
UC1	Ativar Servidor de Banco de Dados	Alta
UC2	Logar Administrador Sistema Desktop	Essencial

Tabela 3 – Caso de Uso.

5.3.1 ESPECIFICAÇÃO DE CASO DE USO

Caso de Uso: Ativar Servidor de Banco de Dados
Identificador: UC1
Atores: Administrador de Redes
Pré-condições: Nenhuma
Fluxo de eventos: <ol style="list-style-type: none">1. O administrador de redes, deverá inicializar o servidor da aplicação IAMHERE, informando a porta liberada para a conexão e clicar no botão Start.2. O servidor de aplicação irá validar se a porta informada está disponível para que o servidor seja inicializado.
Pré-condições: <ol style="list-style-type: none">1. Se a operação for realizada com sucesso, o servidor da aplicação, mudará seu status para ATIVO.2. Se não é apresentada uma mensagem informando que é necessário verificar a porta informada e o status do servidor será INATIVO.

Tabela 4 – Especificação de Caso de Uso.

5.4 DIAGRAMA DE CLASSE

O diagrama de classes é a parte central da UML (Linguagem de Modelagem Unificada, em inglês “*Unified Modelling Language*”), que serve de apoio para a maioria dos demais diagramas, pois é utilizado nas definições das classes utilizadas pelo sistema, determinando os atributos e métodos que cada classe tem, além de estabelecer como as classes se relacionam e trocam informações entre si. (GUEDES, 2009, p.31).

Nas figuras 15 e 16 são apresentados os diagramas gerados, referente ao projeto a ser desenvolvido neste trabalho.

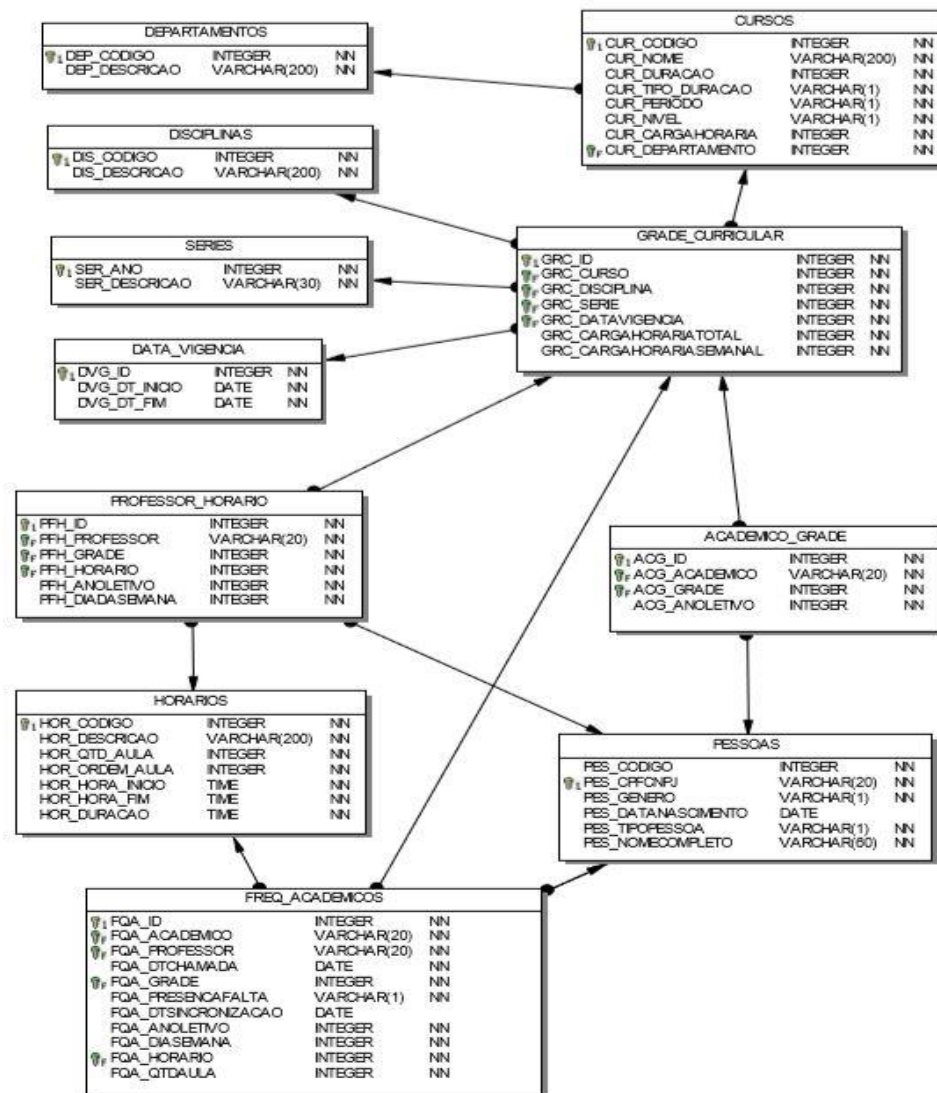


Figura 15 – Diagrama de Classe.

Fonte: Imagem gerada do diagrama desenvolvido no programa IBExpert.

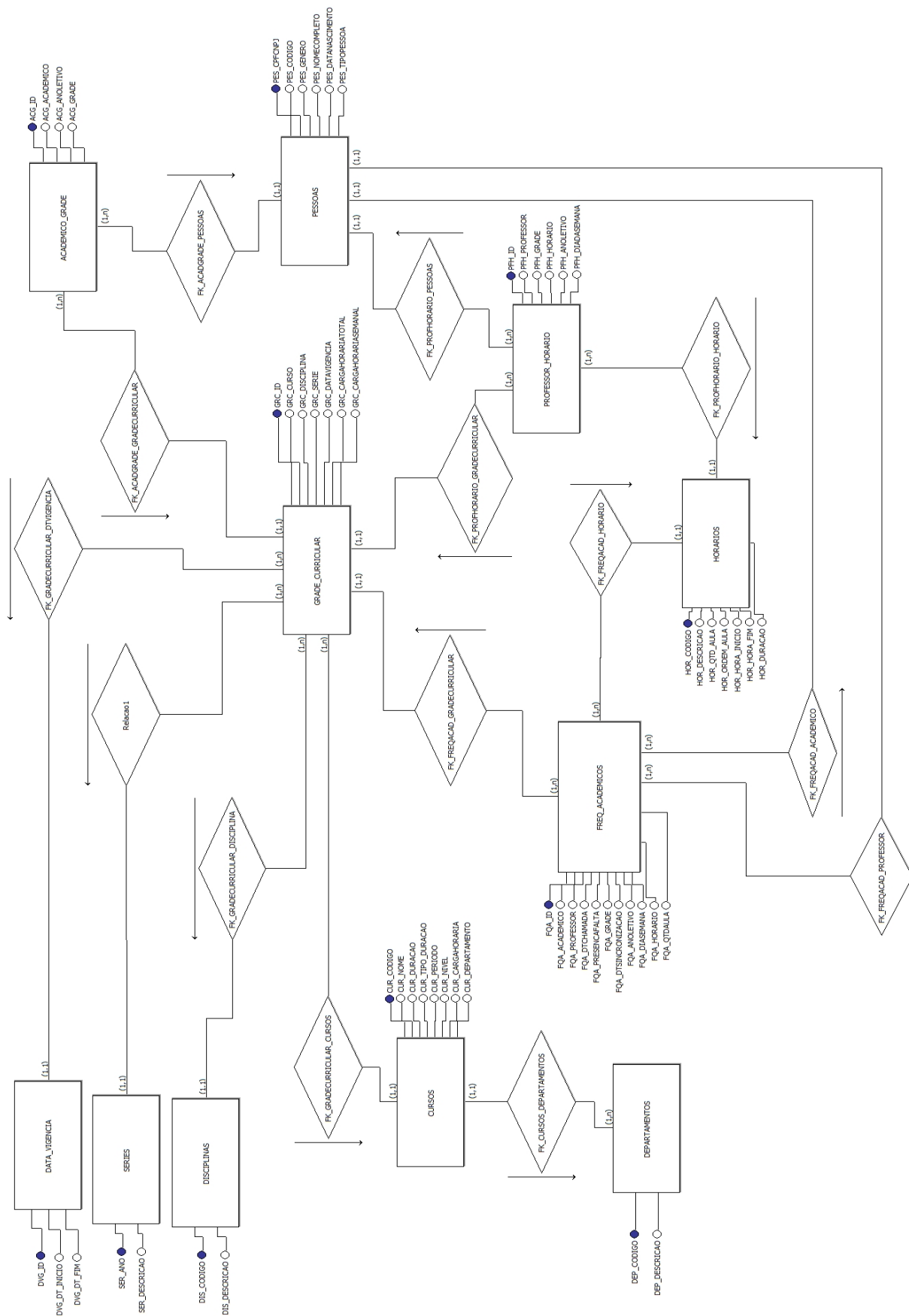


Figura 16 – Diagrama de Classe.

Fonte: Imagem gerada do diagrama desenvolvido no programa brModelo 3.0.

5.5 DICIONÁRIO DE DADOS

Tabela	DISCIPLINAS
Prefixo	DIS
Descrição	Armazenará as informações das disciplinas
Tabela	DEPARTAMENTOS
Prefixo	DEP
Descrição	Armazenará as informações dos departamentos
Tabela	CURSOS
Prefixo	CUR
Descrição	Armazenará as informações dos cursos
Observação	Obtem relacionamento com a tabela: DEPARTAMENTOS
Tabela	DATA_VIGENCIA
Prefixo	DVG
Descrição	Armazenará as informações das data de vigências
Tabela	SERIES
Prefixo	SER
Descrição	Armazenará as informações das séries
Tabela	HORARIOS
Prefixo	HOR
Descrição	Armazenará as informações dos horarios
Tabela	PESSOAS
Prefixo	PES
Descrição	Armazenará as informações das pessoas
Tabela	GRADE_CURRICULAR
Prefixo	GRC
Descrição	Armazenará as informações da grade curricular dos cursos
Observação	Obtem relacionamento com as tabelas: CURSOS, DISCIPLINAS, SERIES e DATA_VIGENCIA
Tabela	ACADEMICO_GRADE
Prefixo	ACG
Descrição	Armazenará as informações da grade curricular dos acadêmicos
Observação	Obtem relacionamento com as tabelas: GRADE_CURRICULAR e PESSOAS
Tabela	PROFESSOR_HORARIO
Prefixo	PFH
Descrição	Armazenará as informações da grade curricular dos horarios de aula dos professores
Observação	Obtem relacionamento com as tabelas: GRADE_CURRICULAR, PESSOAS e HORARIOS
Tabela	FREQ_ACADEMICOS
Prefixo	FQA
Descrição	Armazenará as informações das frequencias em sala de aula dos alunos
Observação	Obtem relacionamento com as tabelas: GRADE_CURRICULAR, HORARIOS e PESSOAS

Figura 17 – Prefixos das Tabelas do Banco de Dados.

Fonte: Informações extraídas apartir da estrutura do banco de dados.

Tabela	DEPARTAMENTOS							
Prefixo	DEP							
Descrição	Armazenará as informações dos departamentos							
Campos								
Nome	Tipo de Dado	Tamanho	Descrição	PK - Primary Key (Chave Primária)	FK - Foreign Key (Chave Estrangeira)	UK - Unique Key (Chave Única)	Not Null	Default
DEP_CODIGO	INTEGER		Código de identificação do departamento	PK_DEPARTAMENTOS			X	
DEP_DESCRICAO	VARCHAR	200	Nome do departamento				X	

Figura 18 – Dicionário de Dados – Tab. Departamentos.

Fonte: Informações extraídas apartir da estrutura do banco de dados.

Tabela	CURSOS							
Descrição	Armazenará as informações dos cursos							
Observações	Esta tabela possui um relacionamento com a tabela DEPARTAMENTOS							
Campos								
Nome	Tipo de Dado	Tamanho	Descrição	PK - Primary Key (Chave Primária)	FK - Foreign Key (Chave Estrangeira)	UK - Unique Key (Chave Única)	Not Null	Default
CUR_CODIGO	INTEGER		Código de identificação do curso.	PK_CURSOS			X	
CUR_NOME	VARCHAR	200	Nome do curso.				X	
CUR_DURACAO	INTEGER		Duração do curso, podendo ser, por exemplo 4 (4 anos).				X	
CUR_TIPO_DURACAO	VARCHAR	1	Tipo da duração do curso, podendo ser: [A] - ANUAL / [S] - SEMESTRAL / [M] - MENSAL				X	A
CUR_PERIODO	VARCHAR	1	Período que o curso será lecionado, podendo ser: [M] - MATUTINO / [V] - VESPERTINO / [N] NOTURNO				X	N
CUR_NIVEL	VARCHAR	1	Nível do curso, podendo ser: [G] - GRADUAÇÃO / [P] - PÓS-GRADUAÇÃO / [T] - TÉCNICO / [M] - MESTRADO / [D] DOUTORADO				X	G
CUR_CARGAHORARIA	INTEGER		Carga horária total do curso.				X	
CUR_DEPARTAMENTO	INTEGER		Chave estrangeira, que referencia o código da tabela de DEPARTAMENTOS.		FK_CURSOS_DEPARTAMENTOS		X	

Figura 19 – Dicionário de Dados – Tab. Cursos.

Fonte: Informações extraídas apartir da estrutura do banco de dados.

Tabela	DISCIPLINAS							
Prefixo	DIS							
Descrição	Armazenará as informações das disciplinas							
Campos								
Nome	Tipo de Dado	Tamanho	Descrição	PK - Primary Key (Chave Primária)	FK - Foreign Key (Chave Estrangeira)	UK - Unique Key (Chave Única)	Not Null	Default
DIS_CODIGO	INTEGER		Código de identificação da disciplina	PK_DISCIPLINAS			X	
DIS_DESCRICAO	VARCHAR	200	Nome da disciplina				X	

Figura 20 – Dicionário de Dados – Tab. Disciplinas.

Fonte: Informações extraídas apartir da estrutura do banco de dados.

5.6 TELAS DO SISTEMA

5.6.1 SISTEMA DESKTOP

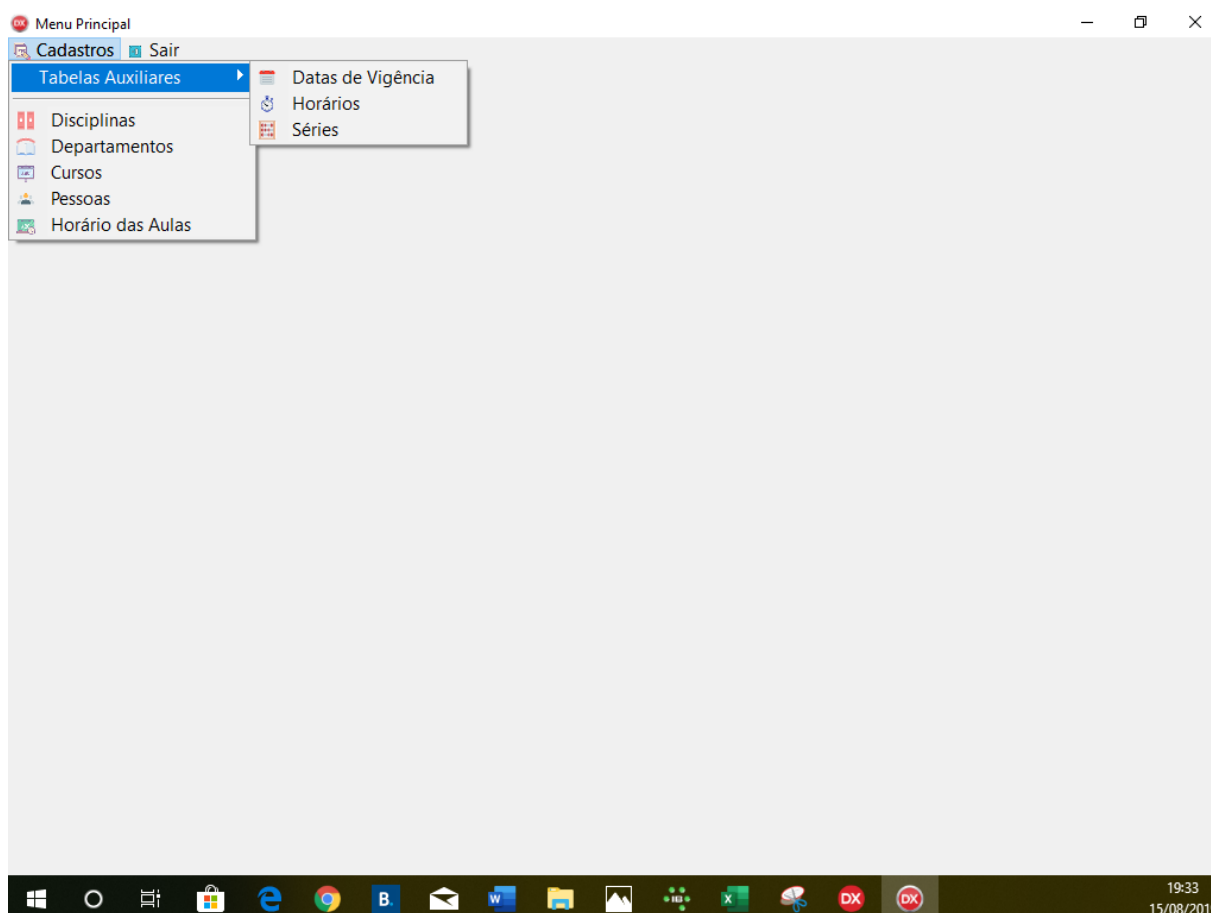


Figura 21 – Menu Principal e as opções disponíveis – Sistema Desktop.

Fonte: Particular.

P001 - Pesquisa Data de Vigência

Opções de Consulta
☒ Código

Listagem de Datas de Vigência

Código	Data Inicial	Data Final
1	01/01/2000	31/12/2020
8	01/08/2019	01/08/2019
9	02/08/2019	02/08/2019

F2 - Novo F3 - Alterar F4 - Excluir Esc - Sair

Dados
Código

Figura 22 – Tela de Pesquisa das Data de Vigência – Sistema Desktop.
Fonte: Particular.

M001 - Manutenções de Data de Vigência

Dados

Código	Data Inicial	Data Final
1	01/01/2000	31/12/2020

F5 - Gravar F7 - Cancelar

Figura 23 – Tela de Manutenção das Data de Vigência – Sistema Desktop.
Fonte: Particular.

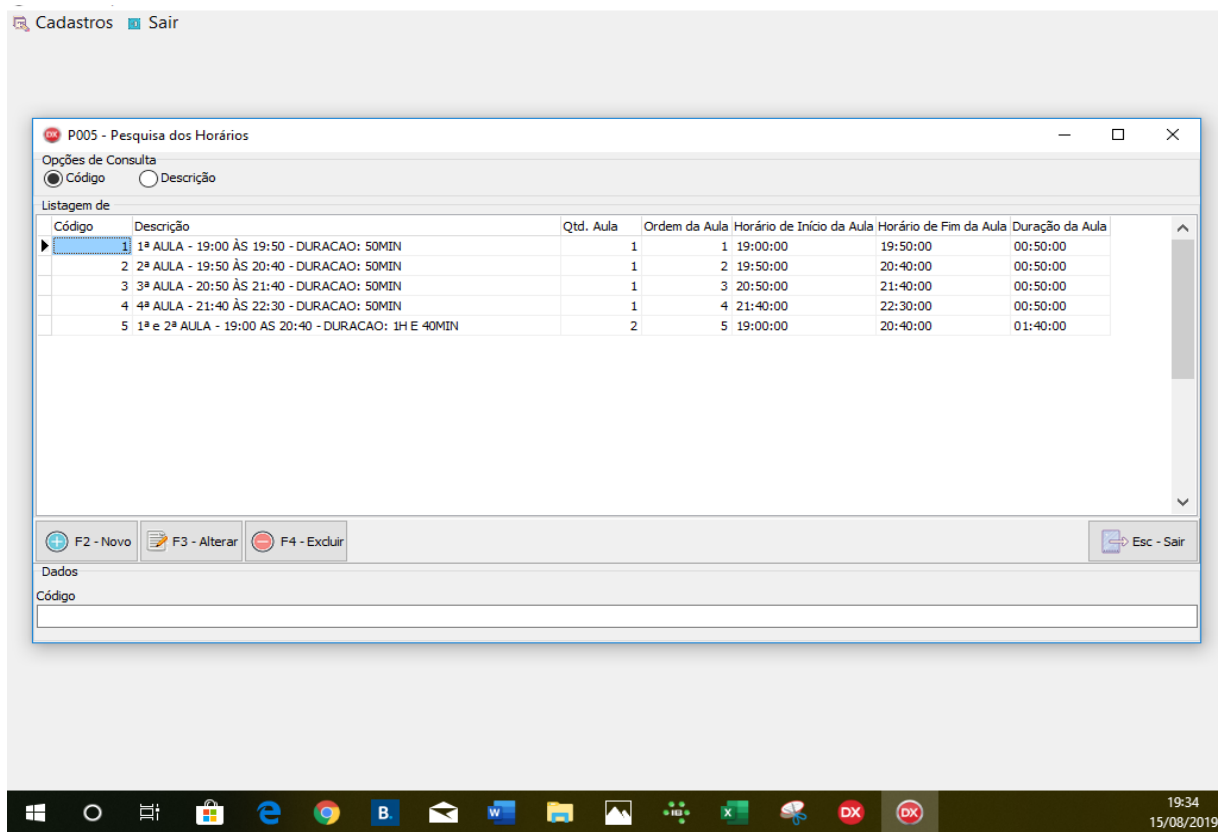


Figura 24 – Tela de Pesquisa dos Horários – Sistema Desktop.

Fonte: Particular.

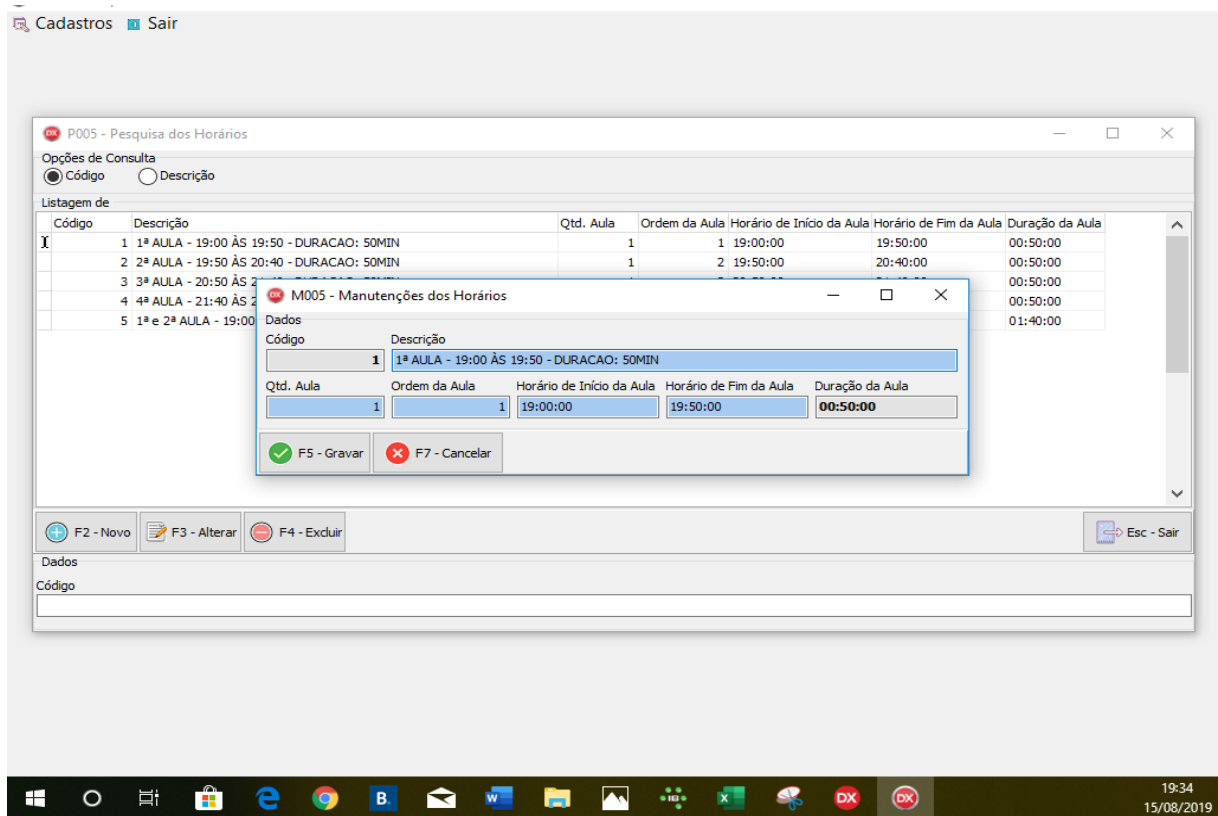


Figura 25 – Tela de Manutenção dos Horários – Sistema Desktop.

Fonte: Particular.

5.6.2 APLICATIVO MOBILE

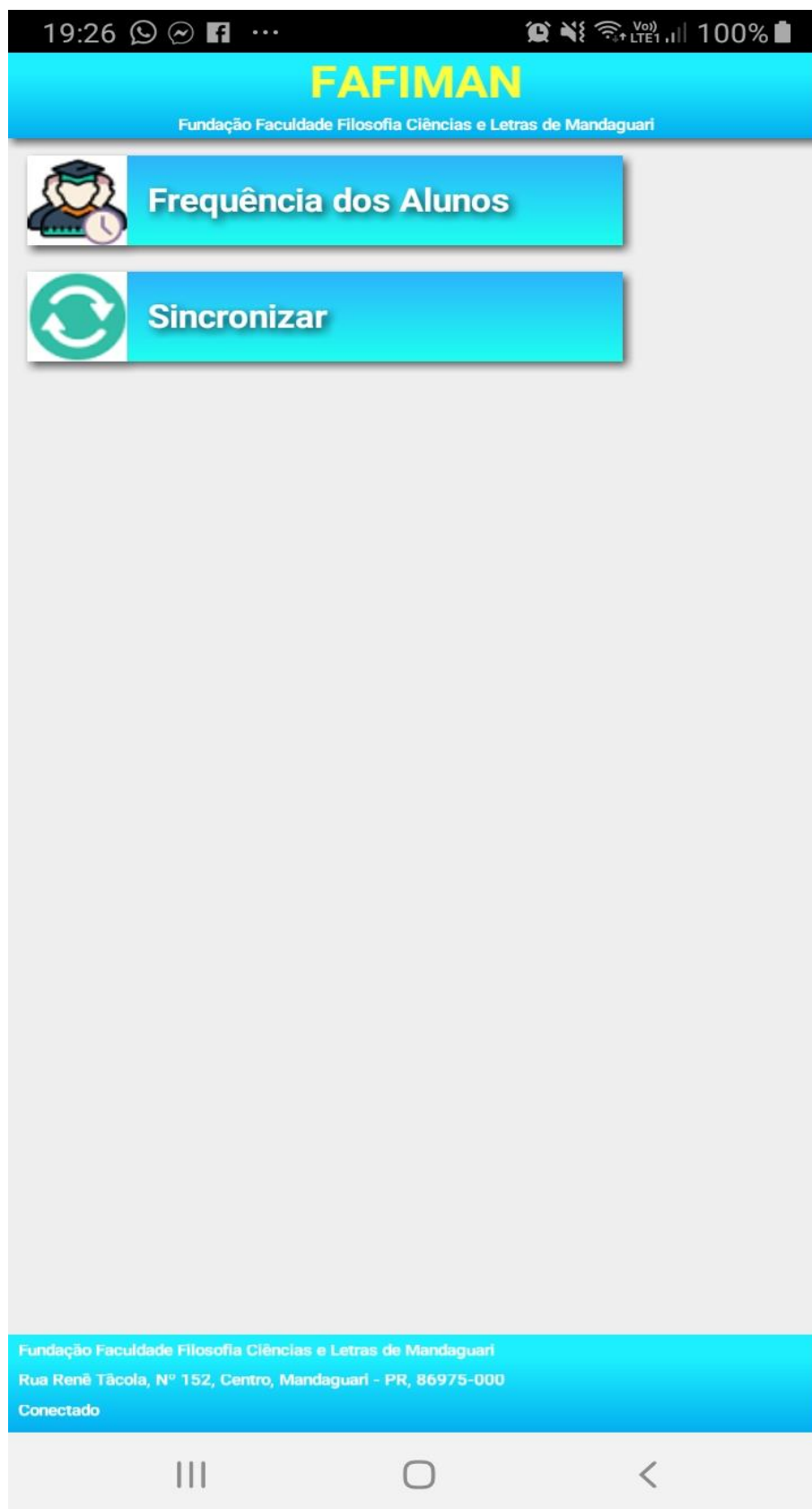


Figura 26 – Menu Principal – Aplicativo.

Fonte: Print do App Instalado no dispositivo móvel particular.

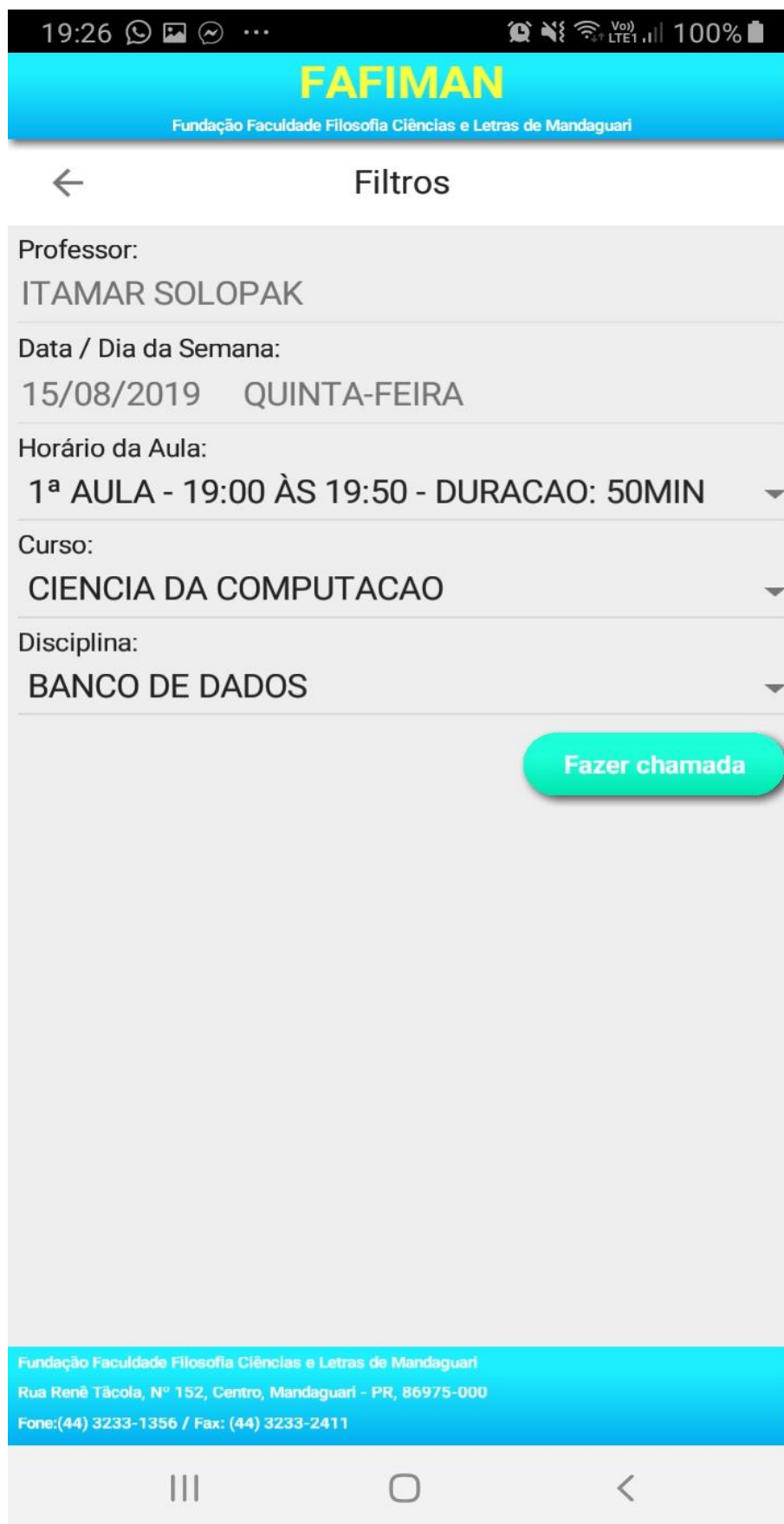


Figura 27 – Tela de filtro para realizar a listagem da chamada – Aplicativo.
Fonte: Print do App Instalado no dispositivo móvel particular.

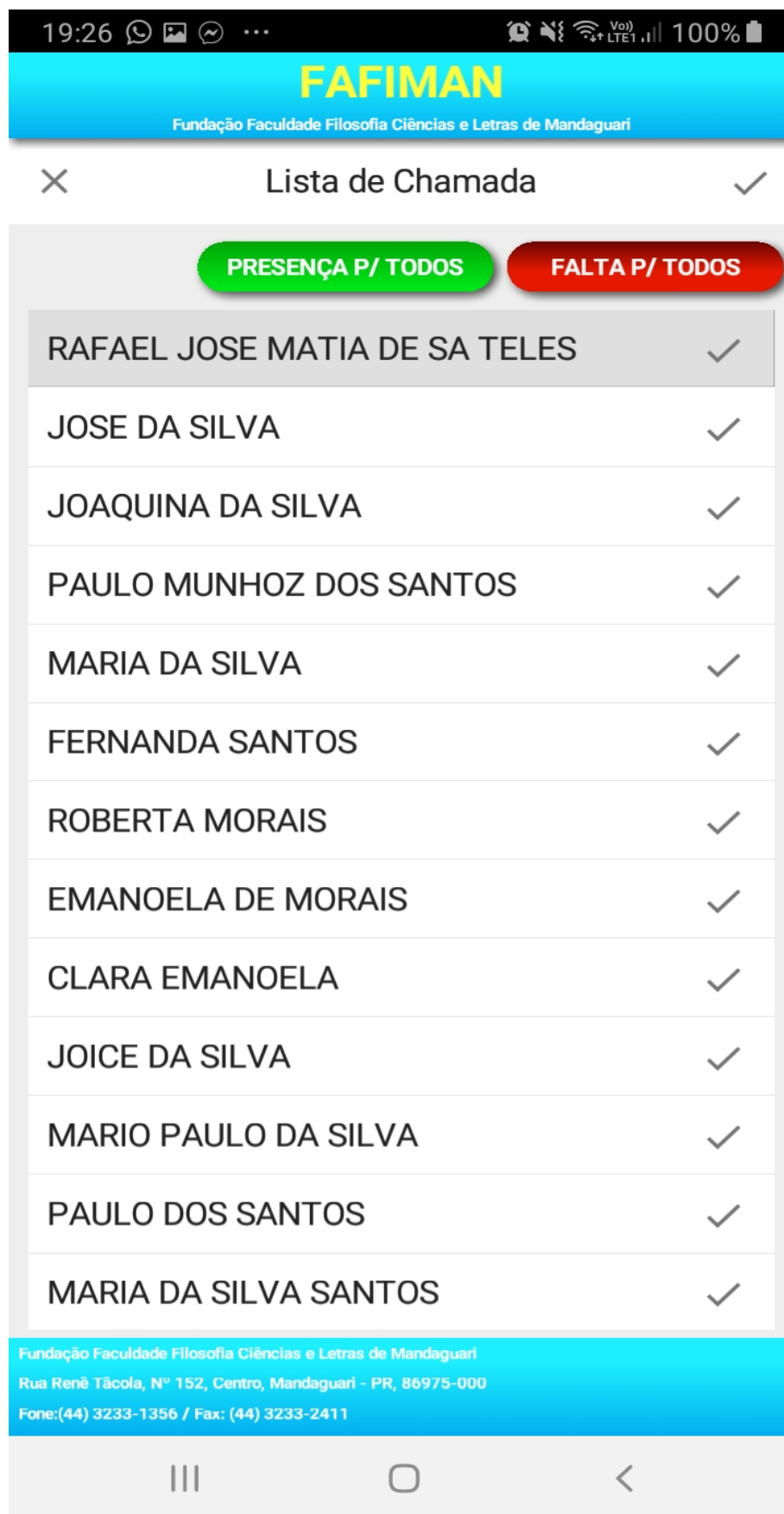


Figura 28 – Tela para realizar a chamada – Aplicativo.
Fonte: Print do App Instalado no dispositivo móvel particular.

6. CONCLUSÃO

A conclusão será apresentada apenas na versão final deste trabalho.

7. REFERENCIAS BIBLIOGRÁFICAS

FERRAZ, José Maria Gusman. **O papel nosso de cada dia**. Jaguariúna - Sp: Embrapa, [1991]. (Artigo). Disponível em: <http://webmail.cnpma.embrapa.br/down_hp/408.pdf>. Acesso em: 14 mar. 2019.

SEMIS, Laís. **É hora de colocar ordem nos documentos da escola**. São Paulo - Sp: Nova Escola Gestão, 2018. Disponível em: <<https://gestaoescolar.org.br/conteudo/1982/e-hora-de-colocar-ordem-nos-documentos-na-escola>>. Acesso em: 15 mar. 2019.

FONSECA, Hamilton. **Desperdício de papel nas empresas pode ser reduzido**. Curitiba-pr: Bem Paraná, 2018. Disponível em: <<https://www.bemparana.com.br/noticia/desperdicio-de-papel-nas-empresas-pode-ser-reduzido#.XUHKfOhKi1t>>. Acesso em: 16 mar. 2019.

MAZETTO JUNIOR, Milton; NAVARRO, Thaís; MAZETTO, Polliana Navarro. **As Verdades do Uso do Papel**. [s. L.]: Popscience, [20--]. Disponível em: <<http://popscience.com.br/as-verdades-do-uso-do-papel>>. Acesso em: 16 mar. 2019.

WRANY, Martina Gonzaga. **Gerenciamento eletrônico de documentos: um estudo de caso**. Rio Grande / Rs: Ufrgs, 2011. Disponível em: <<http://repositorio.furg.br/bitstream/handle/1/5935/Gerenciamento%20eletr%C3%B4nico%20de%20documentos%20-%20um%20estudo%20de%20caso.pdf?sequence=1>>. Acesso em: 20 mar. 2019.

FAYAD; SCHMIDT. **PROJETO DE SOFTWARE ORIENTADO A OBJETO: Frameworks**. Campina Grande: Ufcg, 2015. 7 p. Disponível em: <<http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/frame/oque.htm>>. Acesso em: 16 jun. 2019

AFONSO, Alexandre. **O que é Angular?** Uberlândia, Minas Gerais: Algaworks, 2018. Disponível em: <<https://blog.algaworks.com/o-que-e-angular/>>. Acesso em: 27 jul. 2019.

CARDOSO, HÉlio Carlos. **Curso de Delphi: O que é Delphi?**. Barra da Tijuca, Rio de Janeiro: Devmedia, [20--]. (O que é Delphi?). Disponível em:

<<https://www.devmedia.com.br/view/viewaula.php?idcomp=38188>>. Acesso em: 29 jul. 2019.

MENDES, Wende; SOUZA, Wilson. **Papo Reto: Vue.js**. Moema, São Paulo: Bluesoft Labs, 2018. Disponível em: <<https://labs.bluesoft.com.br/vuejs/>>. Acesso em: 28 jul. 2019.

EMBARCADERO. **Delphi: Perguntas frequentes do Delphi**. Austin, Eua: Embarcadero, [20--]. (Produtos). Disponível em: <<https://www.embarcadero.com/br/products/delphi/faq>>. Acesso em: 29 jul. 2019.

EMBARCADERO. **RAD Studio - Delphi: Perguntas frequentes do RAD Studio**. Austin, Eua: Embarcadero, [20--]. (Produtos). Disponível em: <<https://www.embarcadero.com/br/products/rad-studio/faq>>. Acesso em: 29 jul. 2019.

DALEPIANE, Filipe. **Entenda a Delphi Language**. Barra da Tijuca, Rio de Janeiro: Devmedia, 2014. Disponível em: <<https://www.devmedia.com.br/entenda-a-delphi-language/31353>>. Acesso em: 29 jul. 2019.

YAMAZACK, Wesley; MATOS, Gladstone; PRASS, Fábio Sarturi. **DevCast: Um bate-papo sobre o Delphi**. Barra da Tijuca, Rio de Janeiro: Devmedia, [20--]. (DevCast). Disponível em: <<https://www.devmedia.com.br/um-bate-papo-sobre-o-delphi/38783>>. Acesso em: 29 jul. 2019.

KAWATA, Fabricio Hissao. **Delphi: Artigo Iniciando a construção de apps Android no Delphi**. Barra da Tijuca, Rio de Janeiro: Devmedia, 2014. (Artigo). Disponível em: <<https://www.devmedia.com.br/iniciando-a-construcao-de-apps-android-no-delphi/29711>>. Acesso em: 30 jul. 2019.

GRANATYR, Jones. **Delphi: Artigo FireMonkey e FireDAC: Construindo uma aplicação completa – Parte 1**. Barra da Tijuca, Rio de Janeiro: Devmedia, 2017. (Artigo). Disponível em: <<https://www.devmedia.com.br/firemonkey-e-firedac-construindo-uma-aplicacao-completa-parte-1/38089>>. Acesso em: 30 jul. 2019.

DEVMEDIA. **Delphi: Construindo aplicativos Android no Delphi 10 Seattle**. Barra da Tijuca, Rio de Janeiro: Devmedia, 2016. (Artigo). Disponível em: <<https://www.devmedia.com.br/construindo-aplicativos-android-no-delphi-10-seattle/34092>>. Acesso em: 30 jul. 2019.

MOURÃO, Rodrigo. **Por que escolher o Firemonkey?** Rio de Janeiro - Rj: Rm Factory, 2017. Disponível em: <<http://blog.portalmrfactory.com.br/por-que-escolher-o-firemonkey/>>. Acesso em: 30 jul. 2019.

FLETCH. **Cross-platform development the FireMonkey way.** [s. L.]: Delphi Bistro, 2012. Desenvolvimento multiplataforma a maneira FireMonkey. Disponível em: <<http://delphibistro.com/?p=206>>. Acesso em: 30 jul. 2019.

FERREIRA, Rodrigo. **REST: Princípios e boas práticas.** [s. L.]: Blog Caelum, 2017. (Arquitetura, Inovação). Disponível em: <<https://blog.caelum.com.br/rest-principios-e-boas-praticas/>>. Acesso em: 30 jul. 2019.

CAMPOMORI, Cleber. **REST não é simplesmente retornar JSON: indo além com APIs REST.** São Paulo - Sp: Treinaweb, 2017. (WEB SERVICES). Disponível em: <<https://www.treinaweb.com.br/blog/rest-nao-e-simplesmente-retornar-json-indo-alem-com-apis-rest/>>. Acesso em: 31 jul. 2019.

ALVES, Gustavo Furtado de Oliveira. **O mínimo que você precisa saber sobre JSON para ser um bom programador!** [s. L.]: Dp Dicas de Programação, 2018. Disponível em: <<https://dicasdeprogramacao.com.br/o-que-e-json/>>. Acesso em: 31 jul. 2019.

ELMASRI, Ramez; NAVATHE, Shamkant B.. **SISTEMAS DE BANCO DE DADOS.** 4. ed. São Paulo - Sp: Pearson Education do Brasil Ltda, 2011. 798 p.

FIREBIRD. **Firebird.** [s. L.]: Firebird, [20--]. Disponível em: <<https://firebirdsql.org/en/release-notes/>>. Acesso em: 07 ago. 2019.

DEVMEDIA. **Onfigurando um banco de dados no Firebird.** Barra da Tijuca, Rio de Janeiro: Devmedia, 2008. Disponível em: <<https://www.devmedia.com.br/configurando-um-banco-de-dados-no-firebird/8137>>. Acesso em: 08 ago. 2019.

CANTU, Carlos Henrique. **Firebird: Introdução.** Barra da Tijuca, Rio de Janeiro: Devmedia, 2008. Disponível em: <<https://www.devmedia.com.br/firebird-introducao/7644>>. Acesso em: 08 ago. 2019.

CANTU, Carlos H.. **Conheça o Firebird em 2 minutos**. [s. L.]: Firebird, 2010. Disponível em: <https://www.firebirdnews.org/docs/fb2min_ptbr.html>. Acesso em: 09 ago. 2019.

MAGNO, Alexandre. **Firebird**: O que o torna um banco de dados atraente?. [s. L.]: Argos Tecnologia, [20--]. Disponível em: <https://www.firebaseio.com.br/imgdocs/amagno_fisl6.pdf>. Acesso em: 10 ago. 2019.

TEIXEIRA, José Ricardo. **LiveBindings no Firemonkey**: ligando dados visualmente. Barra da Tijuca, Rio de Janeiro: Devmedia, 2013. Disponível em: <<https://www.devmedia.com.br/livebindings-no-firemonkey-ligando-dados-visualmente/27011>>. Acesso em: 10 ago. 2019.

EMBARCADERO. **LiveBindings in RAD Studio**. [s. L.]: Embarcadero, [20--]. Disponível em: <http://docwiki.embarcadero.com/RADStudio/Rio/en/LiveBindings_in_RAD_Studio>. Acesso em: 11 ago. 2019.

DALEPIANE, Filipe. **LiveBindings**: a evolução no Delphi. Barra da Tijuca, Rio de Janeiro: Devmedia, 2014. Disponível em: <<https://www.devmedia.com.br/livebindings-a-evolucao-no-delphi/30142>>. Acesso em: 11 ago. 2019.

BICALHO, Conrado. Banco de Dados em Dispositivos Móveis instituto de Ciências Exatas e Biológicas. Universidade Federal de Ouro Preto – UFOP.