VINÍCIUS BARBOSA HONÓRIO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA COMUNICAÇÃO ENTRE PROFISSIONAIS DA SAÚDE

LAVRAS-MG

2019

VINICIUS BARBOSA HONÓRIO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA COMUNICAÇÃO ENTRE PROFISSIONAIS DA SAÚDE

Monografia apresentada à Universidade Federal de Lavras, como parte das exigências do Curso de Sistemas de Informação, para a obtenção do título de Bacharel.

Prof. D.Sc. Ramon Gomes Costa
Orientador

LAVRAS – MG
2019

VINÍCIUS BARBOSA HONÓRIO

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL PARA COMUNICAÇÃO ENTRE PROFISSIONAIS DA SAÚDE

Relatório de estágio apresentado à Universidade Federal de Lavras como parte das exigências do Curso de Sistemas de Informação para obtenção do título de Bacharel em Sistemas de Informação.

APROVADA em 22 de Novembro de 2019.

Prof. DSc. Ramon Gomes Costa UFLA
Prof. DSc. Renata Teles Moreira UFLA
Guilherme Camilo Rezende AUTOMATIZA

Prof. D.Sc. Ramon Gomes Costa
Orientador

Prof. Ramon Gomes Costa
DCC / UFLA

LAVRAS – MG
2019

AGRADECIMENTOS

Primeiramente agradeço a Deus, por todas as graças recebidas e que ainda irei receber, pois sei que nunca me abandonará.

Agradeço aos meus pais, Luiz Carlos e Selma por me proporcionarem a oportunidade de concluir esta fase de minha vida, também agradeço por toda a paciência e ensinamentos passados.

Agradeço a minha irmã, Ana Cláudia, pelo companheirismo durante esta etapa. Aos meus amigos e colegas de curso, que de alguma maneira contribuíram para que esta etapa fosse concluída.

Agradeço ao meu orientador, Prof. Dr Ramon Gomes Costa (DCC - UFLA), agradeço pela paciência e por toda a atenção oferecida durante todas as atividades do estágio.

Agradeço todos os docentes da Universidade Federal de Lavras (UFLA), em especial aqueles que o aluno teve a oportunidade de cursar as disciplinas lecionadas pelos mesmos.

Agradeço também a empresa Automatiza por oferecer a grande oportunidade de realização do estágio, proporcionando ao aluno atividades vivenciais importantes para a carreira de um graduado em Sistemas de Informação.

RESUMO

O relatório consiste em descrever a atuação do aluno no papel de desenvolvedor de *software* em um aplicativo *Mobile* denominada de Opinio. O aplicativo tem como objetivo ajudar na discussão de casos clínicos e esclarecimento síncrono de dúvidas entre profissionais especializados e não especializados, para a promoção de uma segunda opinião e, assim, aumentar a capacidade de solucionar e resolver os quadros clínicos, agilizar os procedimentos, melhorar o atendimento e minimizar os riscos. Serão apresentadas descrições das atividades desenvolvidas, o processo de desenvolvimento e as tecnologias utilizadas.

Palavras-chave: Desenvolvimento. Projeto. Software. Tecnologia. *Mobile*.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Por muito tempo foi mantido o paradigma de separação entre teoria e prática e, em consequência, entre a vivência empresarial e o meio acadêmico. Na busca de se quebrar este paradigma, o presente trabalho, ao cumprir os requisitos de formação do curso de Bacharelado em Sistemas de Informação da Universidade Federal de Lavras, pretende associar a vivência empresarial com os conteúdos aprendidos, ao longo do curso, afim de que essa junção possa resultar em estímulos à produção literária quanto ao tema proposto.

Durante o período de estágio, o estudante é capaz de evoluir e desenvolver o conhecimento e habilidades para a função a ser desempenhada. Inserido em um ambiente em que pessoas qualificadas exercem diversas funções, é possível aprimorar o trabalho em equipe, disciplina e o entendimento da importância dos prazos estipulados. Com o trabalho prático do estágio, fica mais fácil identificar as diversas atuações profissionais, em sua área, além de ser melhor preparado para o mercado de trabalho.

A experiência de presenciar a rotina de uma organização prepara o aluno para o mercado de trabalho e oferece-lhe a oportunidade de utilizar os conhecimentos adquiridos, durante a graduação em situações para a resolução de problemas reais. Além disso, o estágio alavanca as relações sociais e comunicativas pelos trabalhos que são realizados em equipe.

Além disto, por meio desta apresentação integrada do plano de estágio e conceitos aprendidos pela matriz curricular e pelas publicações acadêmicas, objetivou-se buscar melhorias, em processos já desenvolvidos, analisar as dificuldades encontradas, bem como explanar sobre as fases do processo de sua criação. Sendo assim, geram-se novos conteúdos para que outras empresas ou outros graduandos e bacharéis possam usar em suas pesquisas.

Deste modo, o trabalho foi dividido em quatro momentos. No primeiro, são feitas considerações iniciais acerca da empresa onde foi realizado o estágio e do projeto a ser realizado, definido no plano de estágio. Num segundo momento, apresentaram-se conceitos e demais conhecimentos teóricos que embasaram a realização do projeto durante o estágio. Feita a revisão de literatura, passou-se à metodologia do projeto, bem como às atividades realizadas e às dificuldades apresentadas, ao longo do processo de desenvolvimento do software. Por fim, foi feita uma análise integrada e crítica acerca da aplicação dos conhecimentos teóricos nas atividades desenvolvidas ao longo do projeto.

2 CONSIDERAÇÕES INICIAIS

Em Janeiro de 2016, a Automatiza realizou um processo seletivo para a contratação de um estagiário para a equipe de desenvolvimento de soluções, com base em Tecnologia da Informação (TI). O estagiário autor deste relatório foi ser aprovado, após uma bateria de entrevistas e testes de aptidão. Durante o estágio, as principais atividades do estagiário foram: desenvolver sistemas e aplicações; realizar manutenção de sistemas e aplicações; projetar e modelar bancos de dados; implantar aplicações em servidores Web e selecionar recursos de trabalho, tais como metodologias de desenvolvimento de sistemas e gerenciamento de tarefas.

O estágio teve como foco principal proporcionar uma experiência profissional, adquirida pelas vivências, na função de desenvolvedor de sistemas Web/Mobile, além de vivenciar as relações entre a teoria que adquirida na faculdade e a prática profissional que é estar imerso, em uma empresa privada, refletindo sobre a aprendizagem acadêmica de um trabalho cotidiano em um desenvolvedor na Automatiza, a fim de obter a experiência e vivência no ambiente privado, ou seja, uma experiência fora do ambiente acadêmico.

Durante o período do estágio, a Automatiza fechou um contrato para desenvolver uma aplicação mobile, que atendesse os requisitos propostos pelo cliente. A aplicação se tratava de um aplicativo de celular que funcionasse nas principais plataformas de telefonia do mercado, àquela época, em que eram os sistemas operacionais Android [1] criados e mantidos pela empresa Google [2]e o iOS [3]criado e mantido pela empresa Apple[4].

2.1 Descrição da empresa

A Automatiza Tecnologia e Automação LTDA é uma empresa fundada, em 1986, na cidade de Lavras, onde mantém sua sede e um quadro de 20 funcionários, os quais ocupam os cargos de desenvolvedores e analistas de suporte técnico, além de cargos administrativos e cargos de limpeza e manutenção do espaço de trabalho.

A Automatiza é uma empresa que atua na área de Tecnologia da Informação, cujo principal enfoque é o desenvolvimento de softwares para a prestação de serviços indispensáveis a clientes de cartórios e da iniciativa privada de todo o país, de modo que esses softwares atendam as necessidades de cada cliente.

[1] https://www.android.com/
[2] https://www.google.com/
[3] https://www.apple.com/br/ios/ios-13/
[4] https://www.apple.com/

O principal perfil de clientes da empresa são os Cartórios de Títulos e Documentos e Pessoas Jurídicas e Cartórios de Protesto divididos por todo o território brasileiro.

Para atender a este perfil, a empresa trabalha com o *software* denominado ProlexNet. Este software é robusto, de alta qualidade e de ótima usabilidade e com o diferencial de ser totalmente utilizado na Web, ou seja, os clientes poderiam ter acesso aos seus registros e criá-los tanto no ambiente de trabalho quanto dentro de duas casas. O ProlexNet tinha como principal compromisso com o usuário, sua consistência e sua facilidade de gerar e administrar os registros típicos, realizados pelos cartórios, com um fluxo intuitivo que objetivava minimizar os erros dos usuários e garantir segurança e eficiência à conferência dos dados do usuário.

Além disto, a empresa apresenta um diferencial também no que diz respeito ao suporte técnico ao usuário, que é realizado de forma eficiente, rápida e com clareza, sendo possível, por meio desse atendimento, auxiliar mesmo àqueles que não possuem conhecimento ou experiência em informática. Esse atendimento é feito pelo acesso remoto ao computador do usuário, a fim de que se possam sanar todas as dúvidas do cliente e resolver os mais diversos tipos de problemas.

Apesar de o ProlexNet ser o principal produto da empresa e ser o *software* que garante maior fatia de lucros, a empresa também atua como uma fábrica de softwares, ou seja, a empresa desenvolve projetos paralelos ao ProlexNet, como é o caso do aplicativo Opinio, no qual o estagiário pôde colaborar desde a concepção até a entrega e sua publicação para permitir que o aplicativo seja baixado, instalado e utilizado pelos seus usuários. O estágio foi realizado durante o desenvolvimento do aplicativo Opinio, desta forma, o enfoque deste trabalho é neste determinado projeto e, em suas particularidades, pois foi neste aplicativo que o estagiário atuou na maior parte do período de estágio.

2.2 Descrição do projeto

O projeto Opinio é o desenvolvimento de um aplicativo que atenda as especificidades da Telemedicina e possa ser implantado em qualquer Hospital.

Primeiramente, deve-se ressaltar que o termo Telemedicina carrega uma diversidade de conceitos da literatura, porém, para fins deste trabalho, será usado um entendimento comum a que a Telemedicina está relacionada a todo o uso de tecnologia para tornar possíveis os cuidados à saúde (WEN, 2008).

Segundo Wen (2008), a Telemedicina trata-se de um conceito que vem sendo consolidado e apresentando grandes evoluções, as quais se devem, em grande parte, aos investimentos em pesquisas nas áreas de telecomunicação, tecnologia da informação e computação. O desenvolvimento tecnológico propiciou uma redução de custos a uma série de tecnologias, tornando-as acessíveis e expandido seu horizonte de aplicação. Esse fato possibilitou a aplicação dos conceitos da Telemedicina, nas diversas áreas da saúde, possibilitando o surgimento da Telemedicina no Brasil.

O aplicativo permite uma comunicação, em tempo real, entre os médicos e profissionais de saúde, proporcionando-lhes uma segunda opinião imediata por meio da troca de informações entre os usuários solicitantes e o especialista, a fim de proporcionar melhor resolutividade no atendimento, propiciando o esclarecimento de dúvidas entre o especializado e o não especializado.

Com base nesses conceitos e través dos avanços tecnológicos, acima mencionados, é que foi possível a criação do atual projeto.

O projeto do Opinio, com o qual o estagiário pôde colaborar, tem como objetivo ser uma solução inovadora que facilita a comunicação entre os profissionais da saúde, a fim de melhorar a qualidade no atendimento aos pacientes por uma aplicação *mobile*.

Ainda, na visão de Wen (2008), um dos pontos importantes da Telemedicina, no Brasil, é que ela propõe um rompimento com o paradigma educacional atual, uma vez que sugere um modelo multiprofissional e transdisciplinar, a fim de se alcançar uma aplicação efetiva de soluções tecnológicas para fins de otimização da educação, planejamento da logística, regulação da assistência médica e implementação de métodos que proporcionem pesquisas baseadas em estratégias de gestão de sustentabilidade e em desenvolvimento de novos modelos de medicina.

3 TECNOLOGIAS UTILIZADAS

A teoria e a prática não são dissociadas uma da outra. Logo objetiva-se, neste tópico, apresentar as principais bases conceituais para o entendimento do trabalho que foi realizado no estágio. Portanto, são relatadas as principais tecnologias e ferramentas utilizadas nas atividades, bem como as motivações à escolha de uma tecnologia em detrimento de outra.

Durante o período do estágio, os conceitos e aplicações de três áreas da tecnologia da informação foram necessários, constantemente: desenvolvimento, gerência de configuração e gerência de projetos.

Os conceitos abordados, na área de desenvolvimento, são: a linguagem C#.Net[5], o *Entity Framework*[6], o padrão *Model View Controller*[7] (MVC), o Xamarin[8], o Xamarin.Forms [9]e o XAML[10] além do banco de dados SQL Server[11] para o armazenamento dos dados.

Sobre a gerência de configuração, será relatado o uso de versionamento de código utilizando o Git/Git-TFS[12]. Para o gerenciamento de projetos, são abordadas as principais características do framework SCRUM[13], tecnologia empregada ao gerenciamento de projetos, juntamente com o Trello[14] para gerenciar as atividades implementadas pela organização

Para diminuir o trabalho necessário de criar uma *interface*, para cada sistema operacional, é foi utilizado o framework Xamarin.Forms, o qual facilita ao desenvolvedor escrever os códigos para as interfaces e considera as diferenças de cada sistema operacional ao fornecer uma biblioteca comum. Para se tornarem transparentes essas diferenças, a interface é usada com uma linguagem de marcação chamada XAML *(Extensible Application Markup Language*) (PROCEDI, 2016).

Ainda, segundo Procedi (2016), o XAML é uma linguagem de marcação declarativa que simplifica o desenvolvimento das interfaces de usuário e, também, a escrita dos códigos, pois é uma linguagem similar ao HTML[15].

---

5 https://docs.microsoft.com/pt-br/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework
6 https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/ef/overview
7 https://docs.microsoft.com/pt-br/aspnet/core/mvc/overview?view=aspnetcore-3.0
8 https://docs.microsoft.com/pt-br/xamarin/
9 https://docs.microsoft.com/pt-br/xamarin/xamarin-forms/
10 https://docs.microsoft.com/en-us/dotnet/desktop-wpf/fundamentals/xaml
11 https://www.microsoft.com/pt-br/sql-server/default.aspx
12 https://docs.microsoft.com/en-us/azure/devops/repos/git/overview-2013?view=tfs-2013
13 https://scrumguides.org/
14 https://trello.com/
15 https://www.w3.org/html/

A maior vantagem do Xamarin é sua forma de programar. Ele utiliza apenas uma linguagem de programação, no caso C#.Net. É notório que, para se aperfeiçoar em uma linguagem de programação e, em um ambiente de programação, exige tempo, dedicação e prática, sendo assim, para se criar alguma aplicação, o programador precisa ter conhecimento sobre a linguagem de programação nativa e, também, do ambiente de desenvolvimento de cada plataforma. Assim, são necessários um custo e um tempo maior no desenvolvimento da aplicação, pois, dependendo do caso, necessário é importante um treinamento da equipe de desenvolvimento para que ela se adapte em cada um desses ambientes (PROCEDI, 2016).

A utilização do Xamarin foi também pelo fato da empresa, onde o estágio foi realizado, possuir uma associação Silver[16] junto à Microsoft[17], o que demonstra a capacidade dos funcionários no domínio da linguagem.

3.1 A linguagem de programação C#

A linguagem de programação C# (pronunciamos “cê sharp”) é uma linguagem de programação orientada a objetos que permite aos desenvolvedores um ambiente propício para criar uma variedade de aplicativos, que rodam dentro do ambiente .NET Framework (MICROSOFT, 2018).

De acordo com a MICROSOFT (2018), o .NET Framework é um termo bastante comum quando se trabalha com a linguagem C# e outras tecnologias da Microsoft. Ele visa ser uma plataforma única para o desenvolvimento e execução de sistemas e aplicações. A ideia para o emprego dessa plataforma é que o programador escreva seu código para uma plataforma e não para um sistema ou dispositivo específico.

A princípio, o C# funcionava apenas no sistema operacional Windows, mas, hoje, funciona 100% em Linux [18] e MacOS [19], graças às iniciativas da Microsoft de apoiar o Software Livre, com a introdução ao NET Core [20] (sua plataforma *open-source* e multiplataforma).

Como a empresa, possuir profissionais altamente capacitados e certificados pela Microsoft, optou-se por escrever a aplicação *mobile* em C#.Net. A linguagem foi responsável não só estruturação do projeto em si, mas também pelo tratamento dos dados da aplicação,

---

16 https://docs.microsoft.com/pt-br/partner-center/learn-about-competencies
17 https://www.microsoft.com/pt-br/
18 https://www.linux.org/
19 https://support.apple.com/pt-br/macos
20 https://docs.microsoft.com/pt-br/aspnet/core/?view=aspnetcore-3.0

criação das *APIs (Application Programming Interface)*, além de organizar e distribuir os dados recuperados junto ao banco de dados.

3.2 *Entity framework*

Em se tratando de desenvolvimento de *software*, escrever códigos SQL é um trabalho muito demorado e trabalhoso, mas é um trabalho fundamental em todos os sistemas informatizados.

No ambiente de desenvolvimento de software, é muito comum encontrar frameworks que façam uma abstração da escrita SQL ao desenvolvedor o que facilita o seu trabalho. Esse tipo de framework é denominado *ORM (Object Relational Mapper).*

O *Entity Framework* é *ORM* da linguagem C#.Net. Segundo a MICROSOFT (2017), o *Entity Framework* é um conjunto de tecnologias do *ADO.NET*[21],desenvolvidas para facilitar o acesso às aplicações de bases de dados de diversos tipos, especialmente, bancos de dados, como SQL Server, Oracle[22], PostgreSQL[23], dentre outros que dão suporte ao desenvolvimento de aplicativos de software orientado a dados.

Conforme a MICROSOFT (2017), os desenvolvedores de aplicativos orientados a dados que utilizam essas tecnologias possuem necessidade de realizar dois objetivos: modelar as entidades, as relações e a lógica dos problemas de negócios que estão solucionando e trabalhar com mecanismos para armazenar e recuperar os dados.

Ainda, de acordo com a MICROSOFT (2017), o *Entity Framework* permite que os desenvolvedores trabalhem com dados, na forma de objetos, sem se preocupar com as tabelas e colunas de banco de dados subjacentes, em que esses dados são armazenados, ou seja, os desenvolvedores podem trabalhar em um nível mais alto de abstração, ao lidar com dados e podem criar e manter aplicativos orientados a dados com menos códigos que em aplicativos tradicionais.

O *Entity Framework* fornece as maneiras de consultar um modelo conceitual e retornar objetos: *LINQ to Entities* (consulta integrada à linguagem), para consultar tipos de entidade que são definidos, em um modelo conceitual, e *Entity SQL* que é uma escrita de linguagem SQL que funciona diretamente com entidades e dá suporte a conceitos de modelo de dados de entidade MICROSOFT (2017).

---

[21] https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/
[22] https://www.oracle.com/br/database/
[23] https://www.postgresql.org/

Durante o estágio, o Entity Framework foi largamente utilizado no *backend* da aplicação para otimizar a criação e manipular os dados, além de escrever consultas de forma mais rápida, dinâmica e eficiente, a fim de recuperar, tratar e alimentar o banco de dados.

3.3 *Model View Controller* (MVC)

O padrão de projeto *Model View Controller* (MVC) propõe dividir a aplicação em três camadas: modelo, visão e controlador. Atualmente o MVC é um padrão de projeto conhecido no ambiente corporativo de TI e é muito útil para arquitetar aplicações de softwares interativos. Sua ideia principal é separar as interfaces de usuário da camada de dados (OMAR, 2004). A seguir serão apresentados os objetivos de cada camada.

- Model: o modelo tem por objetivo definir a essência das regras de negócio, envolvendo as classes do sistema, persistência e acesso aos dados. É responsável por gerenciar as ações recebidas pelo controlador (SOARES, 2015).
- View: a camada de view está relacionada à visualização da aplicação Web, ou seja, às telas que serão mostradas ao usuário. Nessa camada, apenas os recursos visuais devem ser implementados, como janelas, botões e mensagens (SOARES, 2015).
- Controller: o controlador é um código intermediário entre as regras de negócio (camada Model) e a camada View; recebe todas as requisições do usuário. Realiza o processamento de dados informados pelo usuário, repassando-os entre as outras camadas (SOARES, 2015).

Um dos principais benefícios em utilizar o MVC é o menor acoplamento do código, pois é possível o desenvolvimento em paralelo da interface com as regras de negócio. No estágio, o padrão de projeto MVC foi utilizado, durante o processo de desenvolvimento do aplicativo, desde sua modelagem até a etapa de testes.

O padrão MVC também é um facilitador pois oferece alguns benefícios, na construção de um software, como a facilidade no processo de divisão das tarefas, facilidade no reaproveitamento do código, legibilidade do código, facilidade em agregar mais recursos ao projeto, entre outras facilidades.

Durante o estágio, esse padrão foi importante facilitando a organização das classes através da divisão em camadas. A Separação da regra de negócio da camada de exibição dos dados ajuda tanto na reutilização de códigos como na manutenção do sistema.

3.4 Xamarin

O Xamarin é uma plataforma que tem como objetivo principal diminuir a necessidade de escrever códigos, separadamente, para cada tipo de sistema operacional, para dispositivos móveis, sejam eles Android, iOS, entre outros, tornando possível produzir aplicativos mobile escrevendo esses códigos em uma mesma linguagem de programação o C#.Net (RADI, 2016).

Os aplicativos criados, usando Xamarin, são nativos, ou seja, eles exploram toda a capacidade da linguagem de programação, mesmo que não seja a mesma consumida pela plataforma Xamarin. Porém o Xamarin não assegura que seja abolida a necessidade de desenvolvimento repetido, em especial, na parte da *interface*, em que ainda é preciso escrever os códigos específicos para cada tipo de sistema operacional- alvo. O Xamarin certifica que todo o código da regra de negócio, acesso ao banco de dados ou comunicação com servidor sejam implementados uma vez (RADI, 2016).

A experiência do desenvolvedor do Xamarin.Forms envolve a criação e manipulação da interface do usuário, na linguagem XAML, com a adição do *code-behind*, que opera na interface do usuário. À medida que os aplicativos são modificados e aumentam em tamanho, podem surgir problemas complexos de manutenção. Esses problemas incluem o forte acoplamento entre os controles da interface do usuário e a lógica de negócios, o que aumenta o custo de fazer modificações na interface do usuário e a dificuldade de teste de unidade desse código. MICROSOFT (2017).

O padrão MVVM (Model-View-ViewModel) ajuda a separar corretamente a lógica de negócios e de apresentação de um aplicativo da interface do usuário. Manter uma separação clara entre a lógica do aplicativo e a interface do usuário ajuda a resolver inúmeros problemas de desenvolvimento e pode facilitar o teste, a manutenção e a evolução de um aplicativo. Ele também pode melhorar, consideravelmente, as oportunidades de reutilização de código e permite que desenvolvedores e designers de interface do usuário colaborem mais facilmente ao desenvolver suas respectivas partes de um aplicativo MICROSOFT (2017).

A View representa a interface de usuário, pode ser uma tela, uma janela ou uma interface qualquer de entrada e saída de dados. A View não executa nenhuma operação no sistema, ficando responsável apenas por apresentar as informações na tela e receber os comandos do usuário. A ligação entre a View e a ViewModel ocorre pela propriedade *BindingContext* da View, a qual armazena uma referência da ViewModel que, por meio da propriedade Binding, comunica-se com a ViewModel, passando e recebendo dados, enviando comandos e recebendo notificações da ViewModel (MICROSOFT, 2017).

A *ViewModel* é a classe que fornece os dados e executa os comandos que a View necessita. Ela não possui nenhuma referência com a View e sua comunicação ocorre pela exposição de suas propriedades e do Binding feito pela View. É dessa forma que a View é notificada quanto às alterações nos dados da ViewModel e, também, que a ViewModel recebe as modificações feitas na View. A ViewModel se conecta diretamente com a Model, tendo acesso a todas as propriedades e funcionalidades disponibilizadas pelo modelo MICROSOFT (2017).

A *Model* é a classe que contém a definição do modelo dos dados, propriedades e métodos. É comum que essa classe faça a validação dos dados de entrada vindos da View. São classes não visuais que encapsulam os dados do aplicativo. Portanto, o modelo pode representar o modelo de domínio do aplicativo, que, em geral, inclui um modelo de dados junto com a lógica de negócios e validação. Classes de modelo, normalmente, são usadas em conjunto com os serviços ou repositórios que encapsulam o acesso a dados e armazenamento em *cache* (MICROSOFT, 2017).

O *Binding* é o mecanismo que uma *View* usa para se comunicar com a ViewModel. É por meio dessa associação que os dados da View refletem as alterações nas propriedades da *ViewModel*, e a View é notificada quanto às alterações das propriedades da ViewModel. Os *Bindings* podem ser implementados no código ou no XAML, mas são muito mais comuns no XAML, em que ajudam a reduzir o tamanho do código por trás do arquivo. Ao substituir o código procedural em manipuladores de eventos, por código ou marcação declarativa, o aplicativo é simplificado e esclarecido (MICROSOFT, 2017).

Durante o estágio, o Xamarin e o Xamarin.Forms foram as ferramentas utilizadas para a criação interface de usuário. Como já descrito, ele facilita o processo de desenvolvimento de aplicativos móveis multiplataforma, pois seu desenvolvimento é com o uso de uma linguagem de programação única, o C#.Net. Além disso, o Xamarin.Forms oferece a possibilidade da criação de uma interface de usuário comum, tanto para dispositivos iOS quanto para dispositivos Android.

Os pontos fortes do Xamarin.Forms que se destacam são: sua reutilização de código, que torna mais simples o trabalho, sua vasta documentação na internet, o próprio site do Xamarin que possui uma ótima documentação e sua elegância para criar interfaces mais bonitas e amigáveis aos usuários.

Um ponto fraco do Xamarin.Forms é sua falta de emulador para iOS, o que faz com que seja necessário um dispositivo da Apple com sistema operacional Mac OS [24] para compilar e testar sua versão para o sistema operacional iOS da aplicação.

3.5 Ferramenta de desenvolvimento

O *Visual Studio 2017 Community* [25] foi o ambiente de desenvolvimento integrado utilizado para criar a aplicação. Esta versão é gratuita para estudantes e traz uma integração nativa com a plataforma Xamarin e com a linguagem C#.Net.

Além de fornecer uma ferramenta para a escrita de código, o Visual Studio também oferece integração com o TFS [26] (*Team Foundation Service*), que é uma plataforma de serviços *on-line* da Microsoft que propicia serviços de Versionamento de código, testes automatizados, integração contínua e um módulo para gerenciamento de projetos. Dentre todas essas possibilidades, foram usadas as funções de versionamento de código e gerenciador do projeto MICROSOFT (2018).

No decorrer do estágio, o *Visual Studio 2017 Community* foi largamente utilizado por ser um ambiente de desenvolvimento gratuito e por ser uma versão para estudantes. O Visual Studio também traz uma integração com o Xamarin e com a linguagem C#.Net, o que ajuda no processo de desenvolvimento e depuração do código.

O *Visual Studio*, também, possui um emulador do sistema operacional Android integrado a seu ambiente de desenvolvimento, o que facilita na depuração e teste do aplicativo *mobile*.

3.6 Banco de dados SQL Server

O SQL Server fornece um software de análise e gerenciamento de dados de última geração que podem ajudar as organizações a implantar e gerenciar aplicações de missão-crítica que oferecem acesso a dados seguros, controlados e confiáveis 24 horas por dia, sete dias por semana.

---

[24] https://www.apple.com/br/macos/catalina/
[25] https://visualstudio.microsoft.com/pt-br/downloads/?rr=https%3A%2F%2Fwww.google.com%2F
[26] https://docs.microsoft.com/pt-br/visualstudio/releasenotes/tfs2017-relnotes

a) Fornecer percepções empresariais, em tempo real, a partir de fontes espalhadas por toda a organização por relatórios que podem ser customizados por usuários individuais;
b) Desenvolver novas capacidades de análises e serviços mais rapidamente e a um custo menor.

O SQL Server Management Studio foi utilizado, no processo de desenvolvimento, no período de estágio por fornecer ferramentas, a fim desconfigurar, monitorar e administrar instâncias do SQL Server e bancos de dados; ainda é usado para implantar, monitorar e atualizar os componentes da camada de dados usados pelos seus aplicativos, além de criar consultas e scripts.

4 RELAÇÃO TEORIA E PRÁTICA

Como a área de conhecimento era o desenvolvimento de software, foi necessário amplo estudo e dedicação para compreender o funcionamento das plataformas web e mobile. Durante o processo de estágio, as principais atividades na Automatiza, foram: desenvolver sistemas e aplicações; realizar manutenção de sistemas e aplicações; projetar e modelar bancos de dados; implantar aplicações em servidores Web e aplicações Mobile; implementar critérios ergonômicos de navegação; e selecionar recursos de trabalho, tais como metodologias de desenvolvimento de sistemas e gerenciamento de tarefas.

O uso das tecnologias, descritas no Capítulo 2 contribuíram para a complementação técnica e profissional do aluno, permitindo-lhe vivenciar uma rotina encontrada por um profissional na área de desenvolvimento de software.

Durante o curso de bacharelado em Sistemas de Informação da Universidade Federal de Lavras (UFLA), as disciplinas cursadas forneceram bases teóricas, fundamentais para o desenvolvimento das atividades do estágio.

As disciplinas de Algoritmos e Estrutura de Dados I, II e III foram responsáveis pelo primeiro contato com uma linguagem de programação, nas quais foram estudadas as estruturas básicas de uma aplicação computacional.

As disciplinas de Banco de dados I e II foram importantes, pois permitiram o uso comercial em uma aplicação, visto que foram trabalhados os conceitos e técnicas relacionados à persistência de dados. Além disso, foi trabalhada a parte de modelagem de banco de dados e também a complexidade existente em um acesso a dados em um banco de dados relacional.

A disciplina de Sistemas Distribuídos foi responsável pelo desenvolvimento prático, acompanhado pela disciplina a respeito de programação distribuída, principalmente, nos sistemas distribuídos baseados na Web, mostrando boas práticas de programação aliada ao modelo arquitetural (REST).

A disciplina de Engenharia de software I foi importante para o conhecimento dos diversos padrões de projeto existentes no mercado de TI.

De maneira geral, as disciplinas cursadas, no curso de bacharelado em Sistemas de Informação, permitiram ao aluno compreender seus fundamentos, juntamente com suas aplicações e relações com a área de trabalho de um futuro bacharel em Sistemas de Informação.

A disciplina de Programação Orientada a Objetos foi utilizada durante todo o período de estágio. As atividades desenvolvidas no estágio foram importantes, pois proporcionaram ao

aluno maior facilidade, ao cursar a disciplina, pois o estágio proporcionou um conhecimento teórico e prático.

5 OPINIO: UM APLICATIVO MÓVEL PARA A COMUNICAÇÃO ENTRE PROFISSIONAIS DA SAÚDE

Como primeira parte do processo de desenvolvimento, foi realizada uma reunião com o cliente, para que fossem levantados todos os requisitos e funcionalidades que ele julgasse necessários a fim de satisfazer as suas necessidades perante o aplicativo.

De acordo com o cliente, a solução implementada deveria ser inovadora e facilitar a comunicação entre os profissionais médicos e da saúde. O aplicativo tem como objetivo ajudar na discussão de casos clínicos e esclarecimento síncrono de dúvidas entre profissionais especializados e não especializados, para a promoção de uma segunda opinião e, assim, aumentar a capacidade de solucionar e resolver os quadros clínicos, agilizar os procedimentos, melhorar o atendimento e minimizar os riscos.

Para o desenvolvimento das funcionalidades, estas foram divididas em estórias de usuário, que se constituíram pela descrição das necessidades do cliente a serem implementadas. Sendo assim, para o processo de desenvolvimento, foi utilizado o princípio de metodologia ágil *scrum*[27], que é um *framework* utilizado, para organizar e gerenciar trabalhos complexos.

Como segunda parte do processo de desenvolvimento do sistema, criaram-se protótipos a fim de se verificar como seriam as telas do aplicativo. Nessa etapa, também, foram definidas quais tecnologias seriam utilizadas. Como a empresa já possuía uma parceria com a *Microsoft*, optou-se por fazer uso do *Xamarin* como ferramenta de desenvolvimento das interfaces, e o C#.Net como ferramenta de integração e implementação de APIs, que são suas tecnologias. Tanto o *Xamarin* quanto C#.Net são integrados ao Visual Studio, o que ajuda o desenvolvedor na implementação dos seus códigos.

No início do projeto, a equipe se reuniu, para discutir a modelagem do banco de dados, por meio de um diagrama Entidade-Relacionamento (ER). Foi nessa etapa que se criou um modelo conceitual, estimando todos os atributos e ações das funcionalidades do *software.*

Com o diagrama ER modelado e todos os atributos escritos, a equipe de desenvolvimento iniciou a implementação dos códigos e a estruturação das primeiras classes.

O *Entity Framework* possui um recurso chamado migrações, que permite o mapeamento do modelo de dados, produzindo um esquema de banco de dados sem precisar escrever todo o código SQL diretamente no gerenciados do banco de dados.

---

[27] Metodologia Ágil Scrum – http://www.scrum.org.

Após a geração das primeiras migrações, foi possível implementar a primeira *Api* do sistema, que era a funcionalidade de autenticação do usuário, respeitando todos os padrões de segurança. Nela, uma funcionalidade foi desenvolvida, para que fosse possível consultar no banco de dados se o usuário possuía as credenciais válidas para a utilização do aplicativo.

Depois implementada a funcionalidade de autenticação, criou-se a primeira tela do sistema: uma tela de *login*, na qual o usuário digitaria seu nome de usuário e sua senha. Os tipos de usuário que podem fazer acesso ao aplicativo são: solicitante e médico especialista. A Figura 1 apresenta um protótipo da tela de autenticação.

Figura 1 - Protótipo da tela de login.

Fonte: Automatiza

Para a construção das interfaces gráficas, foi utilizada a plataforma Xamarin como forma de diminuir a necessidade de se desenvolver códigos separados para cada sistema operacional dos aplicativos móveis, sendo possível sua criação apenas utilizando códigos da plataforma C#.Net. Os aplicativos usando Xamarin podem ter acesso total às APIs construídas oferecendo estabilidade e ótimo desempenho.

Os componentes das interfaces gráficas do Xamarin podem ser agrupados em *Pages, Layouts, Views* e *Cells.* As *Pages* podem ser de tipos diferentes, como *NavigationPage,*

MasterDetailPage, ContentPage entre outras. Cada uma das *Pages* mencionadas possui comportamentos diferentes (XAMARIN, 2016).

Views são os componentes visuais da tela, na qual se têm os campos de textos, os botões e outros componentes de interface de usuário. As *Cells* são componentes não visuais da tela, porém são usadas para modelar a apresentação visual de outros itens, geralmente, separando os componentes da *View* em forma de grade. Os *Layouts* são componentes utilizados para posicionar os elementos dentro da página, conforme a necessidade de quem implementa o código.

Todos os componentes de interface de usuário do Xamarin foram escritos seguindo o padrão de projeto MVVM (*Movel View ViewModel)*. Para se obter uma ligação entre a *View* e o restante da aplicação, é necessário o uso de uma propriedade chamada Binding. O Binding é responsável pela ligação entre a View e a ViewModel. A ideia desse padrão é permitir uma separação de responsabilidades entre as camadas da interface pelo mecanismo de *Binding* com o objetivo de se obter modularidade, flexibilidade e manutenabilidade ao implementar e dar manutenção aos códigos (DEVMEDIA, 2014).

No padrão MVVM, têm-se os componentes de *View* que representam a interface de usuário em si, sem executar nenhuma ação no sistema. Ela apresenta os dados e recebe comandos do usuário.

A camada *ViewModel* é responsável por receber e repassar os dados para a *View, por meio* dos comandos recebidos da *View*. É na *ViewModel* também que fica boa parte da regra de negócio da interface de usuário.

Por último, tem-se a camada de *Model* que tem como responsabilidade mapear o modelo de dados, propriedades e métodos. Pode-se utilizar algumas regras de negócio na *Model* como validações dos dados recebidos da *View*, por exemplo.

A forma de consultar os dados do banco de dados, para a exibição nas *Views* e, também, a forma como são guardados os dados que se recebem pela interação entre o usuário e o aplicativo foram feitas pelas APIs criadas em C#.Net.

Para fins de testes e para garantir que os dados, que foram resgatados do banco de dados são realmente os que eram esperados, a equipe utilizou a ferramenta do *Microsoft Management Studio* do SQL Server. Este auxilia na consulta ao banco de dados pela linguagem SQL.

O *Microsoft Management Studio* foi recomendado por ser uma tecnologia da *Microsoft* e por ter uma interface simples e direta ao consultar os dados. Todas as formas de escrita e recuperação de dados por parte das APIs são construídas no *back-end* da aplicação. No *back-*

end foram implementadas todas as regras de negócio do aplicativo. Todo código no *back-end* foi escrito em C#.Net e utilizando o Entity Framework, para garantir a persistência dos dados e para abstrair a necessidade de se criar manualmente os códigos em linguagem SQL.

Após a fase de prototipação e implementação dos códigos e a sua aprovação pelo cliente, a equipe produziu de forma definitiva o layout da tela de *login*. A Figura 2 apresenta a tela de *login* presente no aplicativo.

Figura 2 - Tela de login do aplicativo**.**

Fonte: Automatiza

Após a conclusão da tela de autenticação do usuário, a equipe de desenvolvimento continuou à implementação de outras telas do aplicativo. A implementação da tela de autenticação foi o maior desafio da equipe, pois a curva de aprendizado foi mais acentuada. Nenhum membro da equipe tinha experiência com a nova ferramenta de desenvolvimento, nem com o desenvolvimento de aplicações *mobile*. Com o auxílio das documentações da plataforma Xamarin, a equipe conseguiu compreender corretamente suas funcionalidades e fazer a implementação de forma correta.

A tela inicial (ver Figura 3) é a tela a partir da qual o usuário poderá escolher entre as opções de Consultas e Laudos.

Figura 3 – Tela inicial do sistema.

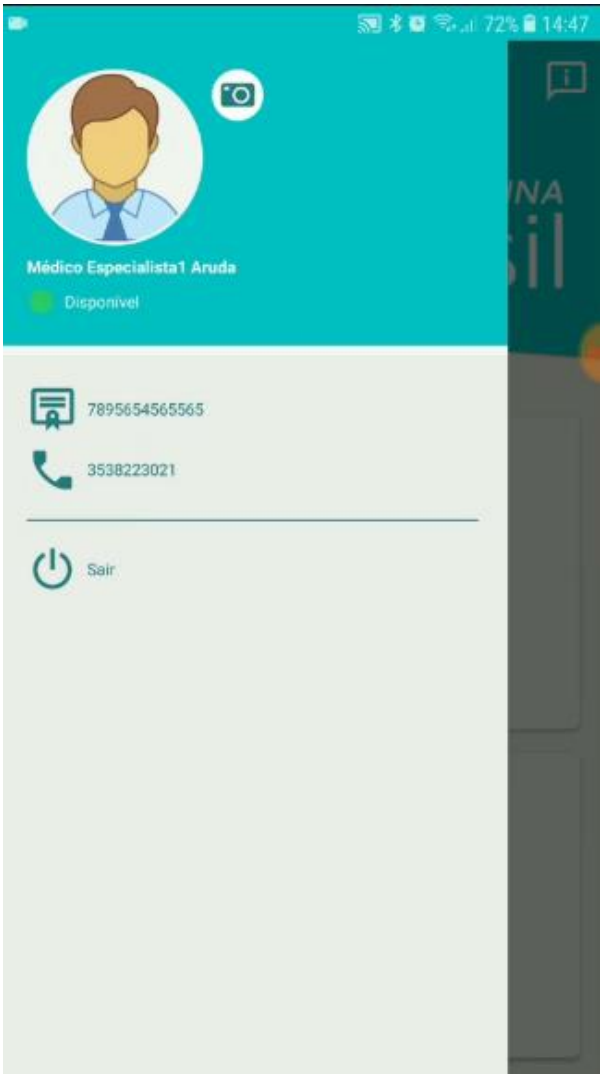
Fonte: Automatiza

O tipo de *Page* utilizado, para a construção da *View* da tela inicial, foi o *MasterDetailPage*, por ter maior flexibilidade ao exibir o menu do usuário.

Fazendo a ação de puxar da esquerda para a direita, ou clicando no ícone de menu (três pontos superior esquerdo), será exibida a tela de Perfil do usuário (ver Figura 4) com suas informações pessoais, como seu nome, o número do CRM (número que os médicos adquirem, após realizar a inscrição no Conselho Regional de Medicina) e sua imagem de perfil.

Caso o usuário ainda não tenha uma imagem no perfil, ele poderá abrir a câmera do seu dispositivo e tirar uma foto com um clique, no ícone da câmera, que está ao lado da imagem ilustrativa.

Figura 4 – Tela de perfil do usuário.



Fonte: Automatiza

Na tela inicial do aplicativo, ao clicar no botão de consultas, será mostrada a lista com todas as consultas do usuário logado. Essa tela também guarda regras de negócio para a separação de perfis de usuário. Caso o usuário seja um médico solicitante, ou seja, um médico que solicita uma segunda opinião a outro médico ou que queira apenas tirar alguma dúvida, haverá um botão que abrirá outra tela com os campos a serem preenchidos, para que se possa criar uma nova consulta. O desenho do botão de adicionar uma nova consulta é representado por um ícone, semelhante à tecla de “soma” de uma calculadora, como mostra a Figura 5.

Figura 5– Tela inicial com o ícone de adicionar consulta.

Fonte: Automatiza

A Figura 6 apresenta a tela de nova consulta, onde o médico logado com o perfil de solicitante vai solicitar uma nova consulta. Nesta tela, têm-se também os campos de texto, onde serão armazenadas as informações fornecidas pelo usuário logado, que deverá preencher detalhadamente os dados necessários para o registro da consulta.

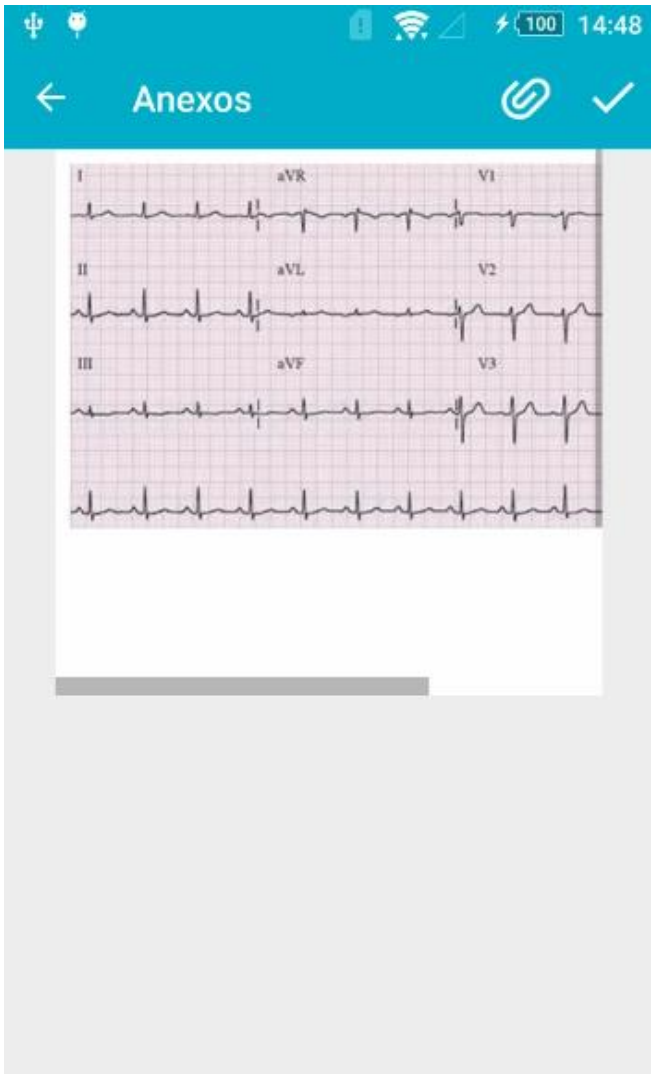
Figura 6 – Tela de nova consulta.

Fonte: Automatiza

Preenchido o formulário de nova consulta, o médico solicitante pode optar por inserir algum anexo, para detalhar melhor o problema do paciente e facilitar o diagnóstico do médico especialista. O tipo de anexo pode ser uma imagem ou qualquer outro tipo de arquivo. Para isso, basta clicar no ícone de “clipe” que está no canto superior à direita (ver Figura 7).

Após preencher corretamente o formulário e inserir os anexos, o usuário deverá finalizar a consulta, clicando no ícone que está ao lado do ícone de anexo. Feito isso, a consulta entrará na listagem de consultas pendentes, ou seja, consultas ainda sem atendimento por parte dos médicos especialistas.

Figura 7 – Tela de adicionar anexos.

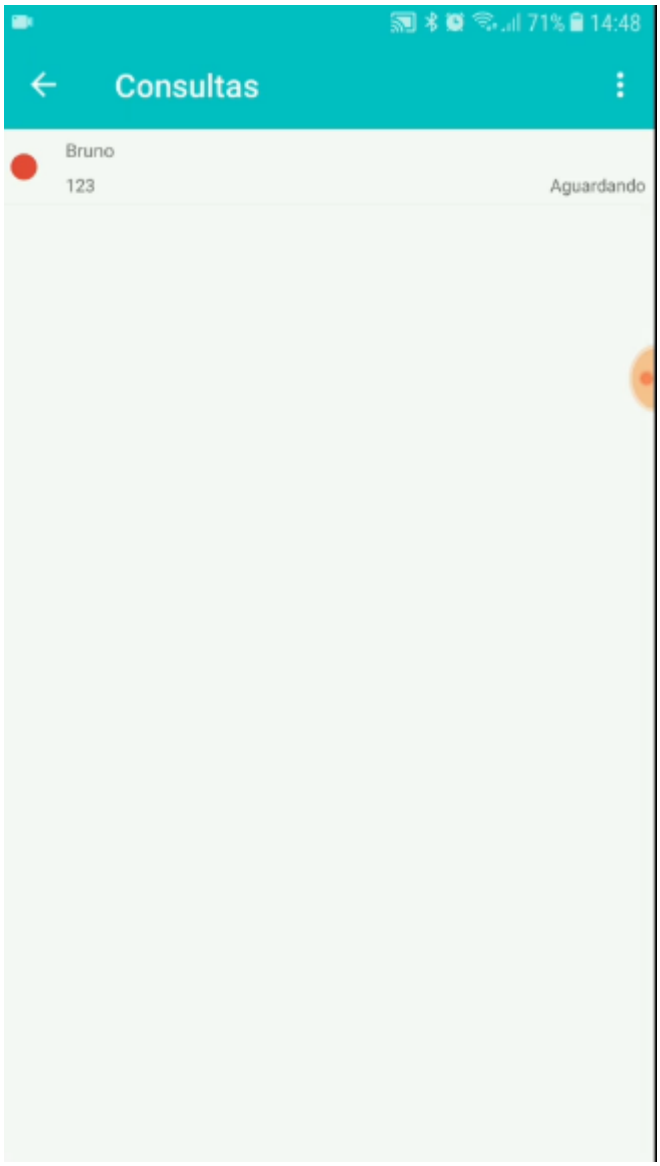


Fonte: Automatiza

Ao final de cada cadastro de novas consultas, o dispositivo móvel do médico, com perfil de usuário especialista, receberá uma notificação informando que uma nova consulta foi cadastrada e está sem atendimento. As consultas cadastradas ficam armazenadas no banco de dados e podem ser recuperadas e apresentadas na tela de listagem de consultas.

A tela de listagem de consultas (ver Figura 8) é uma *View* simples em que é exibida apenas uma lista com todas as consultas existentes no sistema. Cada elemento da lista de consultas contém apenas um ícone com o seu *status*, o nome do paciente, um identificar e o nome do *status*, se aguardado ou consulta já em atendimento. Se a consulta estiver com o ícone vermelho significa que ela está disponível para atendimento. Se o ícone estiver azul, significa que a consulta já está em atendimento.

Figura 8 – Tela de listagem de consultas.



Fonte: Automatiza

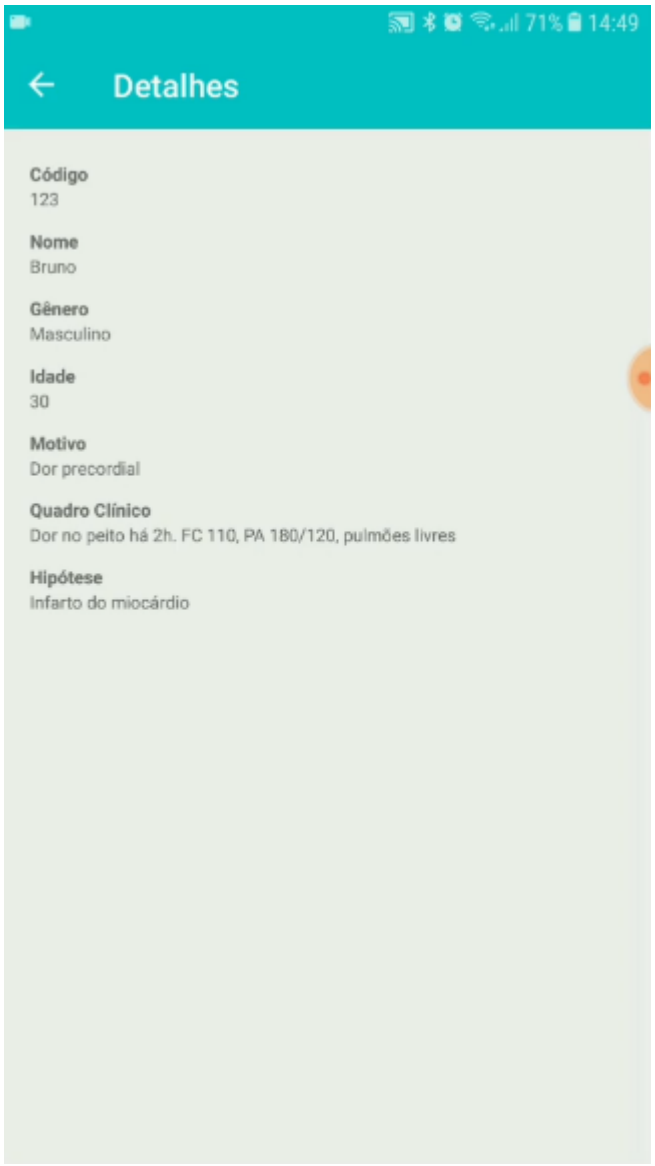
Após atender a consulta, o usuário médico especialista é direcionado a uma *View* com as informações da consulta atendida (ver Figura 9). As informações contidas na *View* são um resumo dos dados cadastrados pelo médico solicitante. Essas informações são: o nome do paciente, o sexo e a sua idade, o motivo que o levou a pedir uma consulta com alguns detalhes e um possível diagnóstico por parte do médico solicitante.

Figura 9 – Tela de informações da consulta.

Fonte: Automatiza

Na tela de informações, há um botão com a descrição “Detalhes”. Ao clicar nesse botão, o médico especialista é redirecionado a uma *View* contendo os detalhes da consulta, como mostra a Figura 10.

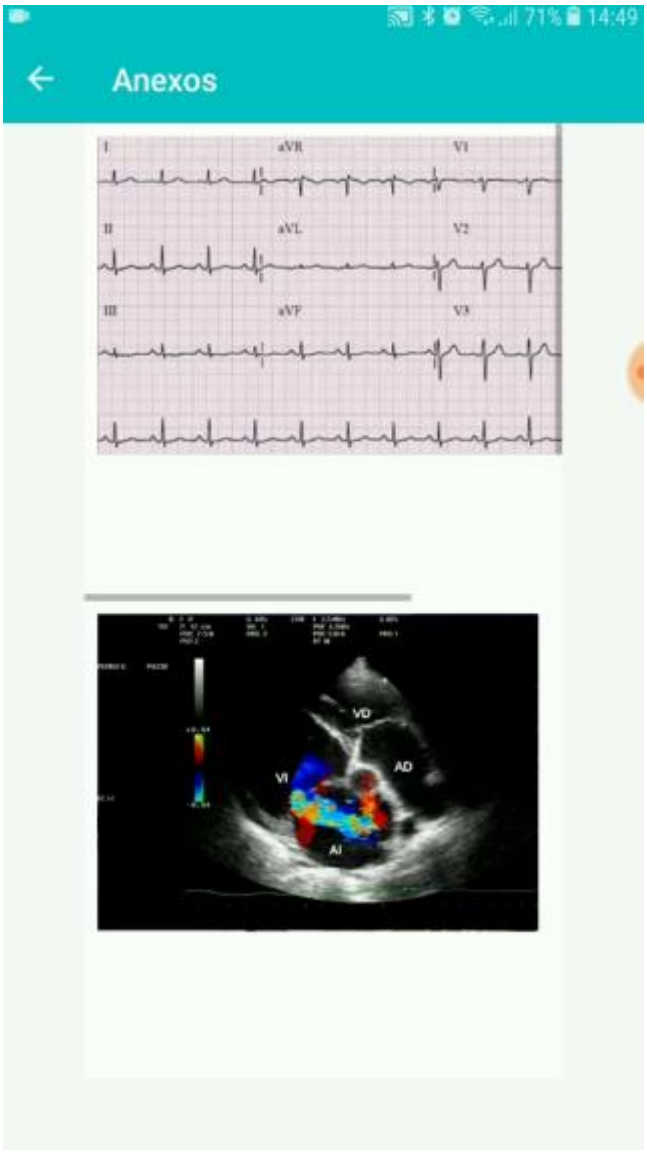
Figura 10 – Tela de detalhes da consulta.



Fonte: Automatiza

Ainda na tela de informações da consulta, há um botão com a descrição “Anexos”, em que são mostradas todas as imagens e arquivos anexados pelo médico solicitante. Para poder armazenar ou ampliar alguma imagem disponibilizada nesta tela, basta que o usuário apenas dê um clique sobre ela. A Figura 11 ilustra a tela com os anexos da consulta.

Figura 11 – Tela de anexos da consulta.



Fonte: Automatiza

Para a troca de mensagens entre os dois médicos, existe um botão com a descrição "*Chat*" na tela de informações da consulta. O principal objetivo do *chat* é abrir um canal de comunicação síncrono entre os dois médicos, para que, por meio da conversa, o médico especialista possa solucionar dúvidas e ajudar no diagnóstico do paciente. O médico solicitante também pode detalhar melhor as informações inseridas na criação da consulta.

Na tela de troca de mensagens (ver Figura 12), os usuários terão à sua disposição um botão, em forma de clipe. Após clicar no botão, abrirá uma janela onde os usuários poderão selecionar algum tipo de anexo, como documentos, arquivos ou imagens.

As mensagens trocadas entre os dois médicos vão gerar notificações nos dispositivos móveis de quem as recebe, caso o aplicativo não esteja aberto, com a finalidade de informar que existem mensagens a serem lidas.

Figura 12 – Tela de troca de mensagens.

Fonte: Automatiza

Caso julgar que sua dúvida fora esclarecida, o médico especialista deve finalizar o atendimento da consulta. Sendo assim, o médico solicitante deve clicar no ícone em formato de “prancheta”, que está presente no canto superior à direita (ver Figura 12). Após isso um alerta de chamado finalizado irá aparecer e ambos os usuários serão redirecionados à tela inicial do aplicativo (ver Figura 13).

Figura 13 – Tela de inicial com o alerta de chamado finalizado.

Fonte: Automatiza

6 DESCRIÇÃO DAS ATIVIDADES

Nesta seção, serão definidas as atividades executadas durante o projeto desenvolvido na Automatiza e suas respectivas durações.

Buscou-se, portanto:

a) Analisar o problema a ser desenvolvido com o objetivo de certificar que a proposta do projeto de software a ser desenvolvida podia ser sistematizada em uma aplicação de software;
b) Identificar requisitos técnicos para o projeto e relatar as necessidades utilizadas para a construção da aplicação. Foram identificados os pontos críticos, que demandaram treinamento e estudo mais aprofundado no desenvolvimento da aplicação;
c) Modelagem de banco de dados para projetar a estrutura de acesso e manipulação de dados. Foi utilizado o SGBD SQL Server. Foi importante projetar essa etapa visando ter o mínimo de mudanças possível durante o andamento do projeto;
d) Realizar treinamento sobre a plataforma Xamarin para o desenvolvimento do aplicativo. O aperfeiçoamento técnico em tecnologia foi realizado por meio de cursos disponibilizados na Web ou pela programação em par (*pair program*), que consiste no acompanhamento das atividades ao lado de algum membro da equipe, que já possua experiência em tecnologia. Para concluir as etapas de treinamento e de analises do problema a ser resolvido, a equipe precisou de cerca de quinze dias;
e) Aplicar de forma prática os conceitos teóricos aprendidos no treinamento;
f) Implementar o código fonte aplicando os conceitos aprendidos no treinamento;
g) Propor um MVP ao cliente e levantar novos requisitos.

O desenvolvimento da aplicação seguiu a metodologia ágil do *Scrum*, que foi vivenciada na disciplina de Gerência de Projetos. O *Scrum* prevê que o projeto a ser desenvolvido é dividido em pequenas partes entregáveis a cada interação com o cliente. Neste projeto, cada interação possui o tempo de quinze dias corridos, denominada Sprint. Foram realizadas reuniões, para o planejamento, a cada período de quinze dias, denominadas *Sprint Planing*.

Durante o período de desenvolvimento, foi feita reunião diária com todos os membros da equipe. Cada membro relatava três conteúdos: o que estava desenvolvendo, o que iria ser desenvolvido e os problemas encontrados, denominada de Daily Meeting. No final de cada

ciclo de desenvolvimento, foram realizadas duas reuniões; uma para apresentar as funcionalidades finalizadas e outra para identificar os problemas encontrados, durante o ciclo de desenvolvimento da Sprint, as quais foram denominadas de *Sprint Review* e *Sprint Retrospective,* respectivamente.

As atividades do período de estágio duraram cerca de um ano e o desenvolvimento do aplicativo durou cerca de 6 meses. Além do desenvolvimento do aplicativo o estagiário também colaborou, como parte das exigências da empresa Automatiza, na implementação e manutenção de outros projetos, como o desenvolvimento do sistema ProlexNet.

No sistema ProlexNet, o estagiário atuou, conforme as exigências da equipe, para o desenvolvimento e correções pontuais de erros no sistema.

7 CONSIDERAÇÕES FINAIS

A obtenção de experiência técnica, na área de Tecnologia da Informação, é um grande benefício para os primeiros anos de carreira de um profissional que deseja atuar no ramo de desenvolvimento de software.

O conhecimento prático das tecnologias utilizadas, durante o estágio, contribuiu significativamente para a experiência e conhecimento atual do aluno. A aplicação prática da teoria aprendida, durante a graduação, foi bem-sucedida, pois o aluno teve um embasamento teórico e prático, oferecido pelas disciplinas cursadas durante o curso de Bacharelado em Sistemas de Informação da Universidade Federal de Lavras (UFLA).

A atividade em equipe teve um papel interessante para o estagiário, pois possibilitou constantes trocas de conhecimento entre os membros da equipe, além de oferecer ao aluno a vivência de desenvolver *softwares* de maneira colaborativa.

A participação, em uma equipe de desenvolvimento de software, contribuiu para que o aluno concluísse o curso de Bacharelado em Sistemas de Informação, com uma experiência com uma equipe de desenvolvimento. Foi um período em que foram aprendidas diversas lições inerentes à carreira de um profissional formado em Sistemas de Informação.

Conclui-se que o estágio realizado, na Automatiza, no papel de desenvolvedor de software, foi um importante complemento, para que o aluno concluísse o curso de Sistemas de Informação da Universidade Federal de Lavras (UFLA).

REFERÊNCIAS

BECK, K. et al. (2001) “Manifesto for Agile Software Development”, Setembro de 2016.

DEVMEDIA. Entendendo o Pattern Model View ViewModel MVVM, 2014. Disponível em<http://www.devmedia.com.br/entendendo-o-pattern-model-view-viewmodel-mvvm/18411>. Acessado em 21 nov. 2016

HEVNER, A., March, S., Park, J., and Ram, S. Design Science in Information Systems Research, 2004. MIS Quarterly, p. 75-105.

MICROSOFT. C# Guide, 29/01/2018. Disponível em < https://docs.microsoft.com/pt-br/dotnet/csharp/>. Acessado dia 01 de junho de 2019.

MICROSOFT. Entity Framework overview, 07/08/2017. Disponível em < https://docs.microsoft.com/pt-br/dotnet/framework/data/adonet/ef/overview>. Acessado dia 01 de junho de 2019.

MICROSOFT. Introduction to Model/View/ViewModel pattern for building WPF apps, October 8, 2005. Disponível em <https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/>. Acessado dia 01 de junho de 2018.

MICROSOFT. Team Foundation Server.Conectar a projetos de equipe no Team Foundation Server. Disponível em <https://msdn.microsoft.com/pt-br/library/ms181475.aspx>. Acessado dia 01 de junho de 2018.

MICROSOFT. The Model-View-ViewModel Pattern, 06/08/2017. Disponível em < https://docs.microsoft.com/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>. Acessado dia 01 de junho de 2018.

MICROSOFT. Xamarin.Forms Data Binding, 01/05/2018 Disponivel em <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/app-fundamentals/data-binding/>. Acessado dia 01 de junho de 2018.

OMAR. Desenvolvimento de Aplicações Web utilizando o MVC Design Pattern. 2004. Disponível em: < https://projetos.inf.ufsc.br/arquivos_projetos/projeto_239/Artigo%20I%20-%20Desenvolvimento%20de%20Aplica%E7%F5es%20Web%20utilizando%20o%20MVC%20Design%20Pattern.pdf >. Acesso em: 06/07/19.

PROCEDI, L (2016), Avaliação do framework Xamarin.Forms para desenvolvimento de aplicativos móveis multiplataforma, criando uma aplicação real, Novembro de 2016.

RADI, A. A. Evaluation of Xamarin Forms for Multiplatform Mobile Application Development, Technical Library Paper, 2016.

SABBAGH, R. (2013), Scrum – Gestão ágil para Projetos de Sucesso, Editora Casa do Código.

SOARES, L. C. da R. CakePHP – um framework Web MVC:. p. 14, 2015. Disponível em:<https://fit.faccat.br/~leonardoseibt/ArtigoJavaScript.pdf>.

WEN, C. L. Telemedicina e Telessaúde – Um panorama no Brasil. Informática Pública, v. 10, n. 2, p. 7-15, 2008.

XAMARIN DEVELOPER. Xamarin.Forms Controls Reference, 2016. Disponível em <https://developer.xamarin.com/guides/xamarin-forms/controls/>. Acesso em 21 nov. 2016.