UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
Curso de Graduação em Sistemas de Informação
Darlan Souza Silva

OS BASTIDORES DO DESENVOLVIMENTO WEB E SUA IMPORTÂNCIA NA SOCIEDADE TECNOLÓGICA

Diamantina
2020

Darlan Souza Silva

OS BASTIDORES DO DESENVOLVIMENTO WEB E SUA IMPORTÂNCIA NA SOCIEDADE TECNOLÓGICA

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Departamento de Computação, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Luciana Pereira de Assis

Diamantina
2020

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Darlan Souza Silva

OS BASTIDORES DO DESENVOLVIMENTO WEB E SUA IMPORTÂNCIA NA SOCIEDADE TECNOLÓGICA

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientadora: Luciana Pereira de Assis

Data de aprovação: 04/12/2020

Profa. Dra. Luciana Pereira de Assis
Faculdade de Ciências Exatas - UFVJM

Prof. Dr. Alessandro Vivas Andrade
Faculdade de Ciências Exatas - UFVJM

Prof. MSc. Luis Henrique Silva Rodrigues

Faculdade de Ciências Exatas - UFVJM

Documento assinado eletronicamente por **Luciana Pereira de Assis**, **Servidor**, em 14/12/2020, às 16:48, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor**, em 14/12/2020, às 17:04, conforme horário oficial de Brasília, com fundamento no art. 6°, § 1°, do Decreto n° 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **0231632** e o código CRC **C3C04A2D**.

Dedico este trabalho a Deus, que me proporciona vitórias e conquistas a cada dia, a minha família, que sempre esteve ao meu lado e a todos que contribuíram para a sua realização.

AGRADECIMENTOS

Agradeço primeiramente a Deus por guiar meu caminho e iluminar minha vida, me dando discernimento e sabedoria para que este trabalho fosse concluído, sem ele nada é possível. A meus pais, Maria e Valter e meu irmão Edilon que sempre estiveram presentes em minha vida, me apoiando, fortalecendo e motivando com muito amor e carinho, fazendo o possível e o impossível para que eu me tornasse a pessoa que sou hoje. Construí meu caráter em cima de seus ensinamentos, muito obrigado por se sacrificarem tanto por mim e acreditarem que eu sou capaz fazer qualquer coisa. Agradeço minha noiva Gabriela, por todo carinho e compreensão durante o período em que estive longe e não pude estar ao seu lado, sendo um dos pilares para a conclusão deste trabalho e sendo minha força em todos os momentos difíceis. Ao meu tio José, por todo apoio e carinho durante a minha infância, sendo uma fonte de inspiração. A todos os docentes que contribuíram para minha formação, em especial a minha orientadora e conselheira Prof.ª Luciana Pereira de Assis, por ter me acompanhado no desenvolvimento deste trabalho e ter me transmitido vários conhecimentos, com sabedoria e paciência. Agradeço muito a cada pessoa com quem pude conviver em Diamantina e que tornaram minha estadia mais agradável e divertida. E a todos que de forma direta ou indireta contribuíram para a conclusão deste trabalho e a realização deste sonho, meu muito obrigado.

RESUMO

O presente trabalho tem por objetivo tornar mais clara a visão sobre o processo de desenvolvimento de softwares. Este processo segue padrões comuns, mas também possui características distintas em contextos variados. Muitas das vezes, estudantes dos cursos de computação e afins, se veem perdidos entre tantas rajadas de conteúdos diferentes sobre a construção de uma aplicação Web, essas informações podem ser encontradas em cursos, fóruns, livros e vários outros tipos de materiais presentes na Internet. Conteúdos dispersos acabam tirando o interesse dos iniciantes em desenvolvimento Web, em se aprofundar neste maravilhoso mundo da programação, pois muitos dados acabam tornando o aprendizado confuso. A fim de amenizar estes impactos de aprendizado, o trabalho que será apresentado a seguir abrange de forma geral como ocorre um processo de desenvolvimento de software. Para exemplificar este processo, o módulo de uma aplicação será desenvolvido com práticas similares as adotadas em uma empresa que trabalha com o desenvolvimento de softwares. Porém, antes de tratar do desenvolvimento propriamente dito, alguns outros assuntos serão abordados no trabalho, como: O surgimento do desenvolvimento Web; As metodologias utilizadas no desenvolvimento de softwares; Como metodologias se aplicam ao projeto que será criado. Após os conceitos necessários para que o desenvolvimento Web se torne mais claro, a parte prática será apresentada com o objetivo de tornar todo o contexto do trabalho mais próximo para o leitor.

Palavras-chave: Aplicação Web. Desenvolvimento de Software. Programação. Metodologias. Aprendizado.

ABSTRACT

This study aims to clarify the scenario about the software development process, which follows common patterns, but it also has different characteristics in different contexts. Students of computer courses and correlated areas often find themselves lost among so many bursts of different content about building Web applications. All this information can be found in courses, forums, books and various other types of materials on the internet. Scattered content can negatively affect the interest of beginners in Web development, and from delving into this wonderful world of programming. A lot of information can make learning confusing. In order to mitigate these learning impacts, this work covers in general how a software development process occurs. To exemplify this process, an application module will be developed with practices similar to those adopted in a company that works with software development. However, before dealing with development itself, some other issues will be addressed, such as the emergence of Web development, the methodologies used in software development, and how the methodologies apply to the project that will be created. After the concepts necessary for Web development become clearer, the practical part will be presented aiming to make the entire context of software development closer to the reader.

Keywords: Web Applications. Software Development. Programming. Methodologies. Learning.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Acompanhar as mudanças e inovações no campo do desenvolvimento Web tem se tornado uma tarefa difícil. As mudanças são constantes desde a sua criação até os dias de hoje. Novas plataformas de desenvolvimento, linguagens de programação, metodologias de desenvolvimento e bancos de dados surgem para atender às novas demandas e tendências que o mercado exige.

O trabalho a seguir tem como interesse direcionar o contato de pessoas com o desenvolvimento Web. Tirar um pouco da obscuridade envolta por essa forma de desenvolvimento tornará o caminho de futuros desenvolvedores de software mais claro quando se trata do processo por trás desta construção. Instigar o interesse de pessoas pela programação, irá gerar inúmeras consequências positivas em um mundo cada vez mais globalizado e tecnológico.

Com a criação da Internet, a sociedade vem inovando cada dia mais seus conhecimentos tecnológicos. Tendo a Web surgindo a partir da Internet, deu-se início a uma grande transformação na vida de todos nós. Segundo Murugesan e Ginigi (2005), o crescimento rápido das aplicações Web tem afetado todos os aspectos de nossas vidas.

No decorrer dos anos diversas habilidades surgiram para auxiliar os desenvolvedores a criarem soluções tecnológicas para diversas áreas. Isso impactou todo o mundo e fez com que a vida cotidiana se torne inimaginável sem o uso dessas ferramentas, que gerenciam desde a bolsa de valores até o caixa de uma pequena padaria.

Com o emaranhado de informações sobre o desenvolvimento Web, torna-se um pouco confuso estabelecer uma relação entre todas as dependências para seu funcionamento. O que há de novo no processo de desenvolvimento Web? Este trabalho busca responder está pergunta, bem como, apresentar exemplos de uso dessas novas metodologias e ferramentas.

Através do texto e do desenvolvimento do módulo de uma aplicação, o leitor adentrará nos bastidores do desenvolvimento Web e descobrirá algumas de suas especificidades, aprendendo com exemplos os conceitos com que se depara diariamente.

1.1 Objetivos

O objetivo geral deste trabalho é apresentar as diretrizes do desenvolvimento Web, para que a estrutura de uma aplicação seja conhecida, juntamente com os elementos necessários para sua criação, desmistificando a ideia de que a programação Web é algo oneroso e de difícil implementação.

Desta forma, busca-se, por meio de algumas análises das tecnologias de desenvolvimento Web, auxiliar desenvolvedores para que obtenham o melhor resultado para seu problema. Isso é feito orientando-os sobre qual melhor linguagem, metodologia de desenvolvimento, arquitetura e outras diretrizes que se adaptam às suas necessidades. Por fim, através de um exemplo, será apresentada na prática a aplicação desses conceitos e métodos vistos no decorrer do texto.

1.1.1 Objetivos específicos

Em relação aos objetivos específicos do trabalho, alguns pontos podem ser levantados:

1. Apresentar um pouco sobre a história do desenvolvimento Web e de como se deu seu crescimento contínuo no decorrer dos anos.
2. Expor alguns conceitos importantes que são necessários quando se fala em desenvolvimento de aplicações Web.
3. Apresentar metodologias envolvidas no processo de criação de um software e o que elas geram de valor durante este processo.
4. Apresentar ferramentas e tecnologias que podem ser utilizadas como apoio para construção de aplicações Web robustas e consistentes.
5. Implementar o módulo de uma aplicação que será construído com ferramentas e tecnologias recentes, tendo como apoio o uso de alguma metodologia de construção de softwares.

1.2 Estrutura do Trabalho

O presente trabalho está estruturado da seguinte forma: O capítulo dois irá abordar um pouco sobre a história da Web; O capítulo três apresenta alguns conceitos importantes para o desenvolvimento Web, juntamente da definição da aplicação da qual será desenvolvido um de seus módulos; O capítulo quatro ficará encarregado de apresentar os processos envoltos no desenvolvimento de um software; O capítulo cinco apresenta uma definição sobre os bancos de dados que são utilizadas no desenvolvimento Web, este capítulo também dará início a construção do módulo de uma aplicação, com o desenvolvimento do seu banco de dados; O capítulo seis apresenta definições sobre frameworks e da continuidade ao desenvolvimento do módulo proposto; O capítulo sete apresenta os testes automatizados relacionados ao módulo desenvolvido nos capítulos cinco e seis. Por fim, no capítulo oito serão apresentadas as conclusões.

2 UM POUCO DE HISTÓRIA

A Internet é responsável por conectar tudo e todos. Seu início se dá na década de 1950, período em que o mundo vivenciava a guerra fria que envolveu a União Soviética e os Estados Unidos (CARVALHO, 2006). Neste período qualquer forma de obter vantagem sobre o inimigo era tida como proveitosa e isso fez com que Dwight D. Eisenhower, o presidente dos Estados Unidos naquele período, criasse, em 1958, a Agência de Projetos de Pesquisa Avançada (Advanced Research Projects Agency, ou ARPA) para produção de inovações tecnológicas. Com o decorrer do tempo a **ARPA** recebeu um "D" de Defense, passando então a ser chamada de **DARPA** (Agência de Projetos de Pesquisa Avançada de Defesa) (CARVALHO, 2006).

Descentralizar informações sempre garantiu que as perder se tornasse mais difícil, essa descentralização também ocorreu na DARPA. Esse processo de troca de dados entre locais diferentes foi denominado **ARPANET** e ocorreu em 1962 (CAMARGO, 2009). Segundo Castells (2001), no ano de 1969, com a ideia vinda da ARPANET de descentralizar informações, houve a comunicação entre algumas universidades dos Estados Unidos, o que fortaleceu ainda mais a rede.

Para Neto (2014), devido as estruturas incompatíveis, as primeiras máquinas não realizavam troca de informações entre si, o que era um grande empecilho para a transmissão de dados naquela época. Por conta disso, em 1970, foi criado o protocolo **NCP** (Network Control Program, ou Programa de Controle de Rede), que possibilitou a comunicação entre estas diferentes máquinas.

Em 1971, com a criação do NCP, deu-se início a expansão dos meios de comunicação, pois começa a surgir um modelo experimental do e-mail. Logo em seguida, em 1972, o e-mail já estava disponível para envio de mensagens eletrônicas, facilitando a troca e obtenção de informação através da rede que, até então, era uma operação trabalhosa e demorada (JURNO; SILVA; SILVA, 2017).

Para Vicentini *et al.* (2005), após o surgimento do e-mail, a rede pôde ser ampliada e surgiram também conexões internacionais, que estabeleceram comunicação entre computadores na Noruega e Inglaterra. Assim sendo, percebemos o quanto a década de 70 foi marcante devido às importantes expansões da ARPANET.

2.1 O surgimento dos protocolos de comunicação

Como visto anteriormente, devido a estrutura diferente dos computadores da época com hardwares diversos que não eram compatíveis entre si, foi necessário a criação de um protocolo para troca de informações que foi batizado de NCP (NETO, 2014).

Segundo Neto (2014), com o decorrer do tempo e a crescente popularidade da ARPANET, estabelecendo conexões internacionais, muitos outros computadores foram inseridos na rede fazendo com que o protocolo NCP, utilizado até então, começasse a apresentar limitações que impactavam na troca de dados entre as máquinas. A fim de solucionar as dificuldades que o NCP enfrentava na transmissão de dados na rede, em 1973 surgiu um novo protocolo

de transmissão que iria revolucionar as comunicações pela rede, o Protocolo de Controle de Transmissão, mais conhecido com **TCP** (Transfer Control Protocol) (TANENBAUM, 2003).

O protocolo TCP era eficaz ao realizar as transferências entre as máquinas, porém deixava a desejar quando se tratava da identificação das máquinas (TANENBAUM, 2003). Albuquerque *et al.* (2003), disserta que, com a criação do protocolo **IP** (Internet Protocol, ou Protocolo de Internet) por Vinton Cerf, esse problema de localização foi resolvido e dessa forma surge o conjunto de protocolos **TCP/IP** que é comumente utilizado até os dias de hoje.

Segundo Neto (2014), após a aplicação do TCP/IP muitas máquinas poderiam fazer parte da rede, eliminando os problemas de comunicação em diferentes redes, como ocorriam com o uso do protocolo NCP. Com isso, a partir de 1983, houve a substituição do protocolo NCP pelo TCP/IP. Essa evolução nos protocolos de comunicação acabou culminando no surgimento da **World Wide Web**.

2.2 O surgimento da World Wide Web

Com o crescimento acelerado e uso constante da Internet muitas trocas de informações já eram realizadas através da rede, mas esses processos ainda eram trabalhosos, onerosos e pouco acessíveis (LINS, 2013). Para resolver esses problemas surge a World Wide Web (Rede mundial de computadores), conhecida popularmente como **Web**.

Segundo Lins (2013), dada a necessidade de melhorar e agilizar o compartilhamento de documentos entre os usuários da Internet, Tim Berners-Lee, um cientista da computação suíço, planejou como se daria uma troca de informações entre computadores ligados em uma rede. Através de estudos e análises chegou à conclusão que utilizaria o **hipertexto** para esta tarefa.

O hipertexto realiza atividades que são inviáveis para textos comuns, isso se dá devido a sua estrutura de palavras-chaves, que são popularmente conhecidas no mundo da programação como **tags**. As *tags* estão presentes nos textos de Web *sites* e sua grande vantagem é devido a capacidade de poderem ser interpretadas pelo computador (LUCCIO, 2010).

Por meio das *tags* é possível a geração de **links** que são responsáveis por criar direcionamentos para outros locais onde poderão estar presentes imagens, vídeos, textos, e demais conteúdos relacionados a um Web *site* (LINS, 2013). As áreas sensíveis que representam os *links* presentes nas páginas podem ser uma palavra, botão, texto ou até mesmo imagens. A Figura 1 apresenta um exemplo de como um *link* é exibido em uma página estática.

Figura 1 – Exemplo de *link* em um página estática

Clique aqui!

Fonte: Próprio autor.

Luccio (2010), aborda que a utilização do hipertexto por Tim Berners-Lee fez com que surgisse a linguagem **HTML** (HyperText Markup Language). O HTML é uma linguagem

de marcação de hipertexto que realiza, dentre outras coisas, operações de criação e navegação entre as páginas na Web. Este feito contribuiu para que ele fosse chamado de *"o pai da Web"*.

A utilização das palavras-chaves foi determinante para o crescimento elevado da Web, pois, por meio delas, *sites* e conteúdos poderiam ser facilmente encontrados e recuperados, tudo devido as *tags* que referenciavam assuntos do *site* (LUCCIO, 2010). No decorrer dos anos, após a expansão da Web, proposta por Berners-Lee, a rede em que as máquinas estabelecem a comunicação passou por alguns momentos de evolução que foram chamados de Web 1.0, Web 2.0 e Web 3.0 (LUCCIO, 2010).

2.2.1 Web 1.0

Vista como a primeira implementação da Web, a Web 1.0 era tida como uma Web apenas de leitura. As páginas Web das aplicações dessa versão da rede eram pobres no que diz respeito a interação com usuário e deixavam a desejar quando o assunto era design e usabilidade (NAIK; SHIVALINGAIAH, 2009). Segundo Luccio (2010), a Web estática possuía muitas informações ligadas a empresas e também a educação e não possibilitava a interação de um usuário externo na produção de conteúdo. Todos os conteúdos presentes nas páginas eram controlados pelos desenvolvedores daquele Web *site*.

O objetivo principal da Web 1.0 era disponibilizar conteúdo relacionado a diversos ramos da sociedade (NAIK; SHIVALINGAIAH, 2009). O foco dessa disponibilização era fazer com que as pessoas que tinham contato com a rede começassem a ter acesso a diversas informações que antes eram de difícil contato. Isso ampliou e atraiu ainda mais olhares para a Internet (LUCCIO, 2010).

Ainda segundo Luccio (2010), como o conhecimento sobre desenvolvimento Web era abstrato, esses *sites* e portais existentes na Web 1.0 eram desenvolvidos por pessoas que possuíam um bom domínio de programação. Como visto anteriormente, estes mesmos programadores eram responsáveis também pelos conteúdos que seriam disponibilizados nas páginas.

2.2.2 Web 2.0

A Web 1.0 era caracterizada pela complexidade quando se tratava do controle e produção de conteúdo para Web *sites*, lembrando que, apenas pessoas que possuíam um bom conhecimento de programação conseguiam gerar conteúdos para páginas Web (LUCCIO, 2010). Com o decorrer do tempo e a crescente utilização de blogs e redes sociais surgiu uma nova forma de interação na rede, chamada de Web 2.0, conhecida também como a Web colaborativa (LUCCIO, 2010).

Para Naik e Shivalingaiah (2009), a segunda geração da Web possui características distintas e marcantes em relação a sua geração passada. Com está evolução, tem início a interação dos usuários como geradores de conteúdo na Internet.

A partir do final da década de 1990 e início dos anos 2000, a Web 2.0 ampliou ainda mais a capacidade de interação dos usuários na rede, com a criação e expansão de ferramentas de colaboração e compartilhamento de informações. Estas ferramentas possibilitavam que qualquer

pessoa com acesso a um computador não fosse apenas um consumidor passivo de informações, mas que também pudesse gerar conteúdos na rede (MELLO, 2016).

2.2.3 Web 3.0

Com a crescente distribuição de *sites*, blogs e redes sociais através da Web 2.0, um emarando de *links* de informações foram surgindo na rede. Com o passar do tempo a proposta de utilizar estes *links* na interação com a rede fez com que surgisse uma nova nomenclatura de Web, a 3.0 (LUCCIO, 2010).

A Web 3.0 faz uso de uma base de dados que permite que diversos conteúdos se tornem acessíveis a outros aplicativos e possam ser usados na obtenção de dados relacionados a uma determinada busca feita por um usuário na Internet (NAIK; SHIVALINGAIAH, 2009).

No exemplo que Luccio (2010) expõe, a Web 3.0, também denominada Web semântica, pode ser usada para gerar um filtro de clínicas médicas presentes na Internet que atendam alguns critérios, como distância máxima e valor. Por meio dos parâmetros apresentados anteriormente, um agente de busca faz uma varredura em *sites* relacionados ao tema e que atendam as especificações definidas, assim que obtém resultados eles são retornados ao usuário que realizou a requisição (LUCCIO, 2010).

Desta forma, percebe-se que a Web 3.0 é baseada nos comportamentos de seus usuários na interação com a rede e age como um filtro para dispor o que mais interessa e atende ao internauta, fazendo com que a navegação se torne mais personalizada para suas necessidades.

Cada umas dessas evoluções da rede trouxeram várias inovações para o desenvolvimento Web, que cresceu muito no decorrer dos anos e vêm alcançando um patamar cada vez mais elevado dentro da Internet. Mas afinal de contas, o que é o desenvolvimento Web?

2.3 O que é o desenvolvimento Web?

Atualmente a maior parte do desenvolvimento de softwares é voltada para Web, visto que, aplicações Web geram diversas vantagens e uma das mais relevantes é a portabilidade das aplicações que faz com que elas sejam gerenciadas facilmente. Aplicações que rodam apenas em máquinas específicas não produzem bons resultados, devido a isso, o conceito de distribuição e paralelismo de sistemas está cada dia ganhando mais espaço.

A vida dos usuários da Internet e desenvolvedores de aplicações se tornou melhor graças ao surgimento do desenvolvimento Web. Com base na definição de Pressman (2006), podemos considerar que o desenvolvimento Web é o processo de criação de um software que possuirá conexão com uma rede, sendo que esse software estará acessível através da Internet ou intranet.

Segundo Pressman (2006), uma aplicação Web, ou WebApp, pode ser uma página estática com poucos elementos, ou um *site* repleto de conteúdo e munido de uma vasta interação com o usuário. A definição do desenvolvimento Web não está ligada a complexidade, mas sim a conceitos que são aplicados para sua criação.

Para que o desenvolvimento de uma aplicação Web aconteça, uma série de processos e de ferramentas são utilizados, pois esta aplicação fará uma interface entre o usuário que irá realizar uma solicitação ao servidor que é o responsável por atendê-la.

Para que o usuário alcance seus objetivos, todo o desenvolvimento Web deve estar bem estruturado e definido. Por exemplo, o **front-end**[1] deve estar alinhado com o **back-end**[2], que por sua vez deve estar bem consistente com o **banco de dados**. Assim sendo, este trabalho surge como uma alternativa para aproximar os leitores dos bastidores do desenvolvimento Web.

Para Pressman (2002), os sistemas criados para a Web possuem algumas características únicas que devem ser seguidas durante o seu desenvolvimento. São elas:

- **Imediatismo:** Um sistema Web pode estar funcional em questão de semanas ou dias, essa característica não é encontrada em outros tipos de software. Porém, para que isso aconteça, os desenvolvedores devem seguir bons métodos para o planejamento, projeto, implementação e testes da aplicação.
- **Segurança:** Toda infraestrutura de uma aplicação precisa de ações de segurança adequadas, pois as informações dos sistemas baseados na Web estão presentes na rede e, consequentemente, torna-se difícil limitar quem terá acesso a mesma. Assim sendo, deve-se implementar políticas eficientes de segurança para proteger os dados dos usuários e das aplicações, tornando as interações mais confiáveis.
- **Estética:** O aspecto de aplicações Web atraem ainda mais as atenções, é inegável que essa característica surte influência nos usuários. Em alguns casos, a estética pode estar relacionada diretamente com o sucesso do produto desenvolvido.

Sistemas Web possuem diversas vantagens, essas vantagens que tornam seu uso cada vez mais atrativo no cotidiano das pessoas. Porém, da mesma forma que as aplicações Web possuem vantagens elas também deixam a desejar em alguns pontos. Algumas das vantagens das aplicações Web, são:

1. **Atualização dinâmica de conteúdo:** Todos os usuários que têm contato com o sistema recebem as atualizações de conteúdo em tempo real.
2. **Disponibilidade de acesso:** Exceto quando estiver passando por indisponibilidades técnicas, a aplicação estará disponível através de qualquer navegador possibilitando que o acesso seja feito em qualquer lugar e a qualquer hora.
3. **Custos com hardware:** Como a aplicação estará em um servidor, se o mesmo for alugado, não será necessário investimento algum em hardware para execução da aplicação.
4. **Alcance de usuários:** A aplicação pode expandir a visibilidade de um negócio, empresa e afins, por meio da Internet, que possui uma conexão mundial.

Algumas desvantagens das aplicações Web, são:

---

[1] São os responsáveis pela parte visual das aplicações. Uma definição mais detalhada será apresentada no capítulo seis.

[2] Auxiliam na implementação das regras estruturais que constituem o sistema. Uma definição mais detalhada será apresentada no capítulo seis.

1. **Instabilidade da rede:** A velocidade da rede é determinante para velocidade do sistema, ou seja, quanto mais tempo uma requisição demora para ser atendida, mais tempo será gasto para o usuário obter uma resposta.
2. **Versões dos navegadores:** Não existe um padrão para todos os navegadores e possivelmente uma aplicação seja exibida de diferentes formas em navegadores distintos. Por exemplo, a formatação de um botão pode ser apresentada com alterações em sua composição em navegadores variados.
3. **Erros no servidor:** Caso ocorra falha no servidor onde a aplicação está hospedada, o serviço ficará indisponível para o usuário.

2.4 Vivenciando o crescimento do desenvolvimento Web

Com o passar dos anos a Web se expandiu de forma acelerada. Pode-se perceber isso olhando para o dia a dia das pessoas, quantas vezes acessam uma rede social, pesquisam por uma receita, ou uma notícia, enfim, essas ações já se tornaram rotineiras no cotidiano da população. Mesmo antes de ser tão comum, no início da Web diversas aplicações já eram acessadas por vários usuários que possuíam contato com a rede. Esses acessos não eram feitos de forma ampla, por não estarem tão disponíveis como nos dias de hoje.

As interações na Web já são necessárias para manutenção da sociedade, que se vê ligada pela rede. Muitos serviços e operações que há cerca de 20 anos atrás eram demorados e caros, agora podem ser feitos com o simples clique de um botão. Trazer tamanha comodidade para as pessoas acaba por facilitar sua rotina, pois ao invés de esperar por horas em uma fila, muitos assuntos já podem ser resolvidos pela Internet, evitando o estresse e otimizando o tempo das pessoas para outras atividades.

Junto deste crescimento acelerado e das constantes vantagens geradas para diversas pessoas, deu-se início o surgimento de várias formas de melhorar a vida de programadores que estão sempre gerando conteúdo para a Web. Da mesma forma que diversos usuários apareceram na Web, a quantidade de desenvolvedores também aumentou no mercado e consequentemente as ferramentas também evoluíram.

Este crescimento no mundo do desenvolvimento fez com que surgisse as metodologias de desenvolvimento, a arquitetura de software, os frameworks, bancos de dados e linguagens de programação que geram muito valor com poucas linhas de código. Todas essas evoluções vieram para facilitar a otimização, extensibilidade, manutenibilidade e vários outros fatores presentes em aplicações Web.

Boas ferramentas geram mais valor. Da mesma forma que um pintor cria traços mais detalhados com bons pincéis, um programador também precisa de uma série de ferramentas que o auxiliem na construção de aplicações Web. Nos capítulos a seguir, um pouco mais sobre o uso desses meios disponíveis para criação de aplicações será apresentado.

3 O DESENVOLVIMENTO WEB

Após uma breve análise histórica podemos agora direcionar nossas discussões para o foco principal deste trabalho que é a criação do módulo de uma aplicação Web. Mas, afinal, quais os passos iniciais para começarmos o desenvolvimento de um WebApp? Saber bem estes passos é o fator determinante para que todas as peças de um projeto se encaixem e o resultado final atenda as expectativas tanto do cliente quanto do desenvolvedor.

3.1 Panorama geral do desenvolvimento Web

Alguns pontos importantes devem ser levantados antes do início da construção de um WebApp. Definir algumas operações que englobam este desenvolvimento torna as interações na construção de uma aplicação mais lúcidas e intuitivas.

De início, devemos entender o conceito de **abstração** que é de grande importância na computação. A abstração consiste em representar um contexto do mundo real no mundo computacional. Portanto, cada pessoa abstrai um problema de forma diferente (SOMMERVILLE, 2003). A abstração é única para cada desenvolvedor, porém alguns passos são comuns e devem ser seguidos para que um problema abstraído possa ser solucionado por meio de softwares.

Primeiramente, para que uma solução seja criada, devemos observar o contexto em que o problema se encontra. Logo, devemos saber bem o escopo do problema, sua abrangência, regras de negócio e todas as relações que ele precisa atender para ser executado.

Assim sendo, para realizar uma boa abstração, o desenvolvimento da aplicação Web começa pelo **levantamento de requisitos**. Nesta fase do desenvolvimento um apanhado geral sobre as especificidades da aplicação deverá ser coletado para identificar o que deve ser criado. Segundo Sommerville (2003), os **requisitos** são o que definem tudo que um sistema irá executar em suas funcionalidades. Por meio dos requisitos é possível identificar as necessidades dos clientes e traçar os caminhos que devem ser tomados para que estas necessidades sejam alcançadas.

Após entender bem os requisitos, passa-se para fase de criação de um **protótipo**. A criação do protótipo ajuda na validação dos requisitos levantados anteriormente, pois o protótipo simula visualmente o sistema que será desenvolvido. Dessa forma, eles se tornam ferramentas extremamente relevantes no processo de desenvolvimento, por possibilitarem ao cliente uma visão geral do sistema para que ele aprove ou rejeite a abstração feita pelo desenvolvedor. Em caso de rejeição, novos protótipos são criados até que as expectativas e necessidades dos requerentes da aplicação sejam atendidas.

A criação de **diagramas** que abordam a aplicação também são uma boa prática para seu sucesso. Por meio dos diagramas podemos ter uma visão mais detalhada de como as operações do sistema se comunicarão. Um dos diagramas mais conhecidos e mais usados é o de **casos de uso**, por meio dele percebe-se como se dá a interação de um sistema com o ambiente no qual ele está inserido. Além deste diagrama há também vários outros, como, o **diagrama de classe**, **diagrama de componentes**, etc. (SOMMERVILLE, 2003).

Estas etapas iniciais também estão inseridas dentro das **metodologias de desenvolvimento**. As metodologias de desenvolvimento auxiliam na organização e distribuição das tarefas determinando quando serão executadas, por quem, duração, dentre outros. Existem dois tipos de metodologia de desenvolvimento de software:

1. **Metodologias tradicionais** de desenvolvimento de software.
2. **Metodologias ágeis** de desenvolvimento de software.

Ambas buscam alcançar certos objetivos em relação ao processo de desenvolvimento, como a manutenibilidade e a consistência (ALMEIDA, 2017). As atividades iniciais de um projeto, como o levantamento de requisitos, não precisam estar ligadas a uma metodologia específica, afinal de contas, elas são aplicadas em qualquer processo de desenvolvimento.

Outra questão fundamental que precisa ser definida previamente é a **arquitetura de software** que será empregada verificando como o sistema será estruturado e como os componentes serão organizados, oferecendo uma visão clara de como será o sistema desenvolvido (SOMMERVILLE, 2003). Além disso, **frameworks** também devem ser definidos em conformidade com os **bancos de dados** para que a aplicação possa ser desenvolvida de forma estruturada e com mais praticidade para os programadores. Assim, pode-se evitar operações que seriam prejudiciais a nossa aplicação, ou podemos tomar medidas favoráveis para o desenvolvimento esquivando de possíveis conflitos futuros.

Durante a implementação do sistema deve-se atentar sempre aos processos de **testes** do software que está sendo criado. Para Sommerville (2003), estes testes são extremamente importantes para verificar se o sistema realmente executa o que é esperado. Caso os resultados apresentem contradições em relação ao proposto inicialmente, correções são realizadas e um novo teste é aplicado. A fase de testes só finaliza quando o sistema se adéqua ao apresentado e validado no início do projeto.

Com isso, através deste apanhado geral sobre o desenvolvimento Web, podemos identificar alguns dos recursos que estão presentes durante este processo de criação de um software (Figura 2). Todos estes recursos se somam para que ao final seja gerada uma aplicação de qualidade.

Figura 2 – Recursos presentes no desenvolvimento de aplicações Web

Fonte: Próprio autor.

Como a Web já está bem difundida existem diversas ferramentas presentes em cada momento do desenvolvimento, desde o levantamento de requisitos até os testes finais no sistema. Todos os conceitos mencionados até então referem-se a procedimentos pré e pós desenvolvimento. Estas etapas são o foco do trabalho e serão detalhadas no decorrer do texto.

Após esse apanhado geral das principais etapas do desenvolvimento, a seção 3.2 irá apresentar a plataforma **Help Me**, uma aplicação Web que será utilizada no decorrer do texto para exemplificar os conceitos apresentados.

3.2 Plataforma Help Me

Muitas pessoas que estão inseridas no ambiente acadêmico sentem muita dificuldade em encontrar o apoio necessário quando tem uma dúvida em alguma disciplina. Sendo assim, o nicho de mercado que o sistema Help Me busca atingir é a comunidade acadêmica. Algumas vezes esta dificuldade vem pela falta de motivação, outras por falta de conhecimento e até mesmo timidez. Esses fatores colaboram para que as dificuldades em determinadas matérias se estendam e acabem prejudicando o discente no decorrer de seus estudos. A plataforma Help Me surge com uma proposta para solucionar problemas relacionados ao aprendizado de alunos em matérias e conteúdos diversos.

A plataforma irá contar com o cadastro de um usuário que busque e ofereça ajuda em diversas áreas e especificidades. Todos os usuários da plataforma Help Me serão conectados e poderão interagir uns com os outros para que a troca de conhecimento em diversas áreas possa ser consolidada de forma rápida e prática.

O texto abaixo exemplifica o uso da plataforma:

"Maria começou seus estudos no curso de Sistemas de Informação e está matriculada na disciplina de Algoritmos e Estruturas de Dados I. Entretanto, Maria possui uma grande dificuldade para entender estruturas de repetição e, por conta disso, busca ajuda por meio da plataforma online Help Me.

Valter já concluiu a disciplina de Algoritmos e Estruturas de Dados I e tem disponibilidade para ajudar com conteúdos relacionados a essa disciplina. Valter está cadastrado na plataforma e poderá ser contactado para tirar dúvidas e auxiliar na disciplina citada.

Por meio da plataforma Help Me, Maria e Valter se conectam e Maria consegue sanar suas dúvidas por meio dos conhecimentos de Valter."

Através do exemplo podemos perceber uma aplicação prática de como a plataforma poderá solucionar problemas e ajudar diversas pessoas. A proposta principal é desenvolver uma aplicação Web abordando os principais conceitos de engenharia de software e as tecnologias atuais mais utilizadas para o desenvolvimento Web.

Por meio desta aplicação um filtro será criado de acordo com seus interesses e assim que for estabelecida a conexão, os envolvidos poderão se contactar para que as dificuldades apresentadas possam ser resolvidas. Mas antes de dar início a construção do módulo que será usado como exemplo, é necessário entender também como ocorre o processo de desenvolvimento de

um software, juntamente de quais as metodologias envoltas por este processo e como elas são usadas.

4 O PROCESSO ENVOLTO NO DESENVOLVIMENTO DE UM SOFTWARE

Quando os softwares começaram a ser criados, a escassez de organização no seu processo de desenvolvimento gerou produtos que muitas vezes não atendiam às necessidades dos clientes. Essa escassez acarretava também em atrasos no prazo de entrega, sistemas com custo muito alto, ou ainda projetos não finalizados devido à falta de qualidade do produto entregue.

Para tornar este processo de desenvolvimento mais organizado, surge a **Engenharia de Software**, com o objetivo de agregar um planejamento cuidadoso, com qualidade e estruturação (PRESSMAN, 2011). Com a Engenharia de Software surgiram também os **processos de software**, como uma série de atividades e ações que, quando relacionadas, levam à produção de um software com qualidade (SOMMERVILLE, 2011).

Os processos são muito importantes, pois através deles a consistência e estrutura podem ser adicionadas a um conjunto de atividades. A importância dessa estrutura está ligada a forma como os softwares são desenvolvidos. Na maioria das vezes, este desenvolvimento envolve diversas pessoas trabalhando em conjunto e criando juntos a solução. Assim sendo, entender bem o processo de estruturação do projeto faz com que todos os envolvidos colaborem de forma positiva em sua criação.

Cada organização desenvolve um modelo de processo que melhor atenda às suas necessidades. A escolha de um processo de software a ser adotado envolve as características dos colaboradores presentes na empresa, características da aplicação que será desenvolvida, tempo necessário para o desenvolvimento e diversos outros fatores.

Dessa forma, percebemos que a criação de um produto de software, assim como qualquer outro produto, possui formas que visam seu controle. Para realizar esse controle existem os processos de software que gerenciam e auxiliam nas etapas de desenvolvimento da aplicação.

Existem muitos **modelos de processo de software** e uma empresa pode definir seus próprios processos, porém Sommerville (2003) destaca que algumas atividades ainda continuam sendo comuns a todos os modelos. São elas:

- **Especificação de software:** Definição das funcionalidades que o software desenvolvido irá executar.
- **Projeto e implementação:** A aplicação será desenvolvida com base em suas especificações. Nesta fase ocorre a implementação da aplicação por meio de uma especificação que foi levantada anteriormente.
- **Validação de software:** A validação é necessária para garantir que as prioridades do cliente estão sendo atendidas de forma satisfatória e estão realmente agregando valor às suas necessidades.
- **Evolução do software:** Caso necessário, o software deve possuir facilidade de evolução, para atender as novas necessidades do cliente que poderão surgir com o decorrer do tempo.

A seguir, na seção 4.1, apresentamos alguns dos modelos de processo de software.

4.1 Modelos de Processo de Software

Os modelos de processo de software também são chamados de **modelos de ciclo de vida do software**. O objetivo de seu uso é a necessidade de um controle que represente o fluxo do processo de criação de um software (SOMMERVILLE, 2011).

Modelos de processo de software representam uma maneira de agregar ordem, estabilidade e controle na criação de softwares (PRESSMAN, 2011). Inúmeras circunstâncias podem acontecer e, as escolhas dos envolvidos na criação será um dos fatores determinantes no sucesso ou fracasso do processo de desenvolvimento.

Atualmente, os modelos são divididos em duas categorias: os **modelos tradicionais** e os **métodos ágeis**. Estas duas categorias podem ser representados na figura que contém os recursos de uma aplicação Web (Figura 3).

Figura 3 – As Metodologias de desenvolvimento de software

Fonte: Próprio autor.

4.1.1 Modelos tradicionais

Os modelos tradicionais de desenvolvimento de software foram o pontapé inicial para construção de aplicações melhores e que garantissem maior qualidade.

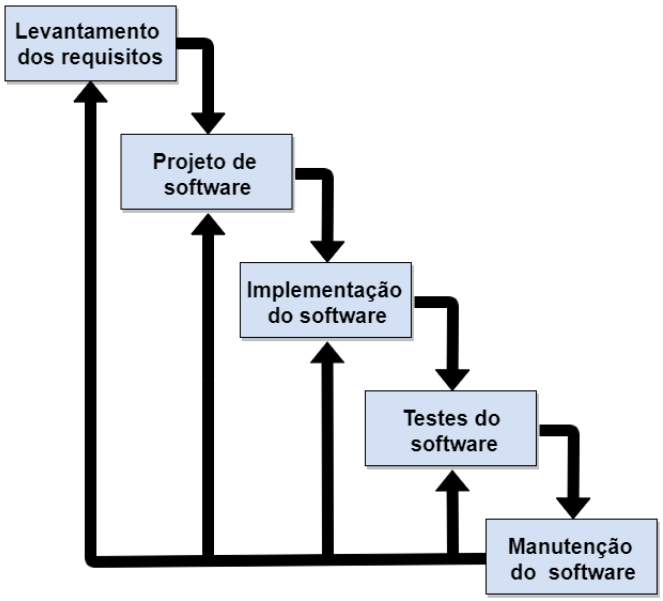
Dentro da categoria dos modelos tradicionais encontram-se diferentes modelos de processo de software. Nas subseções seguintes (4.1.1.1 e 4.1.1.2) detalhamos dois deles: o **cascata** e o **orientado a reuso**.

4.1.1.1 Cascata

Para Moura e Moreira (2012), o desenvolvimento por meio do modelo cascata é sequencial e leva em consideração que a ordem de realização das tarefas terá uma execução pré definida, não se atentando muito ao gerenciamento dos riscos que o projeto posteriormente poderá enfrentar.

Dessa forma o modelo cascata traz à tona o desenvolvimento sequencial do software, ou seja, a aplicação desde seus requisitos até o produto final é desenvolvida passo a passo. Uma curiosidade sobre o nome *"cascata"*, é que ele se originou da própria forma que as tarefas são resolvidas: à medida que uma etapa é concluída, posteriormente, outra tarefa é executada, isso equivale a *"cair"* para outra fase, assim como uma cascata. A Figura 4 apresenta uma imagem ilustrativa de como se estabelece o funcionamento do modelo cascata.

Figura 4 – Estrutura do modelo cascata



Fonte: SOMMERVILLE, 2011. Adaptado.

A primeira etapa é a definição de requisitos, onde é feito um levantamento junto ao cliente das funcionalidades do software, ou seja, o que o software irá fazer. Posteriormente, será criado o projeto de sistemas e de software, onde serão verificados como o software se adequará ao ambiente que está sendo proposto. Na etapa de implementação será realizada a codificação do software e analisamos se o que foi implementado está funcionando corretamente e atendendo os requisitos iniciais.

A próxima fase é a integração de todas as partes do sistema e a realização de testes do sistema como um todo. Nesta fase é feita uma consulta ao cliente para que este possa atestar se o que está sendo desenvolvido realmente é o que ele necessita e se suas expectativas estão sendo atendidas por meio da aplicação. A última etapa é a manutenção em possíveis falhas ou erros cometidos ao longo do projeto.

Como no desenvolvimento de software é quase impossível finalizar totalmente uma etapa para em seguida iniciar a próxima, no modelo cascata, se em algum momento houver a necessidade de alteração, basta retornar para a fase onde ocorreram as modificações. Porém, após a mudança, todas as fases posteriores também serão visitadas e modificadas.

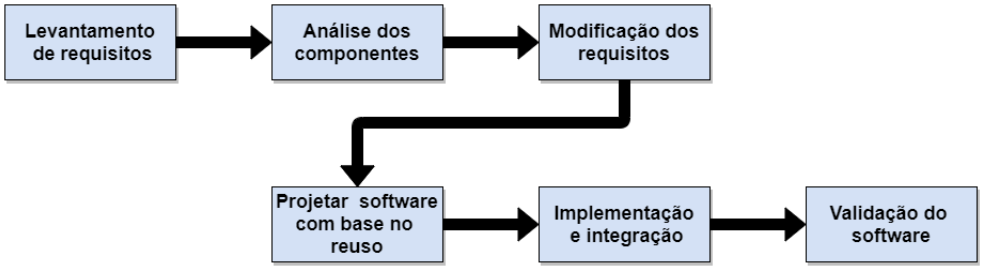
4.1.1.2 Orientado a reuso

Este modelo parte da ideia de que existem componentes de software já desenvolvidos que podem ser reaproveitados em outros projetos e que na criação de um software poderão ser unidos gerando uma nova aplicação (SOMMERVILLE, 2011).

Como exemplo, podemos pensar na criação de um carro, onde as partes que o compõe são “encaixadas” e ao final temos o carro construído. Por meio dessas partes evita-se a elaboração de cada componente do zero, economizando tempo, mão de obra e consequentemente dinheiro.

A Figura 5 ilustra as etapas envolvidas no desenvolvimento orientado a reuso.

Figura 5 – Desenvolvimento orientado a reuso



Fonte: (SOMMERVILLE, 2011). Adaptado.

Como padrão para o início da criação de um software, em um primeiro momento deverá ser feita a especificação de requisitos, pois somente por meio deles que o time de desenvolvimento terá conhecimento do que será construído. Após este levantamento dos requisitos, será verificado quais componentes já existentes se adéquam ao proposto, dadas as especificações feitas na fase inicial. Na terceira fase ocorre a modificação dos requisitos, caso exista um componente similar, mas que não seja igual ao que será desenvolvido, o requisito pode ser alterado para que o componente se encaixe no projeto e seja reutilizado.

Logo após essas fases, é feito o projeto de como será o sistema com os componentes que estão sendo reutilizados. Em seguida, ocorre o desenvolvimento e a integração dos componentes e, por último, temos a validação do sistema.

O desenvolvimento orientado ao reuso facilita e agiliza a criação de aplicações, gerando economias no processo de implementação. Sommerville (2011) afirma que uma das vantagens mais características da Engenharia de Software orientada ao reuso é a diminuição do software que deverá ser desenvolvido, reduzindo também riscos nesta criação. Porém, uma desvantagem ao utilizar a Engenharia de Software orientada ao reuso é que o controle sobre uma possível evolução do sistema será perdido, pois como o software é criado por meio de componentes reutilizados, a organização não possui controle sobre suas novas versões (SOMMERVILLE, 2011).

4.1.2 Métodos ágeis de desenvolvimento de software

Até por volta dos anos 2000, os modelos tradicionais de desenvolvimento eram muito utilizados na criação de aplicações. Com a evolução tecnológica e o surgimento de problemas cada vez mais complexos, os modelos tradicionais de desenvolvimento passaram a ser inviáveis.

A dinamicidade do mercado de software após o ano 2000 fez com que o desenvolvimento de software adotasse um novo caminho. Os processos estavam sendo alterados em grande velocidade, desenvolvedores não poderiam usar métodos tradicionais que exigissem uma maior dedicação de tempo em sua criação (SOMMERVILLE, 2011).

Algumas características dos softwares atuais, que justificam o uso dos métodos ágeis, são a velocidade de mudança das regras de negócio, a forma como os requisitos de uma

aplicação pode ser alterada, entre outros. Devido a essas constantes alterações a agilidade é um fator relevante para que o software consiga ser finalizado atendendo a todas necessidades.

Segundo Ribeiro e Ribeiro (2015), em 2001 vários especialistas em projetos de software criaram o **Manifesto Ágil**, neste manifesto uma série de levantamentos foram feitos no intuito de descobrir quais as melhores formas de criação de software.

O Manifesto Ágil de imediato já expõe que se passa a valorizar:

- As pessoas e as interações mais que as ferramentas e fluxos da operação.
- O software já estar em funcionamento para o usuário em troca de uma documentação extensa sobre o que o sistema faz.
- A interação e compromisso com o cliente como alternativa à burocracia e negociação dos contratos de trabalho.
- A adequação as necessidades do cliente ao invés de seguir um processo estruturado de criação.

Para Sommerville (2011), todos os métodos ágeis possuem princípios gerais que devem seguir, são eles:

- **O envolvimento com o cliente:** O cliente deve possuir um contato constante com o time que está desenvolvendo o software, para que dessa forma acompanhe e auxilie no que será desenvolvido.
- **Entrega incremental**: Incrementos do software são desenvolvidos e avaliados com o cliente.
- **Pessoas e não processos:** As pessoas do time devem desenvolver suas próprias habilidades de trabalho sem que processos predefinidos retenham sua capacidade. Dessa forma as pessoas terão uma liberdade maior para poderem trabalhar.
- **Aceitar mudanças:** Os métodos ágeis foram criados visando mudanças constantes que um software irá sofrer, portanto, o projeto de desenvolvimento do software deve ser executado de forma que aceite mudanças de requisitos, mercado, etc.
- **Manter a simplicidade:** O software, juntamente ao seu processo de desenvolvimento, deve ser o mais simples possível, para não agregar complexidade na criação de uma aplicação.

Existem também as vantagens e desvantagens associadas aos métodos ágeis (SOMMERVILLE, 2011). Dentre as vantagens podemos citar:

- São bem aplicáveis em projetos pequenos.
- Garantem um entrega incremental do software que será desenvolvido, assim o cliente poderá ter em mãos partes funcionais do software no decorrer do desenvolvimento.
- Visam a agilidade no desenvolvimento da aplicação, fazendo com que o cliente tenha o produto em mãos de forma mais rápida.

Quanto às desvantagens, temos que:

- Clientes devem possuir disponibilidade para trabalhar com a equipe de desenvolvimento.

- A equipe pode não se adequar ao formato de desenvolvimento ágil, ou pode não possuir a maturidade necessária para trabalhar dessa maneira.
- A própria organização pode se restringir a não empregar métodos ágeis no desenvolvimento dos sistemas devido a sua cultura muito tradicional.
- Não gera uma documentação detalhada e específica do software desenvolvido, visando a economia de tempo. Esta falta de documentação para algumas empresas (como um banco) pode ser um empecilho.

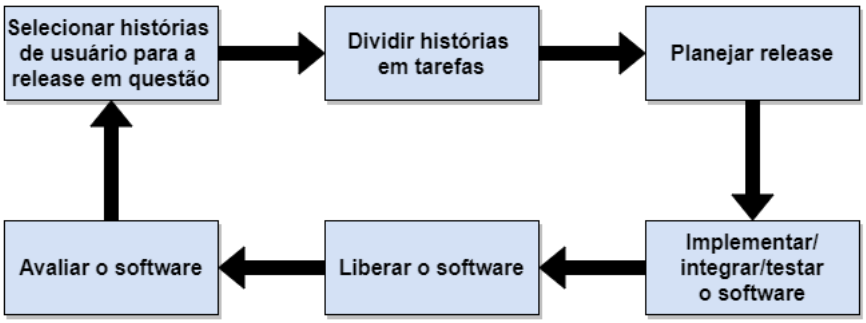
Assim como nos modelos tradicionais de desenvolvimento de software, os ágeis também possuem vários exemplos que podem ser citados, dois dos mais conhecidos e utilizados são o **Extreme Programing (XP)** e o **Scrum**.

4.1.2.1 Extreme Programing (XP)

O Extreme Programing atua com a entrega de incrementos do software. Cada entrega feita é chamada de **release** e está associada a implementação das **histórias de usuário** (ou **user stories**) (RIBEIRO; RIBEIRO, 2015). Uma história de usuário é a representação de uma funcionalidade ou melhoria que será desenvolvida em um sistema.

A Figura 6 expõe uma representação do ciclo de uma *release* em XP.

Figura 6 – Estrutura do Extreme Programing



Fonte: SOMMERVILLE, 2011. Adaptado.

Inicialmente seleciona-se quais as histórias de usuário que farão parte da próxima entrega. Com as histórias de usuário em mãos a implementação das funcionalidades é feita. Esta implementação gera incrementos de sistema que já poderão ser utilizados pelos usuários finais daquela aplicação. Quando o usuário tem acesso a esta liberação ele executa os testes necessários para validar se o que foi entregue atende suas necessidades. Após finalizar o ciclo, o processo é iniciado novamente com outras histórias de usuário.

Para Sommerville (2011), algumas práticas devem ser seguidas quando se trata do desenvolvimento de software por meio do Extreme Programing, são elas:

- **Planejamento incremental:** Cada história será inserida em um cartão de história e sua execução será de acordo com o tempo disponível e a prioridade de sua criação.

- **Pequenas *releases*:** Um conjunto mínimo de funcionalidades é criado para que o cliente seja capaz de ver o sistema funcionando e possa até mesmo executar algumas ações por meio dele.
- **Projeto simples:** o projeto não pode agregar uma grande complexidade para facilitar sua criação.
- **Cliente no local:** o cliente deve estar presente no local, junto aos desenvolvedores.

O cliente que escolhe qual a próxima história que deverá ser implementada de forma que melhor lhe atenda. Assim, percebemos a importância de o cliente estar sempre envolvido na criação do software.

4.1.2.2 Scrum

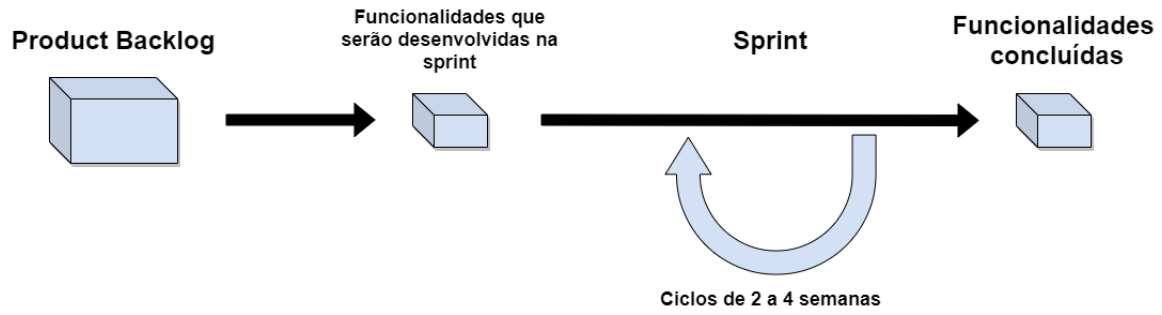
O Scrum é utilizado no gerenciamento de projetos e atua em cenários caóticos onde a dinamicidade dos requisitos, tecnologia e o ambiente em que o sistema será inserido muda de forma constante, fazendo com que não seja possível ter um controle adequado dessas mudanças (SUTHERLAND, 2014).

Segundo Sutherland (2014), o Scrum tem como base 3 pilares fundamentais: Transparência; Inspeção; Adaptação.

1. **Transparência:** Todos sabem realmente tudo que está acontecendo no decorrer do projeto.
2. **Inspeção:** O tempo todo o projeto é inspecionado para garantir que seu andamento está correto.
3. **Adaptação:** Como o projeto está propenso a mudança, o processo Scrum deve possuir também uma boa capacidade de adaptação.

A Figura 7 ilustra a forma como ocorre o Scrum, ou seja, seu passo a passo de execução.

Figura 7 – Estrutura do Scrum



Fonte: Próprio autor.

Para que um projeto seja feito com apoio do Scrum, inicialmente, é necessário haver uma visão geral do que será construído. Esta visão geral é desmembrada em funcionalidades, que serão responsáveis por alcançar os objetivos. Todas as funcionalidades juntas geram uma lista que é chamada de **Product Backlog** (lista de pendências do produto). A ordem de de-

senvolvimento dessas funcionalidades está atrelada às prioridades, ou seja, as funcionalidades que serão desenvolvidas inicialmente são as que agregaram maior valor ao negócio do cliente (SOMMERVILLE, 2011).

Após a definição do Product Backlog, passa-se para a divisão das **Sprints** que irão ser necessárias para criação da aplicação proposta. Uma Sprint é basicamente um ciclo curto de implementação de algumas funcionalidades que estão presentes no Product Backlog. Este "ciclo curto" é definido por cada time que trabalha no desenvolvimento de uma aplicação, mas de forma geral, times e empresas gostam de estabelecer que cada Sprint terá a duração de duas semanas.

Em todo início da Sprint ocorre o chamado **Planejamento da Sprint** (ou Planning), ele é necessário para definir quais itens presentes no Product Backlog serão abordados dentro da Sprint que irá começar. Estas funcionalidades são definidas de acordo com as prioridades de entrega de valor para os usuários do sistema, do tempo disponível para implementação das funcionalidades e da capacidade de trabalho da equipe (SUTHERLAND, 2014). Após a finalização de uma Sprint, um incremento do produto é entregue e, ao criar todas as funcionalidades do Product Backlog, teremos o projeto construído.

Para Sutherland (2014), o Scrum adota também a chamada **Reunião Diária** (ou Daily), que é basicamente a realização de reuniões diárias de 15 minutos que acontecem todo dia no mesmo local e horário. Nestas reuniões cada membro do time de desenvolvimento responde a três perguntas em relação ao desenvolvimento da Sprint.

1. O que você fez ontem?
2. O que vai fazer hoje?
3. Existe algum impedimento para que o objetivo da Sprint seja alcançado?

Ao final de cada Sprint é verificado se o que foi produzido atende às expectativas do cliente. Nesta análise, caso surja alguma mudança que deverá ser realizada, o Product Backlog é atualizado.

Existe também o que chamamos de **Retrospectiva**. Nesta etapa, após a equipe mostrar o resultado da Sprint, é feita uma reunião na qual é discutido o que deu certo no decorrer das semanas de desenvolvimento, o que pode ser melhorado e o que não pode mais ser feito.

Esta é a estrutura fundamental do Scrum, porém vale ressaltar que cada empresa adiciona a ele características únicas que melhor os atenda em suas atividades. Sendo assim, diferentes formatos de Scrum podem ser vistos em diferentes organizações, o mesmo vale para os demais modelos vistos.

4.2 Usando Scrum como metodologia de desenvolvimento

Como a proposta do trabalho é desenvolver um módulo de uma plataforma para fins educacionais, deve-se levar em consideração o uso de algum processo de desenvolvimento de software. Por meio de um processo estruturado de criação, um software pode ser desenvolvido atendendo às expectativas do cliente no prazo estabelecido, com um custo viável e com bom desempenho.

Os modelos de processo de software são essenciais para uma representação mais clara sobre o software que está sendo desenvolvido. Nas seções anteriores mostramos diferentes metodologias de desenvolvimento. Dadas as duas categorias de modelos de processo de software vistos, que são os tradicionais e os ágeis, na presente seção detalharemos a metodologia Scrum que faz parte da metodologia ágil.

Por meio das características presentes no Scrum e de seu constante uso no mercado de desenvolvimento de softwares, ele será adotado como metodologia para desenvolvimento do sistema Web que servirá de exemplo para demonstrar os processos e conceitos apresentados. Antes de estabelecer a atuação que o Scrum terá na aplicação, será feita uma análise de como ele pode ser moldado para atender a uma determinada empresa. Lembrando que os princípios gerais dos métodos ágeis apresentados por Sommerville (2011) continuam sendo base para qualquer um dos métodos propostos.

4.2.1 Personagens do Scrum

Cada empresa pode moldar as metodologias de desenvolvimento da forma que melhor se adequa às suas necessidades e os atenda. A fim de esclarecer ainda mais como estas técnicas se aplicam, a seguir apresentaremos como ocorre o processo de desenvolvimento de uma aplicação por meio da metodologia Scrum em uma empresa que trabalha com a criação de produtos de software.

Antes da análise do processo Scrum, apresentaremos um apanhado geral sobre alguns cargos que estão envolvidos na criação de uma aplicação em uma empresa que tem como objetivo o desenvolvimento de softwares. Vale ressaltar que alguns desses cargos são mencionados por Pressman (2011) e atuam para garantir a qualidade do produto de software do início ao fim.

- **Arquiteto de Software:** É a pessoa com um grande conhecimento de tecnologias e ferramentas existentes na criação de softwares. Seu conhecimento é fruto de experiências diversas que já vivenciou durante sua carreira, isso faz com que ele desenvolva suas capacidades de orientar e determinar o que é melhor para o andamento de um projeto.
- **Product Owner (PO)**: Faz o intermédio entre o cliente e o time de desenvolvimento, o PO é quem deixa a equipe técnica a par do que será criado e prioriza as entregas que serão geradas com base nas necessidades do cliente.
- **Scrum Master (SM):** É responsável por garantir que a equipe siga as boas práticas presentes na metodologia Scrum durante o processo de desenvolvimento de uma aplicação.
- **Desenvolvedor Líder (DL):** É o membro de um time responsável por gerenciar e fiscalizar toda a parte técnica de desenvolvimento. Possui mais experiência com o desenvolvimento e uso de tecnologias e, devido a isso, está presente na criação de um projeto como desenvolvedor e também como um apoio técnico.
- **Equipe Técnica:** Todos os membros que compõe o time de desenvolvimento. São os responsáveis por fazer com que as histórias de usuário levantadas se tornem funcionalidades de um sistema.

- **Área de Negócio:** É o cliente para quem está sendo desenvolvido um novo software, ou ocorrendo integração de funcionalidades e melhorias em um já existente.

4.2.1.1 A metodologia Scrum na prática

O primeiro passo para o desenvolvimento de uma aplicação é uma reunião feita com todos os interessados e envolvidos na criação do software, ou seja, todos os **Stakeholders** (parte interessada). Esta reunião recebe o nome de **Design Sprint**. Esse é o momento de entender qual problema o cliente busca resolver por meio de um software (SCHWABER; SUTHERLAND, 2017).

Com o Design Sprint ocorre um **Brainstorming** (chuva de ideias) relacionado ao projeto. As ideias geradas envolvem todos os desenvolvedores, a liderança técnica, o Arquiteto de Software, enfim, todos os envolvidos na criação da aplicação. Este processo é feito por meio de *"bonequinhos"* físicos que agem como atores, post-its que são colados na parede e diversas outras ferramentas para que a interação com o cliente seja mais precisa e suas "dores", que são as necessidades que os clientes buscam atender por meio de softwares, sejam melhor compreendidas.

Seguindo ainda o Scrum, após o Design Sprint um **Product Backlog** (lista de pendências do produto) será gerado. Este Backlog possui todas as funcionalidades que deverão ser desenvolvidas para que juntas formem um sistema completo que atenda às necessidades do cliente.

Com base no Backlog e visando sempre as necessidades do cliente, o Product Owner (PO) determina as histórias de usuário que devem ser priorizadas em uma **Sprint**. Mesmo possuindo nomes semelhantes a Sprint e o Designer Sprint são processos diferentes, o Designer Sprint é uma etapa de planejamento presente em um projeto, já a Sprint é um ciclo de tempo (geralmente de 2 semanas) em que um time atua em uma série de histórias de usuário definidas pelo PO. Estas histórias quando implementadas geram um conjunto de *releases* da aplicação, ou seja, ao final de cada Sprint novas funcionalidades serão incrementadas no software.

Baseado nas histórias que o Product Owner (PO) julga relevantes para uma Sprint, o time de desenvolvimento passa para o chamado **Planning Poker**, que está presente na metodologia Scrum e nada mais é, que estimar a complexidade de cada história (SERIES, 2005).

Com a definição da pontuação das histórias de usuário e com base na quantidade média de pontos que o time de desenvolvimento consegue implementar por Sprint, é definido quais histórias irão compor a Sprint em questão. Em alguns casos, nem todas as histórias definidas pelo Product Owner serão comportadas em uma única Sprint, fazendo com que algumas histórias só possam ser implementadas em Sprints posteriores.

Ao final do ciclo das semanas da Sprint é realizada sua **Retrospectiva**, onde os pontos positivos e negativos serão levantados pelo time, para que os pontos positivos possam ser ressaltados e tenham continuidade e os pontos negativos possam ser abordados e melhorados nas Sprints futuras. Assim, após finalizar uma Sprint outra terá início e ao final do ciclo das Sprints o produto estará completo e as necessidades do cliente atendidas.

Por meio do exemplo pode-se perceber como a metodologia Scrum pode ser moldada em conformidade com a empresa. Isso faz com que não exista uma metodologia que atenda a todas as organizações da melhor forma, entretanto, existe uma estrutura base que é modificada e pode gerar o crescimento, desenvolvimento e agilidade que busca uma organização.

4.2.2 Aplicando Scrum no projeto Help Me

Após optarmos pelo Scrum como metodologia de desenvolvimento, vamos definir o Design Sprint do projeto Help Me que terá como objetivo a ligação entre usuários por meio de uma plataforma Web. A plataforma oferecerá uma conexão entre pessoas para proporcionar a troca de conhecimento em conteúdos diversos.

Inicialmente, analisaremos as demandas e o problema que o sistema busca solucionar. Na seção 3.2, analisamos que a dor de Maria consiste na ausência de uma plataforma que auxilie a ligação entre pessoas para o compartilhamento de conhecimentos. Já Valter, é apaixonado por ensinar, porém não encontra com frequência pessoas com quem possa compartilhar seus aprendizados.

Neste contexto, Maria e Valter resolvem se unir. Eles perceberam que seus problemas podem ser resolvidos por meio de um sistema Web. Sendo assim, buscam uma empresa de desenvolvimento de softwares para atender suas demandas.

Em um primeiro momento Maria e Valter, que representam a Área de Negócio, fazem contato com uma equipe composta por: Arthur (Arquiteto de Software), Pedro (Product Owner), Samantha (Scrum Master) e Daniel (Desenvolvedor Líder). Neste contato inicial é realizado o Design Sprint, onde a equipe responsável fica a par da situação e das dores de Maria e Valter.

Após definidos os Stakeholders, que são os envolvidos e interessados no sistema, a equipe precisa fazer um levantamento de quais são os pontos chave que a aplicação irá atender, ou seja, identificar quais dores serão sanadas. Esses levantamentos irão gerar as funcionalidades necessárias para que a aplicação atenda à estas necessidades.

Por meio do Design Sprint a equipe definiu que a maior dor dos clientes é:

- *"Dificuldade de estabelecer comunicação com outras pessoas para troca de conhecimento."*

Com isso, a equipe propõe o desenvolvimento de uma plataforma Web onde os interessados em trocar conhecimentos poderão se cadastrar e atribuir quais conhecimentos estão dispostos a compartilhar. Assim, qualquer usuário poderá se dispor a ensinar algo, como também poderá entrar em contato com outro usuário que possa lhe ensinar algo do seu interesse. Por meio do Design Sprint foi definida também o nome da plataforma: **Help Me**.

No sistema proposto, conteúdos poderão ser pesquisados em uma base de dados. Assim que o usuário encontrar o que deseja, ele poderá acessar o perfil da pessoa relacionada ao tema. No perfil será exposto a forma de contato para que possa ocorrer uma comunicação rápida e eficaz entre ambos.

Dessa forma, quando Maria carecer de ajuda em uma matéria (por exemplo, algoritmos e estrutura de dados) e fizer a busca na plataforma, o perfil de Valter será exibido. Valter está cadastrado na plataforma e o conteúdo pesquisado por Maria está na sua lista de conhecimentos. Assim, através do Design Sprint pode ser vislumbrado um cenário geral de como a plataforma irá funcionar e seu desenvolvimento, da forma que melhor atenda às necessidades dos clientes (Maria e Valter).

Após o Design Sprint é feito uma reunião entre Arthur (Arquiteto de Software), Pedro (Product Owner), Samantha (Scrum Master) e a parte técnica composta por Daniel (Desenvolvedor Líder) e todos os demais desenvolvedores. Nesta reunião ocorrerá a definição de quais meios serão usados para execução do projeto: linguagem de programação, banco de dados, frameworks e outras dependências para o desenvolvimento. Esses recursos utilizados no desenvolvimento de um projeto serão apresentados com mais detalhes nos capítulos 5 e 6.

4.2.2.1 Épico e Feature

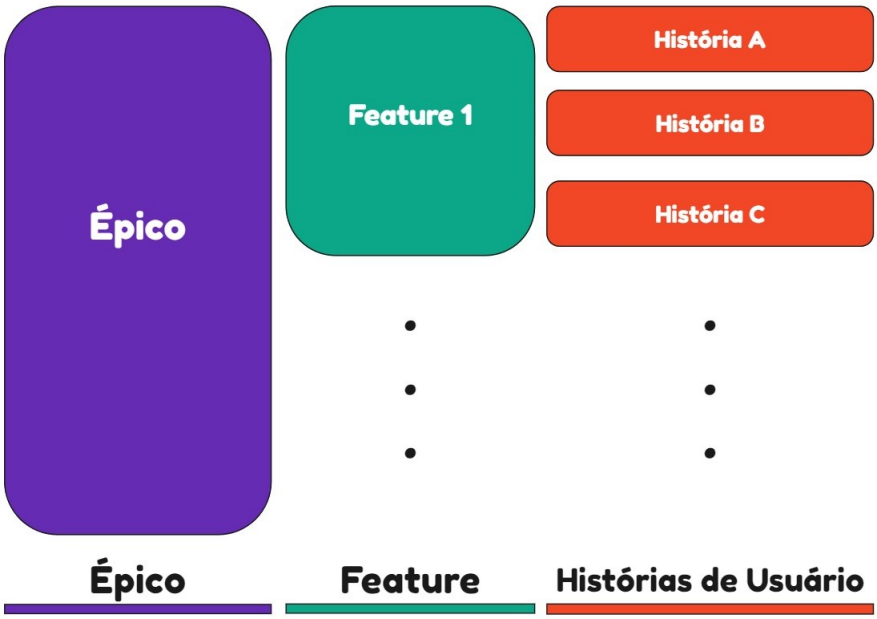
Neste momento, Arthur (Arquiteto de Software) e Pedro (Product Owner), juntamente com a parte técnica responsável pela execução do projeto entram em ação. São eles que definem as tecnologias que serão usadas para desenvolvimento da plataforma Help Me, solicitada por Maria e Valter.

Após esta etapa, iniciamos a construção do Product Backlog. A dor anteriormente verificada na fase de Design Sprint é agora quebrada em **Épicos**. Um Épico é um tópico abrangente sobre o produto que posteriormente será dividido em histórias de usuário. Por isso, o Épico gera um pouco de incerteza quanto à sua complexidade e tamanho (SERIES, 2005).

Entre o Épico e as histórias de usuário existem as **Features**, que são uma aproximação do que o Épico irá produzir. Por meio da Feature, temos uma visão mais clara do que será criado e entregue (PRESSMAN, 2011).

Os responsáveis por definir os Épicos em uma aplicação são Pedro, Maria e Valter, ou seja, o Product Owner e os responsáveis pela Área de Negócio. Após a definição dos Épicos, são desenvolvidas Features, e a partir das Features, são criadas as histórias de usuário que juntas cumprirão o objetivo do Épico. Portanto, cada Épico tem suas Features e cada Feature tem suas histórias de usuário (Figura 8).

Figura 8 – Organização de Épicos, Features e histórias de usuário



Fonte: Próprio autor.

Um conjunto de Épicos formam o Product Backlog de desenvolvimento do projeto e, a partir dele, o desenvolvimento da aplicação tem seu início. Como o objetivo do trabalho é a implementação de um módulo da plataforma Help Me, será considerado apenas 1 Épico definido por Pedro (Product Owner), com apoio de Maria e Valter (Área de Negócio).

O Épico definido foi:

- **"Controle de conta pessoal"**

Com base no Épico a seguinte Feature foi gerada:

- **"Gestão de dados do usuário"**

Desta forma, o Épico identificado anteriormente estará presente no Product Backlog do projeto Help Me, lembrando que, o Product Backlog é gerado por meio da junção de todos os Épicos criados. Com a Feature gerada serão definidas as histórias de usuário da Sprint. A Sprint é o período de algumas semanas, onde algumas histórias de usuário serão desenvolvidas e entregues aos clientes, Maria e Valter.

4.2.2.2 Histórias de Usuário

Com base na Feature definida anteriormente, Pedro (Product Owner) passa agora para o seu detalhamento, ou seja, a criação de histórias de usuário associadas a ela. Lembrando que, uma história de usuário deve trazer a resposta para 3 perguntas dentro de seu escopo, são elas: Quem solicitou a funcionalidade que será desenvolvida? O que se espera produzir? Qual o motivo de se produzir esta funcionalidade?

Após feito o detalhamento ele consegue identificar 3 histórias de usuário:

História A - Cadastrar Usuário

- Eu, como usuário(a), desejo me cadastrar na plataforma para ter acesso a outros usuários e trocar conhecimentos.

História B - Visualizar usuário

- Eu, como usuário(a), desejo visualizar meus dados de cadastro para verificar se estão corretos.

História C - Atualizar usuário

- Eu, como usuário(a), desejo editar meus dados de cadastro para atualizar minhas informações.

Agora, com a definição das histórias que poderão compor uma Sprint, a equipe técnica define **tasks** (ou **tarefas**) associadas a cada uma, para terem uma visão de sua complexidade, e só então partirem para o Planning Poker. Após análise, Daniel (Desenvolvedor Líder) e os demais membros da equipe técnica de desenvolvimento definiram as *tasks* de cada história de usuário. Ao final, ficou estabelecida a seguinte estrutura:

História A - Cadastrar Usuário

- Eu, como usuário(a), desejo me cadastrar na plataforma para ter acesso a outros usuários e trocar conhecimentos.
  1. Criar banco de dados e modelagem da tabela que será utilizada.
  2. Criar projeto no ambiente de desenvolvimento.
  3. Criar tela de cadastro do usuário.
  4. Criar conexão com o banco de dados.
  5. Validar se campos obrigatórios foram preenchidos.
  6. Inserir informações do cadastro na base de dados.

História B - Visualizar usuário

- Eu, como usuário(a), desejo visualizar meus dados de cadastro para verificar se estão corretos.
  1. Criar página de visualização dos dados do usuário.
  2. Buscar dados do banco e preencher os campos da tela.

História C - Atualizar usuário

- Eu, como usuário(a), desejo editar meus dados de cadastro para atualizar minhas informações.
  1. Modificar tela de visualização dos dados do usuário para que ela direcione à tela de atualização.
  2. Criar tela de atualização dos dados.
  3. Enviar dados atualizados para o banco de dados.

Após a definição das histórias e suas *tasks*, a equipe de desenvolvimento passa para estimativa de pontos associadas a elas, com base no Planning Poker.

4.2.2.2.1 Planning Poker

Como visto na seção 4.2.1.1 o Planning Poker é usado para estimar a complexidade de cada história de usuário. Na estimativa das histórias que Pedro (Product Owner) e a equipe técnica definiram será utilizado como pontuação a sequência de Fibonacci, onde o número seguinte é feito sempre pela soma dos dois que o que antecedem: por exemplo, 0, 1, 1, 2, 3, 5, 8, etc.

A sequência de Fibonacci foi escolhida como um indicador para complexidade das histórias de usuário. Nesta estratégia, a incerteza existente no desenvolvimento de uma história, ou seja, a falta de uma assertividade em relação a dificuldade que aquela história possui, é representada pelos intervalos entre os números dessa sequência (SERIES, 2005).

A incerteza de uma história é fruto do desconhecimento de todas as atividades que devem ser feitas para que ela seja concluída, pois a história é apenas estimada. Nesta estimativa tentamos alcançar a definição mais precisa de todas as atividades que a história possui, mas isso não quer dizer que a estimativa é precisa. No decorrer do desenvolvimento podem surgir questões que indiquem que a história estimada era mais ou menos complexa do que o previsto. Esta variação da complexidade em relação ao que foi estimado pode acontecer quando no desenvolvimento da história percebemos que ela tem mais ou menos tarefas do que o estimado.

Para jogar o Planning Poker, definimos inicialmente uma história base, que será a mais simples tratada na Sprint, que neste caso será pontuada pela equipe técnica com a complexidade 1 de acordo com a sequência de Fibonacci. A partir desta história base definiremos em quantas vezes as outras histórias que entraram na Sprint a superam em complexidade e assim é estimada a pontuação de todas as histórias presentes na Sprint. A seguir, será apresentado um exemplo de estimativa com o Planning Poker:

- A história X foi estimada em 3 pontos por meio do Planning Poker, isso quer dizer que ela é 3 vezes mais complexa que uma história de 1 ponto definida pela equipe técnica. Porém, no decorrer de seu desenvolvimento percebem que a história X era apenas 2 vezes mais complexa que a história de 1 ponto, ou seja, a estimativa não foi de acordo com o previsto.

Com esta definição, para dar início ao Planning Poker da Sprint 1 da plataforma Help Me, a equipe de desenvolvimento determina que a história mais simples que receberá a pontuação 1 será a história B:

"Eu, como usuário(a), desejo visualizar meus dados de cadastro para verificar se os dados cadastrados estão corretos."

Após a definição da história base, passa-se agora para definição da pontuação das próximas histórias que serão desenvolvidas na Sprint. Cada membro relacionado com o desenvolvimento técnico das histórias deve jogar o Planning Poker e atribuir um valor para a história em questão. Caso haja divergência entre os valores os membros que divergiram entre si devem defender seu ponto de vista até que toda equipe chegue em um consenso único da pontuação da história em questão.

Após algumas conversas e alinhamentos a equipe técnica chegou à definição da pontuação que cada uma das 2 últimas histórias da Sprint irá possuir.

- Concluíram que a história C é 2 vezes mais complexa que a história B, portanto a história C será definida como tendo 2 pontos.
- Definiram que a história A é 4 vezes mais complexa que a história B, mas como a sequência de Fibonacci não possui o número 4, o valor da pontuação é aproximado para o número posterior ao 4 presente na sequência, que neste caso é o número 5.

Portanto, temos as seguintes pontuações para as histórias presentes na Sprint 1 (Figura 9). Concluímos que a Sprint desenvolvida irá entregar ao todo 8 pontos ao seu término.

Figura 9 – Pontuação da sprint 1

História de usuário	Pontos da História
História A: Cadastrar Usuário	5
História B: Visualizar usuário	1
História C: Atualizar usuário	2
Total de pontos	**8**

Fonte: Próprio autor.

Tendo em mãos a definição da pontuação de cada história presente na Sprint, passamos para a fase de desenvolvimento. Porém, antes disso, apresentaremos como as metodologias e ferramentas trabalham juntas neste processo de criação, atuando de forma rápida e prática na construção da plataforma Help Me.

4.3 Metodologias e ferramentas atuando em conjunto

Metodologias de desenvolvimento de software certamente são muito úteis na construção de uma aplicação. Porém, elas se tornam ainda mais efetivas se forem utilizadas em conformidade com boas ferramentas que apoiam a criação de softwares.

A evolução tecnológica culminou no surgimento de diversos materiais e conteúdos que apoiam o desenvolvimento de aplicações cada vez mais robustas e com maior desempenho. O surgimento de metodologias, ferramentas e linguagens de programação de alto nível, fizeram com que a programação pudesse ser acessível e compreensível a várias pessoas, pois a complexidade antes presente em algumas linguagens foram camufladas. Esta nova visão tornou possível que muitas pessoas, inclusive crianças, adquiram conhecimentos em programação

de forma divertida e descontraída, porque estas curvas de aprendizado, antes existentes, agora foram reduzidas.

Usar boas ferramentas e tecnologias, em conformidade com metodologias de desenvolvimento, é um fator determinante para o sucesso de um projeto e consequentemente da empresa associada a ele. Empresas que atualmente possuem os famosos **legados**, que são sistemas com uma arquitetura antiga ou criado com ferramentas e tecnologias antigas, estão buscando, em muitos casos, atualizá-los. A atualização de legados é uma forma de mantê-los funcionais e corrigir possíveis problemas que surgem quando já estão em produção sendo usados pelo cliente.

A correção de problemas em sistemas que já estão em produção é chamada **sustentação**. A sustentação ocorre com muita frequência nos projetos em que versões de utilização já foram lançadas para o cliente. Ela é realizada à medida que os clientes percebem que algo deveria ter sido feito de outra forma, ou que algo desenvolvido não está funcionando da forma correta.

Estes custos podem ser muito altos para uma organização, pois é levado em consideração que as tecnologias antigas não dispõem dos mesmos recursos e das mesmas praticidades que as atuais no mercado. Esta diferença de versões e funcionalidades tornam algumas sustentações mais complexas do que realmente deveriam ser.

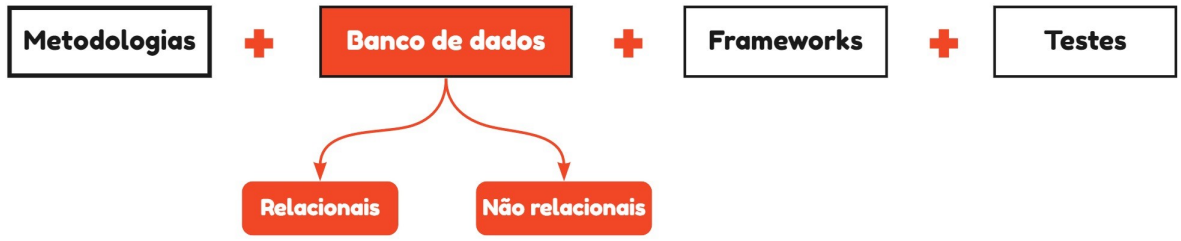
Empresas de **Tecnologia de Informação (TI)** e desenvolvedores, possuem esta visão de que boas ferramentas geram bons resultados e isso pode ser visto no surgimento diário de novas linguagens de programação, bibliotecas e ambientes de desenvolvimento. Todos esses novos recursos e ferramentas que surgem vem com o objetivo de aumentar sempre a praticidade, agregando novas funções na criação de projetos com apoio das metodologias de desenvolvimento de software.

Contudo, mesmo as novas tecnologias gerando facilidades, dificilmente imagináveis pelos primeiros programadores de software, surgiram formas de otimizar ainda mais seu desempenho. Esta otimização se dá através dos famosos **frameworks** que trabalham em conformidade com os **bancos de dados** gerando resultados incríveis para uma aplicação. Os capítulos 5 e 6 vão apresentar detalhes do banco de dados e frameworks que serão utilizados na criação do módulo da plataforma Help Me.

5 BANCO DE DADOS

Seguindo a estrutura dos recursos utilizados no desenvolvimento de software sabemos que uma aplicação que trabalha com armazenamento de dados requer um banco de dados. Os bancos de dados das aplicações são meios pelos quais os dados serão armazenados e posteriormente poderão, se necessário, serem visualizados, atualizados ou até mesmo apagados/-removidos. A seguir será apresentado um pouco mais sobre os bancos de dados, é importante inicialmente saber que eles são divididos em duas categorias, os banco de dados relacionais e os não relacionais, ambas serão abordadas com mais detalhes nas seções 5.1 e 5.2 (Figura 10).

Figura 10 – Bancos de dados



Fonte: Próprio autor.

Podemos pensar em um exemplo associado a plataforma Help Me para exemplificar o uso de um banco de dados em uma aplicação. Na plataforma Help Me, os usuários são inseridos em uma tabela do banco de dados, que contém alguns atributos, como: Nome do usuário; Sobrenome; Telefone de contato; E-mail etc. Se em algum momento os dados do usuário precisarem de alguma alteração, por exemplo nas informações de contato, basta pesquisar por aquele usuário, modificar seus dados e atualizá-lo no banco de dados.

Associado a cada usuário estão os conteúdos que ele está disposto a ensinar e caso um usuário queira remover algum desses conteúdos presentes em sua grade de conhecimentos, basta efetuar uma busca pelo sistema e remover seu registro do banco de dados. Após o conteúdo ser apagado, caso o usuário queira adicionar outro em seu perfil, basta apenas preencher os atributos do cadastro e adicioná-lo ao banco de dados.

Com esse exemplo simples podemos ter uma ideia de como acontece a visualização, atualização, exclusão e criação de um registro em um banco de dados. Devemos levar em consideração também que o banco descrito pode seguir algumas abordagens para sua construção e, com isso, surge um impasse quando se fala da criação de banco de dados, que é sobre qual tipo utilizar.

Existem diversos tipos de bancos de dados relacionais e banco de dados não relacionais que apoiam no armazenamento de dados. Alguns bancos de dados dessas categorias amplamente utilizados podem ser citados, como: SQL Server; Oracle; MySQL; MongoDB; MariaDB; Redis. Eles possuem um grande destaque no mercado. Com estes questionamentos, como saberemos qual tipo de banco de dados melhor se adequará as necessidades da plataforma

Help Me? Qual deles poderia atender e ser melhor para o negócio? Estas perguntas serão respondidas nas seções seguintes.

5.1 Banco de dados relacional

Banco de dados relacional são bancos onde os dados estão presentes em **Tabelas** que fazem relacionamentos entre si (ORACLE, 2020). As colunas representam os **Atributos** e as linhas, ou **Tuplas** (linhas), são registros incluídos nestas tabelas. Como bancos de dados relacionais podemos citar o SQL Server, o MySQL e o Oracle.

Os bancos de dados relacionais possuem associações entre si que na maioria das vezes impedem que alterações possam ser feitas depois que a estrutura do banco é criada, o que o torna pouco flexível. Porém, esta falta de flexibilidade é recompensada em sua consistência, pois os bancos de dados relacionais são muito confiáveis. Esta confiança vem da garantia de que as informações inseridas nas tabelas estejam corretas e sejam rapidamente atualizadas em qualquer outra instância daquele mesmo banco (ORACLE, 2020).

Um exemplo prático pode ser apresentado para que o entendimento de banco de dados relacional se torne ainda mais claro. Para seguir o mesmo raciocínio visto anteriormente, será simulada a tabela de usuários da plataforma Help Me. No exemplo, foram levantados alguns atributos que irão compor as colunas da tabela no banco. A seguir serão apresentados novamente os atributos com os nomes que serão associados a eles na tabela (Figura 11).

Nome da tabela

- usuario

Figura 11 – Atributos da tabela

Nome do usuário	nome
Sobrenome do usuário	sobrenome
E-mail do usuário	email
Celular do usuário	celular
Data de nascimento	dt_nascimento

Fonte: Próprio autor.

Vale ressaltar que a tabela deve possuir um identificador único (Id) que esteja associado a uma única Tupla (linha). Esta identificação é muito importante para qualquer operação realizada na tabela, que terá como referência este identificador nas transações do banco. Portanto, com a inserção do Id teremos a seguinte estrutura na tabela (Figura 12).

Figura 12 – Atributos da tabela

Identificador único do usuário	id_usuario
Nome do usuário	nome
Sobrenome do usuário	sobrenome
E-mail do usuário	email
Celular do usuário	celular
Data de nascimento	dt_nascimento

Fonte: Próprio autor.

Desta forma, teremos a seguinte tabela no banco de dados relacional (Figura 13).

Figura 13 – Atributos iniciais da tabela "usuario"

id_usuario	nome	sobrenome	email	celular	dt_nascimento

Fonte: Próprio autor.

Após a criação da tabela pode-se inserir um usuário, como é mostrado na Figura 14.

Figura 14 – Registro na tabela "usuario"

id_usuario	nome	sobrenome	email	celular	dt_nascimento
1	Paulo	Andrade	pauloandrade@email.com	(99) 99999-9999	07/09/2000

Fonte: Próprio autor.

Dessa forma, percebe-se, que na Tupla onde o id_usuario é igual a 1 temos os seguintes dados:

- Nome do usuário: **Paulo**
- Sobrenome do usuário: **Andrade**
- E-mail do usuário: **pauloandrade@email.com**
- Celular do usuário: **(99) 99999-9999**
- Data de nascimento do usuário: **07/09/2000**

A organização dos dados em tabelas facilita sua visualização, tornando a vida dos desenvolvedores melhor ao estabelecer contato com esses dados. Segundo a Oracle (2020), os bancos de dados relacionais possuem um grande poder, sendo aplicados em grandes ou pequenos projetos, atendendo as necessidades de diversas empresas e é isso que o torna tão poderoso.

5.2 Banco de dados não relacional

Os bancos de dados não relacionais, ou **NoSQL**, vêm com objetivo de dar uma maior flexibilidade nas operações que antes eram realizadas por meio dos bancos de dados relacionais. Com a evolução dos tipos de dados trafegados na Internet, foi surgindo a necessidade de um banco mais flexível e que abrangesse ainda mais dados que os bancos de dados relacionais. A necessidade de velocidade e uma disponibilidade frequente do banco de dados de um sistema, fez com que surgisse um novo tipo de banco, o não relacional (ROCKENBACH *et al.*, 2018). Como exemplo de banco de dados não relacionais podemos citar o MongoDB e o Redis.

Sua estrutura flexível é muito útil e se torna ainda mais prática quando há incertezas em relação a modelagem do banco de uma nova aplicação. Por ser flexível, se em algum momento algum tipo de dado na base precisar ser alterado a modificação não irá gerar um grande impacto no restante do banco, o que não é uma realidade nos bancos de dados relacionais onde as tabelas ficam muito atreladas entre si.

Uma das categorias de bancos de dados não relacionais mais conhecidas são os **Orientados a Documentos** (ROCKENBACH *et al.*, 2018). Para informações serem armazenadas no banco de dados Orientado a Documentos, o documento com os atributos e valores deve ser configurado em um formato como o **JSON** (JavaScript Object Notation, ou Notação de Objetos JavaScript) (JSON, 2020).

A fim de ilustrar ainda mais o conceito de banco de dados não relacional Orientado a Documentos, o mesmo exemplo de armazenamento de dados no banco SQL feito anteriormente será modificado para um documento que poderá ser inserido em um banco NoSQL. Após as adaptações necessárias, o resultado obtido se encontra na Figura 15.

Figura 15 – Documento com dados do usuário

```
{
  "id_usuario": 1,
  "nome": "Paulo",
  "sobrenome": "Andrade",
  "email": "pauloandrade@email.com",
  "celular": "(99) 99999-9999",
  "dt_nascimento": "07/09/2000"
}
```

Fonte: Próprio autor.

Os documentos semelhantes inseridos em um banco não relacional são contidos em coleções e, dessa forma, possuem uma ligação entre si. Como alternativa para auxiliar neste processo de armazenamento de dados em documentos existe o **MongoDB**. O MongoDB é um banco de dados distribuído, que tem como base o padrão da Orientação a Documentos (MONGODB, 2020).

Os bancos de dados NoSQL estão sendo adotados em vários contextos, porém os bancos de dados relacionais não estão sendo deixados de lado, ambos são utilizados em várias empresas, tanto em projetos grandes, quanto nos menores.

5.3 Utilizando banco de dados no projeto Help Me

Na seção 4.2.2 um Épico e uma Feature da plataforma Help Me foram definidos e, com isso, surgiram as histórias de usuário que entraram na primeira Sprint da aplicação. A partir das histórias criadas os desenvolvedores determinaram quais tarefas devem ser realizadas para que as histórias possam ser concluídas. Sendo assim, a Sprint poderá ser iniciada começando pela utilização do banco de dados.

Com a definição de alguns tipos de bancos de dados, passamos agora para utilização deste recurso na plataforma Help Me. Para isso, Arthur e o time de desenvolvedores devem definir como será a estrutura da base de dados do projeto. Após conversarem chegaram à conclusão que utilizarão um banco de dados relacional, devido e sua estrutura lógica e visivelmente clara que facilita a criação da modelagem dos dados. Desta forma, a aplicação Help Me será desenvolvida com apoio do SQL Server garantindo a otimização no uso do banco de dados.

Após a definição do tipo de banco de dados que será utilizado para o desenvolvimento da plataforma Help Me, começa agora o ciclo de desenvolvimento da primeira Sprint da aplicação. Essa Sprint entregará como resultado a criação de um módulo da plataforma. Para que esse módulo seja gerado, as histórias de usuários apresentadas na Figura 16 deverão ser atendidas. Relembrando que as pontuações das histórias apresentadas na Figura 16 já foram levantadas com apoio do Planning Poker na seção 4.2.2.2.1.

Figura 16 – Histórias presentes na Sprint 1

História de usuário	Pontos da História
História A: Cadastrar Usuário	5
História B: Visualizar usuário	1
História C: Atualizar usuário	2
Total de pontos	**8**

Fonte: Próprio autor.

O time de desenvolvedores liderados por Daniel decidiu estimar a pontuação das histórias, levando em consideração, o risco do uso das tecnologias que escolheram para o desen-

volvimento. Nessa etapa é necessário estimar os possíveis problemas que possam ocorrer e que consequentemente acabe atrasando a entrega da Sprint 1.

O time de desenvolvedores da empresa responsável pela criação da plataforma Help Me é composto por 4 **DEVs** (termo que se refere a desenvolvedores de software), incluindo Daniel, o desenvolvedor líder. A equipe definiu que cada DEV atuará em 1 história e que Daniel ficará como suporte para possíveis dificuldades e definições técnicas que os demais tenham no decorrer da Sprint.

Os 3 DEVs restantes são Douglas, Fernanda e Bruno. O time de desenvolvedores acordou a seguinte divisão de histórias para cada um:

- **Douglas:** História A
- **Fernanda:** História B
- **Bruno:** História C

Vale ressaltar que, o desenvolvimento deste módulo seguirá uma abordagem simples, a fim de se tornar mais claro para o leitor, levando em consideração o processo Scrum aplicado na criação do módulo da plataforma Help Me. Após as definições tem-se início o desenvolvimento das histórias da Sprint 1.

5.3.1 História A

Como definido anteriormente pelo time, Douglas irá desenvolver a história A que foi apresentada na seção 4.2.2.2, sua descrição é:

- *Eu, como usuário(a), desejo me cadastrar na plataforma para ter acesso a outros usuários e trocar conhecimentos.*

A história citada conta com as seguintes *tasks* (tarefas) associadas a ela:

1. Criar banco de dados e modelagem da tabela que será utilizada.
2. Criar projeto no ambiente de desenvolvimento.
3. Criar tela de cadastro do usuário.
4. Criar conexão com o banco de dados.
5. Validar se campos obrigatórios foram preenchidos.
6. Inserir informações do cadastro na base de dados.

Inicialmente, Douglas irá trabalhar na primeira *task* que é a modelagem da tabela no banco de dados. A tabela modelada também será utilizada nas demais histórias mapeadas para a Sprint em questão, portanto, ela tem maior prioridade perante as outras. Após a modelagem, Douglas desenvolverá suas demais *tasks* que utilizaram os frameworks *front-end* e *back-end* que serão definidos no capítulo 6. Com a conclusão de todas as suas *tasks* Douglas chegará ao fim de sua história.

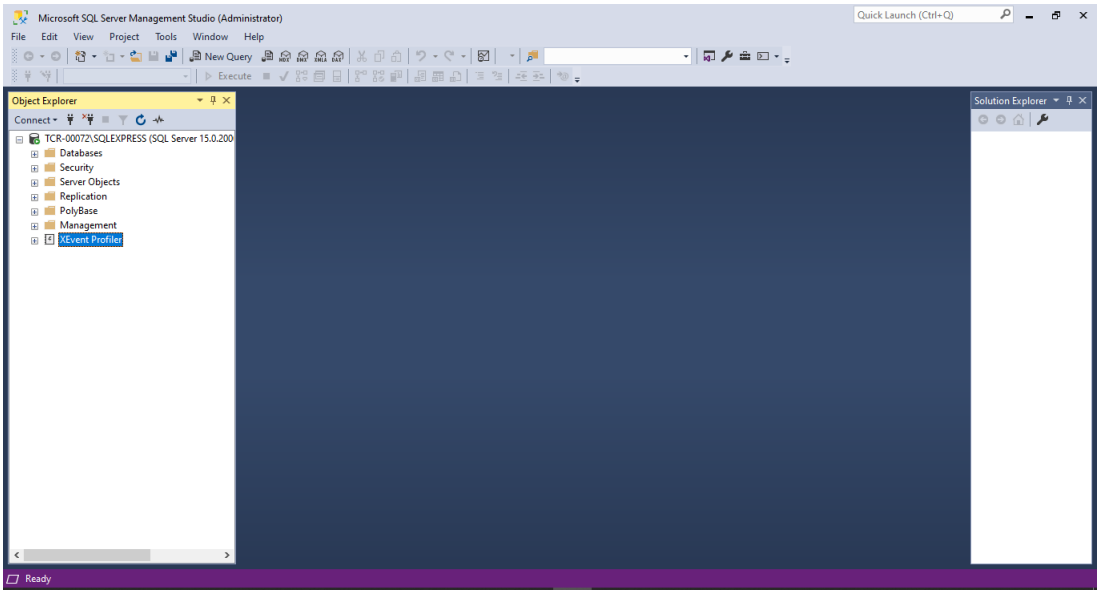
5.3.1.1 Criar banco de dados e modelagem da tabela que será utilizada

O time em conjunto com o arquiteto definiu que o banco de dados usado na plataforma seria um banco relacional. Portando, a primeira coisa a se fazer neste caso é escolher um SGBD (Sistema de Gerenciamento de Banco de Dados) que atenda esta demanda. Para suprir esta necessidade o time escolheu usar o SQL Server e a ferramenta SQL Server Management Stu-

dio (SSMS), disponibilizado pela Microsoft, que auxilia no gerenciamento de bancos de dados SQL Server (MICROSOFT, 2020a).

Após instalar e acessar a ferramenta Douglas se depara com a tela inicial do SQL Server Management Studio (Figura 17).

Figura 17 – Visão inicial do SQL Server Management Studio



Fonte: Próprio autor.

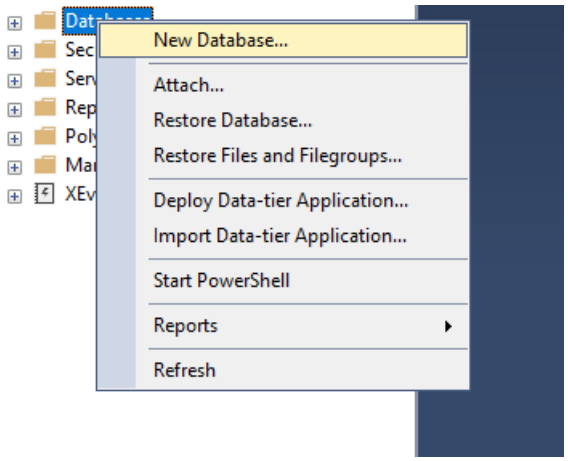
Com a ferramenta pronta para ser utilizada, Douglas começa a seguir os passos para a criação do banco de dados e da tabela que irá receber os dados de cadastro do usuário.

5.3.1.1.1 Criando o banco de dados

Inicialmente, Douglas e o time técnico compostos pelos DEVs e o arquiteto definem que o banco de dados irá ser nomeado como **“helpme”** e para criá-lo uma sequência de passos é seguida por Douglas.

O primeiro passo é clicar como o botão direito do mouse em “Databases” na lateral esquerda do SQL Server Management Studio e depois clicar em “New Database”, como é mostrado na Figura 18.

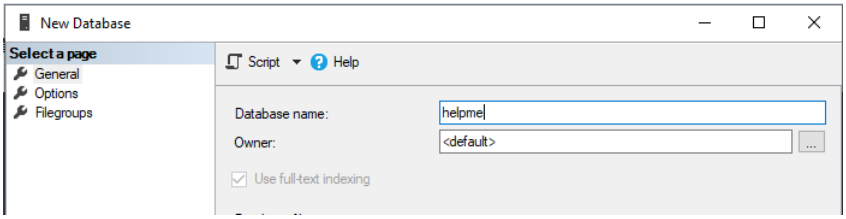
Figura 18 – Escolhendo opção de criação do banco de dados



Fonte: Próprio autor.

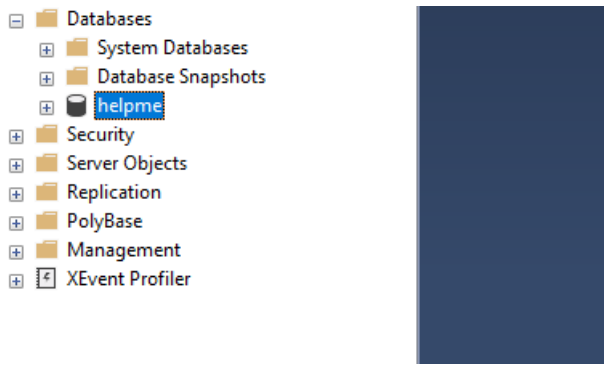
O próximo passo é nomear o banco de dados que na abordagem apresentada será "helpme" (Figura 19). Em seguida basta clicar em "OK" e o banco será criado. A Figura 20 mostra o banco de dados criado ao final de todo o processo feito por Douglas. Após criar o banco de dados, Douglas define qual será a modelagem inicial da tabela que irá armazenar as informações dos usuários na realização do cadastro.

Figura 19 – Nomeando o banco de dados



Fonte: Próprio autor.

Figura 20 – Visualização do banco de dados criado

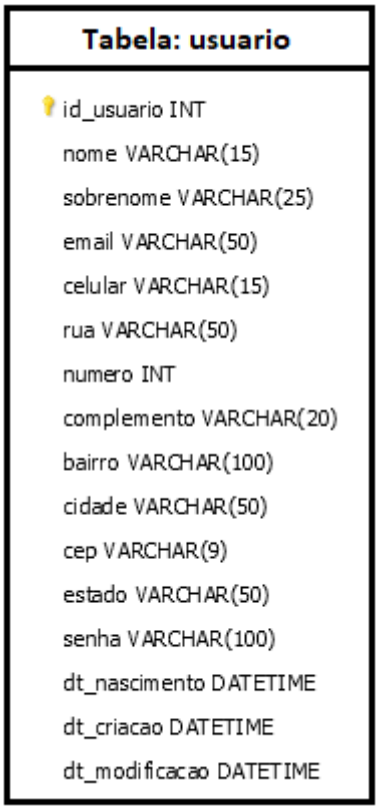


Fonte: Próprio autor.

5.3.1.1.2 Criando tabela para armazenar as informações do usuário no banco de dados

Após reunião da equipe de desenvolvimento, juntamente com Arthur, o Arquiteto de Software, definiram que a tabela que armazenará os dados dos usuários será nomeada como **"usuario"** e terá a seguinte estrutura apresentada no diagrama da Figura 21. Com base no diagrama, Douglas determina quais atributos da tabela obrigatoriamente deverão ser preenchidos (Figura 22).

Figura 21 – Diagrama da tabela usuario



Fonte: Próprio autor.

Figura 22 – Modelagem da tabela para armazenar os dados dos usuários

```
id_usuario INT IDENTITY,
nome VARCHAR(15) NOT NULL,
sobrenome VARCHAR(25) NOT NULL,
email VARCHAR(50) NOT NULL,
celular VARCHAR(15) NOT NULL,
rua VARCHAR(50) NOT NULL,
numero INT NOT NULL,
complemento VARCHAR(20) NULL,
bairro VARCHAR(100) NOT NULL,
cidade VARCHAR(50) NOT NULL,
cep VARCHAR(9) NOT NULL,
estado VARCHAR(50) NOT NULL,
senha VARCHAR(100) NOT NULL,
dt_nascimento DATETIME NOT NULL,
dt_criacao DATETIME NOT NULL,
dt_modificacao DATETIME NOT NULL
CONSTRAINT pk_usuario PRIMARY KEY (id_usuario)
```

Fonte: Próprio autor.

Nesta modelagem, percebemos que o identificador único do usuário é definido como "id_usuario". Com base nele, cada usuário poderá ser reconhecido unicamente na tabela. O id-usuário é a **chave primária** (ou Primary Key) da tabela em questão. A chave primária de uma tabela em um banco de dados relacional gera a possibilidade de estabelecer relacionamentos entre outras tabelas por meio desse campo. Assim, as chaves primárias têm uma grande importância na modelagem da tabela que irá armazenar as informações referentes ao usuário cadastrado na plataforma.

O campo id_usuario terá um auto incremento automático, ou seja, cada novo registro na tabela de usuário irá gerar um novo valor para o campo id_usuário de forma crescente, sendo que, este valor nunca irá se repetir. Além do campo id_usuario, a tabela conta com outros atributos relevantes para o armazenamento de dados dos usuários. Vale ressaltar também que apenas o campo "complemento", mapeado na tabela, que poderá receber um valor **NULL** (Nulo ou Vazio), isso implica que os demais campos, exceto ele, devem ter dados armazenados.

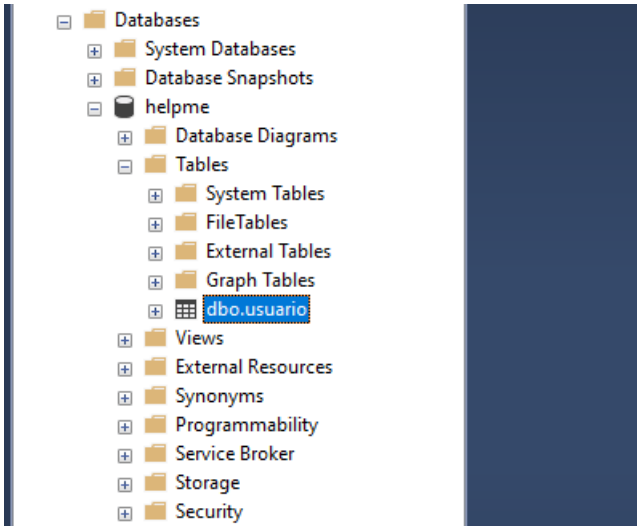
Após estas considerações, basta que uma simples operação de criação seja feita por Douglas para que a tabela seja construída com os campos e especificações definidos por ele e validados pelo time. O **script** para a geração da tabela pode ser observado na Figura 23. Com a execução do *script* a seguinte tabela é criada no banco de dados (Figura 24).

Figura 23 – *Script* de criação da tabela

```
CREATE TABLE usuario(
    id_usuario INT IDENTITY,
    nome VARCHAR(15) NOT NULL,
    sobrenome VARCHAR(25) NOT NULL,
    email VARCHAR(50) NOT NULL,
    celular VARCHAR(15) NOT NULL,
    rua VARCHAR(50) NOT NULL,
    numero INT NOT NULL,
    complemento VARCHAR(20) NULL,
    bairro VARCHAR(100) NOT NULL,
    cidade VARCHAR(50) NOT NULL,
    cep VARCHAR(9) NOT NULL,
    estado VARCHAR(50) NOT NULL,
    senha VARCHAR(100) NOT NULL,
    dt_nascimento DATETIME NOT NULL,
    dt_criacao DATETIME NOT NULL,
    dt_modificacao DATETIME NOT NULL
    CONSTRAINT pk_usuario PRIMARY KEY (id_usuario)
 )
 go
```

Fonte: Próprio autor.

Figura 24 – Tabela "usuario" criada no banco helpme



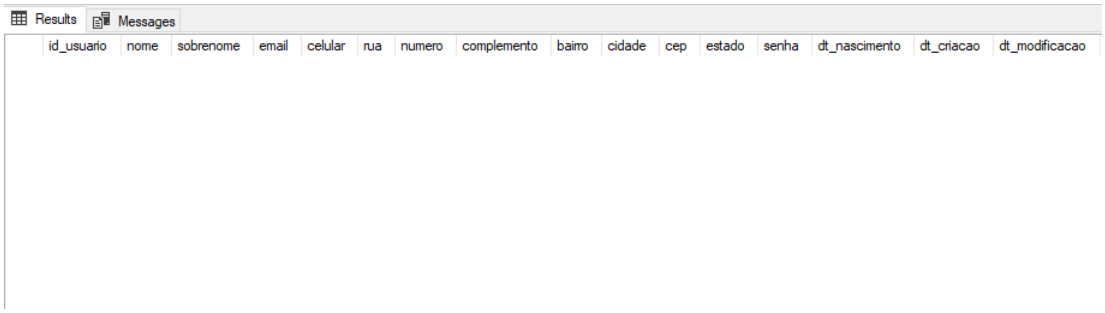
Fonte: Próprio autor.

Para visualizar a tabela Douglas executa uma **consulta (query)** com a seguinte instrução:

```
1                    SELECT * FROM usuario
2
```

Ou seja, selecione todos os registros contidos na tabela "usuario". O resultado da consulta é apresentado na Figura 25.

Figura 25 – Resultado da consulta na tabela de usuários



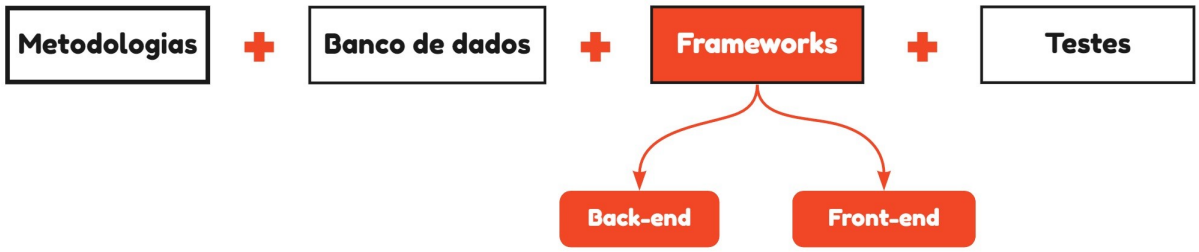
Fonte: Próprio autor.

Dessa forma, com a criação do banco de dados e da tabela que irá armazenar os dados dos usuários, Douglas poderá passar para as próximas *tasks* de sua história. Como as *tasks* restantes envolvem o uso de frameworks, no capítulo 6 os frameworks serão apresentados e em seguida Douglas concluirá sua história. Também no capítulo 6, Fernanda e Bruno realizarão o desenvolvimento das histórias que ficaram responsáveis.

6 FRAMEWORKS

Frameworks são uma estrutura genérica que possibilitam o reuso de classes e objetos para criação de aplicações semelhantes. Portanto, frameworks apoiam os programadores entregando a eles, em alguns casos, funcionalidades que precisam apenas de pequenas alterações para se adaptarem às suas necessidades em um determinado contexto, com uma determinada linguagem. A seguir, um pouco mais sobre os frameworks será apresentado, é importante inicialmente saber que eles são divididos em duas categorias, os ***frameworks back-end*** e os ***frameworks front-end*** (Figura 26).

Figura 26 – Frameworks



Fonte: Próprio autor.

Pode-se entender um framework com o exemplo da construção de um quebra cabeça, onde um conjunto de peças são encaixadas dentro de um formato limitado por laterais. Dessa forma, percebemos que os frameworks também possuem limitações, ou seja, eles atuam apenas em seus respectivos escopos, em suas "caixinhas". Um framework não consegue fazer tudo em todas as aplicações, da mesma forma que é impossível as peças de um quebra cabeça que montam a imagem de uma árvore montarem também a imagem de um cachorro.

Atualmente, a maior parte dos grandes projetos contam com o apoio de frameworks, isso pode ser percebido devido à grande popularidade que eles vêm exercendo no mundo do desenvolvimento Web e também em outros contextos. Esta popularidade se dá ao fato dos frameworks já definirem um esqueleto que a aplicação irá possuir, agilizando assim o trabalho dos desenvolvedores ligados a sua construção (SOMMERVILLE, 2011).

Como um dos objetivos das empresas de TI é desenvolver de forma ágil a demanda de seus clientes, o uso de frameworks é sempre uma boa opção para construção rápida de funcionalidades de um sistema. Os clientes prezam muito pela agilidade na entrega de suas soluções tecnológicas e os frameworks possibilitam esta agilidade visando a reutilização e agregando ainda mais valor com a redução no tempo de entrega do produto.

Com base nas características levantadas sobre os frameworks, a seguir, será apresentada a definição e exemplificação de *frameworks back-end* e *frameworks front-end*. A Figura 27 mostra um exemplo da diferença entre o *back-end* e o *front-end* de uma aplicação. Através da figura percebemos que o *front-end* é apenas o que esta visível ao usuário (como as telas do software), já o *back-end* não é visível ao usuário (como uma conexão com o banco de dados).

Figura 27 – Back-end vs Front-end

Fonte: (MEDIUM, 2020). Adaptado.

Após estas definições um framework de cada especificidade será utilizado na criação do módulo de usuário da aplicação Help Me.

6.1 Frameworks Back-end

Os frameworks *back-end* estão presentes no mundo do desenvolvimento para auxiliar os DEVs na implementação das regras estruturais que constituem o projeto. Uma aplicação Web, na maioria dos casos, possuirá um banco de dados. Uma das responsabilidades do *back-end* é estabelecer esta ligação e trabalhar com os dados presentes na aplicação, como por exemplo, armazenando as informações em um banco de dados.

Podemos dizer que o *back-end* está por trás da construção da lógica e fluxos da aplicação. Fluxo, neste caso, pode ser todos os passos para que uma funcionalidade presente em um sistema seja executada como, por exemplo: A integração entre um fornecedor de cimento com o banco de dados de uma empresa que trabalha com materiais de construção.

- Neste caso, o fluxo seria a conexão com o banco de dados da empresa de materiais de construção. Esta conexão pode ser feita com o apoio de uma **API** (Application Program Interface, ou Interface de Programação de Aplicações) que fornece uma interface que possibilita uma troca mais rápida e fácil de dados entre aplicações distintas sejam elas internas ou externas a uma organização. Neste contexto, a API seria responsável por trafegar os dados entre a empresa de materiais de construção e a empresa que fornece cimento.

O exemplo acima representa de forma simples e genérica como seria um fluxo onde frameworks *back-end* poderiam ser empregados para solução do problema. Entretanto, isso não quer dizer que um framework resolve qualquer demanda, muito pelo contrário, saber qual framework *back-end* usar em cada contexto pode ser um fator determinante para o sucesso de um projeto.

Dentre os framewowrks *back-end* 3 deles são bem conhecidos e difundidos no mercado: .NET Core, Django e Express.

O interessante dos 3 frameworks mencionados é que eles possuem linguagens de programação distintas e também podem ser empregados em ambientes distintos por serem multiplataformas. Segundo Santos (2018), aplicações multiplataformas são práticas devido ao fato de um único código fonte ser executado igualmente em diferentes máquinas, com diferentes sistemas operacionais.

6.1.1 .NET Core

O .NET Core é um framewoek *back-end* desenvolvido pela Microsoft que utiliza a linguagem C Sharp e está voltado para o desenvolvimento de aplicações multiplataformas. Além disso, o .NET Core também é uma plataforma de desenvolvimento de **código aberto** (MICROSOFT, 2020b).

A iniciativa do código aberto surgiu como um movimento para melhoria dos códigos desenvolvidos em uma aplicação. Nesta iniciativa, o código fonte de uma aplicação é compartilhado com a comunidade de desenvolvedores de software e, com isso, vários colaborados tem a possibilidade de contribuir em sua implementação (INITIATIVE, 2020). É muito importante ferramentas possuírem código aberto, pois isso possibilita sua evolução contínua, onde muitas pessoas diferentes podem contribuir para sua manutenção, como é o caso das distribuições Linux, sendo todas **open source** (código aberto).

Outra plataforma de desenvolvimento da Microsoft é o .NET framework, anterior ao .NET Core. Segundo a Microsoft (2020e), há diferenças entre as duas implementações de .NET. Estas diferenças devem ser levadas em consideração na hora de optar pelo uso de algum dos frameworks em um projeto. A seguir, serão apresentadas algumas recomendações para os contextos em que os dois frameworks podem ser usados.

O uso do .NET Core é recomendado se:

- O desenvolvimento for acontecer em plataformas diversas.
- O sistema que será desenvolvido precisa de um alto desempenho.

Estes são apenas 2 dos pontos vantajosos em se utilizar o .NET Core. Agora, serão apresentados também 2 pontos em que é recomendado o uso do .NET Framework:

- A aplicação já existe e usa o .NET Framework.
- Alguns recursos que são utilizados na aplicação desenvolvida com uso do .NET Framework estão indisponíveis no .NET Core.

Com as vantagens e desvantagens apresentadas percebemos que tanto o .NET Framework quanto o .NET Core são aplicados em projetos. Caberá à equipe técnica determinar qual das implementações de .NET atenderá as demandas de um cliente. Muitas organizações estão aderindo ao uso do .NET em todo o mundo, sendo uma boa opção para implementação de microsserviços.

6.1.2 Django

O Django é um framework Web que trabalha com a linguagem Python, muito conhecida e usada no mercado de TI devido a sua facilidade, versatilidade e grande conjunto de bibliotecas que atendem muito bem o desenvolvimento Web (DJANGO, 2020). A linguagem

Python, além de atuar na construção de aplicações Web também é muito usada na ciência de dados, proporcionando a grandes empresas resultados incríveis no que diz respeito a análise de seus dados organizacionais.

O Django chama muita atenção de pessoas iniciantes em frameworks e até mesmo em programação por não ser um framework muito complexo comparado às outras ferramentas e linguagens. Ele dispõe também de um conjunto de funcionalidades prontas e uma comunidade grande e colaborativa. Da mesma forma que o .NET Core, o Django também possui código aberto e pode ser instalado com o apoio do pip (Python Package Installer) o instalador de pacotes para Python (DJANGO, 2020).

6.1.3 Express

O javascript nos últimos anos vem adquirindo um poder enorme no mundo do desenvolvimento de aplicações, deixando de ser uma linguagem unicamente de interações na tela, partindo também para o *back-end*. Surge, então, o Express como um framework Web *back-end* para node.js.

O node.js surgiu em 2009 como um ambiente de execução de código javascript, possuindo também seu código aberto (NODE, 2020). Ele é utilizado em diversas empresas famosas em todo o mundo, como Uber e Netflix, caindo no gosto dos programadores e ficando entre as tecnologias de desenvolvimento mais adoradas. O Express é uma das principais ferramentas para desenvolver aplicações com uso do node.js. Seu objetivo é fazer uso do node.js no *back-end* de forma otimizada e prática, abstraindo muito bem o conceito de framework.

Para que o Express possa ser utilizado é preciso instalá-lo, isso é comum em todas as ferramentas que serão utilizadas em nossas máquinas. Para auxiliar no processo de instalação do Express existe o npm (Node Package Manager), que é o gerenciador de pacotes do node.js. O npm é um gerenciador de pacotes que facilita a obtenção de ferramentas e recursos, pois com apenas uma linha de código uma ferramenta pode ser facilmente instalada, como é o caso do Express. O npm é utilizado mundialmente por desenvolvedores para fazer o compartilhamento de pacotes (NPM, 2020).

6.2 Frameworks Front-end

Após conhecermos um pouco mais sobre os frameworks *back-end*, passamos agora para os frameworks *front-end* que são os responsáveis pela parte visual das aplicações.

Os frameworks *front-end* atuam na interface do usuário, tornando sua experiência mais agradável e intuitiva. Sabe aquela sua sensação marcante de navegação em um *site*, seja ele o Facebook, Youtube, Netflix? Pois bem, essas experiências só se tornam prazerosas se o usuário se sentir bem com o que vê quando tem contato com a aplicação. Este "se sentir bem com o que vê" é a missão que os frameworks *front-end* buscam atender, fazendo com que o usuário tenha uma experiencia diferenciada cada vez que entra em uma página de uma aplicação Web. Dentre os frameworks *front-end* podemos destacar o: Bootstrap, Materialize, Pure.

Para os exemplos apresentados utilizaremos como editor de código o Visual Studio Code (VS Code) da Microsoft. Ele possui diversas funcionalidades e tem a grande vantagem de ser leve quando instalado em uma máquina (MICROSOFT, 2020c).

6.2.1 Bootstrap

O Bootstrap é um framework *front-end* Web mundialmente conhecido e empregado no desenvolvimento de diversas aplicações. O Bootstrap ajuda na criação de *sites* responsivos, ou seja, as páginas de uma aplicação Web criadas por meio dele se adaptam ao tamanho da tela do dispositivo em que a aplicação é usada, dispensando o uso das barras de rolagem que tornam a visualização complicada para o usuário (BOOTSTRAP, 2020).

No próprio *site* do Bootstrap podemos encontrar a documentação que proporciona o desenvolvimento e uso de diversos elementos que já vem inclusos no css, HTML e javascript, como por exemplo a criação de um botão apresentado na Figura 28:

Figura 28 – Trecho de código para criar um botão com Bootstrap

```
<button type="button" class="btn btn-primary">Primary</button>
```

Fonte: Próprio autor.

A classe **"btn btn-primary"** já possui todas as configurações necessárias para tornar um botão interativo e visualmente atraente em uma página.

Podemos adicionar o Bootstrap a um projeto de duas formas:

- Fazendo o download da pasta e adicionando-a no projeto.
- Por meio da URL disponível no *site* para vinculá-lo ao projeto.

A opção de download da pasta do Bootstrap está presente em seu *site* e após o download a pasta gerada pode ser vinculada no cabeçalho (head) do código fonte, da seguinte forma (Figura 29).

Figura 29 – Vinculando pasta do Bootstrap ao projeto

```
<head>
    <link rel="stylesheet" href="bootstrap/dist/css/bootstrap.min.css" />
</head>
```

Fonte: Próprio autor.

A segunda forma de vincular o Bootstrap ao projeto é com o uso de sua URL, como mostra a Figura 30.

Figura 30 – Vinculando URL do Bootstrap ao projeto

```
<head>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
</head>
```

Fonte: Próprio autor.

Vale ressaltar que ao vincular o *link* de acesso ao Bootstrap no projeto, caso durante o desenvolvimento da aplicação você não tenha conexão com a Internet o Bootstrap não irá funcionar (BOOTSTRAP, 2020).

Como todo bom framework *front-end* o Bootstrap dispõe de inúmeras funcionalidades que agilizam o desenvolvimento da parte visual das aplicações. Na aba de "componentes" no próprio *site* do Bootstrap, um conjunto de exemplos prontos para uso já são oferecidos, isso facilita em muito a criação do *front-end* para o desenvolvedor e proporciona um visual agradável ao usuário.

Com o Bootstrap é possível fazer um teste simples de criação de um botão, com o mesmo código visto anteriormente. Na Figura 31 o Bootstrap está sendo inserido ao código por meio dos seus *links* de acesso disponíveis em seu *site*. Na Figura 32 o resultado gerado pela implementação é apresentado.

A estrutura do código é bem simples, primeiramente o Bootstrap é importado no cabeçalho (head) do arquivo, em seguida, no corpo (body) do arquivo HTML o botão é inserido.

Figura 31 – Código HTML para gerar um botão com Bootstrap

```
<html>

<head>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
        integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk" crossorigin="anonymous">
</head>

<body>
    <button type="button" class="btn btn-primary">Primary</button>
</body>

</html>
```

Fonte: Próprio autor.

Figura 32 – Resultado do botão gerado com Bootstrap

Fonte: Próprio autor.

6.2.2 Materialize

O Materialize foi criado com base no Material Design do Google e vem com o objetivo de unificar a experiência de design dos usuários de ferramentas do Google em qualquer plataforma que estiverem sendo executadas (MATERIALIZECSS, 2020).

O Materialize é um concorrente direto do Bootstrap, gerando inúmeras funcionalidades em javascript que agregam um diferencial às páginas Web. Os dois frameworks são concorrentes, pois possuem basicamente as mesmas opções de estilização de páginas, com componentes e recursos semelhantes. Sendo assim, grande parte das funcionalidades feitas com o Bootstrap também podem ser feitas com o Materialize e o contrário também é válido.

A adição do Materialize a um projeto é semelhante à do Bootstrap, sendo que, existe a opção de download de sua pasta, ou incremento do *link* no cabeçalho da aplicação. Da mesma forma do Bootstrap o Materialize dispõe de uma série de elementos que são de fácil uso e que geram resultados incríveis para as telas de uma aplicação.

Semelhante ao exemplo de criação de um botão apresentado anteriormente com uso do Bootstrap a Figura 33, apresenta um exemplo de criação de um botão com apoio do Materialize com seu *link* adicionado no cabeçalho e na Figura 34 poderá ser visualizado o resultado gerado referente a esta implementação.

Figura 33 – Código HTML para gerar um botão com Materialize

```
<html>

<head>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize.min.css">
</head>

<body>
    <a class="waves-effect waves-light btn">button</a>
</body>

</html>
```

Fonte: Próprio autor.

Figura 34 – Resultado do botão gerado com Materialize

Fonte: Próprio autor.

Existe uma grande semelhança entre as duas implementações, fazendo com que um desenvolvedor possa optar tranquilamente por uma delas. Ambas não possuem complexidade cognitiva alta, ou seja, não são difíceis de aprender.

6.2.3 Pure

Mesmo não sendo tão conhecido quanto o Bootstrap ou o Materialize, o Pure já vem sendo muito utilizado devido a sua facilidade e praticidade que agregam muito valor a um projeto. Por conta disso, um exemplo de seu uso também se faz importante dentre os frameworks *front-end*. A forma de integração do Pure ao projeto é semelhante às apresentadas anteriormente e o que chama muito a atenção para seu uso é a leveza que possui nas aplicações, executando de forma rápida e gerando um bom resultado para o usuário (PURE, 2020).

O Pure possui uma documentação bem completa e simples, por meio dela, da mesma forma como foram feitas anteriormente com o uso do *link* de conexão no cabeçalho, um botão pode ser personalizado rapidamente. Essa operação pode ser vista no código apresentado na Figura 35, que gera o resultado mostrado na Figura 36.

Figura 35 – Código HTML para gerar um botão com Pure

```
<html>

<head>
    <link rel="stylesheet" href="https://unpkg.com/purecss@2.0.3/build/pure-min.css"
        integrity="sha384-cg6SkqEOCV1NbJoCu11+bm0NvBRc8IYLRGXkmNrqUBfTjmMYwNKPWBTIKyw9mHNJ" crossorigin="anonymous">
</head>

<body>
    <button class="pure-button pure-button-primary">A Primary Button</button>
</body>

</html>
```

Fonte: Próprio autor.

Figura 36 – Resultado do botão gerado com Pure

Fonte: Próprio autor.

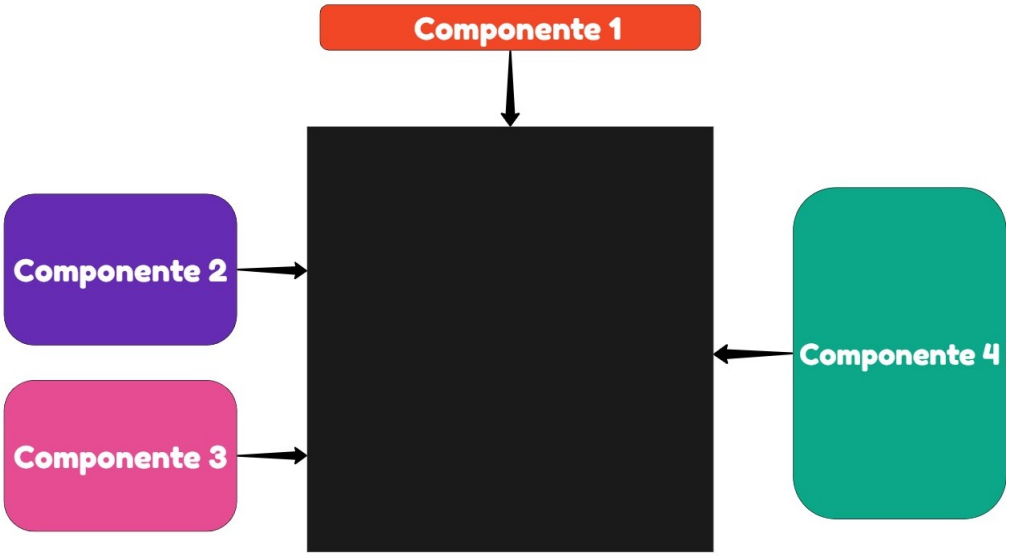
6.3 Microsserviços

Com apoio dos bancos de dados e dos frameworks podemos criar microsserviços referentes a uma aplicação. Um sistema desenvolvido nesta arquitetura é composto pela junção de diversos componentes que quando unidos compõem um grande sistema que realiza várias operações, eles são usados para dividir a complexidade de uma aplicação auxiliando no desempenho das operações realizadas.

Uma aplicação que trabalha com microsserviços é fácil de dar manutenção, pois cada componente tem suas responsabilidades separadas e isso auxilia na detecção de erros. Além disso, um software com microsserviços também é fácil de evoluir, ou seja, um novo componente com novas funcionalidades pode ser criado sem gerar impacto nos demais, visto que, os componentes são desacoplados e não possuem dependências entre si.

Na Figura 37 podemos observar como se dá a estrutura de uma aplicação composta por microsserviços. Cada componente possui suas responsabilidades e esses vários componentes juntos formam uma aplicação completa.

Figura 37 – Estrutura de uma aplicação composta por microsserviços

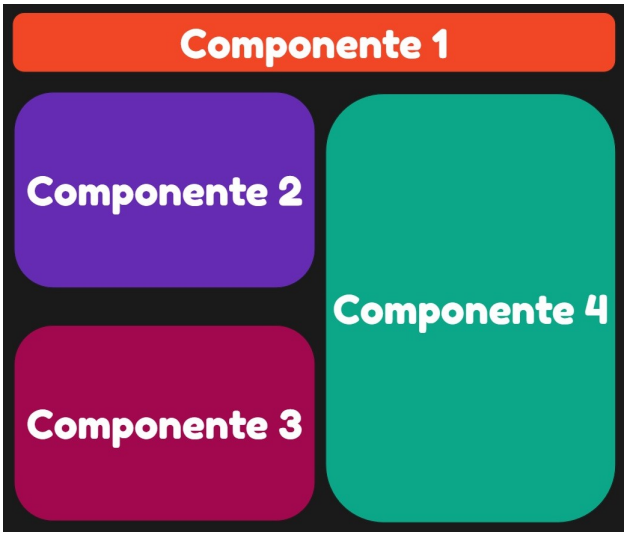


Fonte: Próprio autor.

A arquitetura de software em microsserviços vem sendo muito adotada na criação das novas aplicações, mas antes dessa abordagem se tornar comum os sistemas eram monolitos. Em um software monolito existe uma única aplicação que é responsável por todas as operações do sistema, neste caso o sistema como um todo não é dividido em componentes e acaba sendo um único bloco.

Sistemas monolitos dificultam sua manutenção e criação novas funcionalidades, pois nesta abordagem de desenvolvimento os componentes do sistema podem realizar mais de uma operação dentro do software, ou seja, um componente não possui uma responsabilidade única. A Figura 38 apresenta de forma mais clara como é a estrutura de um software em monolito, onde todos os componentes já são desenvolvidos dentro da própria aplicação.

Figura 38 – Estrutura de um software monólito



Fonte: Próprio autor.

Alguns projetos surgem como monolitos e depois são evoluídos para microsserviços e esta será a abordagem adotada na plataforma Help Me. Inicialmente a equipe contratada por Maria e Valter construirá o módulo de usuário e nas sprints futuras a aplicação será modificada para adotar microsserviços em sua estrutura.

6.4 Utilizando frameworks no projeto Help Me

Após estudar os frameworks *front-end* e *back-end*, o time de desenvolvedores liderado por Daniel, que é o desenvolvedor líder, juntamente do Arquiteto de Software Arthur, fizeram uma reunião para definir quais das tecnologias apresentadas serão utilizadas no desenvolvimento do módulo de usuário da plataforma Help Me.

Nesta reunião definiram que o framework *back-end* adequado para o contexto é o .NET Core, devido a sua versatilidade e desempenho. Perceberam que como será um sistema robusto futuramente, com muitos usuários cadastrados, o .NET Core será muito útil para garantir essa performance. Definiram também que para fazer um intermédio entre o *back-end* e o usuário,

o framework *front-end* que será utilizado no desenvolvimento do projeto será o Bootstrap, que gera designs responsivos atraentes para quem está do outro lado da tela.

Portanto, o *back-end* da plataforma Help Me será desenvolvido na linguagem C Sharp com apoio de seu framework .NET Core e no *front-end* será utilizado o Bootstrap. Dessa forma, Douglas e os demais desenvolvedores poderão dar continuidade em suas histórias.

6.4.1 Continuação da História A

Após a definição dos frameworks, Douglas retoma a história A que possui o seguinte título: "Cadastrar Usuário". E conta com as *tasks* definidas na seção 4.2.2.2:

1. Criar banco de dados e modelagem da tabela que será utilizada.
2. Criar projeto no ambiente de desenvolvimento.
3. Criar tela de cadastro do usuário.
4. Criar conexão com o banco de dados.
5. Validar se campos obrigatórios foram preenchidos.
6. Inserir informações do cadastro na base de dados.

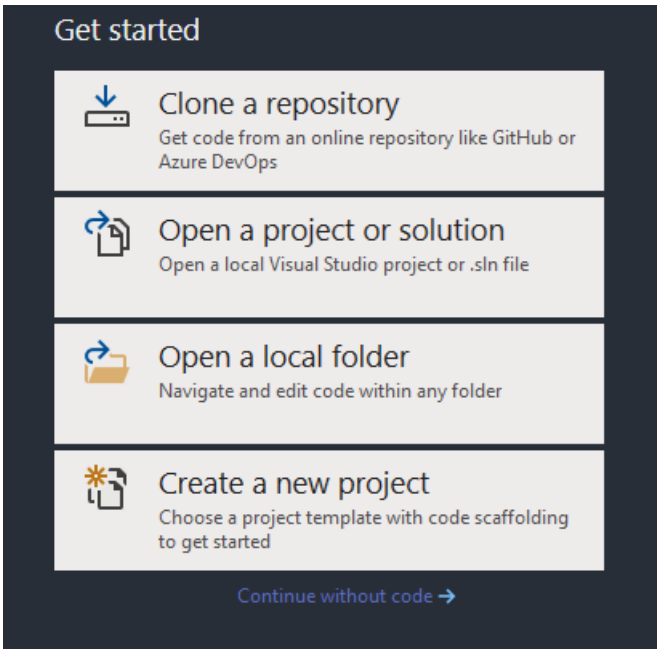
Como a primeira *task* já foi concluída, Douglas dá seguimento ao desenvolvimento das demais, passando agora para a *task* 2.

6.4.1.1 Criar projeto no ambiente de desenvolvimento

O framework *back-end* escolhido pela equipe técnica para criação do projeto foi o .NET Core, para utilizá-lo optaram pela adoção do Microsoft Visual Studio como IDE (Ambiente de Desenvolvimento Integrado). Um IDE ajuda no desenvolvimento de uma aplicação abstraindo complexidades que são imperceptíveis ao programador que trabalha com uma certa linguagem ou atua em um certo projeto. Os IDEs são ferramentas que auxiliam na criação de softwares dispondo de recursos para tornar mais ágil esse processo.

Após seu download e instalação, para criação de um novo projeto, os seguintes passos foram executados por Douglas. Inicialmente ele executa o Visual Studio e posteriormente seleciona a opção "Create a new project", como mostra a Figura 39.

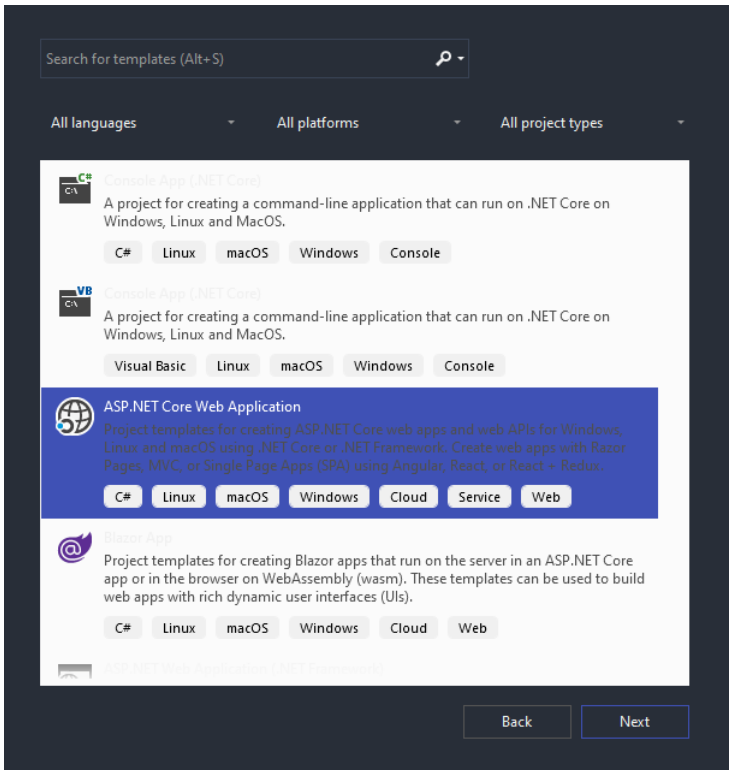
Figura 39 – Iniciando Visual Studio



Fonte: Próprio autor.

Após selecionar esta opção, Douglas define que a aplicação será Web e com base no .NET Core (Figura 40).

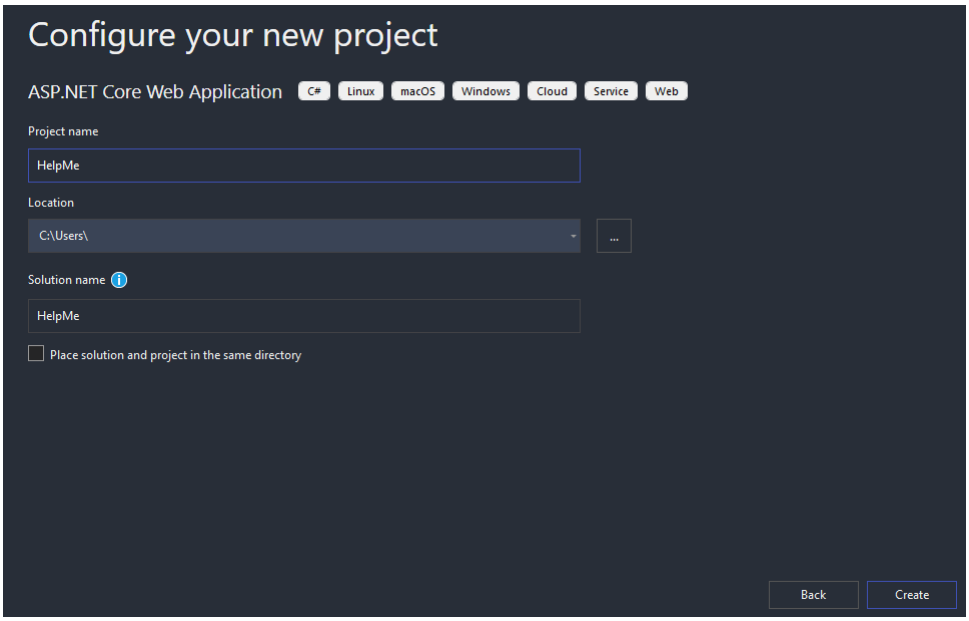
Figura 40 – Definindo que a aplicação será .NET Core



Fonte: Próprio autor.

Com a opção anterior selecionada o próximo passo é definir como a aplicação será nomeada. Com base em algumas conversas o time e o Arquiteto de Software Arthur definiram que o projeto será intitulado **"HelpMe"** (Figura 41).

Figura 41 – Definindo o nome da aplicação

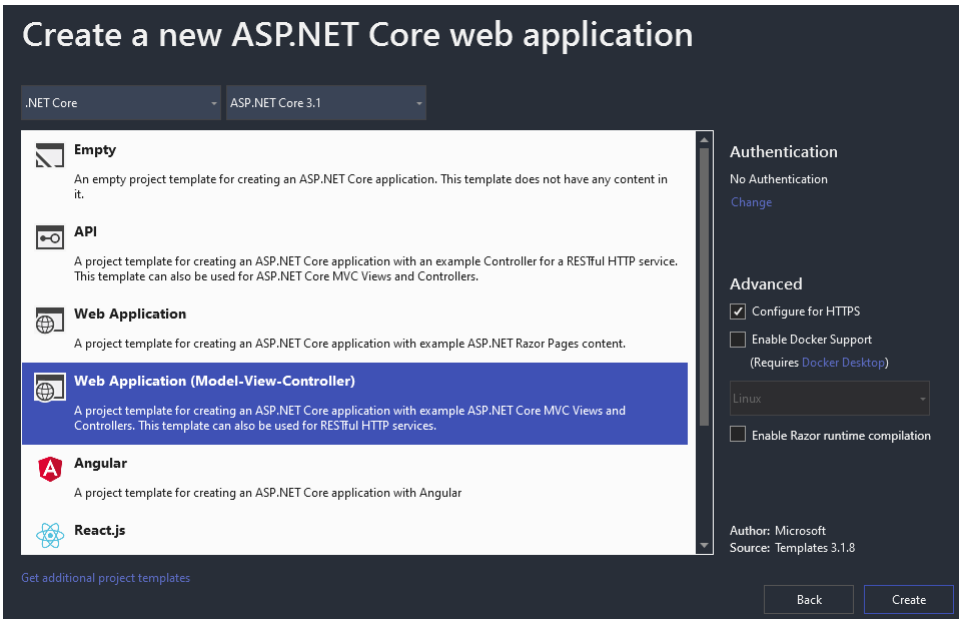


Fonte: Próprio autor.

O último passo é definir a arquitetura **MVC** (Model-View-Controller ou Modelo-Visão-Controle), como apresentado na Figura 42. Está arquitetura é útil pois separa os interesses presentes em uma aplicação. As telas, as operações envolvendo os dados e toda a lógica do sistema são divididas, gerando assim um maior desacoplamento e uma melhor organização por parte dos desenvolvedores (MICROSOFT, 2020f).

A arquitetura MVC tenta dividir as responsabilidades dentro de um software. O Model é o responsável pelo tratamento dos dados do software, a View fica encarregada da parte visual da aplicação (HTML, css, etc.) e o Controller faz o intermédio entre a View e o Model, recebendo os dados das telas da aplicação pela View e enviando para o Model que realiza as operações no banco de dados. Esta divisão faz com que o código desenvolvido seja mais facilmente compreendido e também possa ser reutilizado.

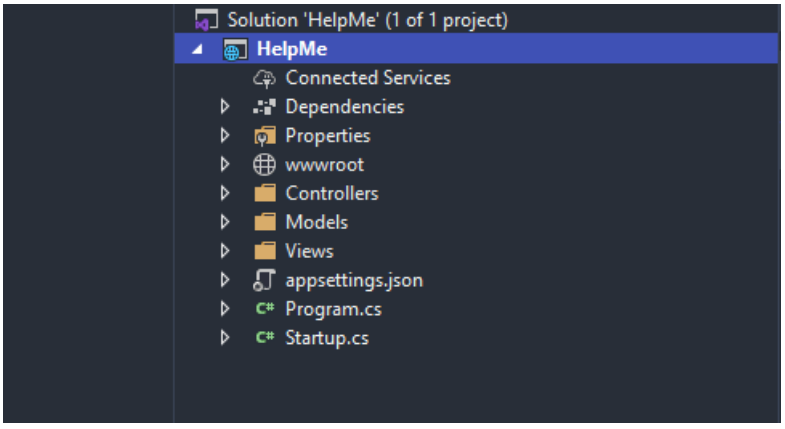
Figura 42 – Definindo padrão MVC para a aplicação



Fonte: Próprio autor.

Após todos os passos apresentados anteriormente o resultado final será uma estrutura pré definida como mostrado na Figura 43.

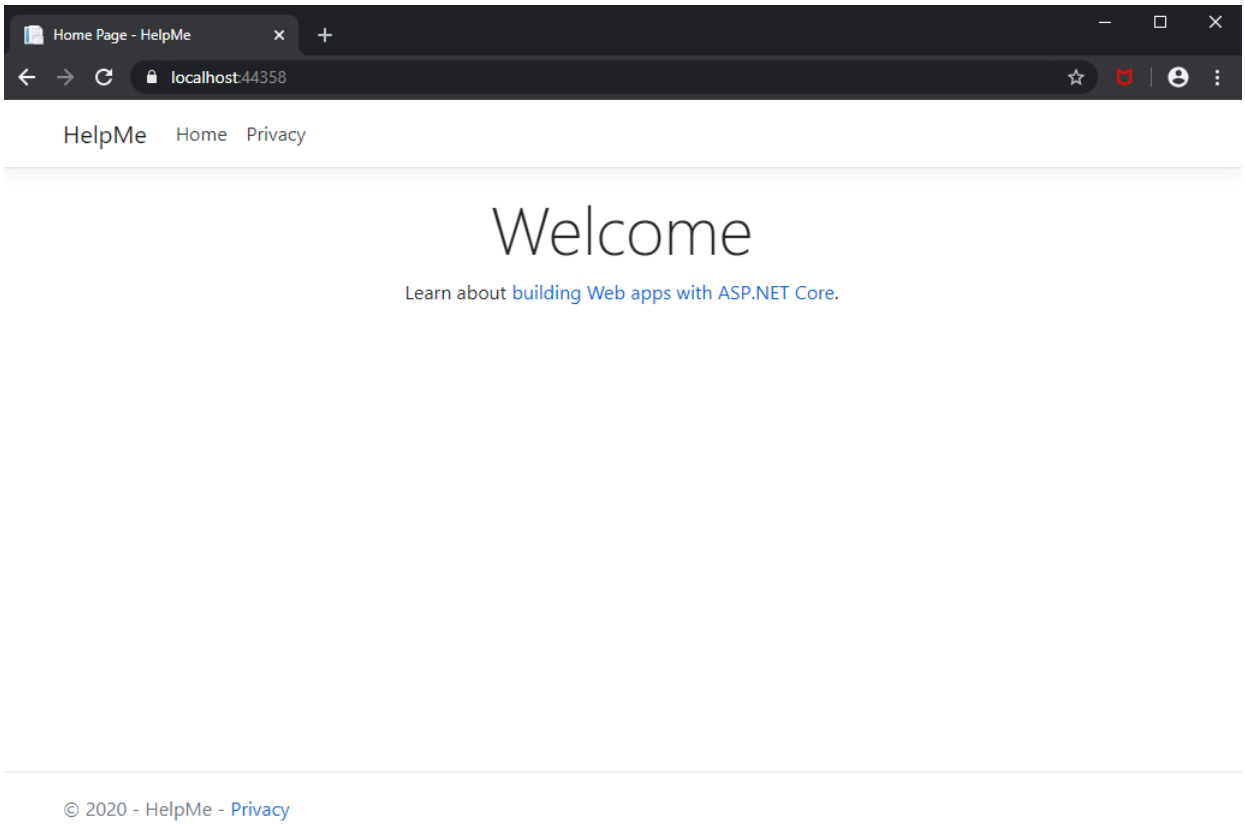
Figura 43 – Visão inicial do projeto



Fonte: Próprio autor.

O projeto já vem com uma estrutura padrão que pode ser visualizada quando Douglas o executa pelo IDE definido, que neste caso é o Visual Studio. Após a execução a tela apresentada na Figura 44 é exibida.

Figura 44 – Visão inicial do projeto padrão em execução



Fonte: Próprio autor.

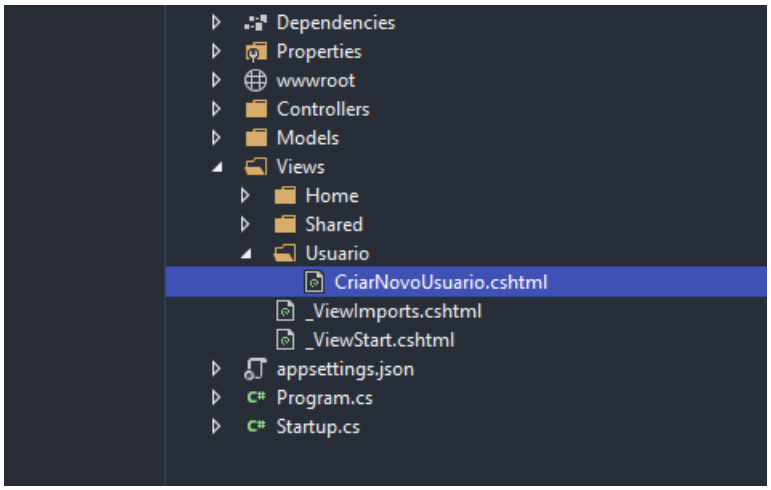
Portanto, após esta sequência de passos, Douglas finaliza mais uma de suas *tasks* que era a criação do projeto no IDE definido.

6.4.1.2 Criar tela de cadastro do usuário

Após criar o projeto no Visual Studio, Douglas pode continuar sua história partindo para próxima *task* que é a de "Criar tela de cadastro do usuário". Vale ressaltar que, neste período entre a criação do banco de dados e do projeto todo o time esteve trabalhando junto em todas as definições. Estes passos iniciais são determinantes nas demais histórias que estão em andamento.

Para iniciar a nova *task* primeiramente Douglas cria uma subpasta nomeada como **"Usuario"** dentro da pasta **"Views"**. A pasta Views já vem definida como padrão na arquitetura MVC criada no IDE escolhido. Dentro dessa subpasta criada, Douglas gera um novo arquivo chamado **"CriarNovoUsuario"** (Figura 45).

Figura 45 – Criação do item que terá detalhes da tela de usuário

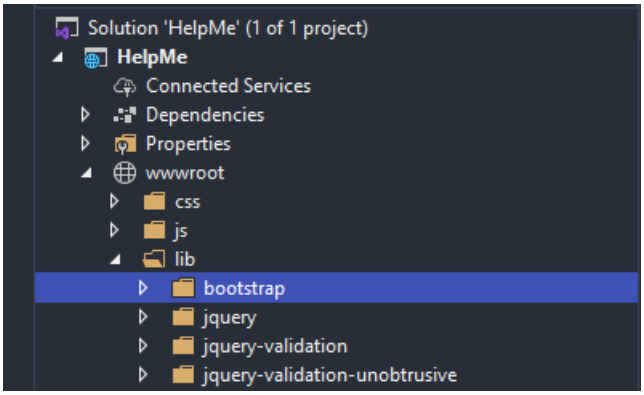


Fonte: Próprio autor.

O item criado terá a extensão **cshtml** que conta com a linguagem de marcação HTML (HyperText Markup Language, ou Linguagem de Marcação de Hipertexto) associada a linguagem C Sharp. Essa linguagem de programação também é utilizada na sintaxe das **Razor Pages**. As Razor Pages possibilitam uma programação com foco na interação de uma aplicação com as páginas do usuário, tornando as operações entre o *back-end* e o *front-end* mais fáceis e práticas (MICROSOFT, 2020d).

Anteriormente, foi definido que como apoio para a parte de *front-end* o time que está desenvolvendo a plataforma Help Me irá utilizar o Bootstrap. A própria estrutura gerada pelo Visual Studio possibilita o uso do Bootstrap como framework *front-end*, sendo que, ele já vem como padrão na pasta **lib** (Figura 46).

Figura 46 – Localização da pasta Bootstrap no projeto



Fonte: Próprio autor.

Para que o Bootstrap possa ser usado nas demais páginas presentes na aplicação basta Douglas importá-lo no head (cabeçalho) do arquivo **_Layout.cshtml** (Figura 47). Esse

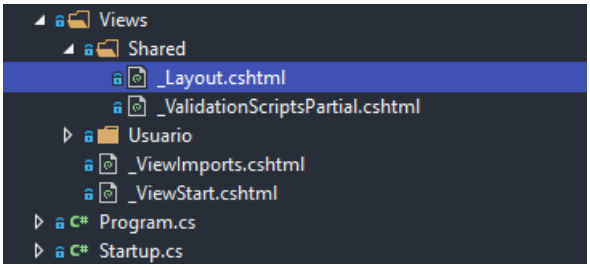
arquivo se encontra na pasta **Shared** (Figura 48) que já vem como padrão na estrutura do projeto criado pelo Visual Studio.

Figura 47 – Importando Bootstrap no head (cabeçalho) do arquivo _Layout.cshtml

```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - HelpMe</title>
    <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
    <link rel="stylesheet" href="~/css/site.css" />
</head>
```

Fonte: Próprio autor.

Figura 48 – Localização do arquivo _Layout.cshtml



Fonte: Próprio autor.

Após Douglas habilitar o uso do Bootstrap no projeto ele cria os campos de entrada (input) que irão receber os dados de cadastro do usuário. Para que as informações sejam armazenadas Douglas usa um formulário com todos os valores mapeados na tabela “usuario” criada no banco de dados na seção 5.3.1.1.2.

Com uso do Bootstrap, Douglas define o layout do formulário que irá possuir os campos de dados que deverão ser preenchidos para a realização do cadastro. A Figura 49, mostra como é feito o preenchimento e obtenção dos dados nos campos “Nome” e “Sobrenome” na tela de cadastro do usuário.

Figura 49 – Entrada de dados no cadastro de usuários

```
<form asp-action="CriarNovoUsuario">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <h3 class="text-center indigo-text font-bold py-4"><strong>Abrir conta</strong></h3>
    <div class="row form-group">
        <div class="md-form col-md-6">
            <label class="control-label">Nome</label>
            <input asp-for="Nome" class="form-control" />
            <span asp-validation-for="Nome" class="text-danger"></span>
        </div>
        <div class="md-form col-md-6">
            <label class="control-label">Sobrenome</label>
            <input asp-for="Sobrenome" class="form-control" />
            <span asp-validation-for="Sobrenome" class="text-danger"></span>
        </div>
    </div>
    <div class="text-center py-4">
        <button class="btn btn-info">Cadastrar <i class="fa fa-paper-plane-o ml-1"></i></button>
    </div>
</form>
```

Fonte: Próprio autor.

Após inserir todos os campos mapeados e realizar a execução do projeto, Douglas tem uma visão geral de como ficou o layout da tela de cadastro de usuários (Figura 50).

Figura 50 – Tela de cadastro de usuários

Fonte: Próprio autor.

Dessa forma, Douglas finaliza mais uma *task* que estava mapeada em sua história.

6.4.1.3 Criar conexão com o banco de dados

Após finalizar a tela onde os dados do usuário serão adicionados, Douglas parte agora para o *back-end* que irá receber as informações e enviá-las para o banco de dados.

Para dar continuidade a inserção dos dados, Douglas deverá criar uma conexão da aplicação com o banco helpme no SQL Server. Para que essa conexão seja estabelecida Douglas irá utilizar o **SqlConnection**. O SqlConnection cria uma conexão e atua como um provedor de dados para operações que envolvam o SQL Server (MICROSOFT, 2020g).

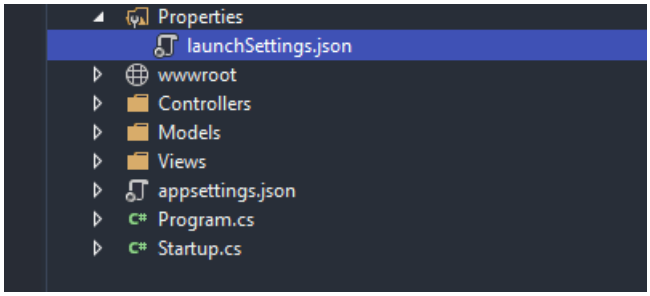
É necessário definir também uma string de conexão que será usada pelo SqlConnection. Este parâmetro é adicionado por Douglas na variável de ambiente **"SQL_SERVER"** (Figura 51) no arquivo **launchSettings** que está na pasta **Properties** (Figura 52).

Figura 51 – Definição da string de conexão com o SQL Server no arquivo launchSettings

```
"HelpMe.Web": {
  "commandName": "Project",
  "launchBrowser": true,
  "applicationUrl": "https://localhost:5001",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development",
    "SQL_SERVER": "Server=TCR-00072\\SQLEXPRESS;Database=helpme;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```

Fonte: Próprio autor.

Figura 52 – Caminho do arquivo launchSettings.json



Fonte: Próprio autor.

Após adicionar a string de conexão no arquivo “launchSettings” é necessário que Douglas crie um método que obtenha essa string e use o SqlConnection para realizar a conexão com o banco (Figura 53).

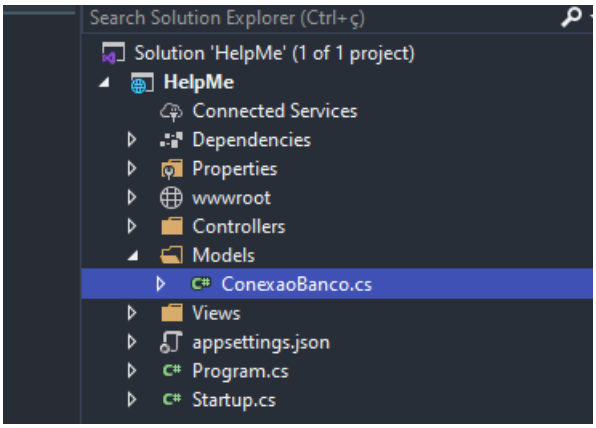
Figura 53 – Pegando variável de ambiente e realizando conexão com banco de dados

```
public IDbConnection GetIDbConnection()
{
    string connectionString = Environment.GetEnvironmentVariable("SQL_SERVER");
    SqlConnection connection = new SqlConnection(connectionString);

    return connection;
}
```

Fonte: Próprio autor.

O método apresentado na Figura 53 está presente na classe “ConexaoBanco” que se encontra na pasta Model do projeto Help Me (Figura 54).

Figura 54 – Localização da classe ConexaoBanco



Fonte: Próprio autor.

Com a conexão em mãos Douglas poderá utilizá-la quando as operações envolvendo o banco forem realizadas.

6.4.1.4 Validar se campos obrigatórios foram preenchidos

Como o time determinou que a maioria dos campos presentes no banco de dados não poderão ser nulos (NOT NULL), algumas validações devem ser inseridas nos campos obrigatórios do sistema. Para realizar está tarefa, Douglas conversando novamente com a equipe técnica definiu que utilizarão o **FluentValidation**.

O FluentValidation é uma biblioteca usada em .NET para criar regras de verificação dos dados que são enviados ao Controller. Seu uso é simples e suas vantagens são grandes, gerando confiança sobre as informações que trafegam no código (FLUENTVALIDATION, 2020).

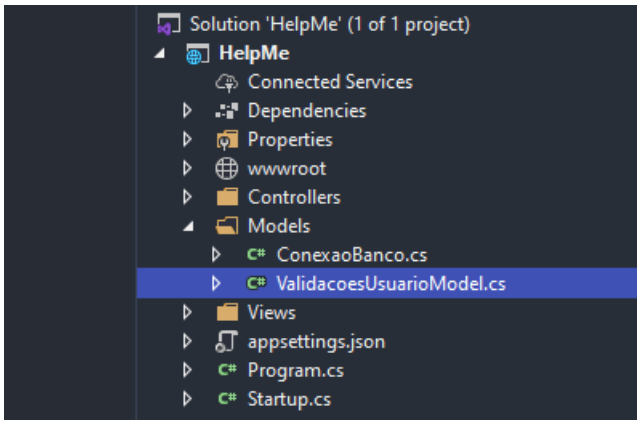
Por meio do FluentValidation, Douglas define quais campos devem ser preenchidos e como serão gerados os alertas na tela para o usuário que não preencher os campos da forma correta (Figura 55). A classe responsável por realizar a validação dos dados do usuário é a **"ValidacoesUsuarioModel"** que está presente na pasta **"Models"** (Figura 56).

Figura 55 – Campos com validações

```
RuleFor(item => item.Nome)
    .NotEmpty().WithMessage("O campo 'Nome' deve ser preenchido")
    .MinimumLength(3).WithMessage("O campo 'Nome' deve possuir mais de 2 letras");

RuleFor(item => item.Sobrenome)
    .NotEmpty().WithMessage("O campo 'Sobrenome' deve ser preenchido")
     .MinimumLength(3).WithMessage("O campo 'Sobrenome' deve possuir mais de 2 letras");

RuleFor(item => item.Celular)
    .NotEmpty().WithMessage("O campo 'Celular' deve ser preenchido")
    .MaximumLength(15).WithMessage("Limite de números excedidos");
```

Fonte: Próprio autor.

Figura 56 – Localização da classe de validação

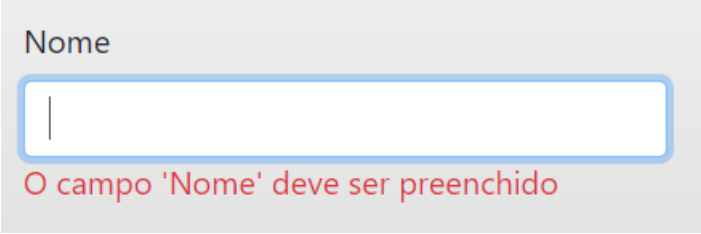


Fonte: Próprio autor.

A Figura 55 mostra o exemplo de algumas validações que devem ser feitas nos campos “Nome”, “Sobrenome” e “Celular”. Dentre as validações Douglas verifica se os 3 campos foram preenchidos pelo usuário, além disso, nos campos “Nome” e “Sobrenome” ele determina que o tamanho mínimo de preenchimento será de 3 caracteres. Também é definido que o tamanho máximo para o campo “Celular” é de 15 caracteres.

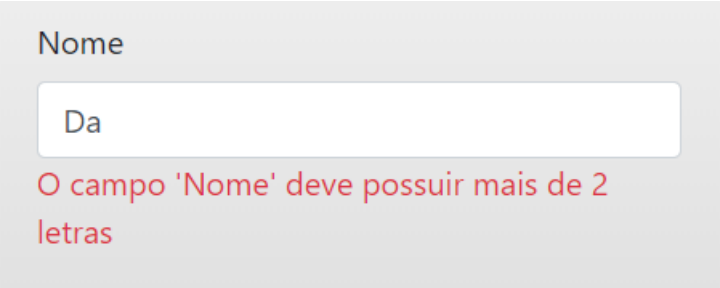
Outras validações são aplicadas por Douglas aos demais itens presentes na tabela e que serão preenchidos pelo usuário na realização do cadastro. Dessa forma, quando algum campo não for preenchido corretamente violando alguma restrição durante o preenchimento, uma notificação será apresentada na tela para o usuário, como mostram as Figuras 57 e 58 após validações para o campo “Nome”.

Figura 57 – Validação quando campo “Nome” estiver vazio



Fonte: Próprio autor.

Figura 58 – Validação do tamanho mínimo de caracteres



Fonte: Próprio autor.

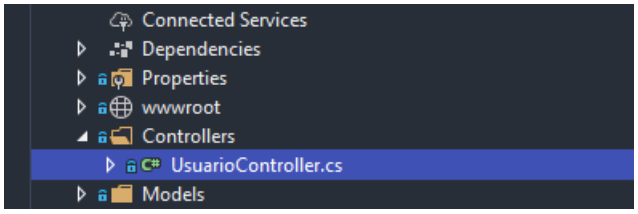
Dessa forma, Douglas finaliza sua *task* de implementar as validações. Após os dados preenchidos pelo usuário passarem pelas validações eles poderão ser inseridos no banco de dados.

6.4.1.5 Inserir informações do cadastro na base de dados

Para que os dados saiam da tela que o usuário preencheu e cheguem até o banco de dados é necessário fazer algumas alterações no *back-end* do sistema, que estão presentes na próxima *task* da história de Douglas.

Com o uso do Razor Pages o direcionamento dos dados da camada View para a Controller é facilitado, isso agiliza o processo e torna a operação mais fácil para o programador. Na pasta **"Controller"** será criada a classe **"UsuarioController.cs"** (Figura 59), que será utilizada nas demais operações que envolverem tratativa de dados do usuário.

Figura 59 – Criando Controller de usuário



Fonte: Próprio autor.

Este Controller terá um método que irá obter um **Model** de usuário (Figura 60).

Figura 60 – Método de criar usuário no Controller

```
public async Task<IActionResult> CriarNovoUsuario(UsuarioModel usuario)
{
    await _usuarioRepository.CriarNovoUsuario(usuario);

    return RedirectToAction("Index");
}
```

Fonte: Próprio autor.

Um Model contém os atributos referentes a um usuário. Os dados que são preenchidos na tela de cadastro são associados ao Model e por meio dele chegam até o banco de dados.

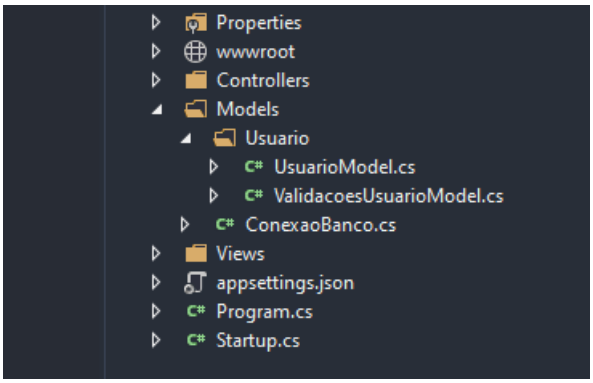
A Figura 61 mostra os atributos que estão presentes dentro do Model **“UsuarioModel”** que está na subpasta **“Usuario”** dentro da pasta “Models”. A classe criada para validação dos dados do usuário (ValidacoesUsuarioModel) também foi movida para a subpasta “Usuario” (Figura 62).

Figura 61 – Dados presentes na classe UsuarioModel

```
public class UsuarioModel
{
    6 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public int IdUsuario { get; set; }
    13 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Nome { get; set; }
    14 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Sobrenome { get; set; }
    14 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Celular { get; set; }
    12 references | Darlan Souza - DTI, 20 days ago | 1 author, 3 changes
    public DateTime DataNascimento { get; set; }
    13 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Rua { get; set; }
    12 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Complemento { get; set; }
    12 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public int Numero { get; set; }
    13 references | Darlan Souza - DTI, 19 days ago | 1 author, 1 change
    public string Bairro { get; set; }
    13 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Cidade { get; set; }
    13 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Estado { get; set; }
    13 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Cep { get; set; }
    9 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Senha { get; set; }
    7 references | Darlan Souza - DTI, 25 days ago | 1 author, 2 changes
    public string Email { get; set; }
}
```

Fonte: Próprio autor.

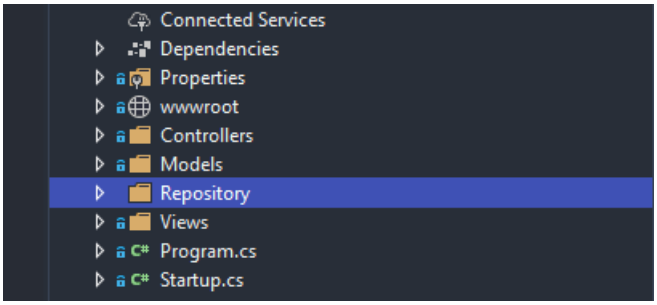
Figura 62 – Pasta da classe UsuarioModel



Fonte: Próprio autor.

Para não deixar a pasta "Models" com muitos arquivos, Douglas resolve dividir sua responsabilidade de tarefas associadas ao tratamento de dados do sistema em mais uma pasta intitulada **"Repository"** (Figura 63). A pasta "Repository" será responsável por realizar os processos relacionados ao banco de dados como conexão, queries e execuções. Dessa forma, o Model com os dados que o usuário preenche na tela serão enviados para a o arquivo "UsuarioModel" e em seguida encaminhados para um Repositório onde será realizada a operação de **INSERT** (inserir) na tabela "usuario" que foi criada.

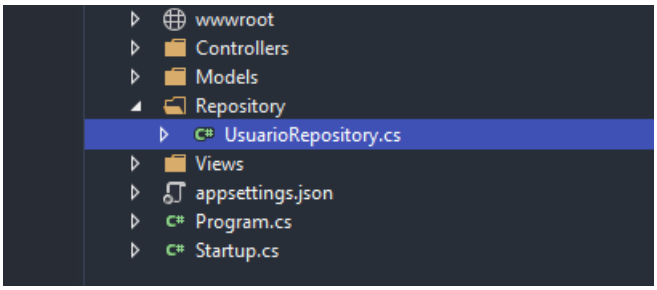
Figura 63 – Exibindo pasta Repository criada



Fonte: Próprio autor.

Douglas utilizará a conexão com o banco de dados na classe **"UsuarioRepository"** que será criada dentro da pasta Repository (Figura 64).

Figura 64 – Criando classe UsuarioRepository

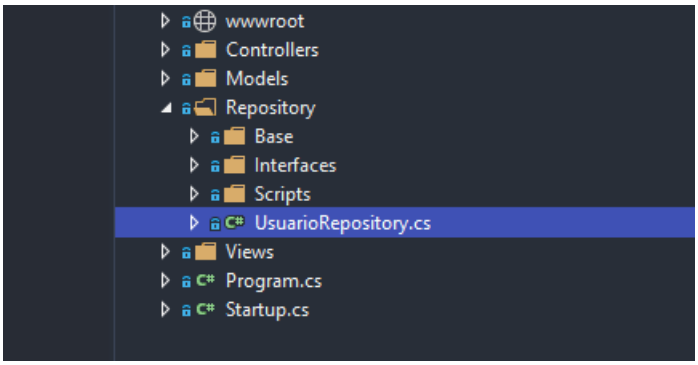


Fonte: Próprio autor.

Dentro desta mesma pasta (Repository) outras 3 subpastas serão criadas (Figura 65). São elas:

- **Base:** Que contém as operações que Douglas realiza para estabelecer conexão com o banco de dados, incluindo a classe apresentada na Figura 54.
- **Interfaces:** Que possui todas as Interfaces que a aplicação utiliza nas operações com o banco.
- **Scripts:** Onde estão os *scripts* de banco de dados que serão utilizados na aplicação.

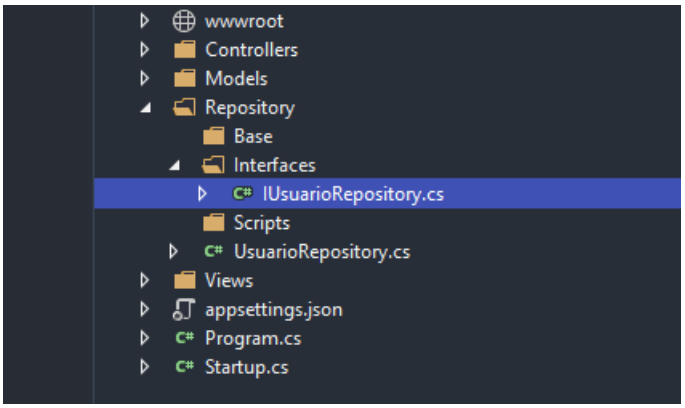
Figura 65 – Subpastas da pasta Repository



Fonte: Próprio autor.

O uso de Interfaces é importante, pois o código fica desacoplado e se torna mais fácil de manter e evoluir, devido a isso, uma Interface de UsuarioRepository é criada dentro da subpasta Interfaces. Como padrão de nomenclatura, no início de uma Interface um "i" maiúsculo deve ser adicionado, portanto, a Interface de UsuarioRepository será nomeada como **"IUsuarioRepository"** (Figura 66).

Figura 66 – Criando Interface de UsuarioRepository



Fonte: Próprio autor.

Dentro da Interface criada, Douglas adiciona um método que será o responsável por fazer a interface de comunicação entre a classe UsuarioController e a classe UsuarioRepository. O método será nomeado de "CriarNovoUsuario" e receberá como parâmetro o mesmo Model que é preenchido no Controller (UsuarioController) com os dados do usuário (Figura 67).

Figura 67 – Método presente na Interface

```
public interface IUsuarioRepository
{
    0 references
    Task CriarNovoUsuario(UsuarioModel usuario);
}
```

Fonte: Próprio autor.

Após o método CriarNovoUsuario presente na Interface IUsuarioRepository ser chamado pelo Controller (UsuarioController) os dados de usuário chegaram na classe UsuarioRepository e poderão ser adicionados no banco de dados através do método **"CriarNovoUsuario"** (Figura 68).

Figura 68 – Método para criar usuário no banco de dados

```
public async Task CriarNovoUsuario(UsuarioModel usuario)
{
    var dataAtual = DateTime.Now;

    DynamicParameters valores = new DynamicParameters();
    valores.Add("@nome", usuario.Nome, DbType.AnsiString);
    valores.Add("@sobrenome", usuario.Sobrenome, DbType.AnsiString);
    valores.Add("@email", usuario.Email, DbType.AnsiString);
    valores.Add("@celular", usuario.Celular, DbType.AnsiString);
    valores.Add("@rua", usuario.Rua, DbType.AnsiString);
    valores.Add("@numero", usuario.Numero, DbType.Int32);
    valores.Add("@complemento", usuario.Complemento, DbType.AnsiString);
    valores.Add("@bairro", usuario.Bairro, DbType.AnsiString);
    valores.Add("@cidade", usuario.Cidade, DbType.AnsiString);
    valores.Add("@cep", usuario.Cep, DbType.AnsiString);
    valores.Add("@estado", usuario.Estado, DbType.AnsiString);
    valores.Add("@senha", usuario.Senha, DbType.AnsiString);
    valores.Add("@dt_nascimento", usuario.DataNascimento, DbType.DateTime);
    valores.Add("@dt_criacao", dataAtual, DbType.DateTime);
    valores.Add("@dt_modificacao", dataAtual, DbType.DateTime);

    await Execute(UsuarioScripts.INSERIR_USUARIO, valores);
}
```

Fonte: Próprio autor.

O método de criação de usuário recebe como parâmetro o Model de usuário que foi preenchido na tela. Os campos **dt_criacao** e **dt_modificacao** receberão a data atual de criação do registro na tabela "usuario". Para que esses valores sejam passados para o *script* que irá fazer a inserção dos dados no SQL Server Douglas usa o **Dapper**.

O Dapper é um **ORM** (Object Relational Mapper, ou Mapeador Relacional de Objeto) que faz o mapeamento entre o banco de dados e a aplicação através da linguagem de programação utilizada, que no projeto desenvolvido está sendo o C Sharp com apoio do .NET Core (TUTORIAL, 2020).

A função **Execute** que está presente no fim do método "CriarNovoUsuario" recebe como parâmetro uma string com o *script* que será usado na criação de um usuário e também os valores que serão utilizados para essa criação (Figura 69).

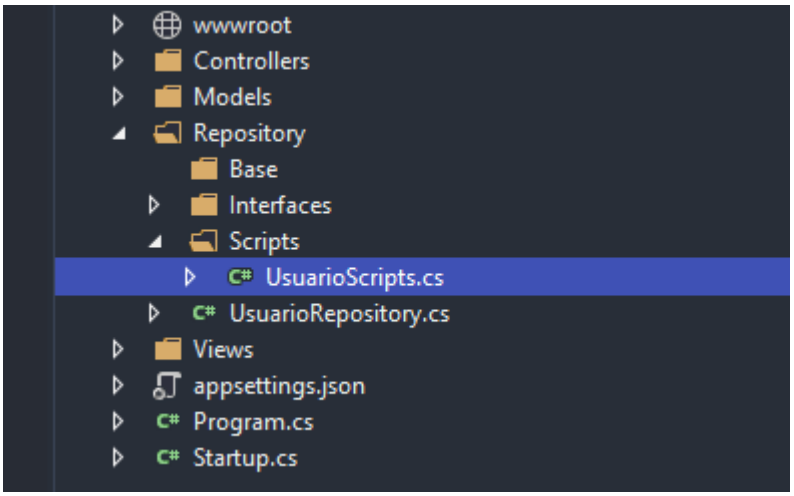
Figura 69 – Chamada do método "Execute" na classe UsuarioRepository

```
await Execute(UsuarioScripts.INSERIR_USUARIO, valores);
```

Fonte: Próprio autor.

Como foi definido anteriormente por Douglas, uma subpasta própria para todos os *scripts* do código foi criada com o nome "Scripts". Dentro desta subpasta foi criada uma classe que será nomeada de **"UsuarioScripts"** (Figura 70).

Figura 70 – Criando classe para os *scripts*



Fonte: Próprio autor.

Dentro desta classe teremos uma constante **INSERIR_USUARIO** (Figura 71) que irá conter o *script* que será passado como parâmetro para o método “Execute” visto anteriormente na Figura 69.

Figura 71 – *Script* para inserir um usuário no banco

```
public const string INSERIR_USUARIO = @"
    INSERT INTO usuario(
        nome,
        sobrenome,
        email,
        celular,
        rua,
        numero,
        complemento,
        bairro,
        cidade,
        cep,
        estado,
        senha,
        dt_nascimento,
        dt_criacao,
        dt_modificacao
    )
    VALUES(
        @nome,
        @sobrenome,
        @email,
        @celular,
        @rua,
        @numero,
        @complemento,
        @bairro,
        @cidade,
        @cep,
        @estado,
        @senha,
        @dt_nascimento,
        @dt_criacao,
        @dt_modificacao
    )";
```

Fonte: Próprio autor.

Quando o *script* anterior é executado os valores passados pelo usuário na tela substituem os campos em que estão presentes o "@". Dessa forma, em uma tabela simples onde apenas o nome e o sobrenome do usuário serão inseridos no banco, o *script* terá a seguinte configuração:

INSERT INTO usuario (nome, sobrenome) VALUES ("André", "Santos")

No exemplo anterior o nome **André** e o sobrenome **Santos** foram inseridos pelo usuário na tela e passados através do Dapper para o *script* acima. Com a execução do *script* teremos o nome e o sobrenome presentes na tabela contida no banco. Portanto, após Douglas realizar todas essas tarefas o cadastro do usuário será realizado no sistema e sua *task* estará finalizada.

Douglas, no decorrer de suas atividades já veio desenvolvendo vários testes para analisar se suas alterações estavam gerando o resultado desejado. Para finalizar, ele realizou uma operação para garantir que o processo de criação de usuário está sendo feito de forma

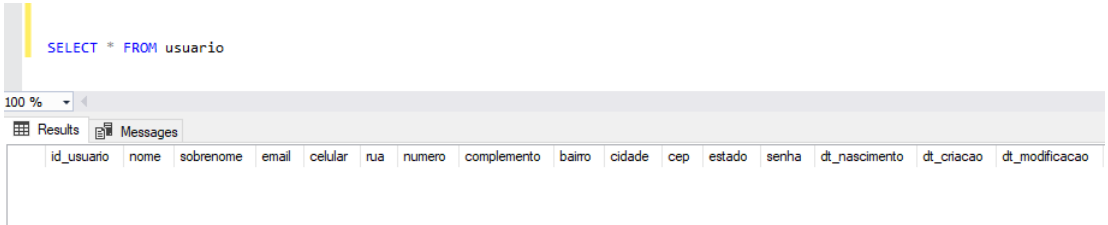
correta durante todo o fluxo. Esta validação se deu com o preenchimento de todos os dados da tela da seguinte maneira (Figura 72).

Figura 72 – Inserindo dados para criar um novo usuário

Fonte: Próprio autor.

Antes de clicar na opção “Cadastrar”, Douglas analisa mais uma vez no SQL Server Management Studio como a tabela “usuario” se encontra no banco de dados helpme. O resultado de sua consulta foi o seguinte (Figura 73).

Figura 73 – Tabela de usuario sem registros

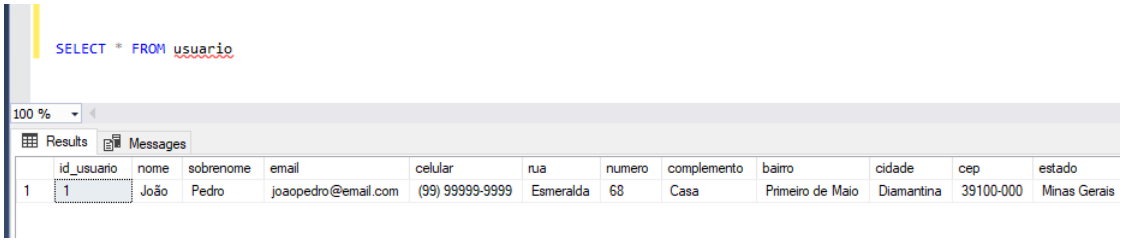


Fonte: Próprio autor.

Portanto, a consulta anterior mostra que antes de realizar a operação a tabela não possui registros. Após Douglas clicar na opção “Cadastrar” ele confere a tabela novamente e

confirma que as informações que ele preencheu na tela foram inseridas da forma correta no banco. Isso pode ser confirmado através da Figura 74, onde uma parte dos dados inseridos na tabela por Douglas podem ser visualizados.

Figura 74 – Tabela usuario com registros



	id_usuario	nome	sobrenome	email	celular	rua	numero	complemento	bairro	cidade	cep	estado
1	1	João	Pedro	joaopedro@email.com	(99) 99999-9999	Esmeralda	68	Casa	Primeiro de Maio	Diamantina	39100-000	Minas Gerais

Fonte: Próprio autor.

Com isso, através das implementações de código, testes e validações com todo o time Douglas finaliza sua história.

6.4.2 História B

Como definido pelo time, Fernanda irá desenvolver a história B que foi apresentada na seção 4.2.2.2, sua descrição é:

- *Eu, como usuário(a), desejo visualizar meus dados de cadastro para verificar se estão corretos.*

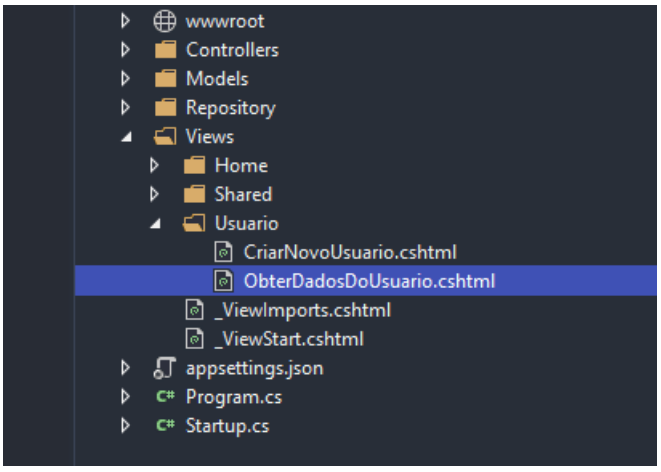
A história citada conta com as seguintes *tasks* associadas a ela:

1. Criar página de visualização dos dados do usuário.
2. Buscar dados do banco e preencher os campos da tela.

6.4.2.1 Criar página de visualização dos dados do usuário

Assim como Douglas, Fernanda também criará um arquivo dentro da subpasta Usuario na pasta Views. Ela define que esse novo arquivo será nomeado de “ObterDadosDoUsuario” (Figura 75).

Figura 75 – Criação do item que terá os dados do usuário cadastrado



Fonte: Próprio autor.

Dentro desse arquivo Fernanda cria a estrutura que será responsável por retornar os dados do banco de dados para a tela na pesquisa de um usuário específico. Dada a análise anterior, Fernanda cria 2 campos “Nome” e “Sobrenome” com a estrutura que estará presente também nos demais campos da tela de visualização dos dados (Figura 76).

Figura 76 – Formulário para visualização dos dados do usuário

```
<form asp-action="ObterDadosDoUsuario">
    <h3 class="text-center indigo-text font-bold py-4"><strong>Consultar dados</strong></h3>
    <div class="row form-group">
        <div class="md-form col-md-6">
            <label>Nome</label>
            <input class="form-control" value="@Html.DisplayFor(model => model.Nome)" disabled />
        </div>
        <div class="md-form col-md-6">
            <label>Sobrenome</label>
            <input class="form-control" value="@Html.DisplayFor(model => model.Sobrenome)" disabled />
        </div>
    </div>
</form>
```

Fonte: Próprio autor.

Com base nesta estrutura, após mapear os campos que serão visíveis para o usuário, Fernanda cria a seguinte tela de visualização de informações (Figura 77).

Figura 77 – Tela para visualizar os dados do usuário

Fonte: Próprio autor.

Dessa forma, Fernanda finaliza a primeira *task* de sua história.

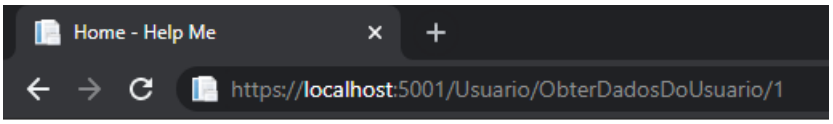
6.4.2.2 Buscar dados do banco e preencher os campos da tela

Para que as operações de *back-end* sejam realizadas, Fernanda usará o mesmo Controller, Interface e Repositório criados por Douglas em sua história. Isso mostra a importância do alinhamento constante de todo o time durante a Sprint para que atividades já realizadas em uma história não sejam duplicadas em outra.

Inicialmente, Fernanda define com todo o time que a primeira abordagem para retornar os dados do usuário será passar o identificador único do usuário (id_usuario) na URL da aplicação. Esta abordagem inicial atende à demanda dos clientes e posteriormente em outras Sprints evoluções deste modelo poderão ser levantadas.

Portando, com o uso do id_usuario presente na tabela os dados dos usuários serão retornados e passados para a tela, sendo que, a configuração da URL ficará como a apresentada na Figura 78.

Figura 78 – Formatação da URL para obtenção de informações do usuário



Fonte: Próprio autor.

A URL mostrada na Figura 78 tem o intuito de esclarecer que os dados retornados para a tela criada por Fernanda irão corresponder ao usuário que tem o id_usuario igual a 1. Dada essa definição, Fernanda cria o método no Controller que irá receber o id_usuário da tela e com base nele retornar os dados do usuário (Figura 79).

Figura 79 – Método para obtenção de dados no Controller

```
[HttpGet]
0 references | Darlan Souza - DTI, 14 days ago | 1 author, 6 changes
public IActionResult ObterDadosDoUsuario(int idUsuario)
{
    Task<UsuarioModel> retorno = BuscarDadosDoUsuario(idUsuario);

    return View("~/Views/Usuario/ObterDadosDoUsuario.cshtml", retorno.Result);
}

3 references | Darlan Souza - DTI, 15 days ago | 1 author, 1 change
public async Task<UsuarioModel> BuscarDadosDoUsuario(int idUsuario)
{
    return await _usuarioRepository.ObterDadosDoUsuario(idUsuario);
}
```

Fonte: Próprio autor.

Com o método de obtenção dos dados do usuário criado no Controller, Fernanda parte agora para a definição da Interface e do Repositório para obtenção dos dados. Para isso, ela cria na Interface IUsuarioRepository um método chamado **ObterDadosDoUsuario** (Figura 80).

Figura 80 – Método para obtenção de dados na Interface

```
public interface IUsuarioRepository
{
    0 references
    Task CriarNovoUsuario(UsuarioModel usuario);
    0 references
    Task<UsuarioModel> ObterDadosDoUsuario(int idUsuario);
}
```

Fonte: Próprio autor.

Como a classe UsuarioRepository usa essa Interface, o método criado por Fernanda deve ser implementado na classe em questão e, através dele, a obtenção dos dados do usuário ocorrerá (Figura 81).

Figura 81 – Método para obtenção de dados no Repositório

```
public async Task<UsuarioModel> ObterDadosDoUsuario(int idUsuario)
{
    DynamicParameters valores = new DynamicParameters();
    valores.Add("@id_usuario", idUsuario, DbType.Int32);

    return await Obter<UsuarioModel>(UsuarioScripts.OBTER_USUARIO, valores);
}
```

Fonte: Próprio autor.

O método apresentado na Figura 81 irá receber como parâmetro o identificador único do usuário (id_usuario) e passá-lo para a constante **OBTER_USUARIO** que também se encontra na subpasta “Scripts” vista anteriormente (Figura 82).

Figura 82 – Query para consulta de dados do usuário

```
public const string OBTER_USUARIO = @"
    SELECT
        id_usuario AS IdUsuario,
        nome AS Nome,
        sobrenome AS Sobrenome,
        celular AS Celular,
        rua AS Rua,
        numero AS Numero,
        complemento AS Complemento,
        bairro AS Bairro,
        cidade AS Cidade,
        cep AS Cep,
        estado AS Estado,
        dt_nascimento AS DataNascimento
    FROM
        usuario
    WHERE
        id_usuario = @id_usuario";
```

Fonte: Próprio autor.

Fernanda define que após a execução dessa query um objeto UsuarioModel é retornado para o Controller (como pode ser visto na Figura 81). Dessa forma, os dados retornados para o Controller são enviados para os campos correspondentes da tela para que possam ser visualizados pelo usuário.

Após estas atividades feitas, Fernanda realiza mais um teste para validar se as informações estão sendo retornadas corretamente para a tela de visualização dos dados. Pesquisando a mesma URL apresentada na Figura 78 ela obteve o seguinte resultado (Figura 83).

Figura 83 – Visualizando dados do usuário

Fonte: Próprio autor.

Com o teste anterior sendo feito e apresentando sucesso, Fernanda finaliza também a sua história. Dessa forma, o cadastro e visualização dos dados do usuário já está concluído, bastando agora a parte de atualização dos dados que será realizada pelo Bruno para que o objetivo da Sprint 1 seja alcançado.

6.4.3 História C

A última história da primeira Sprint do projeto Help Me que foi definida na seção 4.2.2.2 ficou sob responsabilidade do Bruno, sua descrição é:

- *Eu, como usuário(a), desejo editar meus dados de cadastro para atualizar minhas informações.*

A história citada conta com as seguintes *tasks* associadas a ela:

1. Modificar tela de visualização dos dados do usuário para que ela direcione à tela de atualização.
2. Criar tela de atualização dos dados.
3. Enviar dados atualizados para o banco de dados.

6.4.3.1 Modificar tela de visualização dos dados do usuário para que ela direcione à tela de atualização

Bruno inicialmente terá que alterar a tela criada por Fernanda. Esta modificação é necessária para que o usuário ao clicar na opção "Atualizar" possa ser direcionado para a tela onde irá atualizar e salvar seus dados.

Para isso, Bruno adiciona um trecho de código na tela criada por Fernanda, como mostra a Figura 84.

Figura 84 – Adicionando botão na tela de visualização dos dados do usuário

```
<div class="text-center py-4">
    <button class="btn btn-info">
        @Html.ActionLink("Atualizar", "AtualizarDadosDoUsuario", new { idUsuario = Model.IdUsuario })
        <i class="fa fa-edit ml-1"></i>
    </button>
</div>
```

Fonte: Próprio autor.

Dessa forma, Bruno adiciona um botão que irá encaminhar o usuário para a tela **AtualizarDadosDoUsuario** que será criada posteriormente. O botão também enviará o id_usuario para a tela de atualização e com base nele a operação de modificação dos dados poderá ser realizada. O resultado da inserção do botão na tela de visualização pode ser observado na Figura 85.

Figura 85 – Botão na tela de visualização dos dados do usuário

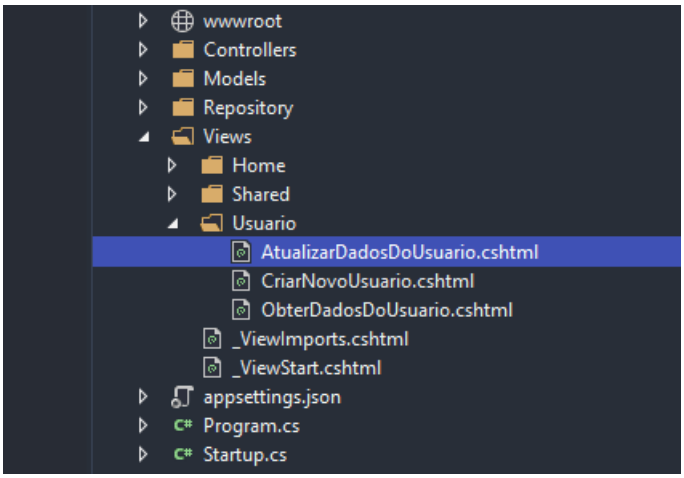
Fonte: Próprio autor.

Assim, Bruno finaliza a primeira *task* da sua história, podendo agora passar para a sua segunda tarefa.

6.4.3.2 Criar tela de atualização dos dados

Para completar a *task* Bruno irá criar dentro da subpasta Usuario que está na pasta Views, um arquivo nomeado de **AtualizarDadosDoUsuario** (Figura 86).

Figura 86 – Criação do item que irá atualizar os dados do usuário



Fonte: Próprio autor.

Dentro desse arquivo Bruno cria a estrutura que será responsável por receber os dados do usuário e realizar a sua atualização. De início, Bruno define como será a estrutura do formulário para os campos “Nome” e “Sobrenome” (Figura 87).

Figura 87 – Formulário para atualização dos dados

```
<form asp-action="AtualizarDadosDoUsuario">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <h3 class="text-center indigo-text font-bold py-4"><strong>Atualizar seus dados</strong></h3>
    <div class="row form-group">
        <div class="md-form col-md-6">
            <label class="control-label">Nome</label>
            <input asp-for="Nome" class="form-control" />
            <span asp-validation-for="Nome" class="text-danger"></span>
        </div>
        <div class="md-form col-md-6">
            <label class="control-label">Sobrenome</label>
            <input asp-for="Sobrenome" class="form-control" />
            <span asp-validation-for="Sobrenome" class="text-danger"></span>
        </div>
    </div>
    <input asp-for="IdUsuario" type="hidden" />
    <div class="form-group text-center">
        <button class="btn btn-info">Salvar Alterações <i class="fa fa-paper-plane-o ml-1"></i></button>
    </div>
</form>
```

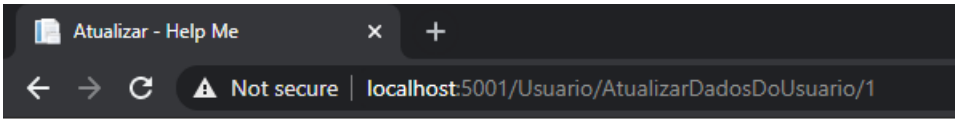
Fonte: Próprio autor.

Percebemos também, através da Figura 87, que Bruno já adiciona o botão que irá executar a operação de atualização dos dados, o nomeando de **“Salvar Alterações”**. Este botão pega o id_usuario e o envia para o Controller. Dessa forma, Bruno finaliza os demais campos que o usuário poderá atualizar e passa para sua próxima tarefa.

6.4.3.3 Enviar dados atualizados para o banco de dados

Seguindo a mesma abordagem de Fernanda, Bruno também utilizará o Controller, Interface e Repositório criados por Douglas em sua história. Além disso, Bruno também passará o identificador único do usuário (id_usuario) para realizar a alteração dos dados no banco de dados (Figura 88).

Figura 88 – Formatação da URL para atualizar os dados do usuário



Fonte: Próprio autor.

Portando, Bruno envia o id_usuario para o Controller e com base nele o Controller poderá obter as informações do usuário e posteriormente atualizar seus dados (Figura 89).

Figura 89 – Método no Controller para atualização de dados do usuário

```
[HttpGet]
0 references | Darlan Souza - DTI, 20 days ago | 1 author, 6 changes
public IActionResult AtualizarDadosDoUsuario(int idUsuario)
{
    Task<UsuarioModel> retorno = BuscarDadosDoUsuario(idUsuario);

    return View(retorno.Result);
}

[HttpPost]
1 reference | Darlan Souza - DTI, 21 days ago | 1 author, 2 changes
public async Task<IActionResult> AtualizarDadosDoUsuario(UsuarioModel usuario)
{
    await _usuarioRepository.AtualizarDadosDoUsuario(usuario);

    return RedirectToAction("Index");
}

3 references | Darlan Souza - DTI, 21 days ago | 1 author, 1 change
public async Task<UsuarioModel> BuscarDadosDoUsuario(int idUsuario)
{
    return await _usuarioRepository.ObterDadosDoUsuario(idUsuario);
}
```

Fonte: Próprio autor.

Os métodos apresentados na Figura 89 contam com um **HttpGet**, que irá preencher os campos da tela do usuário utilizando o método **BuscarDadosDoUsuario**. Os métodos contam também com o um **HttpPost**, que recebe os dados alterados pelo usuário e os envia para o método AtualizarDadosDoUsuario que será adicionado por Bruno na Interface do sistema (Figura 90).

Figura 90 – Método para atualização de dados presente na Interface

```
public interface IUsuarioRepository
{
    0 references
    Task CriarNovoUsuario(UsuarioModel usuario);
    0 references
    Task<UsuarioModel> ObterDadosDoUsuario(int idUsuario);
    0 references
    Task AtualizarDadosDoUsuario(UsuarioModel usuario);
}
```

Fonte: Próprio autor.

Uma vez que a classe UsuarioRepository usa essa Interface, o método que foi criado por Bruno deve ser implementado na classe em questão. Através dela a atualização dos dados do usuário acontecerá (Figura 91).

Figura 91 – Método para atualização de dados no Repositório

```
public async Task AtualizarDadosDoUsuario(UsuarioModel usuario)
{
    var dataAtual = DateTime.Now;

    DynamicParameters valores = new DynamicParameters();
    valores.Add("@id_usuario", usuario.IdUsuario, DbType.Int32);
    valores.Add("@nome", usuario.Nome, DbType.AnsiString);
    valores.Add("@sobrenome", usuario.Sobrenome, DbType.AnsiString);
    valores.Add("@celular", usuario.Celular, DbType.AnsiString);
    valores.Add("@rua", usuario.Rua, DbType.AnsiString);
    valores.Add("@numero", usuario.Numero, DbType.Int32);
    valores.Add("@complemento", usuario.Complemento, DbType.AnsiString);
    valores.Add("@bairro", usuario.Bairro, DbType.AnsiString);
    valores.Add("@cidade", usuario.Cidade, DbType.AnsiString);
    valores.Add("@cep", usuario.Cep, DbType.AnsiString);
    valores.Add("@estado", usuario.Estado, DbType.AnsiString);
    valores.Add("@dt_nascimento", usuario.DataNascimento, DbType.DateTime);
    valores.Add("@dt_modificacao", dataAtual, DbType.DateTime);

    await Execute(UsuarioScripts.ATUALIZAR_USUARIO, valores);
}
```

Fonte: Próprio autor.

O método apresentado na Figura 91 irá receber como parâmetro o Model de usuário. Com base nele o id_usuario será enviado para o *script* **ATUALIZAR_USUARIO** que também se encontra na subpasta “Scripts” (Figura 92). Lembrando que esses novos dados inseridos pelo usuário na atualização também passarão pelas validações desenvolvidas por Douglas em sua história.

Figura 92 – *Script* para atualizar dados do usuário

```
public const string ATUALIZAR_USUARIO = @"
    UPDATE
        usuario
    SET
        nome = @nome,
        sobrenome = @sobrenome,
        celular = @celular,
        rua = @rua,
        numero = @numero,
        complemento = @complemento,
        bairro = @bairro,
        cidade = @cidade,
        cep = @cep,
        estado = @estado,
        dt_nascimento = @dt_nascimento,
        dt_modificacao = @dt_modificacao
    WHERE
        id_usuario = @id_usuario";
```

Fonte: Próprio autor.

A Figura 91 mostra como será a atualização através do **UPDATE** do SQL Server utilizando o id_usuario como o identificador único para o registro no banco. Após estas atividades feitas, Bruno realiza um teste de atualização das informações do usuário com id_usuario igual a 1, para validar se as informações estão sendo atualizadas corretamente. Utilizando a mesma URL apresentada na Figura 88 as seguintes informações do usuário são retornadas (Figura 93).

Figura 93 – Visualizando dados do usuário retornados na tela de atualização

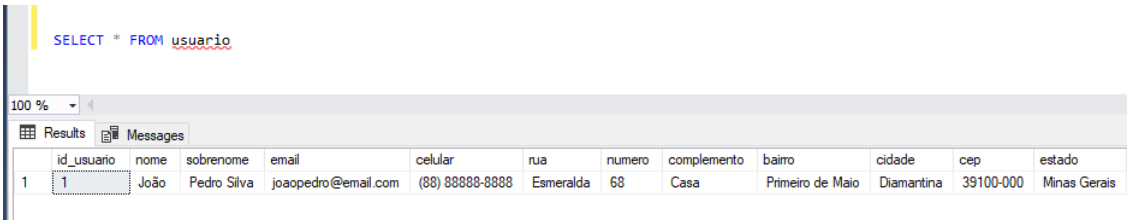
Fonte: Próprio autor.

Desta forma, Bruno realiza a atualização de 2 campos no registro do usuário com o id_usuario igual a 1. Ele altera o campo “Sobrenome” e o campo “Celular” (Figura 94). Após realizar as modificações, Bruno clica no botão “Salvar Alterações”. Conferindo o resultado da modificação no banco ele valida que tudo está funcionando da forma correta como mostra a Figura 95.

Figura 94 – Atualizando campos com dados do usuário

Fonte: Próprio autor.

Figura 95 – Atualização de informações salvas no banco de dados



```
SELECT * FROM usuario
```

100 % | Results | Messages

	id_usuario	nome	sobrenome	email	celular	rua	numero	complemento	bairro	cidade	cep	estado
1	1	João	Pedro Silva	joaopedro@email.com	(88) 88888-8888	Esmeralda	68	Casa	Primeiro de Maio	Diamantina	39100-000	Minas Gerais

Fonte: Próprio autor.

Para verificar a alteração gerada uma comparação da Figura 74 com a Figura 95 pode ser feita. Através das comparações podemos perceber que o campo “sobrenome” foi atualizado de **Pedro** para **Pedro Silva** e o campo “celular” foi atualizado de **(99) 99999-9999** para **(88) 88888-8888**. Como o teste feito por Bruno apresentou sucesso ele finaliza sua última *task*, podendo agora dar por finalizada sua história.

Os testes de usuário são essenciais para o desenvolvimento de um sistema, pois através deles verificamos se a funcionalidade desenvolvida atende o que era esperado. Porém, apenas esses testes não garantem a consistência do sistema, é necessário realizar também testes automatizados da aplicação, como serão apresentados no capítulo 7.

7 REALIZANDO TESTES AUTOMATIZADOS NA APLICAÇÃO

Testes automatizados são de extrema importância em qualquer aplicação (Figura 96). Testes garantem que o aplicativo/software realmente atende o que era esperado. Com base neles o software pode ser evoluído e melhorado. Com estes levantamentos um exemplo de teste será apresentado para a história A, desenvolvida por Douglas e que aborda a criação de um novo usuário.

Figura 96 – Testes

Fonte: Próprio autor.

7.1 Realizando testes

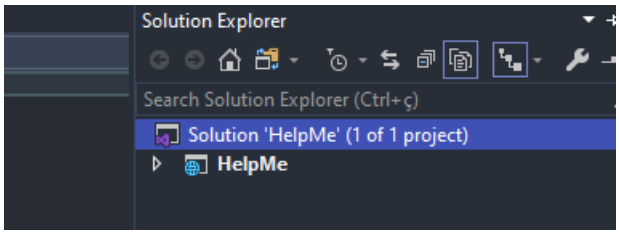
Para que uma funcionalidade seja entregue é necessário que ela passe pela fase de testes. Cada nova funcionalidade de uma aplicação pode ser estruturada como mostra a Figura 97.

Figura 97 – Fluxo de liberação de uma nova funcionalidade

Fonte: Próprio autor.

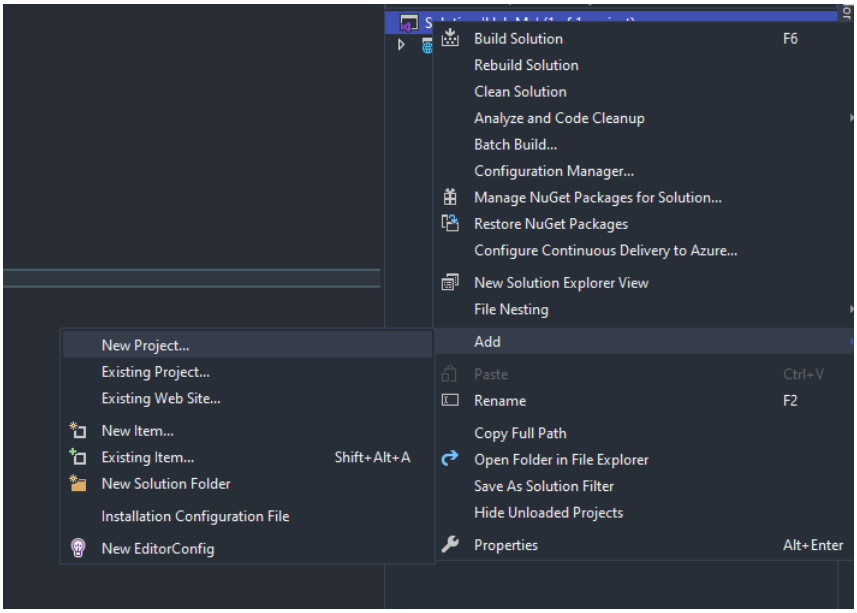
O projeto Helpe Me desenvolvido está dentro de uma "Solution" (Solução) no Visual Studio (Figura 98). Esta Solução pode possuir outros projetos que fazem comunicação entre si. Dessa forma, dentro da Solução apresentada, Douglas gera um novo projeto que será utilizado para criação dos testes das funcionalidades presentes no sistema. Clicando com o botão direito no **Solution HelpMe**, escolhendo a opção **"Add"** e sem seguida a opção **"New Project"** (Figura 99).

Figura 98 – Solução que contém o projeto Help Me



Fonte: Próprio autor.

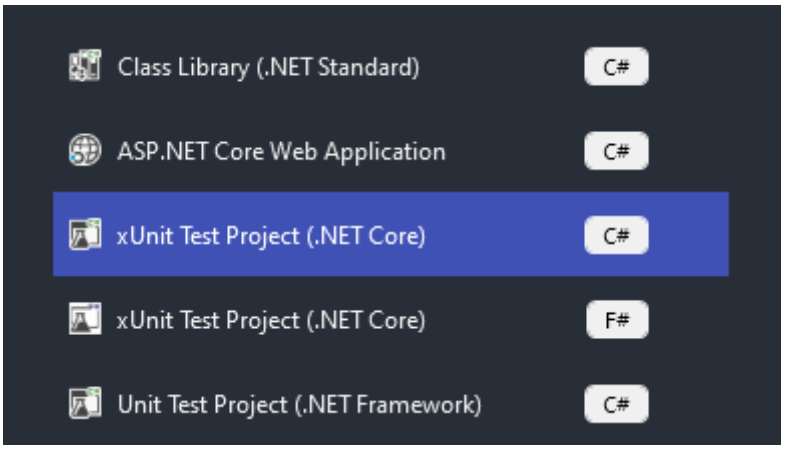
Figura 99 – Criando novo projeto



Fonte: Próprio autor.

O segundo passo feito por Douglas é definir o novo projeto como um projeto de testes (Figura 100).

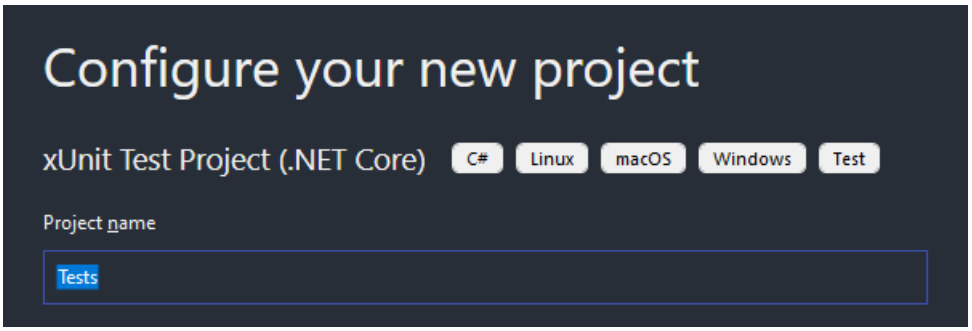
Figura 100 – Caminho para criação de um projeto de testes



Fonte: Próprio autor.

Novamente, conversando com todo o time chegam à conclusão de que o projeto em questão será nomeado de **"Tests"** (Figura 101).

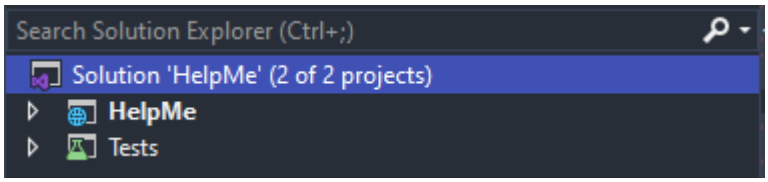
Figura 101 – Definindo nome do projeto de testes



Fonte: Próprio autor.

Ao final desses passos a aplicação HelpMe contará com dois projetos, o **Web**, voltado para a construção do sistema em si e o **Tests**, utilizado para realizar os testes que envolvem as operações do sistema (Figura 102).

Figura 102 – Visualizando projetos

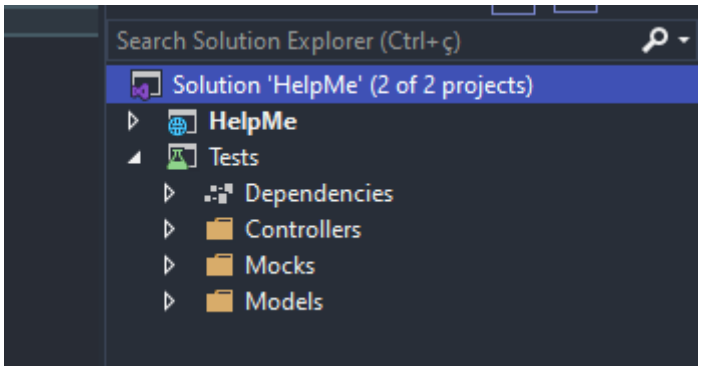


Fonte: Próprio autor.

Dentro do projeto de testes 3 pastas são criadas (Figura 103).

- A pasta **"Controllers"**, que irá realizar os testes das operações que envolvem o Controller da aplicação.
- A pasta **"Models"** que será responsável pelos testes do Model da aplicação.
- A pasta **"Mocks"** onde estarão presentes operações que simulam o banco de dados da aplicação.

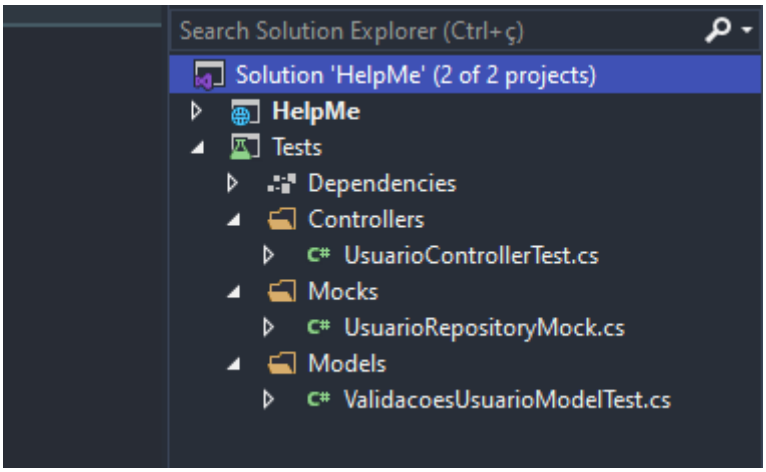
Figura 103 – Pastas do projeto de testes



Fonte: Próprio autor.

Em seguida Douglas cria os arquivos que serão necessários para realização dos testes (Figura 104).

Figura 104 – Arquivos necessários para criação dos testes



Fonte: Próprio autor.

Foram criados 3 arquivos, como mostra a Figura 104.

- O arquivo **UsuarioControllerTest.cs** será utilizado para realizar os testes que envolvem o Controller da aplicação.
- O **ValidacoesUsuarioModelTest.cs** será utilizado para validação do Model preenchido pelo usuário.
- O **UsuarioRepositoryMock.cs** será utilizado para simular as operações de banco que são realizadas na aplicação.

Após a criação da estrutura inicial tem-se início o desenvolvimento dos testes.

7.1.1 Testando validação de dados

O primeiro teste criado por Douglas é o de validação dos campos obrigatórios com uso do FluentValidation. Como o teste envolve o Model do sistema ele será realizado na classe

ValidacoesUsuarioModelTest. Nesta classe serão realizados dois cenários de teste, um deles é quando o Model preenchido pelo usuário for válido e o outro quando o Model for inválido.

Para testar quando o Model é válido, Douglas cria o método **SeValidarUsuario_QuandoDadosValidos_EntaoModelValido**. Este método preencherá o Model UsuarioModel com dados corretos do usuário e posteriormente verificará se é verdade que os dados inseridos são válidos (Figura 105).

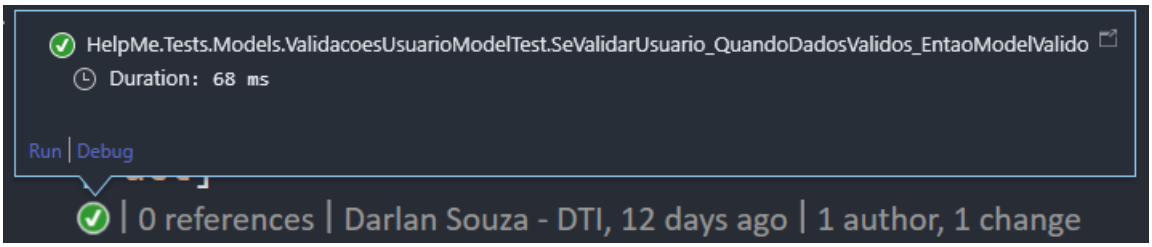
Figura 105 – Testando Model de usuário válido

```
[Fact]
| 0 references | Darlan Souza - DTI, 12 days ago | 1 author, 1 change
public void SeValidarUsuario_QuandoDadosValidos_EntaoModelValido()
{
    ValidacoesUsuarioModel validarUsuario = new ValidacoesUsuarioModel();

    var model = new UsuarioModel()
    {
        Nome = "Darlan",
        Sobrenome = "Souza Silva",
        Celular = "(99) 99999-9999",
        Email = "darlan@email.com",
        Rua = "Rua A",
        Numero = 10,
        Complemento = "Casa",
        Bairro = "Saúde",
        Cidade = "Diamantina",
        Cep = "39100-000",
        Estado = "MG",
        DataNascimento = new DateTime(1998, 9, 7),
        Senha = "abcd1234"
    };

    var resultadoValidacao = validarUsuario.TestValidate(model);

    Assert.True(resultadoValidacao.IsValid);
}
```

Fonte: Próprio autor.

Após a criação do teste, Douglas o executa para verificar se seu resultado está de acordo com o que foi codificado. Analisando o retorno da execução, ele percebe que o teste atende o que havia planejado e que passou com sucesso (Figura 106).

Figura 106 – Retorno do teste onde o UsuarioModel é válido



Fonte: Próprio autor.

Para verificar quando o Model é inválido, Douglas preenche o Model UsuarioModel com algumas informações incompletas, que posteriormente serão bloqueadas nas validações do

FluentValidation. Para isso, ele cria outro método de teste e o nomeia como **SeValidarUsuario_QuandoDadosInvalidos_EntaoModelInvalido**.

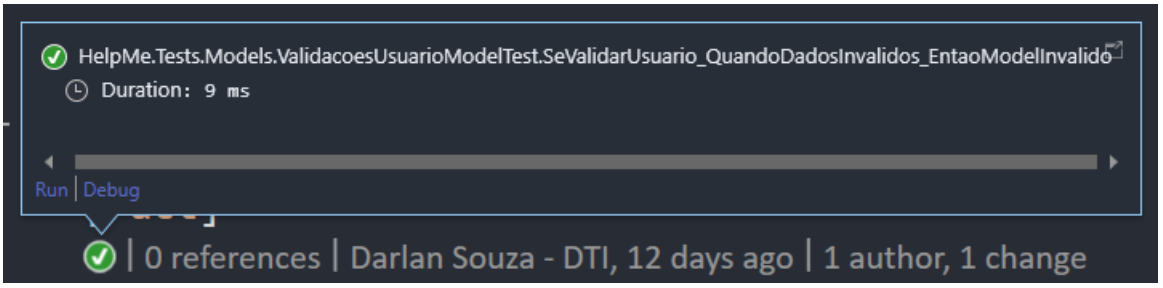
Este método recebe um Model com os dados do usuário e verifica se é falso que os dados inseridos são válidos. Em seguida, é feita uma análise de quais dados estão incorretos e qual mensagem será exibida para o usuário alertando sobre aquele erro (Figura 107). Por último, Douglas executa o teste e percebe que ele atende o planejado (Figura 108).

Figura 107 – Testando Model de usuário inválido

```
[Fact]
0 references | Darlan Souza - DTI, 12 days ago | 1 author, 1 change
public void SeValidarUsuario_QuandoDadosInvalidos_EntaoModelInvalido()
{
    var validarUsuario = new ValidacoesUsuarioModel();

    var model = new UsuarioModel()
    {
        Celular = "(99) 99999-99999",
        Sobrenome = "So",
        Email = "darlan@email.com",
        Rua = "Rua A",
        Numero = 10,
        Complemento = "Casa",
        Bairro = "Saúde",
        Cidade = "Diamantina",
        Estado = "MG",
        DataNascimento = new DateTime(1998, 9, 7),
        Senha = "abcd1234"
    };

    var resultadoValidacao = validarUsuario.TestValidate(model);

    Assert.False(resultadoValidacao.IsValid);
    resultadoValidacao.ShouldHaveValidationErrorFor(item => item.Nome).WithErrorMessage("O campo 'Nome' deve ser preenchido");
    resultadoValidacao.ShouldHaveValidationErrorFor(item => item.Sobrenome).WithErrorMessage("O campo 'Sobrenome' deve possuir mais de 2 letras");
    resultadoValidacao.ShouldHaveValidationErrorFor(item => item.Celular).WithErrorMessage("Limite de números excedidos");
    resultadoValidacao.ShouldHaveValidationErrorFor(item => item.Cep).WithErrorMessage("O campo 'CEP' deve ser preenchido");
}
```

Fonte: Próprio autor.

Figura 108 – Retorno do teste onde o UsuarioModel é inválido



Fonte: Próprio autor.

7.1.2 Testando criação do usuário

O próximo teste é implementado para validar a criação de um usuário. Como o teste envolve o Controller do sistema ele será realizado na classe UsuarioControllerTest. Nesta classe será realizado o cenário de teste onde um usuário será criado com sucesso e para que isso aconteça, o Model de usuário deve ser preenchido corretamente.

No contexto apresentado, dentro da classe UsuarioRepositoryMock é criado um método chamado **CriarNovoUsuario**. Este método recebe como parâmetro um Model de usuário e

retorna que a tarefa de criação do usuário foi concluída, simulando a operação que seria realizada no banco de dados (Figura 109).

Figura 109 – Método de CriarNovoUsuario presente na classe de Mock

```
public Task CriarNovoUsuario(UsuarioModel usuario)
{
    return Task.CompletedTask;
}
```

Fonte: Próprio autor.

Para testar que o usuário foi criado com sucesso, Douglas desenvolve o método **SeCriarNovoUsuario_QuandoUsuarioDarlan_TaskCompleted** (Figura 110).

Figura 110 – Teste de criação de um usuário

```
[Fact]
0 references | Darlan Souza - DTI, 20 days ago | 1 author, 1 change
public void SeCriarNovoUsuario_QuandoUsuarioDarlan_TaskCompleted()
{
    UsuarioController controller = UsuarioController();

    var model = new UsuarioModel()
    {
        Nome = "Darlan",
        Sobrenome = "Souza Silva",
        Celular = "(99) 99999-9999",
        Email = "darlan@email.com",
        Rua = "Rua A",
        Numero = 10,
        Complemento = "Casa",
        Bairro = "Saúde",
        Cidade = "Diamantina",
        Cep = "39100-000",
        Estado = "MG",
        DataNascimento = new DateTime(1998, 9, 7),
        Senha = "abcd1234"
    };

    var retorno = controller.CriarNovoUsuario(model);

    Assert.NotNull(retorno);
    Assert.True(retorno.IsCompleted);
}
```

Fonte: Próprio autor.

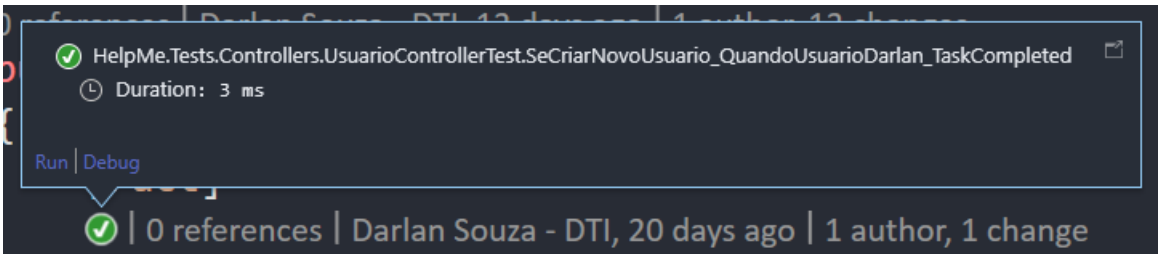
De início, o método de teste apresentado na Figura 110 preenche o UsuarioModel com informações corretas do usuário, em seguida esse Model é enviado para o método CriarNovoUsuario apresentado na Figura 109. O método presente na classe UsuarioRepositoryMock retorna que a tarefa de criação do usuário foi concluída. Em seguida é validado se o retorno

vindo da criação do usuário é diferente de NULL e se o retorno possui uma operação que foi finalizada.

O uso do Mock se encaixa perfeitamente no caso anterior, onde uma operação de banco será realizada, mas não pode ser feita no banco de dados real da aplicação. Este é um exemplo claro da necessidade de utilização de Mocks em testes.

Após implementar o teste que realiza a criação de usuário no sistema, Douglas também o executa. Analisando o resultado percebe que o teste atende o que ele havia planejado (Figura 111).

Figura 111 – Retorno do teste de validação na criação de um usuário



Fonte: Próprio autor.

Da mesma forma que Douglas realizou os testes de sua história, Bruno e Fernanda também deverão realizar os testes das demais histórias presentes na Sprint. Após a conclusão dos testes, terminam as atividades mapeadas para a Sprint 1 da plataforma Help Me.

7.2 Repositórios de código fonte

Normalmente, os códigos são desenvolvidos nas máquinas dos programadores para posteriormente serem colocados em um servidor, só após estas operações que o software poderá ser utilizado pelos usuários. Para que os códigos desenvolvidos não permaneçam apenas na máquina dos programados, correndo risco de serem perdidos, existem os repositórios de código fonte que possibilitam o controle das versões de uma aplicação.

Os repositórios de código fonte trabalham com o armazenamento de código e possibilitam que todos os envolvidos no desenvolvimento de uma aplicação tenham acesso ao que está sendo criado pelo restante do time. Isso é possível graças a unificação do código desenvolvido em um só lugar, que é o repositório. Através do repositório todos os desenvolvedores da aplicação poderão atualizá-lo constantemente com as modificações que estão inserindo na aplicação, estas modificações são fruto da criação de novas funcionalidades, alteração das já existentes e correção de erros.

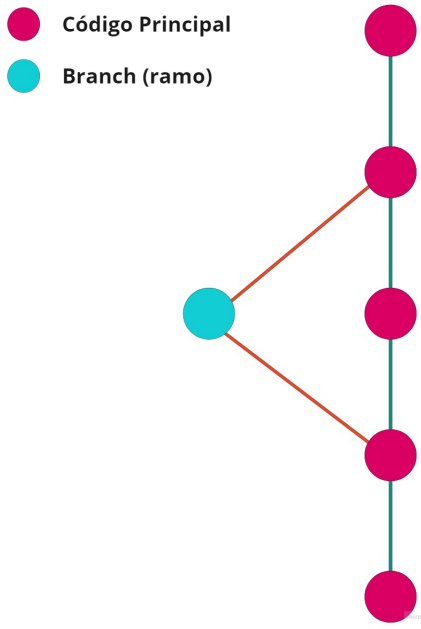
Para auxiliar neste controle de versões do código temos o **Git**. O Git é um sistema de controle de versões amplamente utilizado, uma de suas características marcantes é que ele também é de código aberto (GIT, 2020) (Figura 112). Através do Git é possível criar **branches** (ou ramos) de uma aplicação, estes branches pegam o código principal de uma aplicação e o clonam criando, dessa forma, uma segunda versão do software que possui a mesma estrutura da versão principal, com os mesmos arquivos e funcionalidades.

Figura 112 – Logo do Git

Fonte: (GIT, 2020)

Como o *branch* reflete o código principal, ele pode ser alterado sem que o código principal sofra mudanças. Após as alterações na cópia (*branch*), o ramo de código criado será unido (ou mesclado) com o código principal e, dessa forma, as alterações feitas no *branch* serão enviadas para o código principal. A Figura 113 mostra a geração de um branch a partir do código principal e posteriormente a união do *branch* ao código principal.

Figura 113 – Fluxo do Git



Fonte: Próprio autor.

Para o desenvolvimento do módulo de usuário da plataforma Help Me os desenvolvedores contaram com apoio de um repositório para o versionamento de código, dessa forma agilizaram e otimizaram seus trabalhos.

7.3 Término da sprint 1

Ao término da Sprint, os resultados produzidos são mostrados a Maria e Valter (Área de Negócio) para que validem o que foi criado. Se por algum motivo eles identificarem algo que deve ser alterado, Pedro (Product Owner, ou Dono do Produto) irá definir estas questões como prioritárias para a próxima Sprint, onde um ciclo de desenvolvimento ocorrerá novamente. Estes ciclos são repetidos até que o produto final seja entregue.

Antes da Planning, que irá definir o que será feito na Sprint 2, Samantha (Scrum Master) realiza a retrospectiva da Sprint 1 com todo o time para que questões referentes a Sprint concluída, como os pontos positivos e negativos, possam ser levantados. Com essa retrospectiva, planos de ação são gerados para sanar possíveis dificuldades ou problemas enfrentados pelo time durante o andamento da Sprint 1.

Após a avaliação de Maria e Valter e a retrospectiva de Samantha, todo o time parte agora para Planning, onde as novas demandas da Sprint 2 poderão ser levantadas para que a equipe atue novamente e entregue ainda mais valor aos clientes.

8 CONCLUSÃO

O presente trabalho se deu através de uma contextualização geral sobre o desenvolvimento Web, com apresentação de sua história e de seu uso na sociedade atual. Com isso, conclui-se, por fim, o desenvolvimento do módulo de uma aplicação com uso de tecnologias comumente utilizadas atualmente em diversos projetos.

O conteúdo exposto no decorrer do texto possui uma estrutura fácil de ser entendido por diversos públicos, o que gera muito valor para a sociedade. O texto visa sanar o problema de que informações relacionadas à Tecnologia da Informação (TI) se encontram muito dispersas na Internet, o que dificulta o aprendizado das pessoas nesta área de conhecimento.

Com as informações expostas, o leitor pode conhecer um pouco mais sobre o mundo da programação Web e ter uma noção de quais atividades envolvem essa prática. Sendo assim, o trabalho apresentou como se dá o processo de criação de uma aplicação Web deste o projeto inicial até sua implementação, desmistificando os passos que envolvem a construção de um software. Em relação aos trabalhos futuros, a pretensão é implementar os demais módulos da plataforma Help Me, a fim de gerar mais conhecimento sobre o desenvolvimento de uma aplicação em um contexto organizacional.

REFERÊNCIAS

ALBUQUERQUE, A. A. d.; AMARAL, B. M.; BENTO, F. d. C.; JANNUZZI, F. S.; SPENCER, F.; AMORIM, M. E. d.; FELIPPE, M. R.; MAESTRELLI, M. **Tópicos sobre Protocolos de Comunicação**. [S.l.]: CAT, 2003.

ALMEIDA, G. A. M. d. Fatores de escolha entre metodologias de desenvolvimento de software tradicionais e ágeis. p. 1–105, 2017.

BOOTSTRAP. **Introduction**. 2020. <https://getbootstrap.com/docs/4.5/getting-started/introduction/>. Acessado em 09/06/2020.

CAMARGO, D. A. d. O. Estudo das técnicas de desenvolvimento web e validação deste estudo com um portal para recicladores e produtores de reciclados. 2009.

CARVALHO, M. S. R. M. d. A trajetória da internet no brasil: Do surgimento das redes de computadores À instituição dos mecanismos de governança. 2006.

CASTELLS, M. **A Galáxia da Internet**. ZAHAR, 2001. ISBN 9788571107403. Disponível em: <https://zahar.com.br/sites/default/files/arquivos/trecho_-_a_galaxia_da_internet.pdf>.

DJANGO. **Django at a glance**. 2020. <https://docs.djangoproject.com/en/3.0/intro/overview/>. Acessado em 02/06/2020.

FLUENTVALIDATION. **FluentValidation**. 2020. <https://docs.fluentvalidation.net/en/latest/>. Acessado em 26/09/2020.

GIT. **git –everything-is-local**. 2020. <https://git-scm.com/>. Acessado em 19/12/2020.

INITIATIVE, O. S. **To promote and protect open source software and communities...** 2020. <https://opensource.org/>. Acessado em 02/11/2020.

JSON. **Introdução ao JSON**. 2020. <https://www.json.org/json-pt.html>. Acessado em 30/06/2020.

JURNO, A. C.; SILVA, P. I. R.; SILVA, T. I. R. Notas sobre a história da internet interfaces que expandem a grande rede. 2017.

LINS, B. F. E. A evolução da internet: uma perspectiva histórica. 2013.

LUCCIO, F. D. Do iluminismo à web semântica reflexões sobre a comunicação com base em uma única língua. 2010.

MATERIALIZECSS. **About**. 2020. <https://materializecss.com/about.html>. Acessado em 11/06/2020.

MEDIUM. **Backend Development: an Introduction for Frontend Developers**. 2020. <https://medium.com/@bretcameron/backend-development-an-introduction-for-frontend-developers-ce7fb36848df>. Acessado em 19/12/2020.

MELLO, S. F. M. Comunicação e organizações na sociedade em rede. 2016.

MICROSOFT. **Baixar o SQL Server Management Studio (SSMS)**. 2020. <https://docs.microsoft.com/pt-br/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>. Acessado em 20/09/2020.

MICROSOFT. **Documentação do .NET Core**. 2020. <https://docs.microsoft.com/pt-br/dotnet/core/>. Acessado em 03/06/2020.

MICROSOFT. **Getting Started**. 2020. <https://code.visualstudio.com/docs/>. Acessado em 20/09/2020.

MICROSOFT. **Introdução às Razor páginas no ASP.NET Core**. 2020. <https://docs.microsoft.com/pt-br/aspnet/core/razor-pages/?view=aspnetcore-3.1&tabs=visual-studio>. Acessado em 24/09/2020.

MICROSOFT. **.NET Core versus .NET Framework para aplicativos de servidor**. 2020. <https://docs.microsoft.com/pt-br/dotnet/standard/choosing-core-framework-server>. Acessado em 02/06/2020.

MICROSOFT. **Padrão ASP.NET MVC**. 2020. <https://dotnet.microsoft.com/apps/aspnet/mvc>. Acessado em 22/09/2020.

MICROSOFT. **System.Data.SqlClient Namespace**. 2020. <https://docs.microsoft.com/pt-br/dotnet/api/system.data.sqlclient?view=dotnet-plat-ext-3.1>. Acessado em 24/09/2020.

MONGODB. **The database for modern applications**. 2020. <https://www.mongodb.com/>. Acessado em 01/07/2020.

MOURA, E. L. d.; MOREIRA, M. A. R. Metodologias de desenvolvimento de software. 2012.

MURUGESAN, S.; GINIGI, A. Web engeneering: Principles and techniques. 2005.

NAIK, U.; SHIVALINGAIAH, D. Comparative study of web 1.0, web 2.0 and web 3.0. In: . [S.l.: s.n.], 2009.

NETO, J. B. P. **Redes de computadores**. [S.l.: s.n.], 2014.

NODE. **Run JavaScript Everywhere**. 2020. <https://nodejs.dev/>. Acessado em 04/06/2020.

NPM. **Build amazing things**. 2020. <https://www.npmjs.com/>. Acessado em 07/06/2020.

ORACLE. **O que É um Banco de Dados Relacional?** 2020. <https://www.oracle.com/br/database/what-is-a-relational-database/>. Acessado em 25/06/2020.

PRESSMAN, R. **Engenharia de software**. [S.l.: s.n.], 2002.

PRESSMAN, R. **Engenharia de software**. McGraw-Hill, 2006. ISBN 9788586804571. Disponível em: <https://books.google.com.br/books?id=MNM6AgAACAAJ>.

PRESSMAN, R. S. **Engenharia de Software**. AMGH, 2011. ISBN 9788563308337. Disponível em: <https://books.google.com.br/books?id=eRIOuQAACAAJ>.

PURE. **Get Started**. 2020. <https://purecss.io/start/>. Acessado em 10/06/2020.

RIBEIRO, R. D.; RIBEIRO, H. d. C. e. S. **Gerenciamento de projetos com métodos ágeis**. [S.l.: s.n.], 2015. ISBN 9788591910212.

ROCKENBACH, D. A.; ANDERLE, N.; GRIEBLER, D.; SOUZA, S. Estudo comparativo de bancos de dados nosql. 2018.

SANTOS, D. S. d. Sistema de recomendação de frameworks para desenvolvimento multiplataforma em dispositivos móveis. 2018.

SCHWABER, K.; SUTHERLAND, J. The scrum guide. 2017.

SERIES, R. C. M. **Agile Estimating and Planning**. [S.l.]: PEARSON, 2005. ISBN 0-13-147941-5.

SOMMERVILLE, I. **Engenharia de Software**. [S.l.]: PEARSON, 2003. ISBN 9788579361081.

SOMMERVILLE, I. **Engenharia de software**. PEARSON BRASIL, 2011. ISBN 9788579361081. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>.

SUTHERLAND, J. **Scrum: a arte de fazer o dobro do trabalho na metade do tempo**. [S.l.]: leya, 2014. ISBN 9788544100882.

TANENBAUM, A. S. **Redes de computadores**. [S.l.]: Campus, 2003. ISBN 9788535211856.

TUTORIAL, D. **Dapper**. 2020. <https://dapper-tutorial.net/dapper>. Acessado em 26/09/2020.

VICENTINI, L.; LANZONI, E.; FRANZOTTI, V.; YONENAGA, W. H. Introdução da tecnologia de voz sobre ip em redes corporativa. 2005.


UFVJM