Fatec São Paulo
Faculdade de Tecnologia de São Paulo
Departamento de Tecnologia da Informação

ALEXANDRE JOSÉ DE SOUSA

SELEÇÃO DE STACKS PARA APLICAÇÕES WEB

SÃO PAULO
2023

Faculdade de Tecnologia de São Paulo

ALEXANDRE JOSÉ DE SOUSA

SELEÇÃO DE STACKS PARA APLICAÇÕES WEB

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas
Orientador: Professor, Mestre e Doutor Aristides Novelli Filho

SÃO PAULO
2023

DEDICATÓRIA

Dedico à Lucila e a Adorico.

AGRADECIMENTOS

A princípio gostaria de agradecer ao meu orientador e professor Aristides Novelli Filho, não apenas por ter aceito me orientar e ter despendido tempo auxiliando na melhoria deste trabalho, mas também por ter escutado minhas ideias sobre quais temas abordar para este trabalho e fornecer sua opinião, ainda quando eu era discente em seu curso de Engenharia de Software II.

Aos meus colegas que conheci durante o período que cursei Analise e Desenvolvimento de Sistema na Faculdade de Tecnologia de São Paulo. Compartilhamos momentos de dificuldade em determinadas ocasiões do curso, porém, isso nos proporcionou a oportunidade de auxiliarmos uns aos outros, e assim construirmos um sólido vínculo, onde todos pudemos desfrutar de bons momentos.

A duas pessoas em especial, pelas quais tenho imenso carinho e estima, Michel Barros e Adriana Santana. Muito obrigado por terem me fornecido a oportunidade que eu precisava para concretizar uma fase de importante mudança em minha vida. Dessa porta que vocês me abriram coisas maravilhosas surgiram, e todas elas pertencentes à bela estirpe das coisas que nascem das nobres ações e do coração.

Por fim, ao Luiz Fernando de Moraes, por muito ter me ensinado sobre a engenharia, construção e desenvolvimento de software na prática. Além, claro, de todo o auxílio fornecido quando surgia uma problemática deveras complexa. Apesar das brincadeiras, possuo grande respeito por ti.

A todos, sou muito grato.

RESUMO

O trabalho analisa o cenário geral de desenvolvimento e posterior manutenção de uma aplicação web, confrontando-o com fatores relevantes na seleção do conjunto de soluções tecnológicas – stack – para seu desenvolvimento e manutenção, tencionando optar pelo stack mais adequado, viável e com maior probabilidade de êxito no projeto. O cenário foi submetido a uma análise, iniciando pelo tipo de projeto, dimensão e complexidade, seguido pelas especificações do sistema e suas funcionalidades, tempo para colocação deste no mercado, sua escalabilidade e extensibilidade, manutenção futura, recursos humanos para o projeto e pós-projeto, culminando na seleção e exame de possíveis stacks, e terminado na seleção de um conjunto de soluções tecnológicas adequado baseando-se no custo de sua adoção. Como resultado desta análise foi obtido o stack com a maior probabilidade de sucesso na implantação do projeto do sistema supracitado.

Palavras-chave: Desenvolvimento. Engenharia de Software. Stack. Web.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RAM – Random Access Memory

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

JRE – Java Runtime Environment

PHP – Hypertext Preprocessor

LAMP – Linux, Apache, MySQL, PHP/Perl/Python

HTTP – Hypertext Transfer Protocol

NCSA – National Center for Supercomputing Applications

MVP – Minimum Viable Product

CMS – Content Management System

SRS – Software Requirements Specification

NPM – Node Package Manager

MVC – Model, View, Controller

SPA – Single Page Application

SUMÁRIO

1 INTRODUÇÃO

Hoje comumente usadas por milhões de pessoas por dia, essa rotina faz com que por vezes passe despercebido o fato das aplicações web serem de extrema importância para diversas áreas da sociedade atual. Tais aplicações têm sido um importante instrumento no desenvolvimento de grandes setores industriais e empresas de diversos ramos, bem como na melhoria do lazer e qualidade de vida das pessoas.

Empresas como Netflix, Amazon, Airbnb e Nubank são exemplos de destaque em seus segmentos que fazem o uso em larga escala de sistemas web. Assim, o desenvolvimento e manutenção desses sistemas são de extrema importância para o negócio, visto o quão digitais são essas companhias.

O processo de operacionalizar esses sistemas passa por diversas etapas de projeto antes mesmo que sejam criadas as primeiras linhas de código, uma vez que o seu projeto é composto por um conjunto de projetos que envolvem outras áreas além das diretamente relacionadas à tecnologia (PRESSMAN, 2011).

Um ponto importante a se considerar durante o planejamento é a decisão sobre qual conjunto de soluções tecnológicas optar para desenvolvimento e posterior manutenção da aplicação, em razão não apenas da quantidade de fatores de projeto que impactam essa determinação, mas também pela quantidade de ferramentas disponíveis ao uso. Assim, equipes que se abstém à análise e estudo para decidir qual o melhor conjunto adotar, podem causar riscos de desperdício de tempo, dinheiro e recursos à empresa, bem como ter dificuldades para encontrar desenvolvedores aptos para fazer a manutenção da aplicação, futura necessidade de reescrever os códigos para um outro stack, ter problemas de segurança e dificuldades para implementar novas funcionalidades (POCZWARDOWSKI, 2020).

Este trabalho vem apresentar uma metodologia através da qual é feita uma análise sistemática fundamentada nos fatores de sistema e projeto relevantes para a seleção de stacks no desenvolvimento e manutenção de aplicações web, visando apoiar a escolha do conjunto de soluções tecnológicas mais adequado e viável para o desenvolvimento de um sistema. Com isso o trabalho colabora com o fornecimento de informações sobre diretrizes e limitação do campo de escolha das soluções tecnológicas disponíveis, tendo em decorrência desse processo a opção de stack com maior probabilidade de êxito no projeto.

A metodologia empregada dividiu-se em quatro partes. Na primeira foram definidos os fatores que possuem grande influência na seleção das soluções tecnológicas, fatores estes selecionados, tratados, organizados e ordenados com base em pesquisas

bibliográficas de conceituados autores e empresas de desenvolvimento de software. Na segunda foram analisados os fatores selecionados na etapa anterior, objetivando como resultado métricas que sirvam como base para escolha do conjunto de soluções tecnológicas e delimitem o conjunto universo de opções. Na terceira, selecionados conjunto(s) adequado(s) para operacionalização da aplicação, baseando-se nas nos resultados das análises das métricas previamente obtidas. Na quarta e última etapa, foi analisado e selecionado o stack mais viável dentre os eleitos na terceira etapa, seleção esta baseada no custo de sua implementação.

1.1 Objetivo

1.1.1 Objetivo Geral

Analisar o cenário geral de desenvolvimento e posterior manutenção de uma aplicação web, confrontando-o com fatores relevantes na seleção do conjunto de soluções tecnológicas – stack – para seu desenvolvimento e manutenção, com o intuito de optar pelo stack mais adequado, viável e com maior probabilidade de êxito no projeto ao final da análise.

1.1.2 Objetivo Específico

Tencionando lograr o objetivo principal, torna-se necessário a execução de objetivos específicos. São eles:

- Definir os fatores de grande importância e impacto na seleção do conjunto de soluções tecnológicas;
- Analisar o cenário da aplicação web, confrontando-o com fatores de grande impacto, obtendo como produto métricas para serem usadas como base de escolha das soluções tecnológicas;
- Selecionar soluções adequadas para operacionalização da aplicação;
- Analisar e selecionar o stack mais viável baseando-se no custo de sua implementação.

1.2 Justificativa

Este trabalho justifica-se pela sua colaboração com informações resultantes de uma análise sistemática fundamentada em fatores de grande impacto para a determinação do conjunto de soluções tecnológicas a ser usado para o desenvolvimento e manutenção de

aplicações web, com o intuito apoiar a escolha do stack mais adequado, viável e com maior probabilidade de êxito no projeto.

1.3 Metodologia

O presente trabalho será elaborado através do desenvolvimento de um modelo baseado em fatores de grande importância para a escolha adequada e viável do conjunto de soluções tecnológicas a ser empregado em uma aplicação web; fatores estes selecionados, tratados, organizados e ordenados com base em pesquisas bibliográficas de conceituados autores e empresas de desenvolvimento de software. O modelo elucida quais são esses fatores, a delimitação que o produto resultante da análise de um cenário de uma aplicação web baseado nesses fatores engendra no conjunto universo de soluções tecnológicas, o subconjunto de soluções adequadas resultante da delimitação supracitada, e a seleção do stack mais viável dentre as opções adequadas que compõem o referido subconjunto, baseando-se no custo de sua implementação. A metodologia, em sumo, é separada nas seguintes etapas.

Na primeira são definidos os fatores que possuem grande influência na seleção. Tais fatores são baseados em bibliografias e experiências empíricas de autores e empresas de software durante o desenvolvimento de sistemas, em especial aplicações web.

Na segunda etapa é analisado o cenário de uma aplicação web baseando-se nesses fatores, objetivando através do resultado dessa análise obter métricas que delimitarão dentro dos possíveis conjuntos de soluções tecnológicas os mais adequados.

Na terceira são elegidos os conjuntos mais adequados baseando-se nas métricas decorrentes da etapa anterior.

Na quarta e última etapa é selecionado entre os mais adequados o stack mais viável, baseando-se no custo de sua implementação.

1.4 Estrutura do Trabalho

Quanto ao que tange à estruturação deste trabalho, este foi dividido em oito capítulos. No primeiro são apresentados os objetivos geral e específicos, assim como sua justificativa, um breve resumo da metodologia e a estrutura do trabalho. O segundo capítulo retrata, de forma concisa, uma súmula do início das aplicações web e a popularização do termo stack, iniciando-se na década de 90 e seguindo até meados dos anos 2000. No terceiro o tema abordado são as características de projetos web e suas especificidades. No quarto são

apresentadas as premissas para a escolha de um conjunto de soluções tecnológicas, tratando dos fatores de sistema e projeto que tem grande impacto na eleição desses conjuntos dentro contexto das aplicações web. O quinto apresenta alguns exemplos de stacks para aplicações web, bem como suas tecnologias componentes e vantagens. No sexto é apresentado um protótipo de avaliação, onde é exibido um cenário fictício, com fatores os quais serão analisados visando obter as métricas para base de escolha do conjunto de soluções tecnológicas e determinação do mais adequado e viável. O sétimo apresenta os resultados da aplicação da metodologia, com as métricas e stack obtidos. No oitavo e último capítulo apresentam-se as considerações finais, onde é feita uma reflexão sobre o desenvolvimento do trabalho, os óbices encontrados, o confronto entre as expectativas no início do trabalho e os resultados obtidos no final dele e possíveis trabalhos/estudos que poderiam dar sequência à metodologia apresentada ou deste fazer uso.

2 INÍCIO DAS APLICAÇÕES WEB E O TERMO STACK

> Nos primórdios da computação, os usuários interagiam com as aplicações em servidores por meio de terminais, ou thin clients. Isso era eficaz, pois significava que o aplicativo estava rodando em um só lugar, e todos estavam usando o mesmo. A ascensão do computador pessoal e desktops individuais mais potentes permitiram uma mudança para fat clients, onde cada vez mais a lógica de negócios era executada em programas distribuídos para os desktops dos usuários finais. Isso causou problemas com a distribuição desses aplicativos para todas as pessoas que deles faziam uso, bem como mantinham diferentes versões em cada dispositivo. Era uma maneira lenta de fazer as coisas. Havia muitos benefícios nos computadores se tornarem menores e mais portáteis, permitindo que surgissem nas casas das pessoas, e não apenas em grandes instalações conectadas. Porém, um mundo sempre conectado ainda estava muito distante. (NORTHWOOD, 2018, p. 4)

Aplicações web, desde a sua concepção, são de extrema importância para as atividades comuns à sociedade, bem como o desenvolvimento da internet como um todo. Tendo seu uso a priori aplicado aos departamentos de tecnologia de organizações, expandiram-se para lares e demais nichos, tornando-se populares nas décadas que seguem os anos de 1990 e 2000. Assim, hoje, muitos dos serviços consumidos por usuários para suas necessidades são baseados nesse tipo de aplicação.

Tendo este modelo sua idealização na década de 1990, as aplicações usadas até então faziam uso de um arquétipo no qual era necessário fazer a instalação do código em cada um dos computadores dos usuários, de forma local, com o programa pré-compilado servindo como interface para o usuário. Uma vez que houvesse uma atualização do código, era necessário que este fosse atualizado em cada uma das máquinas onde foi instalado, o que acrescentava nos custos de suporte e diminuía a produtividade das equipes. Além, as aplicações da época eram comumente desenvolvidas baseadas em uma arquitetura e sistema operacional, e fazer o porte delas era deveras oneroso.

Buscando solucionar os problemas supracitados e com o lançamento do primeiro navegador em 1990, por Tim Berners-Lee, programadores começaram a desenvolver aplicações que eram executadas através do navegador do usuário do sistema, sendo esta ação

colaborada posteriormente com a criação da Hypertext Markup Language (HMTL), também por Tim Berners-Lee, no ano de 1993, e da Cascading Style Sheets (CSS) no ano seguinte, por Håkon Wium Lie. Nos primeiros modelos desenvolvidos com esse intuito, os usuários recebiam páginas web estáticas de um servidor, o que dificultava a experiência de interatividade, uma vez que quando ocorria alguma alteração na página, era demandado certo tempo para o envio das informações ao servidor e aguardo do seu retorno. Isso ocorria não apenas por serem os primeiros modelos desenvolvidos nesse segmento, mas também pela tecnologia disponível na época, onde, a critério de exemplo, um microprocessador operava a aproximadamente 50 megahertz, e os computadores pessoais tinham apenas alguns megabytes de memória RAM.

No ano de 1995, Java 1.0 foi lançado pela Sun Mycrosystems. Foi um grande lançamento, com a promessa de ser uma nova linguagem de programação segura, orientada a objetos, com sintaxe familiar e independente de plataformas. Um notável aspecto trazido por ela eram os chamados applets, programas Java que executavam com uma interface gráfica para o usuário em uma página web.

A promessa do applet Java era que programas poderiam ser escritos apenas uma vez e executados em qualquer plataforma que possuísse um Java Runtime Environment (JRE) e um navegador web. Essa filosofia *write once, run anywhere* visava fazer da distinção entre executar programas em um sistema operacional Mac, Windows ou Unix irrelevante. O Netscape Navigator foi o primeiro navegador web a suportar os applets Java (KOPEC, 2014).

No mesmo ano, o programador Brendan Eich criou a linguagem JavaScript. Na época trabalhando na Netscape, o navegador da empresa foi o primeiro a ser lançado com suporte a essa nova linguagem. Diferente dos applets Java, o JavaScript deveria ser uma linguagem leve de script.

O nome JavaScript, a época de lançamento, foi usado como uma estratégia de marketing, fazendo uso da fama que a linguagem Java tinha na época. Porém, esse nome foi padronizado como ECMAScript pela Ecma International posteriormente.

Conforme essa linguagem ganhava popularidade rapidamente, logo começou a ser adaptada para os demais navegadores web. Apesar de ser um marco importante, essas adaptações trouxeram consigo também problemas de incompatibilidade, que foram sendo corrigidos ao longo das novas versões desses navegadores.

Em um contexto onde os applets ainda eram lentos em seu carregamento e a linguagem JavaScript não era sofisticada o suficiente, em 1996, foi apresentado o Macromedia Flash. O maior desafio dessa tecnologia era melhorar as engines e APIs do

JavaScript, permitindo que programadores inserissem animações em suas aplicações. Mais tarde, a Adobe Flash introduziu o uso de áudio e vídeo em tais animações, interações as quais não exigiam comunicação com o servidor.

Apesar de referências anteriores, o termo Stack começa a ser popularizado a partir de 1998, com o LAMP Stack. Stack é um termo que refere-se a um conjunto de componentes, isto é, soluções tecnológicas, que são utilizadas conjuntamente para desenvolver e/ou dar suporte/manutenção às aplicações. LAMP trata-se de um acrônimo para Linux – sistema operacional open source, criado por Linus Trovalds em 1991, Apache HTTP Server – software que atua como servidor web, criado por Rob McCool em 1995, época em que fazia parte da NCSA, MySQL – sistema gerenciador de banco de dados relacionais, criado por uma empresa sueca e lançado em 1994 - e PHP/Perl/Python - representando a linguagem de programação usada, um stack para aplicações web. Nesse contexto, Linux é o termo que representa os sistemas operacionais que utilizam o Kernel Linux, Apache HTTP Server o software que processa as solicitações via HTTP, MySQL o sistema gerenciador do banco de dados relacional, que armazenará as informações processadas, e PHP/Perl/Python a linguagem de programação utilizada no server-side.

Com a evolução dos hardwares e das tecnologias existentes até então, bem como o surgimento de novos navegadores e tecnologias, o campo de desenvolvimento de aplicações web tornou-se cada vez mais rico e mais complexo, fornecendo não apenas inúmeras ferramentas e componentes para que os desenvolvedores pudessem desenvolver sistemas, mas também um maior desafio e exigência de planejamento mais apurado quando da escolha do stack, isto é, do conjunto de soluções tecnológicas a serem usadas nos projetos.

3 PROJETO DE APLICAÇÕES WEB

Um projeto de uma aplicação web abrange atividades técnicas e não técnicas, envolvendo diversos profissionais, como engenheiros de software e designers gráficos. A criação e desenvolvimento desse tipo de projeto permite criar um modelo o qual pode ser avaliado em relação a sua qualidade e aperfeiçoado antes da geração de código e conteúdo. Nesse processo há grande participação e contribuição dos usuários finais, visando o melhor projeto possível, obtendo assim um artefato que contempla, como apresentado por Pressman (2011, p. 338), “um modelo de projeto abrangendo questões de conteúdo, estética, arquitetura, interface, navegação e de projeto de componentes”.

O projeto é a atividade da engenharia que conduz a um produto de alta qualidade. Em relação às aplicações web, os atributos gerais mais relevantes à qualidade dessas são: usabilidade, funcionalidade, confiabilidade, eficiência e facilidade de manutenção (OLSINA apud PRESSMAN, 2011, p. 339). Mais tarde, Jeff Offut (2022 apud PRESSMAN, 2011, p. 339) acrescentou outros quatro atributos à lista: segurança, disponibilidade, escalabilidade e tempo para colocação no mercado. Tais atributos sintetizam um norte para o desenvolvimento e manutenção de sistemas desse tipo no longo prazo.

Em sua coluna sobre projetos para a web, Jean Kaiser (2002 apud PRESSMAN, 2011, p. 341) sugere um conjunto de objetivos de projeto que podem ser aplicados a praticamente qualquer aplicação web, independente do domínio de aplicação, tamanho ou complexidade. São eles: simplicidade, consistência, identidade, robustez, navegabilidade, apelo visual e compatibilidade.

Como outros projetos, os relacionados às aplicações web são também compostos por etapas. Roger Pressman (PRESSMAN, 2011) apresenta uma pirâmide de projeto como representação gráfica das etapas que compõem tais aplicações e suas relações.

Figura 3.1 – Pirâmide de projeto para aplicações web

Fonte: PRESSMAN (2011)

Tais etapas são descritas a seguir.

3.1 Etapas que Compõem Uma Aplicação Web

O projeto de aplicações web abrange seis etapas principais, as quais são orientadas por informações obtidas durante a modelagem de requisitos. São elas:

- **Projeto de componentes:** representa a estrutura interna detalhada dos elementos funcionais da aplicação;
- **Projeto de arquitetura:** se concentra na estrutura geral de hipermídia de todos os objetos de conteúdo e funções;
- **Projeto de navegação:** define como o usuário final navega pela estrutura de hipermídia;
- **Projeto de conteúdo:** usa o modelo de conteúdo desenvolvido durante a análise como base para estabelecer o projeto de objetos do conteúdo;
- **Projeto estético:** estabelece o layout que o usuário final verá;
- **Projeto de interface:** estabelece mecanismos de layout e interação que definem a interface do usuário.

3.1.1 Projeto de Componentes

As aplicações web atuais oferecem funções de processamento sofisticadas, tais como processamento localizado para gerar recursos de navegação e conteúdo de forma dinâmica, fornecer recursos de cálculo ou processamento de dados apropriados para o campo da aplicação, fornecer consultas e acesso a banco de dados e estabelecer interfaces de dados com sistemas corporativos externos. Para que essas capacidades sejam alcançadas, é necessário projetar e construir componentes de programa.

Os métodos de projeto usados para software tradicionais se aplicam em sua maioria às aplicações web quanto aos componentes, com poucas, quando há, modificações. O ambiente de implementação, linguagens de programação, padrões de projeto, estruturas e software podem variar um pouco, porém, a abordagem de projeto geral permanece a mesma (PRESSMAN, 2011).

3.1.2 Projeto de Arquitetura

O projeto de arquitetura está ligado aos objetivos estabelecidos para uma aplicação web, ao conteúdo a ser apresentado, aos usuários que visitarão a página e à filosofia de navegação que foi estabelecida. Com isso, são identificadas a arquitetura de conteúdo, que foca na maneira pela qual os objetos de conteúdo são estruturados para apresentação e navegação, e a arquitetura da aplicação web, que lida com a maneira pela qual a aplicação é estruturada para administrar a interação com o usuário, tratar tarefas de processamento interno, navegação efetiva e apresentação do conteúdo.

Em grande parte dos casos, o projeto de arquitetura é conduzido em paralelo com os projetos de interface, estético e de conteúdo. Como a arquitetura pode ter uma forte influência sobre a navegação, as decisões tomadas durante as etapas de projeto influenciam o trabalho conduzido durante o projeto de navegação (PRESSMAN, 2011).

3.1.3 Projeto de Navegação

O projeto de navegação é baseado na arquitetura que a aplicação tem como definida e em seus componentes. Assim, são definidos os percursos de navegação que permitirão aos usuários acessarem o conteúdo e as funções do sistema. Para tanto, é

necessário identificar a semântica de navegação para diferentes usuários do sistema e definir a mecânica para atingir a navegação (PRESSMAN, 2011).

3.1.4 Projeto de Conteúdo

O projeto de conteúdo aborda duas tarefas de projeto diferentes. A priori são desenvolvidos uma representação de projeto para objetos de conteúdo e os mecanismos necessários para estabelecer seus relacionamentos. Além, são criadas informações em um objeto de conteúdo específico. Esta última tarefa também pode ser executada por profissionais não relacionados à área de desenvolvimento, como redatores publicitários (PRESSMAN, 2011).

3.1.5 Projeto Estético

O projeto estético, também conhecido como design gráfico, é o esforço artístico que complementa os aspectos técnicos do projeto de aplicações web. Nessa fase são estudadas e definidas concepções de layouts para interação usuário/sistema. Sem essa etapa, o sistema poderia ser funcional, porém sem a característica atrativa para os usuários.

Para o desenvolvimento dessa etapa faz-se necessário o entendimento de quem serão os usuários finais e a interpretação de qual identidade visual eles têm preferência (PRESSMAN, 2011).

3.1.6 Projeto de Interface

Quando o usuário interage com a aplicação, é aplicado um conjunto de princípios fundamentais e diretrizes de projeto, como reduzir a carga de memória do usuário e tornar a interface consistente.

Os objetivos de uma interface quando desse tipo de sistema são: estabelecer uma janela consistente para o conteúdo e a funcionalidade fornecidos pela interface, guiar o usuário através de uma série de interações com a aplicação e organização as opções de navegação e conteúdo disponíveis para o usuário. Para tal, deve-se primeiro usar a estética do projeto para estabelecer um aspecto coerente (PRESSMAN, 2011).

4 PREMISSAS PARA A ESCOLHA DE UMA STACK

Para a escolha do conjunto de soluções tecnológicas adequado para o desenvolvimento de um sistema, é necessário ir além do correto levantamento e compreensão dos requisitos, considerando também diversos outros fatores. As aplicações web, por sua vez, possuem qualidades específicas que demandam um estudo amplo sobre dependências de seus atributos, tais como o uso intensivo de rede, evolução contínua, simultaneidade, imediatismo, segurança e outros. Como citado por Roger S. Pressman (PRESSMAN, 2011): “Atualmente, as WebApps evoluíram para sofisticadas ferramentas computacionais que não apenas oferecem funções especializadas (*stand-alone functions*) ao usuário final, como também foram integradas aos bancos de dados corporativos e às aplicações de negócio.”.

Escolher o adequado stack para operacionalizar uma aplicação web pode ser uma tarefa herculana, dada natureza dinâmica e complexa da produção de software moderna (GORBACHENKO, c2022). Tal como outras atividades de mesma magnitude, não há uma solução universal para a escolha de um determinado conjunto de soluções tecnológicas para desenvolver e manter tais aplicações (SHIBU, 2021). Porém, como um dos principais aspectos para a construção de uma aplicação web com êxito, existem fatores que podem ser analisados, explorados e estudados para que se possa escolher o stack tecnológico mais apropriado para o desenvolvimento e manutenção de uma determinada aplicação web.

4.1 Fatores-Base para Decisão

Como supracitado, existem fatores que podem ser perscrutados a fim de chegar à opção de stack que possua a maior probabilidade de êxito no desenvolvimento e manutenção de uma aplicação. Nesta seção será apresentado um conjunto desses fatores que podem amparar na decisão do stack e contribuir para a melhor escolha. Abaixo, são elencados os que compõem esse conjunto:

- Tipo de projeto, dimensão e complexidade do sistema;
- Especificações do sistema e funcionalidades;
- Tempo para colocação no mercado;
- Escalabilidade e extensibilidade;
- Manutenção;
- Recursos Humanos;
- Seleção e análise de possíveis stacks;

- Custo de adoção do stack.

Esses fatores são apresentados em maiores detalhes a seguir.

4.1.1 Tipo de projeto, Dimensão e Complexidade do Sistema

O tipo de projeto, sua dimensão e complexidade são importantes premissas para a definição de quais tecnologias usar. Pequenos, médios e grandes projetos possuem diferentes necessidades a serem atendidas. Projetos de pequeno porte, como MVPs, sites de página única, portfólios, revistas digitais e outros, por vezes podem exigir, por sua natureza, um desenvolvimento mais rápido e simples. Nesses casos, podem ser usadas ferramentas como CMS ou WordPress (SHIBU, 2021), ou frameworks full stack, como Ruby on Rails (GORBACHENKO, c2022).

Projetos de médio porte, como lojas online, multimídia digital e mesmo algumas aplicações de empresas, por exemplo, podem exigir um stack mais complexo, com a possibilidade de envolver múltiplos sistemas, frameworks e integrações entre plataformas.

Projetos de grande porte e maior complexidade, tais como plataformas de trading, redes sociais e marketplaces, exigem maior velocidade de resposta, disponibilidade de serviços e escalabilidade. Dadas essas características, tais projetos demandam stacks que podem envolver diversas linguagens, além de outras condições já expressas.

4.1.2 Especificações do Sistema e Funcionalidades

Trata-se do que o sistema deve fazer, os serviços que deve oferecer e suas restrições. Tais informações podem ser obtidas através do processo de elicitação dos requisitos do sistema, que pode ter como resultado um documento tal qual a Especificação de Requisitos de Software. Como explica Sommerville (2011): “O documento de requisitos de software, às vezes chamado Especificação de Requisitos de Software (SRS — do inglês *Software Requirements Specification*), é uma declaração oficial do que os desenvolvedores do sistema devem implementar. Deve incluir tanto os requisitos de usuário para um sistema quanto uma especificação detalhada dos requisitos de sistema. Em alguns casos, os requisitos de usuário e de sistema são integrados em uma única descrição. Em outros, os requisitos de usuário são definidos em uma introdução à especificação de requisitos de sistema.”

Além, cabe fazer uma pesquisa de mercado para compreender sistemas já existentes e estudar funcionalidades adicionais que podem ser usadas como vantagens competitivas (HORIACHKO, c2022).

4.1.3 Tempo para Colocação no Mercado

Esse fator refere-se ao período entre a ideação e a disponibilidade desse sistema para o mercado. Muitas startups e empresas de software consideram este como um fator crítico, uma vez que o rápido desenvolvimento e implantação oferecem vantagem competitiva (GORBACHENKO, c2022). Como Pressman menciona no capítulo em que trata sobre projetos de webapps (PRESSMAN, 2011): “Embora o tempo para colocação de um produto no mercado não seja um verdadeiro atributo de qualidade no sentido técnico, é uma medida de qualidade do ponto de vista comercial. A primeira WebApp a atender determinado segmento de mercado em geral captura um número desproporcional de usuários finais”.

Para isso, pode-se contar com práticas de desenvolvimento ágil, como Extreme Programming – XP, Scrum e Kanban (VALENTE, 2020), soluções prontas ou frameworks que permitem uma fácil integração (SHIBU, 2021).

4.1.4 Escalabilidade e Extensibilidade

Para uma escolha assertiva sobre o stack a ser utilizado, deve-se considerar não apenas as necessidades atuais do sistema, mas também demandas subsequentes. Para tal, é necessário que haja a possibilidade de expansão do sistema de forma proporcional às demandas, como, a título de exemplo, escalar os servidores para receberem um aumento exponencial de usuários, ou estender a capacidade do sistema de processar mais solicitações sem perca de qualidade, aumento no tempo de resposta, ou mesmo que o sistema fique inoperável.

Ferramentas como Angular, React, Vue, Node.js, Golang e Ruby on Rails têm excelente escalabilidade (SHIBU, 2021).

4.1.5 Manutenção

A manutenção trata do processo geral de mudança e adaptação em um sistema depois que ele é liberado para uso. As alterações feitas no software podem ser simples mudanças para correção de erros de codificação, até mudanças mais extensas para correção de

erros de projeto, ou melhorias significativas para corrigir erros de especificação ou acomodar novos requisitos (SOMMERVILLE, 2011). Neste caso, deve-se optar pelo stack que ofereça a melhor produtividade para os desenvolvedores poderem trabalhar na manutenção do sistema.

4.1.6 Recursos Humanos

Trata-se dos recursos humanos que trabalharão no sistema. É um importante fator conhecer quais as tecnologias compreendidas e/ou dominadas pela equipe responsável pelo desenvolvimento do sistema. Isso envolve conhecimento sobre a plataforma na qual planeja-se desenvolver a aplicação, linguagens, frameworks e outros.

Conforme a solução tecnológica a qual se planeja optar, há de se analisar o reflexo desta na equipe. Se os integrantes a dominam, se não possuem conhecimento sobre ela e necessitarão de tempo para dominá-la, a compreendem parcialmente, mas o suficiente para aplicar a solução, ou se é necessário buscar novos profissionais e especialistas no mercado (GORBACHENKO, c2022). Neste último caso, vale analisar a disponibilidade de profissionais/especialistas relacionados à tecnologia pretendida, uma vez que, caso sejam escassos, isso pode significar dificuldades futuras no desenvolvimento e manutenção do sistema, valendo avaliar optar por outra tecnologia que possua uma disponibilidade maior de pessoas qualificadas a lidar com ela.

4.1.7 Seleção e Análise de Possíveis Stacks

Analisar os fatores anteriores possibilita selecionar opções de possíveis stacks a serem usados no sistema. Cada um desses stacks tem alta dependência do seu ecossistema, assim, é importante avaliar para cada um deles e seus componentes a dimensão da sua comunidade de usuários e colaboradores, suporte, controle de versões, documentação disponível e pareceres de usuários dessas soluções (HORIACHKO, c2022).

Outros pontos importantes a analisar são a segurança e privacidade das tecnologias componentes do stack, performance e se possuem custos de licenças. Como descrito no artigo *How to Choose the Right Technology Stack for Web Application Development* (SHIBU, 2021), vale optar por implementar soluções que sejam de código aberto, o que, além de auxiliar com um custo inicial de desenvolvimento menor, também permite seu uso com alterações ilimitadas.

Por fim, cabe analisar implementações atuais de sucesso, em especial nas grandes empresas. O resultado dessa análise pode ser um indicativo de qualidade de tais soluções tecnológicas (HORIACHKO, c2022).

4.1.8 Custo de Adoção do Stack

Enfim, analisar o custo de adoção do stack mais adequado e compará-lo com o orçamento disponível para o desenvolvimento do sistema. Caso inviável, deve-se fazer a análise para a opção seguinte mais adequada como solução.

5 EXEMPLOS DE STACKS PARA APLICAÇÕES WEB

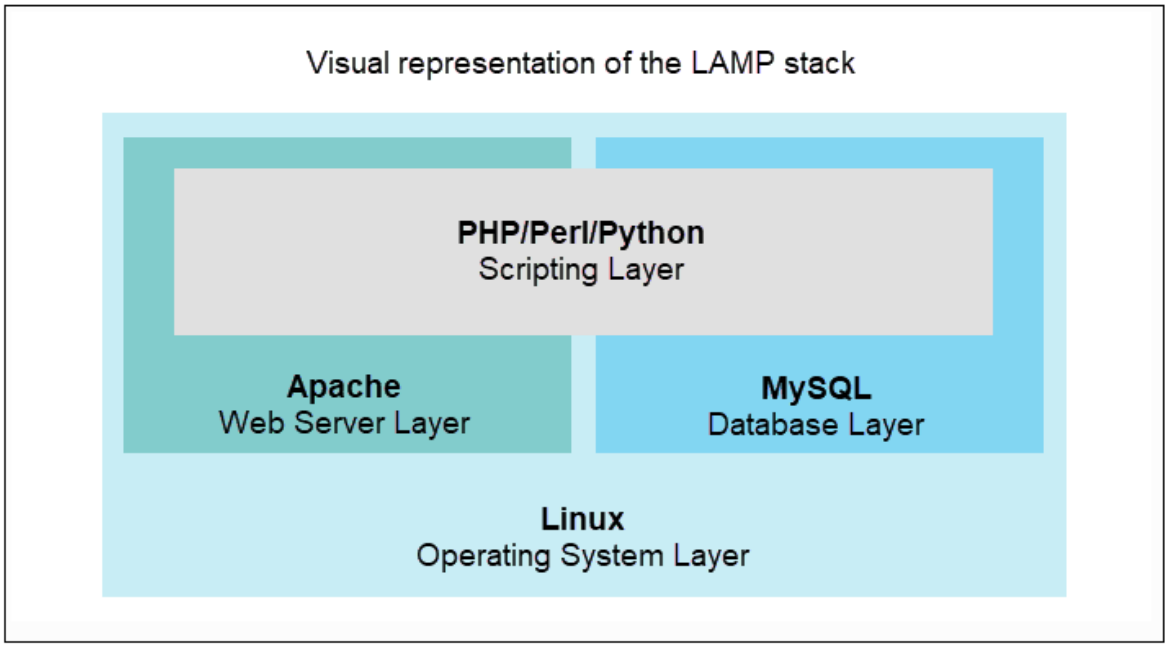
Stack é um termo que representa um conjunto de componentes, isto é, soluções tecnológicas, que são utilizadas conjuntamente para desenvolver e/ou fornecer suporte às aplicações. Neste capítulo são apresentados alguns exemplos de stacks já consolidados e amplamente usados na área de tecnologia.

5.1 LAMP

Sendo o primeiro stack popularmente conhecido, o LAMP é um stack de código aberto. Sua sigla é um acrônimo para Linux, Apache HTTP Server, MySQL e PHP/Perl/Python (SIMIC, 2022).

A imagem abaixo traz uma representação visual em camadas desse stack.

Figura 5.1 – Representação visual do LAMP em camadas



Fonte: SIMIC (2022)

5.1.1 Linux

Linux é um sistema operacional de código aberto, flexível e customizável. Por sua natureza e vantagens, esse é considerado a base do stack e uma ótima solução para executar os demais componentes (SIMIC).

5.1.2 Apache HTTP Server

O Apache é um software de servidor web que é executado no Linux quando no contexto do LAMP. Tem o papel de processar requisições e transmitir respostas através de redes usando o protocolo HTTP.

Esse software possui módulos de multiprocessamento, permitindo que seja configurado como um servidor baseado em processos, com sua arquitetura modular suportando funcionalidades como manipulação de protocolo modular, por exemplo.

Esse software também provê processos aplicados aos dados que são gerenciados pelo servidor, os quais são conhecidos como filtros. Com isso, tanto conteúdos estáticos quanto dinâmicos podem ser criptografados, escaneados e comprimidos usando tais funcionalidades (SIMIC, 2022).

5.1.3 MySQL

O MySQL é um sistema de gerenciamento de banco de dados que suporta SQL e tabelas relacionais. Com ele, é possível criar e manter bancos de dados dinâmicos de nível empresarial.

Por ser multiplataforma, pode ser incluso em demais stacks que utilizem de outros sistemas operacionais que não o Linux. Uma das principais desvantagens do MySQL, especialmente em comparação com soluções não relacionais como o MongoDB, é o quesito de escalar apenas verticalmente. Dado que também é ineficiente no manuseio de grandes bancos de dados, usar o MySQL em projetos que esperam muito tráfego requer um planejamento cuidadoso (SIMIC, 2022).

5.1.4 PHP

O PHP é uma linguagem de programação que combina todos os elementos do LAMP e permite que sites e aplicativos web sejam executados com eficiência. Quando um visitante abre a página da web, o servidor processa os comandos PHP e envia os resultados para o navegador do visitante.

O PHP é a quarta camada do stack original porque interage bem com o MySQL. É comumente usado para desenvolvimento web porque é uma linguagem de tipagem dinâmica e pode ser incorporada em HTML, tornando-o rápido e fácil de trabalhar.

A sigla “P” no acrônimo LAMP também pode se referir a duas outras linguagens de programação – Perl ou Python. Todas as três são ferramentas dinâmicas úteis para criar ambientes nos quais os aplicativos podem ser desenvolvidos com êxito (SIMIC, 2022).

5.1.5 Vantagens e Desvantagens

5.1.5.1 Vantagens

- Trata-se de uma solução de código aberto. Os códigos-fonte das soluções componentes desse stack são compartilhados e disponibilizados para que as pessoas possam fazer alterações e melhorias, aprimorando seu desempenho geral;
- Fácil personalização. Os usuários podem substituir cada componente por outra solução de código aberto para atender às necessidades de uma aplicação específica;
- É fácil encontrar suporte devido ao tamanho da comunidade;
- É um stack maduro e fácil de configurar.

5.1.5.2 Desvantagens

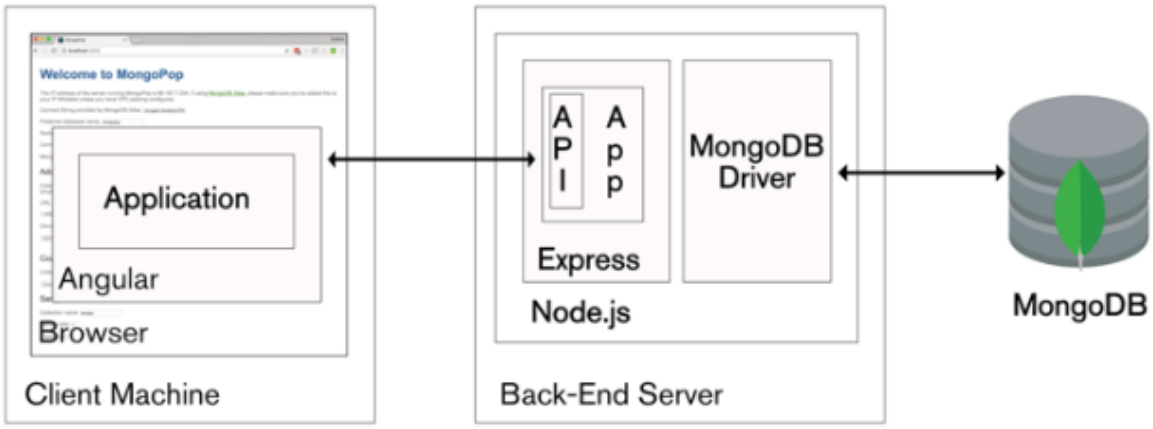
- Não suporta sistemas operacionais diferentes do Linux;
- A propriedade relacional do MySQL pode tornar o stack menos eficiente e flexível do que seus concorrentes que usam soluções não relacionais.
- O Apache pode ter problemas de desempenho sob grande carga de trabalho;
- Alternar entre codificação em Python e PHP no lado do servidor e usar JavaScript no lado do cliente pode interromper o fluxo de trabalho de desenvolvimento.

5.2 MEAN

O MEAN, bem como o LAMP, é um stack de código aberto que fornece uma estrutura de ponta a ponta para a construção de aplicações web dinâmicas, iniciando no topo (código executando no navegador) para baixo (banco de dados). Sua sigla é um acrônimo para Angular, Express, Node.js e MongoDB.

A imagem abaixo traz uma representação do MEAN em camadas.

Figura 5.2 – Representação visual do MEAN em camadas



Fonte: MORGAN (2017)

5.2.1 MongoDB

O MongoDB é um banco de dados orientado a documentos, de código aberto e não relacional. Diferente de bancos de dados relacionais, que armazenam dados em uma estrutura linhas-colunas, o MongoDB armazena documentos JSON em coleções e esquemas dinâmicos (MORGAN, 2017).

5.2.2 Express

Express é um framework para aplicações web que executa códigos back-end usando JavaScript. É executado como um módulo dentro do ambiente do Node.js. É possível executar todo o conjunto de regras de negócio da aplicação dentro do Express, além de permitir gerar HMTL para ser renderizado no navegador. Também pode ser utilizado para prover APIs REST, permitindo que a camada de front-end acesse recursos que necessita, como o banco de dados, por exemplo (MORGAN, 2017).

5.2.3 Angular

O Angular lida com o front-end da aplicação e permite executar códigos JavaScript no navegador do usuário, implementando assim uma interface reativa. Uma interface reativa fornece ao usuário um feedback imediato à medida que ele fornece a entrada de dados, em contraste aos formulários web estáticos, onde é necessário fornecer todos os dados primeiro, enviá-los e aguardar o retorno da aplicação.

Após o lançamento do Angular 2, ele foi completamente reescrito em Typescript, um superconjunto do JavaScript. Atualmente, Typescript também é recomendado para uso em aplicações que usam Angular (MORGAN, 2017).

5.2.4 Node.js

O Node.js é um JRE que executa o back-end de aplicações. Ele é baseado no mecanismo V8 do Google que é usado nos navegadores Chrome, além de oferecer funcionalidades essenciais para a implementação de aplicações web, incluindo protocolos de rede, tal como o HTTP. Módulos de terceiros, como o driver do MongoDB, podem ser instalados no ambiente usando a ferramenta NPM.

O Node.js é um sistema assíncrono e orientado a eventos, onde a aplicação faz solicitações e pode continuar em outras tarefas importantes para o processo, ao invés de interromper enquanto aguarda a resposta. Quando a solicitação tem sua resposta, a aplicação é avisada dos resultados via call-back, o que permite um grande número de operações serem executadas em paralelo, o que pode ser essencial para muitas aplicações.

O MongoDB, banco de dados usado nesse stack, também foi projetado para ser assíncrono, assim pode-se ter uma boa vantagem da utilização de ambos quando usado esse conjunto (MORGAN, 2017).

5.2.5 Variações

Posteriormente surgiram dois outros stacks que substituíam o Angular pelo React e Vue, sendo denominados MERN e MEVN stacks.

6 PROTÓTIPO DE AVALIAÇÃO

Para a apresentação e aplicação da metodologia proposta, é feito o uso de um cenário fictício, no qual uma grande empresa de suplementos decide por expandir seu negócio através da internet, disponibilizando seus produtos através de um novo website a ser desenvolvido.

6.1 Cenário

Uma grande empresa brasileira de suplementos decide por expandir seu negócio através da internet. Para esse propósito, o sistema a ser desenvolvido é um website que disponibiliza para compra todos os produtos da empresa. Para que um cliente possa comprar algum desses produtos, é necessário fazer um cadastro e criar uma conta no sistema, a qual contém seus dados pessoais, tais como documentos, endereços, formas de contato e pagamento, bem como disponibiliza ao cliente todos os registros de suas compras, como os produtos que comprou, acompanhamento de entregas e formas de pagamento utilizadas.

A empresa possui um catálogo de produtos pequeno, e cada produto possui uma página dedicada, para atender às especificidades de cada um deles. Conforme novos produtos forem adicionados ao catálogo, novas páginas devem ser criadas.

Além do supracitado, o website conta com a área de acesso à conta do cliente, atendimento via chat, sistema de pagamento online e acompanhamento logístico de entregas atualizado frequentemente, sendo estas três últimas funcionalidades disponibilizadas por microsserviços de empresas terceiras.

Além do uso de microsserviços de terceiros, o sistema faz uso do banco de dados corporativo.

O prazo para colocação do sistema no mercado para que a empresa obtenha vantagem competitiva em relação às demais em seu segmento é de dois anos.

A empresa possui uma área de tecnologia pequena, não especializada em desenvolvimento, mas voltada para gerenciamento e manutenção de softwares comumente usados em escritórios.

6.2 Emprego da Metodologia

A seguir é aplicada a metodologia apresentada neste trabalho.

6.2.1 Tipo de Projeto, Dimensão e Complexidade do Sistema

O projeto apresentado trata-se de um sistema de médio porte, um website para vendas de produtos online. Espera receber um grande número de clientes, ação que tem por consequência alta demanda nos serviços usados no sistema e simultaneidade dessas operações. Além de acesso à base de dados corporativa, faz uso de microsserviços de terceiros.

As funcionalidades descritas fazem uso intensivo da rede, acrescido das demandas de imediatismo de resposta, alta performance e interface responsiva, para que o usuário tenha uma boa percepção do sistema.

Dado que a base de dados armazena dados sensíveis dos clientes, é imprescindível implementar uma segurança robusta no sistema, bem como fazer uso das melhores práticas do mercado e adotar soluções tecnológicas atualizadas e seguras.

Com isso, torna-se inviável a utilização de tecnologias como CMS ou WordPress, pois, por se tratar de um sistema de médio porte e mais complexo, envolvendo múltiplos sistemas e integrações entre plataformas, é necessário utilizar um stack mais complexo.

6.2.2 Especificações do Sistema e Funcionalidades

O sistema é um website para vendas de produtos. Precisa ter sempre atualizado o número de produtos disponíveis em estoque. Permite que o cliente acesse sua conta, onde constam seus dados, histórico de compras e consulta de pedidos, onde pode checar o *statu*s da entrega de seus pedidos. Ao final do processo de compra, a plataforma apresenta formas de pagamento para que o cliente possa selecionar. Também faz parte de sua composição um chatbot de autoatendimento, que atende às necessidades e dúvidas em primeiro nível e escala a questão para um atendente humano quando sua base de conhecimento não supri a necessidade do atendido.

Como estudo de caso de sistema semelhante, cabe citar o website da empresa Growth Supplements (https://www.gsuplementos.com.br).

6.2.3 Tempo para Colocação no Mercado

O tempo de colocação no mercado, isto é, período desde a ideação até a sua disponibilidade online para os clientes, é de dois anos.

Para atendimento do prazo, cabe utilizar práticas de desenvolvimento ágil, como Extreme Programming – XP, Srum, Kanban ou outro que melhor a equipe se adapte, e tecnologias e/ou frameworks que permitam uma fácil integração.

6.2.4 Escalabilidade e Extensibilidade

Cada cliente que optar por comprar produtos da empresa através do sistema, necessita fazer seu cadastro e criar uma conta nele. Essa conta contém tanto seus dados pessoais quanto seu histórico de compras e acompanhamento de pedidos, os quais são armazenados na base de dados corporativa.

Cada produto possui uma página dedicada de acordo com as especificidades do produto. Quando um novo produto é adicionado, por consequência há a necessidade de criar uma nova página específica.

Além, o sistema conta com atendimento via chat, pagamento online e acompanhamento logístico de entregas, funcionalidades fornecidas por microsserviços de terceiros.

Citados os pressupostos, observa-se que há a possibilidade de crescimento exponencial de cadastros de clientes na base de dados utilizada pelo sistema e simultaneidade de chamadas de serviço, acessos e atendimento, o que pode causar prejuízo à experiência do usuário através do aumento de tempo de resposta, ou mesmo uma inoperação do sistema, caso as tecnologias escolhidas sejam defasadas ou limitadas.

Essas premissas reforçam a inviabilidade do uso de CMS ou WordPress para o desenvolvimento. Por outro lado, ferramentas como Angular, React, Vue, Node.js – MEAN, MERN, MEVN Stacks – e Ruby on Rails possuem excelente escalabilidade.

6.2.5 Manutenção

A manutenção trata do processo geral de mudança e adaptação em um sistema depois que ele é liberado para uso. Além das naturais alterações que podem ocorrer no sistema, como erros de codificação, correção de erros de projeto, correção de especificações ou acomodação de novos requisitos, há também a dinâmica do negócio que o sistema suporta e a especificação de criação de páginas específicas para cada produto.

Neste caso, é viável optar por um stack que ofereça a melhor produtividade para a área técnica poder trabalhar na manutenção do sistema.

6.2.6 Recursos Humanos

Dados os tópicos apresentados até então, cabe a análise dos recursos humanos que trabalharão no sistema.

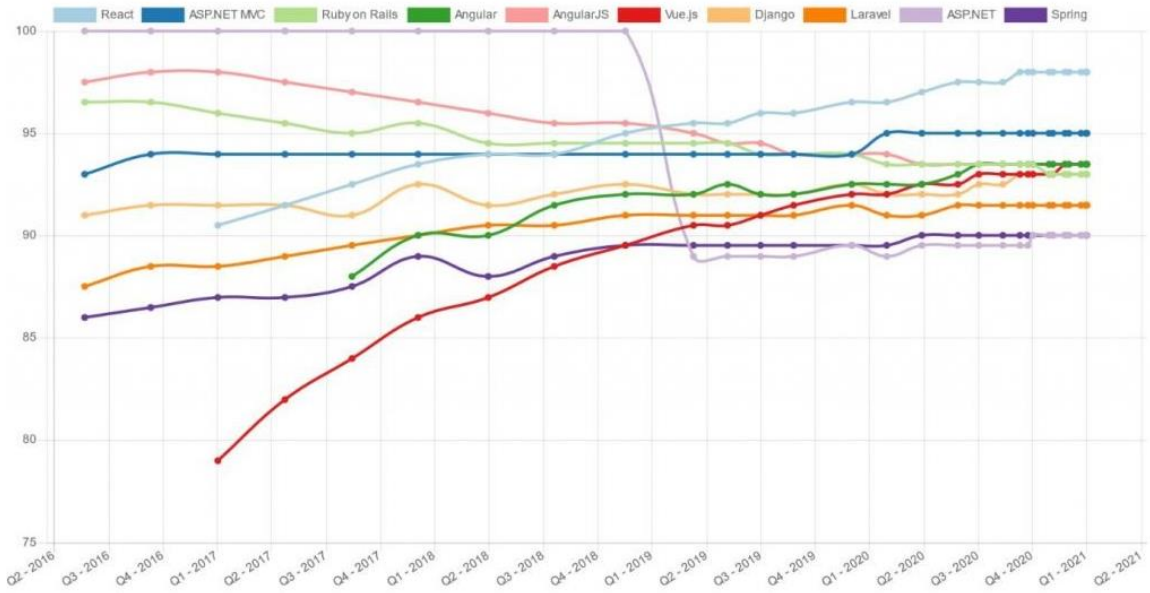
Como apresentado anteriormente, a empresa possui uma área de tecnologia pequena, não especializada em desenvolvimento, mas voltada para gerenciamento e manutenção de softwares comumente usados em escritórios. Neste caso, não há a necessidade de fazer um levantamento das tecnologias – linguagens, frameworks e outros – compreendidas e/ou dominadas pela equipe. Assim, o esforço empreendido deve ser direcionado para um levantamento das tecnologias em alta no mercado e a disponibilidade de profissionais/especialistas relacionados a elas, levando-se em conta as premissas do projeto, suas análises e resultados obtidos, como feito até então.

Segundo o website hotframeworks, que meça a popularidade de frameworks e baseia suas notas em dois parâmetros, os quais:

1. Pontuação do GitHub: número de estrelas que um repositório de um framework possui no GitHub;
2. Pontuação do Stack Overflow: número de perguntas no Stack Overflow marcadas com o nome do framework.

São apresentados os seguintes comparativos:

Figura 6.1 – Ranking de popularidade de frameworks, de Q2 - 2016 a Q2 – 2021



Fonte: hotframeworks (2023)

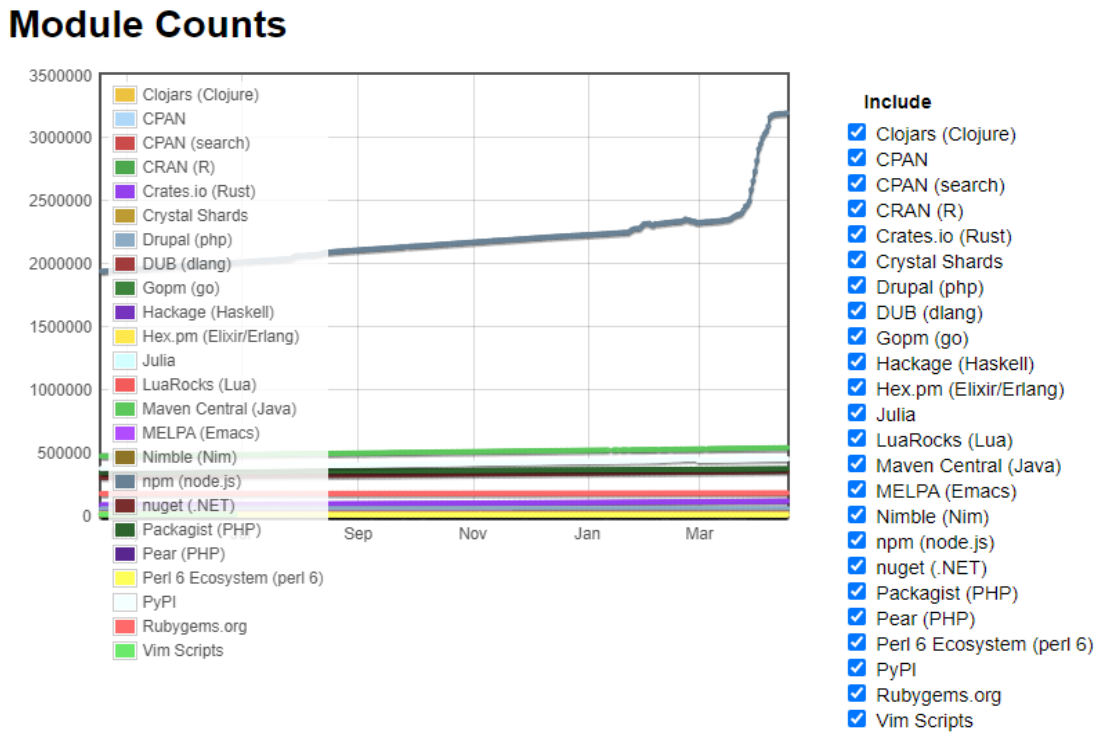
Tabela 6.1 – Os vinte primeiros frameworks mais populares

Framework	Github Score	Stack Overflow Score	Overall Score
React	99	97	98
ASP.NET MVC		95	95
Angular	91	96	93
Ruby on Rails	87	99	93
AngularJS	90	97	93
Vue.js	100	87	93
Django	89	97	93
Laravel	90	93	91
ASP.NET	80	100	90
Spring	86	94	90
Express	88	87	87
Flask	89	83	86
Meteor	86	80	83
Symfony	81	86	83
CodeIgniter	79	86	82
JSF		81	81
Ember.js	80	78	79
.NET Core	77	80	78
Google Web Toolkit		77	77
CakePHP	72	80	76

Fonte: hotframeworks (2023)

E o website Module Counts, que apresenta a popularidade de módulos baseado em um levantamento diário via data scraping usando cron job em websites relevantes:

Figura 6.2 – Ranking de popularidade de módulos



Fonte: Module Counts (2023)

Tabela 6.2 – Crescimento de popularidade de módulos no último ano

time period ○ all time ◉ last year ○ last 90 days ○ last 30 days ○ last 7 days

	Apr 10	Apr 11	Apr 12	Apr 13	Apr 14	Apr 15	Apr 16	Avg Growth
Clojars (Clojure)	29900	29902	29905	29909	29910	29911	29911	2/day
CPAN	44137	44138	44139	44141	44141	44141	44142	1/day
CPAN (search)								5/day
CRAN (R)	19323	19333	19348	19361	19354	19359	19358	6/day
Crates.io (Rust)	110770	110883	110966	111058	111157	111243	111363	99/day
Crystal Shards								2/day
Drupal (php)	49913	49918	49922	49933	49943	49947	49949	6/day
DUB (dlang)	2319	2319	2323	2325	2325	2325	2326	1/day
Gopm (go)	22377	22377	22377	22377	22377	22377	22377	0/day
Hackage (Haskell)	17078	17080	17084	17085	17086	17086	17088	2/day
Hex.pm (Elixir/Erlang)	14261	14266	14270	14275	14280	14290	14293	5/day
Julia								0/day
LuaRocks (Lua)	3415	3415	3419	3421	3422	3424	3424	1/day
Maven Central (Java)	535219	535390	535590	535789	536048	536286	536362	190/day
MELPA (Emacs)								2/day
Nimble (Nim)	2121	2121	2122	2122	2122	2122	2123	0/day
npm (node.js)	3188233	3189483	3191398	3192787	3192334	3194314	3195882	1275/day
nuget (.NET)	349423	349680	349938	350128	350292	350546	350660	206/day
Packagist (PHP)	368281	368376	368487	368601	368702	368781	368827	91/day
Pear (PHP)	603	603	603	603	603	603	603	0/day
Perl 6 Ecosystem (perl 6)	2162	2162	2162	2162	2161	2161	2163	0/day
PyPI	445596	445872	446131	446387	446686	446963	447160	261/day
Rubygems.org	176054	176070	176090	176103	176119	176135	176145	15/day
Vim Scripts								0/day

Fonte: Module Counts (2023)

Percebe-se o crescimento e popularidade de tecnologias como React, Angular, Ruby on Rails, Vue e o grande destaque do módulo npm, usado no Node.js, comparado aos demais apresentados. O cruzamento de tais dados denota a popularidade dessas tecnologias atualmente entre analistas e desenvolvedores, servindo como um bom parâmetro sobre a disponibilidade de recursos humanos disponíveis para lidar com tais tecnologias. Cabe também ressaltar levantamentos em sistemas fidedignos relacionados às vagas e propostas de trabalho, que reforçam a escolha correta.

Tais tecnologias supracitadas foram exaltadas não só pelas suas posições no ranking de popularidade, mas também por estarem condizentes com os resultados das análises das etapas anteriores.

6.2.7 Seleção e Análise de Possíveis Stacks

Baseando-se no levantamento e resultados das etapas anteriores, são selecionadas as seguintes tecnologias como possíveis soluções tecnológicas para o desenvolvimento, implantação e manutenção do sistema:

- React;
- Angular;
- Vue;
- Node.js;
- Ruby on Rails.

Assim, cabe analisar cada uma das tecnologias selecionadas de perspectivas como: componentes que possuem, dimensão da comunidade, suporte, controle de versões, documentação disponível, parecer de usuários, segurança e privacidade, performance, se possui custos de licença, casos de sucessos e/ou outros que colaborem com métricas que auxiliem na composição da noção de valor e probabilidade de sucesso que a solução tecnológica pode ter.

6.2.7.1 Vantagens

6.2.7.1.1 React

Algumas vantagens do React são: flexível para definição de padrão de trabalho; permite uso de JavaScript, ECMAScript 6, TypeScript, JSX e outros; uso de Browser Tools; fácil migração entre versões; permite alta reutilização de código, facilidade na manutenção e evolução do código; bastante popular; permite mobile com o React Native.

Exemplos de empresas que fazem uso do React: Netflix, Spotify, Twitter, Facebook, Airbnb, Americanas, Uber, PayPal e Walmart.

6.2.7.1.2 Angular

Algumas vantagens do Angular são: grande robustez, isto é, quase tudo que é necessário a uma aplicação se encontra nele nativamente; possui mais tempo no mercado que os demais supracitados; usa padrão MVC – Model, View, Controller; HTML apartado do JavaScript, HttpCliente; documentação detalhada, utiliza TypeScript nativamente.

Exemplos de empresas que fazem uso do Angular: Toro, Azure, Forbes, BMW, Santander.

6.2.7.1.3 Vue

Algumas vantagens do Vue são: framework progressivo, leve e completo; Documentação detalhada, fácil aprendizagem; crescimento em popularidade, adaptável, permite o uso de ECMAScript 6, JavaScript, TypeScript e outros; Browser Tools.

Exemplos de empresas que fazem uso do Vue: Glovo, GitLab, Behance, Renner.

6.2.7.1.4 Node.js

Algumas vantagens do Node.js são: ecossistema npm; desenvolvimento rápido; utiliza JavaScript; rápido tempo de resposta; fácil de se encontrar profissionais que lidam com essa tecnologia; permite JavaScript full-stack; leve.

Exemplos de empresas que fazem uso do Node.js: LinkeIn, Netflix, Uber, Walmart, Paypal, IBM, Dow Jones.

6.2.7.1.5 Ruby on Rails

Algumas vantagens do Ruby on Rails são: fácil leitura, compreensão e utilização; rica coleção de ferramentas; opções de automação de testes; maduro e estável; escalabilidade.

Exemplos de empresas que fazem uso do Ruby on Rails: Shopify, Twitter, Instacart, Twitch, Zendesk.

Dados os resultados das análises e o levantamento das vantagens das tecnologias elencadas, opta-se pelas opções React e Node.js. Tal decisão é reforçada pela utilização de ambas as tecnologias em grandes empresas, como Netflix, Uber, PayPal e Walmart.

Desta forma, a opção de stack mais viável neste caso é um que inclua estas duas tecnologias. Um stack que inclui ambas e possui vantagens com a utilização dos demais componentes em conjunto é o MERN stack. Algumas das vantagens que a utilização desse stack traz são: paradigma JavaScript Everywhere, paradigma JSON Everywhere, dispensa da camada ORM para armazenamento de dados, performance do Node.js, ecossistema npm e sistema SPA.

6.2.8 Custo de Adoção do Stack

O MERN é um stack completo (full stack) de código aberto, o que viabiliza sua adoção em relação ao parâmetro custo.

7 RESULTADOS

A apresentação dos resultados neste capítulo visa, além de expor o corolário, melhor elucidar o apresentado na metodologia.

Foi utilizado para apresentação e aplicação da metodologia proposta um cenário fictício, no qual uma grande empresa de suplementos decidiu por expandir seu negócio através da internet, disponibilizando seus produtos através de um novo website.

Deu-se então o início da análise do cenário adotado. Na primeira etapa foi examinado o tipo do projeto, sua dimensão e complexidade. Com isto, foi observado que as funcionalidades descritas faziam uso intensivo da rede, acrescido das demandas de imediatismo de resposta, alta performance e interface responsiva. A base de dados armazenaria dados sensíveis dos clientes, sendo imprescindível a implementação de uma segurança robusta no sistema, bem como fazer uso das melhores práticas do mercado e adotar soluções tecnológicas atualizadas e seguras. Assim, tornou-se inviável a utilização de tecnologias como CMS ou WordPress, pois, por se tratar de um sistema de médio porte e mais complexo, envolvendo múltiplos sistemas e integrações entre plataformas, fazia-se necessário utilizar um stack mais complexo.

Em seguida, feita a análise das especificações e funcionalidades do sistema, elucidando necessidades como a atualização contínua do número de produtos no estoque, criação e acesso de contas dos clientes no sistema, onde constariam seus dados, histórico de compras e consulta de pedidos, seleção de formas de pagamento, chatbot de autoatendimento em primeiro nível e, por fim, apresentado um estudo de caso de sucesso.

Seguiu-se então com o diagnóstico do tempo para colocação no mercado, no caso, dois anos, inferindo a viabilidade na utilização de práticas de desenvolvimento ágil, como Extreme Programming – XP, Srum, Kanban ou outro que melhor a equipe se adaptasse, e tecnologias e/ou frameworks que permitissem uma fácil integração, para atendimento do prazo.

Por conseguinte, houve o estudo de sua escalabilidade e extensibilidade, onde observou-se que haveria a possibilidade de crescimento exponencial de cadastros de clientes na base de dados utilizada pelo sistema e simultaneidade de chamadas de serviço, acessos e atendimento, o que poderia causar prejuízo à experiência do usuário através de aumento de tempo de resposta, ou mesmo uma inoperação do sistema, caso as tecnologias escolhidas fosse defasadas ou limitadas. Tais premissas reforçaram a inviabilidade do uso de CMS ou WordPress para o desenvolvimento. Por outro lado, apresentou ferramentas como Angular,

React, Vue, Node.js – MEAN, MERN, MEVN Stacks – e Ruby on Rails como opções com excelente escalabilidade.

Em seguida houve a análise do cenário de manutenção, elucidando a viabilidade de optar por um stack que ofereceria a melhor produtividade para a área técnica poder trabalhar na manutenção do sistema.

Seguiu-se com o exame do fator recursos humanos. No cenário, como a empresa possuía uma área de tecnologia pequena, não especializada em desenvolvimento, mas voltada para gerenciamento e manutenção de softwares comumente usados em escritórios, não houve a necessidade de fazer um levantamento das tecnologias – linguagens, frameworks e outros – compreendidas e/ou dominadas pela equipe. Assim, o esforço empreendido foi direcionado para um levantamento das tecnologias em alta no mercado e a disponibilidade de profissionais/especialistas relacionados a elas. Assim, foram usados como base e cruzados dados dos websites hotframeworks, que meça a popularidade de frameworks, e Module Counts, que apresenta a popularidade de módulos, obtendo como resultado a informação sobre o crescimento e popularidade de tecnologias como React, Angular, Ruby on Rails, Vue, e o grande destaque do módulo npm, usado no Node.js. O cruzamento de tais dados denota a popularidade dessas tecnologias atualmente entre analistas e desenvolvedores, o que serviu como parâmetro sobre a disponibilidade de recursos humanos para lidar com tais tecnologias. Tais tecnologias supracitadas foram exaltadas não só pelas suas posições no ranking de popularidade, mas também por estarem condizentes com os resultados das análises das etapas anteriores.

Foi feita então a seleção e análise de possíveis stacks, baseando-se no levantamento e resultados das etapas anteriores, apresentando suas vantagens e empresas que deles fazem uso, optando-se então pelas tecnologias React e Node.js. Desta forma, a opção de stack mais viável neste caso foi um que incluísse estas duas tecnologias. Um stack que inclui ambas e possui vantagens com a utilização de demais componentes em conjunto é o MERN stack, sendo este selecionado como resultado desta etapa.

Por fim, foi feito o exame do custo de adoção desse stack, que, por ser completo e de código aberto, viabilizou sua adoção em relação ao parâmetro custo. Assim, como resultado final, foi escolhido o MERN stack como o mais adequado, viável e com maior probabilidade de êxito para desenvolvimento e manutenção da aplicação analisada.

8 CONSIDERAÇÕES FINAIS

O desenvolvimento do estudo presente neste trabalho permitiu uma melhor interpretação da importância do processo de seleção das tecnologias a serem usadas no desenvolvimento e manutenção de aplicações web.

Ao decorrer da metodologia foram apresentadas as fases junto as suas respectivas contribuições parciais para a formação integral do modelo, usando como base um determinado cenário, que foi usado com o objetivo de melhor elucidar o modelo e levar a uma fácil compreensão do trabalho.

Um óbice apresentou-se durante o decorrer deste trabalho, sendo ele relativo à dificuldade em encontrar bibliografias que pudessem apoiar na pesquisa.

A metodologia desenvolvida atendeu às expectativas apresentadas no início do trabalho, gerando resultado satisfatório e permitindo compreender fatores que podem maximizar a probabilidade de escolha de um conjunto de soluções tecnológicas adequado e que tenha êxito no desenvolvimento e manutenção de aplicações web, assim contribuindo com informações que podem ser usadas em estudos futuros, como análise de fatores que impactam no sucesso de projetos de aplicações web e conjuntos de soluções tecnológicas adequados para tipos específicos de aplicações.

REFERÊNCIAS

CANGUÇU, Raphael. **Porque utilizar o Ruby on Rails em seu projeto**. codificar. Disponível em: <https://codificar.com.br/porque-utilizar-o-ruby-on-rails-em-seu-projeto/>. Acesso em: 23 de abril de 2023.

CLARK, Mariana. **Dez principais startups usando Ruby on Rails**. Back4app. Disponível em: <https://blog.back4app.com/pt/dez-principais-startups-usando-ruby-on-rails/>. Acesso em: 23 de abril de 2023.

FREIRE, Henrique. **React, Vue, Angular, conheça suas vantagens e desvantagens e qual é melhor para seus projetos**. Medium. Disponível em: <https://henrique-freire.medium.com/react-vue-angular-conhe%C3%A7a-suas-vantagens-e-desvantagens-e-qual-%C3%A9-melhor-para-seus-projetos-53734bb3d37f>. Acesso em: 23 de abril de 2023.

From History of Web Application Development. Devsaran, 2016. Disponível em: <https://www.devsaran.com/blog/history-web-application-development>. Acesso em: 10 de maio de 2022.

GORBACHENKO, Pavel. **A Comprehensive Guide to Modern Web Development Stacks**. Enkonix, c2022. Disponível em: <https://enkonix.com/blog/web-development-stacks/>. Acesso em: 12 de outubro de 2022.

HORIACHKO, Andrii. **How to Choose the Best Technology Stack for Web Application Development: 10 Tips to Know**. Softermii, c2022. Disponível em: <https://www.softermii.com/blog/10-tips-in-choosing-the-best-tech-stack-for-your-web-application>. Acesso em: 12 de outubro de 2022.

hotframeworks, **Find your new favorite web framework**. Disponível em: <https://hotframeworks.com/>. Acesso em: 16 de abril de 2023.

KOPEC, David. **Dart for Absolute Beginners**. 1. Ed. New York, United States: Editora Apress, 2014.

KÜNZEL, Paulo. **Vantagens e Desvantagens de usar o Node.JS**. MUNDOJS. Disponível em: <https://www.mundojs.com.br/2018/10/16/vantagens-e-desvantagens-de-usar-o-node-js/>. Acesso em: 23 de abril de 2023.

Medium, **A brief history of web app**. Disponível em: <https://oleg-uryutin.medium.com/a-brief-history-of-web-app-50d188f30d>. Acesso em: 10 de maio de 2022.

Module Counts. Disponível em: <https://hotframeworks.com/>. Acesso em: 16 de abril de 2023.

MORGAN, Andrew. Introducing the MEAN and MERN stacks. **MongoDB**, 2017. Disponível em: <https://www.mongodb.com/blog/post/the-modern-application-stack-part-1-introducing-the-mean-stack>. Acesso em: 06 de novembro de 2022.

NORTHWOOD, Chris. **The Full Stack Developer – Your Essential Guide to the Everiday Skills Expected of a Modern Full Stack Web Developer**. 1. Ed. Manchester, United Kingdom: Editora Apress, 2018.

O que é o Nodejs e quais suas vantagens e desvantagens?. valuehost. Disponível em: <https://www.valuehost.com.br/blog/o-que-e-o-nodejs/>. Acesso em: 16 de abril de 2023.

POCZWARDOWSKI, Michał. **What Are the Risks of Choosing Wrong Technology for Your Web App?**. netguru. Disponível em: <https://www.netguru.com/blog/choosing-wrong-technology-risks>. Acesso em: 30 de outubro de 2022.

PRESSMAN, Roger S.. **Engenharia de Software – Uma Abordagem Profissional**. 7. Ed. Porto Alegre, Brasil: Editora AMGH, 2011.

SHIBU, Abhirami Janak. **How to Choose the Right Technology Stack for Web Application Development**. NeoITO, 2021. Disponível em: <https://www.neoito.com/blog/how-to-choose-technology-stack/>. Acesso em: 12 de outubro de 2022.

SIMIC, Sofija. **What is LAMP Stack?**. phoenixNAP, 2022. Disponível em: <https://phoenixnap.com/kb/what-is-a-lamp-stack>. Acesso em: 06 de novembro de 2022.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. Ed. São Paulo, Brasil: Editora Pearson, 2011.

SUBRAMANIAN, Vasan. **Pro MERN Stack – Full Stack Web App Development with Mongo, Express, React, and Node**. 2. Ed. Bangalore, Karnataka, India: Editora Apress, 2019.

VALENTE, Marco Tulio. **Engenharia de Software Moderna – Princípios e práticas para desenvolvimento de software com produtividade**. 1. Ed. Belo Horizonte, Brasil: Editora Independente, 2020.

Wikipedia, **Progressive web application**. Disponível em: <https://en.m.wikipedia.org/wiki/Progressive_web_application>. Acesso em: 15 de maio de 2022.

Wikipedia, **Web application**. Disponível em: <https://en.m.wikipedia.org/wiki/Web_application>. Acesso em: 10 de maio de 2022