APLICAÇÃO DE GOVERNANÇA DE DADOS EM SOLUÇÃO DE *SOFTWARE* PARA PESQUISA DE PLACAS EM UMA EMPRESA DO SETOR DE INDÚSTRIA E COMÉRCIO DE PEÇAS AUTOMOTIVAS

Cleber Ferreira Shimizu
Leandro dos Santos Canestraro
Lucelia Mildemberger
Mateus Eduardo Braz

RESUMO

Este artigo descreve o trabalho de pesquisa de campo realizado pelos estudantes do oitavo período do curso de Bacharelado Engenharia de Software do Centro Universitário Senai de São José dos Pinhais em uma empresa de indústria e comércio de peças automotivas no mesmo município. O objetivo geral é propor uma solução para pesquisa de informações de veículos através de suas placas, a fim de garantir que a peça desejada e fornecida pela empresa estudada corresponda ao modelo do veículo do cliente. Os objetivos específicos são identificar os requisitos do cliente, formular uma proposta de solução e testar a viabilidade e adequação da solução proposta com dados fictícios. Como metodologia foram aplicadas a pesquisa bibliográfica e entrevista informal com integrantes da empresa para levantamento de requisitos, além da modelagem de software. Um dos principais conceitos adotados foi a Governança de Tecnologia da Informação (GTI). A governança de tecnologia da informação é um conjunto de medidas que assegura uma gestão adequada dos dados colhidos por uma empresa, garantindo o bom uso e a preservação dessas informações. Os resultados obtidos foram uma aplicação que faz uso intensivo de várias fontes de dados na qual a utilização da Governança se tornou imprescindível para a solução proposta, que se configura como um Produto Minimamente Viável (MVP), ou seja, funcional, mas ainda não finalizado para fins comerciais.

Palavras-chave: Governança de tecnologia da informação. Software Web. APIs.

SUMMARY

This article describes the field research work carried out by students in the eighth period of the Software Engineering Bachelor's degree course at Centro Universitário Senai de São José dos Pinhais in an automotive parts industry and trade company in the same municipality. The general objective is to propose a solution for searching vehicle information through their license plates, in order to ensure that the part desired and supplied by the studied company corresponds to the customer's vehicle model. The specific objectives are to identify customer requirements, formulate a proposed solution and test the feasibility and suitability of the proposed solution with fictitious data. As a methodology, bibliographical research and informal interviews with company members were applied to gather requirements, in addition to software modeling. One of the main concepts adopted was Information Technology Governance (GTI). Information technology governance is a set of measures that ensure adequate management of data collected by a company, guaranteeing the good use and preservation of this information. The results obtained were an application that makes intensive use of several data sources in which the use of Governance became essential for the proposed solution, which is configured as a Minimally Viable Product (MVP), that is, functional, but not yet finalized. for commercial purposes.

Keywords: Information technology governance. Web Software. APIs.

1.INTRODUÇÃO

Por meio da adoção de boas práticas de governança corporativa, as empresas podem melhorar sua gestão, reduzir riscos e aumentar a confiança dos investidores, além de valorizar a empresa no mercado. A implementação de mecanismos de governança corporativa eficazes pode ser um fator chave na diferenciação das empresas, permitindo que se destaquem em um ambiente de negócios cada vez mais competitivo (POLIZEL, 2017).

Inserida no contexto da governança corporativa, a governança de TI traz melhorias na gestão dos recursos de TI, permitindo que a organização otimize seus investimentos em tecnologia. Desta forma, é possível reduzir riscos, garantindo que os dados e sistemas de TI sejam protegidos adequadamente (HARMER, 2014).

É importante destacar que a governança de TI não se trata apenas de garantir que a tecnologia seja usada de maneira adequada e segura, mas também de garantir que a TI esteja alinhada com as metas e objetivos do negócio. A adoção de práticas de governança de TI permite que a TI seja gerenciada de forma mais estratégica, levando a um melhor uso dos recursos e a uma maior eficiência e eficácia dos processos de TI (VAN GREMBERGEN; DE HAES, 2020).

Na governança de TI, a governança de dados envolve as ações, os processos e tecnologias que oferecem suporte durante todo o ciclo de vida dos dados a fim de mantê-los seguros, privados, precisos, disponíveis e utilizáveis.

Este trabalho descreve uma proposta de *software* para solucionar um problema real em uma empresa, considerando aspectos de governança de dados e de TI atuais da organização estudada.

1.1 CONTEXTO DA EMPRESA

As informações do contexto da empresa foram obtidas por meio de entrevista informal com os responsáveis pelo departamento de TI (Tecnologia da Informação) em 06 e 28 de março de 2023. A empresa abordada neste artigo está atualmente

sediada em São José dos Pinhais e atua no mercado de componentes elétricos e mecânicos para motocicletas. Em 1960, em Joinville, Santa Catarina, surgiu com o foco de produzir magnetos e bobinas de ignição. Em 1995, transferiu-se para o município de São José dos Pinhais, Paraná. Presente em todo o Brasil, abastece um total de 39 marcas, com experiência de 60 anos de atuação no mercado. Suas atividades são baseadas nos pilares: cultivar a confiança dos clientes por meio de um excelente atendimento, pontualidade na entrega e qualidade dos produtos. tudo isso sempre com respeito, seriedade e comprometimento (EMPRESA ESTUDADA, 2023).

Dentro da organização existem duas *startups* que visam o melhoramento dos processos internos, aplicando e desenvolvendo tecnologia. Atualmente, devido à baixa oferta de mão de obra especializada no mercado de trabalho, a empresa preza pelo desenvolvimento profissional e acadêmico de seus colaboradores (EMPRESA ESTUDADA, 2023).

O fomento de tecnologia interna vai de encontro os objetivos atuais da governança corporativa e de TI - em particular, com os esforços em melhorar e expandir a comunicação dentro do negócio. É essencial que os meios de comunicação de uma empresa atendam as demandas e expectativas dos clientes, evidenciando transparência, confiabilidade e responsabilidade nas relações comerciais (EMPRESA ESTUDADA, 2023).

O desafio da empresa é garantir que as peças, encomendadas pelos clientes por meio do *site*, correspondam exatamente ao modelo de veículo para evitar os custos de devolução por compras erradas.

1.2 OBJETIVOS

Foram definidos um objetivo geral e três objetivos específicos para serem atingidos com o desenvolvimento deste trabalho, quais sejam:

1.2.1 Objetivo Geral

Propor uma solução para pesquisa de informações de veículos através de suas placas, a fim de garantir que a peça desejada e fornecida pela empresa corresponda ao modelo do veículo do cliente.

1.2.2 Objetivo Específicos

a) Identificar os requisitos do cliente;
b) Formular uma proposta de solução;
c) Testar a viabilidade e adequação da solução proposta com dados fictícios.

1.3 JUSTIFICATIVA

Uma empresa moderna baseia seus negócios nos dados que coleta e manipula, de modo a tomar decisões estratégicas que garantam o seu crescimento. A governança de dados é composta pelos padrões internos que fazem a gestão da forma como os dados são coletados, armazenados, processados e descartados, o controle de acesso e a definição de quais dados são protegidos pela governança (RÊGO, 2013).

No contexto deste trabalho, a empresa estudada lida com dados de diversas fontes e diversos níveis de sensibilidade, desde dados pessoais de clientes finais até dados relacionados ao mercado e à sua própria situação financeira. Dessa forma, garantir o bom uso desses dados, sua segurança, privacidade, integridade e disponibilidade torna-se uma tarefa importante e estratégica (EMPRESA ESTUDADA, 2023).

1.4 METODOLOGIA

Para se realizar a proposta de *software* será necessário levantar os requisitos do cliente através de entrevistas não estruturadas com os *stakeholders*. A entrevista aos *stakeholders* no desenvolvimento de *software* é um processo em que membros relevantes do projeto são entrevistados para se obter informações sobre suas necessidades, expectativas e opiniões em relação ao *software* que está sendo desenvolvido (MALL, 2018).

De acordo com Wilson (2013), a entrevista não estruturada é uma abordagem flexível e aberta, comumente utilizada em etapas de levantamento de requisitos em projetos de desenvolvimento de software, para coletar informações dos stakeholders. Nessa técnica, o entrevistador não segue um roteiro pré-determinado de perguntas, permitindo uma interação espontânea com os participantes. Esta flexibilidade favorece a exploração de tópicos emergentes, assim revelando informações valiosas e insights inesperados.

A pesquisa bibliográfica foi utilizada para fundamentar os conceitos necessários para o desenvolvimento deste trabalho. Este tipo de pesquisa consiste em consultar e analisar uma ampla gama de fontes bibliográficas, como livros, artigos científicos, dissertações, teses, relatórios técnicos, entre outros, sobre um determinado tema de estudo (GIL, 2017).

Ainda, de acordo com Severino (2014), a pesquisa bibliográfica desempenha um papel fundamental no processo de produção do conhecimento científico. É essencial selecionar cuidadosamente as fontes a serem consultadas, considerando sua relevância e confiabilidade. Para tal, é importante avaliar a data de publicação, reputação das editoras ou revistas, e manter o foco em fontes que abordem o tema estudado de maneira abrangente e atualizada.

Outras ferramentas foram utilizadas para o desenvolvimento da solução tais como a modelagem da solução, considerando-se a arquitetura mais simples possível para se obter um *software* funcional (McELROY, 2016).

A modelagem de *software* é o processo de criar representações esquemáticas de sistemas de *software* usando notações, símbolos e diagramas. Desta forma, é possível expor visualmente as decisões de *design*, requisitos e especificações de sistemas de *software*, facilitando a análise e comunicação (WAZLAWICK, 2019).

Em seguida, desenvolve-se este *software* como um MVP (*Minimum Viable Product*) visando testar a usabilidade e a adequação às expectativas do cliente (McELROY, 2016).

O MVP é uma versão simplificada de um produto que contempla apenas as funcionalidades essenciais, visando atender às necessidades iniciais dos usuários. Essa abordagem permite que desenvolvedores obtenham feedback valioso dos usuários o mais cedo possível no processo de criação, possibilitando ajustes e melhorias contínuas (RIES, 2017).

Enfim, a proposta será avaliada pelos clientes, inicialmente utilizando dados fictícios. Mediante a aprovação dos níveis de verossimilhança e estrutura do protótipo apresentado, a solução pode ser implementada em seus sistemas vigentes, utilizando dados e estruturas reais.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais conceitos abordados durante o desenvolvimento deste trabalho, que embasam o processo de criação da proposta para o problema levantado.

2.1 GOVERNANÇA CORPORATIVA

Segundo Blok (2020), a governança corporativa é um conjunto de práticas e políticas que visa aumentar a transparência, a prestação de contas e a responsabilidade das organizações perante seus *stakeholders*. A adoção de boas práticas de governança corporativa pode trazer diversos benefícios, como a melhoria

da gestão da empresa, a redução de riscos, o aumento da confiança dos investidores e a valorização da empresa no mercado.

A transparência, prestação de contas e responsabilidade corporativa são elementos fundamentais da governança corporativa, que ajudam a garantir a sustentabilidade dos negócios e o respeito aos interesses de todos os *stakeholders* envolvidos. A governança corporativa tem como objetivo aumentar a confiança e a credibilidade das empresas, estabelecendo mecanismos de controle e monitoramento para garantir a tomada de decisões estratégicas mais éticas e responsáveis (BLOK, 2020).

2.2 GOVERNANÇA DE TI (GTI)

De acordo com Weill e Ross (2006), a governança de TI é o sistema de decisões e responsabilidades que define as direções estratégicas, as metas e os objetivos da TI de uma organização, além de assegurar que essas metas sejam atingidas e que os riscos associados à TI sejam gerenciados de forma adequada. Desta forma, por meio de práticas efetivas de governança de TI, é possível trazer diversos benefícios para uma organização.

A boa governança de TI busca estabelecer ou aumentar a transparência, permitindo que as partes interessadas na organização compreendam melhor como a TI é usada e como os recursos são gerenciados. Desta forma, a governança de TI maximiza o valor entregue pela TI à organização, garantindo que a TI esteja alinhada com os objetivos estratégicos e ajudando a impulsionar a inovação (HARMER, 2014).

2.3 PRINCÍPIOS DA GOVERNANÇA DE TI

Na visão de Senft e Gallegos (2008), os princípios da governança de TI são baseados em transparência, responsabilidade, equidade, conformidade e orientação para resultados. Esses princípios devem ser implementados para garantir a efetividade e eficiência da governança de TI em uma organização.

Além disso, como destacado por Rêgo (2013), a governança de TI deve ser fundamentada em cinco princípios básicos: alinhamento estratégico, criação de valor, gestão de riscos, gestão de recursos e mensuração de desempenho. Esses princípios complementam os mencionados por Rêgo (2013), fornecendo um guia completo para a implementação da governança de TI em uma organização.

O alinhamento estratégico é fundamental para garantir que a tecnologia da informação esteja alinhada aos objetivos estratégicos da empresa. A criação de valor, por sua vez, é importante para garantir que a TI contribua para a criação de valor para a organização. A gestão de riscos é crucial para minimizar os riscos associados ao uso da tecnologia da informação, enquanto a gestão de recursos é essencial para garantir que os recursos de TI sejam utilizados de maneira eficiente e eficaz. Finalmente, a mensuração de desempenho permite que os gestores avaliem a eficácia da governança de TI e façam ajustes quando necessário (SENFT; GALLEGOS, 2008).

2.4 OBJETIVOS DA GOVERNANÇA DE TI

De acordo com Van Grembergen e De Haes (2020), os principais objetivos da governança de TI incluem alinhar a TI com os objetivos de negócio da organização, garantir a entrega de valor por meio da TI, gerenciar os riscos de TI de forma adequada e garantir a conformidade com as leis e regulamentos aplicáveis.

Na visão de ISACA (2019), os objetivos da governança de TI consistem em alinhar a TI com os objetivos do negócio, fornecer valor aos *stakeholders*, gerenciar os riscos relacionados à TI, garantir conformidade com as leis e regulamentações aplicáveis e maximizar o retorno sobre os investimentos em TI

Segundo Weill e Ross (2006), os objetivos da governança de TI são garantir o uso efetivo e eficiente da tecnologia da informação em uma organização. Para atingir esses objetivos, é importante implementar os princípios da governança de TI, que incluem transparência, responsabilidade, equidade, conformidade e orientação

para resultados, bem como os princípios fundamentais de alinhamento estratégico, criação de valor, gestão de riscos, gestão de recursos e mensuração de desempenho.

2.5 ESTRUTURA DA GOVERNANÇA DE TI

A governança de TI é fundamental para garantir a efetividade e a eficiência das operações de uma organização. A estrutura da governança de TI inclui diversas políticas, processos e procedimentos que visam garantir que a TI esteja alinhada com os objetivos do negócio e que os riscos associados ao uso da tecnologia sejam gerenciados adequadamente (WEILL; ROSS 2006).

De acordo com Weill e Ross (2006), a estrutura da governança de TI é composta por três elementos principais: a tomada de decisão, o monitoramento e a estratégia. A tomada de decisão é responsável por definir as prioridades da TI e alocar recursos para projetos e iniciativas. O monitoramento é responsável por avaliar o desempenho da TI e garantir que as metas e objetivos sejam alcançados. Por fim, a estratégia é responsável por definir a direção da TI e garantir que as iniciativas estejam alinhadas com os objetivos do negócio.

Além desses elementos, a estrutura da governança de TI também inclui a definição de papéis e responsabilidades para os envolvidos no processo de governança. Isso envolve a definição de comitês de governança de TI, a atribuição de responsabilidades para os gestores de TI e a definição de papéis e responsabilidades para os usuários finais (CALDER, 2014).

Segundo Calder (2014), a estrutura da governança de TI também deve incluir mecanismos de comunicação eficazes para garantir que todas as partes interessadas estejam cientes das decisões e atividades de governança. Isso inclui a divulgação de políticas e procedimentos de TI, bem como a realização de treinamentos e *workshops* para garantir que os usuários finais compreendam os riscos associados ao uso da tecnologia.

2.6 BENEFÍCIOS DA BOA GOVERNANÇA DE TI

A implementação de boas práticas de governança de TI traz uma série de benefícios para as organizações. De acordo com Van Grembergen e De Haes (2020), a adoção de práticas de governança de TI efetivas pode trazer diversos benefícios para a organização, incluindo a melhoria da utilização dos recursos de TI, a capacitação dos recursos humanos envolvidos, a otimização dos processos de negócio e o aumento do valor entregue pela tecnologia.

Além disso, a governança de TI traz transparência e prestação de contas para as decisões de TI, ajudando a reduzir riscos e aumentando a confiança dos *stakeholders*. Com a governança de TI, os riscos associados à tecnologia são gerenciados de forma mais efetiva, permitindo que as organizações respondam melhor a situações adversas (WEILL; ROSS, 2006).

2.7 BANCOS DE DADOS

Os bancos de dados são uma tecnologia central em muitas aplicações modernas, permitindo que grandes quantidades de dados sejam gerenciadas, processadas e armazenadas de forma eficiente e segura (DATE, 2004).

Os bancos de dados são capazes de manter a integridade dos dados, garantindo que as informações armazenadas sejam precisas e confiáveis, e suportam a interação de várias pessoas e sistemas simultaneamente sem que ocorram conflitos (DATE, 2004).

A principal vantagem dos bancos de dados é a capacidade de armazenar grandes quantidades de informações de forma estruturada. Isso permite que as informações sejam recuperadas de forma rápida e eficiente, o que é especialmente importante em ambientes de negócios onde o tempo é crítico (DATE, 2004).

Para que estes objetivos possam ser alcançados, os dados precisam ser dispostos de forma a facilitar a extração de informação. Em meados de 1970 surgiu em projetos da IBM o modelo relacional de banco de dados, que atualmente é

amplamente adotado por sistemas do mundo todo. O modelo estipula a criação de entidades, que são pequenos conjuntos de dados de estrutura fixa e que se relacionam umas com as outras. “A maior vantagem do modelo relacional sobre seus antecessores é a representação simples dos dados e a facilidade com que consultas complexas podem ser expressas” (MACÁRIO, 2005, p. 2).

Além das qualidades pertinentes aos dados, os sistemas gerenciadores de bancos de dados contribuem para a confiabilidade e eficiência das soluções. “O gerenciamento de banco de dados continua a ganhar importância conforme mais e mais dados tornam-se disponíveis *online* e ainda mais acessíveis através da rede de computadores” (RAMAKRISHNAN; GEHRKE, 2008, p. 6). De acordo Ramakrishnan e Gehrke (2008) é responsabilidade dos Sistemas Gerenciadores de Bancos de Dados (SGBDs) garantir:

a) Independência de Dados: os aplicativos que consomem a base da dados não devem ter acesso detalhado sobre o armazenamento dos dados, mas apenas uma visão abstrata suficiente para o processamento externo;
b) Acesso Eficiente aos Dados: isso é realizado por meio de técnicas de armazenamento e consulta otimizadas para carga e velocidade;
c) Integridade e Segurança dos Dados: o controle de acesso é realizado utilizando técnicas de autenticação e autorização a funcionalidades delimitadas por usuário;
d) Administração de Dados: os gerenciadores oferecem ferramentas para otimização de registros repetidos, e também para criação de soluções mais robustas no contexto de arquitetura de infraestrutura;
e) Acesso Concorrente e Recuperação de Falha: o gerenciador cuida internamente dos acessos simultâneos, tornando a experiência externa mais simples ao mesmo tempo que padroniza a comunicação de respostas de sucesso e falha;

f) Tempo Reduzido de Desenvolvimento de Aplicativo: muitas ferramentas de consulta e inserção de dados simplificam o uso externo com interfaces de alto nível já validadas, reduzindo custos e tempo de desenvolvimento e testes.

2.7.1 Linguagem SQL

A linguagem *Structured Query Language* (SQL) é uma linguagem de consulta estruturada usada para gerenciar e manipular bancos de dados relacionais. Desenvolvida originalmente pela IBM nos anos 70, a SQL tornou-se uma linguagem padrão na indústria de bancos de dados. A linguagem SQL permite que os usuários acessem e manipulem dados em um banco de dados por meio de comandos específicos (RAMAKRISHNAN; GEHRKE, 2008).

Uma das principais características da linguagem SQL é sua capacidade de manipular dados de várias tabelas em um banco de dados relacionado. Os usuários podem criar consultas que combinam dados de diferentes tabelas em um único resultado, com operações de agregação em dados, como somas, médias e contagens. Os usuários podem usar comandos SQL para agrupar dados com base em diferentes critérios, permitindo que eles obtenham informações úteis e resumidas a partir de grandes conjuntos de dados (MACÁRIO, 2005).

Muitas vezes agregadas no mesmo nome “SQL”, as linguagens *Data Definition Language* (DDL) e *Data Manipulation Language* (DML) seguem propósitos similares ao padronizar e simplificar o gerenciamento e uso do banco de dados, porém oferecendo recursos de gerenciamento de banco de dados: a capacidade de criar, modificar e excluir tabelas e outros objetos, bem como controlar o acesso de usuários e garantir a integridade dos dados (RAMAKRISHNAN; GEHRKE, 2008).

Estas linguagens de banco de dados são utilizadas por usuários humanos capacitados e por aplicações já programadas, alimentando, consumindo e configurando as bases de dados, localmente ou através da internet (RAMAKRISHNAN; GEHRKE, 2008).

2.8 APLICAÇÕES *WEB*

As aplicações *web* são uma forma de *software* que pode ser acessada através de um navegador da *web* e usada para várias finalidades, como comércio eletrônico, gerenciamento de projetos e redes sociais, as aplicações *web* estão se tornando cada vez mais populares devido à sua acessibilidade e facilidade de uso (SHARMA, 2018).

A camada de apresentação é responsável por exibir a interface gráfica da aplicação para o usuário, essa camada pode ser desenvolvida usando tecnologias como *Hyper Text Markup Language* (HTML), *Cascading Style Sheet* (CSS) e JavaScript, que permitem a criação de páginas *web* dinâmicas e interativas (DUCKETT, 2015).

A camada de aplicação é responsável por processar as solicitações do usuário e fornecer as respostas correspondentes, essa camada pode ser desenvolvida utilizando diferentes tecnologias, como servidores *web*, *frameworks* de desenvolvimento de *software* e linguagens de programação, entre outros (CANTELON *et al*., 2019).

Por fim, a camada de dados é responsável por armazenar e gerenciar os dados da aplicação, essa camada pode ser composta por diferentes tipos de bancos de dados, como bancos de dados relacionais e bancos de dados NoSQL (CHODOROW, 2013).

Existem diferentes tecnologias e *frameworks* que podem ser utilizados para o desenvolvimento de aplicações *web*, a escolha da tecnologia mais adequada para o desenvolvimento de uma aplicação *web* depende das necessidades específicas do projeto, como requisitos de desempenho, segurança e escalabilidade (SHARMA, 2018).

Em conclusão, as aplicações *web* são um tipo importante de *software* que pode ser acessado através de um navegador da *web*. Elas são compostas por diferentes camadas e podem ser desenvolvidas usando diferentes tecnologias e *frameworks*, dependendo das necessidades do projeto (SHARMA, 2018).

2.8.1 Protocolos de Comunicação *WEB*

Os protocolos de comunicação *web* são de extrema importância para garantir a transferência segura de dados na internet. De acordo com Kurose e Ross (2017), alguns dos protocolos mais utilizados na *web* incluem *Hyper Text Transfer Protocol* (HTTP), *Hyper Text Transfer Protocol Secure* (HTTPS), *File Transfer Protocol* (FTP) e *WebSocket.*

O protocolo HTTP é utilizado para transmitir informações entre navegadores e servidores *web*, permitindo que os usuários acessem diferentes tipos de conteúdo *online*. Segundo Kurose e Ross (2017), o protocolo HTTP é um protocolo sem estado, o que significa que ele não mantém informações de sessão entre as requisições.

Por sua vez, o protocolo HTTPS é uma versão segura do protocolo HTTP que utiliza criptografia para proteger as informações transmitidas entre o navegador e o servidor. Como destacam Tanenbaum e Wetherall (2011), o uso do protocolo HTTPS é fundamental para garantir a privacidade dos usuários e proteger informações confidenciais, como senhas e informações bancárias.

Já o protocolo FTP é utilizado para transferir arquivos entre o servidor e o cliente. Conforme explica Kurose e Ross (2017), o protocolo FTP é um protocolo antigo, mas ainda amplamente utilizado em todo o mundo, especialmente para a transferência de arquivos grandes.

O protocolo *WebSocket*, por sua vez, é uma tecnologia de comunicação em tempo real que permite que os servidores *web* enviem dados para os navegadores sem a necessidade de uma solicitação específica. De acordo com Tanenbaum e Wetherall (2011), o protocolo *WebSocket* é amplamente utilizado em aplicações *web* que requerem comunicação bidirecional em tempo real, como jogos *online* e aplicações de bate-papo.

Existem outros protocolos de comunicação *web*, como o protocolo *File Transfer Protocol over SSL* (FTPS), que é uma versão segura do protocolo FTP, e o protocolo *Real Time Streaming Protocol* (RTSP), que é utilizado para a transmissão de conteúdo de áudio e vídeo em tempo real. Todos esses protocolos são

fundamentais para a transmissão de dados na *web* e são amplamente utilizados em diferentes contextos e aplicações (KUROSE; ROSS, 2017).

2.8.2 *Application Programming Interface* (API)

As interfaces de programação de aplicativos, ou APIs, tornaram-se parte integrante do desenvolvimento de *software* moderno, permitindo comunicação e interação entre diferentes sistemas de *software*. As APIs servem como um conjunto de protocolos e ferramentas que permitem aos desenvolvedores criar e integrar aplicativos, permitindo que eles interajam entre si e compartilhem dados e funcionalidades (BOJINOV, 2015).

As APIs atuam como intermediários que permitem a comunicação entre diferentes componentes de *software* ou sistemas, permitindo que eles trabalhem juntos de maneira coordenada. Definem, portanto, uma forma padronizada para as aplicações interagirem entre si, especificando as regras e protocolos de comunicação (DOGLIO, 2015).

As APIs podem ser projetadas para uma ampla variedade de finalidades, desde o acesso a dados e serviços até a execução de operações complexas, e são amplamente usadas no desenvolvimento da *web*, desenvolvimento de aplicativos móveis e outras áreas de desenvolvimento de *software* (BOJINOV, 2015).

No entanto, é importante observar que as APIs também apresentam desafios e considerações. A segurança é um aspecto crucial do *design* da API, pois as APIs podem expor dados e funcionalidades confidenciais. Portanto, mecanismos adequados de autenticação, autorização e criptografia de dados devem ser implementados para garantir a segurança e a privacidade da API e de seus usuários. Além disso, a documentação e o controle de versão da API são essenciais para manter a compatibilidade e fornecer instruções claras aos desenvolvedores sobre como usar a API com eficiência (NANDAA, 2018).

2.9 MODELAGEM DE *SOFTWARE*

A modelagem de *software* é o processo de criar representações esquemáticas de sistemas de *software* usando notações, símbolos e diagramas para representar seus aspectos estruturais, comportamentais e funcionais. É uma abordagem visual e gráfica para representar sistemas de *software* e seus componentes, relacionamentos e interações de maneira clara e concisa. A modelagem de *software* permite que os desenvolvedores capturem e documentem as decisões de *design*, requisitos e especificações de sistemas de *software*, tornando mais fácil entender, comunicar e analisar o comportamento e a estrutura do sistema (WAZLAWICK, 2019).

A modelagem de *software* desempenha um papel crucial no processo de desenvolvimento de *software*, oferecendo vários benefícios que contribuem para o sucesso geral dos projetos de *software*. Algumas das principais razões pelas quais a modelagem de *software* é importante são (GUEDES, 2018):

a) Visualização: os modelos de *software* fornecem uma representação visual do sistema de *software*, tornando mais fácil para as partes interessadas entender a arquitetura, os componentes e as interações do sistema. A visualização aprimora a comunicação, a colaboração e a tomada de decisões entre os membros da equipe, partes interessadas e usuários, levando a um entendimento compartilhado do sistema;
b) *Design*: A modelagem de *software* permite que os desenvolvedores criem e refinem o *design* do sistema de *software* de maneira sistemática e estruturada. Ele ajuda a identificar os componentes, relacionamentos e interações entre os elementos do sistema, permitindo que os desenvolvedores tomem decisões de *design* informadas e criem um sistema robusto e escalável;
c) Análise: os modelos de *software* fornecem um meio para analisar o comportamento, o desempenho e a confiabilidade dos sistemas de *software*.

Ao simular, validar e verificar o comportamento do sistema, os modelos podem ajudar na identificação de possíveis problemas, riscos e defeitos no início do processo de desenvolvimento, reduzindo a probabilidade de erros dispendiosos e retrabalho;

d) Documentação: os modelos de *software* servem como uma ferramenta de documentação que captura as decisões de *design*, requisitos e especificações do sistema de *software*. Os modelos podem ser usados como referência para entender o comportamento, a estrutura e a funcionalidade do sistema, facilitando as atividades de manutenção, depuração e solução de problemas;

e) Reutilização: os modelos de *software* podem ser reaproveitados no projeto, permitindo que os desenvolvedores acelerem as etapas de análise, *design* e documentação. Isso fornece consistência e aumenta a eficiência e produtividade nos projetos de *software*.

2.10 PROTOTIPAGEM DE *SOFTWARE*

A prototipagem é uma técnica amplamente utilizada no desenvolvimento de *software* que envolve a criação de um modelo funcional de um sistema ou de um recurso específico para coletar *feedback*, avaliar opções de *design* e validar requisitos. A prototipagem permite que os desenvolvedores criem rapidamente representações tangíveis de soluções de *software*, permitindo que as partes interessadas visualizem e interajam com a funcionalidade, interface e comportamento do sistema (TOWEH, 2019).

A prototipagem de *software* oferece vários benefícios que contribuem para o sucesso geral dos projetos de desenvolvimento de *software*. Alguns dos principais benefícios da prototipagem de *software* são (McELROY, 2016):

a) *Feedback* antecipado: A prototipagem permite que as partes interessadas, como usuários, clientes e patrocinadores do projeto, forneçam *feedback* antecipado sobre a funcionalidade, interface e comportamento do sistema.

Isso ajuda a identificar problemas, lacunas de requisitos e preocupações de usabilidade no início do processo de desenvolvimento, reduzindo a probabilidade de erros dispendiosos e retrabalho em estágios posteriores;

b) Validação de Requisitos: A prototipagem permite a validação de requisitos, fornecendo uma representação tangível da funcionalidade do sistema. As partes interessadas podem interagir com o protótipo e fornecer *feedback* sobre se o sistema atende aos seus requisitos ou não;

c) Exploração de *design*: a prototipagem permite que os desenvolvedores explorem diferentes opções no projeto avaliem sua viabilidade de maneira tangível. Ao criar vários protótipos com diferentes alternativas de *design*, os desenvolvedores podem comparar e avaliar os prós e contras de cada opção, levando a decisões de *design* informadas e a uma solução de *software* mais robusta;

d) Comunicação aprimorada: a prototipagem facilita a comunicação eficaz entre a equipe, permitindo que os desenvolvedores possam expor suas capacidades e intenções, deixando evidente para o cliente os planos de desenvolvimento e permitindo melhores decisão de *design*.

3 ANÁLISE DOS DADOS DA EMPRESA

Neste capítulo serão apresentadas a análise de requisitos levantados baseados na necessidade da empresa estudada, as soluções levantadas, um protótipo para demonstração de viabilidade e a elaboração de um plano de ação para o cliente.

3.1 ANÁLISE DE REQUISITOS

A análise de requisitos é uma metodologia amplamente utilizada pelas empresas para identificar necessidades, o que permite melhor compreender um problema e projetar soluções mais certeiras e otimizadas. No caso da empresa

estudada, seu desejo maior é oferecer uma solução de busca de produtos compatíveis com um modelo de veículo, recebendo apenas a placa veicular como dado de entrada. Neste cenário, que fora explorado pela equipe em reuniões online com a empresa cliente, deu-se que sua maior dificuldade atualmente é encontrar uma API para consultar dados de veículos a partir da placa.

Em um cenário problemático como este, os requisitos da solução podem estar correlacionados às causas desse problema – e, portanto, é necessário entendê-las. É imprescindível identificar as possíveis razões que levaram à dificuldade em encontrar uma API para obtenção de dados de veículos. Dentre as possíveis causas, podem ser consideradas:

a) Fraqueza nos critérios: é possível que a empresa estudada não tenha realizado uma pesquisa ampla o suficiente para encontrar a API adequada para visualização de veículos. Talvez não tenham procurado em fontes confiáveis ou não tenham levado em conta critérios relevantes para a escolha da API;
b) Falta de comunicação entre as equipes: é possível que a equipe responsável por encontrar a API não tenha se comunicado adequadamente com as demais equipes para entender quais eram as necessidades específicas da empresa, a nível de negócio e desenvolvimento;
c) Escolha inadequada de uma API: é possível que a empresa estudada tenha escolhido uma API inadequada para visualização de veículos. A API escolhida pode não ter as funcionalidades necessárias para atender às necessidades da empresa, pode não ser compatível com o sistema utilizado pela organização ou pode ter se demonstrado inviável financeiramente.

3.2 ALTERNATIVAS DE SOLUÇÃO

A necessidade de visualizar informações sobre veículos é essencial para as operações diárias da empresa, bem como para a tomada de decisões estratégicas e

para melhorar seus processos. No entanto, encontrar a solução certa pode ser complicado, já que existem muitas APIs diferentes no mercado e cada uma tem suas vantagens e desvantagens.

Após uma análise detalhada, a equipe responsável pela solução do problema identificou três possíveis soluções para a empresa estudada. A primeira solução é uma API que pertence ao DENATRAN. Essa API oferece informações precisas e detalhadas sobre veículos e é amplamente utilizada por empresas em todo o país. No entanto, essa solução exige um certificado digital e impõe custos significativos por requisição, sendo possíveis fatores limitantes para a organização.

A segunda solução identificada é uma API gratuita, que está atualmente em fase de testes. Este é um projeto desenvolvido sem suporte, e em seu estado atual, não funcionou em nossos testes preliminares – apesar de ter funcionado em versões antigas de seu projeto. Embora essa solução tenha um custo muito menor do que a API do DENATRAN, ainda está em fase de desenvolvimento e pode não ser tão confiável quanto a solução paga. Dado o estado atual, a equipe responsável pela solução do problema deve avaliar cuidadosamente se a solução gratuita atende às necessidades específicas da empresa estudada.

A terceira solução encontrada é uma API que acessa a base do DENATRAN, e funciona como um intermediário, simplificando a burocracia de acesso aos dados. A API é denominada API Placas, e também impõe um custo para realizar as requisições, mas não exige certificado digital – reduzindo custos a curto prazo. Na plataforma da API Placas é possível gerenciar o plano de assinatura escolhido, o token de autenticação para integração em sistemas, e até mesmo criar uma conta gratuita para realização de testes. Não pudemos apurar se este projeto será mantido no futuro, mas isso precisa ser considerado, pois pode se tornar um impeditivo a longo prazo.

É importante ressaltar que as soluções identificadas têm seus prós e contras, e os envolvidos mais avidamente na implementação da solução devem avaliar cuidadosamente cada uma delas antes de tomar uma decisão final. Algumas das questões que devem ser consideradas incluem o custo, a precisão e a confiabilidade

das informações fornecidas pela API, a compatibilidade com o sistema utilizado pela empresa estudada e as necessidades específicas da organização, além da disponibilidade e longevidade da API.

A equipe responsável pela solução do problema deve envolver os usuários da empresa estudada e outras partes interessadas no processo de tomada de decisão para garantir que a escolha final atenda às necessidades da organização. A equipe também deve realizar testes adicionais para avaliar a qualidade da solução gratuita em desenvolvimento antes de decidir se é uma escolha viável para a empresa estudada.

3.3 PROPOSTA DE SOLUÇÃO

Para esta proposta de solução, sugerimos o uso da API Placas. A escolha se deu pela facilidade de integração, cuja assinatura e token de conexão podem ser gerenciados no site da aplicação, e pela não necessidade de certificação digital para operação.

Estes fatores contribuem para que o projeto possa ser conduzido com o mínimo de interrupções ou manutenções possíveis - o que se traduz indiretamente à menores custos à longo prazo, comparado à outras APIs e serviços. O baixo risco e impacto em caso de dificuldades técnicas, aliado da implementação de serviços confiáveis e bem documentados, são fatores constituintes em um projeto de TI de sucesso.

Para solução do problema da empresa estudada, é necessário integrá-la ao sistema existente da empresa – isto é, incluindo seu catálogo digital e a interface de pesquisa do usuário final. Para isso, a equipe deve seguir os seguintes passos:

a) Analisar a documentação da API Placas: A equipe deve se familiarizar com a documentação da API e entender como ela funciona. Isso inclui compreender a estrutura dos dados que a API retorna, os parâmetros de

consulta que podem ser utilizados, as limitações e requisitos de autenticação;

b) Preparar os dados: Antes de integrar a API ao sistema da empresa estudada, é preciso garantir que os dados disponíveis no catálogo digital estejam bem estruturados e organizados, a fim de que possam ser facilmente relacionados aos dados retornados pela API. A equipe pode precisar fazer ajustes na base de dados para que ela possa ser integrada com a API. Também pode ser proveitoso manipular os resultados da API a fim de melhorar sua compatibilidade com a base de dados do catálogo digital;

c) Desenvolver as integrações: Com o acesso da API configurado e os dados de entrada e de relacionamento prontos, a equipe pode começar a desenvolver as integrações. É importante definir os pontos de integração do sistema com a API, como por exemplo, definir como as consultas serão feitas – diretamente pelo front-end, ou passando por um gerenciador back-end proprietário da empresa. É importante também prever possíveis erros e exceções que podem acontecer durante o processo de integração;

d) Realizar testes: Após o desenvolvimento das integrações, é preciso realizar testes para garantir que tudo esteja funcionando corretamente. Isso inclui testar os fluxos de consulta da API, as atualizações de dados, e as integrações entre os sistemas. Os testes devem ser feitos em ambiente de testes, e caso sejam encontrados problemas, eles devem ser corrigidos antes da implementação em produção;

e) Implementar em produção: Após os testes, a equipe pode implementar as integrações em produção. É importante monitorar o sistema para garantir que tudo esteja funcionando como esperado e realizar ajustes para melhorar a performance ou corrigir problemas;

f) Manter o sistema: Uma vez que o sistema está funcionando em produção, é preciso monitorá-lo e mantê-lo atualizado. Isso inclui atualizar os

softwares, corrigir problemas e garantir que o sistema continue atendendo às necessidades da empresa.

3.4 PROTÓTIPO COMO PROVA DE CONCEITO

Como prova de conceito da viabilidade de uso desses serviços digitais em formato de API de consulta, nossa equipe aplicou a API Placas em um protótipo de sistema WEB, visando demonstrar a experiência do usuário final. Nesta demonstração, o usuário insere a placa de sua moto, e a partir do modelo coletado pela API Placas, são retornadas peças compatíveis com seu modelo.

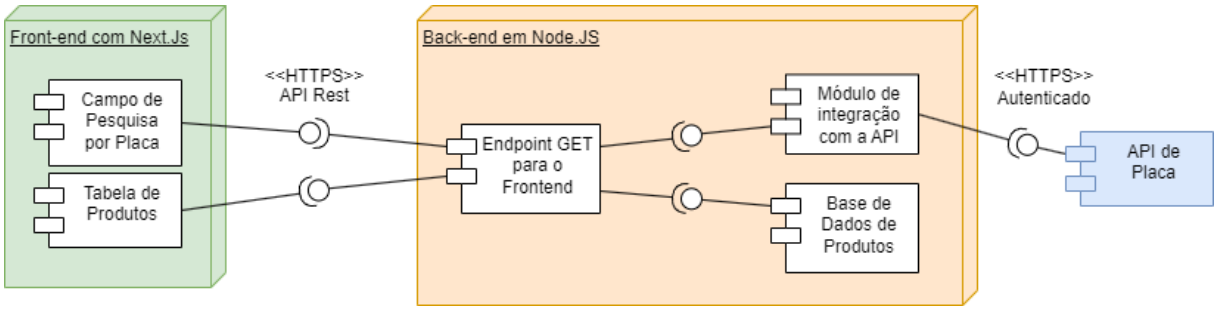
Além do serviço de consulta de placa, a aplicação faz uso de uma base de dados fictícia que armazena informações sobre modelos e peças. Essa base de dados foi criada com base no negócio da empresa cliente mediante a compreensão de seu catálogo digital, considerando apenas as entidades relacionais envolvidas nos modelos de veículos e produtos compatíveis.

A aplicação foi desenvolvida utilizando diversas tecnologias, incluindo Next.js e Node.js. O Next.js é um framework de construção de páginas web de código aberto oferecido pela Vercel queoferece funcionalidades de geração de sites estáticos para aplicativos da web baseados em React. Já o Node.js é uma plataforma de desenvolvimento que permite a criação de aplicações em JavaScript do lado do servidor. O MySQL é um banco de dados relacional, cujo acesso é realizado apenas pelo back-end, de forma controlada e segura. O banco de dados e outras estruturas de dados mais simplistas foram experimentadas durante a construção do protótipo, visando simular o armazenamento dos dados relacionados, de modelos de veículos e peças compatíveis.

3.4.1 Diagrama UML de Componentes

O diagrama de componentes da Figura 1 visa evidenciar as diferentes partes deste protótipo de sistema web, quais suas responsabilidades e como eles se comunicam entre si.

FIGURA 1: DIAGRAMA UML DE COMPONENTES



FONTE: OS AUTORES (2023).

No âmbito do front-end, a área de interação do usuário conta com um campo de pesquisa por placa e uma área para exibição dos produtos retornados. A interface do usuário faz requisições HTTPS ao servidor back-end em Node.js por meio de uma API Rest.

O processamento da resposta inicia com a consulta na API Placas. A comunicação com o serviço online faz uso de um token de validação, que permite o provedor do serviço contar as requisições por cliente, bloqueando o acesso à usuários não autorizados ou sem requisições disponíveis.

Uma vez que as informações do veículo sejam coletadas, o modelo de veículo obtido é usado em uma pesquisa de produtos, acessando a base de dados fictícia. A adoção do banco de dados pode depender de alguns fatores, incluindo o formato atual do catálogo da empresa, que pode estar disposto em uma planilha ou outra forma de armazenamento de texto e imagens. Quando os produtos retornarem da base de dados, a resposta é encaminhada ao front-end, que se encarrega de validar as informações recebidas para popular a tabela de produtos.

É importante ressaltar que a aplicação apresentada foi construída para fins de demonstração e validação de conceito, e que a base de dados fictícia utilizada poupa detalhes de meio e estrutura, buscando apenas reproduzir uma relação genérica entre modelos de veículos e peças compatíveis. No ambiente de produção real, essa base de dados deverá ser substituída por um catálogo digital, em um sistema de banco de dados mais robusto, completo e representativo de seu negócio.

3.4.2 Interface do Usuário

A interface do usuário foi construída com o framework Next.js, que abstrai as camadas de segurança e autenticação para comunicação com servidores, e facilita a manipulação de dados e criação de estruturas dinâmicas. A estrutura das páginas é feita em HTML, a estilização com CSS, e os comportamentos lógicos são escritos em Typescript, e o Next.js abrange e aprimora o workflow destes três escopos do front-end.

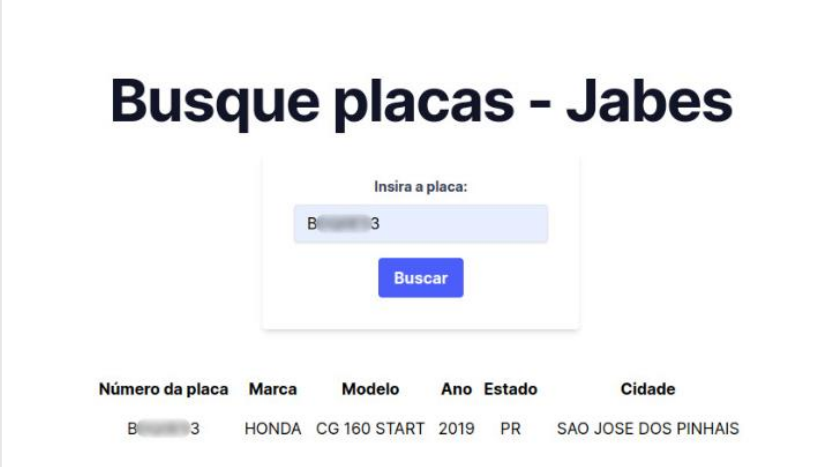
Visualmente, o protótipo conta com um campo simples para inserção da placa, que deve seguir o padrão das placas utilizadas em território brasileiro, e um botão “Buscar” para executar a busca. Conforme exposto na Figura 2.

FIGURA 2: INTERFACE GRÁFICA EM ESTADO INICIAL

FONTE: OS AUTORES (2023).

Quando o usuário insere uma placa de formato válido e pressiona o botão buscar, uma requisição será feita ao servidor em Node.js. O back-end consulta a API Placas, autenticando a requisição com o token fornecido pelo serviço, e retorna à interface gráfica as informações do modelo de veículo obtido. As informações de número da placa, marca, modelo, ano, estado e cidade são expostas, como mostra a Figura 3.

FIGURA 3: INTERFACE GRÁFICA EM PESQUISA DE PLACA

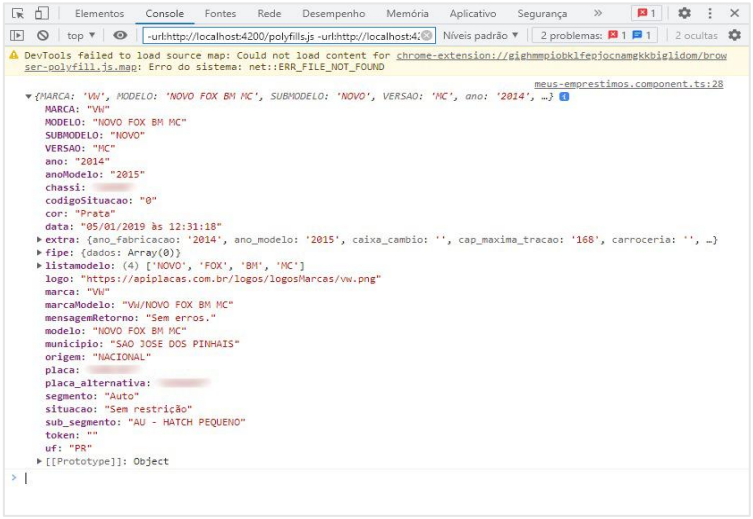


FONTE: OS AUTORES (2023).

Para executar a pesquisa dos produtos, nosso protótipo solicita apenas o modelo, dada a estrutura relacional simples. No entanto, outros dados além do modelo são exibidos para que o usuário possa confirmar sua busca.

A API Placas retorna muitos dados que podem ser utilizados pela empresa estudada. A Figura 4 expõe grande parte da estrutura de dados fornecida.

FIGURA 4: ESTRUTURA PARCIAL DOS DADOS RETORNADOS DA API PLACAS



FONTE: OS AUTORES (2023).

Ao clicar nos dados do veículo retornados, em um ato de validação do modelo do veículo, os produtos compatíveis serão coletados e exibidos na parte inferior da interface gráfica, como mostra a Figura 5.

FIGURA 5: EXIBIÇÃO DE PRODUTOS RELACIONADOS AO MODELO

FONTE: OS AUTORES (2023).

Desta forma, é possível observar que para determinada placa, cujo modelo do veículo é “CG 160 START”, o sistema exibe os produtos “Chicote A-1", “Atuador 2XV” e “Bobina HY7”. A API Placas atende as necessidades da empresa a nível técnico, como exposto pelo protótipo, ao tornar viável a correlação entre placa e produto.

Em um sistema real, voltado para clientes finais, é ideal que a consulta por produtos traga informações relevantes disponíveis no catálogo, como por exemplo: nome; descrição; uma ou mais fotos; peso; dimensões; características visuais, mecânicas e elétricas; em contextos de *e-commerce*, preço e disponibilidade à pronta entrega.

3.5 PLANO DE AÇÃO

Para assegurar que o sistema proposto de fato atenda ao desejado pela empresa, é recomendado que sejam realizadas as sete ações propostas, descritas na coluna “O que será feito”, que vão desde avaliar a compatibilidade e requisitos da nova API com o banco de dados local até documentar o processo de integração, conforme descritas no plano de ação do Quadro 1. Na coluna “Por que será feito” descreve-se a razão pela qual a ação é necessária. As ações foram divididas em semanas, na coluna “Quando será realizado” e sob a responsabilidade do setor de TI, descritos na coluna “Quem será responsável”. Na coluna “Como será feito”, descreve-se o método sugerido para que a ação seja realizada.

QUADRO 1 – PLANO DE AÇÃO

Ação	O que será feito?	Por que será feito?	Quem será responsável?	Quando será realizado?	Como será feito?
1	Avaliar a compatibilidade e requisitos da nova API com o banco de dados local	Compreender a estrutura do banco de dados e como a API deverá ser adaptada para poder acessar as informações	Equipe de TI da empresa e Desenvolvedores da API	Semana 1	Reunião com os desenvolvedores e equipe técnica para analisar a documentação da API e verificar a compatibilidade com o banco de dados
2	Desenvolver um plano de integração e fluxo de dados	Estabelecer um plano claro para a integração entre o banco de dados e a nova API	Equipe de TI da empresa e Desenvolvedores da API	Semana 2	Criar um documento que descreva o processo de integração, incluindo as etapas necessárias e as tecnologias a serem utilizadas
3	Configurar as permissões e autenticação necessárias	Garantir a segurança e o acesso adequado aos dados do banco de dados	Equipe de TI	Semana 3	Definir as políticas de permissão e autenticação para o acesso à API e ao banco de dados
4	Desenvolver e testar os scripts de conexão	Criar os scripts necessários para conectar o banco de dados à nova API	Desenvolvedores da API	Semana 4	Desenvolver e testar os scripts de conexão, utilizando a linguagem de programação adequada
5	Implementar a integração entre a API e o site da empresa	Conectar a API ao site da empresa para disponibilizar	Desenvolvedores da API	Semana 5	Integrar a API ao código do site e realizar testes para garantir o

		os dados em tempo real			correto funcionamento
6	Realizar testes de integração e monitoramento contínuo	Verificar a integridade dos dados e o desempenho da integração	Equipe de TI	Semana 6	Executar testes abrangentes para garantir a estabilidade da integração e implementar monitoramento contínuo para identificar possíveis problemas
7	Documentar o processo de integração	Registrar todos os passos e configuraçõe s realizados durante a integração	Equipe de TI	Semana 7	Criar documentação detalhada sobre o processo de integração, incluindo configurações, scripts e possíveis soluções de problemas

FONTE: OS AUTORES (2023)

Acredita-se que estas ações propostas possam contribuir para implantar a solução para o desafio da empresa estudada, contribuindo para que os objetivos dela sejam atendidos, visto que a proposta de solução incorpora os requisitos de cliente.

4. CONSIDERAÇÕES FINAIS

A proposta de sistema apresentada neste trabalho contribui positivamente para os avanços de desenvolvimento de software na empresa estudada, levantando tecnologias de diversos níveis de robustez, maturidade e monetização para atender suas necessidades. Tanto o objetivo geral quanto os objetivos específicos deste

trabalho foram atingidos, visto que foi proposta uma solução para pesquisa de informações de veículos através de suas placas, a fim de garantir que a peça desejada, fornecida pela empresa, corresponda fielmente ao modelo do veículo do cliente.

A metodologia utilizada neste trabalho demonstrou ser suficiente, pois supriu as expectativas da equipe e permitiu que o objetivo geral e específicos fossem alcançados. O levantamento de tecnologias e serviços em API evidenciou que a integração da API Placas nas soluções web da empresa estudada é uma estratégia promissora, permitindo sobrepor as funcionalidades dos campos de pesquisa por modelo existentes e facilitar a conexão com seus catálogos digitais, minimizando possíveis atritos.

Contudo, a continuação do projeto se faz necessária para que o produto possa ser aplicado como ferramenta de mercado e comércio, dentro das expectativas e intenções da empresa estudada. A integração da API Placas pode ser construída diretamente em suas soluções WEB, a nível de *front-end*, sobrepondo as funcionalidades dos campos de pesquisa por modelo atuais, para que a conexão com seus catálogos digitais possa ocorrer com o mínimo de atrito possível.

Além dos esforços necessários de integração à sistemas exigentes, é possível desenvolver outros módulos em apoio à ferramenta. Ao armazenar informações de placas já pesquisadas – dado o cumprimento das obrigações legais exigidas pela Lei Geral de Proteção de Dados, é possível poupar as requisições pagas na API Placas, caso não exista a necessidade de utilizar dados atualizados dos veículos. Outra oportunidade é a construção de um algoritmo para sugerir “modelos de veículo próximos”, que busca nas bases da empresa modelos de veículo similares caso a API Placas retorne um modelo de veículo que não tenha sido cadastrado em seu catálogo digital.

REFERÊNCIAS

BLOK, Marcella. **Compliance e governança corporativa**. Freitas Bastos, 2020.

BOJINOV, Valentin. **RESTful Web API Design with Node.js**. Packt Publishing, 2015.

CALDER, A.,Watkins, S **IT Governance**: An International Guide to Data Security and ISO27001/ISO27002. Kogan Page Publishers, 2014.

CANTELON, Mike et al. **Node.js** - Guia do Desenvolvedor. Publicação do autor, 2019.

CHODOROW, Kristina. **MongoDB** - Guia do Usuário. Publicação do autor, 2013.

DATE, Christopher J. **Introdução a sistemas de bancos de dados**. Elsevier Brasil, 2004.

DOGLIO, Fernando. **Pro REST API Development with Node.js**. Apress, 2015.

DUCKETT, Jon. **JavaScript e jQuery**: Desenvolvimento de Interfaces Web Interativas. Publicação do autor, 2015.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa**. 6 ed. São Paulo: Atlas, 2017.

GUEDES, Gilleanes TA. **UML 2 -** Uma abordagem prática. Novatec Editora, 2018.

HARMER, Geoff. **Governance of enterprise IT based on COBIT 5**: a management guide. IT Governance Ltd, 2014.

ISACA. **COBIT 2019 Framework**: Introduction and Methodology. Rolling Meadows, IL: ISACA, 2019.

KUROSE, Jim F.; ROSS, Keith W. **Redes de computadores e a Internet**: uma abordagem top-down. 6. ed. São Paulo: Pearson, 2017.

MACÁRIO, Carla Geovana do N.; BALDO, Stefano Monteiro. **O modelo relacional**. Instituto de Computação Unicamp. Campinas, 2005.

MALL, Rajib. **Fundamentals of software engineering**. PHI Learning Pvt. Ltd., 2018.

McELROY, Kathryn. **Prototyping for designers**: Developing the best digital and physical products. "O'Reilly Media, Inc.", 2016.

NANDAA, Anthony. **Beginning API Development with Node.js**: Build highly scalable, developer-friendly APIs for the modern web with JavaScript and Node.js. Packt Publishing Ltd, 2018.

POLIZEL, Caio. **Governança corporativa na educação superior**. Saraiva Educação SA, 2017.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. **Sistemas de gerenciamento de banco de dados**. AMGH Editora, 2008.

RÊGO, Bergson Lopes. **Gestão e governança de dados:** promovendo dados como ativo de valor nas empresas. Brasport, 2013.
RIES, Eric. **A Startup Enxuta**. Leya Brasil, 2017.

SENFT, Sandra; GALLEGOS, Frederick. **Information technology control and audit.** CRC Press, 2008.

SEVERINO, Antônio Joaquim. **Metodologia do Trabalho Científico.** São Paulo, Cortez Editora, 2014.

SHARMA, Aneeta. **Full-Stack Web Development with Vue.js and Node**: Build scalable and powerful web apps with modern web stack: MongoDB, Vue, Node.js, and Express. Packt Publishing, 2018.

TANENBAUM, Andrew S.; WETHERALL, David J. **Redes de Computadores**. 5. ed. São Paulo: Pearson, 2011.

TOWEH, Freedom. **Does Prototyping Help or Hinder Good Requirements?** What Are the Best Practices for Using This Method?. Trafford Publishing, 2019.

VAN GREMBERGEN, W.; DE HAES, S **Governance of Enterprise IT based on COBIT® 5:** A Management Guide. Berlin: Springer. 2020.

WAZLAWICK, Raul**. Engenharia de software:** conceitos e práticas. Elsevier Editora Ltda., 2019.

WEILL, Peter; ROSS, Jeanne. W. **Governança de TI:** tecnologia da informação. São Paulo: M. Books, 2006.

WILSON, Chauncey. **Interview Techniques for UX Practitioners**: A User-Centered Design Method. Morgan Kaufmann, 2013.

[Recebido/Received: Abril 30, 2023; Aceito/Accepted: Agosto 29, 2023]