Relato de Experiência do Desenvolvimento de um Sistema Web de Gestão para a Concessionária Apollo Veículos

Ali O. Chehade[1], **Ariel M. Rodrigues**[1], **Claudir F. Júnior**[1], **Daniele S. Fonseca**[1], **Rafael A. Scarpelli**[1], **Rômulo J. Bogoni**[1], **Frank W. C. de Oliveira**[1], **Marcelo F. Terenciani**[1]

[1]Campus Paranavaí – Instituto Federal do Paraná (IFPR)
Paranavaí – PR – Brasil

{20230006450, 20230010210, 20230006370 20230006334,
{20230009210, 20230006183}@estudantes.ifpr.edu.br
{frank.willian, marcelo.terenciani}@ifpr.edu.br

Este resumo apresenta um sistema web desenvolvido ao longo do ano letivo de 2025 nas disciplinas de Construção de Software e Projeto Integrador no curso de Engenharia de Software do Instituto Federal do Paraná (IFPR). As disciplinas tem como objetivo oferecer aos alunos uma simulação de ambiente de trabalho próximo ao mercado real. Para este trabalho, foi selecionada como estudo de caso a empresa Apollo Veículos, por representar uma situação comum em pequenas e médias empresas que ainda utilizam registros manuais ou semi-automatizados.

O principal problema identificado foi a ausência de uma ferramenta proprietária digital para a divulgação do inventário de veículos. Isso limitava a manutenibilidade do sistema e adição de novas funcionalidades, o que impacta na visibilidade da empresa no mercado e dificulta a prospecção de clientes. Além disso, a falta de sincronização de dados entre o catálogo interno de veículos e as plataformas externas gera inconsistências de informação e, consequentemente, a perda de potenciais oportunidades de negócio.

Diante desse cenário, definiu-se como escopo o desenvolvimento de um sistema *web* para a concessionária, contemplando tanto a gestão interna de veículos quanto a exposição do catálogo ao público. Para isso, formou-se uma equipe e definir os papéis de cada integrante baseando-se no *framework* Scrum [Schwaber and Sutherland 2020], realizar o levantamento de requisitos junto aos proprietários da empresa, documentar e estruturar os requisitos no formato de histórias de usuário, definir tecnologias adequadas para *front-end*, *back-end* e banco de dados, prototipar a interface no Figma para validação de usabilidade, e, implementar as funcionalidades prioritárias no sistema.

O desenvolvimento do projeto foi organizado a partir da combinação de práticas de Engenharia de Software e do *framework* ágil *Scrum*, de forma a estruturar tanto o levantamento de requisitos quanto a implementação técnica [Pressman and Maxim 2016, Schwaber and Sutherland 2020]. O levantamento de requisitos ocorreu em reuniões com os *stakeholders* (clientes), realizadas com visitas à empresa, conversas por aplicativo de mensagens ou em pequenos *brainstorms*. Esse processo permitiu identificar as necessidades principais do sistema e validar as prioridades junto aos usuários [Wiegers and Beatty 2013, Sommerville 2019, Kotonya and Sommerville 1998]).

A equipe foi composta por seis integrantes, divididos nos papéis de *Scrum Master*, *Product Owner* e quatro desenvolvedores, sendo dois responsáveis pela interface gráfica (*front-end*) e dois pelas regras de negócio (*back-end*). As atividades foram conduzidas em modelo híbrido, com encontros presenciais nos laboratórios do IFPR e tarefas remotas. O trabalho foi organizado em *Sprints* de quatro a seis semanas, adaptadas ao calendário acadêmico, e acompanhado por meio de quadros Kanban [Anderson 2015].

O Github [Peter and Beer 2014] foi utilizado para gestão da equipe e de código. Dentro da plataforma foi criada uma organização[1] para utilização do quadro Kanban, juntamente com 3 repositórios. Um deles foi utilizado para o *front-end*, um para o *back-end* e um inicial para organização da *LandingPage*. Além disso, foram definidas as permissões de "Membros" para os quatro desenvolvedores e as de "Donos" ao *Scrum Master*, ao *Product Owner* e aos professores das disciplinas regulares que trabalham com esse projeto.

Com os requisitos definidos e a equipe organizada, foram escolhidas as tecnologias que dariam suporte à implementação. No *front-end*, utilizou-se a biblioteca *React*, que possibilitou a construção de uma interface modular e reutilizável. Antes da codificação, as interfaces gráficas de usuário foram prototipados no Figma [Figma 2025], permitindo validar a usabilidade e alinhar expectativas entre os membros da equipe. No *back-end*, adotou-se o *framework Spring Boot*, devido à robustez e ao suporte à criação de *APIs RESTful*. Para o armazenamento e gerenciamento das informações, foi empregado o banco de dados MySQL [Oracle 2025]. O versionamento do código foi realizado com Git/GitHub, o que viabilizou o trabalho colaborativo e a integração contínua.

As funcionalidades priorizadas foram definidas por meio de Histórias de Usuário (HUs), como cadastro de veículos, listagem com filtros, edição e exclusão. O desenvolvimento foi dividido em duas fases principais: projetos de interface (*UI/UX*), realizado no Figma, e implementação técnica. O mínimo produto viável (*MVP*) incluiu o painel administrativo, a vitrine on-line e operações básicas de cadastro, consulta, edição e exclusão (*CRUD*) de usuários e veículos. Também foi implementado um sistema de filtros que permitiu maior praticidade na busca pelo catálogo.

A validação contínua do sistema foi feita pelo *Scrum Master* ao fim de cada *Sprint* e, posteriormente, apresentada aos clientes pelo *Product Owner* comparando as entregas com os critérios de aceitação definidos. Para facilitar essa tarefa, o sistema foi implantado em ambiente de testes, para que o cliente pudesse acessar remotamente[2]. Essa avaliação garantiu que os requisitos mínimos fossem atendidos e permitiu pequenos ajustes antes da apresentação final.

Os principais desafios enfrentados foram a gestão de tempo e a estimativa de esforço para cada tarefa. Houve também dificuldades na questão do cliente, onde um dos clientes iniciais a qual tinha interesse no sistema, acabou desistindo no início do processo. Como a equipe já estava com várias histórias de usuário e protótipos definidos, optou-se por buscar outra empresa do mesmo segmento, evitando mudanças no escopo. A transição foi rápida — em cerca de 15 dias já haviam sido discutidos com o novo cliente os pontos de alteração e os itens adicionais desejados. Isso permitiu seguir com o projeto sem prejuízos significativos ao cronograma.

[1] https://github.com/CS-PI-2025-TekoBit

[2] https://sandbox.apollo.tekobit.com.br/home

O projeto resultou em um sistema funcional, atendendo ao escopo estabelecido e possibilitando a simulação de um processo real de desenvolvimento. Além do aprendizado técnico em *React*, *Spring Boot* e *DevOps*, a equipe aprimorou práticas de organização, trabalho em equipe, resolução de problemas, comunicação e documentação.

Outro resultado relevante obtido pela equipe foi o registro detalhado do histórico de desenvolvimento nos repositórios GitHub. No repositório do *back-end*, foram criadas 15 *branches*, que representam ramificações do código utilizadas para o desenvolvimento paralelo de funcionalidades, e realizados aproximadamente 75 *commits*, correspondentes aos registros de alterações no código. Já no repositório do *front-end*, foram criadas 34 *branches* e contabilizados cerca de 150 *commits*.

Considerando que a equipe era composta por quatro desenvolvedores, obteve-se uma média aproximada de 12 *branches* e 56 *commits* por integrante. Essa diferença entre os repositórios reflete o escopo das atividades: enquanto o *back-end* concentrou um número menor de histórias de usuário, o *front-end* demandou mais iterações de código devido ao maior volume de ajustes visuais e integrações com a interface.

Como limitações, destacam-se a ausência de integração com meios de pagamento e a não implementação de módulos analíticos. Para trabalhos futuros, sugere-se a integração com plataformas de venda, como OLX e Web Motors; a implementação de agendamento de *test-drives*; e, a criação de relatórios gerenciais para apoio à tomada de decisão. Essas melhorias podem ampliar o valor estratégico do sistema, aproximando-o ainda mais de um ambiente de aplicação comercial.

Referências

Anderson, D. J. (2015). *Kanban: sucesso evolutivo com o método ágil*. Casa do Código, São Paulo.

Figma (2025). Figma — a ferramenta de design de interface colaborativa. `https://www.figma.com`. Acessado em: 24 de set. de 2025.

Kotonya, G. and Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Chichester.

Oracle (2025). Mysql. `https://www.mysql.com`. Acessado em: 24 de set. de 2025.

Peter, A. and Beer, B. (2014). *Introducing GitHub*. O'Reilly Media, Sebastopol, CA.

Pressman, R. S. and Maxim, B. R. (2016). *Engenharia de Software: Uma Abordagem Profissional*. AMGH, 8 edition.

Schwaber, K. and Sutherland, J. (2020). *O Guia do Scrum: O guia definitivo para o Scrum: As regras do jogo*. Scrum.org.

Sommerville, I. (2019). *Engenharia de Software*. Pearson Education do Brasil, 10 edition.

Wiegers, K. and Beatty, J. (2013). *Software Requirements*. Microsoft Press, Redmond, WA, 3 edition.