UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MARCELO MARCOS VICHAR JUNIOR

NOVA VERSÃO DO LETSTUR - SISTEMA DE AGÊNCIAS DE PASSEIO TURÍSTICO

GUARAPUAVA

2024

MARCELO MARCOS VICHAR JUNIOR

NOVA VERSÃO DO LETSTUR - SISTEMA DE AGÊNCIAS DE PASSEIO TURÍSTICO

New version of LetsTur - Tourism tour agencies system

Monografia de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, Campus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet.

Orientador: Prof. Dr. Paulo Henrique Soares

GUARAPUAVA

2024

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

MARCELO MARCOS VICHAR JUNIOR

NOVA VERSÃO DO LETSTUR - SISTEMA DE AGÊNCIAS DE PASSEIO TURÍSTICO

Monografia de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, Campus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet.

Data de aprovação: 08/maio/2024

Prof. Paulo Henrique Soares
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Luciano Ogiboski
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Renata Luiza Stange Carneiro Gomes
Doutora
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

GUARAPUAVA

2024

Dedico este trabalho à minha esposa Danielle.

AGRADECIMENTOS

Agradeço a todos os que contribuíram para a realização deste trabalho. Em especial, quero expressar minha imensa gratidão à minha esposa Danielle, cujo apoio incondicional e incentivo foram fundamentais em cada etapa deste percurso. Sua presença constante e seu encorajamento incansável foram a luz que iluminou os momentos mais desafiadores da minha formação.

À minha família, gostaria de estender meu profundo agradecimento. Seu apoio e amor foram pilares essenciais durante toda a jornada acadêmica. Cada palavra de encorajamento, cada gesto de confiança, fortaleceu minha determinação e inspirou meu compromisso em alcançar meus objetivos.

Não posso deixar de mencionar meu orientador, Prof. Dr. Paulo Henrique Soares, cuja orientação sábia e apoio constante foram fundamentais para a conclusão deste trabalho. Seu conhecimento, paciência e dedicação foram verdadeiramente inspiradores, moldando não apenas este projeto, mas também meu desenvolvimento acadêmico e profissional como um todo.

Expresso também minha gratidão a todos os professores da UTFPR, cuja expertise e dedicação foram essenciais para minha formação. Cada aula, cada orientação, contribuiu significativamente para meu crescimento intelectual e pessoal. Sou imensamente grato por ter tido a oportunidade de aprender com profissionais tão capacitados e dedicados.

Também agradeço à própria UTFPR, pela excelência acadêmica e pelas oportunidades de aprendizado e desenvolvimento que oferece. Esta instituição não apenas me proporcionou uma educação de qualidade, mas também me inspirou a buscar constantemente a excelência em tudo o que faço.

Por fim, gostaria de expressar minha sincera gratidão ao Sr. Guilherme Mazur e à empresa Lets. Durante minha formação acadêmica, tive a oportunidade de estagiar e trabalhar profissionalmente neste ambiente estimulante e desafiador.

A todos que de alguma forma contribuíram para esta conquista, meu mais sincero obrigado. Este trabalho não seria possível sem o apoio, incentivo e orientação de cada um de vocês.

A persistência é o caminho do êxito.
(Charles Chaplin)

RESUMO

Este trabalho concentra-se na implementação de uma nova versão do Sistema Letstur, uma plataforma utilizada por agências de passeios turísticos no Brasil. Devido ao avanço das tecnologias e conceitos ao decorrer dos anos, atualmente o Letstur poderia ser muito beneficiado com melhorias referentes à tecnologias utilizadas e eficiência operacional. O foco principal deste trabalho, é desenvolver uma solução que se baseie na modernização tecnológica, automação de processos operacionais e melhoria da experiência do usuário. Isso incluirá a atualização da infraestrutura tecnológica, a simplificação de fluxos de trabalho e a criação de uma interface mais fluida e responsiva. O trabalho abrange desde a análise inicial do sistema Letstur até a implementação prática das melhorias propostas. Será medido o impacto das mudanças implementadas, monitorando a eficiência operacional, e a melhora na interface da nova versão do Letstur.

Palavras-chave: letstur.

ABSTRACT

This work focuses on the implementation of a new version of the Letstur System, a platform used by tourist tour agencies in Brazil. Due to the advancement of technologies and concepts over the years, Letstur could currently benefit greatly from improvements regarding the technologies used and operational efficiency. The main focus of this work is to develop a solution that is based on technological modernization, automation of operational processes and improvement of the user experience. This will include updating technological infrastructure, simplifying workflows and creating a more fluid and responsive interface. The work ranges from the initial analysis of the Letstur system to the practical implementation of the proposed improvements. The impact of the implemented changes will be measured, monitoring operational efficiency, and the improvement in the interface of the new version of Letstur.

Keywords: letstur.

LISTA DE FIGURAS

Figura 1 – Tela inicial da aplicação atual **15**

Figura 2 – Tela de vendas da aplicação atual **16**

Figura 3 – Ilustração de quadro kanban **23**

Figura 4 – Fluxo gitflow **23**

Figura 5 – Fluxo Http **24**

Figura 6 – Requisição HTTP **25**

Figura 7 – Resposta Http **26**

Figura 8 – Arquivo principal de configuração Docker Compose **29**

Figura 9 – Arquivo de configuração do ambiente de desenvolvimento **30**

Figura 10 – Arquivo de configuração do ambiente de produção **31**

Figura 11 – Arquivo de *workflow* para formatação de código na *Application Programming Interface* - Interface de Programação de Aplicação (API) . . . **33**

Figura 12 – Console de execução de *workflow* para formatação de código na API . **34**

Figura 13 – Domínios da API **35**

Figura 14 – Arquivo de modelo de empresa **36**

Figura 15 – Arquivo de migração de empresa **37**

Figura 16 – Fluxograma - Requisição na API **39**

Figura 17 – Configurações de permissionamento **40**

Figura 18 – Site oficial do Letstur **42**

Figura 19 – Design System **43**

Figura 20 – Exemplo de requisição no Insomnia **44**

Figura 21 – Tela de cadastro de vendas. **45**

Figura 22 – Falha na responsividade na tela de vendas. **46**

Figura 23 – Modelagem de empresas e usuários no banco de dados **48**

Figura 24 – Modelagem da área de vendas no banco de dados **49**

Figura 25 – Modelagem da área de eventos no banco de dados **50**

Figura 26 – Modelagem da área de passeios no banco de dados **51**

Figura 27 – Tela de login **52**

Figura 28 – Tela de dashboard **53**

Figura 29 – Tela de listagem de usuários **54**

Figura 30 – Tela de listagem de empresas **55**
Figura 31 – Tela de listagem de eventos **56**
Figura 32 – Tela de formulário de eventos **57**
Figura 33 – Tela de calendário **58**
Figura 34 – Tela de listagem de vendas **59**
Figura 35 – Etapa 1 do formulário de vendas **60**

LISTA DE ABREVIATURAS E SIGLAS

Siglas

API	*Application Programming Interface* - Interface de Programação de Aplicação
AWS	*Amazon Web Services*
CD	*Continuous Deployment* - Entrega contínua
CI	*Continuous Integration* - Integração contínua
CSS	*Cascading Style Sheets* - Folhas de Estilo em Cascatas
DDD	*Domain driven Design* - Design Direcionado ao Domínio
HTML	*HyperText Markup Language* - Linguagem de Marcação de Hipertexto
HTTP	*Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto
ORM	*Object-Relational Mapping* - Mapeamento objeto-relacional
PIB	Produto Interno Bruto
SaaS	*Software as a Service* - Software como Serviço
SGBD	Sistema de gerenciamento de banco de dados
SQL	*Structured Query Language* - Linguagem de consulta estruturada

SUMÁRIO

1 INTRODUÇÃO 12

1.1 Considerações iniciais 12

1.2 Objetivos 13

1.2.1 Objetivo geral 13

1.2.2 Objetivos específicos 13

1.3 Justificativa 13

1.4 Estrutura do trabalho 14

2 LETSTUR 15

3 MATERIAIS E MÉTODOS 17

3.1 Materiais 17

3.1.1 Protótipo 17

3.1.2 API 17

3.1.3 ORM 18

3.1.4 Banco de dados 18

3.1.5 Interface 19

3.1.6 Ferramenta para testes 19

3.1.7 Versionamento 20

3.1.8 Infraestrutura 20

3.1.9 Ferramenta para kanban 20

3.2 Métodos 21

3.2.1 Avaliação da atual versão do sistema 21

3.2.2 Análise e levantamento de requisitos 21

3.2.3 Estruturação do sistema 22

3.2.4 Organização de tarefas 22

3.2.5 Fluxo de versionamento 22

3.2.6 Codificação 23

3.2.7 Testes 25

3.2.8 Documentação 26

4 DESENVOLVIMENTO 28

4.1 Configuração de ambiente 28

4.1.1 Configuração de ambientes 28
4.1.2 Integração CI/CD 31
4.2 Estruturação da API 34
4.2.1 Organização por domínio através de DDD 34
4.2.2 Modelos e migrações do Laravel 35
4.2.3 Fluxo de uma requisição 37
4.2.4 Permissionamento 38
4.3 Estruturação dos testes 41
4.4 Estruturação da interface 41
4.4.1 Design System 42
4.4.2 Estrutura de pastas da aplicação de interface 43
4.5 Estruturação da documentação 44
5 RESULTADOS 45
5.1 Avaliação da atual versão do sistema 45
5.2 Análise e levantamento de requisitos 46
5.3 Modelagem do banco de dados 47
5.3.1 Usuários e empresas 48
5.3.2 Vendas de eventos 49
5.3.3 Cadastro de eventos 49
5.3.4 Cadastro de passeios 49
5.4 Interface 51
6 CONSIDERAÇÕES FINAIS 61
6.1 Conclusões 61
6.2 Trabalhos futuros 61
REFERÊNCIAS 63

1 INTRODUÇÃO

A seção de Introdução é dedicada a fornecer uma visão geral do trabalho, explicando o contexto, relevância e objetivos que serão abordados ao longo do seu desenvolvimento.

1.1 Considerações iniciais

A indústria do turismo desempenha um importante papel na economia brasileira, correspondendo a 8,1% do Produto Interno Bruto (PIB) nacional, considerando que o Brasil é conhecido internacionalmente por suas belezas naturais e sua costa. No entanto, apesar do potencial, esse mercado ainda não foi totalmente aproveitado. O país não ocupa uma posição de destaque significativo no cenário turístico global, registrando um fluxo de turistas que representa apenas entre 1% e 2% do total mundial (SEBRAE, 2022).

Um dos principais fatores que impedem que o Brasil se torne uma potência turística, é a falta de qualidade nos serviços prestados por determinadas empresas do setor, que muitas vezes oferecem serviços manuais ou com ferramentas ultrapassadas (BBC, 2022). As agências de passeios turísticos, também conhecidas como operadores turísticos, desempenham um papel fundamental na criação de boas experiências para os viajantes, que esperam agilidade e segurança quando fazem aquisição de um determinado serviço da empresa. No entanto, à medida que o setor evolui rapidamente, muitas dessas agências ainda enfrentam desafios significativos relacionados à eficiência operacional, satisfação do cliente e evolução tecnológica.

O sistema Letstur (2014), no mercado desde 2014, foi criado com o intuito de suprir as necessidades dessas agências. A plataforma oferece funcionalidades que agilizam o trabalho de operadores de passeios turísticos. Entre essas funcionalidades, destacam-se a emissão de *vouchers*, a efetivação e histórico de vendas, a gestão de cadastros de passeios, veículos, guias entre outras funcionalidades. O Letstur atua como um agente por trás das operações diárias, auxiliando as agências, com intuito de fornecer serviços de qualidade aos seus clientes, enquanto mantém um registro organizado de suas operações. Atualmente, o sistema está consolidado e conta com diversos clientes com foco na região Nordeste do Brasil, especialmente na cidade de Recife, e tem como planejamento aumentar sua lista de agências operantes nos próximos anos.

No entanto, apesar da sua estabilidade, e do seu papel importante nas agências de passeios turísticos, o Letstur pode vir a enfrentar desafios significativos em virtude da sua longa trajetória e do rápido avanço tecnológico que ocorreu nos últimos anos. Estes desafios podem ser resumidos em algumas áreas-chave, como a utilização de tecnologias de menor performance e eficiência, e a operação da plataforma em dispositivos móveis. Diante deste cenário, torna-se evidente a necessidade de uma reestruturação do sistema. É nesse contexto que, neste trabalho, propôs-se a implementação de uma nova versão do sistema, bem como o desenvolvimento de novas interações e funcionalidades.

Além da modernização tecnológica, também foi considerada a experiência dos operadores de vendas, usuários mais assíduos do sistema. A nova versão proposta do Letstur, introduz uma interface mais amigável e responsiva, simplificando o processo de vendas de passeios, e melhorando a experiência do usuário que utiliza essa funcionalidade frequentemente.

1.2 Objetivos

Nessa seção, são mencionados os objetivos principais desse trabalho.

1.2.1 Objetivo geral

O objetivo principal deste trabalho é desenvolver uma versão mais moderna do sistema Letstur, tornando-o uma ferramenta mais eficiente e centrada no usuário, a fim de otimizar os serviços prestados para as agências de passeios turísticos.

1.2.2 Objetivos específicos

- Através de análise na atual versão do sistema, e levantamento de requisitos, propor uma aplicação mais eficiente e centrada no usuário.
- Desenvolver uma aplicação de manutenção facilitada e com cobertura de testes, a fim de facilitar alterações futuras.
- Entregar uma solução responsiva e eficaz, tanto em computadores, quanto em dispositivos móveis.

1.3 Justificativa

Embora tenha atendido inicialmente às necessidades, é importante notar que a evolução da tecnologia e o surgimento de novas soluções na área de atuação do sistema Letstur destacaram a necessidade de modernização. As melhorias planejadas representam uma oportunidade significativa para aprimorar a eficiência e a qualidade dos serviços prestados.

A nova versão do sistema aborda os desafios previamente identificados, a fim de possibilitar uma infraestrutura mais ágil e de fácil manutenção. Isso resulta em respostas mais rápidas na resolução de problemas e em uma maior simplicidade na implementação de melhorias, contribuindo para uma experiência mais satisfatória das agências de passeios turísticos. A eliminação da redundância de dados proporciona informações mais precisas e confiáveis, fortalecendo a credibilidade dos serviços oferecidos.

Uma interface mais fluida e responsiva para dispositivos móveis foi uma das melhorias propostas pela nova versão, permitindo um melhor aproveitamento do painel administrativo do sistema.

Portanto, a implementação da nova versão é justificada com base nas melhorias que ela tende a trazer para as operações do sistema. Ao abordar esses desafios, este trabalho entrega uma solução mais eficiente, que atende às necessidades das agências de forma mais eficaz, enquanto enquanto busca facilitar expansões e manutenções futuras.

1.4 Estrutura do trabalho

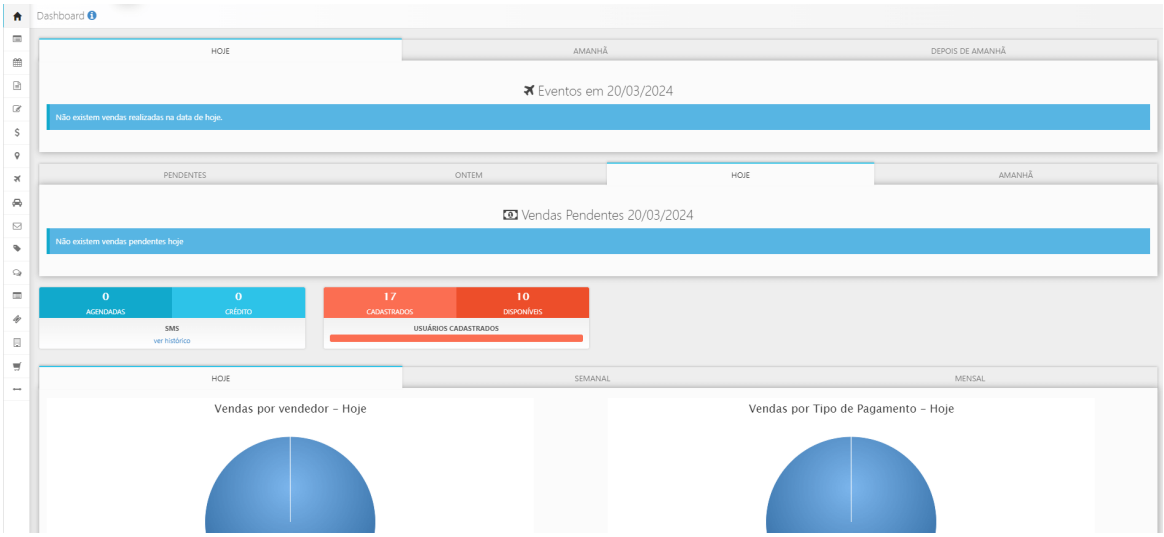
Ao longo deste trabalho, serão exploradas as etapas da modernização, os benefícios esperados e as estratégias para consolidar o Letstur como uma ferramenta indispensável para agências de passeios turísticos. No Capítulo 2, será abordado o funcionamento atual do sistema Letstur, e seus principais desafios encontrados atualmente. Os métodos e materiais que foram utilizados para a construção da solução, estão descritos no Capítulo 3. As decisões tomadas durante o desenvolvimento do trabalho, estão descritas no Capítulo 4. Os resultados obtidos na implementação do trabalho, estão dispostos no Capítulo 5. No Capítulo 6, estão as considerações finais deste trabalho.

2 LETSTUR

O Letstur (2014) é uma aplicação web voltada para a administração de agências de passeios turísticos, na área de turismo receptivo. O sistema foi desenvolvido e começou a operar em 2014, com o intuito de atender as demandas específicas de uma única agência de passeios turísticos, que na época, era administrada pelos mesmos responsáveis pelo sistema Letstur. Após o sucesso inicial da aplicação, sua versatilidade e eficiência levaram à decisão de disponibilizar unidades do sistema como *Software as a Service* - Software como Serviço (SaaS) para diversas agências. Essa mudança estratégica resultou na multiplicação do Letstur por importantes localidades turísticas brasileiras, como Recife e Fernando de Noronha, solidificando sua presença e impacto no setor de turismo nacional.

Durante sua jornada desde a concepção até os dias atuais, o Letstur demonstrou um crescimento comercial. Inicialmente desenvolvido para atender a uma agência específica, expandiu-se e já atendeu cerca de 30 empresas do ramo turístico. Ao longo desse período, o sistema foi responsável por facilitar o processo de compra de eventos para aproximadamente 700 mil clientes finais, evidenciando seu impacto positivo no mercado e sua importância para as agências de passeio turístico (LETSTUR, 2014). A Figura 1 exibe a tela inicial da aplicação, com as principais informações para os administradores das agências.

Figura 1 – Tela inicial da aplicação atual

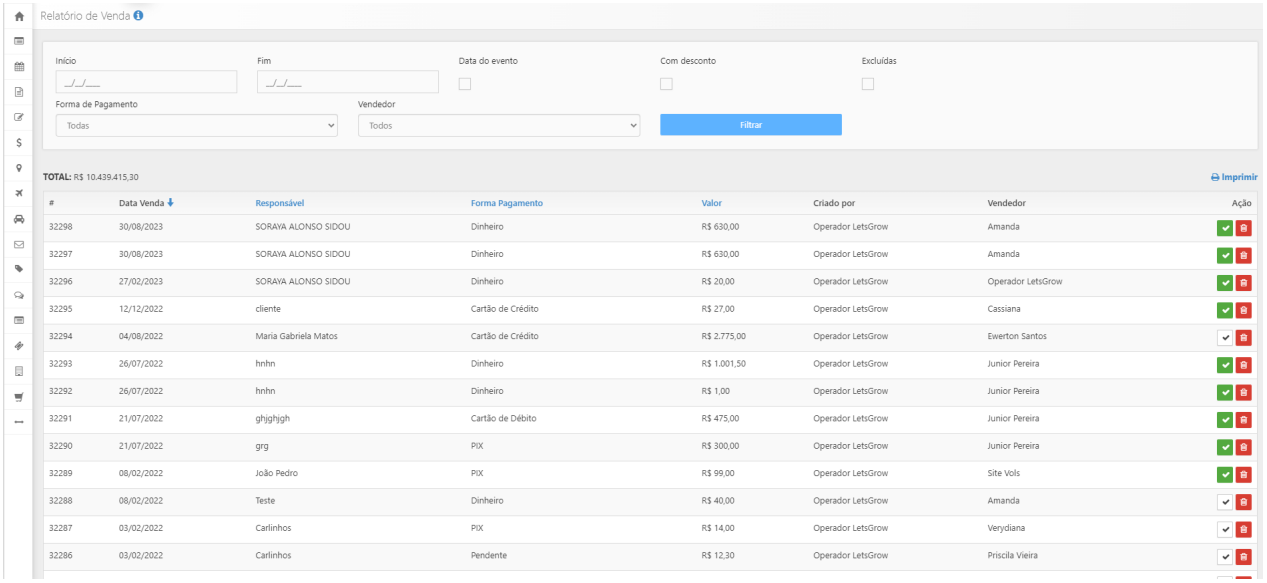


Fonte: Letstur (2014).

O Letstur oferece um conjunto de funcionalidades projetadas para atender às necessidades das agências de passeio turístico. Entre suas principais características está a gestão de vendas de eventos, permitindo que as agências organizem e comercializem passeios e atividades de forma eficiente. Além disso, o sistema oferece a emissão de *vouchers* personalizados, facilitando o processo de reserva para os clientes finais. Com calendários de eventos integrados, as agências podem visualizar e coordenar itinerários e disponibilidade de forma intuitiva. Além disso, o Letstur oferece controle de recursos, incluindo gestão de motoristas, veículos, guias

de passeios, e outros itens relacionados à área turística. A Figura 2 exibe a tela de vendas da aplicação, a principal área do sistema.

Figura 2 – Tela de vendas da aplicação atual



Fonte: Letstur (2014).

No entanto, à medida que o tempo passa, o Letstur se depara com novos desafios significativos. Atualmente, o sistema enfrenta algumas limitações tecnológicas que podem potencialmente prejudicar seu crescimento e abrangência no mercado de operadores de passeios turísticos. As tecnologias utilizadas em sua criação agora estão obsoletas, tornando-se uma tarefa árdua manter o sistema atualizado e seguro. Além disso, a falta de responsividade para dispositivos modernos limita sua utilidade em um mercado cada vez mais móvel. Outro ponto crítico que requer otimização é a maneira como os cadastros são realizados na aplicação, com muitos campos desnecessários que precisam ser preenchidos, resultando em uma lentidão operacional evitável.

Um desafio adicional é a estrutura atual do sistema, onde cada instância da aplicação é operada individualmente. Isso não apenas dificulta a manutenção eficiente, mas também atrasa qualquer atualização ou correção que seja necessária. Uma simples modificação, exigiria intervenção em todas as unidades ativas do sistema, resultando em uma complexidade desnecessária e de lenta resolução para as agências clientes. Essa fragmentação no ambiente do Letstur representa um obstáculo significativo para sua escalabilidade e agilidade operacional no futuro.

Diante desses pontos de melhoria, torna-se necessário uma revisão e modernização da aplicação. O atual cenário pode ser otimizado para que o sistema não se torne defasado a médio e longo prazo.

3 MATERIAIS E MÉTODOS

Neste capítulo, serão reportados os materiais e métodos utilizados para alcançar o objetivo do trabalho.

3.1 Materiais

Nos tópicos a seguir, estão descritas as principais tecnologias que serão utilizadas para o desenvolvimento do sistema, separadas por áreas do trabalho.

3.1.1 Protótipo

Para a elaboração do *design* das interfaces na nova versão do Letstur, foi optado pela utilização do Figma[1], uma plataforma colaborativa dedicada à construção de layouts e protótipos de design de interfaces (CANALTECH, 2021). Embora existam diversas ferramentas disponíveis para o design de interfaces, como Sketch, Adobe XD e InVision, a escolha do Figma destaca-se por suas características. O Figma é uma solução baseada em nuvem, permitindo colaboração em tempo real entre vários membros, independentemente da localização geográfica. Essa abordagem colaborativa promove a eficiência e facilita a comunicação. A escolha do Figma no contexto do Letstur é embasada em sua acessibilidade, flexibilidade e na capacidade de proporcionar um ambiente de design dinâmico.

3.1.2 API

Na construção da aplicação que rodará no servidor, foi utilizada a linguagem PHP[2], acompanhado do Laravel[3], um *framework* de aplicações *web* com sintaxe expressiva e elegante, segundo sua documentação oficial (LARAVEL, 2024d).

Ao optar pelo PHP em detrimento de outras linguagens de programação em alta, como TypeScript, Ruby, Java, entre outras, várias vantagens se destacam. Primeiramente, a ampla adoção e longa história do PHP o tornam uma escolha sólida para o desenvolvimento *web*, com uma vasta comunidade de desenvolvedores e suporte disponível online. Além disso, a simplicidade e flexibilidade do PHP permitem uma curva de aprendizado mais leve, o que é de grande importância para um projeto, que futuramente, será acoplado a empresa, e sofrerá manutenções e melhorias constantes, como o caso do Letstur. Além disso,a gama de *frameworks* e bibliotecas disponíveis facilita o desenvolvimento rápido e escalável de aplicações *web* complexas. Por fim, a performance aprimorada do PHP, especialmente com a introdução de versões mais recentes,

[1] https://www.figma.com/
[2] https://www.php.net/
[3] https://laravel.com/

oferece uma execução mais rápida e eficiente de código, atendendo às demandas de projetos modernos e de alta performance (PHP, 2024).

Ao escolher o Laravel em comparação com outros frameworks PHP, como Symfony ou CodeIgniter, diversas vantagens se evidenciam. Em primeiro lugar, a robustez de recursos do Laravel o destacam como uma boa escolha para o desenvolvimento web. Seu sistema de roteamento, migrações de banco de dados e *Object-Relational Mapping* - Mapeamento objeto-relacional (ORM) integrado simplificam significativamente o processo de desenvolvimento, permitindo que desenvolvedores foquem na lógica de negócios em vez de tarefas repetitivas (LARAVEL, 2024d).

Um outro ponto de suma importância, diz respeito a familiaridade da equipe de desenvolvimento da empresa responsável pelo Letstur, com as ferramentas citadas, considerando que os desenvolvedores tem histórico no desenvolvimento e manutenção de aplicações de diferentes níveis de complexidade, utilizando as presentes ferramentas.

3.1.3 ORM

Para estabelecer a comunicação entre a aplicação e o banco de dados, foi utilizado o Eloquent[4], um ORM já introduzido no Laravel, que é utilizado como uma alternativa à escrita de *Structured Query Language* - Linguagem de consulta estruturada (SQL) ou a utilização de outra ferramenta de acesso ao banco de dados, como construtores de consultas SQL.

Ao utilizar o Eloquent, cada tabela do banco de dados possui um modelo correspondente que é utilizado para interagir com aquela tabela. Além de recuperar registros do banco de dados, os modelos do Eloquent também permitem inserir, atualizar e excluir registros da tabela (LARAVEL, 2024c).

Embora existam diversas opções de ORM disponíveis, a escolha pelo Eloquent é respaldada por suas características, oferecendo uma solução eficiente para a camada de acesso ao banco de dados no ambiente PHP e Laravel.

3.1.4 Banco de dados

Para a gestão eficiente do armazenamento, o sistema adotou um banco de dados relacional, sendo o PostgreSQL[5] a ferramenta designada para o gerenciamento. O PostgreSQL destaca-se como uma escolha sólida devido à sua robustez, desempenho e suporte extensivo a recursos avançados de SQL.

Embora existam várias opções de Sistema de gerenciamento de banco de dados (SGBD) disponíveis, como MySQL, SQLite e Microsoft SQL Server, a seleção do PostgreSQL é fundamentada em suas características distintivas. Este SGBD de código aberto oferece uma

[4] https://laravel.com/docs/11.x/eloquent
[5] https://www.postgresql.org/

arquitetura sólida, capacidade de expansão horizontal, e uma comunidade ativa que contribui para sua constante evolução e aprimoramento (POSTGRES, 2023).

A escolha do PostgreSQL para a nova versão do Letstur é motivada pela sua capacidade comprovada de lidar com cargas de trabalho complexas, transações robustas e suporte eficaz para aplicações que requerem relacionamentos complexos, o que pode ser de suma importância no decorrer do desenvolvimento do sistema.

3.1.5 Interface

Para desenvolver a aplicação que rodará nos navegadores dos usuários, foi utilizada a linguagem TypeScript[6], juntamente da ferramenta VueJs[7]. Além disso, foi utilizado o *framework* Nuxt, responsável por criar aplicações VueJs com mais confiança e praticidade (NUXT, 2023). Para implementar as telas e elementos previamente prototipados, foi utilizada a ferramenta Tailwind, uma estrutura *Cascading Style Sheets* - Folhas de Estilo em Cascatas (CSS) utilitária composta de classes implementadas para construir *design*, diretamente na marcação *HyperText Markup Language* - Linguagem de Marcação de Hipertexto (HTML) (TAILWIND, 2023).

A escolha dessas tecnologias, assim como para a API, foi baseada não apenas em suas características individuais, mas também no conhecimento prévio e na experiência da equipe de desenvolvimento da empresa responsável pelo Letstur. A familiaridade com essas tecnologias permite uma transição mais eficiente para o desenvolvimento da aplicação, garantindo que a equipe possa se concentrar na implementação de funcionalidades e na entrega de valor ao cliente, em vez de gastar tempo em curva de aprendizado. Além disso, a combinação dessas tecnologias, tende a oferecer uma abordagem moderna e escalável para o desenvolvimento da aplicação, garantindo uma melhor experiência de usuário final e um código bem estruturado e de fácil manutenção.

3.1.6 Ferramenta para testes

A biblioteca escolhida para implementação de testes automatizados na nova versão do Letstur é o Laravel Test, uma ferramenta baseada em PHPUnit[8], um dos mais conceituados *frameworks* de testes para PHP. O Laravel Test, destaca-se como ferramenta de testes, pelo fato de já ser integrado ao ecossistema Laravel, o que torna sua configuração mais simples e rápida, possibilitando que os desenvolvedores foquem na escrita dos testes de suas funcionalidades (LARAVEL, 2024e).

A agilidade do Laravel Test na execução de testes é outro fator importante para sua escolha, já que propõe garantir respostas imediatas aos desenvolvedores e acelerando o ciclo

[6] https://www.typescriptlang.org/
[7] https://vuejs.org/
[8] https://phpunit.de/index.html

de desenvolvimento. Em comparação com outras soluções, o Laravel Test se destaca pela sua eficiência e pela capacidade de execução de testes em paralelo.

3.1.7 Versionamento

Para a gestão do versionamento do sistema, foi adotado o Git[9], com repositórios hospedados na plataforma GitHub[10]. O Git é um sistema de controle de versão distribuído, amplamente reconhecido por sua eficiência e flexibilidade, enquanto o GitHub oferece uma plataforma de desenvolvimento colaborativo que hospeda projetos na nuvem.

Apesar da existência de alternativas como Bitbucket e GitLab, a escolha do Git em conjunto com o GitHub é motivada por suas particularidades. O Git, por ser descentralizado, permite um histórico de versionamento eficiente e a facilidade de trabalhar em ambientes distribuídos. O GitHub, além de servir como repositório remoto, proporciona ferramentas adicionais para colaboração, como rastreamento de problemas (issues), integração contínua e revisões de código (GITHUB, 2023).

3.1.8 Infraestrutura

Para simplificar as configurações de ambientes da aplicação, foi utilizado o Docker[11], uma plataforma *open source* que facilita a criação e administração de ambientes isolados, além do Docker Compose[12], uma ferramenta usada para definir e executar aplicativos de vários contêineres do Docker (DOCKERDOCS, 2023).

Embora existam outras opções para a virtualização de ambientes, como Vagrant e Kubernetes, a escolha do Docker é fundamentada em suas características específicas. O Docker proporciona a criação de contêineres leves e portáteis, garantindo consistência nos ambientes de desenvolvimento, teste e produção. O Docker Compose, por sua vez, simplifica a orquestração de múltiplos contêineres, possibilitando uma configuração eficiente e fácil de reproduzir.

3.1.9 Ferramenta para kanban

Na busca por uma solução eficaz de *Kanban*, foi optado pela plataforma Jira, pois destaca-se como uma escolha estratégica. Ao eleger o Jira para a gestão *Kanban*, destacam-se vantagens não tão desenvolvidas em outras soluções, como Trello e Asana. O Jira se destaca pela sua robusta integração, permitindo uma visão abrangente do fluxo de trabalho, além de

[9] https://git-scm.com/
[10] https://github.com/
[11] https://www.docker.com/
[12] https://docs.docker.com/compose/

apresentar alta flexibilidade, um rastreamento de tarefas satisfatório para o uso na nova versão do Letstur, e uma interface intuitiva (APPMASTER, 2022).

3.2 Métodos

Nesta seção, serão apresentados as metodologias utilizadas para conduzir o desenvolvimento do sistema proposto. As atividades abrangem desde a identificação do problema até a implantação da solução.

3.2.1 Avaliação da atual versão do sistema

O primeiro objetivo do trabalho foi identificar os principais pontos de melhorias do sistema atual. Para isso, foram realizados testes funcionais e uma análise na atual versão do sistema, buscando um melhor entendimento das melhorias a serem implementadas na nova versão do Letstur.

A análise detalhada da versão atual do sistema, incluindo a identificação de fluxos de trabalho e pontos de melhoria, desempenha um papel fundamental no processo de desenvolvimento da nova aplicação. Essa avaliação permite a compreensão das necessidades e desafios enfrentados pelo sistema existente, além de identificar áreas onde as funcionalidades podem ser aprimoradas para atender melhor aos requisitos dos usuários e às demandas do mercado. Ao analisar os fluxos de trabalho atuais, é possível identificar gargalos, redundâncias e ineficiências que podem ser eliminados ou otimizados na nova aplicação. Além disso, a análise do sistema atual pode fornecer informações sobre as expectativas dos usuários finais e os padrões de uso, orientando o design e o desenvolvimento da nova aplicação para garantir uma experiência de usuário aprimorada. Em resumo, a análise do sistema atual é crucial para garantir que a nova aplicação atenda às necessidades específicas do negócio, aumentando a eficiência operacional.

3.2.2 Análise e levantamento de requisitos

Nesta etapa, foi utilizada a metodologia de levantamento de requisitos. Trata-se do processo de compreensão e identificação das necessidades que devem ser solucionadas pelo sistema que será desenvolvido, definindo a função que o software vai desempenhar. Os requisitos são divididos entre funcionais, que tratam das funcionalidades que o sistema deve ter, e não funcionais, que são recursos não considerados funcionalidades, mas características do sistema, como restrições, segurança, confiabilidade, velocidade, validações, entre outros (MARQUES, 2018).

3.2.3 Estruturação do sistema

Após a conclusão do levantamento de requisitos do sistema, procede-se à etapa seguinte do ciclo de desenvolvimento, que consiste na modelagem do banco de dados da aplicação. Essa fase é crucial para a estruturação eficiente e organização dos dados que serão manipulados pelo sistema. A definição física de tabelas, relacionamentos e atributos proporciona uma base para o armazenamento e recuperação de informações, a fim de otimizar a eficácia operacional e garantir a integridade dos dados ao longo do ciclo de vida do software.

Simultaneamente à modelagem do banco de dados, foi realizada a prototipação das interfaces gráficas do sistema. Essa etapa visa fornecer uma representação visual do *layout* e fluxo de interação do usuário com o software. A prototipação das telas permite antecipar e refinar a experiência do usuário antes mesmo da implementação completa do sistema, proporcionando a oportunidade de ajustes e melhorias com base no *feedback* inicial. Essa abordagem acelera o desenvolvimento, e contribui para a identificação precoce de possíveis desafios de usabilidade.

3.2.4 Organização de tarefas

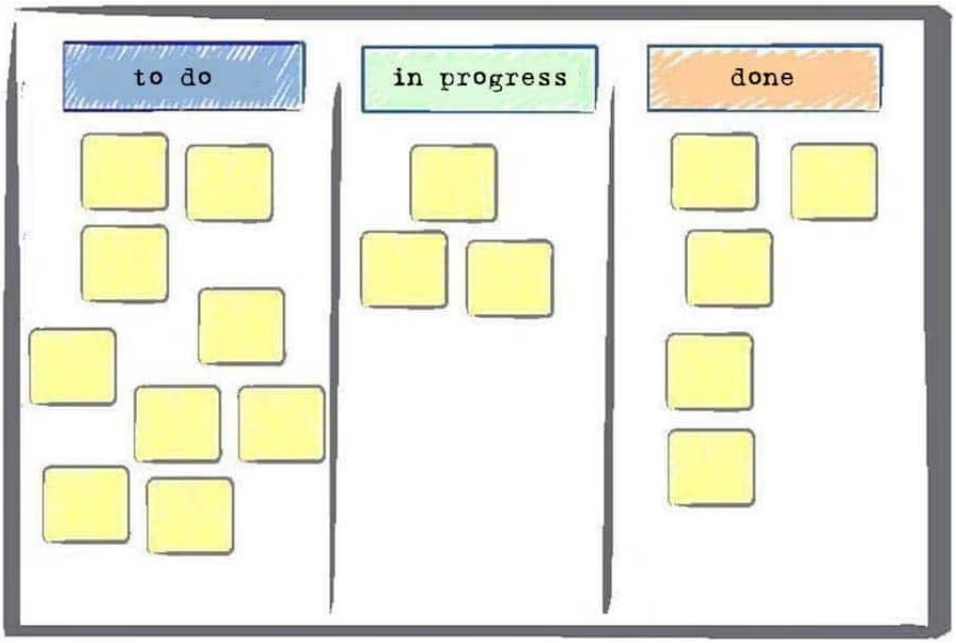
Para controlar a prioridade e andamentos das tarefas nas etapas de desenvolvimento do trabalho, foi utilizado um quadro *Kanban*. Um quadro *Kanban* é uma ferramenta de gerenciamento de projeto ágil que auxilia na visualização de tarefas, limitação de trabalho em andamento e maximização de eficiência. Ele pode ajudar equipes a estabelecer ordem no processo diário. Os quadros *Kanban* usam cartões, colunas e melhorias contínuas para ajudar as equipes de tecnologia e serviço a se comprometerem com a quantidade certa de trabalho e finalizarem suas tarefas no tempo adequado. Um quadro *Kanban* na sua forma mais básica, possui 3 colunas: *To do*, para tarefas que devem ser realizadas, *In progress*, onde estão localizadas as tarefas que estão em andamento, e *Done*, para exibir tarefas que já foram finalizadas (MIRO, 2023).

A Figura 3, ilustra um quadro *kanban* básico, com as colunas *to do*, *in progress* e *done*. Durante o desenvolvimento do sistema, um quadro muito semelhante ao da figura foi utilizado, sem necessidade de acréscimo de colunas adicionais para organização.

3.2.5 Fluxo de versionamento

Para gerenciar as versões do sistema, e garantir que erros sejam corrigidos sem prejudicar outras atividades, foi utilizado o *Gitflow*. O *Gitflow* é um modelo de fluxo de trabalho do *Git*, que define um padrão de ramificação com base nos lançamentos das versões do projeto. O *gitflow* atribui ás *branchs* principais do sistema, funções durante o lançamento do trabalho (ATLASSIAN, 2023).

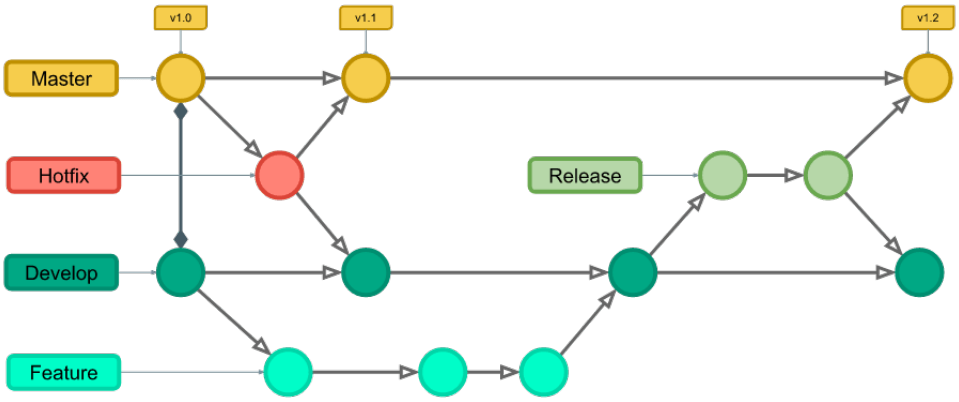
Figura 3 – Ilustração de quadro kanban



Fonte: (T2INFORMATIK, 2023).

A Figura 4, ilustra um fluxo básico que o *gitflow* utiliza para a organização do trabalho. Durante o desenvolvimento da nova versão do Letstur, esse mesmo esquema de branches apresentado na figura foi utilizado, com as tarefas sendo realizadas a partir da *branch develop*, e atribuindo a *branch master*, a responsabilidade de ser o ramo mais estável durante o desenvolvimento e lançamento.

Figura 4 – Fluxo gitflow



Fonte: (ALURA, 2023).

3.2.6 Codificação

O processo de codificação do sistema se divide em duas etapas: a codificação do *backend* e do *frontend*. Essa segmentação permite uma abordagem mais específica e eficiente na

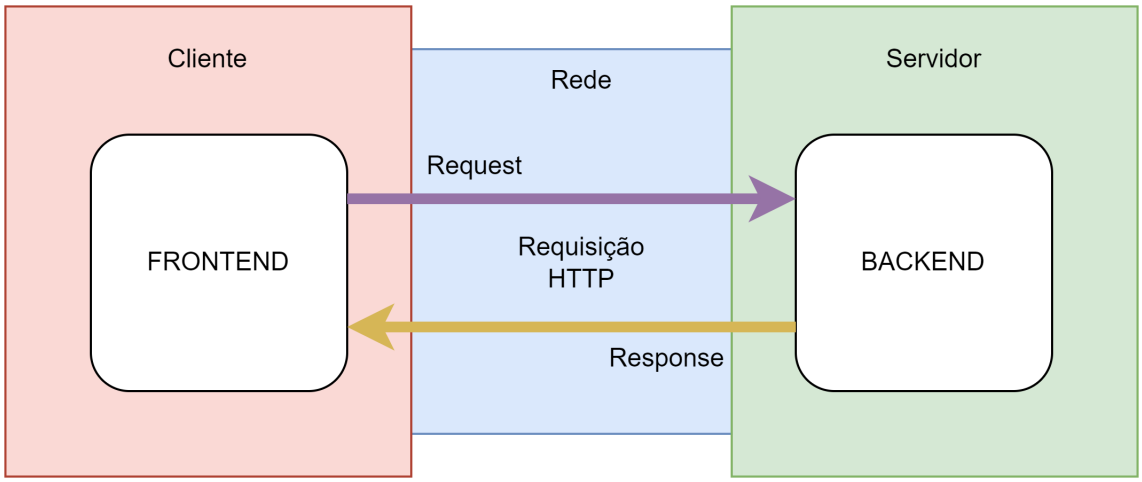
implementação do software, garantindo uma divisão clara de responsabilidades entre o servidor e a interface do usuário.

Na etapa de codificação do *backend*, foi realizada a implementação das funcionalidades essenciais que operam no servidor. Serão aplicados artefatos consolidados de boas práticas de programação, como os princípios SOLID, a fim de criar uma API robusta e flexível. O SOLID, conjunto de princípios da programação orientada a objetos, orienta o desenvolvedor na criação de código mais coeso, escalável e de fácil manutenção. Cada letra do acrônimo SOLID representa um princípio específico (Responsabilidade Única, Aberto/Fechado, Substituição de Liskov, Segregação de Interface e Inversão de Dependência), os quais são aplicados de maneira a assegurar a modularidade e a extensibilidade do código, facilitando a manutenção e evolução do sistema ao longo do tempo (ARAGãO, 2017).

Paralelamente, na codificação do *front-end*, foi implementada a interface do usuário, a fim de garantir uma experiência fluida e intuitiva para o usuário final. A aplicação de conceitos modernos de desenvolvimento *front-end*, como frameworks JavaScript, facilita a criação de interfaces dinâmicas e responsivas. Ademais, a utilização de práticas de codificação consistentes, como a adoção de padrões de projeto e a modularização do código, contribui para a legibilidade e a escalabilidade do sistema.

Além da implementação do software com a divisão clara de responsabilidades entre o lado do servidor e a interface do usuário, é crucial considerar a comunicação entre essas duas partes via protocolo *Hypertext Transfer Protocol* - Protocolo de Transferência de Hipertexto (HTTP). Essa comunicação estabelece a conexão essencial entre o cliente e o servidor, permitindo a troca de dados e solicitações entre eles de maneira eficiente. A Figura 5, ilustra a conexão entre a interface (cliente) e a API do sistema (servidor), e como o protocolo HTTP permite essa conexão.

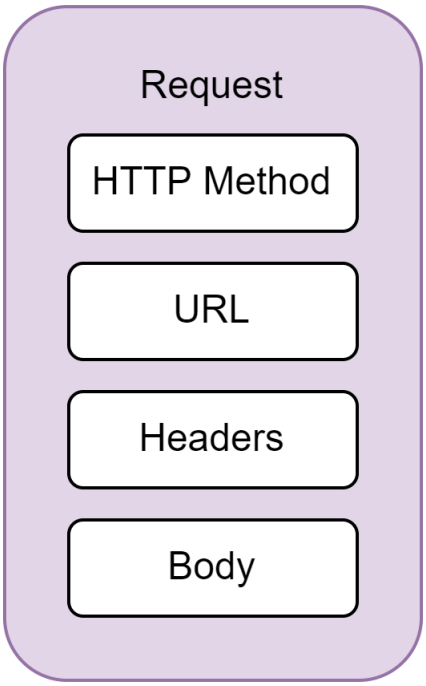
Figura 5 – Fluxo Http



Fonte: (TRAINEECODEPLAYS, 2023).

O protocolo HTTP define regras de comunicação para que dois dispositivos conectados na rede possam trocar dados entre si. Dessa forma, softwares instalados em dispositivos diferentes podem trabalhar em conjunto. O fluxo de uma requisição se inicia com uma chamada do cliente para o servidor. Nela estão contidos um método HTTP, uma URL, *Headers* e um *Request Body*, que são dados extras necessários para o processamento do recurso requisitado, ilustrado na Figura 6. Ao final da requisição, é produzida uma resposta (Response). Nela estão contidos um *Status Code*, *Headers* e um *Response Body*, contendo os dados retornados para o Cliente, assim como na Figura 7 (TRAINEECODEPLAYS, 2023).

Figura 6 – Requisição HTTP



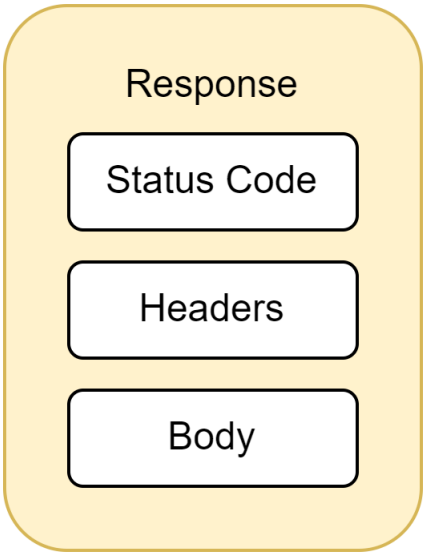
Fonte: (TRAINEECODEPLAYS, 2023).

A aplicação de boas práticas de programação em ambas as camadas do desenvolvimento é fundamental para garantir a coesão global do trabalho. A abordagem orientada a objetos respaldada pelos princípios SOLID não só promove a manutenibilidade e a extensibilidade do código, mas também facilita a colaboração entre desenvolvedores e a compreensão do sistema como um todo, fortalecendo a integração e a eficácia da comunicação entre o frontend e o backend via HTTP.

3.2.7 Testes

A API do Letstur contará com testes automatizados, a fim de manter as funcionalidades do sistema confiáveis e de fácil manutenção futura. Para implementar testes satisfatórios e controlar o custo operacional, o sistema deverá contar com 2 tipos de testes:

Figura 7 – Resposta Http



Fonte: (TRAINEECODEPLAYS, 2023).

- **Testes de unidade**: são os testes realizados na menor parte testável de uma aplicação, independentemente da sua interação com outras partes do código.
- **Testes de integração**: Têm como objetivo testar um conjunto de unidades interagindo entre si. Alguns casos comuns de cobertura de testes de integração são testes realizados na comunicação com o banco de dados, interfaces, micro-serviços, etc.

Na estruturação dos testes para a aplicação, há a implementação de uma quantidade maior de testes unitários em comparação com os testes de integração. Essa abordagem decorre da necessidade de validar o funcionamento individual de cada componente de código de forma isolada. Os testes unitários proporcionam uma verificação detalhada e rápida da lógica de negócios em um nível granular, permitindo uma detecção precoce de falhas e simplificando a manutenção do código ao longo do tempo. Embora os testes de integração desempenhem um papel crucial na verificação da interação entre os diferentes componentes do sistema, geralmente são menos numerosos e mais demorados de serem elaborados e executados, pois muitas vezes precisam estabelecer conexões externas à aplicação, como o banco de dados.

3.2.8 Documentação

Dado que o Letstur se configura como um sistema empresarial, com perspectivas de ser mantido e evoluído por uma equipe de desenvolvedores futuramente, destaca-se como prática essencial no desenvolvimento da sua nova versão a elaboração de uma documentação abrangente.

A escrita de documentação desempenha um papel crucial no processo de desenvolvimento e manutenção do *software*, fornecendo uma fonte centralizada e compreensível de informações. Ao documentar detalhadamente as funcionalidades e as regras de negócio da aplicação, cria-se um recurso valioso que possibilita a disseminação consistente do conhecimento entre os membros da equipe de desenvolvimento, tanto atuais quanto futuros.

A documentação atua como um guia integral, oferecendo uma visão panorâmica do sistema e facilitando a compreensão das camadas funcionais e estruturais da aplicação. Isso não apenas acelera o processo de integração de novos desenvolvedores à equipe, mas também assegura uma boa transição entre diferentes fases do ciclo de vida do software, como manutenções corretivas, evolutivas ou adaptativas.

Além disso, a documentação desempenha um papel vital na estabilidade e na atualização contínua do sistema. Ao manter a documentação alinhada com as mudanças implementadas, garante-se que os desenvolvedores tenham acesso a informações precisas e atualizadas sobre as funcionalidades e o comportamento do sistema. Isso é particularmente crucial em um contexto empresarial, onde a estabilidade e a consistência das operações são imperativas para o sucesso contínuo das atividades comerciais (GUIADEV, 2023).

4 DESENVOLVIMENTO

Neste capítulo serão apresentadas tomadas de decisões relevantes obtidas durante o desenvolvimento do sistema.

4.1 Configuração de ambiente

Nesta seção, estão descritas algumas configurações relevantes para os diversos ambientes de uso da aplicação.

4.1.1 Configuração de ambientes

No desenvolvimento da API, foi decidido pelo uso de Docker e Docker Compose para facilitar a configuração dos ambientes de trabalho. O Docker permite criar contêineres que armazenam todos os componentes que o sistema necessita para funcionar, como a própria aplicação, o banco de dados e ferramentas adicionais, como monitoração de disparo de *emails* e *jobs*, por exemplo.

Com o Docker Compose, torna-se possível organizar esses contêineres de uma forma dinâmica e simplificada, criando arquivos de configuração separados para diferentes ambientes, como desenvolvimento e produção. Essa abordagem facilita a organização das aplicações e garante que cada ambiente tenha suas próprias configurações específicas. A Figura 8 exibe o arquivo principal de configuração de ambiente, utilizado no projeto.

Figura 8 – Arquivo principal de configuração Docker Compose

```yaml
volumes:
  app-postgres-data:
    driver: local

services:
  app:
    container_name: ${APP_NAME}-app
    image: webdevops/php-nginx:8.2-alpine
    volumes:
      - ./project:/app
    working_dir: /app
    ports:
      - ${APP_PORT}:80
    environment:
      - WEB_DOCUMENT_ROOT=/app/public
      - PHP_MEMORY_LIMIT=${APP_MEMORY_LIMIT}
      - php.opcache.enable_cli=1

  postgres:
    container_name: ${APP_NAME}-postgres
    image: postgres:16-alpine
    volumes:
      - app-postgres-data:/var/lib/postgresql/data
    environment:
      - POSTGRES_DB=${POSTGRES_DB}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    ports:
      - ${POSTGRES_PORT}:5432

  redis:
    container_name: ${APP_NAME}-redis
    image: redis:5.0
    command: --appendonly no
    ports:
      - ${REDIS_PORT}:6379

  worker:
    container_name: ${APP_NAME}-worker
    build: environment/docker/worker/.
    volumes:
      - ./project:/app
    environment:
      - PHP_MEMORY_LIMIT=${WORKER_MEMORY_LIMIT}
```

codesnap.dev

Fonte: Autoria própria (2024).

No ambiente de desenvolvimento, foram configurados volumes para montar o código-fonte local dentro do contêiner da aplicação, e o banco de dados, também local, em um contâiner

com imagem *postgres*, além de utilizar contêineres específicos de um ambiente de desenvolvimento, como banco de dados de testes, e monitoramento de envio de emails. Isso permite fazer alterações no código e ver os resultados imediatamente, sem precisar reiniciar todo o ambiente. Além disso, variáveis de ambiente específicas foram definidas para simular diferentes situações, facilitando o teste e a depuração. A Figura 9 exibe o arquivo de configuração do ambiente de desenvolvimento.

Figura 9 – Arquivo de configuração do ambiente de desenvolvimento

```
services:
  app:
    user: ${APP_UID}:${APP_GID}

  postgres-test:
    image: postgres:16-alpine
    environment:
      - POSTGRES_DB=${POSTGRES_TEST_DB}
      - POSTGRES_USER=${POSTGRES_TEST_USER}
      - POSTGRES_PASSWORD=${POSTGRES_TEST_PASSWORD}
    tmpfs:
      - /var/lib/postgresql/data
    ports:
      - ${POSTGRES_TEST_PORT}:5432

  mailcatcher:
    container_name: ${APP_NAME}-mailcatcher
    image: schickling/mailcatcher
    ports:
      - ${MAILCATCHER_PORT}:1080
      - 1025:1025
```

Fonte: Autoria própria (2024).

Já no ambiente de produção, as configurações foram otimizadas para garantir o melhor desempenho e segurança possíveis, que são pontos essenciais para esse tipo de ambiente. A Figura 10 exibe o arquivo de configuração de ambiente de produção.

A manipulação de múltiplos arquivos *Docker compose*, está prevista na documentação oficial, e pode ser feita através de linhas de comandos docker (DOCKER, 2024). Para facilitar o processo, foi criado um script na pasta raíz do projeto, que possui comandos centralizados para subir e derrubar os ambientes pré-configurados, com ações como *start-development*, *start-prod*, *stop*, entre outros.

Figura 10 – Arquivo de configuração do ambiente de produção



```
services:
  app:
    restart: unless-stopped

  worker:
    restart: unless-stopped

  postgres:
    restart: unless-stopped

  redis:
    restart: unless-stopped
```

Fonte: Autoria própria (2024).

4.1.2 Integração CI/CD

A implementação de *Continuous Integration* - Integração contínua (CI) e *Continuous Deployment* - Entrega contínua (CD) desempenha um papel crucial no ciclo de vida do desenvolvimento do projeto. A adoção dessas práticas garante uma abordagem consistente e automatizada para a integração de código, testes e implantação de novas funcionalidades.

Por meio do CI, cada alteração de código é integrada automaticamente ao repositório principal. Isso permite a detecção precoce de possíveis conflitos de integração e erros, contribuindo para a estabilidade do código base. Além disso, a execução automática de testes unitários e de integração durante o processo de integração contínua ajuda a identificar e corrigir inconsistências de forma eficiente (AWS, 2024). No contexto das aplicações do Letstur, os CI foram utilizados para a execução de testes, e para a formatação automatizada do código, garantindo a conformidade com os padrões estabelecidos.

O CD, por sua vez, automatiza o processo de implantação do software em diferentes ambientes, como desenvolvimento, teste e produção. Com a configuração adequada, as atualizações de código podem ser entregues de forma rápida e confiável aos usuários finais, a fim de garantir uma experiência contínua e sem interrupções (AZURE, 2024). No Letstur, o CD foi empregado para realizar o *deploy* automatizado em ambientes hospedados na *Amazon Web Services* (AWS), especificamente no serviço EC2 para hospedagem da API.

A integração do CI/CD no projeto possibilita diversas melhorias, incluindo:

Rápida detecção de problemas: Qualquer problema introduzido no código é identificado precocemente durante o processo de integração contínua, permitindo correções imediatas.

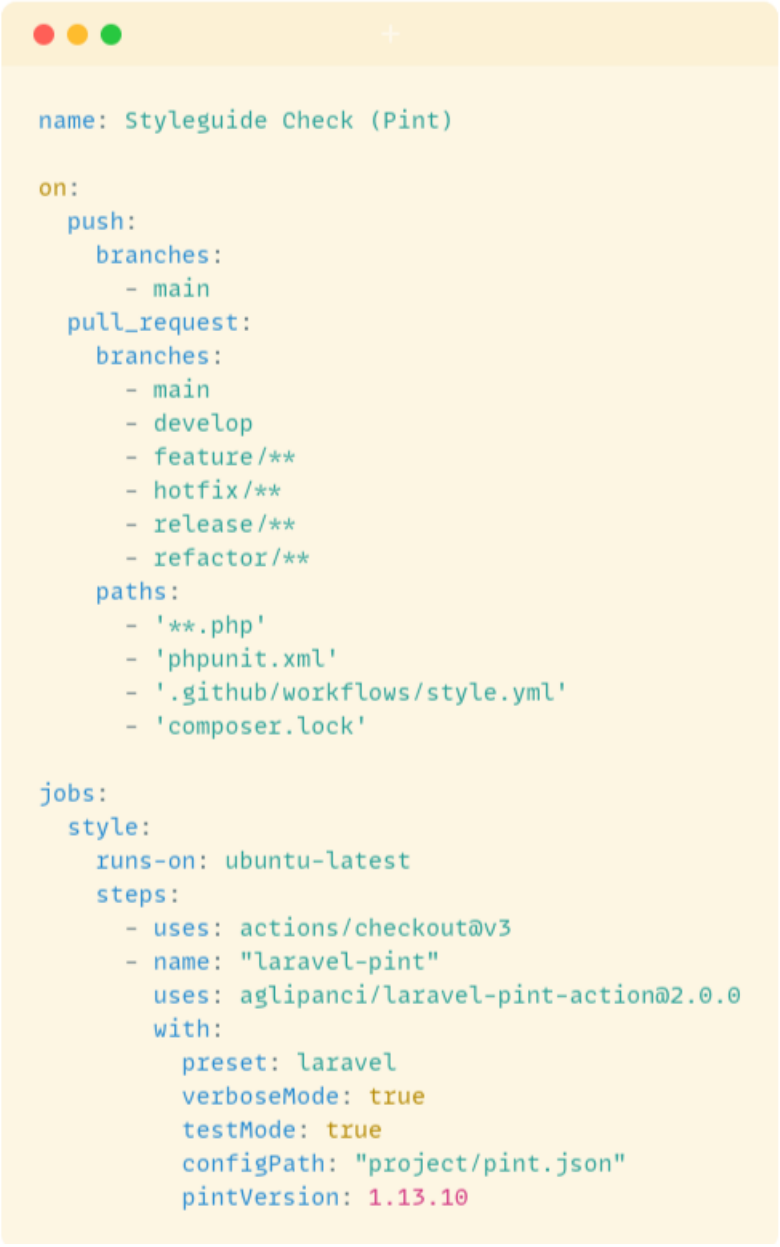
Entrega contínua de valor: Novas funcionalidades são entregues aos usuários finais de forma consistente e confiável, garantindo uma experiência de usuário aprimorada.

Maior confiabilidade: A automação dos processos de teste, formatação e implantação reduz a probabilidade de erros humanos e garante uma maior confiabilidade do software.

Agilidade no desenvolvimento: A automação dos processos de integração e implantação libera os desenvolvedores para se concentrarem em tarefas de maior valor, acelerando o ciclo de desenvolvimento.

Para integração do CI/CD no código da aplicação, foram utilizados os *workflows* do GitHub. A ferramenta proporciona uma maneira eficaz de automatizar e orquestrar os estágios de construção, teste e implantação do software diretamente no ambiente de controle de versão. Esses *workflows* permitem a definição de ações específicas a serem executadas em resposta a eventos no repositório, como *push* de código ou criação de *pull requests* (GITHUB, 2024). A Figura 11 é um exemplo real de implementação de *workflow* no projeto, enquanto a Figura 12, exibe o console de execução desse *workflow* no *GitHub*.

Figura 11 – Arquivo de *workflow* para formatação de código na API



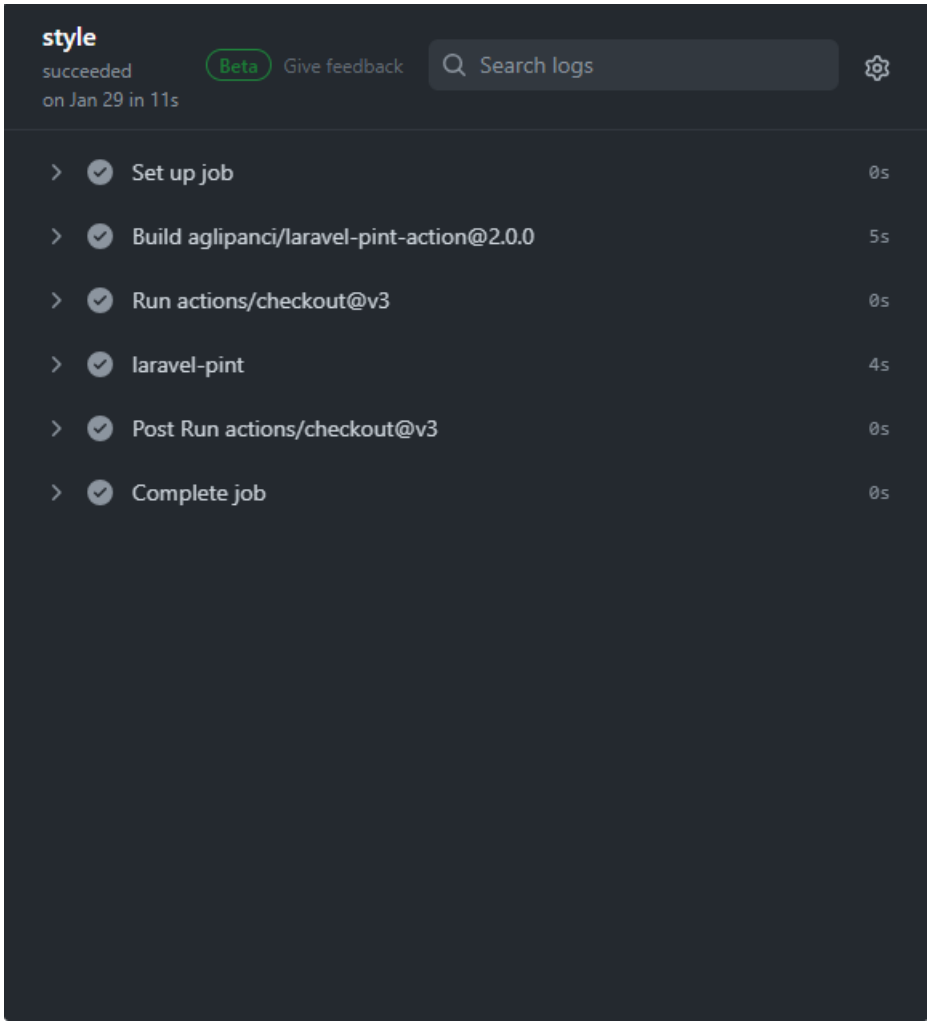
```yaml
name: Styleguide Check (Pint)

on:
  push:
    branches:
      - main
  pull_request:
    branches:
      - main
      - develop
      - feature/**
      - hotfix/**
      - release/**
      - refactor/**
    paths:
      - '**.php'
      - 'phpunit.xml'
      - '.github/workflows/style.yml'
      - 'composer.lock'

jobs:
  style:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: "laravel-pint"
        uses: aglipanci/laravel-pint-action@2.0.0
        with:
          preset: laravel
          verboseMode: true
          testMode: true
          configPath: "project/pint.json"
          pintVersion: 1.13.10
```

Fonte: Autoria própria (2024).

Figura 12 – Console de execução de *workflow* para formatação de código na API



Fonte: Autoria própria (2024).

4.2 Estruturação da API

Nesta seção, estão descritas algumas decisões relevantes para o desenvolvimento da API do sistema.

4.2.1 Organização por domínio através de DDD

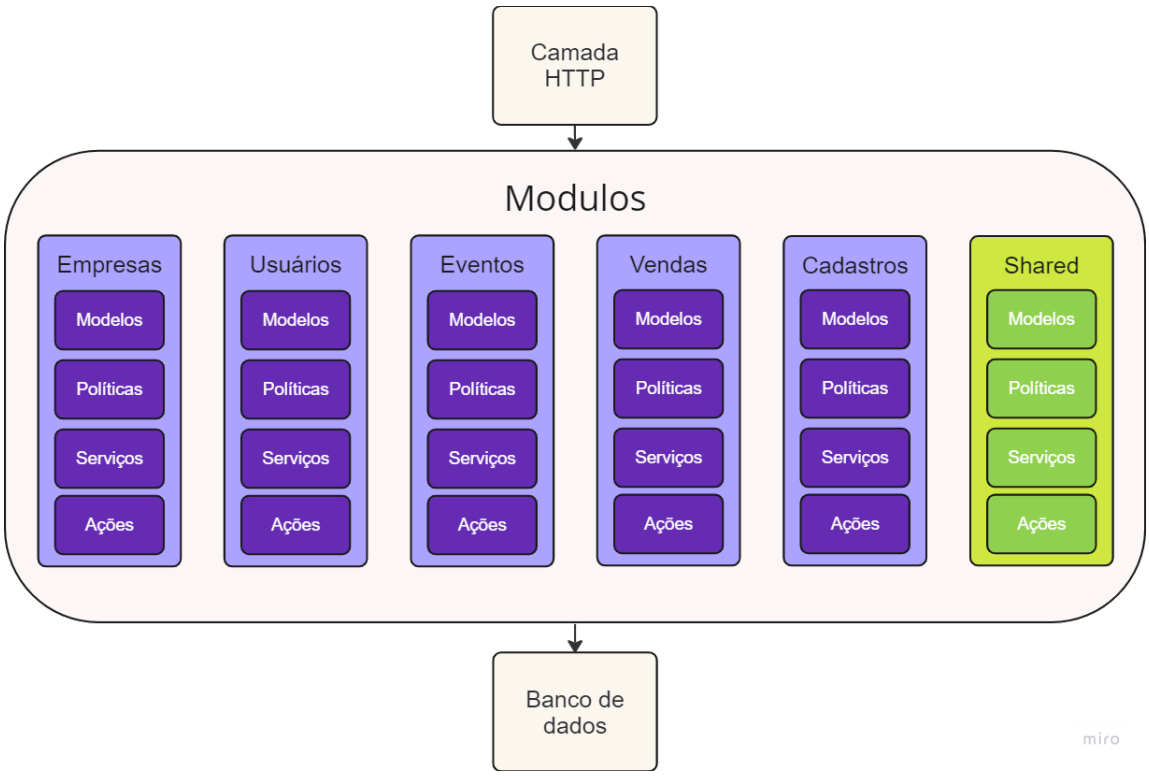
O *Domain driven Design* - Design Direcionado ao Domínio (DDD) é uma abordagem de design de software que coloca o foco principal no domínio do problema, modelando-o de forma a refletir as regras de negócio e os conceitos do mundo real (VALENTE, 2022). Na nova versão do Letstur, foram adotados alguns princípios do DDD para estruturar a arquitetura do software, garantindo uma melhor organização e compreensão das funcionalidades do sistema.

A camada de domínio, um dos principais componentes do DDD, é responsável por abrigar a lógica de negócio e os conceitos centrais do domínio da aplicação. Cada domínio iden-

tificado no projeto (empresas, usuários, vendas, eventos, cadastros e domínio compartilhado), é representado por um conjunto de modelos, serviços, políticas, entre outros artefatos, que encapsulam as regras de negócio específicas desse domínio.

Por exemplo, no domínio de usuários, podemos ter modelos de entidade como usuário, perfil e permissão, representando os elementos essenciais desse contexto. No domínio de vendas, modelos como item da venda, passageiro e voucher. Essa organização por domínios permite uma melhor compreensão das responsabilidades e interações entre os diferentes componentes do sistema. A Figura 13 ilustra a divisão dos módulos da camada de domínio da aplicação.

Figura 13 – Domínios da API



Fonte: Autoria própria (2024).

Em resumo, a aplicação do DDD no projeto proporciona uma série de benefícios, incluindo uma melhor organização das funcionalidades, uma comunicação mais eficaz e uma modelagem mais precisa do domínio do problema. Essa abordagem ajuda a garantir que o software desenvolvido atenda às necessidades do negócio de forma eficiente e sustentável ao longo do tempo.

4.2.2 Modelos e migrações do Laravel

Como apresentado anteriormente, foi adotado o Laravel para o desenvolvimento da aplicação. Uma das características fundamentais do *framework* é seu sistema de gerenciamento

de modelos e migrações, que desempenha um papel crucial na manutenção da consistência e integridade da base de dados em conjunto com a aplicação.

Os modelos no Laravel representam a interação entre a aplicação e os dados armazenados no banco de dados. Cada modelo corresponde a uma tabela no banco de dados e define as relações entre os diferentes conjuntos de dados. Através dos modelos, é possível realizar operações de consulta, inserção, atualização e exclusão de forma simplificada e intuitiva (LARAVEL, 2024c). A Figura 14, exibe o modelo de empresas, desenvolvido no sistema.

Figura 14 – Arquivo de modelo de empresa

```php
class Company extends Model
{
    use HasFactory, SoftDeletes;

    protected $fillable = [
        'name',
        'trading_name',
        'cnpj',
        'ie',
        'phone',
        'secondary_phone',
        'email',
    ];

    protected static function newFactory()
    {
        return CompanyFactory::new();
    }

    public function users()
    {
        return $this->hasMany(User::class);
    }
}
```

Fonte: Autoria própria (2024).

As migrações no Laravel são responsáveis por gerenciar a estrutura do banco de dados de forma programática e versionada. As migrações são arquivos de código que descrevem as alterações na estrutura do banco de dados, como a criação de tabelas, adição de colunas ou índices, entre outras modificações. Ao utilizar migrações, é possível garantir que a estrutura do banco de dados evolua de forma controlada e consistente ao longo do tempo, facilitando a manutenção e o versionamento do sistema (LARAVEL, 2024a). A Figura 15, exibe o arquivo de migração referente ao modelo de empresas.

Figura 15 – Arquivo de migração de empresa

```php
return new class extends Migration
{
    public function up(): void
    {
        Schema::create('companies', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('trading_name');
            $table->string('cnpj');
            $table->string('ie')->nullable();
            $table->string('phone');
            $table->string('secondary_phone')->nullable();
            $table->string('email');
            $table->timestamps();
            $table->softDeletes();
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('companies');
    }
};
```

Fonte: Autoria própria (2024).

Uma das principais vantagens do uso de modelos e migrações no Laravel é a garantia de que a base de dados permaneça atualizada e sincronizada com a aplicação. Como as alterações na estrutura do banco de dados são realizadas através de migrações, é possível acompanhar e controlar todas as mudanças feitas na base de dados ao longo do desenvolvimento do projeto. Isso proporciona uma maior segurança e consistência nos dados, evitando problemas de integridade e divergência entre a aplicação e o banco de dados.

Além disso, a utilização de modelos e migrações facilita a interação com o banco de dados, fornecendo uma camada de abstração que simplifica o acesso e manipulação dos dados. Os modelos e migrações do Laravel são integrados com o ORM Eloquent.

4.2.3 Fluxo de uma requisição

O processo de tratamento de uma requisição na API segue uma sequência estruturada, visando garantir a consistência e segurança das operações realizadas. Abaixo, estão descritos os passos desse fluxo para uma requisição simples:

1. **Validação da Requisição**: A requisição passa por um processo de validação, no qual são verificados os dados enviados pelo cliente. Esse processo é realizado através de classes de requisição (Requests) do Laravel, que definem as regras de validação para cada tipo de requisição.

2. **Verificação de Políticas e Permissões**: Após a validação, são verificadas as políticas e permissões associadas à requisição. As políticas do Laravel permitem definir regras de autorização com base em diferentes critérios, como o tipo de usuário ou o contexto da solicitação.

3. **Tratamento no Controlador**: Com a validação e autorização concluídas, a requisição é encaminhada para o controlador correspondente. O controlador é responsável por receber a requisição, realizar qualquer manipulação necessária nos dados e repassar o controle para o domínio da aplicação.

4. **Manipulação no Domínio da Aplicação**: No domínio da aplicação, são executadas as ações necessárias para atender à requisição. Isso pode envolver operações de persistência ou recuperação de dados no banco de dados, conforme as necessidades específicas da solicitação.

5. **Retorno ao Controlador**: Após o processamento no domínio da aplicação, os resultados são retornados ao controlador. Esses resultados podem incluir dados a serem enviados como resposta à requisição, bem como informações sobre o estado da operação.

6. **Formatação da Resposta**: No controlador, os dados retornados são formatados para serem entregues ao cliente de acordo com o formato especificado na requisição. Isso é geralmente feito através de recursos (Resources) do Laravel, que definem como os dados devem ser apresentados na resposta.

7. **Resposta à Requisição**: Por fim, a resposta é enviada de volta ao cliente, contendo os dados solicitados no formato especificado na requisição. Por se tratar de uma API, a resposta é padronizada no formato JSON.

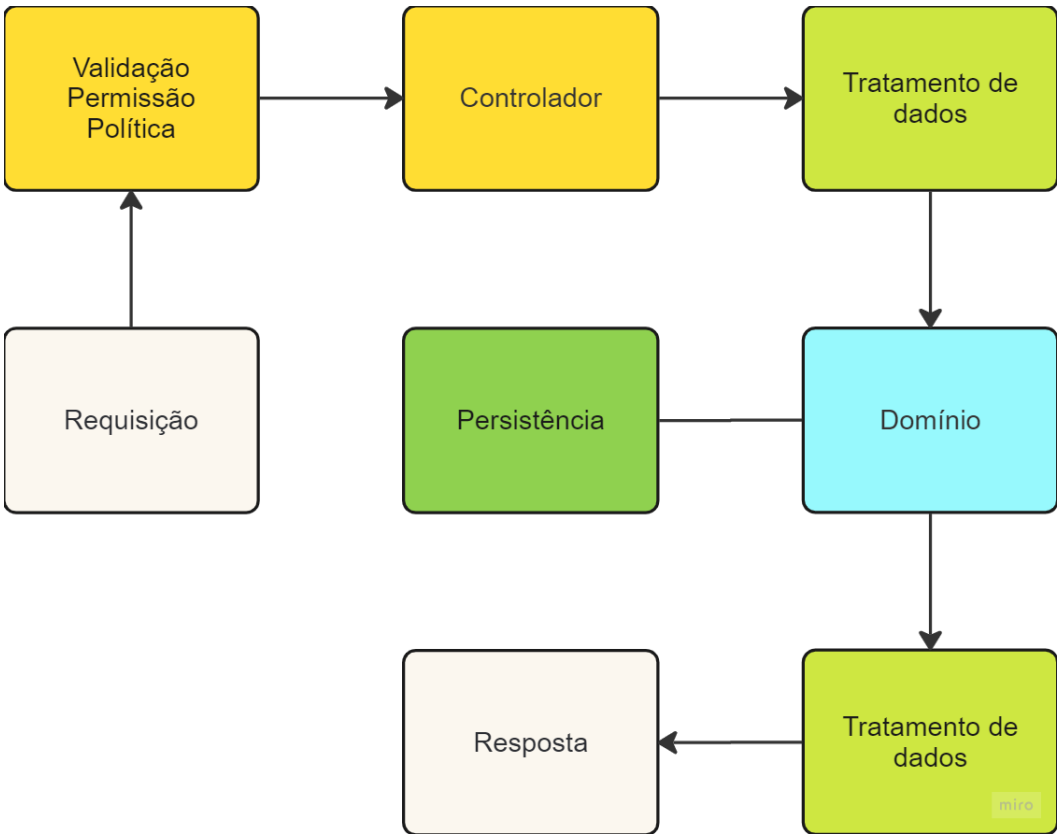
Esse fluxo estruturado busca garantir uma abordagem consistente e segura para o tratamento de requisições na API, seguindo as melhores práticas de desenvolvimento com Laravel. A Figura 16 ilustra o fluxo citado acima.

4.2.4 Permissionamento

Na aplicação da API, foi adotada uma estratégia de implementação de um sistema de permissões para garantir o controle adequado de acesso aos recursos do sistema, visando a segurança e a integridade dos dados. Essa abordagem baseia-se na atribuição de funções e permissões aos usuários, proporcionando uma estrutura organizada e flexível para gerenciar os direitos de acesso.

Inicialmente, são definidas as funções ou papéis que os usuários podem desempenhar dentro da aplicação. Essas funções podem incluir diferentes níveis de acesso e responsabilida-

Figura 16 – Fluxograma - Requisição na API



Fonte: Autoria própria (2024).

des, que atualmente no Letstur são: Administrador geral, administrador de empresa e operador de vendas da empresa.

Após a definição das funções, essas são atribuídas aos usuários com base em suas responsabilidades e privilégios dentro da aplicação. Por exemplo, um administrador pode ter acesso às funcionalidades do sistema que envolvem várias empresas e usuários, enquanto um administrador de empresa terá permissões mais limitadas, restritas apenas às tarefas específicas da sua empresa.

Paralelamente, são definidas as permissões associadas a cada função, especificando quais recursos ou ações um usuário com determinada função pode realizar na aplicação. Essas permissões podem incluir operações como adicionar, editar, visualizar ou excluir dados, dependendo das necessidades do sistema e das responsabilidades dos usuários.

As permissões são então associadas às funções correspondentes, permitindo que os usuários recebam automaticamente as permissões atribuídas à função que desempenham. Isso simplifica o processo de gerenciamento de permissões e garante uma abordagem consistente e escalável para o controle de acesso.

Durante o uso da aplicação, as permissões são verificadas em tempo real para garantir que apenas usuários autorizados possam acessar determinados recursos ou executar determinadas ações. Essas verificações são integradas em pontos críticos do código, como rotas e

controladores, garantindo que o acesso aos recursos seja estritamente controlado de acordo com as permissões atribuídas.

Em casos de tentativas de acesso não autorizado, o sistema responde com a recusa ao recurso, informando-o sobre a falta de permissões necessárias para realizar a ação desejada. Isso contribui para uma experiência do usuário mais transparente e ajuda a evitar erros ou violações de segurança.

Para controlar todo esse sistema de permissionamento junto ao Laravel, foi utilizado o Spatie Laravel Permission, que simplifica o trabalho de associação de papéis e permissões dentro da aplicação, e cria tabelas específicas para isso no banco de dados (SPATIE, 2024). A Figura 17 exibe como o arquivo de configurações do Laravel Permission, associa os papeis com suas devidas permissões.

Figura 17 – Configurações de permissionamento

```php
class PermissionsAndRoles
{
    public static function getRoles(): array
    {
        return [
            RoleEnum::ADMIN => [
                'users' => $permissions('users'),
                'companies' => $permissions('companies'),
            ],

            RoleEnum::COMPANY_OPERATOR => [
                'events' => collect(['view', 'view any']),
                'tours' => collect(['view', 'view any']),
                'sales' => collect(['view', 'view any', 'create', 'update', 'delete']),
            ],

            RoleEnum::COMPANY_ADMIN => [
                'users' => $permissions('users'),
                'companies' => collect(['view', 'update']),
                'drivers' => $permissions('drivers'),
                'vehicles' => $permissions('vehicles'),
                'tour-guides' => $permissions('tour-guides'),
                'locales' => $permissions('locales'),
                'tour-types' => $permissions('tour-types'),
                'tours' => $permissions('tours'),
                'events' => $permissions('events'),
                'sales' => collect(['view', 'view any', 'confirm', 'delete']),
            ],
        ];
    }
}
```

Fonte: Autoria própria (2024).

4.3 Estruturação dos testes

Na API da nova versão dos Letstur, foi adotada uma abordagem abrangente de testes utilizando o framework Laravel, aproveitando suas ferramentas integradas para garantir a robustez e estabilidade do sistema.

Foram implementados testes unitários e testes de integração para cobrir diferentes aspectos da aplicação e garantir que cada componente funcione conforme o esperado. Os testes unitários foram utilizados para testar cada classe com regras de negócio no sistema de forma isolada, garantindo que cada unidade de código funcione corretamente e produza os resultados esperados.

Além disso, os testes de integração foram empregados para testar as funcionalidades do sistema como um todo, verificando a interação entre os diferentes componentes e garantindo que todas as partes funcionem em conjunto. Esses testes foram executados utilizando um banco de dados de teste, proporcionando um ambiente realista para simular o comportamento da aplicação em produção.

Um recurso fundamental do Laravel que foi amplamente utilizado durante os testes de integração são as *factories*. As *factories* são classes que permitem a criação de dados de teste de forma rápida e fácil, fornecendo uma maneira conveniente de gerar dados fictícios para os testes. Isso simplifica significativamente a escrita dos testes de integração, pois elimina a necessidade de criar manualmente registros de banco de dados para cada cenário de teste.

As *factories* do Laravel são altamente configuráveis e permitem a geração de dados realistas com facilidade. Elas podem ser personalizadas para criar instâncias de modelos com diferentes atributos e relações, tornando-as extremamente versáteis para atender às necessidades específicas de cada teste (LARAVEL, 2024b).

Ao utilizar as *factories* do Laravel, foi possível agilizar o processo de escrita dos testes de integração e garantir uma cobertura abrangente das funcionalidades do sistema. Isso contribui para a detecção precoce de erros e problemas de integração, permitindo suas correções antes de serem implantados em produção.

4.4 Estruturação da interface

Nesta seção, estão descritas algumas decisões relevantes para o desenvolvimento da interface do sistema.

4.4.1 Design System

No desenvolvimento deste projeto, uma abordagem para definir a identidade visual da aplicação foi adotada para garantir a consistência e a usabilidade da interface, resultando na criação de um *design system* robusto e coeso.

Uma das principais considerações durante o processo de *design* foi a análise do site da empresa Letstur, responsável pelo sistema. Observar e compreender a identidade visual e os padrões de *design* já estabelecidos pela empresa permitiu alinhar a nova interface com a imagem da marca e garantir uma experiência de usuário mais consistente em todas as plataformas, dando importância a responsividade da aplicação. A Figura 18 é uma captura do site oficial do Letstur.

Figura 18 – Site oficial do Letstur



Fonte: Letstur (2014).

Além disso, foi elaborado um protótipo de telas que serviu como base para o desenvolvimento do *design* da interface. Esse protótipo definiu a estrutura e o fluxo de navegação da aplicação, proporcionando uma direção clara para o desenvolvimento do design visual.

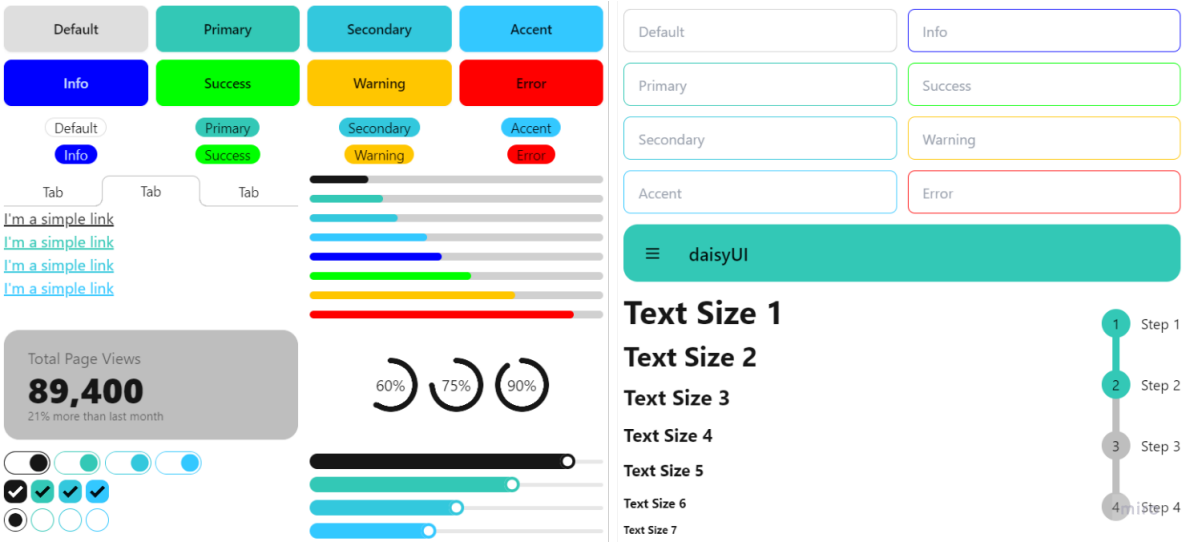
Para implementar o *design* da interface, foi decidido utilizar o *plugin* Daisy UI do Tailwind. O Daisy UI é uma extensão do Tailwind que fornece uma biblioteca de componentes prontos para uso, simplificando o processo de criação de interfaces responsivas e esteticamente agradáveis DAISYUI (2024).

Uma das vantagens do Daisy UI é a variedade de componentes disponíveis, que abrangem desde botões e formulários até *cards* e diálogos. Esses componentes pré-construídos aceleram o desenvolvimento da interface, permitindo o foco maior na funcionalidade da aplicação e menor na implementação detalhada dos elementos de design.

Além disso, o Daisy UI facilita o trabalho com cores e fontes, oferecendo uma gama de opções para personalizar a aparência da interface de acordo com as preferências e necessidades do projeto. Isso permite uma flexibilidade maior no *design* da interface, garantindo que ela se adapte adequadamente à identidade visual da marca e às expectativas dos usuários. A

Figura 19 ilustra o *design system* da aplicação e a aparência de alguns componentes utilizados durante o desenvolvimento do sistema.

Figura 19 – Design System



Fonte: Autoria própria (2024).

4.4.2 Estrutura de pastas da aplicação de interface

A estrutura da aplicação de interface, desenvolvida com Vue.js e Nuxt.js, segue uma abordagem organizada e modular, visando garantir a escalabilidade, manutenção e reutilização de código. A organização dos arquivos e componentes da aplicação é baseada nos domínios funcionais do sistema, que incluem autenticação, empresas, usuários, vendas, eventos e cadastros.

A estrutura de diretórios do Nuxt é aplicada, com páginas, componentes, serviços, entre outros artefatos, e em cada diretório é feita uma separação pelos domínios funcionais aplicáveis. Essa abordagem de organização por domínios facilita a localização e manutenção dos arquivos relacionados a cada parte da aplicação, tornando o código mais legível e coeso.

Além disso, os componentes são separados em dois tipos principais: componentes específicos do domínio e componentes compartilhados. Os componentes específicos do domínio contêm a lógica e a apresentação relacionadas às funcionalidades específicas desse domínio, como formulários de cadastro de usuários ou listagens de vendas. Por outro lado, os componentes compartilhados são genéricos, reutilizáveis e não contêm lógica específica do domínio, como botões, barras de navegação ou modais.

Essa separação entre componentes específicos do domínio e compartilhados permite uma maior reutilização e uma melhor manutenção na base de código da aplicação. Os componentes compartilhados podem ser facilmente importados e utilizados em várias partes da aplicação, reduzindo a duplicação de código e aumentando a consistência visual e funcional.

Além da organização dos componentes, a estrutura da aplicação também inclui serviços, que são responsáveis por realizar chamadas de API, gerenciar o estado da aplicação ou executar outras tarefas relacionadas à lógica de negócios. Esses serviços são agrupados dentro de cada domínio funcional, garantindo uma separação clara das responsabilidades e uma melhor organização do código.

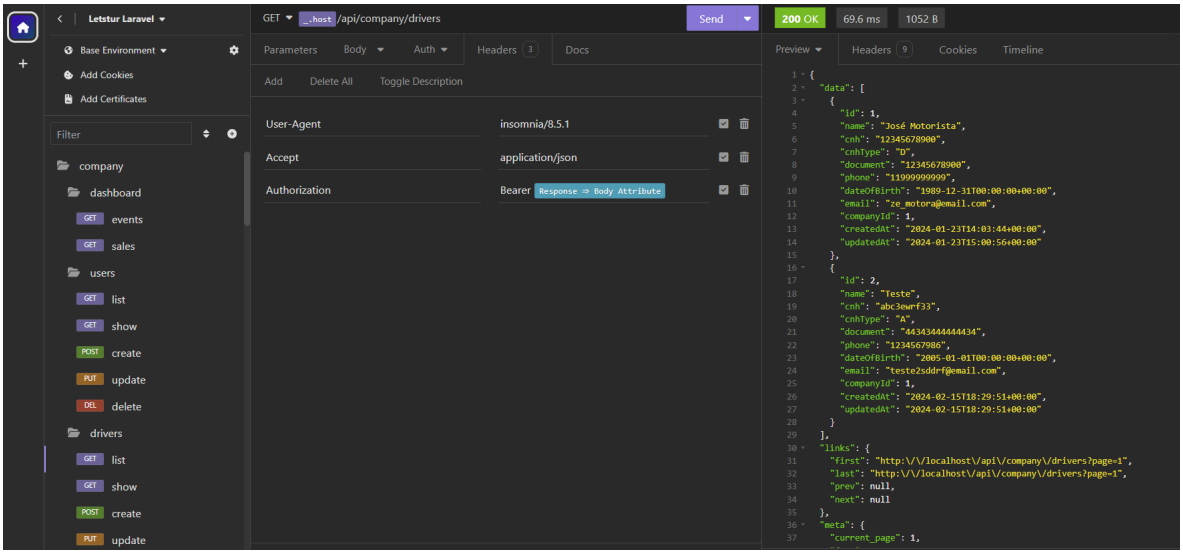
4.5 Estruturação da documentação

A documentação da API é parte essencial do desenvolvimento da aplicação, pois fornece uma referência clara e precisa sobre como interagir com os *endpoints* da API. No projeto, a documentação da API foi criada e mantida utilizando a plataforma Insomnia, uma ferramenta para testes de API e documentação (INSOMNIA, 2024).

A documentação no Insomnia foi estruturada de forma a refletir a organização dos *endpoints* e recursos da API. Cada *endpoint* é cuidadosamente documentado, incluindo informações detalhadas sobre sua funcionalidade, parâmetros de entrada, métodos HTTP suportados, e exemplos de solicitações e respostas.

Para garantir que a documentação da API esteja sempre atualizada, é adotado um processo de importação e exportação de arquivos YAML do Insomnia. Sempre que há alterações nos *endpoints* ou funcionalidades da API, a documentação é atualizada no Insomnia e exportada para um arquivo YAML, que é então integrado ao projeto. Ao importar o arquivo dentro da aplicação do Insomnia, é possível ter todas as informações necessárias para fazer as requisições a API, conforme a Figura 20.

Figura 20 – Exemplo de requisição no Insomnia



Fonte: Autoria própria (2024).

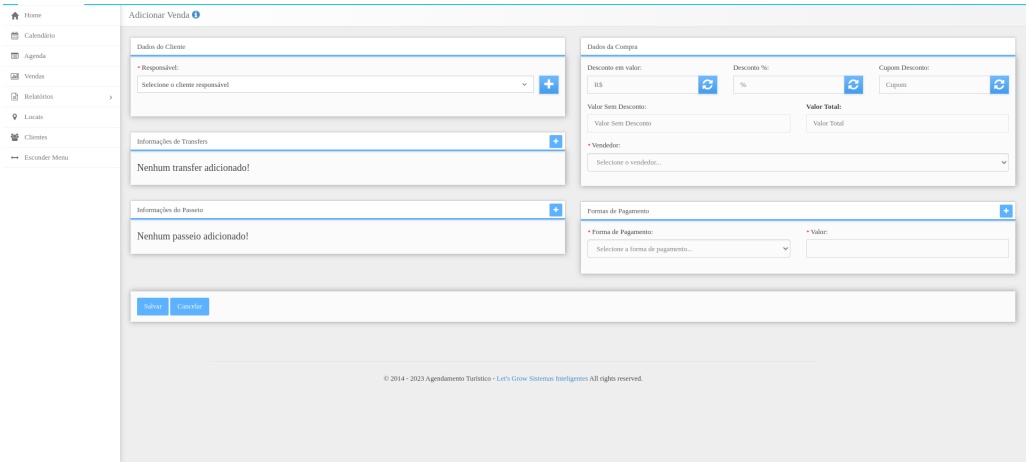
5 RESULTADOS

Este capítulo apresenta o que foi obtido como resultado do trabalho, enquanto este trabalho foi desenvolvido.

5.1 Avaliação da atual versão do sistema

Após a análise do fluxo da plataforma, percebeu-se que atualmente o sistema enfrenta algumas limitações tecnológicas, que futuramente podem inviabilizar seu crescimento e abrangência na área de operadores de passeio turístico. Um dos principais pontos de possível melhoria da aplicação, é seu painel administrativo. Ao realizar uma venda, uma das funções utilizadas com mais frequência no sistema, diversas informações irrelevantes precisam ser preenchidas, conforme pode ser visto na Figura 21, tornando o processo lento, o que pode vir a afetar a satisfação dos operadores de vendas das agências que consomem o serviço.

Figura 21 – Tela de cadastro de vendas.

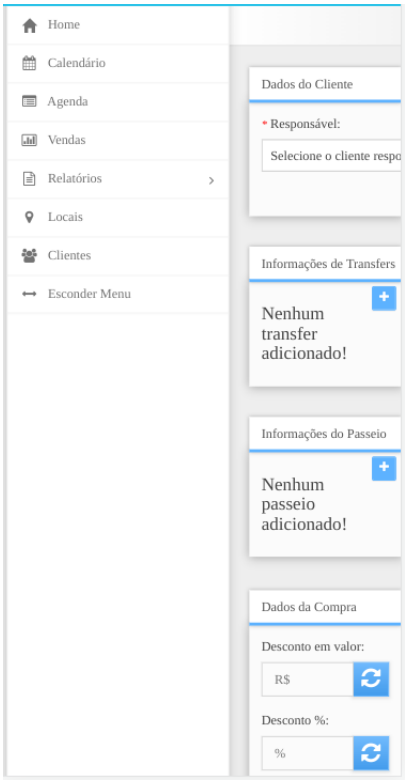


Fonte: Letstur (2014).

Outro aspecto que pode ser otimizado no sistema é a falta de responsividade. Em um contexto no qual a maioria das empresas opta por soluções que sejam funcionais em dispositivos móveis, é importante que a plataforma seja adaptada e otimizada para *smartphones* e *tablets*. As falhas de responsividade no Letstur, torna-o um sistema limitado quando comparado a outras soluções. A Figura 22, retrata como o sistema é renderizado em dispositivos com telas menores, como celulares e tablets.

Devido ao longo período que o sistema está em uso, juntamente ao fato das tecnologias utilizadas originalmente não receberem mais manutenção e cuidados satisfatórios quando comparadas a tecnologias mais recentes, o Letstur está exposto a possíveis problemas relacionados ao desempenho de aplicação e busca de dados. Esta limitação pode afetar diversas funcionalidades do sistema, comprometendo a eficiência de suas operações. Tarefas que, em um ambiente ideal, seriam conduzidas de maneira ágil e precisa, podem vir a ser afetadas por

Figura 22 – Falha na responsividade na tela de vendas.



Fonte: Letstur (2014).

restrições tecnológicas. Como consequência, é possível estimar uma operação menos eficaz ao decorrer dos próximos anos, o que contribuirá para uma experiência insatisfatória por parte dos operadores de vendas, e administradores de agências.

Um dos principais desafios decorrentes do modelo SaaS atual do Letstur é a multiplicidade de instâncias da aplicação. Esse cenário fragmentado não apenas dificulta a manutenção e atualização eficientes do sistema, como também pode comprometer a consistência e integridade dos dados das agências clientes.

5.2 Análise e levantamento de requisitos

Após a avaliação da versão atual do sistema, tornou-se necessário o levantamento dos requisitos que guiam o desenvolvimento da próxima versão do Letstur. Esse processo de análise é crucial para assegurar que a nova implementação do sistema seja concebida de maneira estratégica, alinhada aos objetivos específicos de satisfação dos requisitos identificados.

O Quadro 1, apresentado a seguir, detalha de forma abrangente os requisitos funcionais da aplicação. Os requisitos elencados, constituem o núcleo do planejamento de desenvolvimento, fornecendo uma base sólida para a concepção de funcionalidades que atendam às expectativas e demandas dos usuários finais.

Já o Quadro 2 se destina a catalogar e especificar os requisitos não-funcionais da aplicação, ampliando a abrangência da análise para além das funcionalidades. Esses requisitos, muitas vezes imperceptíveis ao usuário final, desempenham um papel crucial na definição de parâmetros como desempenho, segurança, escalabilidade e usabilidade. Ao incorporar esses elementos não-funcionais, busca-se garantir não apenas a eficácia operacional do sistema, mas também sua funcionalidade em aspectos que transcendem a execução de tarefas, a fim de garantir a qualidade do sistema.

Quadro 1 – Requisitos funcionais do sistema

Requisito	Usuário	Descrição
RF001	Todos	Autenticação com email e senha
RF002	Todos	Edição de perfil e senha
RF003	Administrador geral	Cadastro de usuários
RF004	Administrador geral	Cadastro de agências
RF005	Agência	Calendário de passeios
RF006	Administrador de agência	Edição de dados da agência
RF007	Administrador de agência	Cadastro de operadores
RF008	Administrador de agência	Cadastro de eventos
RF009	Administrador de agência	Cadastro de passeios
RF010	Administrador de agência	Cadastro de motoristas
RF011	Administrador de agência	Cadastro de guias
RF012	Administrador de agência	Cadastro de veículos
RF013	Administrador de agência	Relatório de vendas
RF014	Administrador de agência	Confirmação de vendas
RF015	Administrador de agência	Cancelamento de vendas
RF016	Operador de vendas	Ponto de vendas de eventos
RF017	Operador de vendas	Emissão de voucher
RF018	N/A	Envio de voucher por email

Fonte: Autoria própria (2024).

5.3 Modelagem do banco de dados

A modelagem do banco de dados é uma fase crucial no desenvolvimento da nova versão do sistema, desempenhando um papel fundamental na organização e estruturação dos dados que serão armazenados e gerenciados pela aplicação. Após o levantamento e análise de dados, foi implementado uma modelagem física de dados, que posteriormente foi traduzida em

Quadro 2 – Requisitos não-funcionais do sistema

Requisito	Descrição
RNF001	A interface do usuário deve ser intuitiva
RNF002	A infraestrutura do sistema deve ser construída com contêineres do docker
RNF003	A API deve ser construída utilizando PHP e Laravel
RNF004	A interface deve ser construída utilizando VueJs
RNF005	A interface do usuário deve ser responsiva
RNF006	A aplicação deve suportar pelo menos 1000 acessos simultâneos
RNF007	O tempo de resposta das requisições não devem ultrapassar 3 segundos
RNF008	Nenhuma agência pode visualizar informação de outras
RNF009	O sistema deve ser compatível com os principais navegadores.
RNF010	Deve ser possível acessar o sistema em dispositivos móveis.
RNF011	O sistema deve ser capaz de ser implantado em ambientes de nuvem.

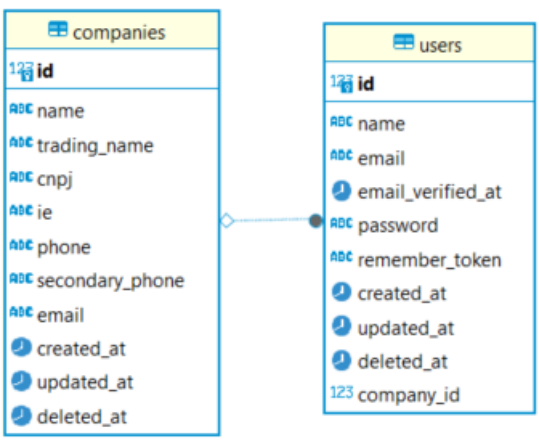
Fonte: Autoria própria (2024).

um banco de dados real, cujo a API o gerencia através de regras de negócios. A modelagem física de dados do sistema proposta inicialmente, para melhor entendimento e visualização, está divida em alguns contextos chave, descritos a seguir.

5.3.1 Usuários e empresas

Na nova versão do sistema Letstur, foi definido que cada empresa pode ter vários usuários vinculados, porém um usuário nem sempre precisa estar vinculado a uma empresa, como é o caso dos administradores gerais. A Figura 23 exibe a modelagem dos dados de usuários e empresas,

Figura 23 – Modelagem de empresas e usuários no banco de dados

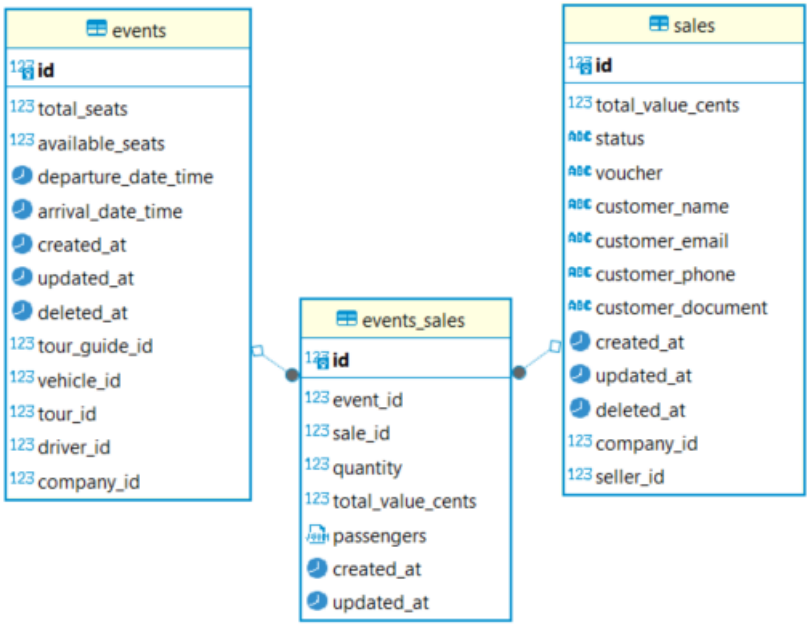


Fonte: Autoria própria (2024).

5.3.2 Vendas de eventos

A área de vendas no banco de dados, é responsável por armazenar todas as informações relacionadas aos eventos vendidos, como cliente, valores, passageiros, número do voucher, etc. A Figura 24 exibe a modelagem dos dados da área de vendas,

Figura 24 – Modelagem da área de vendas no banco de dados



Fonte: Autoria própria (2024).

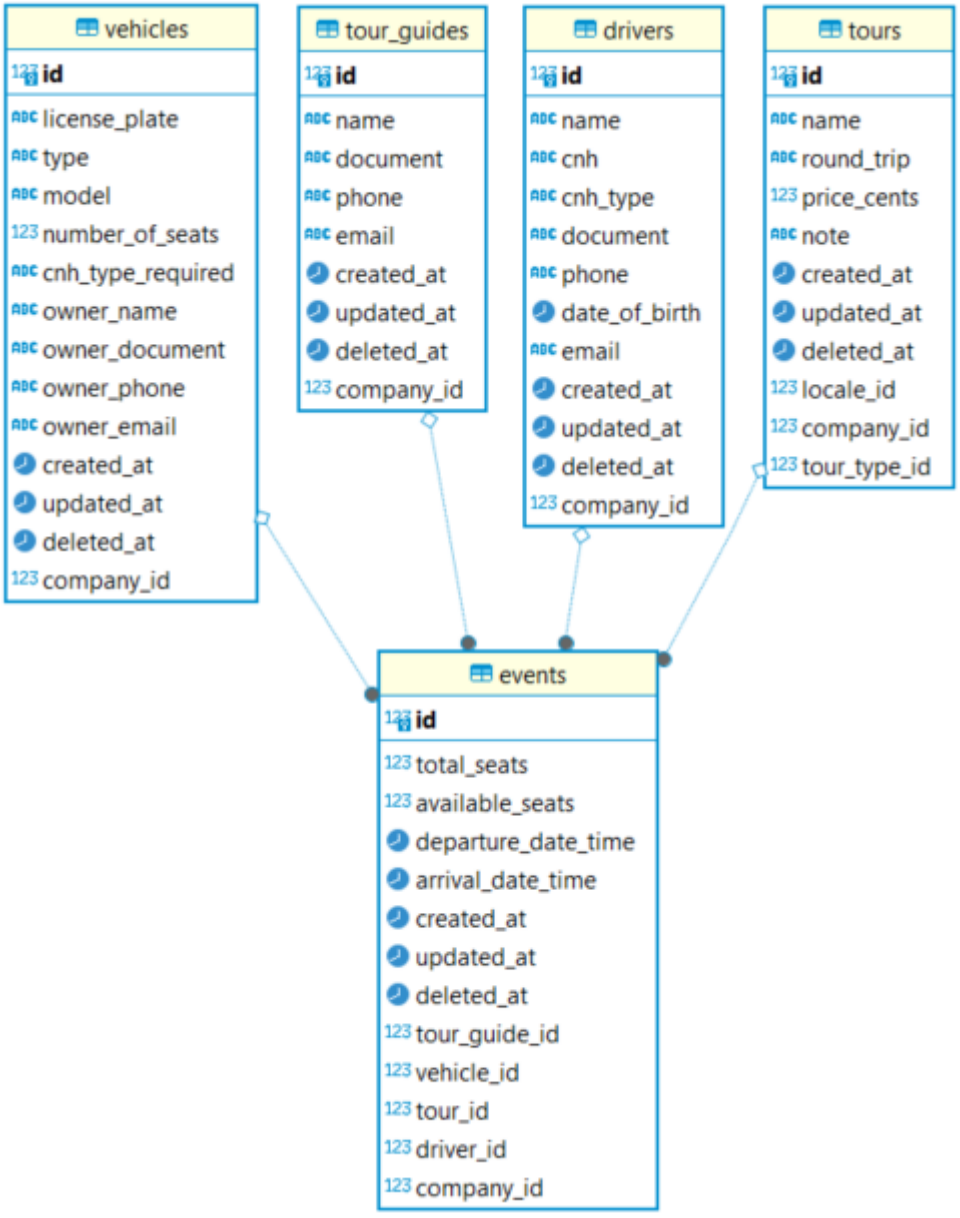
5.3.3 Cadastro de eventos

Nas tabelas relacionadas aos eventos, ficarão salvas informações como veículo utilizado, guia de passeio, motorista e local do passeio de cada evento cadastrado, além de outras informações pertinentes sobre o evento. A Figura 25 exibe a modelagem dos dados da área de cadastro de eventos,

5.3.4 Cadastro de passeios

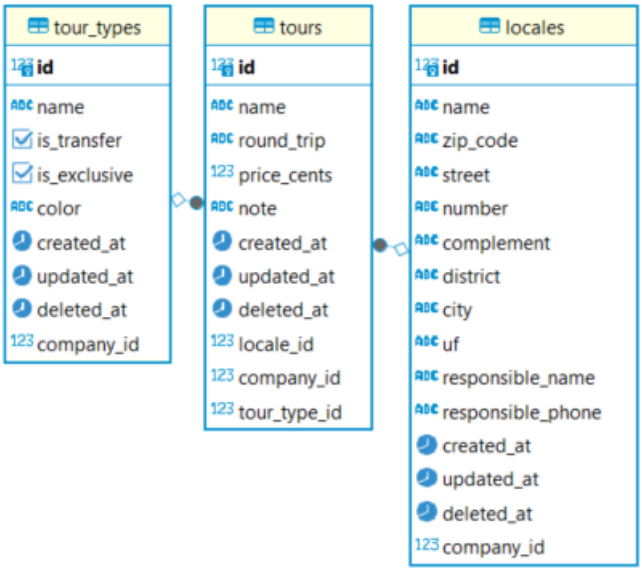
Na nova versão, cada passeio precisará ser vinculado a um tipo de passeio e a um local, o que facilita e agiliza posteriormente, a criação de novos passeios, já que muitas informações não precisarão ser preenchidas novamente. A Figura 26 exibe a modelagem dos dados da área de cadastro de passeios,

Figura 25 – Modelagem da área de eventos no banco de dados



Fonte: Autoria própria (2024).

Figura 26 – Modelagem da área de passeios no banco de dados



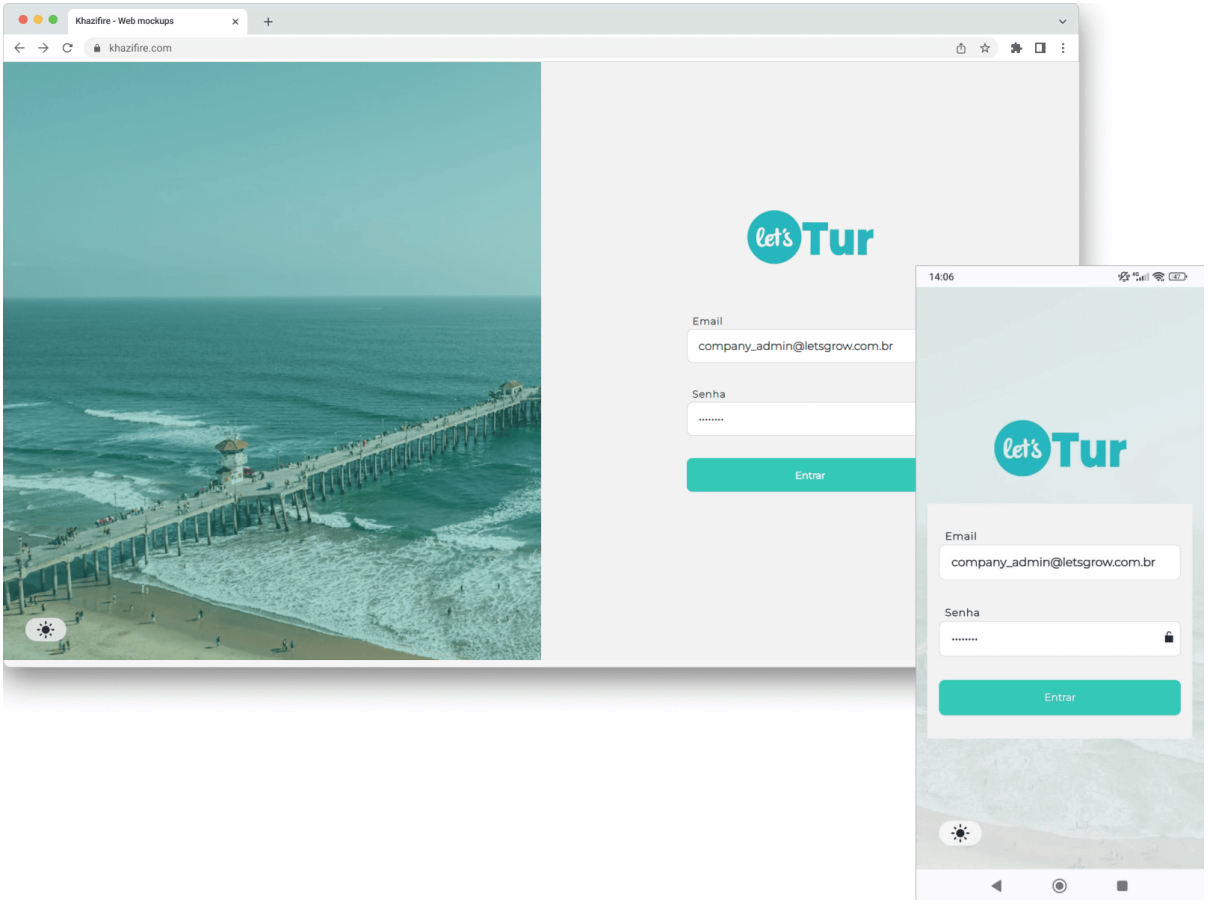
Fonte: Autoria própria (2024).

5.4 Interface

Após as definições de requisitos, modelagem de dados e prototipação, através do desenvolvimento da API e interface, chegou-se aos resultados, que estão expostos a seguir.

A área de *login* é a porta de entrada da aplicação, onde os usuários previamente cadastrados podem se autenticar, e acessar os recursos que tem permissão. A Figura 27 exibe a tela de *login*.

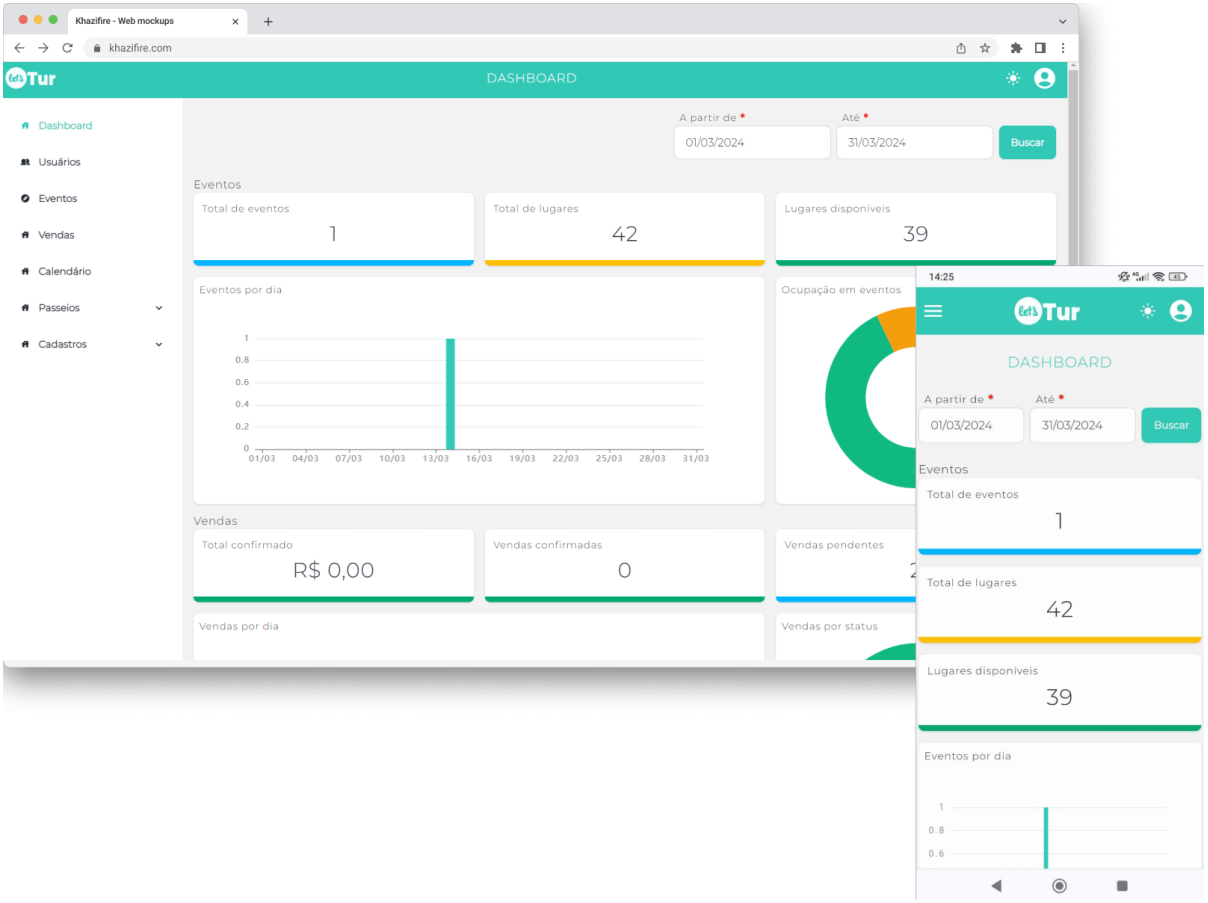
Figura 27 – Tela de login



Fonte: Autoria própria (2024).

Após acessar a área autenticada da aplicação, o usuário chega a tela de *dashboard*, que contém informações de consolidação sobre eventos e vendas. Um administrador de empresa, vê informações sobre todos os operadores de venda, enquanto um operador, visualiza apenas informações das próprias vendas. A Figura 28 exibe a tela de *dashboard*.

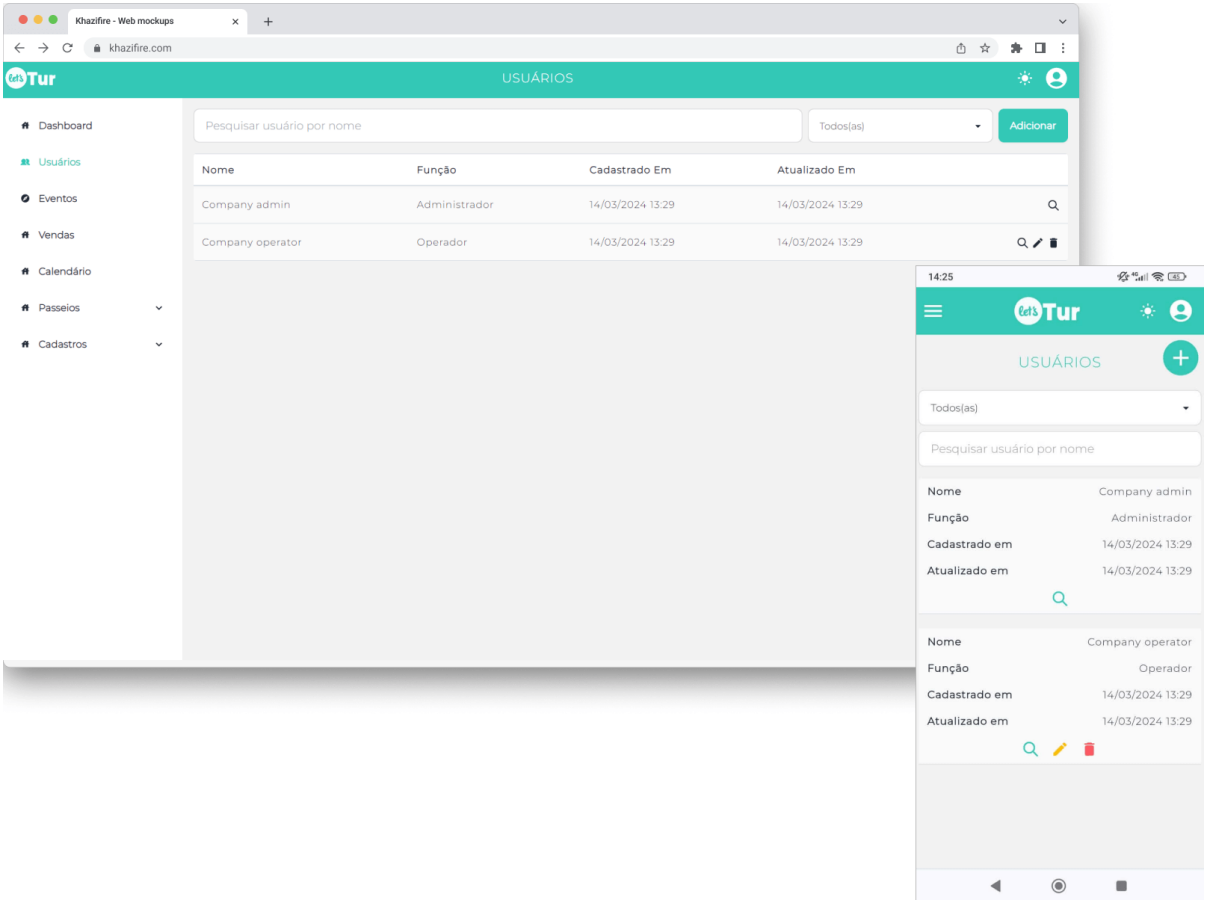
Figura 28 – Tela de dashboard



Fonte: Autoria própria (2024).

A área de usuários da aplicação pode ser visualizada pelos administradores gerais, que podem visualizar informações de todos os usuários da aplicação, e pelos administradores das empresas, que conseguem visualizar informações dos usuários de suas empresas. Na área de usuários também é possível criar, editar e excluir usuários. A Figura 29 exibe a listagem de usuários na interface.

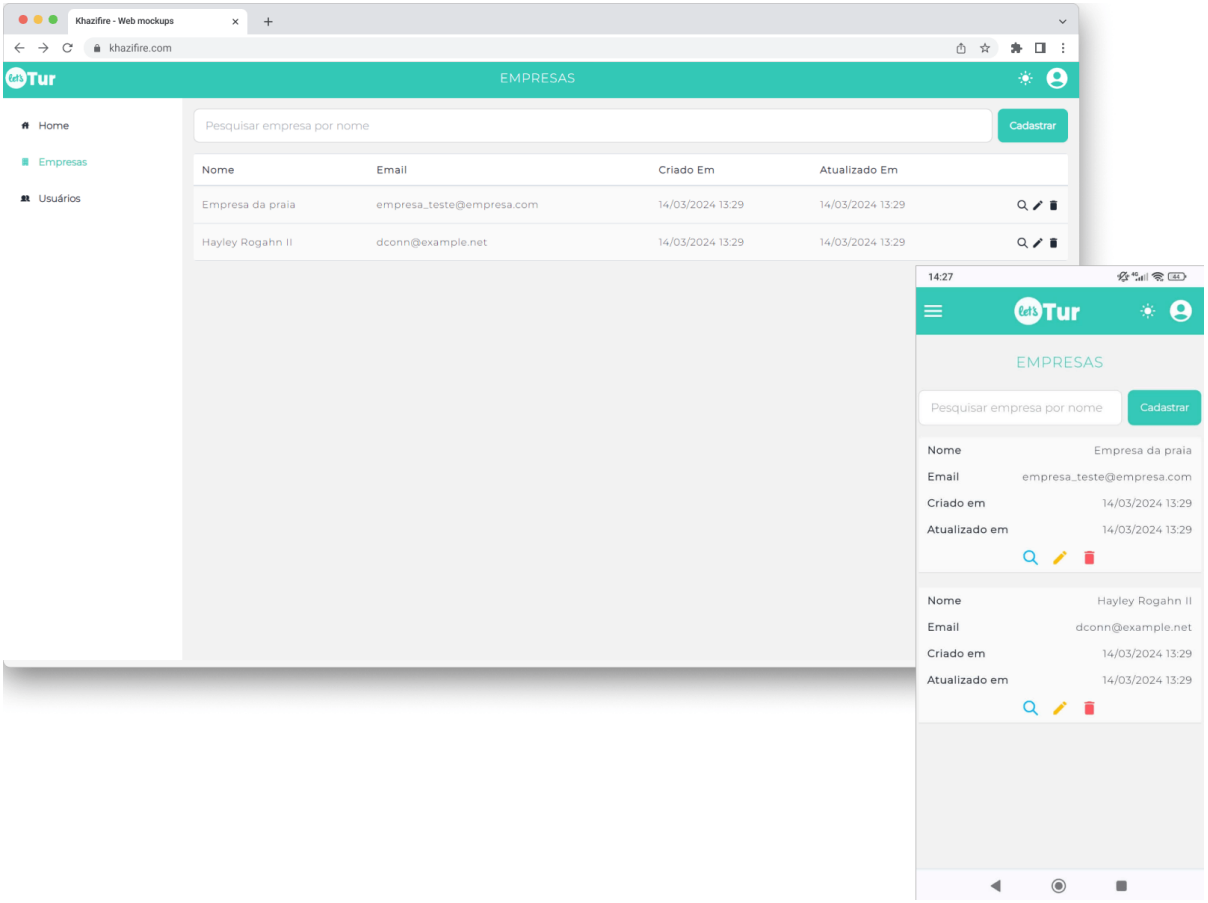
Figura 29 – Tela de listagem de usuários



Fonte: Autoria própria (2024).

A aplicação também possui uma área de controle de empresas, que é acessada pelos administradores gerais para administrar todas as empresas que usam o sistema. A Figura 30 exibe a listagem de empresas.

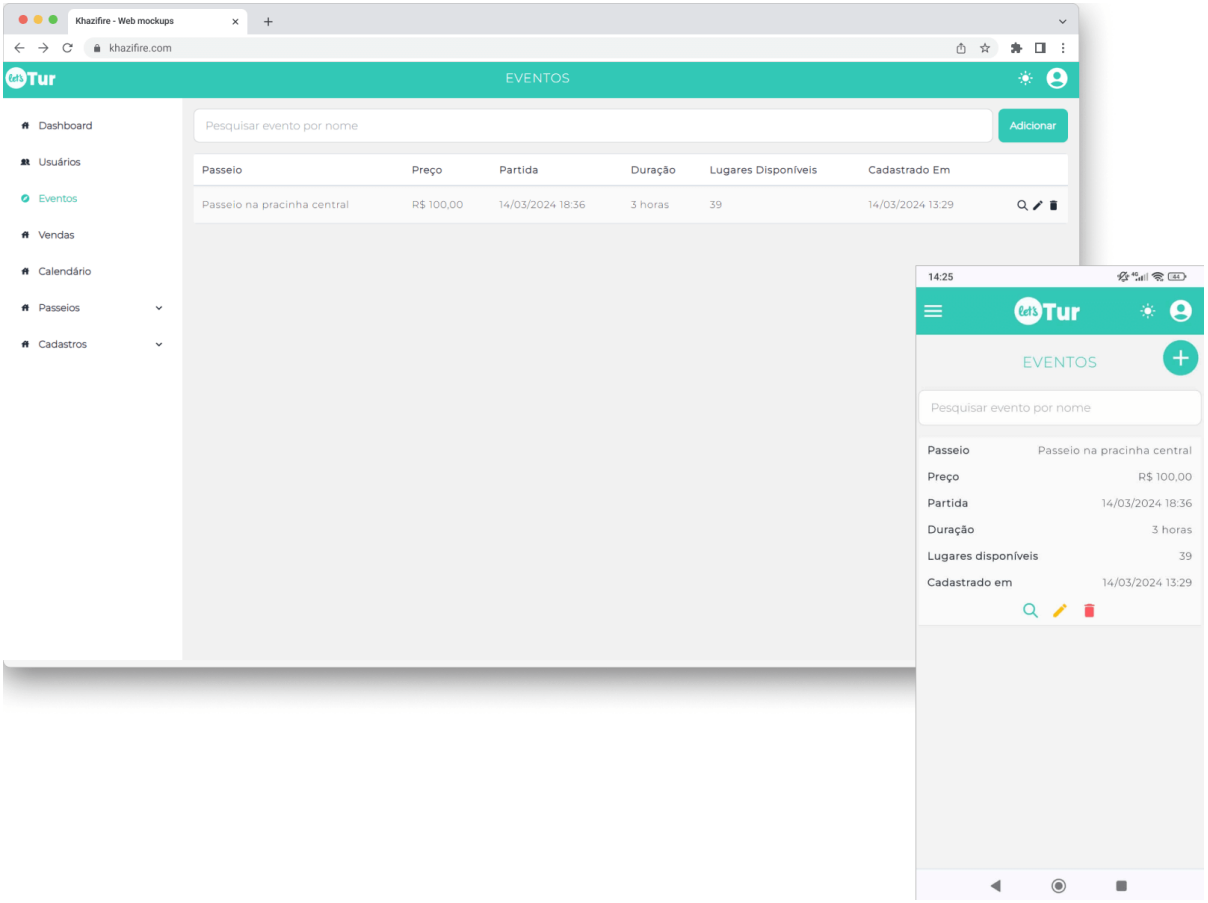
Figura 30 – Tela de listagem de empresas



Fonte: Autoria própria (2024).

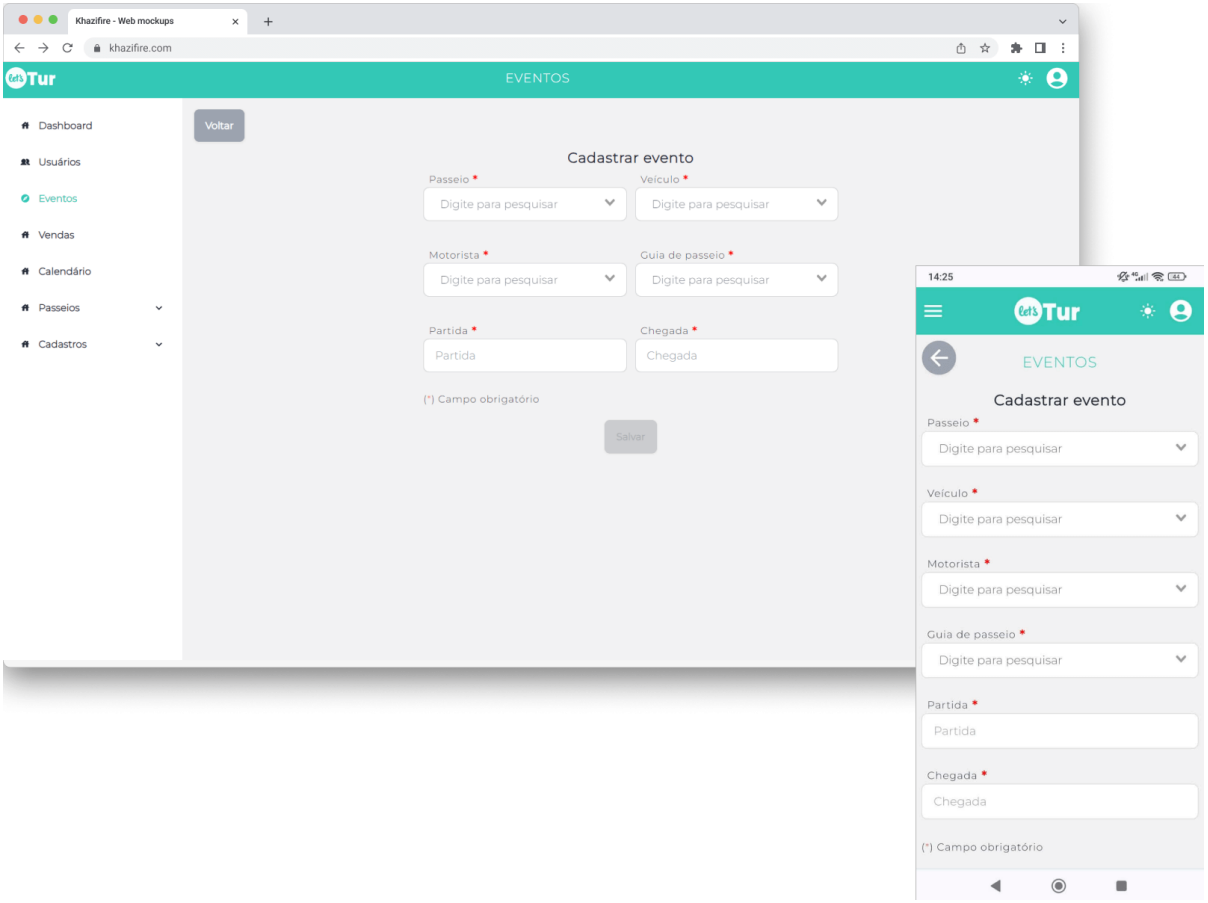
A área de eventos da aplicação, é um dos pontos centrais, onde os administradores conseguem cadastrar as informações relevantes de todos os eventos, como veículo utilizado, motorista, passeio, assentos totais e disponíveis, entre outros dados importantes pro controle dos eventos. A Figura 31 exibe a listagem de eventos criados, enquanto a Figura 32 exibe o formulário de criação/edição de eventos.

Figura 31 – Tela de listagem de eventos



Fonte: Autoria própria (2024).

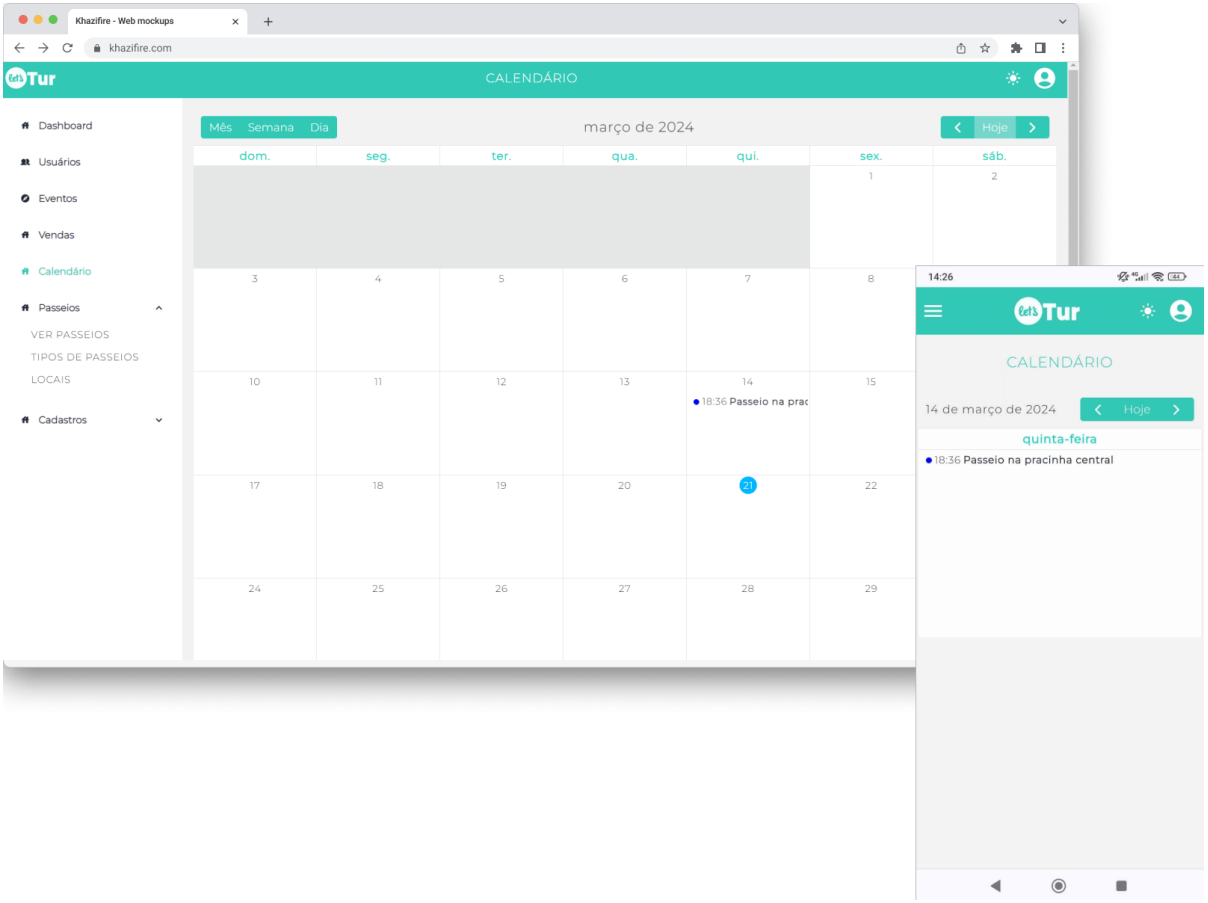
Figura 32 – Tela de formulário de eventos



Fonte: Autoria própria (2024).

Para auxiliar a visualização dos eventos próximos, foi criada a tela de calendário, que exibe as informações importantes para os operadores, de uma maneira mais rápida e intuitiva, conforme exibido na Figura 33.

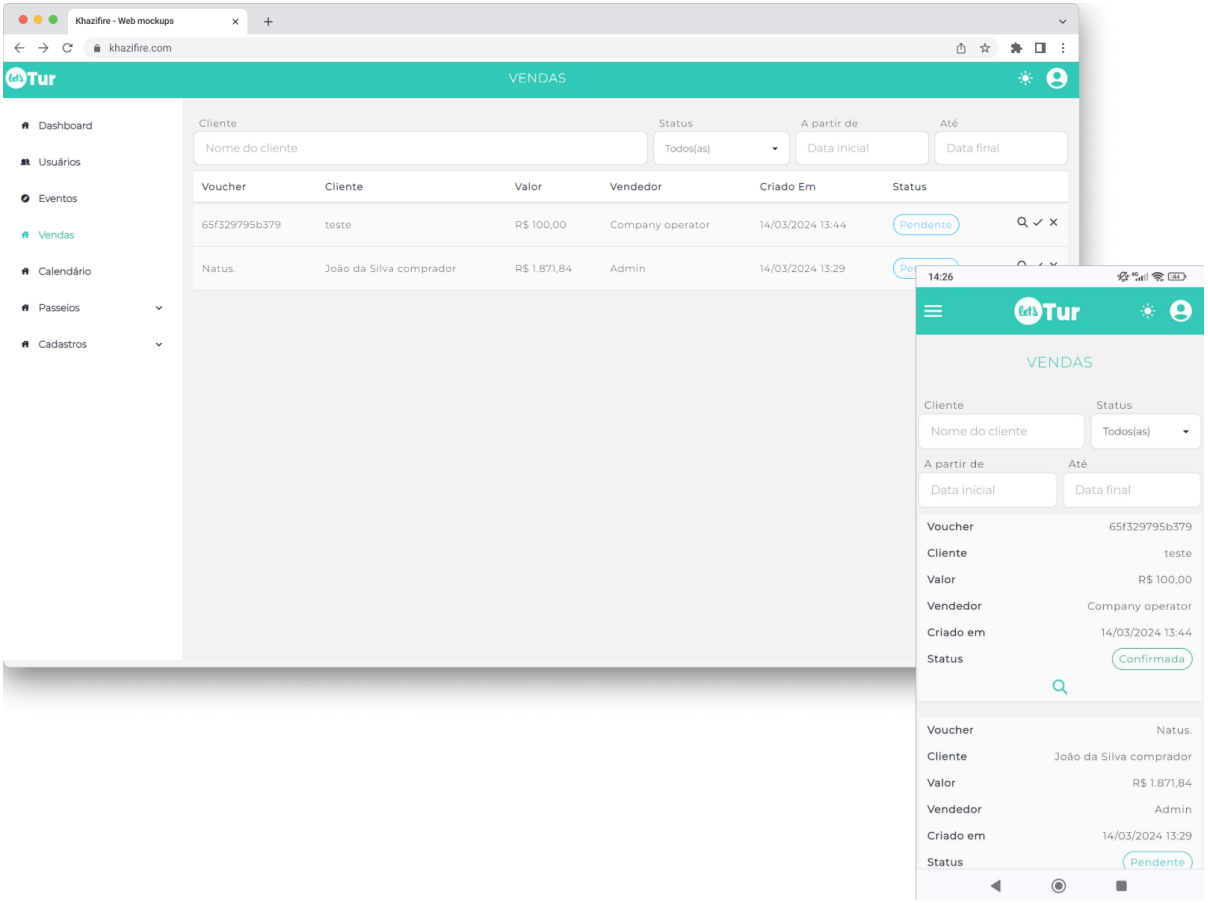
Figura 33 – Tela de calendário



Fonte: Autoria própria (2024).

A área de vendas da aplicação, pode ser acessada pelos administradores de empresas, que conseguem visualizar as vendas de todos os operadores, e tomar decisões sobre elas, como confirmar a venda, o que envia um voucher ao cliente, ou cancelar, o que libera os assentos utilizados nos eventos. A área de vendas também pode ser acessada pelos operadores, que podem visualizar, cancelar e cadastrar vendas. A Figura 34 exibe a listagem de vendas no sistema.

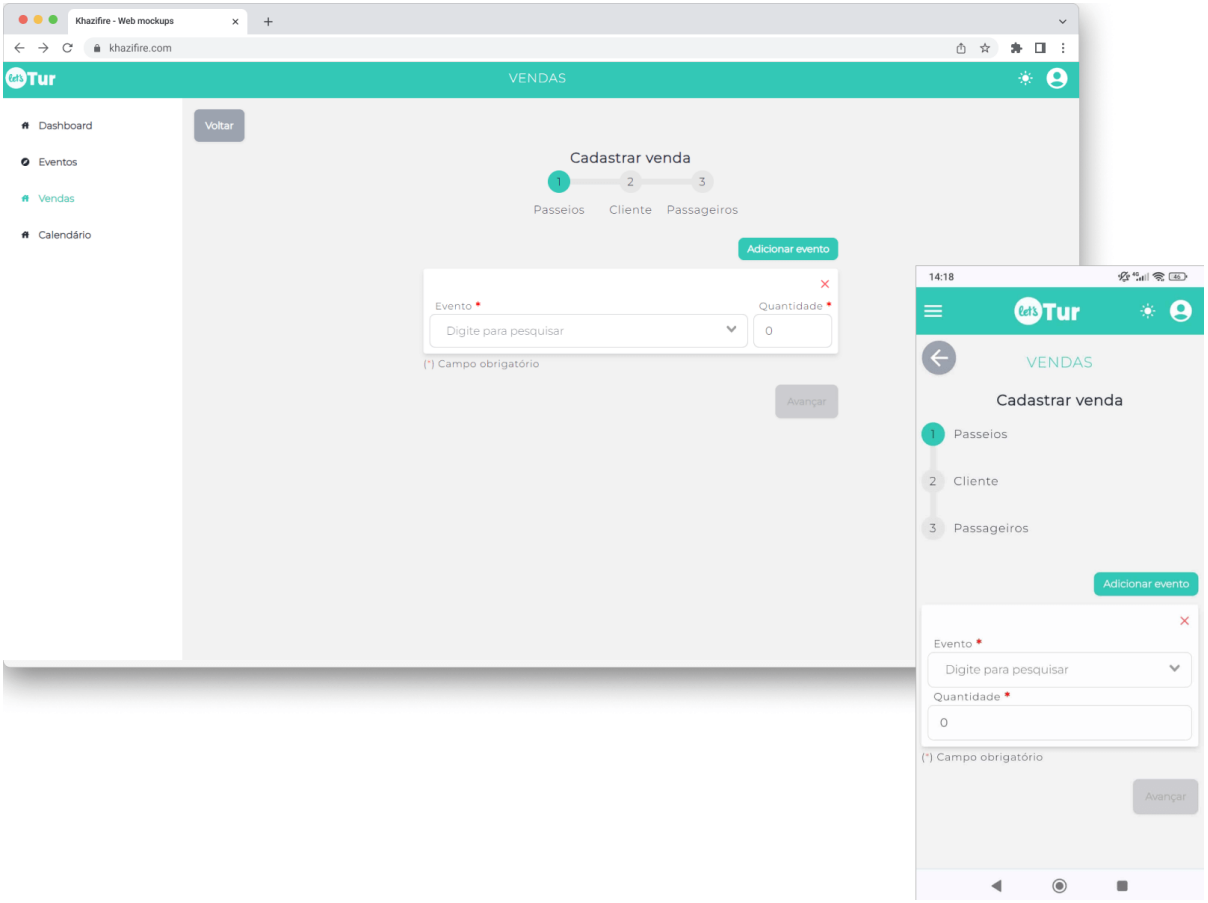
Figura 34 – Tela de listagem de vendas



Fonte: Autoria própria (2024).

O formulário de vendas sofreu uma grande refatoração, quando comparado a versão original do sistema. Agora, a venda é cadastrada por passos, sendo separada em etapas para melhor organização. A Figura 35 exibe o primeiro passo do formulário, onde os operadores registram quais eventos serão vendidos. Os demais passos possuem *layout* semelhante e são utilizados para separar diferentes informações durante o cadastro de uma venda.

Figura 35 – Etapa 1 do formulário de vendas



Fonte: Autoria própria (2024).

O sistema também apresenta outras telas responsáveis pelos cadastros auxiliares no controle de eventos. A área de passeios da aplicação, conta com três subáreas, que são as telas de passeio, tipos de passeio e locais. Um passeio sempre tem um tipo de passeio e um local pré-definido. Para o cadastro de eventos, também foram implementadas áreas de cadastros de motoristas, guias de passeios e veículos.

6 CONSIDERAÇÕES FINAIS

Este capítulo descreve as conclusões e trabalhos futuros do presente trabalho.

6.1 Conclusões

Esse trabalho propôs o desenvolvimento de uma nova versão do já existente sistema de gerenciamento de agências de passeios turísticos Letstur, tendo como finalidade a implementação de uma aplicação mais moderna e de manutenção facilitada, quando comparada a versão original. As tarefas propostas, abrangeram desde a análise da versão atual do sistema para levantamento de requisitos, até a implementação de API e interface, conforme descrito no Capítulo 1. Os resultados obtidos ao longo do desenvolvimento, podem ser visualizados no Capítulo 5.

Após comparar os objetivos do trabalho com os resultados obtidos, conclui-se que esses objetivos foram alcançados, apesar dos desafios encontrados durante o desenvolvimento da aplicação. Todos os requisitos funcionais e não-funcionais definidos previamente foram atendidos pela nova aplicação, que futuramente será utilizada pelas agências clientes do Letstur. Dentre os objetivos alcançados, destaca-se um sistema centrado no usuário, escalável, responsivo e com suas funcionalidades otimizadas. A eliminação de redundância de informações, outro ponto relevante para a nova versão, cooperou efetivamente para que o sistema seja de fácil e rápida utilização por seus usuários. A nova versão do Letstur, durante seu desenvolvimento, foi coberta de testes automatizados e amplamente documentada, pontos que facilitam a detecção de possíveis falhas ou comportamentos estranhos, agilizando manutenções futuras.

Assim, o novo Letstur representa um significativo avanço tecnológico, oferecendo uma solução confiável e adaptável para as agências de passeios turísticos, com um foco contínuo na satisfação do usuário e na eficiência operacional.

6.2 Trabalhos futuros

Apesar de terem sido alcançados todos os objetivos estabelecidos para este trabalho, é importante reconhecer que ainda há etapas fundamentais a serem realizadas para garantir o sucesso contínuo do projeto. Uma das próximas fases críticas é a implementação da aplicação em produção, permitindo que os clientes utilizem e testem a nova versão do Letstur. A coleta de *feedbacks* nesse estágio é essencial para identificar possíveis melhorias e ajustes finos antes do lançamento completo.

No âmbito tecnológico, existem ainda algumas funcionalidades que precisam ser desenvolvidas e integradas ao sistema. Uma dessas funcionalidades é a implementação de vendas de *transfers*, que são eventos específicos para transporte entre hotéis, aeroportos e locais de

passeio. Essa adição enriquecerá a oferta do Letstur, proporcionando uma solução mais abrangente para as agências de passeios turísticos e seus clientes.

Outro ponto crucial é a separação dos bancos de dados, de modo que cada agência tenha seu próprio banco de dados dedicado. Isso garantirá maior segurança e desempenho para cada agência de turismo que utiliza o Letstur. Além disso, essa separação facilitará a manutenção e o gerenciamento dos dados, permitindo uma experiência mais fluida e eficiente para todos os usuários.

Portanto, essas iniciativas são essenciais para fortalecer a nova versão do sistema Letstur, aumentar sua competitividade e garantir uma boa experiência para seus usuários.

REFERÊNCIAS

ALURA. **Git Flow: entenda o que é, como e quando utilizar**. 2023. Disponível em: https://www.alura.com.br/artigos/git-flow-o-que-e-como-quando-utilizar. Acesso em: 05 Nov. 2023.

APPMASTER. **O que é JIRA? Visão geral e guia completo**. 2022. Disponível em: https://appmaster.io/pt/blog/o-que-e-jira. Acesso em: 11 Dez. 2023.

ARAGãO, T. **SOLID — Princípios da Programação Orientada a Objetos**. 2017. Disponível em: https://medium.com/thiago-aragao/solid-princ%C3%ADpios-da-programa%C3%A7%C3%A3o-orientada-a-objetos-ba7e31d8fb25#:~:text=SOLID%20s%C3%A3o%20princ%C3%ADpios%20ou%20boas,a%20qualquer%20linguagem%20de%20programa%C3%A7%C3%A3o. Acesso em: 06 Out. 2023.

ATLASSIAN. **Saiba tudo sobre o Gitflow Workflow**. 2023. Disponível em: https://www.atlassian.com/br/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=O%20que%20%C3%A9%20Gitflow%3F,por%20Vincent%20Driessen%20no%20nvie. Acesso em: 05 Nov. 2023.

AWS. **O que significa integração contínua?** 2024. Disponível em: https://aws.amazon.com/pt/devops/continuous-integration/. Acesso em: 15 Mar. 2024.

AZURE. **O que é a entrega contínua?** 2024. Disponível em: https://learn.microsoft.com/pt-br/devops/deliver/what-is-continuous-delivery. Acesso em: 15 Mar. 2024.

BBC. **4 fatores que impedem que Brasil vire potência no turismo apesar do potencial**. 2022. Disponível em: https://www.bbc.com/portuguese/brasil-63671736. Acesso em: 08 set. 2023.

CANALTECH. **O que é Figma**. 2021. Disponível em: https://canaltech.com.br/software/o-que-e-figma/. Acesso em: 05 Out. 2023.

DAISYUI, D. **Daisy Ui**. 2024. Disponível em: https://daisyui.com/. Acesso em: 15 Mar. 2024.

DOCKER, D. **Merge Compose Files**. 2024. Disponível em: https://docs.docker.com/compose/multiple-compose-files/merge/. Acesso em: 15 Mar. 2024.

DOCKERDOCS. **Docker Compose overview**. 2023. Disponível em: https://docs.docker.com/compose/. Acesso em: 05 Out. 2023.

GITHUB. **O que é github**. 2023. Disponível em: https://ebaconline.com.br/blog/o-que-e-github. Acesso em: 05 Out. 2023.

GITHUB, D. **About workflows**. 2024. Disponível em: https://docs.github.com/en/actions/using-workflows/about-workflows. Acesso em: 15 Mar. 2024.

GUIADEV. **Documentação Técnica**. 2023. Disponível em: https://guia.dev/pt/pillars/software-architecture/technical-documentation.html. Acesso em: 11 Dez. 2023.

INSOMNIA, D. **Insomnia**. 2024. Disponível em: https://insomnia.rest/. Acesso em: 15 Mar. 2024.

LARAVEL, D. **Database: Migrations**. 2024. Disponível em: https://laravel.com/docs/11.x/migrations. Acesso em: 15 Mar. 2024.

LARAVEL, D. **Eloquent: Factories**. 2024. Disponível em: https://laravel.com/docs/11.x/eloquent-factories. Acesso em: 15 Mar. 2024.

LARAVEL, D. **Eloquent: Getting started**. 2024. Disponível em: https://laravel.com/docs/11.x/eloquent. Acesso em: 15 Mar. 2024.

LARAVEL, D. **Laravel, the PHP framework for web artisans**. 2024. Disponível em: https://laravel.com/. Acesso em: 15 Mar. 2024.

LARAVEL, D. **Testing**. 2024. Disponível em: https://laravel.com/docs/11.x/testing. Acesso em: 15 Mar. 2024.

LETSTUR. **Sistema de gerenciamento de agências turísticas**. 2014. Disponível em: http://www.lets.tur.br/. Acesso em: 07 sep. 2023.

MARQUES, R. **Como realizar o Levantamento de Requisitos no desenvolvimento de software**. 2018. Disponível em: https://www.cedrotech.com/blog/levantamento-de-requisitos-e-desenvolvimento-de-softwares/#:~:text=De%20forma%20simples%2C%20o%20levantamento,que%20o%20software%20vai%20desempenhar. Acesso em: 06 Out. 2023.

MIRO. **O que é Kanban**. 2023. Disponível em: https://miro.com/pt/agile/o-que-e-quadro-kanban/. Acesso em: 05 Nov. 2023.

NUXT. **The intuitive web framework**. 2023. Disponível em: https://nuxt.com/. Acesso em: 05 Out. 2023.

PHP, D. **PHP, hypertext preprocessor**. 2024. Disponível em: https://www.php.net/. Acesso em: 15 Mar. 2024.

POSTGRES. **O que é postgres**. 2023. Disponível em: https://www.postgresql.org/. Acesso em: 05 Out. 2023.

SEBRAE. **A importância dos atrativos turísticos do Brasil**. 2022. Disponível em: https://sebrae.com.br/sites/PortalSebrae/artigos/a-importancia-dos-atrativos-turisticos-do-brasil,4db2a30bd0f13810VgnVCM100000d701210aRCRD#:~:text=O%20turismo%20no%20Brasil%20corresponde,turistas%20venham%20visitar%20o%20pa%C3%ADs.&text=O%20turismo%20%C3%A9%20uma%20das,no%20Brasil%20n%C3%A3o%20%C3%A9%20diferente. Acesso em: 07 set. 2023.

SPATIE, D. **Laravel-permission**. 2024. Disponível em: https://spatie.be/docs/laravel-permission/v6/introduction. Acesso em: 15 Mar. 2024.

T2INFORMATIK. **What is a Taskboard?** 2023. Disponível em: https://t2informatik.de/en/smartpedia/taskboard/. Acesso em: 05 Nov. 2023.

TAILWIND. **Rapidly build modern websites without ever leaving your HTML.** 2023. Disponível em: https://tailwindcss.com/. Acesso em: 05 Out. 2023.

TRAINEECODEPLAYS. **Desenvolvimento Web - Protocolo HTTP**. 2023. Disponível em: https://www.tabnews.com.br/TraineeCodeplays/desenvolvimento-web-protocolo-http. Acesso em: 15 Mar. 2024.

VALENTE, M. T. **Domain-Driven Design (DDD): Um Resumo**. 2022. Disponível em: https://engsoftmoderna.info/artigos/ddd.html. Acesso em: 15 Mar. 2024.