

# ISPGAYA

instituto superior politécnico

Licenciatura em Engenharia Informática  
Projeto de Engenharia Informática em Contexto Empresarial  
3º ano

Pedro Manuel Maia Barbosa

[2019102346@ispgaya.pt](mailto:2019102346@ispgaya.pt)

Instituto Superior Politécnico Gaya

Licenciatura em Engenharia Informática

## **DASHBOARD STATS WEB/E-COMMERCE**

**Relatório de estágio orientado pelo Professor Doutor Fernando Luís  
Ferreira de Almeida, supervisionado por Ivo Pereira e apresentada ao  
Instituto Superior Politécnico Gaya**

Março de 2024



## **Dedicatória**

Quero dedicar este projeto à minha família, cujo apoio inabalável foi a força motriz por trás de cada passo dado nesta jornada. Desde o momento que me candidatei ao projeto vocês estiveram ao meu lado, a incentivar-me a sonhar alto, persistir diante dos desafios e alcançar o melhor de mim.

Cada momento dedicado a este projeto foi permeado pela vossa presença. Nos momentos de dúvida, foi o vosso encorajamento que me impulsionou a seguir em frente. Nas conquistas.

A vocês, que sempre acreditaram no meu potencial e me motivaram a almejar mais na vida, dedico este projeto com profunda gratidão. Que o vosso apoio contínuo seja sempre a minha inspiração para perseguir os meus sonhos e alcançar novos horizontes.

## **Agradecimentos**

Gostaria de expressar os meus sinceros agradecimentos a todas as pessoas e entidades que estiveram ao meu lado durante o meu estágio de cinco meses, um período crucial e enriquecedor na minha jornada académica.

Primeiramente, quero agradecer aos meus amigos e familiares pelo apoio constante e pelo incentivo durante este período. O vosso encorajamento e motivação foram fundamentais para me manter focado nos meus objetivos e superar os desafios que surgiram ao longo do caminho.

Agradeço também aos professores da minha faculdade, Ispgaya, pelo apoio e orientação ao longo não só do estágio, mas também do curso. Os vossos ensinamentos e conselhos foram inestimáveis para o meu desenvolvimento profissional e pessoal.

À empresa E-goi e a todos os seus colaboradores, expresso a minha profunda gratidão por proporcionarem um ambiente de trabalho tão acolhedor e estimulante. A oportunidade de fazer parte desta equipa e contribuir para os projetos foi verdadeiramente enriquecedora. Agradeço especialmente à minha equipa de integrações e aos colegas de trabalho que partilharam os seus conhecimentos e experiências comigo.

A todos vocês, o meu mais sincero obrigado por fazerem parte desta jornada e por tornarem esta experiência tão memorável e gratificante.

## Resumo

Este relatório é a prova final de avaliação da minha licenciatura em Engenharia Informática. Nele, é sumariado de forma clara e objetiva o projeto ‘Dashboards Stats Web/E-commerce’ desenvolvido durante o estágio na empresa E-goi, que consistiu na criação de uma dashboard para os clientes conseguirem analisar os dados estatísticos dos seus websites dentro da plataforma E-goi. O objetivo principal foi fornecer uma ferramenta que permitisse visualizar informações como visitas e dados de e-commerce de maneira acessível e intuitiva.

A análise de dados de visitas e vendas em um site de e-commerce é crucial para qualquer negócio que pretende ser competitivo no mercado digital. Compreender o comportamento dos visitantes no site permite identificar padrões de navegação, preferências de produtos e potenciais barreiras à conversão de vendas. Dados como a número de vistas, as páginas mais visitadas, o tempo de permanência e as taxas de rejeição ajudam a otimizar a experiência do utilizador e, consequentemente, aumentar as taxas de conversão.

A contribuição deste projeto para a área de Engenharia Informática e e-commerce é significativa, pois exemplifica a aplicação prática de técnicas de análise de dados e visualização de informação em um contexto empresarial real. Este projeto demonstra a importância da combinação de competências técnicas e analíticas na engenharia informática, além de destacar a relevância da personalização e da usabilidade na criação de soluções de software que atendem às necessidades específicas do mercado de e-commerce.

O relatório descreve o processo de desenvolvimento da dashboard, onde é detalhado o trabalho desenvolvido para alcançar esses resultados, incluindo a fase inicial de estudo, modelação e idealização, o desenvolvimento de soluções, testes e implementação, e as futuras implementações. São discutidas também a seleção de métricas relevantes, a implementação das funcionalidades necessárias, os desafios enfrentados ao longo do projeto, e as tecnologias usadas.

## **Abstract**

This report is the final assessment of my bachelor's degree in computer engineering. It clearly and objectively summarizes the 'Dashboards Stats Web/E-commerce' project developed during the internship at E-goi, which involved creating a dashboard for clients to analyse statistical data from their websites within the E-goi platform. The main objective was to provide a tool that allows visualizing information such as visits and e-commerce data in an accessible and intuitive manner.

Analysing visit and sales data on an e-commerce site is crucial for any business that aims to be competitive in the digital market. Understanding visitor behavior on the site allows identifying navigation patterns, product preferences, and potential barriers to sales conversion. Data such as the number of visits, the most visited pages, the time spent on the site, and bounce rates help optimize the user experience and, consequently, increase conversion rates.

The contribution of this project to the field of Computer Engineering and e-commerce is significant, as it exemplifies the practical application of data analysis and information visualization techniques in a real business context. This project demonstrates the importance of combining technical and analytical skills in computer engineering, as well as highlighting the relevance of customization and usability in creating software solutions that meet the specific needs of the e-commerce market.

The report describes the dashboard development process, detailing the work carried out to achieve these results, including the initial study, modelling, and ideation phase, the development of solutions, testing and implementation, and future implementations. The selection of relevant metrics, the implementation of necessary functionalities, the challenges faced throughout the project, and the technologies used are also discussed.

## **Lista de abreviaturas e siglas**

API - Application Programming Interface

CR – Code Review

JSON - JavaScript Object Notation

KPI - Key Performance Indicator

LD - Lead Developer Integrations

MVC - Model View Controller

PM - Product Manager

PS - Pedido de Suporte

QA - Quality Assurance

SRE - Site Reliability Engineering

UML - Unified Modeling Language

## Índice

---

Dedicatória.....	iii
Agradecimentos .....	iv
Resumo .....	v
Abstract.....	vi
Lista de abreviaturas e siglas .....	vii
1 Introdução .....	1
1.1 Contextualização do Relatório.....	1
1.2 A empresa .....	1
1.3 Breve Descrição do Projeto .....	1
1.4 Estrutura do Relatório.....	1
2 Contextualização .....	3
2.1 Justificação e contexto de aplicação do tema do projeto na área da Engenharia Informática .....	3
2.2 Justificação da Escolha do Projeto e Intenções Futuras .....	4
2.3 Caracterização da entidade de estágio .....	5
2.4 Metodologia.....	8
2.5 Contexto do Estágio .....	11
2.6 Projeto similares e associados .....	12
2.7 Bases teóricas técnicas.....	16
3 Modelação e desenvolvimento do projeto .....	20
3.1 Requisitos .....	20
3.2 Análise e descrição da arquitetura do sistema. ....	26
3.3 Diagrama de fluxo de utilização .....	28
3.4 Diagrama de fluxo de sistema .....	29
3.5 Diagrama de sequência.....	31
3.6 Atividades realizadas .....	33



3.7	Tomadas de decisão.....	57
4	Testes de validação.....	60
5	Cronograma.....	62
5.1	Gant Previsão.....	62
5.2	Gantt do Projeto.....	63
5.3	Comparação entre os Mapas de Gantt.....	64
6	Meios previstos e meios necessários.....	67
6.1	Meios humanos.....	67
6.2	Material fornecido pela empresa.....	67
6.3	Tecnologias utilizadas.....	68
7	Conclusões.....	70
7.1	Análise de resultados.....	71
8	Referências bibliográficas.....	72
8.1	Bibliografia Consultada.....	72
8.2	Bibliografia Citada.....	73
9	Glossário.....	74
10	Anexos.....	76
10.1	Tabelas de descrição detalhada de casos de uso.....	76
10.2	Tabelas de testes.....	79

## Índice de figuras

Figura 1 Logo E-goi.....	6
Figura 2 Estrutura Organizacional.....	6
Figura 3 Departamento de Produto.....	7
Figura 4 Estrutura equipa de Integrações .....	7
Figura 5 Pós desenvolvimento.....	11
Figura 6 Interface Matomo .....	12
Figura 7 Interface Google Analytics.....	13
Figura 8 Estrutura MVC .....	18
Figura 9 Casos de uso .....	20
Figura 10 Diagrama de Sequência Caso de Uso - Editar Dashboard .....	21
Figura 11 Diagrama de Sequência Caso de Uso - Visualizar dados Visitas.....	22
Figura 12 Diagrama de Sequência Caso de Uso - Visualizar dados E-commerce .....	23
Figura 13 Diagrama de Sequência Caso de Uso - Editar Métricas Apresentadas .....	23
Figura 14 Diagrama de Sequência Caso de Uso - Editar como os dados são apresentados.....	24
Figura 15 Diagrama de componentes .....	26
Figura 16 Diagrama de fluxo de utilização.....	28
Figura 17 Diagrama de fluxo de sistema .....	29
Figura 18 Diagrama de sequência.....	31
Figura 19 Aba de dashboards .....	33
Figura 20 Interface de criação de dashboard .....	34
Figura 21 Interface de adição de widget.....	34
Figura 22 Interface de estrutura de dashboard.....	34
Figura 23 VisitsSummary resposta exemplo .....	36
Figura 24 UserCountry resposta exemplo .....	37
Figura 25 Actions.getPageUrls resposta exemplo .....	39
Figura 26 Goals.getItemsName resposta exemplo .....	40
Figura 27 Código Verificação de dados vazios .....	41
Figura 28 Código estrutura do array de retorno.....	42
Figura 29 Código de iteração sobre os dados recebidos.....	42

Figura 30 Ficheiros dos widgets .....	45
Figura 31 Relação com o modelo MVC .....	45
Figura 32 Template editor do widget de visitas.....	46
Figura 33 Mapa exemplo do AmCharts.....	48
Figura 34 Gráfico circular exemplo do AmCharts .....	49
Figura 35 Gráfico circular com 100 setores .....	50
Figura 36 Gráfico circular com 10 setores .....	52
Figura 37 Gráfico de linhas .....	53
Figura 38 Gráfico circular .....	53
Figura 39 Mapa com visitas.....	54
Figura 40 Total das métricas.....	54
Figura 41 Tabela das páginas com visitas .....	55
Figura 42 Interface de traduções.....	56
Figura 43 Mapa de Gantt previsto .....	62
Figura 44 Mapa de Gantt do projeto .....	63

## Índice de tabelas

---

Tabela 1 Matomo vs Google Analytics .....	15
Tabela 2 Requisitos não funcionais .....	25
Tabela 3 Códigos ISO 3166-1 .....	48
Tabela 4 Cor #0084C2.....	51
Tabela 5 Cor #E0F5FB .....	51
Tabela 6 Gradiente de cores .....	51
Tabela 7 RF01 Caso de uso 'Editar dashboard'.....	76
Tabela 8 RF02 Caso de uso 'Analisar Web Stats' .....	76
Tabela 9 RF03 Caso de uso 'Analisar E-commerce' .....	77
Tabela 10 RF04 Caso de uso 'Editar modos de visualização' .....	77
Tabela 11 RF05 Caso de uso 'Editar métricas apresentadas' .....	78
Tabela 12 Teste RF01 .....	79
Tabela 13 Teste RF02 .....	79
Tabela 14 Teste RF03 .....	79
Tabela 15 Teste RF04 .....	80
Tabela 16 Teste RF05 .....	80
Tabela 17 Teste RF06 .....	80
Tabela 18 Teste RF07 .....	81
Tabela 19 Teste RF08 .....	81
Tabela 20 Teste RF09 .....	81
Tabela 21 Teste RF10 .....	82
Tabela 22 Teste RF11 .....	82
Tabela 23 Teste RF12.....	82

## 1 Introdução

### 1.1 Contextualização do Relatório

O presente relatório constitui o culminar de um projeto desenvolvido no âmbito da unidade curricular de projeto do curso de Engenharia Informática, com enfoque na criação de uma dashboard para apresentação de dados estatísticos dos sites dos clientes da empresa E-goi. Esta iniciativa surgiu da necessidade de fornecer aos utilizadores uma ferramenta eficaz para acompanhar o desempenho dos seus sites, desde métricas de visitas até resultados de e-commerce.

### 1.2 A empresa

A E-goi, uma empresa de renome no setor de marketing digital e destaca-se não apenas pela sua abordagem inovadora, mas também pelo compromisso constante com a excelência em tecnologia e atendimento ao cliente. Com uma ampla gama de serviços e soluções adaptáveis a empresas de todos os portes, a E-goi posiciona-se como uma parceira estratégica essencial para o sucesso das iniciativas digitais dos seus clientes.

No contexto deste projeto, o objetivo principal foi implementar uma solução que permitisse visualizar e interpretar os dados de forma intuitiva dentro da plataforma que a E-goi oferece aos seus clientes, contribuindo para uma compreensão mais profunda do comportamento dos sites dos seus clientes.

### 1.3 Breve Descrição do Projeto

O cerne deste projeto reside na criação de uma dashboard de fácil utilização e centralizado na plataforma E-goi, o qual oferecerá aos clientes da E-goi uma plataforma intuitiva para acesso rápido aos dados estatísticos fundamentais para as suas operações online. Por meio deste dashboard, os clientes terão a capacidade de analisar o desempenho dos websites, acompanhar métricas essenciais de e-commerce, compreender o comportamento dos utilizadores e, assim, melhorar as suas decisões para melhorar a sua presença digital e alavancar os resultados dos seus negócios.

### 1.4 Estrutura do Relatório

A estrutura do relatório será organizada da seguinte forma: Na primeira parte do documento, serão explorados os principais conceitos relacionados ao estudo, bem como a empresa e soluções existentes relevantes na área. Na segunda parte, será detalhada a metodologia adotada para o desenvolvimento do projeto, juntamente com os principais

resultados obtidos. Na terceira parte, serão analisados e discutidos os dados recolhidos durante a execução do projeto. Por fim, serão apresentadas as conclusões do estudo, juntamente com as principais referências bibliográficas utilizadas.

## 2 Contextualização

### 2.1 Justificação e contexto de aplicação do tema do projeto na área da Engenharia Informática

A Engenharia Informática abrange uma vasta gama de áreas de aplicação, desde o desenvolvimento de software até à análise de dados. Neste contexto, o projeto de criação de widgets para visualização de dados de visitas e e-commerce desempenha um papel crucial na integração de informações importantes para os clientes da E-goi.

A justificação para este projeto reside na crescente importância da análise de dados para orientar as estratégias de negócio. Com a explosão de informações disponíveis online, as empresas enfrentam o desafio de extrair insights significativos dos dados brutos. Os widgets criados neste projeto oferecem uma solução elegante para esse desafio, proporcionando aos clientes da E-goi visualizar facilmente métricas-chave relacionadas com visitas e atividades de e-commerce em tempo real.

Assim, a aplicação deste tema na Engenharia Informática reflete a resposta a uma necessidade atual, bem como a possibilidade de expansão e inovação dentro do campo da tecnologia, fornecendo soluções práticas e funcionais que podem evoluir.

## 2.2 Justificação da Escolha do Projeto e Intenções Futuras

Optar por este projeto de estágio revela-se como uma decisão estratégica alinhada com os meus interesses e aspirações profissionais, especialmente considerando a minha intenção de prosseguir estudos no Mestrado em Engenharia Informática - Engenharia de Dados. Esta escolha é fundamentada em diversos motivos que se entrelaçam de forma coerente e promissora para o meu percurso académico e profissional.

Em primeiro lugar, a natureza do projeto, focada na análise e manipulação de dados para a criação de uma dashboard, está intrinsecamente ligada ao campo da Engenharia de Dados. Ao trabalhar com dados reais provenientes dos websites dos clientes da empresa E-goí, terei a oportunidade de desenvolver competências práticas em recolha, limpeza, análise e visualização de dados - competências fundamentais para qualquer profissional que queira destacar-se no campo da engenharia de dados.

Para além disso, a realização deste projeto proporciona uma imersão valiosa no universo da análise de dados num contexto empresarial, onde fui desafiado a compreender as necessidades dos clientes, identificar padrões nos dados e propor soluções eficazes para apresentação e interpretação dessas informações. Esta experiência prática foi extremamente enriquecedora para o meu desenvolvimento profissional e deu-me uma base sólida de conhecimento para explorar de forma mais aprofundada durante o mestrado.

Ao ingressar no Mestrado em Engenharia Informática - Engenharia de Dados, espero poder aproveitar plenamente os conhecimentos e competências adquiridos durante o estágio para aprofundar o meu entendimento teórico sobre as metodologias, técnicas e ferramentas empregadas na análise e gestão de grandes volumes de dados. O projeto de estágio serviu como uma fonte rica de experiência prática e estudos, permitindo-me contextualizar e aplicar os conceitos teóricos abordados ao longo da licenciatura.

Portanto, a escolha deste projeto não apenas satisfaz as exigências imediatas do estágio, mas também se alinha de forma estratégica com os meus objetivos de carreira a longo prazo, pois oferece uma plataforma sólida para a minha especialização e crescimento profissional no campo da Engenharia de Dados.



## 2.3 Caracterização da entidade de estágio

A E-goi é uma empresa de tecnologia sediada em Portugal, especializada em soluções de marketing digital e automação de marketing. Fundada em 2004, a E-goi tem sido reconhecida como uma das principais empresas do setor, oferece uma ampla gama de serviços para ajudar os clientes a alcançar os seus objetivos de marketing.

Em 2023, a E-goi foi classificada como a 20ª melhor empresa para se trabalhar em Portugal, destacando-se pelo ambiente de trabalho colaborativo, oportunidades de crescimento profissionais e cultura de inovação. Além disso, a E-goi é reconhecida pela sua abordagem centrada no cliente e compromisso com a excelência em todos os aspetos do negócio. Com um forte foco na qualidade, fiabilidade e satisfação do cliente, a E-goi mantém-se como líder de mercado no setor, oferecendo resultados excecionais e impulsionando o sucesso aos seus clientes.

Além de ser reconhecida como uma excelente empresa para se trabalhar, a E-goi disponibiliza aos seus clientes uma plataforma completa de marketing digital. Esta plataforma integra diversas ferramentas e funcionalidades poderosas, tais como automação de e-mail, SMS, notificações push, automação de marketing, CRM e análise de dados. Com esta plataforma, os clientes da E-goi têm acesso a uma solução abrangente para gerir e otimizar as suas campanhas de marketing, de forma a melhorar o envolvimento do cliente, aumentar as conversões e impulsionar o crescimento do negócio. Além disso, a E-goi oferece suporte técnico dedicado e recursos de formação para garantir que os clientes tirem o máximo partido da plataforma, proporcionando uma experiência excecional e resultados mensuráveis. Com a plataforma da E-goi, os clientes beneficiam de uma abordagem integrada e eficaz para as suas estratégias de marketing digital, ajudando-os a alcançar os seus objetivos com maior eficiência e sucesso. Esta plataforma é bastante complexa e encontra-se, até à data, em constante desenvolvimento e evolução. Para além disso, durante o processo de desenvolvimento de todas as componentes que constituem o E-goi são utilizadas várias tecnologias, tais como Java, Python, Angular, HTML, CSS, JavaScript, PHP, entre outras.



Figura 1 Logo E-goi

### 2.3.1 Estrutura Organizacional

A E-goi está dividida em vários departamentos, cada um com seu próprio conjunto de procedimentos e deveres, assim como várias empresas. A visão geral dos diversos departamentos que compõem a E-goi é ilustrada na Figura 2. Cada departamento dentro da empresa tem um impacto no desenvolvimento do produto, seja diretamente ou indiretamente.

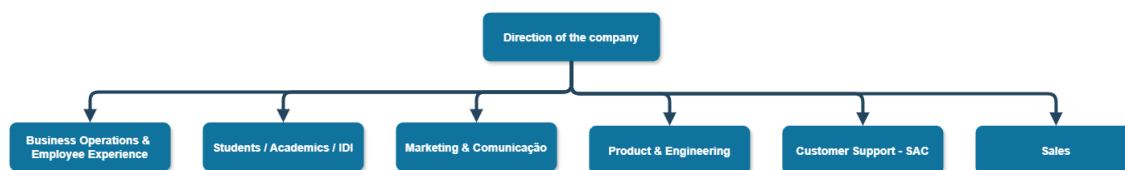


Figura 2 Estrutura Organizacional

O estágio foi realizado no departamento de *Product & Engineering*. O departamento de *Product & Engineering*, está principalmente focado em criar produtos que ofereçam soluções aos utilizadores para os seus problemas, proporcionem valor aos seus negócios e ajudem a alcançar os seus objetivos. Este departamento possui três camadas. Cada camada é composta por várias equipas de produto que partilham um objetivo comum, colaboram entre si e concentram-se em KPI's e objetivos comparáveis. Por outro lado, uma equipa multidisciplinar responsável por bens ou componentes do ecossistema específicos é conhecida como equipa de produto. Têm um gestor que estabelece os objetivos, planos e operações diárias para cada um deles. São independentes, auto-organizadas e autogeridas. Uma visão geral da estrutura do departamento de produto é apresentada na figura 3:

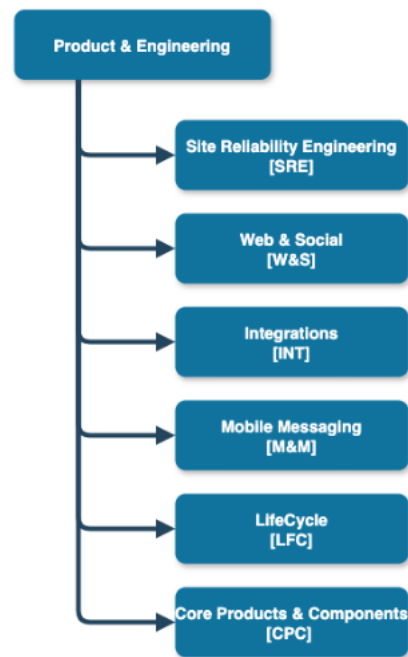


Figura 3 Departamento de Produto

Dentro do departamento de *Product & Engineering*, o estágio foi especificamente na equipa de integrações. Esta equipa é responsável por desenvolver e manter as integrações entre a plataforma da E-goi e outras ferramentas e serviços utilizados pelos clientes da empresa. Alguns exemplos dessas plataformas são: Facebook, Magento, WordPress, Vtex, Prestashop 8, etc...

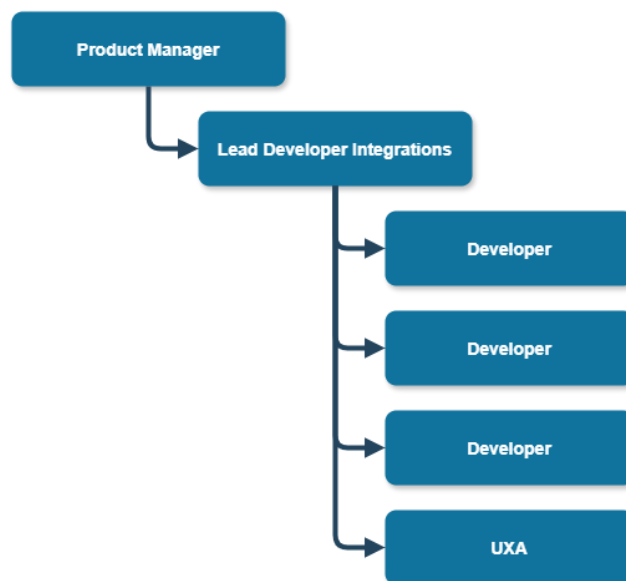


Figura 4 Estrutura equipa de Integrações

## 2.4 Metodologia

A metodologia utilizada neste projeto foi cuidadosamente planeada e executada para garantir eficiência e qualidade em todas as etapas do desenvolvimento. Antes de iniciar qualquer atividade de desenvolvimento, foi realizada uma fase de estudo aprofundado sobre os requisitos e objetivos do projeto.

Uma parte fundamental foi a realização de reuniões semanais, agendadas para todas as segundas-feiras de manhã, com toda a equipa de integração. Nestas reuniões, eram discutidos os progressos realizados na semana anterior, bem como o planeamento das atividades para a semana em curso. O objetivo principal destas reuniões era manter toda a equipa informada sobre o estado dos projetos em curso, o que permitia que o líder da equipa fosse informado sobre qualquer problema ou obstáculo encontrado, e se necessário solicitar assistência aos membros da equipa, caso necessário.

Além das reuniões semanais, foram realizadas reuniões de acompanhamento a cada duas ou três semanas com o orientador de estágio. Estas reuniões tinham como objetivo principal verificar se tudo estava a correr conforme o planeado e identificar eventuais problemas ou dificuldades encontradas durante o desenvolvimento do projeto. O orientador de estágio também prestava orientação e apoio em termos de gestão de tempo e definição de prioridades, desde a integração na empresa até à conclusão do relatório final.

É importante destacar que, além das reuniões formais, existia uma cultura de abertura para tirar dúvidas a qualquer momento do dia, não só com os membros da equipa de integrações, mas também com outras equipas da empresa. Esta comunicação aberta e contínua contribuiu para um ambiente de trabalho colaborativo e produtivo.

Cada etapa foi executada em colaboração com o líder da equipa. Primeiramente, definiam-se os objetivos específicos de cada fase, em linha com as necessidades do projeto. Em seguida, realizava-se uma análise detalhada dos requisitos, com discussões frequentes com o PM para garantir um alinhamento claro com as metas estabelecidas.

Após a definição dos objetivos e requisitos, iniciava-se o estudo das tecnologias necessárias para implementar as funcionalidades desejadas. Muitas vezes, essa pesquisa era conduzida com base em fontes recomendadas pelos colegas de equipa, que partilhavam experiências e conhecimentos relevantes.

Além disso, sempre que possível, foi analisado o código existente na empresa com o objetivo de encontrar soluções semelhantes que pudessem ser adaptadas para o projeto em questão. Desta forma manteve-se uma estrutura consistente e alinhada com os padrões de desenvolvimento da empresa, contribuindo para uma integração sem percalços com os sistemas existentes na plataforma E-goi.

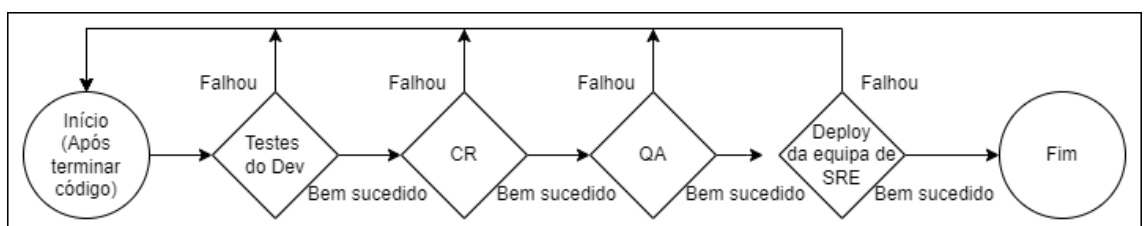
Somente após uma análise cuidadosa e uma compreensão completa dos requisitos e tecnologias envolvidas é que se avançava para a fase de desenvolvimento, garantindo assim uma abordagem sólida e orientada para os resultados em todas as etapas do projeto.

Além disso, como parte da metodologia da empresa, antes de qualquer código ser lançado em produção para garantir a qualidade e integridade do projeto são seguidos os seguintes passos:

1. Testes pelo Desenvolvedor (DEV): Inicialmente, o próprio desenvolvedor (no caso, eu) realiza uma série de testes para verificar se todas as funcionalidades estão operacionais e se o código atende aos requisitos estabelecidos.
2. Inserir no gitlab o as alterações através dos seguintes comandos:
3. `git push "nome_da_minha_branch"`: Este comando é usado para enviar as alterações realizadas na branch local para a branch remota correspondente.
  - a. `git fetch`: Este comando permite visualizar as alterações feitas nos arquivos locais.
  - b. `git checkout "enviroment"`: Este comando permite mudar para a branch especificada, neste caso, “enviroment”. Atualiza os arquivos no diretório de trabalho para corresponder à versão armazenada nessa branch.
  - c. `git pull origin "enviroment"`: Este comando é usado para atualizar a branch local com as alterações da branch remota indicada. No caso, puxa as alterações da branch “enviroment” do repositório “origin”.
  - d. `git merge fix/BB-"numero_issue/bug"`: Este comando é utilizado para incorporar as alterações da branch

especificada (neste caso, fix/BB-“número\_issue/bug”) na branch atual (se existirem conflitos entre as branches, será necessário resolvê-los manualmente).

- e. `git push origin "enviroment":` Semelhante ao primeiro comando, este comando é usado para enviar as alterações que foram feitas na branch local (que agora inclui as alterações mescladas da outra branch) para a branch remota especificada.
4. **Code Review por *Lead Developer*:** Após o passo anterior as alterações já são acessíveis então essa informação é encaminhada para o LD. O código é submetido a uma revisão de código pelo LD. Ele analisa o código com o objetivo de encontrar possíveis problemas de lógica, performance ou boas práticas de programação. Se algum problema for encontrado, o código é novamente encaminhado para o desenvolvedor, dependendo da natureza das alterações necessárias.
5. **Testes de Qualidade (QA) por UXA (*User Experience Analyst*):** Após o code review é encaminhado para o UXA, responsável pela realização dos testes de qualidade. Ele verifica minuciosamente todas as funcionalidades para garantir que o software atenda aos padrões de qualidade definidos. Caso algum problema seja identificado, o código é devolvido ao desenvolvedor com as devidas indicações de correção.
6. ***Release/Deploy* pela equipa de SRE:** Após a aprovação do code review, o código é encaminhado para a equipa de SRE para realizar o processo de *release* e *deploy*. Essa equipa garante a integração do código no projeto e realiza o lançamento do software, garantindo a sua disponibilidade e estabilidade.



#### Figura 5 Pós desenvolvimento

Esta abordagem metodológica permitiu que apenas código de alta qualidade e funcionalidade seja implementado nos ambientes de produção e garantiu que o projeto fosse concluído dentro dos prazos estabelecidos e com os padrões de qualidade desejados.

## 2.5 Contexto do Estágio

O estágio na empresa E-goi surge no contexto da evolução contínua da sua plataforma, visando fornecer aos seus clientes ferramentas ainda mais poderosas e abrangentes para a análise de dados dos seus websites. Como uma empresa líder no setor de marketing digital, a E-goi reconhece a importância fundamental da análise de dados para o sucesso das estratégias online dos seus clientes.

Com a funcionalidade existente de *Track & Engage*, que regista e armazena uma ampla gama de dados dos websites dos clientes, desde visitas até resultados de e-commerce, a E-goi já oferecia uma base sólida para análise. No entanto, reconhecendo a crescente demanda por informações detalhadas e úteis para otimização e tomada de decisões, a empresa decidiu expandir ainda mais essa capacidade.

Assim, o estágio foi concebido como parte integrante deste processo de evolução, com o objetivo principal de desenvolver um dashboard avançado e intuitivo que permitisse aos clientes da E-goi aceder e interpretar facilmente os dados estatísticos dos seus websites.

## 2.6 Projeto similares e associados

### 2.6.1 Matomo

Durante a fase de investigação, foi analisado o dashboard do Matomo. O Matomo atua como um serviço terceiro à E-goi, e fornece o serviço de tracking e armazenamento de dados. Esta plataforma oferece uma interface para visualizar dados de tracking de sites, que permite que os utilizadores analisem o desempenho dos seus sites. Porém sendo um serviço terceiro a E-goi não fornece acesso a esses dashboard já que é em uma plataforma exterior.

No entanto, a E-goi aspira a ir além, visando criar uma solução mais completa e centralizada na sua própria plataforma. O objetivo é integrar os dados da API do Matomo num dashboard ainda mais intuitivo e completo, diretamente dentro da sua própria plataforma. Dessa forma, os clientes da E-goi terão acesso a uma análise de dados mais abrangente e personalizada, adaptada às suas necessidades específicas.

A integração dos dados vindos da API Matomo no dashboard da E-goi visa fornecer aos utilizadores uma experiência de análise de dados mais eficiente e intuitiva, permitindo-lhes extrair insights valiosos para melhorar o desempenho dos seus websites e campanhas de marketing. Este aprimoramento na capacidade analítica é fundamental para manter a competitividade no mercado digital em constante evolução.

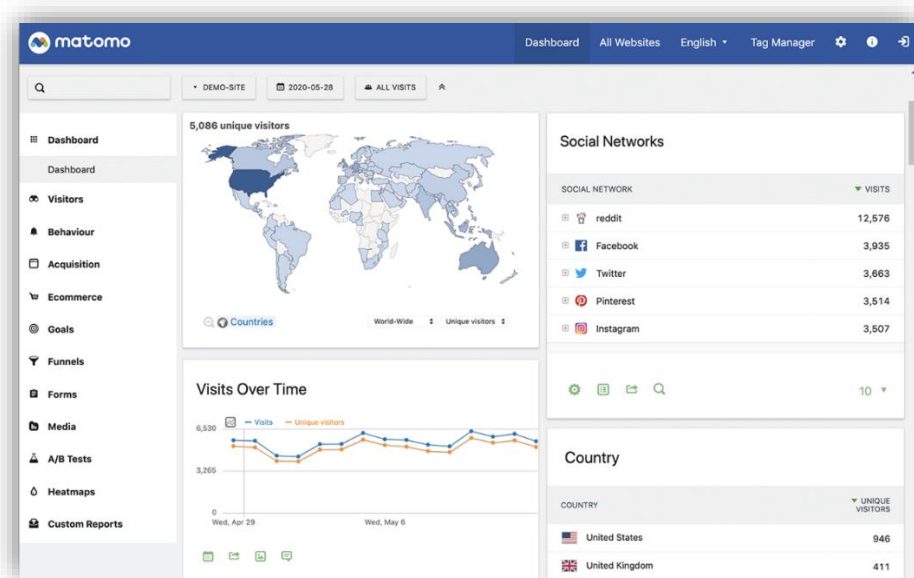


Figura 6 Interface Matomo



## 2.6.2 Google Analytics

Com Google Analytics, é possível analisar o envolvimento total dos clientes com o negócio. Permite ainda que ofereça melhores experiências e alcance resultados mais eficazes através da medição centrada no cliente em sites e aplicações. Além disso, fornece insights inteligentes para melhorar o retorno do investimento (ROI) através do uso aprendizagem automática da Google para extrair insights preditivos de dados, como prever quais utilizadores estão mais propensos a comprar ou abandonar os serviços. Com relatórios partilháveis e uma interface fácil de usar, dando a possibilidade de analisar dados e partilhar rapidamente.

"to understand the customer journey and improve marketing ROI." (Google Analytics, 04/2024.).

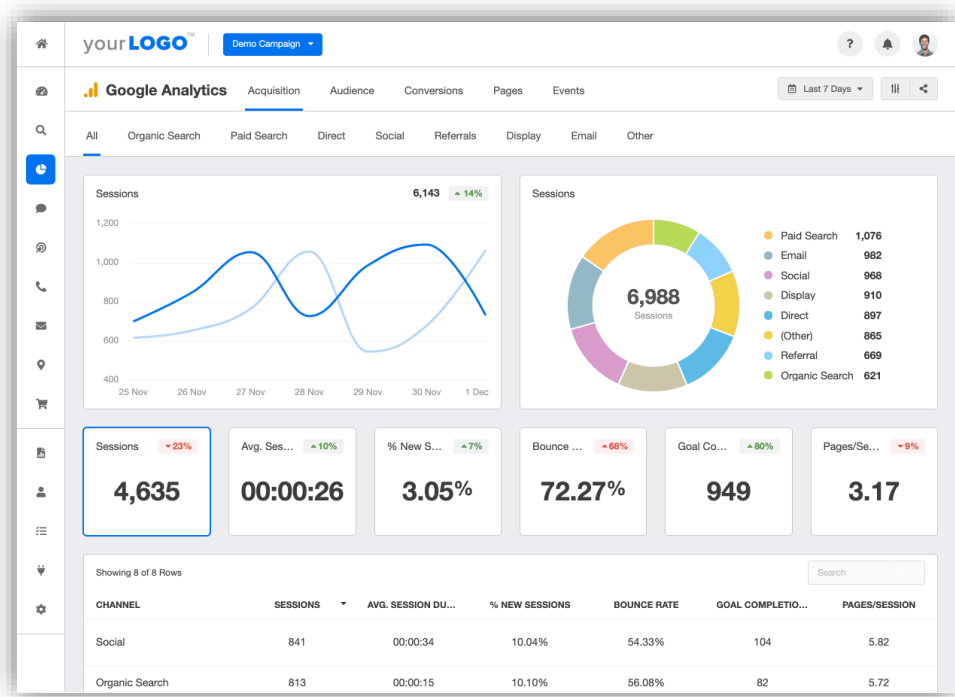


Figura 7 Interface Google Analytics

O Google Analytics assim como o matomo é uma plataforma de análise estatísticas web (ou sejam, são concorrentes), apesar deste ser mais conhecido por ser da enorme empresa google tem duas características que são consideradas desvantagens:

- Custos: Embora o Google Analytics ofereça uma versão gratuita, ela tem limitações em termos de número de visualizações de página e recursos avançados, o que pode ser um problema para empresas com alto tráfego ou necessidades específicas de análise.
- Questões de privacidade: Como o Google Analytics é hospedado pela Google, há preocupações com privacidade e segurança dos dados, especialmente considerando as políticas de privacidade em constante mudança da Google

No entanto esta plataforma é também conhecida pelas seguintes vantagens:

- Acessibilidade: O Google Analytics é conhecido pela sua interface intuitiva e fácil de usar.
- Integração com outras ferramentas do Google: Como parte do ecossistema do Google, o Google Analytics se integra perfeitamente com outras ferramentas, como Google Ads e Google Data Studio, o que facilita o acompanhamento e a análise de desempenho de campanhas de marketing.

### 2.6.3 Quadro teórico

Tabela 1 Matomo vs Google Analytics

Característica	Matomo	Google Analytics
<b>Propriedade dos dados</b>	Dados armazenados nos servidores do Google	Dados armazenados no próprio servidor do utilizador ( <i>self-hosted</i> ) ou na nuvem (Matomo Cloud)
<b>Privacidade</b>	Depende das políticas do Google, com potencial compartilhamento de dados	Controlo total sobre os dados, maior foco na privacidade do utilizador
<b>Open source</b>	Não	Sim
<b>Custo</b>	Gratuito para a maioria dos utilizadores, mas há uma versão paga	Versão <i>self-hosted</i> gratuita, versão na nuvem paga
<b>Integrações</b>	Ampla gama de integrações, especialmente com produtos do Google	Integrações disponíveis, mas geralmente através de plugins ou customizações
<b>Facilidade de Implementação</b>	Fácil de implementar, com um snippet de código	Requer mais configuração, especialmente na versão self-hosted
<b>Customização</b>	Menos flexível em termos de customização de relatórios e interface	Altamente customizável, especialmente com plugins
<b>Suporte e Comunidade</b>	Suporte oficial do Google, ampla comunidade e recursos online	Comunidade ativa de código aberto, suporte pago disponível para Matomo Cloud

A escolha entre estas plataformas vai depender das necessidades específicas de cada empresa.

## 2.7 Bases teóricas técnicas

A secção de bases teóricas deste relatório desempenha um papel fundamental ao estabelecer os fundamentos conceituais e metodológicos necessários para a compreensão e desenvolvimento do projeto 'Dashboards Stats Web/E-commerce'. Este capítulo serve como alicerce intelectual sobre o qual todo o trabalho prático é construído, proporcionando uma visão abrangente das teorias, princípios e abordagens relevantes ao contexto do projeto, além disso, este capítulo estabelece conexões entre teoria e prática, fornecendo uma base sólida para a implementação e desenvolvimento da dashboard proposta.

### 2.7.1 Model View Controller

Arquitetura de software que separa a aplicação em três componentes principais: Model (gestão de dados), View (interface do utilizador) e Controller (lógica de negócio).

"This 'separation of concerns' provides for a better division of labor and improved maintenance. Some other design patterns are based on MVC" (Mozilla, n.d.).

Model (Modelo):

O modelo representa os dados e a lógica de negócios da aplicação. É responsável por lidar com o estado da aplicação e realizar operações relacionadas aos dados, como leitura e escrita no banco de dados ou manipulação de arquivos. Por exemplo, numa aplicação de gestão de tarefas, o modelo poderia incluir classes que representam as tarefas, os utilizadores, e métodos para realizar operações como adicionar, remover ou atualizar tarefas.

View (Vista):

A vista é a interface do utilizador com a qual os utilizadores finais interagem. Apresenta os dados do modelo ao utilizador de uma forma compreensível e fornece os meios para os utilizadores interagirem com a aplicação. A vista não deve conter lógica de negócios, seu único propósito é mostrar a informação de maneira adequada para o utilizador. Por exemplo, numa aplicação web, a vista poderia ser uma página HTML que mostra a lista de tarefas e fornece botões para adicionar novas tarefas.

### Controller (Controlador):

O controlador age como intermediário entre o modelo e a vista. Responde às ações do utilizador, interpreta as solicitações da vista e realiza as operações correspondentes no modelo. Também é responsável por atualizar a vista quando o modelo muda. Por exemplo, numa aplicação web, o controlador seria um script ou uma classe que recebe os pedidos HTTP do browser, que processa os dados e atualiza o modelo conforme necessário, e depois determina qual vista deve mostrar ao utilizador em resposta ao pedido.

O modelo MVC oferece várias vantagens que o tornam uma escolha popular para o desenvolvimento de software:

- **Separação de preocupações:** O MVC divide a aplicação em três componentes distintos, cada um com responsabilidades específicas. Isso facilita a manutenção e a evolução do código, pois as alterações em um componente geralmente têm menos impacto nos outros.
- **Reutilização de código:** Como a lógica de negócios está separada do código responsável pela apresentação e pela interação do utilizador, é mais fácil reutilizar o código em diferentes partes da aplicação ou em projetos futuros.
- **Testes:** A separação clara entre o modelo, a vista e o controlador torna mais fácil testar cada componente isoladamente. Isso facilita a criação de testes automatizados para garantir que a aplicação funcione conforme o esperado.
- **Desenvolvimento paralelo:** Equipas de desenvolvimento podem trabalhar em paralelo em diferentes aspetos da aplicação, já que cada componente do MVC pode ser desenvolvido independentemente.
- **Flexibilidade:** O MVC permite diferentes tecnologias para cada componente, conforme necessário.

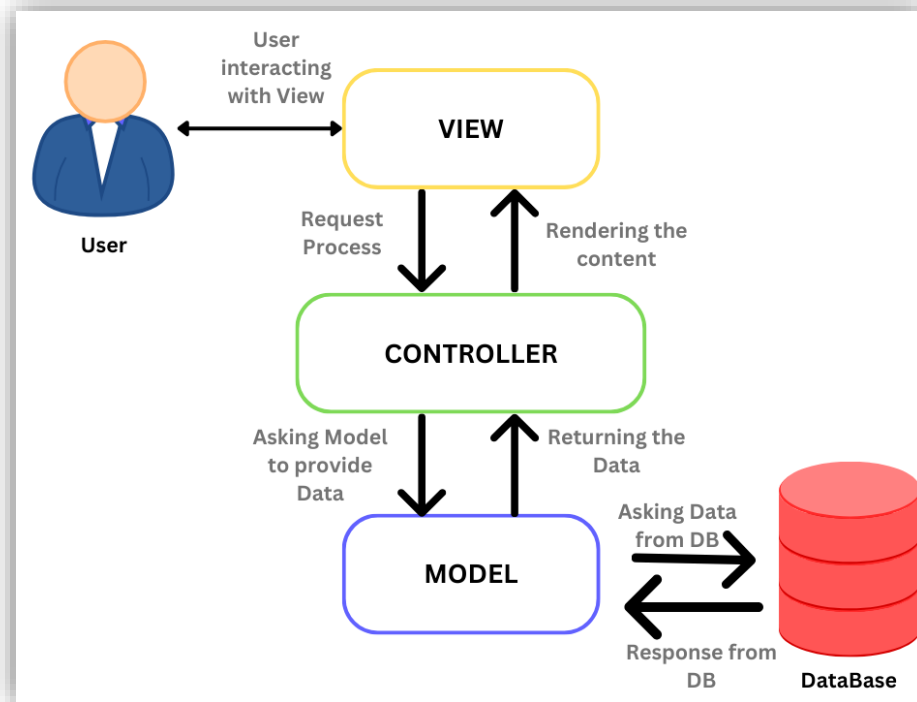


Figura 8 Estrutura MVC

Em resumo, o MVC divide uma aplicação em três componentes distintos para separar as preocupações e tornar o código mais modular, sustentável e escalável. Isso facilita o desenvolvimento e a evolução da aplicação, já que cada componente pode ser modificado ou substituído independentemente.

### 2.7.2 O que é uma API?

API é a sigla correspondente a *Application Programming Interface* (Interface de Programação de Aplicação). As APIs são interfaces que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos.

“A interface pode ser pensada como um contrato de serviço entre duas aplicações. Esse contrato define como as duas se comunicam usando solicitações e respostas.”

Amazon Web Services. (04/24). O que é uma API? Recuperado de <https://aws.amazon.com/pt/what-is/api/>

### 2.7.3 O que é um widget?

Um widget é um componente interativo e modular que pode ser incorporado em páginas de um website para fornecer funcionalidades específicas ou exibir informações relevantes. Os widgets oferecem uma forma eficaz de melhorar a experiência do utilizador ao permitir o acesso rápido a conteúdos ou serviços sem a necessidade de navegar para além da página atual. Estes elementos podem assumir diversas formas e funções, desde simples botões de partilha em redes sociais até pequenas aplicações completas que exibem dados dinâmicos em tempo real. Os widgets são altamente personalizáveis e podem ser adaptados para se integrarem harmoniosamente no design de uma página.

Em suma, os widgets web são ferramentas flexíveis e versáteis que ajudam a enriquecer o conteúdo e a funcionalidade das páginas da internet, oferecendo aos utilizadores uma experiência mais interativa e conveniente.

"O padrão MVC é fundamental para a organização de aplicações web, separando a lógica de apresentação da lógica de negócios e facilitando a manutenção e escalabilidade do software" (Buschmann, Henney, & Schmidt, 1996).

### 2.7.4 RESTful

RESTful é um tipo de arquitetura de software que são baseados em REST (Representational State Transfer). O REST define um conjunto de funções como GET, PUT, DELETE e assim por diante para que os utilizadores consigam aceder a dados do servidor através de HTTP requests. Os métodos HTTP são então usados para realizar operações sobre esses recursos. Por exemplo, o método GET é utilizado para obter dados de um recurso, o método PUT para atualizar um recurso existente e o método DELETE para remover um recurso.

Esta abordagem baseada em recursos e operações padronizadas torna os sistemas RESTful altamente flexíveis, escaláveis e fáceis de entender e manter. Além disso, permite a construção de APIs web robustas e eficientes, que podem ser facilmente utilizadas.

"A criação de APIs RESTful é essencial para o desenvolvimento de aplicações web modernas, facilitando a comunicação entre sistemas de maneira eficiente e escalável" (Bessereau, 2015).

### 3 Modelação e desenvolvimento do projeto

#### 3.1 Requisitos

Os diagramas Use Case (casos de uso) descrevem interações específicas entre os utilizadores e um sistema, especificando sequências de eventos para realizar uma função particular.

No seguinte diagrama é representado o caso de uso para este projeto, representado na Figura 10.

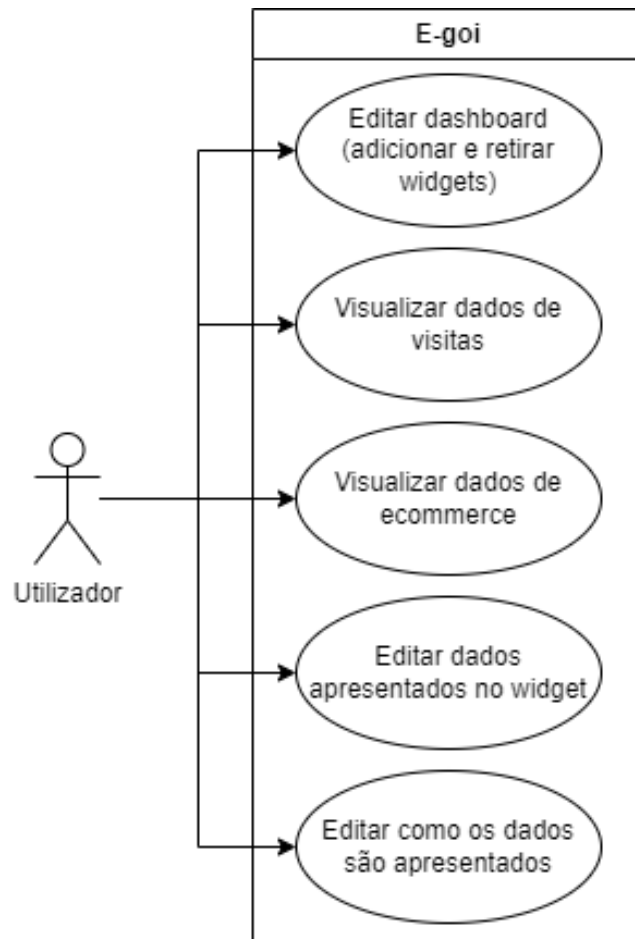


Figura 9 Casos de uso



### 3.1.1 Requisitos Funcionais

No âmbito deste projeto, foram identificados vários casos de uso que descrevem as principais interações dos utilizadores com o dashboard. Abaixo, apresento uma descrição detalhada de cada caso de uso:

#### 1. RF01 Editar dashboard (adicionar e remover widgets):

Este caso de uso permite aos utilizadores editar a composição do widget, adicionar novos widgets para visualizar diferentes conjuntos de dados ou remover widgets existentes para ajustar o dashboard.

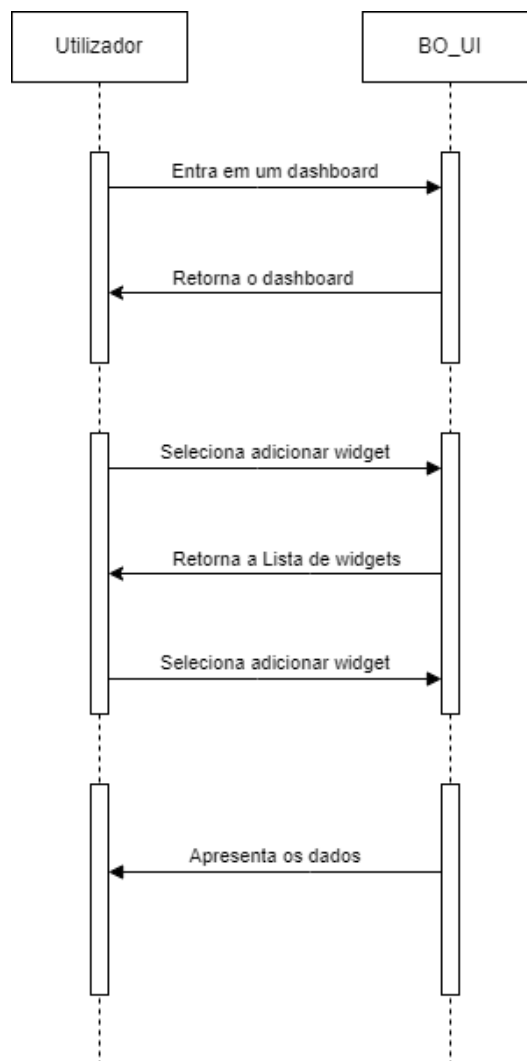


Figura 10 Diagrama de Sequência Caso de Uso - Editar Dashboard

Fluxo do Caso de uso:

1. O utilizador acede à página de edição do dashboard.
2. O utilizador seleciona a opção para adicionar um novo widget.
3. O sistema exibe uma lista de opções de widgets disponíveis.

4. O utilizador seleciona o widget desejado para adicionar ao painel.
  5. O sistema adiciona o novo widget ao painel do utilizador.
  6. O utilizador tem a opção de remover widgets existentes, seguindo um processo semelhante.
2. RF02 Visualizar dados de visitas:

Este caso de uso permite aos utilizadores visualizar dados estatísticos relacionados a visitas e atividades de e-commerce nos seus sites.

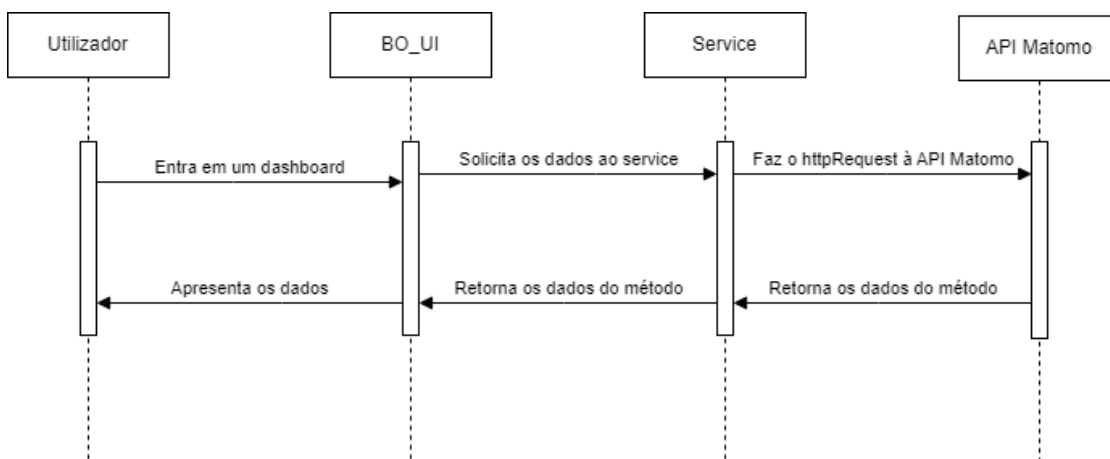


Figura 11 Diagrama de Sequência Caso de Uso - Visualizar dados Visitas

Fluxo do caso de uso:

1. O utilizador acede ao dashboard (que já tenha widgets adicionados).
  2. O sistema exibe automaticamente os widgets configurados para mostrar os dados de visitas e e-commerce.
  3. O utilizador pode interagir com os widgets para explorar as diferentes métricas e visualizações de dados.
3. RF03 Visualizar dados de e-commerce:

Este caso de uso permite aos utilizadores visualizar dados estatísticos relacionados a visitas e atividades de e-commerce nos seus sites.

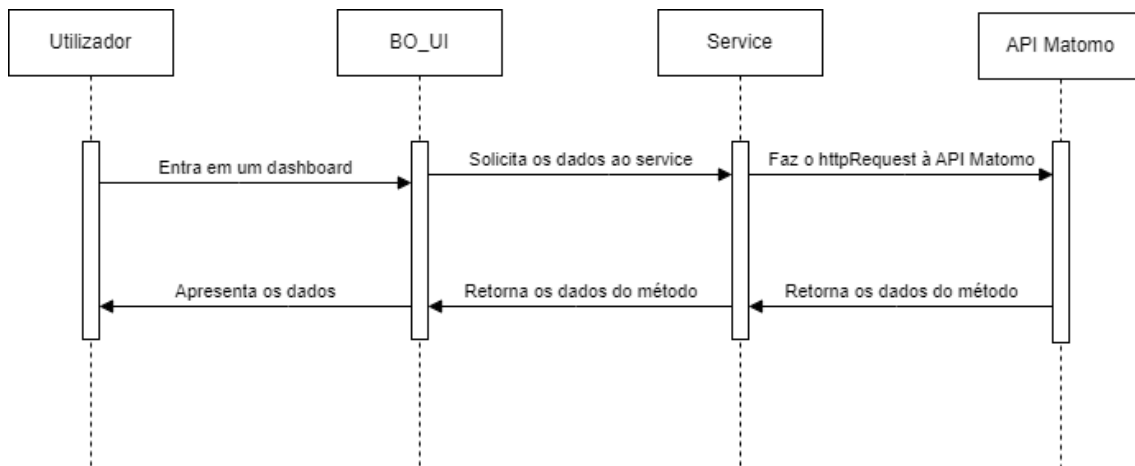


Figura 12 Diagrama de Sequência Caso de Uso - Visualizar dados E-commerce

Fluxo do caso de uso:

1. O utilizador acede ao dashboard (que já tenha widgets adicionados).
2. O sistema exibe automaticamente os widgets configurados para mostrar os dados de visitas e e-commerce.
3. O utilizador pode interagir com os widgets para explorar as diferentes métricas e visualizações de dados.

#### 4. RF04 Editar Métricas Apresentadas:

Este caso de utilização permite aos utilizadores personalizar as métricas específicas que desejam visualizar nos seus widgets de estatísticas.

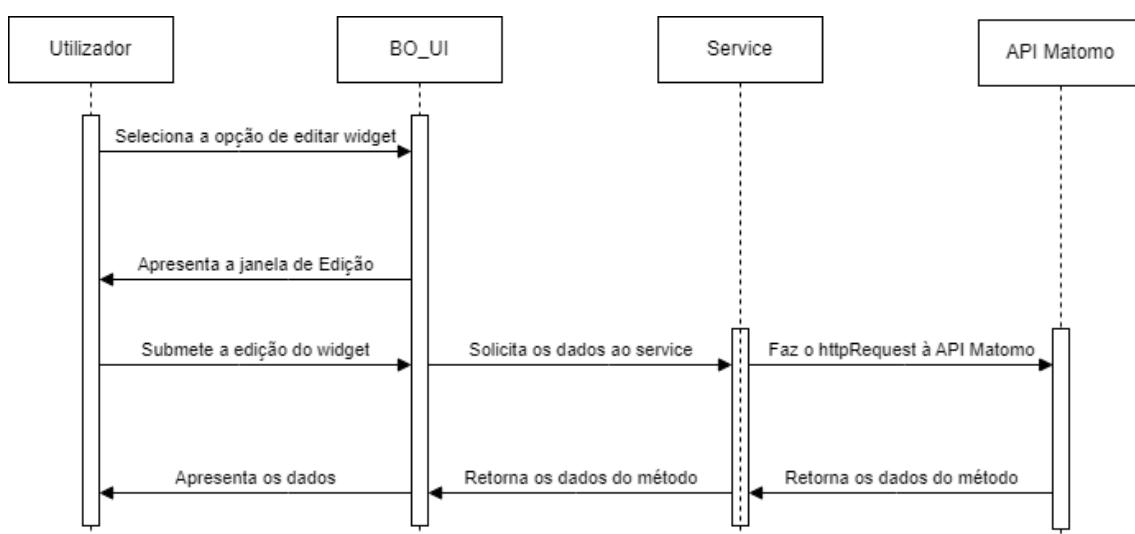


Figura 13 Diagrama de Sequência Caso de Uso - Editar Métricas Apresentadas

Fluxo do caso de uso:

1. O utilizador acede à página de edição do widget.
2. O sistema exibe uma lista de métricas disponíveis para o widget selecionado.
3. O utilizador pode selecionar as métricas desejadas entre as opções fornecidas.
4. O sistema atualiza o widget para exibir as métricas selecionadas pelo utilizador.
5. RF05 Editar como os dados são apresentados (gráficos, mapas, etc.):

Este caso de uso permite aos utilizadores personalizar o modo de apresentação dos dados nos seus widgets, pode escolher entre diferentes tipos de visualização, como gráficos de barras, gráficos de pizza, mapas, entre outros.

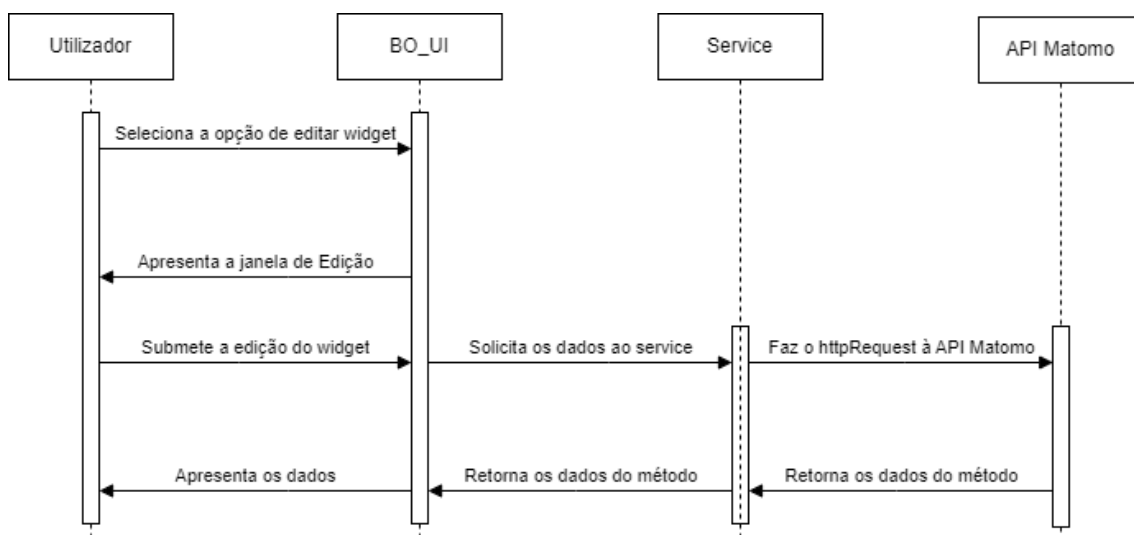


Figura 14 Diagrama de Sequência Caso de Uso - Editar como os dados são apresentados

Fluxo do caso de uso:

1. O utilizador acede à página de edição do widget.
2. O sistema exibe uma opção para selecionar o tipo de visualização desejado.
3. O utilizador pode escolher entre os diferentes tipos de visualização disponíveis.
4. O sistema atualiza o widget para exibir os dados de acordo com o tipo de visualização selecionado pelo utilizador.

Estes são os principais casos de uso identificados para do projeto, abrangendo as principais funcionalidades e interações que os utilizadores terão com a aplicação. As descrições detalhadas destes casos de uso estão em anexo.

### Requisitos Não Funcionais

Por outro lado, os requisitos não-funcionais consistem nas propriedades do sistema, mais concretamente portabilidade, usabilidade, desempenho, segurança, entre outras.

Tabela 2 Requisitos não funcionais

<b>Identificador</b>	<b>Requisito</b>	<b>Descrição</b>
RNF01	Desempenho	O sistema deve ser capaz de lidar com grandes volumes de dados e proporcionar tempos de resposta rápidos.
RNF02	Segurança	O sistema deve garantir a segurança dos dados e garantir que os utilizadores apenas tenham acesso aos dados dos seus sites.
RNF03	Usabilidade	O sistema deve ser intuitivo e fácil de usar, mesmo para utilizadores sem conhecimentos técnicos avançados.
RNF04	Manutenção	O sistema deve ser fácil de manter e atualizar, com uma arquitetura MVC.
RNF05	Compatibilidade	O sistema deve ser compatível com uma variedade de dispositivos e browsers, para garantir uma experiência consistente para todos os utilizadores.

### 3.2 Análise e descrição da arquitetura do sistema.

Na Figura 15 está representado um diagrama de componentes que representa a arquitetura do sistema deste projeto.

Este diagrama faz parte dos diagramas UML (Unified Modeling Language) usados em engenharia de software para descrever a arquitetura de componentes de um sistema.

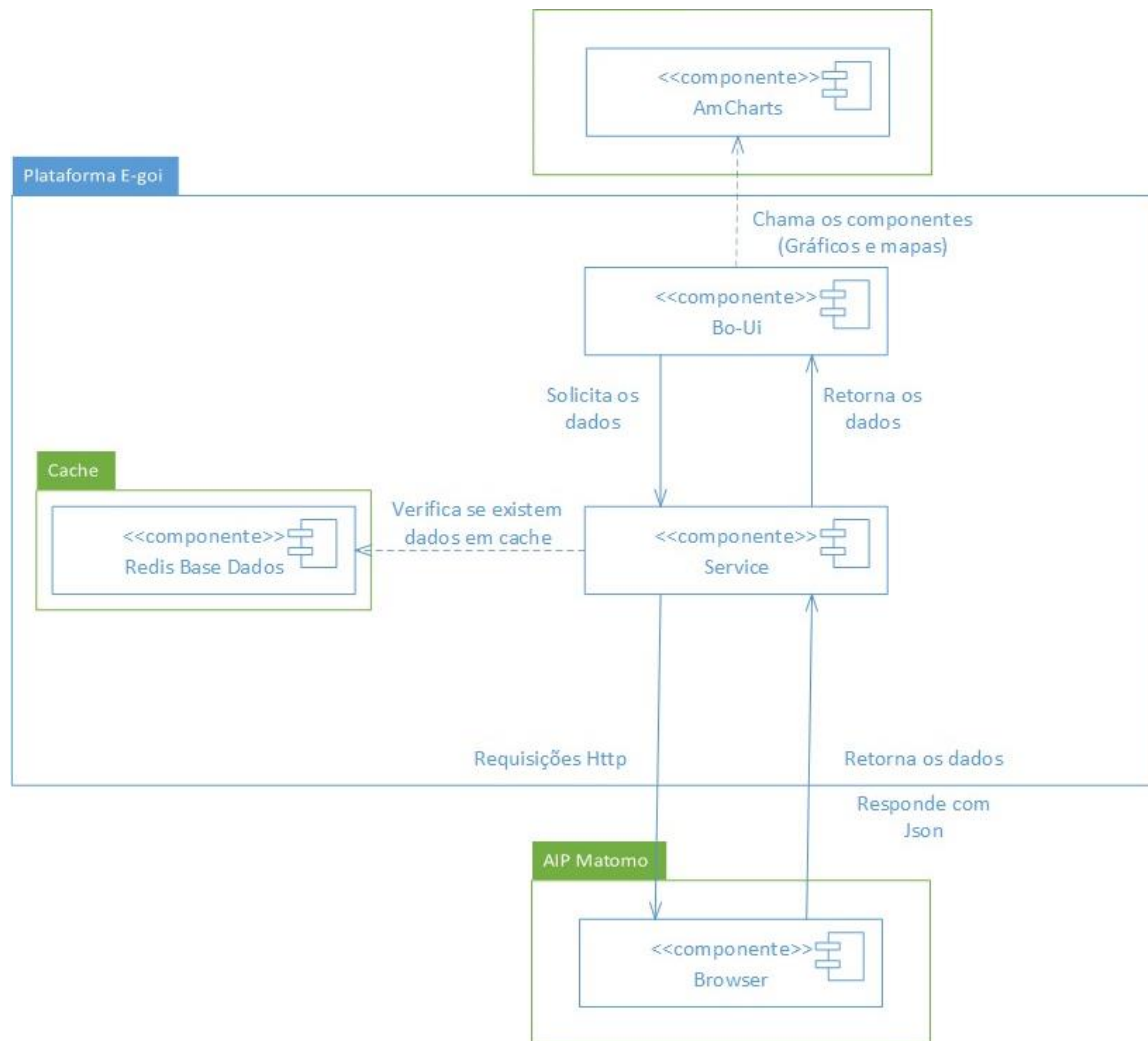


Figura 15 Diagrama de componentes

Principais Componentes:

- Bo-ui: É o projeto de frontend da plataforma E-goi.
- Service: É o projeto de backend da plataforma E-goi onde são feitos os pedidos dos dados e o tratamento desses mesmos dados. As requisições são feitas por HTTP à API do Matomo, que responde a essas requisições com os dados solicitados no formato JSON, que é comum para APIs.
- Redis: É a base de dados em cache utilizada pela E-goi para armazenar temporariamente os dados de forma a evitar consultas repetidas do service à API.
- API Matomo: É API onde o service faz o pedido dos dados
- AmCharts: É o componente que oferece os gráficos e mapas para serem apresentados nos widgets.

### 3.3 Diagrama de fluxo de utilização

Na Figura 16 está representado o diagrama de fluxo que mostra o processo de utilização do dashboard e os seus widgets. O processo é responsável por realizar as seguintes etapas:

- 1) Abrir/Criar Dashboard: O utilizador abre um dashboard existente ou cria um novo.
- 2) Visualizar Dados dos Widgets: O utilizador visualiza os dados dos widgets que já foram inseridos no dashboard.
- 3) Adicionar/Eliminar Widget: O utilizador pode:
  - i) Adicionar novos widgets ao dashboard.
  - ii) Eliminar widgets existentes do dashboard.
- 4) Editar Widgets: O utilizador pode editar as configurações dos widgets existentes.
- 5) Visualizar Dashboard Atualizado: O utilizador visualiza o dashboard atualizado com as alterações feitas (widgets adicionados, eliminados ou editados).
- 6) Fim: O fluxo termina com a visualização do dashboard atualizado.

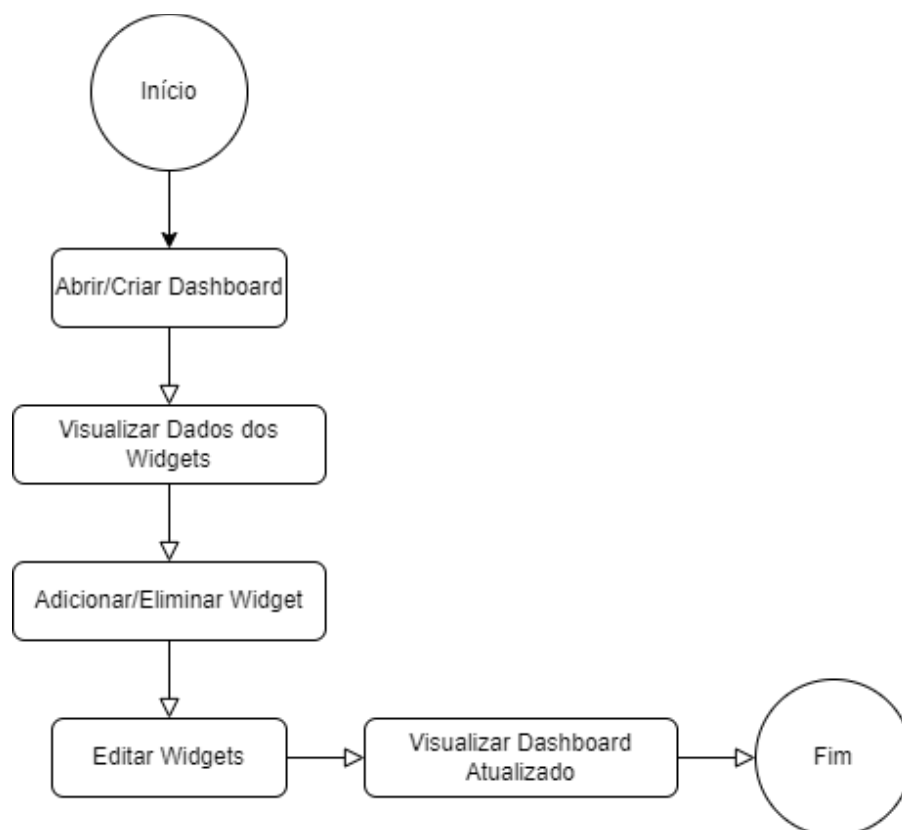


Figura 16 Diagrama de fluxo de utilização



### 3.4 Diagrama de fluxo de sistema

O diagrama de fluxo de sistema seguinte ilustra o processo geral de utilização dos dashboards e os seus widgets, bem como a obtenção dos dados através da API do Matomo para a apresentação nos widgets.

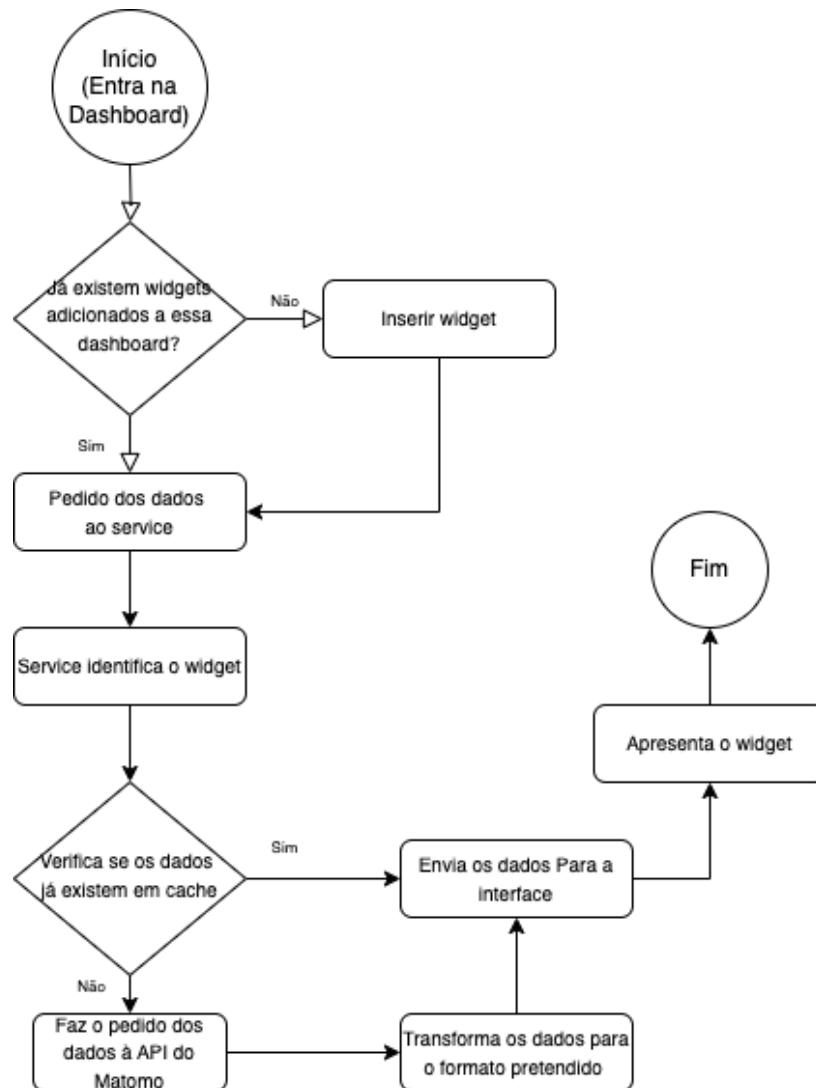


Figura 17 Diagrama de fluxo de sistema

O diagrama começa com a verificação se já existem widgets adicionados à dashboard:

#### Decisão Inicial: Existência de Widgets

- Pergunta: "Já existem widgets adicionados a essa dashboard?"
- Resposta Sim: Se a resposta for afirmativa, o fluxo segue para a ação de pedido de dados.

- Resposta Não: Se a resposta for negativa, o fluxo orienta o utilizador a adicionar um novo widget.

**Ação: Inserção de Widget**

- Quando não existem widgets, a próxima etapa é a inserção de um novo widget na dashboard. Este passo é essencial para garantir que a dashboard contém os elementos necessários para a visualização dos dados.

**Ação: Pedido de Dados ao Serviço**

- Após a inserção de um novo widget ou a confirmação da existência de widgets, o próximo passo é realizar um pedido de dados ao serviço responsável.
- Descrição: "Pedido dos dados ao serviço" implica a comunicação com o serviço interno para identificar os widgets e verificar se os dados estão disponíveis em cache.

**Decisão: Verificação de Dados em Cache**

- Pergunta: "Os dados já existem em cache?"
- Resposta Sim: Se os dados estão em cache, são enviados diretamente para a interface.
- Resposta Não: Se os dados não estão em cache, o serviço faz um pedido à API do Matomo para obter os dados necessários.

**Ação: Pedido de Dados à API do Matomo**

- Se os dados não estão em cache, é feito um pedido à API do Matomo para recuperar os dados necessários.
- Descrição: "Faz o pedido dos dados à API do Matomo" implica a comunicação com o serviço externo do Matomo para recuperar os dados relevantes.

**Ação: Transformação e Envio de Dados**

- Após a obtenção dos dados, estes são transformados para o formato necessário e enviados para a interface do utilizador.

- Descrição: "Transforma os dados para o formato pretendido" e "Envia os dados para a interface" garantem que os dados são apresentados corretamente no widget da dashboard.

### Ação: Apresentação do Widget

- Finalmente, os dados são apresentados no widget da dashboard.
- Descrição: "Apresenta o widget" completa o processo, garantindo que o utilizador pode visualizar os dados atualizados.

O diagrama de fluxo de utilização oferece uma representação visual do processo de verificação e adição de widgets a uma dashboard e da obtenção de dados via API do Matomo. Ao utilizar este diagrama, é possível garantir uma implementação coerente e bem organizada das tarefas descritas, promovendo uma gestão eficaz da dashboard e dos dados associados.

## 3.5 Diagrama de sequência

Na Figura 18 está representado um diagrama de sequência que representa a arquitetura do sistema deste projeto.

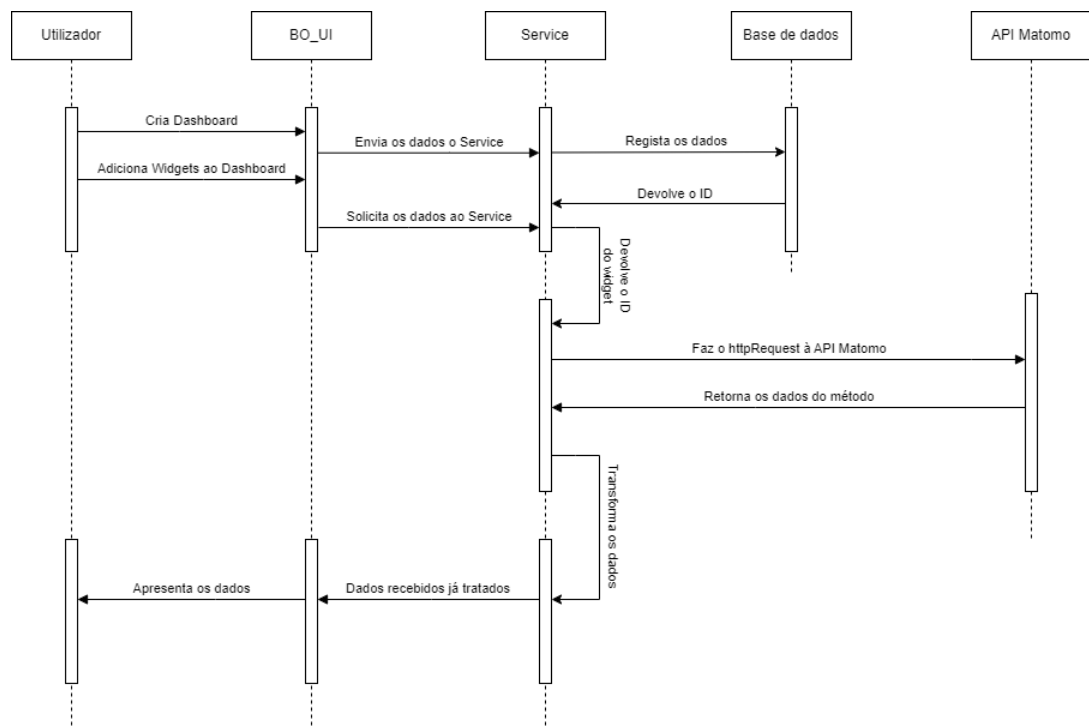


Figura 18 Diagrama de sequência

Criação de *Dashboard* e Adição de *Widgets*:

- O utilizador inicia o processo ao criar um *dashboard* personalizado e adicionando *widgets* conforme as suas necessidades e preferências.

Envio de Dados pelo BO\_UI para o *Service*:

- Uma vez que o utilizador configura o *dashboard* com os *widgets* desejados, o BO\_UI envia os dados associados a esses *widgets* para o *Service*. Estes dados podem incluir informações sobre a interação do utilizador com os *widgets*.

Registo de Dados na Base de Dados e Retorno de ID do *Widget*:

- O *Service* recebe os dados do BO\_UI e procede ao registo destes na base de dados do sistema. Após o registo bem-sucedido, o *Service* retorna um identificador único (ID) para cada *widget* em relação ao *dashboard* em que foram adicionados. Este ID é crucial para identificar e recuperar os dados associados a cada *widget* posteriormente.

Solicitação à API *Matomo* para Obtenção de Dados do *Widget*:

- Além de registar os dados na base de dados interna, o *Service* realiza uma solicitação à API *Matomo*, uma ferramenta de análise de dados, para obter informações adicionais sobre os *widgets*. Esta API fornece dados estatísticos detalhados, como o número de visualizações, cliques ou outras métricas relevantes, que complementam as informações recolhidas pelo sistema.

Tratamento e Apresentação dos Dados ao Utilizador:

- Os dados recebidos da API *Matomo* são então processados e transformados pelo *Service*, de modo a estarem prontos para apresentação. Após o tratamento dos dados, estes são retornados ao BO\_UI, que por sua vez os apresenta ao utilizador no *dashboard*. Os *widgets* são atualizados dinamicamente com as informações mais recentes, proporcionando ao utilizador uma visualização precisa e em tempo real dos dados relevantes.

### 3.6 Atividades realizadas

#### 3.6.1 Reunião Inicial com LD *Integrations* e PM *Integrations*

Uma reunião inicial foi agendada com Renato Gomes (LD *Integrations*) e com Miguel Azevedo (PM *Integrations*). O principal objetivo desta reunião foi informar-me sobre os objetivos do projeto. Durante a reunião, foi discutido o processo atual de *tracking* de dados, que está a ser realizado através do Matomo. Nesse contexto, foram identificados os primeiros pontos a serem analisados para garantir o sucesso do projeto.

#### 3.6.2 Instalação dos projetos

A instalação dos projetos da E-goi necessários para o desenvolvimento do *dashboard*, no caso o ‘*service*’ (que é a parte de *backend* da plataforma, ou seja, onde são feitos os pedidos dos dados e onde são tratados esses mesmo dados) e o bo-ui (que é a parte *frontend* da plataforma ou seja define o aspeto da interface e como são mostrados os dados). Esta foi uma fase mais demorada que o esperado já que a instalação dos projetos deu bastantes problemas, estes problemas foram resolvidos com ajuda da equipa de SRE já que estes foram os que programaram o instalador.

#### 3.6.3 Análise dos *dashboards*

Nesta análise dos *dashboards* disponíveis, notei que todos eles partilham uma característica fundamental: a capacidade de serem adaptáveis e editáveis. Isso deve-se ao facto de todas as métricas serem apresentadas através de *widgets*, que podem ser facilmente personalizados e rearranjados conforme as necessidades do utilizador.

No caso oferece as seguintes possibilidades:

- Criar múltiplos *dashboards* por cliente:

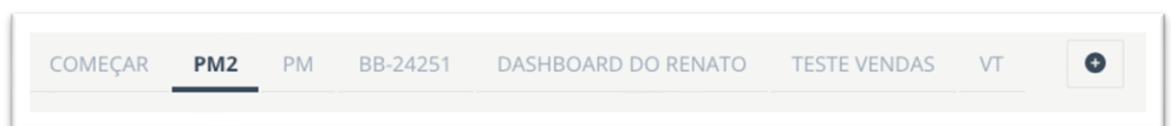


Figura 19 Aba de dashboards

- Criação de dashboards:

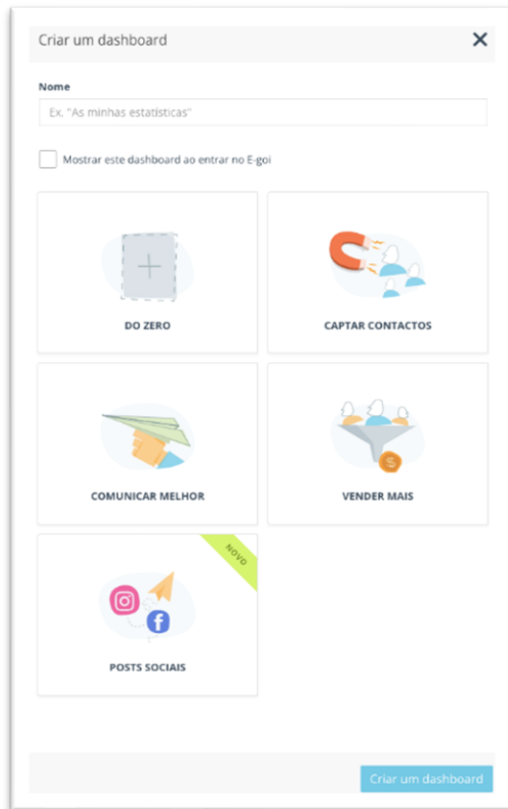


Figura 20 Interface de criação de dashboard

- Adicionar vários widget à Dashboard:

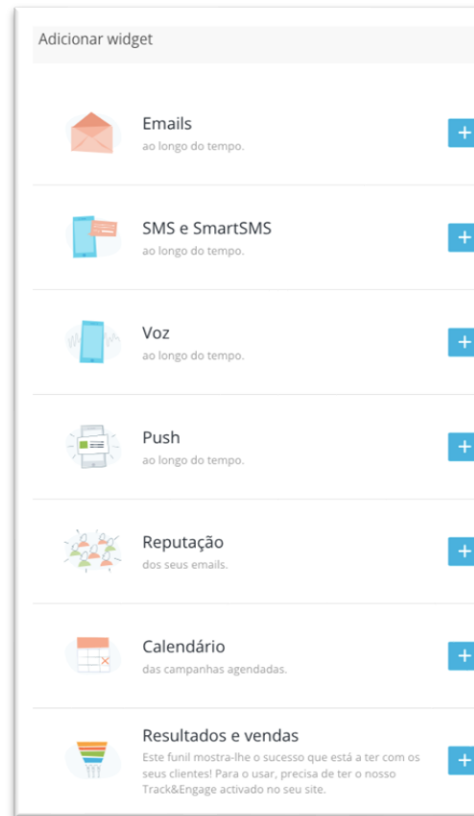


Figura 21 Interface de adição de widget

- Várias estruturas de apresentação por *dashboard*:

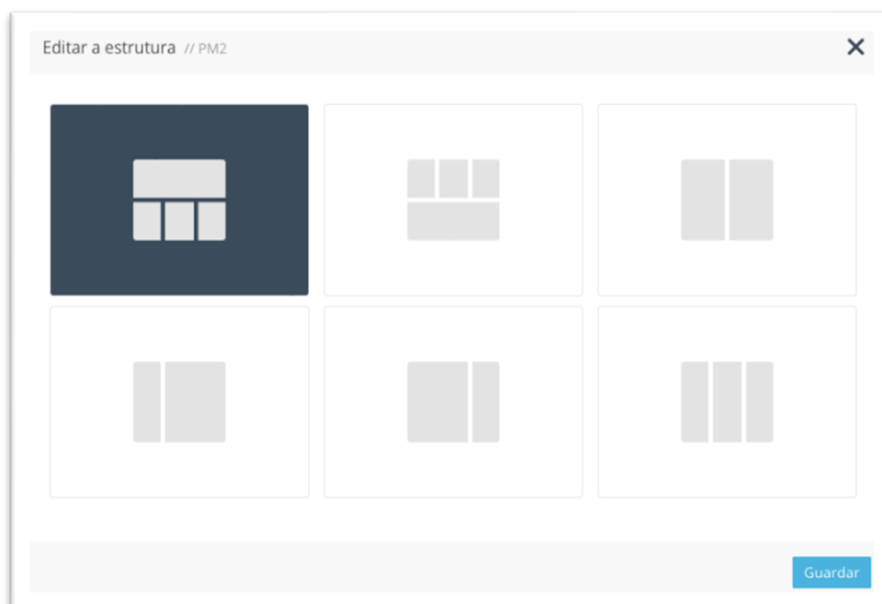


Figura 22 Interface de estrutura de dashboard

Esta abordagem proporciona uma flexibilidade significativa na configuração dos *dashboards*, permitindo aos utilizadores escolher e organizar as métricas de acordo com as suas preferências e objetivos específicos.

Assim, esta análise reforça a importância dos *widgets* como uma ferramenta central na construção de *dashboards* eficientes e intuitivos, capazes de fornecer insights valiosos e relevantes para os utilizadores.

#### 3.6.4 Estudo da API do Matomo

Estudar a API Matomo revelou-se uma etapa fundamental no desenvolvimento deste projeto. Dedicar tempo à análise detalhada da documentação fornecida pela plataforma foi essencial para compreender a disponibilidade dos dados e os métodos de acesso disponíveis na API.

Durante essa análise, identifiquei os parâmetros necessários para cada método da API e compreendi as diferentes formas de para aceder aos dados. Além disso, explorei as possibilidades de filtragem e ordenação dos dados, garantindo assim a extração das informações relevantes para o projeto.

Através dos parâmetros do pedido HTTP à API do Matomo, é possível definir como os dados são agrupados em termos de tempo, as datas limites dos dados, o limite de dados retornados e o formato dos dados. Por exemplo, o parâmetro *'period'* permite agrupar as visitas ao site por dias, semanas, meses, anos ou para um período específico entre duas datas determinadas. O parâmetro *'date'* define as datas limites dos dados, enquanto o *'filter\_limit'* determina o limite de dados retornados. O formato dos dados pode ser definido pelo parâmetro *'format'*.

Esta etapa de estudo e análise da API foi crucial para o planeamento e desenvolvimento do backend dos widgets. Permitiu-me ter uma compreensão sólida das capacidades e limitações da API do Matomo, orientando-me na seleção dos métodos (e dos parâmetros) a serem utilizados para alcançar os objetivos.

#### 3.6.5 Desenvolvimento do backend

Após uma análise detalhada da interface do Matomo e da sua API, foi determinado que os principais dados a serem mostrados na *dashboard* seriam

relacionados às visitas e ao e-commerce. No que diz respeito às visitas, decidiu-se incluir dados gerais de visitas, bem como informações sobre as visitas por país e por página do site do cliente. Quanto aos dados de e-commerce, ficou estabelecido que seriam apresentados os resultados de vendas e informações sobre carrinhos abandonados. Essa seleção foi feita com base na relevância desses dados para fornecer insights úteis aos utilizadores do *dashboard*.

Para obter esses dados, foram utilizados os seguintes métodos da API do Matomo:

- VisitsSummary.get: Este foi o método adotado para obter dados gerais de visitas, como o número total de visitas, quantidade de páginas visitadas, entre outros.

[https://demo.matomo.cloud/?module=API&method=VisitsSummary.get&idSite=1&period=day&date=yesterday&format=JSON&token\\_auth=anonymous](https://demo.matomo.cloud/?module=API&method=VisitsSummary.get&idSite=1&period=day&date=yesterday&format=JSON&token_auth=anonymous)

```
{
  "nb_uniq_visitors": 9606,
  "nb_users": 1,
  "nb_visits": 10642,
  "nb_actions": 31015,
  "nb_visits_converted": 1053,
  "bounce_count": 6152,
  "sum_visit_length": 2758287,
  "max_actions": 76,
  "bounce_rate": "58%",
  "nb_actions_per_visit": 2.9,
  "avg_time_on_site": 259
}
```

Figura 23 VisitsSummary resposta exemplo

Após analisar os dados do método VisitsSummary.get, identifiquei que os dados mais relevantes para serem apresentados na dashboard são o *'nb\_uniq\_visitors'* e o *'nb\_visits'*.

- O *'nb\_uniq\_visitors'*, ou número de visitas ao site, representa a contagem de indivíduos distintos que visitaram o site durante o período especificado. Esta métrica é útil para entender quantos utilizadores únicos o site está a atrair.



- Já o *'nb\_visits'*, ou número de visitas, representa o total de visitas ao site, incluindo visitas repetidas pelo mesmo utilizador. Essa métrica é importante para avaliar o nível de atividade geral no site.

Ao apresentar esses dois dados, será possível ter uma visão abrangente da atividade do site em termos de visitas, combinando informações sobre o número de visitantes únicos e o total de visitas, fornecendo insights valiosos sobre o envolvimento dos utilizadores e o desempenho geral do site.

- `UserCountry.getCountry`: Este foi o método adotado para obter informações sobre as visitas agrupadas por país, dando assim a possibilidade de visualização de dados geográficos sobre a origem das visitas aos sites.

[https://demo.matomo.cloud/?module=API&method=UserCountry.getCountry&idSite=1&period=day&date=yesterday&format=JSON&token\\_auth=anonymous](https://demo.matomo.cloud/?module=API&method=UserCountry.getCountry&idSite=1&period=day&date=yesterday&format=JSON&token_auth=anonymous)

```
[
  {
    "label": "Estados Unidos",
    "nb_uniq_visitors": 1837,
    "nb_visits": 1954,
    "nb_actions": 5130,
    "nb_users": 0,
    "max_actions": 55,
    "sum_visit_length": 433829,
    "bounce_count": 1217,
    "nb_visits_converted": 190,
    "goals": {↔},
    "nb_conversions": 196,
    "revenue": 19343.050000000001,
    "code": "us",
    "logo": "plugins/Morpheus/icons/dist/flags/us.png",
    "segment": "countryCode==us",
    "logoHeight": 16
  },
]
```

Figura 24 UserCountry resposta exemplo

Após analisar os dados obtidos através do método `UserCountry.getCountry`, identifiquei que os dados mais relevantes para serem

apresentados na dashboard são *'label'*, *'nb\_uniq\_visitors'*, *'revenue'* e *'code'*. Essas informações são cruciais para compreender a origem geográfica das visitas do site e os valores de vendas.

- O *'label'* corresponde ao nome do país de onde os visitantes estão a aceder ao site. Esta informação é essencial para identificar a localização dos utilizadores e entender melhor o público-alvo do site.
- O *'nb\_uniq\_visitors'* representa o número de visitantes únicos provenientes de cada país. Essa métrica permite avaliar a popularidade do site em diferentes regiões e identificar quais países contribuem mais para o tráfego global.
- O *'revenue'* indica o valor total das vendas geradas por visitantes de cada país. Esta métrica é fundamental para analisar o desempenho de vendas em diferentes mercados e identificar oportunidades de crescimento.
- Por fim, o *'code'* corresponde ao código do país conforme definido pelo padrão ISO 3166. Essa informação é útil para identificar de forma única cada país e realizar análises geográficas mais precisas.

Ao apresentar esses dados na dashboard, será possível ter uma visão detalhada da distribuição geográfica dos visitantes, o seu impacto nas vendas e as oportunidades de expansão em diferentes mercados.

- `Actions.getPageUrls`: Este foi o método adotado para obter dados específicos sobre as visitas agrupadas por página do site do cliente, dando a possibilidade de analisar o desempenho de cada página.

[https://demo.matomo.cloud/?module=API&method=Actions.getPageUrls  
&idSite=1&period=day&date=yesterday&format=JSON&token\\_auth=anonymo  
us](https://demo.matomo.cloud/?module=API&method=Actions.getPageUrls&idSite=1&period=day&date=yesterday&format=JSON&token_auth=anonymo-us)

```
{
  "label": "products",
  "nb_visits": 2256,
  "nb_hits": 3480,
  "sum_time_spent": 376520,
  "nb_hits_following_search": 71,
  "entry_nb_visits": 986,
  "entry_nb_actions": 7212,
  "entry_sum_visit_length": 934769,
  "entry_bounce_count": 79,
  "exit_nb_visits": 701,
  "goals": {↔},
  "avg_page_load_time": 0,
  "avg_time_on_page": 108,
  "bounce_rate": "8%",
  "exit_rate": "31%",
  "segment": "pageUrl=^https%253A%252F%252Fdivezone.net%252Fproducts",
  "idsubdatatable": 108
},
```

Figura 25 Actions.getPageUrls resposta exemplo

Após analisar os dados obtidos através do método Actions.getPageUrls, identifiquei que as informações mais relevantes para serem apresentadas na dashboard são 'label', 'nb\_visitors' e 'time\_spent'.

- O '*label*' corresponde ao URL das páginas do site visitadas pelos utilizadores. Essa informação é essencial para identificar quais páginas têm maior afluência.
- O '*nb\_visitors*' representa o número total de visitantes que acederam a cada página do site. Essa métrica permite avaliar a popularidade de cada página e identificar aquelas que geram mais interesse aos utilizadores.
- O '*time\_spent*' indica o tempo médio despendido pelos visitantes em cada página do site. Essa métrica é fundamental para avaliar o envolvimento dos utilizadores com o conteúdo de cada página e identificar oportunidades de otimização para aumentar o tempo de permanência e até identificar que existem páginas em que os utilizadores passem demasiado tempo por falta de fluidez em algum processo do site como por exemplo um checkout.

Ao apresentar esses dados na dashboard, será possível ter uma visão detalhada do desempenho de cada página do site, identificar quais são as mais

visitadas e avaliar o envolvimento dos utilizadores com o conteúdo oferecido. Isso permitirá tomar decisões informadas para melhorar a experiência do utilizador e otimizar o site para alcançar os objetivos definidos.

- `Goals.getItemsName`: Este foi o método adotado para obter informações sobre os resultados de vendas e carrinhos abandonados, fornecendo dados importantes relacionados ao e-commerce. Para obter informações sobre produtos abandonados em carrinho basta habilitar o parâmetro `'abandoned_carts'` ao chamar esse método.

[https://demo.matomo.cloud/?module=API&method=Goals.getItemsName&idSite=1&period=day&date=yesterday&abandonedCarts=0&format=JSON&token\\_auth=anonymous](https://demo.matomo.cloud/?module=API&method=Goals.getItemsName&idSite=1&period=day&date=yesterday&abandonedCarts=0&format=JSON&token_auth=anonymous)

```
{
  "label": "Basic Wetsuit",
  "revenue": 54000,
  "quantity": 108,
  "orders": 107,
  "avg_price": 500,
  "avg_quantity": 1,
  "conversion_rate": "0%",
  "segment": "productName==Basic+Wetsuit"
},
```

Figura 26 `Goals.getItemsName` resposta exemplo

Após analisar os dados obtidos através do método `Goals.getItemsName`, identifiquei que as informações mais relevantes para serem apresentadas na dashboard são `'revenue'`, `'quantity'` e `'orders'`.

- O `'revenue'` sendo o valor total de vendas de um produto em específico.
- A `'quantity'` sendo a quantidade vendida de um produto em específico.
- O `'orders'` a quantidade de encomendas que existem com o produto.

Com estes campos é possível calcular o valor total de vendas relativamente a uma loja

Esses métodos foram selecionados com base na sua capacidade de fornecer os dados necessários para os diferentes widgets da dashboard, oferecendo assim uma visualização abrangente e informativa das métricas relevantes.

### 3.6.6 Pedidos à API

Após a definição dos dados a serem apresentados, avancei para a implementação dos pedidos à API do Matomo utilizando requisições HTTP dentro do serviço do projeto. Desenvolvi um método denominado "makeResponseHttpClient", o qual recebe os parâmetros necessários e realiza o pedido à API. Este método é chamado por vários outros métodos, cada um correspondente a um pedido específico, para retornar os dados conforme necessário.

Depois de obter os dados da API do Matomo, que são fornecidos no formato JSON, foi necessário realizar um processo de tratamento. Neste caso, seguindo a abordagem utilizada nos outros widgets da E-goi, os dados passaram por um processo de transformação. Essa etapa é responsável por remover dados desnecessários, lidar com valores nulos e agrupar as informações relevantes. Desta forma, é garantido que apenas os dados necessários são apresentados e são evitados erros causados por valores nulos.

Como é feita essa transformação?

- Verificação de dados vazios: Inicialmente, os dados recebidos são verificados para garantir que não estejam vazios. Se o array estiver vazio, a função retorna imediatamente um array vazio, minimizando o processamento desnecessário.

```
if (empty($data) || !is_array($data)) {  
    return [];  
}
```

Figura 27 Código Verificação de dados vazios

- Criação da estrutura do array de retorno: Caso existam dados válidos, a função começa criando a estrutura do array que será retornado. Esta estrutura é definida de acordo com os requisitos da dashboard e pode variar dependendo do método chamado para obter os dados (por exemplo, por dia, página, país, etc.).

```
$returnData = [
    'by_date' => [],
    'by_hour' => [],
    'by_weekday' => [],
    'by_country' => []
]
```

Figura 28 Código estrutura do array de retorno

- Iteração sobre os dados: Em seguida, a função itera sobre os dados recebidos. Durante essa iteração, apenas os campos relevantes são copiados para o array de retorno. Isso é feito para garantir que apenas os dados necessários sejam incluídos na visualização da dashboard.
- Tratamento de valores nulos: Durante a cópia dos campos, verifica-se se os valores são nulos. Se um valor for nulo, ele é substituído por um valor vazio ou zero, conforme apropriado. Isso ajuda a garantir a integridade e consistência dos dados apresentados na dashboard.

```
foreach ($data as $category => $values) {
    if ($category === 'visits' && is_array($values)) {
        foreach ($values as $date => $v) {
            $returnData['by_date'][$date] = [
                'visits' => !empty($v['nb_uniq_visitors']) ? (int) $v['nb_uniq_visitors'] : 0,
                'page_views' => !empty($v['nb_visits']) ? (int) $v['nb_visits'] : 0,
                'actions' => !empty($v['nb_actions']) ? (int) $v['nb_actions'] : 0,
                'max_actions_in_one_visit' => !empty($v['max_actions']) ? (int) $v['max_actions'] : 0,
                'avg_time_on_site' => !empty($v['avg_time_on_site']) ? (int) $v['avg_time_on_site'] : 0,
                'avg_actions_per_visit' => !empty($v['nb_actions_per_visit']) ? (int) $v['nb_actions_per_visit'] : 0,
                'bounce_rate' => !empty($v['bounce_rate']) ? (int) $v['bounce_rate'] : 0
            ];
        }
    }
}
```

Figura 29 Código de iteração sobre os dados recebidos

Ao seguir esses passos, a função 'transformDashboard' prepara os dados recebidos da API do Matomo para serem apresentados de forma adequada na dashboard, garantindo que apenas as informações relevantes sejam incluídas e que os valores nulos sejam tratados corretamente.

Após o desenvolvimento completo do backend do projeto, foi necessário avançar para o frontend, o que implicou a criação de uma estrutura na base de dados para definir os campos relativos ao widget (modos de visualização e métricas) e valores predefinidos dos widgets e para armazenar as opções de visualização quando alteradas no frontend.

Esta abordagem envolve a definição de valores predefinidos na base de dados para cada widget. Quando um widget é adicionado a uma dashboard, ele é inserido com os valores predefinidos correspondentes. Por exemplo, podem ser definidos os seguintes valores predefinidos:

```
"period": "last7Days",
```

```
"show_revenue": true,
```

```
"show_visits": true,
```

Esses valores predefinidos garantem que os *widgets* tenham uma configuração inicial consistente e funcional assim que são adicionados à interface do utilizador. Além disso, ao permitir que as opções de visualização sejam armazenadas no *frontend*, os utilizadores podem personalizar a apresentação dos dados de acordo com as suas preferências individuais, proporcionando uma experiência mais personalizada e flexível no dashboard.

### 3.6.7 Desenvolvimento do frontend

Após testar e analisar que o serviço retornava todos os dados necessários e na estrutura correta, avancei para o projeto bo-ui (frontend).

Analisei a estrutura dos componentes, bem como a integração com os dados fornecidos pelo service. Essa análise proporcionou-me uma percepção valiosa para o design e implementação dos novos widgets, de maneira a garantir uma experiência de utilizador coesa e intuitiva em todo o sistema.

Após analisar os arquivos dos widgets existentes, torna-se evidente a relação com o modelo MVC, uma vez que a organização dos arquivos dos widgets é feita por funcionalidades distintas. Os dados são obtidos em um arquivo específico, tratados em outro e finalmente exibidos em outro. Essa estrutura facilita a manutenção e a compreensão do código, permitindo uma separação clara das responsabilidades entre as diferentes partes do sistema.

Nestes widgets, o primeiro (applyFn.js) ficheiro é responsável por estabelecer a comunicação com o projeto 'service', que por sua vez é encarregue de realizar solicitações e tratar os dados provenientes do Matomo, registando-os para uso posterior.

Além disso, há um arquivo que determina qual template deve ser utilizado com base na presença ou ausência de dados. Se não houver dados disponíveis, o template exibe uma mensagem que indica que é necessário configurar o widget, incluindo informações como o domínio do site que se pretende analisar as estatísticas. Por outro lado, quando os dados estão disponíveis e o widget está configurado, outro template é utilizado para exibir esses dados de forma adequada. Neste template, os dados serão exibidos de acordo com as configurações selecionadas pelo utilizador e tratadas previamente no controlador. Desta forma é garantido que os dados sejam apresentados de forma apropriada, seguindo o formato e as preferências definidas pelo utilizador durante a configuração do widget.

No controlador('nome\_do\_widget'Ctrl.js), os dados são processados e estruturados de acordo com as configurações definidas pelo utilizador no editor de widget. Este processo envolve a seleção das métricas desejadas, como por exemplo o número de visitas, tempo despendido no site, etc., bem como a definição dos formatos de apresentação, como gráficos de linhas, gráficos de setores ou mapas. Assim, o



controlador assegura que as métricas seleccionadas sejam tratadas adequadamente para a geração dos gráficos correspondentes. Existe controlador para o editor do widget e um controlador geral.

```
▼ tePagesDataStats
JS applyFn.js
<> edit.tpl.html
JS editCtrl.js
<> tePagesDataStats.tpl.html
JS tePagesDataStatsCtrl.js
```

Figura 30 Ficheiros dos widgets

Diagrama de ficheiros dos widgets

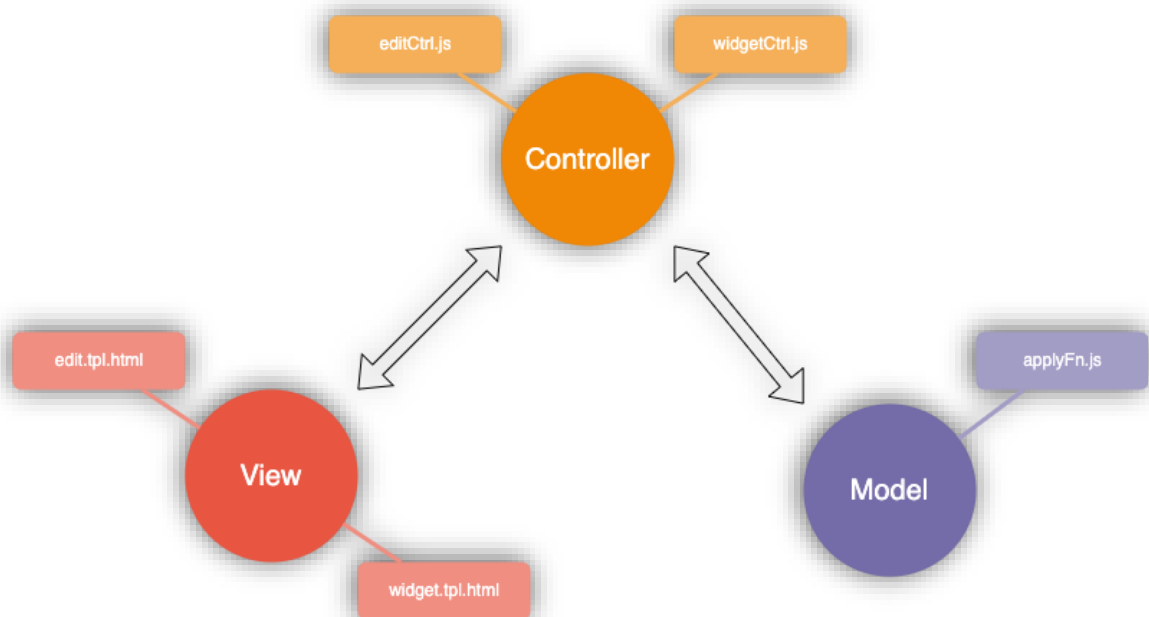


Figura 31 Relação com o modelo MVC

### 3.6.8 Função dos ficheiros presentes em todos os widgets

Os seguintes 5 ficheiros existem em todos os widgets da plataforma E-goi.

#### **applyFn.js**

Este ficheiro estabelece a comunicação com o service para solicitar os dados necessários para apresentar o widget.

**edit.tpl.html**

Este é o template do frame de configuração do widget, permite a configuração de:

- Nome do widget, que é exibido no dashboard.
- Domínio de onde os dados serão apresentados (o cliente pode ter vários domínios associados à sua conta E-goi).
- Período: o utilizador pode seleccionar para ver os dados nos últimos 7 dias, 1 mês e 3 meses.
- Resultado: define o modo de visualização dos dados, variando de widget para widget, mas incluindo opções como gráfico de linhas, gráfico de setores, mapas e métricas (apresenta os valores totais das métricas para o período seleccionado).
- Métricas: os dados a serem mostrados, que variam de widget para widget e podem incluir visualizações, vendas, entre outros.

The screenshot shows a configuration window titled 'Editar widget // Visitas ao Site'. It contains several sections for customizing the widget:

- Nome:** A text input field containing 'Visitas ao Site'.
- Domínio:** A dropdown menu showing '391250.commercesuite.com.br'.
- Período:** A dropdown menu showing 'Últimos 7 dias'.
- Resultados:** A section with the subtitle 'Uma visão das estatísticas das suas visitas ao domínio. Defina o que vai querer ver.' It includes three checked checkboxes: 'Tendência', 'Desempenho', and 'TR/Dashboard/Widgets/Stats/Edit/Views/Maps'.
- Métricas:** A section with the subtitle 'Escolha as métricas que poderá visualizar.' It includes two checked checkboxes: 'Visitas' and 'Páginas Visitadas'.

Figura 32 Template editor do widget de visitas

**edit.Ctrl.js**

Este ficheiro controla os dados definidos pelo utilizador no configurador do widget (edit.tpl.html), incluindo nome, domínio, período, resultado (modo de visualização) e métricas.

**widget.tpl.html**

Este é o template que verifica se já existem dados para chamar o template geral dos widgets (widget.template.html) ou exibir o template para completar as configurações do widget.

**widgetCtrl.js**

Este ficheiro controla os dados do widget, determinando como os dados serão tratados com base nas configurações definidas no editor do widget. Isso inclui preparar os dados no formato necessário para as visualizações específicas, uma vez que o formato dos dados varia de acordo com as visualizações selecionadas.

### 3.6.9 De que maneira seriam apresentados os dados obtidos?

Decidir como seriam apresentados os dados obtidos foi um dos passos que demorou mais do que o previsto, começando pelos dados por país inicialmente pensei em apresentar os dados em formato de texto da seguinte forma:

- Ordenar o array de países com base no número de visitas em ordem decrescente.
- Extrair os três primeiros países do array ordenado.
- Somar o número de visitas dos países restantes.
- Retornar os três primeiros países e a soma das visitas dos países restantes.

Ao considerar a apresentação dos dados por país em texto no widget, percebi que embora seja uma abordagem válida, não era a mais eficaz. Decidi então explorar outras formas de visualização que pudessem melhorar a experiência do utilizador. Durante essa investigação, deparei-me com a possibilidade de utilizar um mapa do

AmCharts, uma biblioteca JavaScript conhecida por oferecer uma vasta gama de opções para criar gráficos e mapas interativos em dashboards. Essa solução proporciona uma representação visual mais intuitiva e atrativa dos dados por país, facilitando a compreensão e análise por parte dos utilizadores.



Figura 33 Mapa exemplo do AmCharts

Decidi, então, dedicar-me à análise da documentação do AmCharts referente aos mapas. Durante esse estudo, constatei que os dados necessários para identificar os países eram os códigos ISO 3166. Estes códigos são padronizados internacionalmente e utilizados para representar países e as suas subdivisões em diversos contextos, como correio, internet, passaportes e transferências bancárias. Verifiquei que esses códigos eram uma das informações disponíveis no JSON de resposta da API do Matomo, o que me permitiria integrá-los de forma direta e eficiente na visualização dos mapas do AmCharts.

Tabela 3 Códigos ISO 3166-1

País	Código ISO 3166-1
Estados Unidos	US
Canadá	CA
Reino Unido	GB

Para os dados relativos às páginas, optei por criar uma comparação entre eles em vez de apresentar os valores absolutos de visitas e tempo médio em cada página. Com esse objetivo em mente, cheguei à conclusão de que a melhor maneira de representar esses dados seria com um gráfico de setores, também utilizando a biblioteca AmCharts.

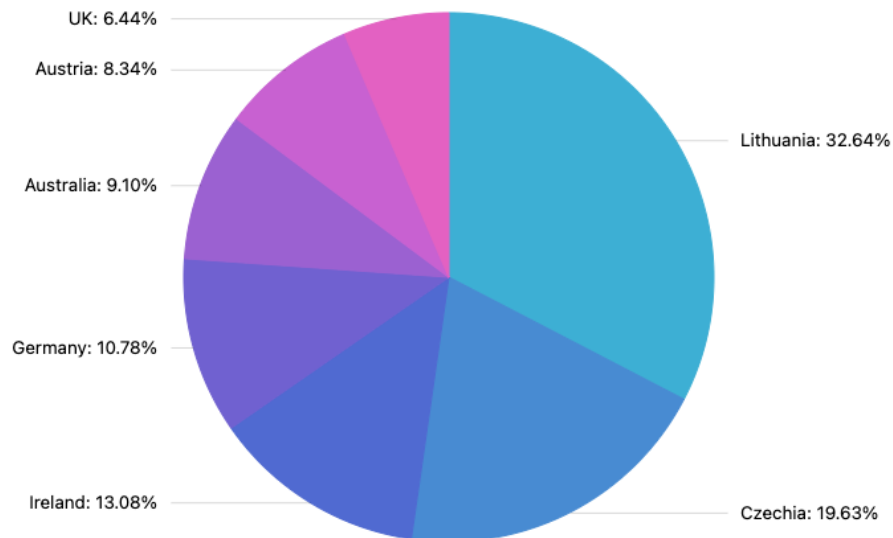


Figura 34 Gráfico circular exemplo do AmCharts

No início, o gráfico foi populado com todos os dados das páginas provenientes do Matomo. No entanto, surgiu um problema: no caso de haver uma grande quantidade de páginas, a maioria das quais era irrelevante devido ao baixo número de visitas. Isso resultou em um gráfico excessivamente preenchido e com excesso de informação, tornando difícil a sua interpretação.

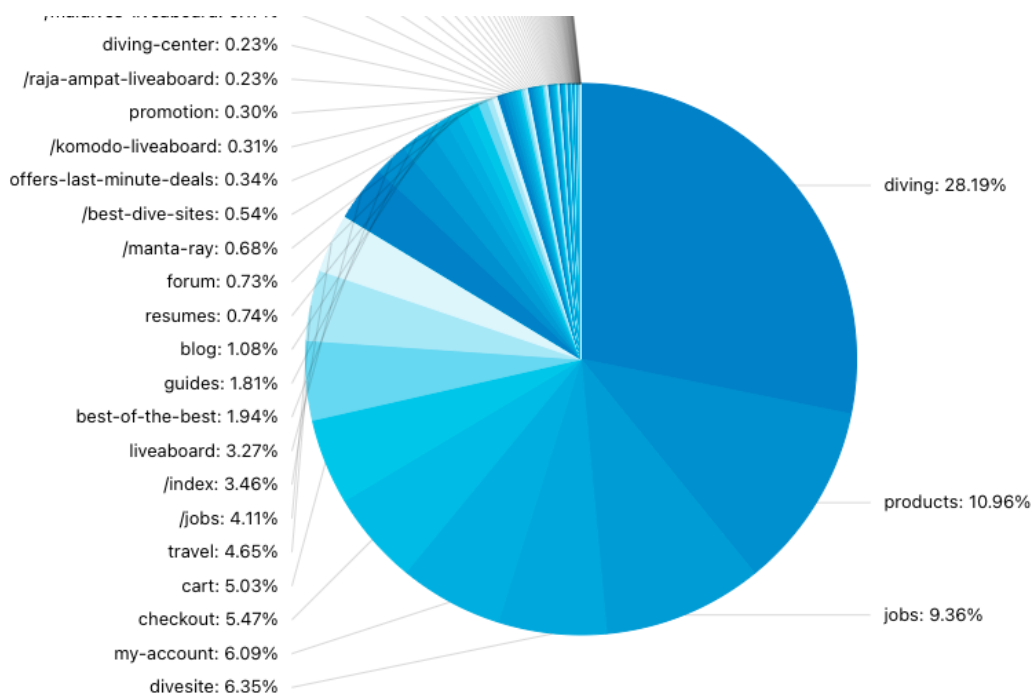


Figura 35 Gráfico circular com 100 setores

Após uma reunião com o Gestor de Projeto (PM), foi decidido que apenas as 10 páginas com mais visitas seriam apresentadas no gráfico circular, já que os dados recebidos da API do Matomo pelo retorna os dados das páginas organizados de forma descendente pelas visitas em cada página. Essa decisão foi tomada com o intuito de evitar a apresentação de dados irrelevantes de páginas de menor importância, tornando assim o gráfico mais intuitivo e facilitando a comparação entre as páginas mais relevantes.

### 3.6.10 Definir as cores para o gráfico de setores

Para escolher as 10 cores para os vários setores do gráfico pie, foi seguido um processo que envolveu o cálculo a partir de duas cores base. Foram escolhidas as duas seguintes cores ‘#0084C2’ = ■ e ‘#E0F5FB’ = ■ com base nas cores usadas na plataforma para manter a plataforma harmoniosa em termos de cores.

O cálculo consistiu em realizar uma interpolação linear no espaço de cores RGB (*Red*, *Green*, *Blue*). Este método de interpolação envolve ajustar gradualmente os valores de cada componente de cor para obter uma transição suave entre a cor inicial e cor final.

Por exemplo, para calcular as cores intermediárias, primeiro decompus as cores escolhidas nos seus valores de vermelho, verde e azul.

Esses são os valores RGB de cada componente de cor em uma escala de 0 a 255.

Para a cor #0084C2:

Tabela 4 Cor #0084C2

Vermelho (R)	0
Verde (G)	132
Azul (B)	194







Para a cor #E0F5FB:

Tabela 5 Cor #E0F5FB

Vermelho (R)	224
Verde (G)	245
Azul (B)	251

Em seguida, foi calculada a diferença entre os valores de cada componente de cor das duas cores base, depois, foi dividida essa diferença pelo número de cores intermediárias desejadas (no caso, 8), para determinar a quantidade de mudança em cada componente de cor necessária para cada etapa intermediária. Os seguintes valores foram obtidos com esses cálculos:

Tabela 6 Gradiente de cores

#0084C2	
#0091C9	
#009DD0	
#00A7D6	
#00AEDA	
#26BAE0	
#4DC6E5	
#80D7ED	
#B3E7F4	
#E0F5FB	

Então, apliquei essa mudança incremental a cada componente de cor, ajustando gradualmente os valores em direção à cor final, para obter uma transição suave entre as

cores base. Este método permitiu criar uma paleta de cores equidistantes e harmoniosas para os diversos setores do gráfico pie, garantindo uma representação visualmente agradável e distinta para cada página apresentada.

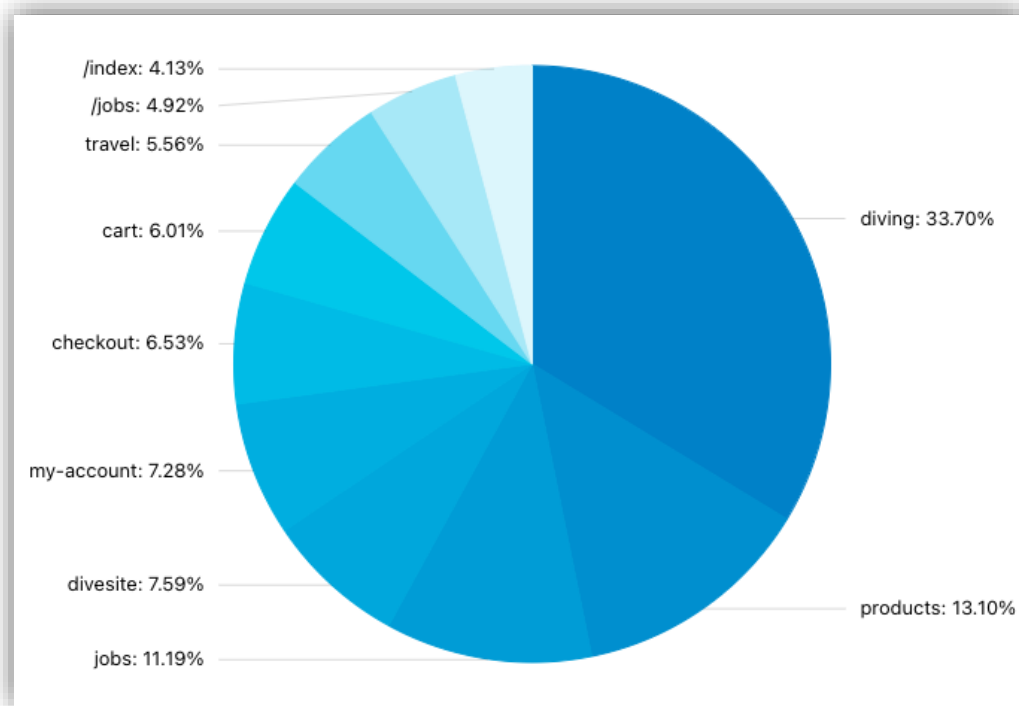


Figura 36 Gráfico circular com 10 setores

Porém esta escolha de apresentar apenas as 10 primeiras páginas para assegurar acessibilidade cria o problema de ocultar dados de páginas que para situações específicas e clientes específicos podem ser relevantes, então para solucionar este problema optou-se por criar também um modo de visualização tabela para este widget de forma a mostrar esses dados para todas as tabelas.

### 3.6.11 Modos de visualização dos dados

Os widgets criados oferecem diferentes modos de visualização dos dados, entre eles:

- Gráficos de Linhas

Este modo de visualização apresenta os dados por dia em um determinado período, desta forma é facilitada a comparação dos valores entre os dias. No seguinte exemplo mostra as visualizações nos últimos 3 meses.



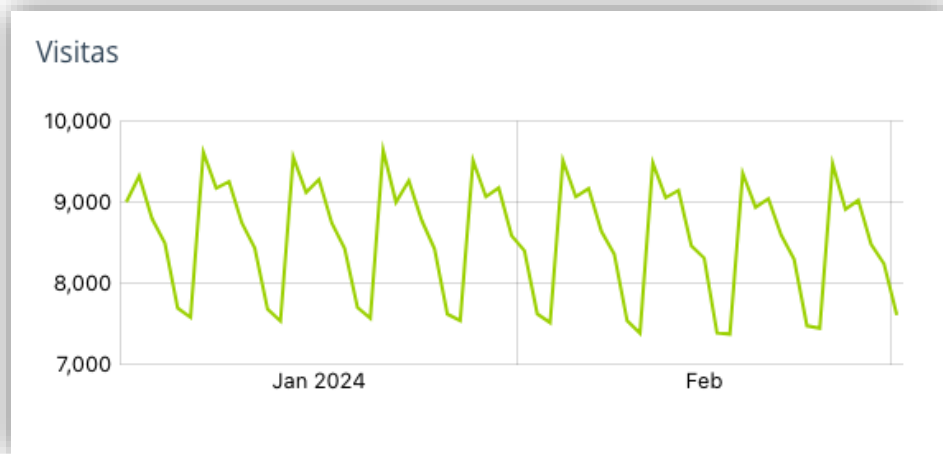


Figura 37 Gráfico de linhas

- Gráficos Circulares

Este modo de visualização apresenta os dados agrupados por um determinado campo

No seguinte exemplo mostra as visitas a um site por página.

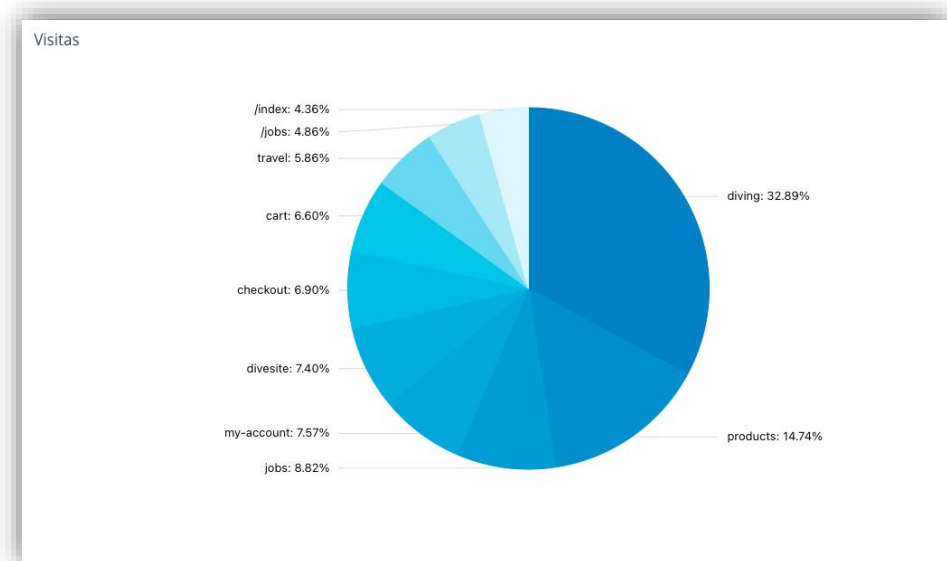


Figura 38 Gráfico circular

- Mapas

Este modo de visualização apresenta os dados agrupados por país facilitando a comparação de dados entre países e tornando a análise mais acessível.

No seguinte exemplo mostra um mapa com as visitas a um site por país.

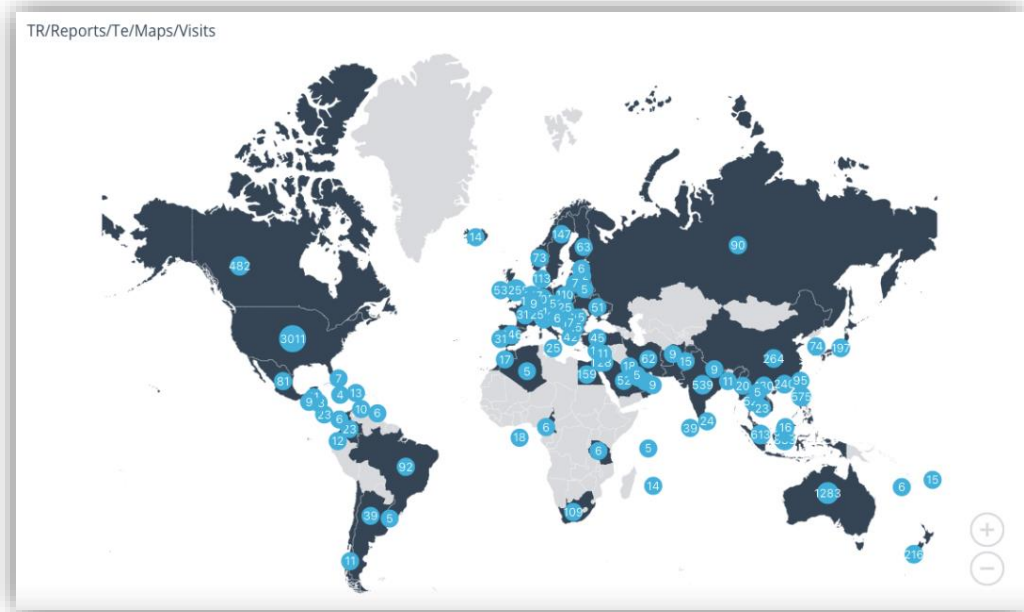


Figura 39 Mapa com visitas

- Métricas

Este modo de visualização apresenta o valor total das métricas-

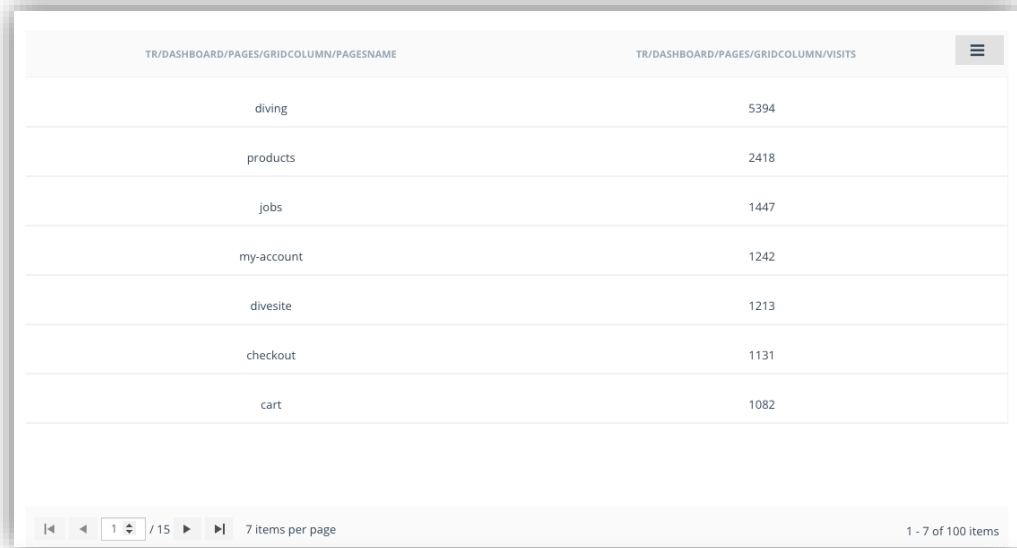
No seguinte exemplo mostra o total das métricas de visualizações:



Figura 40 Total das métricas

- Tabelas

Este modo de visualização apresenta os dados em uma tabela, este modo de visualização é útil para grandes listas de dados já que este inclui paginação na tabela.



TR/DASHBOARD/PAGES/GRIDCOLUMN/PAGESNAME	TR/DASHBOARD/PAGES/GRIDCOLUMN/VISITS
diving	5394
products	2418
jobs	1447
my-account	1242
divesite	1213
checkout	1131
cart	1082

Figura 41 Tabela das páginas com visitas

### 3.6.12 Modos de visualização em cada widget

- Widget Web stats:
  - Gráfico linear
  - Mapa
  - Métricas.
- Widget E-commerce
  - Gráfico linear
  - Mapa
  - Métricas.
- Widget Pages Status
  - Gráfico circular
  - Tabela.

### 3.6.13 Traduções

A E-goi, enquanto empresa internacional com foco não apenas em Portugal, mas também em Espanha e na América Latina, reconhece a importância de fornecer a sua plataforma nas linguagens nativas desses mercados. Com o objetivo de proporcionar uma experiência de utilizador mais acessível e intuitiva, foi essencial incorporar traduções dos textos e *labels* presentes na plataforma.

Toda a plataforma E-goi oferece suporte para quatro idiomas: português de Portugal (pt-pt), português do Brasil (pt-br), inglês (ing) e espanhol (esp). A inclusão destes idiomas assegura que os utilizadores da E-goi em diferentes regiões possam interagir com a plataforma na sua língua nativa, melhorando assim a usabilidade e a satisfação do cliente.

Para a implementação das traduções, foi criado um sistema de internacionalização que permite a fácil adição e manutenção dos diferentes idiomas. Esta abordagem garante que todas as funcionalidades e componentes, incluindo os novos widgets e dashboards de estatísticas e e-commerce desenvolvidos, estejam disponíveis e sejam compreensíveis para todos os utilizadores, independentemente da sua localização geográfica.

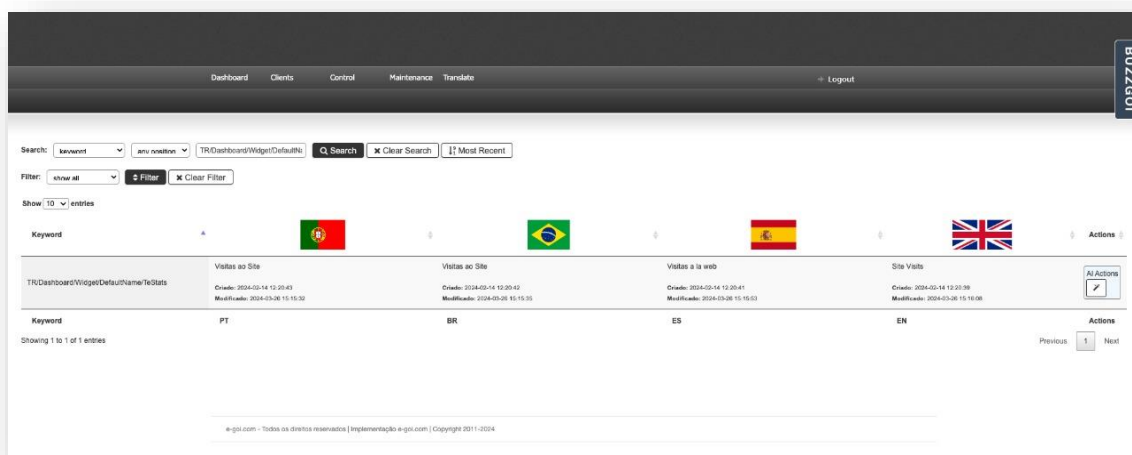


Figura 42 Interface de traduções

A plataforma *admin* da E-goi relativamente às traduções, ilustrada na imagem anterior, utiliza um sistema de tradução baseado em *keywords* para identificar os textos

a serem traduzidos. Cada texto a ser traduzido é identificado por uma *keyword*, como no exemplo 'TR/Dashboard/widget/Default/name/TeStats'. Estas *keywords* são introduzidas no código da plataforma, que ao serem processadas são substituídas pelas respectivas traduções que estão guardadas em uma base de dados, fazendo a relação com a linguagem configurada para o cliente. Desta forma, a plataforma consegue apresentar os textos na língua nativa do utilizador, garantindo uma experiência mais personalizada e eficiente.

### 3.7 Tomadas de decisão

Durante o desenvolvimento do projeto, enfrentei uma série de desafios que exigiram uma abordagem cuidadosa e decisões ponderadas para garantir o sucesso da implementação dos widgets.

#### 3.7.1 Que dados pretendemos mostrar?

Durante o processo de desenvolvimento, deparei-me com uma questão crucial relacionada à seleção dos dados a serem exibidos na dashboard. A decisão sobre quais informações priorizar e como organizá-las para proporcionar uma visão clara e concisa foi um dos primeiros desafios que enfrentei.

Para solucionar esta questão, reuni com o PM e analisamos a dashboard do matomo para tirarmos ideias já que os dados vinham de lá, no caso decidimos que seriam apresentados os dados de visitas, e-commerce gerais, visitas/vendas agrupados por país e visitas agrupadas por página.

#### 3.7.2 Por quais dados começar?

Após a decisão de quais dados usar decidimos a ordem de criação pela ordem de importância para o cliente, sendo a seguinte ordem dados de visitas, e-commerce e visitas por página

### 3.7.3 Quais métodos da API usar?

- `VisitsSummary.get`
- `UserCountry.getCountry`
- `Actions.getPageUrls`
- `Goals.getItemsName`

### 3.7.4 Como mostrar estes dados?

Estudo da plataforma matomo leva à escolha dos dados que queremos mostrar.

### 3.7.5 Como apresentar os dados por país?

Inicialmente, considerei destacar os países com o maior número de visitas, seguido pela apresentação de um total agregado para os restantes países. Essa abordagem visava permitir aos utilizadores obter rapidamente uma compreensão dos países mais relevantes em termos de tráfego, enquanto oferecia uma visão geral do desempenho global da plataforma.

No entanto, após uma análise mais aprofundada, decidi abandonar essa solução inicial. A razão para isso foi a perceção de que apresentar os dados dessa forma resultaria em uma exibição muito bruta e não ofereceria a experiência de utilizador desejada. Em vez disso, optei por explorar outras opções de visualização que permitissem uma apresentação mais intuitiva e dinâmica dos dados, proporcionando uma compreensão mais fácil e rápida das informações apresentadas.

### 3.7.6 Utilizar os componentes AmCharts para apresentar os dados?

Durante o processo de desenvolvimento, enfrentei a decisão de utilizar os componentes existentes na plataforma ou criar componentes para o projeto. Após avaliar os prós e contras de cada abordagem, optei por seguir a recomendação do LD da equipa e utilizar os componentes existentes.

A decisão de utilizar os componentes existentes foi motivada pela intenção de evitar sobrecarregar o projeto com novos desenvolvimentos e garantir a consistência e a

manutenção da estrutura existente. Ao optar por utilizar os componentes já existentes, pude aproveitar a vasta experiência da equipa na utilização desses elementos. Isso assegurou uma integração mais suave e eficiente, evitando potenciais problemas de compatibilidade ou desempenho que poderiam surgir ao desenvolver novos componentes. Além disso, essa escolha proporcionou a vantagem adicional de poder contar com a expertise da equipa para esclarecer dúvidas ou resolver eventuais desafios durante o processo de implementação.

### 3.7.7 Quantos widgets criar?

Inicialmente, planejei criar quatro widgets distintos: um para os dados gerais de visitas, outro para os dados gerais de e-commerce, um terceiro para os dados de visitas por página e um quarto para os dados de vendas e visitas por país que seriam apresentados em mapas. No entanto, durante a fase de planeamento, surgiu a preocupação de que a criação de 4 widgets adicionais poderia sobrecarregar a interface e confundir os utilizadores. Para evitar essa complexidade desnecessária, decidi otimizar a utilização dos widgets existentes. Desisti da ideia de criar um widget só para apresentar dados por países e adicionei aos widgets que continham dados de e-commerce e visitas o modo de visualização por mapa. Esta abordagem simplificada mantém a interface do utilizador mais limpa e coesa, enquanto fornece acesso fácil a todas as informações necessárias.

## 4 Testes de validação

No âmbito dos testes realizados para o projeto, foram conduzidos testes abrangentes tanto no backend quanto no frontend, de forma a garantir a qualidade e o correto funcionamento do sistema.

No backend, foram efetuados testes em todas as requisições HTTP para verificar a correta receção dos dados provenientes do Matomo. Estes testes foram essenciais para assegurar que os dados estavam a ser obtidos de forma adequada e sem erros. Identificadores: RF01, RF02, RF03, RF04.

Além disso, foram realizados testes nos dados recebidos pelos widgets após serem processados e transformados. Estes testes foram essenciais para garantir a precisão e integridade dos dados apresentados nos widgets, assegurando que estavam a ser exibidos de acordo com as configurações e requisitos definidos. Identificadores: RF05, RF06, RF07.

Ainda na parte do backend, foi realizado um teste de segurança relacionado à obtenção de dados da API do Matomo. Embora a API key do cliente seja inserida automaticamente pela plataforma da E-goí, foi necessário garantir que a API associada à conta do cliente não pudesse ser comprometida. Para isso, testei a inserção de uma API key diferente antes de realizar o pedido. O objetivo foi assegurar que os pedidos não pudessem ser manipulados, garantindo assim a integridade e a segurança dos dados. Este teste confirmou que apenas a API key correta associada à conta do cliente pode ser utilizada para obter os dados. Identificador: RF08.

Para a realização destes testes backend, foi utilizada a aplicação Postman. Esta ferramenta facilita consideravelmente os testes, pois permite fazer requisições HTTP de forma intuitiva e eficiente. Com o Postman, é possível configurar e enviar diversas solicitações, incluindo GET, POST, PUT e DELETE, assim como visualizar as respostas recebidas do servidor de maneira clara e organizada.

Já no frontend, foram conduzidos testes de acessibilidade para verificar se todas as funcionalidades estavam acessíveis e se mensagens de erro eram exibidas corretamente em situações onde o utilizador não inseria informações obrigatórias, como por exemplo o domínio do site que pretendia obter dados.



Em termos de acessibilidade foi testado a responsividade da aplicação com vários tamanhos de ecrãs.

Ainda no frontend, foram realizados testes para verificar se as alterações realizadas nas configurações dos widgets afetavam corretamente os dados recebidos e apresentados, de maneira a garantir que a consistência e precisão das informações exibidas na interface do utilizador.

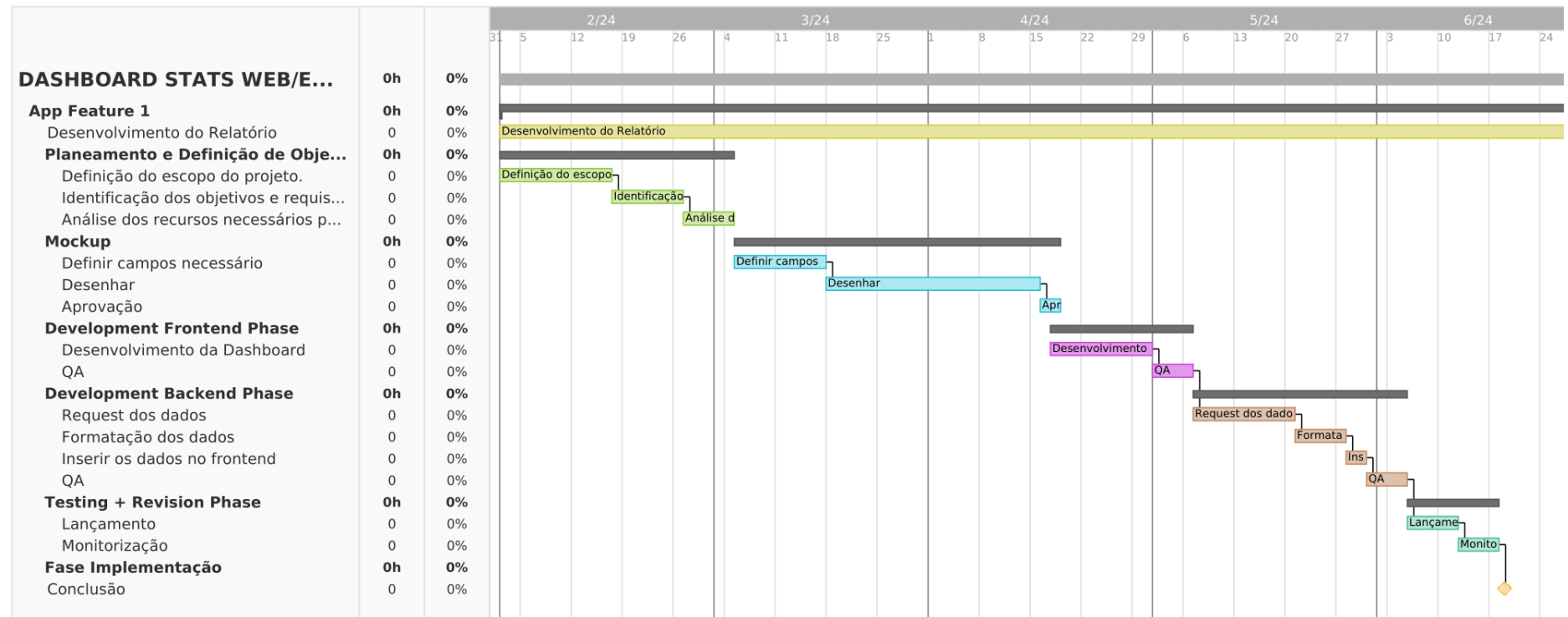
Identificadores: RF09, RF10, RF11.

## 5 Cronograma

### 5.1 Gant Previsão

O mapa de Gantt apresentado na seguinte página reflete as tarefas e os marcos que eu pensava que seriam necessários para a realização do projeto antes de iniciar o estágio. Este planejamento preliminar foi baseado nas informações e objetivos fornecidos inicialmente, e procurou abranger todas as etapas principais do projeto, desde a fase de planejamento até a entrega final.

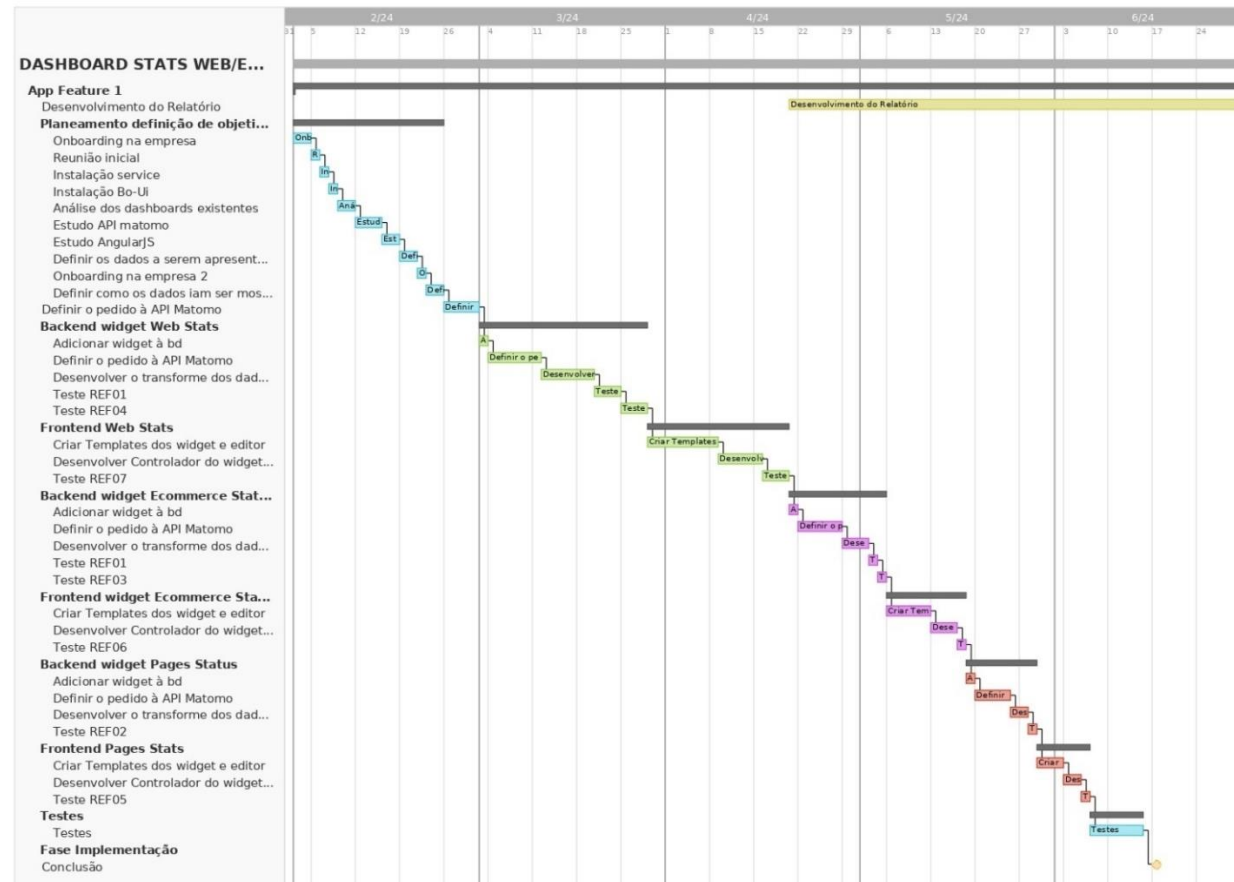
Figura 43 Mapa de Gantt previsto



## 5.2 Gantt do Projeto

O mapa de Gantt na seguinte página ilustra as tarefas e os marcos que realmente ocorreram durante o estágio. Diferente do planeamento inicial, este cronograma foi ajustado conforme as necessidades e desafios encontrados ao longo do desenvolvimento do projeto.

Figura 44 Mapa de Gantt do projeto



### 5.3 Comparação entre os Mapas de Gantt

Comparação entre o mapa de Gantt inicialmente previsto e o mapa de Gantt real, que reflete a execução efetiva das atividades ao longo do estágio.

#### 5.3.1 Diferenças Principais

##### **Início da Realização do Relatório**

No mapa de Gantt previsto, estava planeado que a realização do relatório começaria logo no início do estágio, paralelamente às outras atividades. No entanto, no mapa de Gantt real, o início da elaboração do relatório foi adiado para após o desenvolvimento do segundo widget. Este ajuste permitiu uma maior concentração no desenvolvimento técnico inicial, acumulando dados e resultados suficientes para serem documentados de forma mais substancial e organizada quando o relatório foi efetivamente iniciado.

##### **Fase de Preparação Prolongada**

A fase de preparação até ao desenvolvimento foi mais longa do que o previsto. Inicialmente, esperava-se que esta fase fosse relativamente breve, mas na prática, verificou-se a necessidade de uma preparação mais detalhada e abrangente. Este tempo adicional foi necessário para uma compreensão aprofundada dos requisitos do projeto, seleção de ferramentas e tecnologias apropriadas, e a realização de estudos preliminares que forneceram uma base sólida para o desenvolvimento subsequente.

##### **Tempo de Desenvolvimento dos Widgets**

Outra diferença significativa observada foi a redução progressiva do tempo de desenvolvimento de cada widget. No planeamento inicial, assumiu-se um tempo constante para o desenvolvimento de cada componente. No entanto, na prática, verificou-se que o tempo necessário para o desenvolvimento de cada widget foi diminuindo gradualmente. Esta redução deveu-se à curva de aprendizagem e à melhoria contínua dos processos de desenvolvimento à medida que o estágio progredia. Com a experiência adquirida no desenvolvimento do primeiro widget, os processos tornaram-se mais eficientes, resultando numa diminuição do tempo necessário para completar os widgets subsequentes.

### **Tempo Despendido em Testes**

Contrariamente ao planeado inicialmente, foi necessário despende mais tempo em testes. A complexidade dos componentes desenvolvidos e a necessidade de garantir a qualidade e funcionalidade do produto final exigiram uma fase de testes mais prolongada. Este tempo adicional foi crucial para identificar e corrigir bugs, bem como para realizar ajustes finos que melhoraram a performance e a usabilidade dos widgets.

#### **5.3.2 Comparação Detalhada**

Relatório:

- Previsão: Realização do relatório desde o início.
- Real: Início do relatório após o segundo widget.

Preparação:

- Previsão: Fase de preparação breve.
- Real: Fase de preparação mais longa e detalhada.

Desenvolvimento dos Widgets:

- Previsão: Tempos de desenvolvimento constantes.
- Real: Diminuição progressiva do tempo de desenvolvimento.

Testes:

- Previsão: Tempo moderado de testes.
- Real: Tempo de testes significativamente maior.

#### **5.3.3 Conclusão da comparação**

A comparação entre o mapa de Gantt previsto e o real destaca a importância da flexibilidade e adaptação no planeamento de projetos. A decisão de adiar a elaboração do relatório permitiu uma maior concentração no desenvolvimento técnico inicial, enquanto a diminuição progressiva do tempo de desenvolvimento dos widgets refletiu a eficiência crescente à medida que o estágio avançava. Além disso, o tempo adicional dedicado aos testes foi fundamental para assegurar a qualidade do produto final, e uma fase de preparação mais longa permitiu uma base sólida para o desenvolvimento do

projeto. Estes ajustes foram cruciais para a conclusão bem-sucedida do projeto, evidenciando a dinâmica e a evolução contínua das atividades ao longo do estágio.

## 6 Meios previstos e meios necessários

Ao iniciar este projeto, inicialmente não tinha uma percepção clara da complexidade técnica envolvida. A falta de informações detalhadas sobre os requisitos e a abrangência do projeto contribuiu para uma visão subestimada da quantidade de tecnologias e recursos necessários.

Prevendo uma abordagem mais simples e superficial, eu imaginava que as ferramentas e métodos utilizados seriam semelhantes aos que normalmente são empregues em contextos académicos. Esperava, por exemplo, que o trabalho consistisse principalmente na manipulação de uma única base de dados e no desenvolvimento de dashboards utilizando tecnologias convencionais como HTML, PHP e uma framework para apresentação dos dados.

No entanto, a realidade revelou-se bastante diferente, uma vez que o projeto exigiu uma gama mais ampla e sofisticada de ferramentas e técnicas. A complexidade do ambiente empresarial e as expectativas da empresa impulsionaram a necessidade de utilizar tecnologias mais avançadas e métodos de desenvolvimento mais robustos.

Assim, ao longo do processo, foi necessário adaptar e expandir os meios previstos inicialmente, incorporando uma variedade de tecnologias adicionais.

### 6.1 Meios humanos

Os meios humanos foram fundamentais para o desenvolvimento deste projeto, especialmente durante a fase de desenvolvimento. O principal apoio foi fornecido pelo LD da equipa de integrações, que me orientou constantemente ao longo do processo. Além disso, contei com a contribuição do TL, o Miguel Azevedo, que analisou e definiu comigo os dados a serem apresentados no dashboard, também recebi suporte do UX, João Vale.

Fora da minha equipa, recebi assistência do Bruno Martins, da equipa de SRE, que ajudou na instalação dos projetos internos (service e bo-ui). Além disso, o João Silva, que implementou os componentes do AmCharts na plataforma E-goí, auxiliou-me na implementação desses componentes (mapas e gráficos) nos meus widgets.

### 6.2 Material fornecido pela empresa

No primeiro dia de estágio a empresa forneceu os seguintes materiais:

- Macbook
- Ecrã
- Rato
- Teclado
- Cabos

## 6.3 Tecnologias utilizadas

### 6.3.1 Linguagens

- PHP: Utilizado em todo o backend do widget, desde a realização do pedido à API do Matomo até a transformação dos dados recebidos para serem apresentados no frontend.
- AngularJS: Um *framework* JavaScript utilizado para construir o frontend da aplicação.
- Javascript: A linguagem JavaScript foi amplamente utilizada no desenvolvimento do projeto devido à sua versatilidade e capacidade de criar funcionalidades interativas e dinâmicas no lado do cliente.
- HTML5: A linguagem de marcação utilizada para estruturar o conteúdo dos widgets.

### 6.3.2 Aplicações

- Visual Studio Code foi o editor de texto/código para este projeto devido a várias razões, é uma ferramenta altamente popular e amplamente utilizada na indústria de desenvolvimento de software, o que significa que oferece suporte a uma ampla gama de linguagens de programação, incluindo aquelas usadas neste projeto. Além disso, o Visual Studio Code é conhecido pela sua interface amigável e intuitiva
- Postman é uma aplicação que permite aos desenvolvedores enviar solicitações HTTP para APIs e visualizar as respostas, facilitando o teste e a *debug* durante o desenvolvimento do projeto, foi uma ferramenta essencial neste projeto, utilizada para efetuar pedidos ao projeto service a fim de testar a recolha dos dados necessários para os widgets. Foi fundamental nos testes dos dados retornados, para assim garantir a integridade e precisão das informações apresentadas nos widgets. Além disso, a aplicação oferece funcionalidades avançadas como a criação de coleções de requisições, a automação de testes e a



geração de documentação, o que otimiza o processo de desenvolvimento e garante maior confiabilidade e consistência nos testes realizados.

### 6.3.3 Outras Tecnologias

- Matomo API: Uma API RESTful que permite extrair dados estatísticos e analíticos do Matomo para serem utilizados na aplicação.
- AmCharts: Desde gráficos simples até mapas complexos, o AmCharts oferece uma variedade de recursos e configurações que permitem personalizar completamente a aparência e o comportamento dos elementos visuais. Com suporte para uma variedade de gráficos e integração simples em projetos web, o AmCharts é uma escolha popular para aqueles que procuram uma solução flexível para visualização de dados.
- JSON (*JavaScript Object Notation*): É um formato de dados leve e fácil de ler, frequentemente usado para transferência de dados entre um servidor e um utilizador, bem como para armazenar e enviar dados. Ele é baseado na sintaxe de objetos JavaScript e é amplamente utilizado devido à sua simplicidade e flexibilidade. Este é o formato de dados utilizado, uma vez que é retornado pela API do Matomo.
- GitLab: É a plataforma escolhida pela E-goí para armazenar o código-fonte de todas as aplicações, por oferecer recursos que facilitam a gestão eficiente do desenvolvimento das aplicações.

## 7 Conclusões

Ao longo deste projeto, pude vivenciar uma jornada de aprendizagem enriquecedora, tanto em termos de conhecimento técnico quanto de habilidades interpessoais. A utilidade do projeto enquanto aprendizado foi inegável, pois permitiu-me aplicar e aprofundar os meus conhecimentos em desenvolvimento de software.

O estágio também desempenhou um papel crucial no meu aprendizado sobre o funcionamento interno de uma empresa e o trabalho em equipa. A colaboração com colegas de equipa e a interação com diferentes departamentos proporcionaram uma compreensão mais ampla dos processos empresariais e da importância da cooperação para alcançar objetivos comuns.

No que diz respeito aos objetivos definidos no início do projeto, posso afirmar que foram alcançados. Desde a conceção da ideia até a entrega final da dashboard, consegui cumprir os prazos estabelecidos e alcançar os principais marcos ao longo do desenvolvimento. Destaco especialmente a implementação de funcionalidades-chave, como a visualização de dados de visitas e vendas por país e páginas, que atendem às necessidades dos utilizadores e agregam valor à empresa.

A contribuição do projeto para os objetivos da empresa foi significativa. As novas possibilidades para a dashboard tornam a plataforma mais eficiente e intuitiva para análise de dados estatísticos, permite que os utilizadores acedam informações relevantes de forma rápida e precisa. Isso certamente aumentará a produtividade e a tomada de decisões informadas dentro da empresa.

Durante o desenvolvimento do projeto, enfrentamos diversos desafios, desde questões técnicas até obstáculos organizacionais. No entanto, aprendi que enfrentar esses desafios com determinação e trabalho em equipa leva a soluções criativas e eficazes. A gestão eficiente do tempo foi fundamental para superar esses desafios e garantir o progresso contínuo do projeto em todas as suas fases.

Como recomendações para o futuro, sugiro a continuidade do aprimoramento da interface, incluindo a integração de informações mais específicas e a otimização do desempenho para garantir uma experiência ainda melhor para os utilizadores.

## 7.1 Análise de resultados

Após a conclusão do projeto e a implementação da dashboard, foi realizada uma análise detalhada ao resultado. Os resultados superaram as expectativas, uma vez que a dashboard apresentou um aspeto altamente intuitivo e dinâmico, indo além da simples exibição de dados brutos. Um dos pontos destacados foi o feedback positivo recebido da empresa, que elogiou especialmente a utilização dos gráficos, como o mapa, que excedeu as expectativas. Esses elementos não apenas complementaram os dados apresentados, mas também enriqueceram a experiência do utilizador, possibilitando uma compreensão mais profunda e rápida das informações. Foi possível observar melhorias significativas em termos de usabilidade e visualização dos dados. A dashboard proporcionou uma experiência de análise mais eficaz, que permite aos utilizadores compreender rapidamente as informações apresentadas. Isso demonstra que a escolha das ferramentas e técnicas adequadas foi fundamental para o sucesso do projeto.

Em suma, os resultados obtidos com a concretização deste projeto refletem não apenas a realização dos objetivos estabelecidos, mas também uma melhoria significativa na capacidade de análise de dados por parte dos utilizadores. O feedback positivo da empresa reforça ainda mais o sucesso da implementação da dashboard.

## 8 Referências bibliográficas

### 8.1 Bibliografia Consultada

- [1] Amcharts. (n.d.). *Amcharts Demos*. Recuperado de <https://www.amcharts.com/demos/>
- [2] Kumar, P., & Singh, A. (2021). A Comprehensive Study on Web Development Technologies. *Vidhyayana Journal*, 6(3), 89-101. Recuperado de <https://vidhyayanaejournal.org/journal/article/view/353>
- [3] Garlan, D., & Shaw, M. (2024). A Framework for Evaluating Software Architecture. *Preprints*, 202404.1860. Recuperado de <https://www.preprints.org/manuscript/202404.1860/v1>
- [4] API Matomo. (n.d.). *Matomo Reporting API*. Recuperado de <https://developer.matomo.org/api-reference/reporting-api>
- [5] Codecademy Team. (n.d.). HTTP requests. Codecademy. Recuperado em 11 de março de 2024, de <https://www.codecademy.com/article/http-requests>
- [6] Bessereau, F. (2015). *Designing a REST API*. Recuperado de [https://blog.bessereau.eu/assets/pdfs/api\\_rest.pdf](https://blog.bessereau.eu/assets/pdfs/api_rest.pdf)
- [7] Github. (n.d.). *Learn Git*. Recuperado de [https://docs.gitlab.com/ee/tutorials/learn\\_git.html](https://docs.gitlab.com/ee/tutorials/learn_git.html)
- [8] Guedes, M. (n.d.). O que é MVC? Blog da TreinaWeb. Recuperado em 1 de março de 2024, de <https://www.treinaweb.com.br/blog/o-que-e-mvc>
- [9] Manual PHP. (n.d.). *PHP Manual*. Recuperado de <https://www.php.net/manual/en/index.php>
- [10] Nierstrasz, O., Ducasse, S., & Gîrba, T. (2009). *Model-View-Controller by Example*. Recuperado de <https://rmod-files.lille.inria.fr/FreeBooks/ByExample/25%20-%20Chapter%2023%20-%20Model-View-Controller.pdf>
- [11] Net Ninja. (2016, fevereiro 24). *AngularJS Tutorials* [Playlist de vídeos]. Recuperado de [https://www.youtube.com/watch?v=FIUCU13dJyo&list=PL4cUxeGkcC9gsJS5QgFT2IvWIX78dV3\\_v](https://www.youtube.com/watch?v=FIUCU13dJyo&list=PL4cUxeGkcC9gsJS5QgFT2IvWIX78dV3_v)
- [12] Redis. (n.d.). *Redis for Dummies*. Recuperado de <https://redis.io/redis-for-dummies/>

## 8.2 Bibliografia Citada

Amazon Web Services. (n.d.). *What is API?*. Recuperado de <https://aws.amazon.com/pt/what-is/api/>. Citado na p. 18

Bessereau, F. (2015). *Designing a REST API*. Recuperado de [https://blog.bessereau.eu/assets/pdfs/api\\_rest.pdf](https://blog.bessereau.eu/assets/pdfs/api_rest.pdf) Citado na p. 19

Buschmann, F., Henney, K., & Schmidt, D. C. (1996). *Model-View-Controller (MVC)*. In *Pattern-Oriented Software Architecture Volume 1: A System of Patterns* (pp. 125-143). Recuperado de <http://www.laputan.org/pub/papers/POSA-MVC.pdf> Citado na p. 19

Google Marketing Platform. (n.d.). *Google Analytics*. Recuperado de <https://marketingplatform.google.com/about/analytics/>. Citado na p. 13

## 9 Glossário

**AmCharts:** Biblioteca de gráficos em JavaScript usada para criar gráficos interativos e visualizações de dados. Foi utilizada para apresentar os dados nos widgets.

**API (Interface de Programação de Aplicações):** Interface que permite a comunicação entre diferentes sistemas ou componentes de software. No contexto deste projeto, a API usada foi a do Matomo para obter dados.

**Backend:** Parte do sistema responsável pelo processamento e gestão dos dados. No projeto, inclui a comunicação com a API do Matomo e a transformação dos dados para serem utilizados no frontend.

**Base de Dados:** Sistema organizado para armazenar, gerir informações. Neste projeto, é utilizada para armazenar as configurações padrão e as opções de visualização dos widgets e os seus dados em cache.

**Dashboard:** Interface gráfica que apresenta os dados analíticos de forma organizada e visualmente acessível. Neste projeto, os dashboards são compostos por vários widgets.

**Framework:** Conjunto de ferramentas e bibliotecas que facilitam o desenvolvimento de software.

**Frontend:** Parte do sistema que interage diretamente com o utilizador.

**JSON (JavaScript Object Notation):** Formato de troca de dados leve e fácil de ler. Utilizado para transmitir dados entre o backend e o frontend.

**Keywords:** Identificadores utilizados no código para associar elementos da interface do utilizador com as traduções armazenadas na base de dados.

**Matomo:** Plataforma de análise *web open-source* utilizada para rastrear e reportar estatísticas de visitas.

**MVC (Model-View-Controller):** Arquitetura de software que separa a aplicação em três componentes principais: Model (gestão de dados), View (interface do utilizador) e Controller (lógica de negócio). Estrutura adotada para organizar o projeto.

**Pedidos HTTP:** Pedidos feitos através do protocolo HTTP para obter ou enviar dados entre cliente e servidor. Funções como GET, PUT, DELETE são usadas para interagir com a API do Matomo.

**REST (*Representational State Transfer*):** Arquitetura de software para sistemas distribuídos. Base das APIs utilizadas no projeto.

**RESTful:** Refere-se aos serviços web que seguem os princípios do REST, permitindo a comunicação e manipulação de dados através de requisições HTTP.

**Teste de Acessibilidade:** Verificação para garantir que a aplicação é acessível a todos os utilizadores. No projeto, testes foram realizados para garantir que mensagens de erro fossem apresentadas corretamente e que as opções de configuração dos widgets fossem funcionais.

**Teste de *Backend*:** Testes realizados para garantir que os dados obtidos da API do Matomo são corretamente recebidos e transformados. Incluem a verificação das requisições HTTP e a validação dos dados antes de serem enviados ao frontend.

**Teste de *Frontend*:** Testes realizados na interface do utilizador para garantir que as funcionalidades dos widgets estão operacionais, as configurações são corretamente guardadas e que as visualizações dos dados são precisas.

**UX (Experiência do Utilizador):** Experiência do utilizador ao interagir com o sistema. Envolve a facilidade de uso, eficiência e satisfação do utilizador.

**Widget:** Componente da interface que exibe dados específicos, como estatísticas de visitas ou vendas. Os widgets podem ser configurados pelo utilizador para apresentar os dados em diferentes formatos, como gráficos ou mapas.

## 10 Anexos

### 10.1 Tabelas de descrição detalhada de casos de uso

Tabela 7 RF01 Caso de uso 'Editar dashboard'

DESCRIÇÃO DETALHADA DE CASOS DE USO	
<b>Identificador:</b>	RF01
<b>Descrição:</b>	Este Use Case descreve o processo pelo qual um utilizador edita a dashboard, desde adicionar a retirar widgets
<b>Atores:</b> Utilizador: O utilizador da plataforma. E-goi: Plataforma E-goi	
<b>Fluxo de Eventos:</b> O utilizador entra na aba de dashboards. O utilizador carrega no botão de adicionar widget. O utilizador seleciona o widget pretendido. O utilizador configura o widget.	
<b>Pré-condições:</b> O utilizador deve estar logado na plataforma E-goi.	
<b>Pós-condições:</b> O widget é adicionado com sucesso.	
<b>Exceções:</b> Se a plataforma estiver indisponível, o utilizador não poderá aceder.	

Tabela 8 RF02 Caso de uso 'Analisar Web Stats'

DESCRIÇÃO DETALHADA DE CASOS DE USO	
<b>Identificador:</b>	RF02
<b>Descrição:</b>	Este Use Case descreve o processo onde analisa os dados pelo widget de Web Stats.
<b>Atores:</b> Utilizador: O utilizador da plataforma. E-goi: Plataforma E-goi. Matomo: Plataforma onde ficam guardados os dados.	
<b>Fluxo de Eventos:</b> O utilizador após ter o widgets adicionado à sua dashboard e configurado esse widget vai ser carregado com dados vindos do matomo. Análise dos dados pretendidos.	
<b>Pré-condições:</b> O utilizador deve estar logado na plataforma E-goi. O utilizador deve ter adicionado e configurado o widget.	
<b>Pós-condições</b> Inputs valiosos para análise de visitas dos sites do cliente.	
<b>Exceções:</b> Se a plataforma estiver indisponível, o utilizador não poderá aceder. O widget deve estar bem configurado.	



Tabela 9 RF03 Caso de uso 'Analisar E-commerce'

DESCRIÇÃO DETALHADA DE CASOS DE USO	
<b>Identificador:</b>	RF03
<b>Descrição:</b>	Este Use Case descreve o processo onde analisa os dados pelo widget de E-commerce.
<b>Atores:</b> Utilizador: O utilizador da plataforma. E-goi: Plataforma E-goi. Matomo: Plataforma onde ficam guardados os dados.	
<b>Fluxo de Eventos:</b> O utilizador após ter o widgets adicionado à sua dashboard e configurado esse widget vai ser carregado com dados vindos do matomo. Análise dos dados pretendidos.	
<b>Pré-condições:</b> O utilizador deve estar logado na plataforma E-goi. O utilizador deve ter adicionado e configurado o widget.	
<b>Pós-condições</b> Inputs valiosos para análise de vendas e carrinhos abandonado dos sites do cliente.	
<b>Exceções:</b> Se a plataforma estiver indisponível, o utilizador não poderá aceder. O widget deve estar bem configurado.	

Tabela 10 RF04 Caso de uso 'Editar modos de visualização'

DESCRIÇÃO DETALHADA DE CASOS DE USO	
<b>Identificador:</b>	RF04
<b>Descrição:</b>	Este Use Case descreve o processo onde são editados os modos de visualização nos widgets
<b>Atores:</b> Utilizador: O utilizador da plataforma. E-goi: Plataforma E-goi.	
<b>Fluxo de Eventos:</b> Para iniciar a edição do widget o utilizador deve carregar no botão de editar do widget. No editar do widget ir até à secção de Resultados. Selecionar os modos em que pretende ver os dados. Submeter a edição	
<b>Pré-condições:</b> O utilizador deve estar logado na plataforma E-goi. O utilizador deve ter adicionado e configurado o widget.	
<b>Pós-condições</b> N/a.	
<b>Exceções:</b> Se a plataforma estiver indisponível, o utilizador não poderá aceder. O widget deve estar bem configurado.	

Tabela 11 RF05 Caso de uso 'Editar métricas apresentadas'

DESCRIÇÃO DETALHADA DE CASOS DE USO	
<b>Identificador:</b>	RF05
<b>Descrição:</b>	Este Use Case descreve o processo onde são editadas as métricas apresentadas nos widgets
<b>Atores:</b> Utilizador: O utilizador da plataforma. E-goi: Plataforma E-goi.	
<b>Fluxo de Eventos:</b> Para iniciar a edição do widget o utilizador deve carregar no botão de editar do widget. No editar do widget ir até à secção de Métricas. Selecionar as métricas em que pretende analisar. Submeter a edição.	
<b>Pré-condições:</b> O utilizador deve estar logado na plataforma E-goi. O utilizador deve ter adicionado e configurado o widget.	
<b>Pós-condições</b> N/a	
<b>Exceções:</b> Se a plataforma estiver indisponível, o utilizador não poderá aceder. O widget deve estar bem configurado.	

## 10.2 Tabelas de testes

### 10.2.1 Testes aos pedidos ao Matomo

Tabela 12 Teste RF01

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF01
<b>Objetivo:</b>	Verificar se o matomo retorna os dados necessários no método UserCountry.getCountry
<b>Especificação de Entradas</b> Http Request com os parâmetros necessários	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente aos países	
<b>Outros</b> NA	
<b>Dependências</b> NA	

Tabela 13 Teste RF02

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF02
<b>Objetivo:</b>	Verificar se o matomo retorna os dados necessários no método Actions.getPageUrls
<b>Especificação de Entradas</b> Http Request com os parâmetros necessários	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente aos países	
<b>Outros</b> NA	
<b>Dependências</b> NA	

Tabela 14 Teste RF03

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF03
<b>Objetivo:</b>	Verificar se o matomo retorna os dados necessários no método Goals.getItemsName
<b>Especificação de Entradas</b> Http Request com os parâmetros necessários	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente aos países	
<b>Outros</b> Deve ser feito para com o parâmetro de 'Abandoned_carts' a 0 e 1	
<b>Dependências</b> NA	

Tabela 15 Teste RF04

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF04
<b>Objetivo:</b>	Verificar se o matomo retorna os dados necessários no método <code>VisitsSummary.get</code>
<b>Especificação de Entradas</b> Http Request com os parâmetros necessários	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente aos países	
<b>Outros</b> NA	
<b>Dependências</b> NA	

## 10.2.2 Testes aos pedidos dos dados do frontend ao service

Tabela 16 Teste RF05

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF05
<b>Objetivo:</b>	Verificar os dados recebidos do service para o widgets Pages Status
<b>Especificação de Entradas</b> Get widgetize ao service com o id relativamente ao widgets Page Status	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente aos países por dia	
<b>Outros</b> NA	
<b>Dependências</b> Teste RF02	

Tabela 17 Teste RF06

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF06
<b>Objetivo:</b>	Verificar os dados recebidos do service para o widgets Ecommerce Status
<b>Especificação de Entradas</b> Get widgetize ao service com o id relativamente ao widgets Ecommerce Status	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente a vendas	
<b>Outros</b> NA	
<b>Dependências</b> Testes RF03 e RF01	

Tabela 18 Teste RF07

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF07
<b>Objetivo:</b>	Verificar os dados recebidos do service para o widgets de Web Status
<b>Especificação de Entradas</b> Get widgetize ao service com o id relativamente ao widgets Web Status	
<b>Especificação de Saídas</b> a. JSON com todos os dados relativamente a visitas ao site por dia	
<b>Outros</b> NA	
<b>Dependências</b> RF04	

### 10.2.3 Teste de segurança

Tabela 19 Teste RF08

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF08
<b>Objetivo:</b>	Verificar se o sistema consegue autenticar na API do Matomo corretamente.
<b>Especificação de Entradas</b> - Token de API válido e inválido.	
<b>Especificação de Saídas</b> - Sucesso na autenticação com token válido. - Falha na autenticação com token inválido.	
<b>Outros</b> - Verificar mensagens de erro apropriadas em caso de falha.	
<b>Dependências</b> Acesso à API do Matomo.	

### 10.2.4 Testes de acessibilidade

Tabela 20 Teste RF09

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF09
<b>Objetivo:</b>	Garantir que os widgets se atualizam dinamicamente com novos dados.
<b>Especificação de Entradas</b> - Dados atualizados.	
<b>Especificação de Saídas</b> - Widgets atualizados em tempo real. - Nenhum erro ou atraso durante a atualização.	
<b>Outros</b> Verificar a atualização sem atualizar a página.	
<b>Dependências</b> Todos os testes anteriores	

Tabela 21 Teste RF10

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF10
<b>Objetivo:</b>	Garantir que os widgets são responsivos e funcionam corretamente em diferentes dispositivos.
<b>Especificação de Entradas</b> Várias resoluções de ecrã (desktop, tablet, telemóvel).	
<b>Especificação de Saídas</b> - Widgets exibidos corretamente em todas as resoluções. - Nenhum layout quebrado ou sobreposição de elementos.	
<b>Outros</b> Verificar usabilidade e experiência do utilizador.	
<b>Dependências</b> Todos os testes anteriores	

Tabela 22 Teste RF11

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF11
<b>Objetivo:</b>	Garantir que são apresentadas mensagens de erro quando não é inserido o domínio.
<b>Especificação de Entradas</b> Não inserir domínio.	
<b>Especificação de Saídas</b> - Mensagem de erro.	
<b>Outros</b> NA	
<b>Dependências</b> NA	

### 10.2.5 Teste Geral

Tabela 23 Teste RF12

ESPECIFICAÇÃO DE CASO DE TESTE	
<b>Identificador:</b>	RF12
<b>Objetivo:</b>	Verificar se edições de todos os widgets estão funcionais
<b>Especificação de Entradas</b> Alterar todas as opções do editor de todos os widgets (domínio, período, vistas e métricas)	
<b>Especificação de Saídas</b> a. Obter os dados pretendidos	
<b>Outros</b> NA	
<b>Dependências</b> Todos os testes anteriores	